BEVEZETÉS A SZÁMÍTÁSELMÉLETBE 2

Írta: Szeszlér Dávid

Lektorálta: Wiener Gábor

Készült a

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudományi és Információelméleti Tanszék

gondozásában.

ISBN 978-963-421-884-5

Utolsó módosítás: 2025. május 14.

Tartalomjegyzék

1. fejezet

Gráfelméleti alapfogalmak

A hálózat fogalma a 21. század tudományának egyik meghatározó kulcsfogalmává vált, ami nélkül számos, egymástól távol eső terület kutatása ma már elképzelhetetlen volna; ilyen a számítástudomány mellett például a molekuláris biológia, a fizika, a szociológia, a nyelvészet és a közgazdaságtan. Sőt, a hálózatelmélet mára önálló tudományággá fejlődött.

Mit is jelent az, hogy hálózat? A kifejezés jelentése nem teljesen egységes a különböző alkalmazásokban, de az mindegyik esetben közös, hogy a hálózat valamilyen fajta elemekből és a közöttük lévő kapcsolatokból áll. Például egy számítógéphálózat esetében az elemek – vagy más néven: csomópontok – különböző számítógépek, amelyek közül bizonyos párok összeköttetésben állnak (például kábelen keresztül). Egy úthálózat csomópontjai az útkereszteződések, amelyek közül bizonyosakat közvetlen (vagyis további kereszteződést nem érintő) útszakaszok kötnek össze. Hálózatok jönnek létre a különböző közösségi oldalakon is, aminek a csomópontjai a felhasználók, akik egymást ismerősnek jelölve kapcsolatba kerülhetnek.

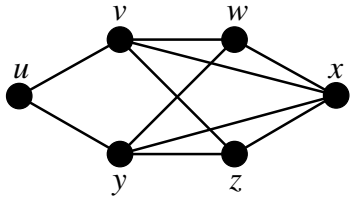
A *gráf* és a *hálózat* kifejezések nagyjából azonos jelentéssel bírnak, de a matematikán belül az előbbi honosodott meg; gráfelméletnek pedig a matematika ezekkel foglalkozó ágát nevezik. Bár az első, ide tartozó eredmények a 18. századból származnak, a gráfelmélet a 20. század első felében indult valódi fejlődésnek. A világ első, gráfelméletről szóló tankönyve 1936-ban jelent meg és Kőnig Dénes, magyar matematikus írta német nyelven. Mára ez a terület hatalmasra terebélyesedett és rengeteg kisebb ágra oszlott; az idevágó alapismeretek pedig minden informatikus számára is nélkülözhetetlenek.

1.1. A gráf fogalma

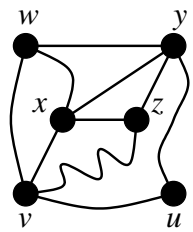
Egy gráf csomópontokból és ezek közötti összeköttetésekből áll; az előbbieket a gráfelméletben meghonosodott elnevezések szerint *csúcsoknak* vagy *pontoknak*, az utóbbiakat *éleknek* nevezzük. (A csúcs és él nevek a térgeometriából származnak: a poliéderek, vagyis síklapok által határolt testek csúcsainak és éleinek hálózata

is gráfot alkot.) Egy G gráf csúcsainak halmazát $V(G)$, az éleinek halmazát $E(G)$ jelöli (ahol a V és E jelölések a csúcs, illetve él szavak angol megfelelőinek, a *vertex*, illetve *edge* szavaknak a kezdőbetűiből származnak); a $G = (V,E)$ jelölés pedig azt fejezi ki, hogy a G gráf csúcshalmaza $V(G) = V$ és élhalmaza $E(G) = E$.

A gráfokat gyakran úgy ábrázoljuk, ahogyan az az 1.1. ábrán is látható: a csúcsokat pöttyök, az éleket köztük vezető vonalak reprezentálják. Az viszont érdektelen, hogy a csúcsoknak megfelelő pöttyöket hogyan helyezzük el a síkon és az éleknek megfelelő vonalakat hogyan vezetjük közöttük. Így például az 1.1a és az 1.1b ábrák ugyanazt a gráfot ábrázolják: bár a két rajz ránézésre más, de a két esetben azonos a csúcshalmaz és ugyanazok a csúcspárok vannak összekötve. Az sem okoz problémát, ha különböző éleknek megfelelő vonalak keresztezik egymást csúcsokon kívül, mint például az 1.1a ábrán (mások mellett) a v-t x-szel és a w-t y-nal összekötő élek esetében; csak arra kell vigyázni, hogy ezek a vonalak ne menjenek át csúcsokat ábrázoló pöttyökön (leszámítva persze azt a kettőt, amit összekötnek).



1.1a ábra



1.1b ábra

Bár az 1.1. ábrán láthatóhoz hasonló rajzok alkalmasak gráfok ábrázolására, de fontos tisztázni, hogy a gráf fogalma alapvetően nem geometriai természetű: ezek az ábrák csak kényelmesen szemléltethetővé teszik számunkra a gráfokat, de nem azonosak azokkal. Ráadásul a gyakorlati alkalmazásokban előkerülő gráfok egy része olyan rengeteg csúcsot és élt tartalmaz, hogy a lerajzolásuk reménytelen volna. (Például ebben a jegyzetben is előkerül majd olyan probléma, aminek a megoldásához egy többszáz millió csúcsú gráfot fogunk konstruálni; természetesen ezt sem a rajzával adjuk majd meg.)

Azt a tényt, hogy a G gráf egy e éle által összekötött két csúcs egyike v, szokás úgy is kifejezni, hogy v az e egyik *végpontja*. Ugyanezt fejezzük ki azzal is, ha azt mondjuk, hogy az e él *illeszkedik* a v csúcsra. Ha pedig egy gráf két csúcsát él köti össze, akkor ezeket a csúcsokat *szomszédosnak* nevezzük.

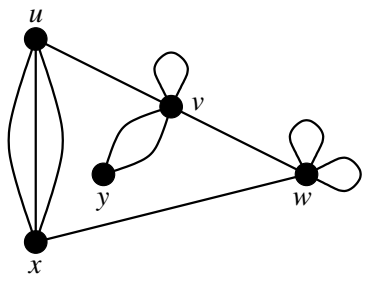
Bár a fentiek talán elég pontosan körülírják a gráf fogalmát, mégis szeretnénk megfogalmazni egy, a matematikai precizitás igényeinek megfelelő, ennél pontosabb definíciót is. Ennek az alapgondolata nagyon egyszerű: egy gráf megadásához ismernünk kell a csúcsait és ezen kívül tudnunk kell, hogy mely csúcspárok szomszédosak; az éleket tehát a legegyszerűbb a csúcsokból alkotott pároknak, vagyis két elemű részhalmazoknak tekinteni.

1.1. Definíció. *Legyen $V \neq \emptyset$ tetszőleges, véges halmaz. Legyen továbbá E egy olyan halmaz, aminek minden eleme a V egy két elemű részhalmaza. Ekkor a $G = (V,E)$ párt* egyszerű gráfnak *nevezzük, aminek a* csúcshalmaza $V(G) = V$ *és az* élhalmaza $E(G) = E$.

Például az 1.1. ábrán látható gráf csúcshalmaza $V = \{u,v,w,x,y,z\}$, az E élhalmaza pedig egy tíz elemű halmaz, aminek az elemei maguk is halmazok, mégpedig V bizonyos, két elemű részhalmazai: $E = \big\{\{u,v\},\{u,y\},\{v,w\},\ldots,\{y,z\}\big\}$.

A definícióban kikötöttük, hogy V véges halmaz; bár a gráfelméletnek van egy olyan ága, ami végtelen gráfokkal foglalkozik (amiknek tehát végtelen sok csúcsuk és élük is lehet), de ilyenek ebben a jegyzetben nem fordulnak elő, $V(G)$ és $E(G)$ végességét mindig feltételezni fogjuk. Az 1.1. Definíció kizárja, hogy egy gráfnak nulla darab csúcsa legyen, de azt nem, hogy (tetszőleges csúcsszám mellett) nulla darab éle legyen; az ilyen gráfokat *üres gráfnak* szokták nevezni.

Az 1.1. Definícióban bevezetett fogalmat nem véletlenül neveztük *egyszerű* gráfnak: van ugyanis két olyan jelenség, amiket ez a definíció nem enged meg, de a gráf fogalmába mégis bele szoktak érteni, mert bizonyos alkalmazásokban hasznosak lehetnek. Az első ezek közül az, hogy egy gráfban megengedett, hogy két csúcs között több különböző él is fusson; az ilyeneket *párhuzamos éleknek* vagy *többszörös éleknek* szokás nevezni. Például az 1.2. ábra gráfjában a v és y csúcsok között futó két él párhuzamos, u és x között pedig három párhuzamos él is fut. A másik jelenségre is látunk példát az 1.2. ábra gráfján: előfordulnak olyan, *hurokélnek* nevezett élek, amik egy csúcsot saját magával kötnek össze (a w-re például két ilyen is illeszkedik). A hurokéleknek tehát csak egy végpontja van.



1.2. ábra

A fentiek szerint tehát az egyszerű gráfok azok a gráfok, amik nem tartalmaznak párhuzamos élt és hurokélt. Ezek valóban „egyszerűek” abban az értelemben, hogy bármely két, különböző csúcsot tekintve csak két eset lehetséges: vagy szomszédosak, vagy nem. A gráfelmélet sok alkalmazásában csak egyszerű gráfokról esik szó: például egy közösségi oldal felhasználói közötti kapcsolatokat reprezentáló gráf nyilván egyszerű, hiszen egyetlen felhasználó sincs összekötve saját magával (vagyis a gráfban nincs hurokél) és két különböző felhasználó között legföljebb egy él futhat (vagyis nincsenek párhuzamos élek sem). Ha azonban egy gráf például egy

térképet reprezentál, akkor ebben könnyen előfordulhatnak hurokélek és párhuzamos élek is: lehetséges, hogy egy útszakasz két végpontja ugyanabban a kereszteződésben legyen és az is, hogy két kereszteződés között több útszakasz fusson.

Természetesen lehetséges volna a (nem feltétlen egyszerű) gráfok általános fogalmára is az 1.1-hez hasonló, precíz definíciót alkotni, de ettől ebben a jegyzetben eltekintünk, mert elég körülményes volna. (A hurokél fogalmát még könnyű volna kezelni, ha $E(G)$-ben egy elemű részhalmazokat is megengednénk; a valódi kényelmetlenséget a párhuzamos élek okozzák, mert a halmaz fogalma kizárja, hogy egy elem több példányban is szerepeljen egy halmazban.)

A nem (feltétlen) egyszerű gráfok körében is alkalmazni fogjuk az $e = \{u, v\}$ jelölést annak a kifejezésére, hogy az $e \in E(G)$ él végpontjai az $u \in V(G)$ és $v \in V(G)$ csúcsok. Így teszünk annak ellenére is, hogy ez két okból sem teljesen precíz: egyrészt, ha e_1 és e_2 párhuzamos élek az u és v csúcsok között, akkor e_1-re és e_2-re is alkalmazhatjuk az $e_1 = \{u, v\}$, illetve az $e_2 = \{u, v\}$ jelölést, ami azt a téves állítást sugallja, hogy $e_1 = e_2$; másrészt, ha e a v csúcsra illeszkedő hurokél, akkor az $e = \{v, v\}$ jelölés szokatlan, a halmaz alapfogalmától idegen.

1.2. Feladat. Egy patinás londoni sakk klubban történt bűntény felderítése közben Sherlock Holmes megkérte Dr. Watsont, hogy kérdezze meg a bűntény estéjén a klubban tartózkodó vendégektől, hogy hány partit játszottak aznap este. Watson végigjárta a tíz érintett klubtagot és a válaszokat megmutatta Holmesnak: 4, 2, 3, 4, 5, 6, 5, 4, 6, 4. A nagy detektív mélyet szívott a pipájából, majd így szólt:

– Valaki tévedett, vagy – ami rosszabb – szándékosan hazudott.

Honnan tudta?

Megoldás: Legyenek a G gráf csúcsai a bűntény estéjén a klubban tartózkodó vendégek és az aznap este lezajlott sakkpartik feleljenek meg G éleinek: bármely két klubtag között pontosan annyi él fusson G-ben, ahány partit azok egymással játszottak (ami persze lehet nulla is). Ekkor a Dr. Watson által begyűjtött számok az egyes csúcsokra illeszkedő élek számának felelnek meg.

Mit kapunk, ha ezeket a számokat összeadjuk? Az összeghez G minden éle pontosan kettővel járul hozzá; valóban, ha $e = \{u, v\}$ a G egy éle (aminek a végpontjai tehát az u és a v csúcsok), akkor e az u-ra és a v-re illeszkedő élek között is beszámítódik az összegbe, de a többi csúcsra illeszkedő élek között nem. Következésképp ez az összeg egyenlő kell legyen a G élei számának a duplájával.

A Dr. Watson által bemutatott számok összege azonban 43; mivel ez páratlan, ezért nem lehet az élek számának a duplája. Sherlock Holmes tehát innen tudhatta, hogy nem létezik olyan gráf, amiben az egyes csúcsokra illeszkedő élek számai a Dr. Watson által mutatott számsor tagjaival egyeznek meg. □

Sherlock Holmes fenti gondolatmenete a gráfelmélet egyik legegyszerűbb és legalapvetőbb tételét bizonyítja be. Egy G gráfban a $v \in V(G)$ csúcs *fokának* (vagy *fokszámának*) nevezzük és $d(v)$-vel jelöljük a v-re illeszkedő G-beli élek számát; azonban a v-re illeszkedő hurokélek (ha vannak) duplán számítanak, egy helyett kettővel járulnak hozzá $d(v)$-hez. Így például az 1.2. ábra gráfjában $d(y) = 2$,

$d(u) = d(x) = 4$ és $d(v) = d(w) = 6$. Ha G egyszerű gráf, akkor $d(v)$ egyenlő v szomszédainak a számával (hiszen a v-re illeszkedő élek másik végpontjai mind különbözők), nem feltétlen egyszerű gráfokban azonban ez már nem igaz.

Az 1.2. Feladat G gráfjában hurokélek persze nem voltak, hiszen magával senki sem sakkozhatott (de párhuzamos élek lehettek, ha két klubtag több partit is játszott egymással). A csúcsok fokának a definíciójában éppen azért kötöttük ki, hogy a hurokélek duplán számítanak, hogy Sherlock Holmes megfigyelése a hurokéleket tartalmazó gráfokra is igaz maradjon.

1.3. Tétel. *Legyen G tetszőleges gráf. Ekkor*

$$\sum_{v \in V(G)} d(v) = 2 \cdot |E(G)|,$$

vagyis G-ben a csúcsok fokainak az összege egyenlő G élszámának a kétszeresével.

Bizonyítás: G minden éle kettővel járul hozzá a csúcsok fokának összegéhez: ha $e = \{u, v\}$ nem hurokél, akkor egyszer u-nál és egyszer v-nél számítódik be az összegbe, ha pedig e a w csúcsra illeszkedő hurokél, akkor w fokát növeli kettővel. Így az összegzés eredménye valóban az élszám kétszerese. □

1.2. Részgráf, komplementer gráf

Minden gráf (kivéve az egy csúcsú, üres gráfot) tartalmaz további, kisebb gráfokat, amiknek a csúcsai és élei az eredeti gráf csúcsai és élei közül kerülnek ki. Ha például a G gráf egy térképet reprezentál, akkor előfordulhat, hogy egy alkalmazásban G-ből csak egy adott országrész térképére, vagy a teherautóval bejárható útszakaszokból álló térképre van szükségünk. Ezeket a G gráf *részgráfjainak* nevezzük.

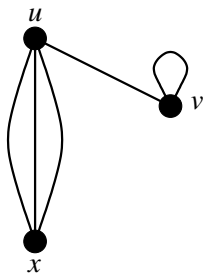
Egy G gráf részgráfjait tehát bizonyos csúcsok és élek elhagyásával kaphatjuk G-ből. Ezeknek az elhagyásoknak azonban nyilván úgy kell történniük, hogy minden elhagyott csúccsal együtt az összes rá illeszkedő élt is elhagyjuk, különben a megmaradó csúcsok és élek nem alkotnának gráfot.

1.4. Definíció.

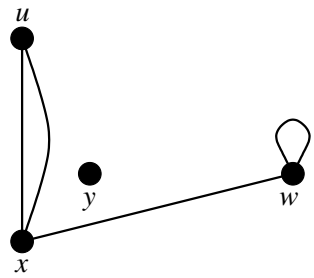
- *A (legalább két csúcsú) G gráfból a $v \in V(G)$* csúcs törlése *azt jelenti, hogy $V(G)$-ből elhagyjuk v-t és $E(G)$-ből elhagyjuk az összes, v-re illeszkedő élt.*
- *Egy $e \in E(G)$* él törlése *azt jelenti, hogy $E(G)$-ből elhagyjuk e-t (de ez e végpontjait nem érinti).*
- *Azokat a gráfokat, melyek megkaphatók G bizonyos csúcsainak és éleinek törlésével (amiknek a száma akár nulla is lehet), G* részgráfjának *nevezzük.*

A definíciónak közvetlen következménye, hogy minden gráfnak részgráfja saját maga is. Az 1.3. ábrán az 1.2. ábra gráfjának két különböző részgráfja látható. Ezek

közül az első az y és w csúcsok (és az ezekre illeszkedő élek) törlésével készült, a második pedig úgy, hogy a v csúcs törlése után kapott gráfból még a w-re illeszkedő egyik hurokélt és az u és x között futó három él egyikét is töröltük. (Az 1.3b ábra gráfjában az y csúcsra egyetlen él sem illeszkedik, de attól még y eleme a gráf csúcshalmazának; az ilyen csúcsokat *izolált pontnak* nevezzük.)



1.3a ábra



1.3b ábra

A részgráfok között külön jelentőséggel bírnak azok, amik az 1.3a ábra gráfjához hasonlóan csak csúcsok törlésével készülnek.

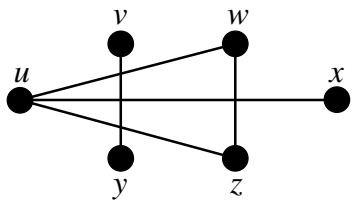
1.5. Definíció. *A G gráf egy H részgráfját* feszített részgráfnak *nevezzük, ha H megkapható G bizonyos csúcsainak a törlésével, élek további törlése nélkül. Ha a $V(H) = X \subseteq V(G)$ halmaz csúcsai maradnak meg a törlések után (vagyis a $V(G) \setminus X$ halmaz csúcsait töröltük), akkor H-t az X* csúcshalmaz által feszített részgráfnak *hívjuk.*

A H részgráf feszített volta tehát azt jelenti, hogy H-nak a G összes olyan élét tartalmaznia kell, amiknek a végpontjai $V(H)$-beliek. Így például az 1.3a ábra gráfja feszített részgráfja az 1.2. ábrán látható G gráfnak, de az 1.3b ábra gráfja már nem az (mert hiányzik belőle G két olyan éle, amiknek a végpontjai az $\{x, y, u, w\}$ halmazban vannak). G feszített részgráfjai tehát kölcsönösen egyértelműen megfelelnek $V(G)$ részhalmazainak: G csúcsainak minden részhalmaza egyértelműen meghatározza G egy feszített részgráfját.

Egy részgráfból természetesen nem rekonstruálhatók a törölt csúcsok és élek, vagyis egy részgráf készítésekor „információt veszítünk". Létezik azonban egy olyan, igen egyszerű fogalom is, amivel egy gráfból úgy készíthető egy másik, hogy ezzel ne veszítsünk információt; ez azonban már csak egyszerű gráfok esetén értelmezett. Erre példaként tekintsük ismét a (kölcsönösnek feltételezett) ismeretségek gráfját egy társaságban vagy egy közösségi oldalon: a csúcsok az emberek és két különböző ember pontosan akkor szomszédos, ha ismerik egymást. Ezzel a gráffal azonos „információt" hordoz az a másik gráf, aminek a csúcsai ugyanazok az emberek, de két különböző embert akkor kötünk össze éllel, ha nem ismerik egymást. Valóban: teljesen mindegy, hogy melyik gráf adott, bármelyikből megtudhatjuk, hogy két adott ember ismeri-e egymást vagy sem. Ezt a másik gráfot az eredeti gráf *komplementerének* nevezzük.

1.6. Definíció. *A G egyszerű gráf* komplementerének *nevezzük és $\overline{G}$-vel jelöljük azt a gráfot, amire $V(\overline{G}) = V(G)$ és $E(\overline{G}) = \big\{\{u,v\} : u,v \in V(G), \{u,v\} \notin E(G)\big\}$. Szövegben: $\overline{G}$ csúcshalmaza azonos G csúcshalmazával és két különböző csúcs pontosan akkor szomszédos $\overline{G}$-ben, ha G-ben nem szomszédosak.*

Fontos tehát rögzíteni, hogy a komplementer fogalma csak egyszerű gráfokra értelmezett. Például az 1.4 ábra gráfja az 1.1. ábrán (két példányban) is látható G gráf $\overline{G}$ komplementere.



1.4. ábra

1.7. Feladat. A 21 csúcsú G egyszerű gráfról és annak a $\overline{G}$ komplementeréről tudjuk, hogy mindkettőnek van 8 darab 4 fokú és 5 darab 10 fokú pontja. Hány éle van G-nek?

Megoldás: Jelölje $d_G(v)$, illetve $d_{\overline{G}}(v)$ a v csúcs fokát G-ben, illetve $\overline{G}$-ben. Nyilván $d_G(v) + d_{\overline{G}}(v) = 20$ minden v csúcsra igaz, mert v bármely, tőle különböző csúccsal G és $\overline{G}$ közül pontosan az egyikben szomszédos (de saját magával egyikben sem). Így $\overline{G}$ 10 fokú pontjai azonosak G 10 fokú pontjaival, $\overline{G}$ 4 fokú pontjai viszont G-ben 16 fokúak. Ezek alapján G-ben már minden csúcs fokát ismerjük: 8 darab 4 fokú, 5 darab 10 fokú és 8 darab 16 fokú csúcsa van.

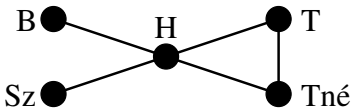
Ezek szerint G-ben a csúcsok összfokszáma $8 \cdot 4 + 5 \cdot 10 + 8 \cdot 16 = 210$, így az 1.3. Tétel szerint G éleinek száma ennek a fele, vagyis 105. □

1.3. Gráfok izomorfiája

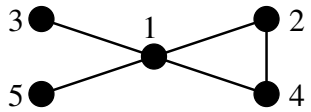
Megváltozik-e egy gráf attól, ha egy v csúcsát lecseréljük egy vadonatúj, másik elemre, amit ugyanazokkal a csúcsokkal kötünk össze (és ugyanannyi él mentén), mint amikkel v volt szomszédos? A válasz szorosan véve igen, hiszen a gráf fogalmát úgy értelmeztük, hogy az egy csúcshalmazból és egy élhalmazból áll, így két gráf nem lehet azonos, ha a csúcshalmazuk különböző. Valójában azonban érezhető, hogy egy ilyen csere a gráfelmélet szempontjából érdektelen.

Az 1.5a ábra például az ismeretségek gráfját ábrázolja abban a társaságban, amit a házigazda és négy vendége, Bakács úr, Szakács úr, Takács úr és Takácsné alkot; a

vendégek közül csak a Takács házaspár tagjai ismerik egymást, a házigazda viszont mindenkit ismer. Az 1.5b ábrán látható gráf csúcshalmaza pedig az $\{1,2,3,4,5\}$ számokból áll és két különböző szám pontosan akkor szomszédos, ha a kisebbik osztója a nagyobbiknak.



1.5a ábra



1.5b ábra

Hasonlóan a fenti példához, az 1.5 ábra két gráfját is különbözőnek kell tekintenünk, hiszen a csúcshalmazaik mások. Mégis, minden érdemi – vagyis a gráfelmélet számára releváns – szempont szerint azonosak. Ezt a jelenséget definiálja pontosan a gráfok *izomorfiájának* a fogalma. Ennek az alapgondolata az, hogy ha a G és H gráfok csúcsai párba állíthatók úgy, hogy a G-beli csúcsok közti élek megfelelnek a párjaik között futó éleknek, akkor G és H „lényegében azonosak”, vagyis pontos kifejezéssel izomorfak. Ezt a párbaállítást pedig egy kölcsönösen egyértelmű függvény adja meg, ami minden G-beli csúcshoz a H-beli párját rendeli hozzá. (A függvény kölcsönös egyértelműsége azt jelenti, hogy H minden csúcsa pontosan egy G-beli csúcshoz van hozzárendelve.)

1.8. Definíció. *A G és H gráfok* izomorfak, *ha létezik olyan $f : V(G) \to V(H)$ kölcsönösen egyértelmű függvény, amire a következő teljesül: bármely $u, v \in V(G)$ csúcsok esetén az u és v között futó G-beli élek száma egyenlő az $f(u)$ és $f(v)$ között futó H-beli élek számával. Ennek a jele: $G \cong H$.*

A definícióban az f-re tett feltétel az $u = v$ esetben is értelmes: ilyenkor azt mondja, hogy a v-re illeszkedő G-beli hurokélek száma egyenlő az $f(v)$-re illeszkedő H-beli hurokélek számával.

Ha például G, illetve H jelöli az 1.5a, illetve az 1.5b ábrák gráfjait, akkor a $G \cong H$ izomorfiát bizonyítja az az $f : \{\text{B, T, H, Sz, Tné}\} \to \{1,2,3,4,5\}$ függvény, amire $f(\text{B}) = 3$, $f(\text{T}) = 2$, $f(\text{H}) = 1$, $f(\text{Sz}) = 5$ és $f(\text{Tné}) = 4$. (Valóban, az öt fős társaság bármely két tagja pontosan akkor ismeri egymást, ha a nekik az f által megfeleltetett két szám között oszthatóság áll fenn.)

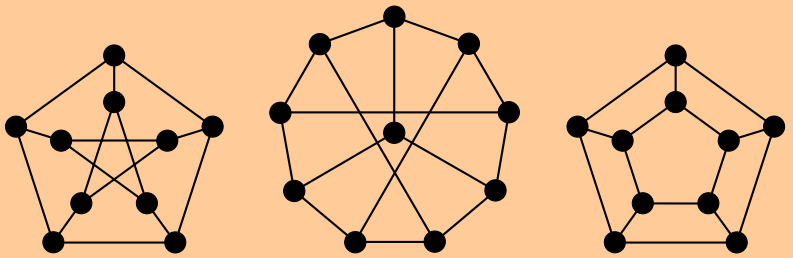
Mivel az izomorfia fogalma két gráf „lényegében azonos” voltának ad precíz értelmet, elvárható, hogy ha $G \cong H$ és $H \cong K$ teljesül a G, H és K gráfokra, akkor $G \cong K$ is fennálljon. Ez a tulajdonság, amit *tranzitivitásnak* szoktak nevezni, valóban teljesül: ha $f : V(G) \to V(H)$, illetve $g : V(H) \to V(K)$ igazolják az első két izomorfiát az 1.8. Definíció szerint, akkor $g \circ f$ (vagyis a két függvény kompozíciója) könnyen láthatóan igazolja a $G \cong K$ izomorfiát. (Ugyancsak teljesül két további, igen egyszerű és a fogalom intuitív jelentése alapján elvárható tulajdonság: a *reflexivitás*, ami szerint $G \cong G$ minden gráfra igaz és a *szimmetria*, ami szerint $G \cong H$ esetén $H \cong G$ is igaz. Az előbbit az 1.8. Definíciónak megfelelően az identitás függvény bizonyítja, ami tehát minden $v \in V(G)$ csúcshoz saját magát rendeli, az utóbbi

pedig az inverz függvénnyel bizonyítható: ha $f : V(G) \to V(H)$ igazolja a $G \cong H$ izomorfiát, akkor $f^{-1} : V(H) \to V(G)$ igazolja a $H \cong G$ izomorfiát.)

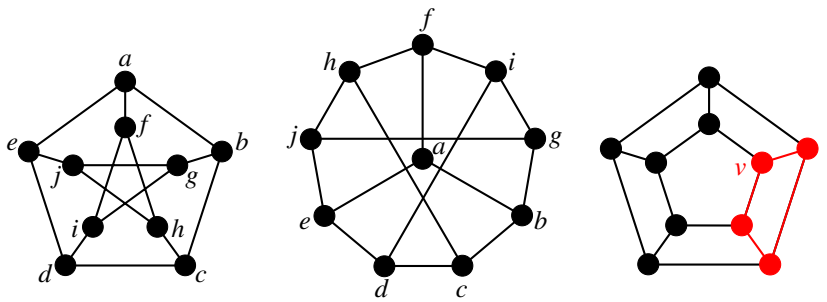
Az 1.8. Definíció egy megfelelő f függvény létezését írja elő $G \cong H$ teljesüléséhez, az azonban előfordulhat, hogy több különböző f függvény is alkalmas az izomorfia bizonyítására. Például az 1.5. ábra példájának esetében módosíthatjuk a fenti f függvényt úgy, hogy $f(\mathrm{B}) = 5$ és $f(\mathrm{Sz}) = 3$ legyen (és f-nek a másik három elemen felvett értéke változatlan maradjon), az így kapott függvény is megfelel az 1.8. Definíciónak.

Szemben azzal, amit az 1.5 ábra példája talán sugall, általában egyáltalán nem könnyű feladat eldönteni, hogy két adott gráf izomorf-e vagy sem. Gondoljunk vissza például az 1.1 ábra két gráfjára: ezek persze izomorfak, hiszen ez a két rajz éppen azt volt hivatva illusztrálni, hogy a csúcsok és élek elhelyezkedése érdektelen a gráfok ábrázolásakor; így az izomorfiát az 1.8. Definíciónak megfelelően igazolja az a függvény, ami az azonos betűkkel jelölt csúcsokat rendeli egymáshoz. Ha azonban a betűk nem szerepeltek volna eleve a két rajzon, akkor okozhatott volna némi fejtörést a megfelelő párok megtalálása.

1.9. Feladat. Melyek izomorfak az alábbi gráfok közül?



Megoldás: Az első két gráf izomorf, amit például a csúcsoknak az alábbi ábrán látható betűzésével lehet igazolni: nem nehéz ellenőrizni, hogy a bal oldali gráf bármely két csúcsa pontosan akkor szomszédos, ha a középső gráf azonosan betűzött két csúcsa szomszédos.



A fenti ábrán látható betűzéssel tehát megadtunk egy, az 1.8. Definíciónak megfelelő függvényt: ez a bal oldali gráf bármely csúcsához a középső gráf azonosan

betűzött csúcsát rendeli. Az ábrából persze nem derül ki, hogy az izomorfiát bizonyító betűzés hogyan született, de a valóság az, hogy emögött nem áll semmilyen biztos módszer vagy algoritmus, csak szisztematikus próbálkozás. Az viszont kétségtelen, hogy ha a betűzést (vagyis a csúcshalmazok közötti függvényt) valaki megadja, akkor annak az ellenőrzése, hogy ez valóban megfelel az 1.8. Definíciónak, már könnyű feladat. Éppen ezért, gyakran nagyobb kihívást jelent annak az igazolása, hogy két adott gráf nem izomorf: ehhez ugyanis azt kell belátni, hogy a csúcsaiknak semmilyen betűzése (vagyis semmilyen, a csúcshalmazokat egymáshoz rendelő függvény) nem felel meg az 1.8. Definíciónak.

A jobb szélső gráf esetében pontosan ez a helyzet: azt állítjuk ugyanis, hogy ez nem izomorf a másik két gráffal. (Mivel ezekről már kiderült, hogy egymással izomorfak, ezért az 1.8. Definíció után, az izomorfia tranzitivitásával kapcsolatban írtak szerint a harmadik vagy mindkettővel izomorf, vagy egyikkel sem.) Ennek a bizonyításához figyeljük meg a harmadik gráfnak a fenti ábrán pirossal kiemelt részletét: ez egy négyszög, vagyis egy (tetszőleges) v csúcsból három további csúcs érintésével a gráf élei mentén haladva visszajuthatunk v-be. Ha a harmadik gráf izomorf volna a másik kettővel, akkor nyilván azokban is kellene lennie négyszögnek. (Valóban, ha két gráf izomorfiáját az f függvény igazolná az 1.8. Definíció szerint, akkor az első gráfban négyszöget alkotó csúcsok f-fel vett képei a másodikban is négyszöget alkotnának.) Nem nehéz azonban ellenőrizni, hogy nem ez a helyzet: ugyanis az első két gráf bármelyik v csúcsát is szemeljük ki, ennek a szomszédai között nincs két olyan, amiknek v-n kívül is van közös szomszédja. Ezzel tehát beláttuk, hogy a harmadik gráf valóban nem izomorf a másik kettővel. □

> Az érdekesség kedvéért megemlítjük, hogy a fenti feladatban szereplő első (két) gráfot Julius Petersen (1839 – 1910) dán matematikus után Petersen-gráfnak nevezik. Ez speciális jelentőséggel bír a gráfelméletben, számos probléma kapcsán kerül elő, mint példa vagy ellenpélda valamilyen tulajdonság vagy állítás teljesülésére.

Ha két gráf izomorf, akkor persze a csúcsaik és éleik száma is egyenlő. Sőt, mivel az izomorfiát bizonyító függvény által egymásnak megfeleltetett csúcsok foka is nyilván mindig egyenlő, ezért izomorf gráfok csúcsainak a fokait felírva a két esetben ugyanazt a számsorozatot kell kapnunk. Fontos azonban hangsúlyozni, hogy ennek a fordítottja nem igaz: ha két gráfnak ugyanannyi csúcsa van és a pontok fokai is azonosak, attól még nem feltétlen izomorfak. Erre példát mutatnak a fenti feladat gráfjai: bár mindháromnak tíz csúcsa van és minden csúcs foka három, az utolsó gráf mégsem izomorf a másik kettővel.

1.4. Összefüggőség

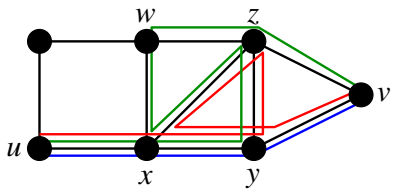
A gráfelmélet számtalan alkalmazásában fordul elő, hogy egy gráf valaminek a továbbítására vagy szállítására használt hálózatot modellez. Magától értetődő példa erre, ha G egy térképet reprezentál, de hasonló a helyzet egy számítógép-hálózatot vagy egy légitársaság által üzemeltetett járatokat ábrázoló gráf esetében is – és még hosszan lehetne sorolni a példákat. Az alábbi definíció az ilyen alkalmazásokhoz szükséges, a gráfban való „mozgást” leíró alapfogalmakat vezeti be.

1.10. Definíció.

- *A $P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ sorozatot* élsorozatnak *nevezzük a G gráfban, ha $v_0, v_1, \ldots, v_k \in V(G)$ a gráf csúcsai, $e_1, e_2, \ldots, e_k \in E(G)$ a gráf élei és minden $i = 1, 2, \ldots, k$ esetén $e_i = \{v_{i-1}, v_i\}$ (vagyis az e_i él végpontjai v_{i-1} és v_i).*
- *A P élsorozat* végpontjai *v_0 és v_k. P-t az u-ból v-be vezető* élsorozatnak *mondjuk, ha a végpontjai (valamilyen sorrendben) u és v.*
- *A P élsorozat* hossza *k, vagyis a benne szereplő élek száma.*
- *A P élsorozatot* sétának *nevezzük, ha az $e_1, e_2, \ldots, e_k$ élek közül bármely kettő különböző.*
- Útnak *pedig akkor nevezzük P-t, ha a $v_0, v_1, \ldots, v_k \in V(G)$ csúcsok közül bármely kettő különböző.*

A definíció közvetlen következménye, hogy minden út egyben séta is; valóban, a csúcsok ismétlődésének a kizárása az élek ismétlődését is kizárja, hiszen egy ismétlődő él végpontjai is többször szerepelnének az élsorozatban. A definíció szerint G egyetlen csúcsa is utat alkot G-ben, aminek a hossza nulla.

Egy élsorozat elemei tehát (szemben azzal, amit a fogalom neve sugall) felváltva csúcsok és élek. Ha azonban G egyszerű gráf, akkor fölösleges megadni az élsorozatban szereplő éleket is, hiszen szomszédos csúcsok között csak egy él futhat; ilyenkor tehát az élsorozat által érintett csúcsok (sorrendben való) felsorolása már meghatározza az élsorozatot.



1.6. ábra

Az 1.6. ábrán három élsorozatot látunk az ábra gráfjában, mindegyik az u csúcsból a v-be vezet. A kékkel jelölt, sorra az (u, x, y, v) csúcsokat érintő élsorozat út, hiszen nem ismétlődnek benne a csúcsok. A zölddel jelölt, az (u, x, y, z, x, w, z, v) csúcsokat érintő élsorozat már nem út (hiszen az x és z csúcsokat kétszer is tartalmazza), de séta, hiszen az általa érintett élek nem ismétlődnek. A piros élsorozat viszont, ami sorra az (u, x, y, z, x, y, v) csúcsokat érinti, már nem is séta, mert az $\{x, y\}$ élen kétszer is áthalad.

1.11. Állítás. *Ha a G gráfban létezik az u csúcsból a v-be vezető élsorozat, akkor létezik u-ból v-be vezető út is.*

Bizonyítás: Legyen P_1 egy u-ból v-be vezető élsorozat G-ben. Ha P_1 út, akkor készen vagyunk a bizonyítással. Ha nem, akkor létezik olyan x csúcs, amit egynél többször

érint. Vágjuk most ki P_1-ből az x első és utolsó előfordulása közötti szakaszt (abban az értelemben, hogy P_1-nek ebből az x-től x-ig tartó szakaszából csak maga az x csúcs maradjon meg), a kapott sorozat legyen P_2. Ekkor P_2 is u-ból v-be vezető élsorozat, de a hossza (vagyis a benne szereplő élek száma) már kisebb P_1-énél.

Ha P_2 már út, akkor megint készen vagyunk a bizonyítással. Ha nem, akkor P_2 is tartalmaz ismétlődő csúcsot, így a fentiek szerint ismét kivágható belőle egy szakasz úgy, hogy egy P_2-nél is rövidebb, u-ból v-be vezető P_3 élsorozatot kapjunk. Ez az eljárás véges sok lépés után megáll és szolgáltatja a kívánt, u-ból v-be vezető utat, mert az általa előállított élsorozatok hossza folyamatosan csökken. □

Ha a fenti bizonyításban leírt eljárást például az 1.6. ábra piros élsorozatára alkalmazzuk, akkor az (x,y,z,x) csúcsokat érintő szakasz kivágása (vagyis pusztán az x csúccsal való helyettesítése) után épp a kékkel jelölt utat kapjuk. Ha viszont a zölddel jelölt sétára alkalmazzuk ezt az eljárást, akkor szintén az (x,y,z,x) csúcsokat érintő szakasz kivágása után az (u,x,w,z,v) csúcsokat sorra érintő úthoz jutunk; de alkalmazhatjuk a bizonyításban leírt eljárást úgy is, hogy a zöld séta z-től z-ig tartó szakaszát vágjuk ki, ebben az esetben az (u,x,y,z,v) utat kapjuk.

1.12. Feladat. Legyen v a G gráfnak egy páratlan fokszámú csúcsa. Mutassuk meg, hogy létezik G-ben olyan v-ből induló (pozitív hosszúságú) út, aminek a másik végpontja is páratlan fokszámú.

Megoldás: Induljunk el v-ből és járjunk be „vaktában” egy P sétát G-ben – vagyis mindig csak arra vigyázzunk, hogy olyan élen lépjünk tovább, ami korábban még nem szerepelt P-ben. Azt állítjuk, hogy ha P a w csúcsban akad el, akkor w páratlan fokú és $w \neq v$. A P séta ugyanis párosával „fogyasztja” az általa érintett, v-től különböző csúcsok éleit: egy $u \neq v$ csúcson való áthaladása nyilván kettőt használ fel u élei közül. Így w valóban páratlan fokú kell legyen, különben P-nek az utolsó w-be való belépése után még volna olyan, w-re illeszkedő él, amit P korábban nem érintett, így P nem akadhatna el w-ben. Ugyanez a gondolat mutatja azt is, hogy $w \neq v$; valóban, P első, v-ből induló éle elhasznál egyet v élei közül, így az ezután megmaradó, v-re illeszkedő élek száma már páros, vagyis P v-ben sem akadhat el.

Megmutattuk tehát, hogy létezik egy v-ből w-be vezető P séta (ahol $w \neq v$ és w páratlan fokú). Így az 1.11. Állítás szerint v-ből w-be vezető út is van G-ben.

(Később egy másik megoldást is adunk ugyanerre a feladatra; lásd az 1.18. Feladatot.) □

1.13. Következmény. *Ha a G gráfban létezik az u csúcsból a v-be vezető út és létezik a v csúcsból a w-be vezető út is, akkor létezik az u-ból a w-be vezető út is.*

Bizonyítás: Vezessen a P út u-ból v-be, a Q út pedig v-ből w-be. Ekkor P és Q összekapcsolásával egy u-ból w-be vezető R élsorozatot kapunk. Bár R nem feltétlen út (hiszen lehetnek olyan csúcsok és élek, amik P-ben és Q-ban is szerepelnek, így

ezeket R egynél többször tartalmazza), de az 1.11. Állítás miatt R létezéséből egy u-ból w-be vezető út létezése is következik. □

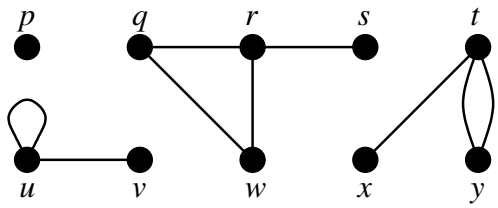
Érdemes a fenti bizonyítást az 1.6. ábra gráfján szemléltetni. Ha például a P út az u csúcsból a z-be vezet és sorban az (u,x,y,z) csúcsokat érinti, a z-ből v-be vezető Q út pedig a (z,x,y,v) csúcsokon halad át, akkor P és Q valóban utak, de az összekapcsolásukkal éppen az ábrán pirossal jelölt R élsorozatot kapjuk. Ez tehát nem út (sőt, még csak nem is séta), de a fent leírtak szerint valóban nyerhető belőle egy u-ból v-be vezető (az ábrán kékkel jelölt) út.

Minden olyan alkalmazásban, ahol a gráfot (a szakasz elején írtak szerint) valaminek a továbbítására vagy szállítására használjuk, nyilván alapvető elvárás, hogy a gráf bármely csúcsából bármelyik másikba el lehessen jutni az élek mentén haladva. Ezért a következő fogalom kiemelten fontos a gráfelméletben.

1.14. Definíció. *A G gráfot* összefüggőnek *nevezzük, ha bármely $u,v \in V(G)$, $u \neq v$ csúcsaira létezik G-ben u-ból v-be vezető út.*

Fontos megjegyezni, hogy ha a definíciót úgy módosítanánk, hogy bármely két, különböző csúcs közötti élsorozat létezését követelnénk meg, ettől az összefüggőség fogalma nem változna; valóban, az 1.11. Állítás szerint u és v közötti élsorozat létezéséből u-ból v-be vezető út létezése is következik (ennek a fordítottja pedig magától értetődően igaz).

A fejezetben eddig szerepelt gráfok szinte kivétel nélkül mind összefüggők voltak, de az 1.4 ábra gráfja nem az: például a v csúcsból (saját magán kívül) csak az y-ba lehet eljutni (úton vagy élsorozaton), a gráf összes többi csúcsába már nem; ugyancsak nyilván nem összefüggő az 1.7 ábra gráfja sem.



1.7. ábra

1.15. Feladat. A 100 csúcsú G egyszerű gráf v csúcsának a foka 66, az összes többi csúcsának a foka legalább 33. Mutassuk meg, hogy G összefüggő.

Megoldás: Ha $w \neq v$ tetszőleges, v-vel nem szomszédos csúcs, akkor v-nek és w-nek kell legyen közös szomszédja. Valóban, mivel egyszerű gráfokban a csúcsok fokszáma azonos a szomszédaiknak a számával, ezért v-nek és w-nek együttesen legalább $66+33=99$ szomszédja van. Így lehetetlen, hogy ezek között ne legyen közös,

mert v-n és w-n kívül G-nek csak 98 csúcsa van. Ezért v és bármely $w \neq v$ csúcs között van G-ben (legföljebb kettő hosszúságú) út.

Ebből következik, hogy bármely $x, y \in V(G)$ csúcsok között létezik G-ben élsorozat: ilyet kapunk, ha egy x-ből v-be vezető utat összekapcsolunk egy v-ből y-ba vezetővel. Így az 1.11. Állítás szerint x-ből y-ba vezető út is van G-ben, vagyis G valóban összefüggő.

(Később egy másik megoldást is adunk ugyanerre a feladatra; lásd az 1.19. Feladatot.) □

Szemléletesen érezhető, hogy ha egy gráf nem összefüggő, akkor „több különálló, önmagán belül összefüggő részből áll"; például az 1.7 ábra gráfjának négy ilyen része van (közülük az egyik csak a p izolált pontból áll). Nem magától értetődő viszont, hogy hogyan értelmezzük az ezeknek a részeknek megfelelő fogalmat; az alábbi definíció ezt vezeti be.

1.16. Definíció. *Legyen G egy gráf és $v \in V(G)$ egy tetszőleges csúcsa.*

- *A $w \in V(G)$ csúcsot* v-ből úton elérhetőnek *(vagy röviden: v-ből elérhetőnek) nevezzük, ha létezik G-ben v és w között út.*
- *Jelölje X a v-ből úton elérhető G-beli csúcsok halmazát. Ekkor az X által feszített részgráfot a v csúcs* komponensének *nevezzük.*
- *A G gráfnak egy K feszített részgráfja akkor* összefüggő komponens *(vagy röviden:* komponens*) G-ben, ha G-nek van olyan csúcsa, aminek a komponense K.*

Nyilván minden csúcsból elérhető saját maga is (nulla hosszúságú úton), így minden csúcs definíció szerint benne van a saját komponensében. Például az 1.7 ábra gráfjában a q csúcsból az $X = \{q, r, s, w\}$ halmaz csúcsai érhetők el, így az X által feszített (négy élű) K részgráf komponens G-ben. Az X másik három csúcsából is az X elemei érhetők el úton, így ezeknek a komponense is K. A p csúcsból viszont nem érhető el saját magán kívül más csúcs, így egymaga alkot egy (él nélküli) komponenst. Ennek a gráfnak összesen négy komponense van, a másik kettőt az $\{u, v\}$, illetve a $\{t, x, y\}$ csúcshalmazok feszítik. Az 1.4 ábra gráfjának pedig két komponense van, ezeket a $\{v, y\}$ és az $\{u, w, x, z\}$ csúcshalmazok feszítik.

A definíciónak közvetlen következménye az a (szintén természetesnek érződő) állítás, hogy egy G gráf pontosan akkor összefüggő, ha egyetlen komponense van. Valóban: ha G összefüggő, akkor minden v csúcsának a komponense G, mert v-ből úton elérhető az összes többi csúcs; megfordítva, ha G-nek egyetlen komponense van és az K, akkor bármely $x, y \in V(G)$ esetén x és y komponense is K kell legyen, így x és y egymásból elérhetők, vagyis G összefüggő (és $K = G$).

Ahogyan az a fogalom nevéből is sejthető, a komponensek mindig összefüggők. Továbbá a fenti példák alapján érezhető, hogy a komponensek mindig diszjunkt részhalmazokra vágják fel egy gráf csúcshalmazát. Az alábbi tétel épp ezeket mondja ki.

1.17. Tétel. *Bármely G gráf esetén*
(i) G komponensei összefüggők;
(ii) G minden csúcsát G-nek pontosan egy komponense tartalmazza.

Bizonyítás: Legyen K egy komponens G-ben, tartozzon például a v csúcshoz. Ha $x, y \in V(K)$ tetszőleges csúcsok, akkor (az 1.16. Definíció szerint) mindketten elérhetők v-ből úton. Így az 1.13. Következmény szerint x és y egymásból is elérhetők. Mivel K bármely két csúcsa között vezet út, ezért az valóban összefüggő.

Rátérve *(ii)* bizonyítására, annyi ebből magától értetődő (és fentebb már említettük is), hogy minden csúcs benne van egy komponensben: a sajátjában. Tegyük fel, hogy a $v \in V(G)$ csúcs a saját komponensén kívül a w csúcséban is benne van. Jelölje ezt a két komponenst K_v, illetve K_w; meg kell mutatnunk, hogy $K_v = K_w$. Legyen ezért $x \in V(K_w)$ tetszőleges csúcs. Mivel $v \in V(K_w)$ is igaz, ezért v és x is elérhetők úton w-ből, így az 1.13. Következmény szerint ezek egymásból is elérhetők. Ebből következik, hogy $x \in V(K_v)$ is igaz. Beláttuk tehát, hogy $V(K_w) \subseteq V(K_v)$, de a fordított irányú tartalmazás ugyanígy megmutatható: egy $y \in V(K_v)$ csúcs esetén y elérhető v-ből, v-pedig $v \in V(K_w)$ miatt w-ből, így az 1.13. Következmény miatt y is w-ből, vagyis $y \in V(K_w)$ valóban igaz. Mivel K_v és K_w feszített részgráfok és a csúcshalmazaik a fentiek szerint azonosak, ezért $K_v = K_w$ valóban igaz. □

A fenti tételből valóban következik, hogy egy gráf komponensei diszjunkt részekre vágják a csúcshalmazát (hiszen minden csúcs benne van egy komponensben és ha két különböző komponens csúcshalmaza nem volna diszjunkt, akkor a metszetükben lévő csúcsok egynél több komponensbe tartoznának). A tétel választ ad arra a kérdésre is, hogy ha K a G gráf egy komponense, akkor G-nek melyik v csúcsaira teljesül, hogy a v komponense éppen K? A válasz az, hogy ezek a v-k épp a K-ba tartozó csúcsok; vagyis a „K komponens csúcsai”, illetve az „azok a v-k, amiknek a komponense K” megfogalmazások ugyanazokat a csúcsokat takarják. Valóban: $v \in V(K)$ esetén v komponense csak K lehet, mert v benne van a saját komponensében, de egynél többen nem lehet; megfordítva, ha v komponense K, akkor $v \in V(K)$ nyilvánvaló.

Fontos hangsúlyozni, hogy bár a komponensek az 1.17. Tétel szerint összefüggő, feszített részgráfok, ennek a megfordítása nem igaz: nem minden összefüggő, feszített részgráf komponens. Például az 1.7 ábra gráfjában az $X = \{q, r, s\}$ csúcshalmaz által feszített részgráf (ami egy kettő hosszúságú út) is összefüggő, de nem komponens (hiszen nincs olyan csúcsa a gráfnak, amiből elérhető csúcsok halmaza pontosan az X csúcsaiból állna).

A komponens fogalmának, illetve a fenti tételnek a birtokában rövidebb megoldást adhatunk két, korábban már látott feladatra.

1.18. Feladat. Legyen v a G gráfnak egy páratlan fokszámú csúcsa. Mutassuk meg, hogy létezik G-ben olyan v-ből induló (pozitív hosszúságú) út, aminek a másik végpontja is páratlan fokszámú.

Megoldás: Legyen G-ben a v komponense K. Ebben kell legyen v-n kívül is páratlan fokú csúcs, különben a K-beli csúcsok fokszámösszege páratlan volna, ami ellentmondana az 1.3. Tételnek. Egy ilyen csúcs és v között pedig (a komponens definíciója szerint) vezet út. □

1.19. Feladat. A 100 csúcsú G egyszerű gráf v csúcsának a foka 66, az összes többi csúcsának a foka legalább 33. Mutassuk meg, hogy G összefüggő.

Megoldás: Legyen G-ben a v komponense K. Ekkor K legalább 67 csúcsú, hiszen v-n kívül benne kell lennie v 66 szomszédjának is (amik mind különbözők, hiszen G egyszerű gráf). Hasonló érvelés mutatja, hogy ha G-nek volna K-n kívül egy másik K' komponense is, akkor annak legalább 34 csúcsúnak kellene lennie, hiszen minden csúcsa legalább 33 fokú. Ez azonban lehetetlen, mert K-nak és K'-nek együttesen már legalább $67+34=101$ csúcsa volna. Következésképp G-nek csak egy komponense van, vagyis összefüggő. □

Az összefüggő komponens fogalmát többféle módon is lehet definiálni. Mi az egyszerűsége miatt az 1.16. Definícióban írt utat választottuk, de sok tankönyv azokat az összefüggő, feszített részgráfokat nevezi komponensnek, amik nem bővíthetők az összefüggőség megtartásával. Pontosabban: a K feszített részgráfot akkor nevezik komponensnek, ha K összefüggő, de bárhogyan választunk egy $v \in V(G) \setminus V(K)$ csúcsot, G-nek a $V(K) \cup \{v\}$ csúcshalmaz által feszített részgráfja már nem összefüggő. Megmutatjuk, hogy a két definíció ekvivalens.

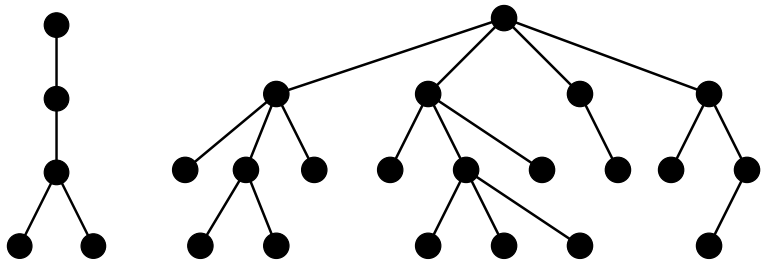
Tegyük fel először, hogy K az x csúcs komponense (az 1.16. Definícióban írt értelemben). Az 1.17. Tételből tudjuk, hogy K összefüggő. Ha egy $v \in V(G) \setminus V(K)$ csúcsra a $V(K) \cup \{v\}$ által feszített részgráf is összefüggő volna, akkor (mivel ennek x és v is csúcsai) volna út x és v között. Ez azonban lehetetlen, hiszen az 1.16. Definíció szerint $V(K)$ az összes, x-ből úton elérhető csúcsot tartalmazza, de v-t nem. Így K-ra az előző bekezdésben írt definíció valóban teljesül.

Tegyük most fel, hogy K a fentebb írt, alternatív definíció értelmében komponens és legyen $x \in V(K)$ egy tetszőleges csúcsa. Mivel K összefüggő, ezért minden csúcsa úton elérhető x-ből. Ha volna olyan $z \notin V(K)$ csúcs, ami szintén elérhető egy P úton x-ből, akkor induljunk el P mentén x-ből z-be és álljunk meg az első olyan csúcsnál, ami nem $V(K)$-beli. Ilyen létezik, mert $z \notin V(K)$; jelölje ezt v, a v-t megelőző csúcsot P-n pedig u. Ekkor a $V(K) \cup \{v\}$ halmaz által feszített részgráf összefüggő, mert bármely $y \in V(K)$ csúcsból vezet út u-ba (mert K összefüggő és $y, u \in V(K)$), ezt pedig az $\{u, v\}$ éllel (és v-vel) megtoldva y-ból v-be vezető utat kapunk. Ezért valóban igaz, hogy $V(K) \cup \{v\}$ bármely két csúcsa között vezet út G-ben (hiszen két $V(K)$-beli csúcs esetében ez K összefüggősége miatt magától értetődő). Mivel $V(K) \cup \{v\}$ a fenti, alternatív definíció teljesülése miatt nem feszíthet összefüggő részgráfot, ezért a $z \notin V(K)$ csúcs mégsem lehet x-ből úton elérhető. Ezzel tehát megmutattuk, hogy K az 1.16. Definíció értelmében az x csúcs komponense.

1.5. Fák

Képzeljük el azt a gráfot, aminek a csúcsai egy ember összes leszármazottai és két csúcs akkor szomszédos, ha a megfelelő két ember szülő-gyermek viszonyban áll

egymással. Az ilyen gráfokat a köznyelv családfának nevezi, a gráfelméleti szaknyelv pedig *fának*; az 1.8. ábrán két ilyen gráf látható. (Bár ezek az ábrák azt sugallják, hogy a legfelül elhelyezkedő csúcs a közös ős és alatta soronként az egyre későbbi generációk jelennek meg, de természetesen a rajz ezeknek a gráfoknak az esetében is esetleges; így például mindkét gráf bármelyik csúcsára kioszthatnánk a közös ős szerepét és ennek megfelelően átrajzolhatnánk az ábrákat.)



1.8a ábra **1.8b ábra**

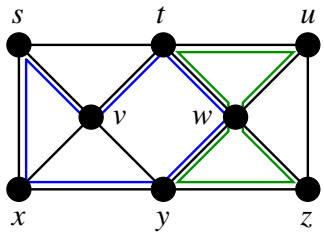
Fák a gráfelmélet számos alkalmazásában kerülnek elő, sok más mellett például olyan esetekben, amikor a csúcsok egy folyamat különböző állapotainak felelnek meg: a kezdőállapotban a folyamat több különböző irányba ágazhat, de mindegyik ágon több különböző további irány nyílhat, stb. Fa szerkezetű a legtöbb fájlrendszer is, itt a csúcsok a mappák és a fájlok. Az is ki fog derülni ebben a szakaszban, hogy a fák (egy pontosan definiált értelemben, lásd az 1.26. Tételt) az összefüggő gráfok „csontvázainak" is tekinthetők.

Felmerül persze a kérdés, hogy hogyan lehet a fenti leírás alapján kialakuló intuitív képnek megfelelő, pontos definíciót alkotni a fa fogalmáról. A fák láthatóan összefüggő gráfok, de ennél nyilván többről van szó: egy fában az élek mentén haladva csak akkor juthatunk vissza egy korábbi csúcsba, ha közben valamikor „visszafordulunk". Az alábbi definícióban bevezetett alapfogalmak – számos más alkalmazás mellett – ennek a pontos megfogalmazását is lehetővé teszik.

1.20. Definíció. *A G gráfban a $P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ élsorozatot*
- körsétának *nevezzük, ha P séta és $v_0 = v_k$;*
- körnek *nevezzük, ha P egy pozitív hosszúságú körséta és a $v_0, v_1, \ldots, v_{k-1}$ csúcsok mind különbözők.*

A körséta tehát azt jelenti, hogy a gráf egy csúcsából indulva úgy haladunk az élek mentén, hogy végül ugyanabba a csúcsba érkezünk vissza, de közben egy élen sem haladunk át egynél többször. A kör fogalma pedig azt jelenti, hogy a körséta során még a csúcsok sem ismétlődhetnek – kivéve azt, hogy az első és utolsó csúcs azonos. (Figyeljük meg, hogy a definíció csak a $v_0, v_1, \ldots, v_{k-1}$ csúcsok különbözőségét köti ki, de $v_0 = v_k$ következik abból, hogy P körséta.) Ebből persze adódik, hogy a körök egyben körséták is, de ez fordítva nem igaz. Az 1.9. ábra gráfjában

például kékkel jelölve egy kört látunk, zölddel jelölve pedig egy olyan körsétát, ami nem kör (mert w-n kétszer is áthalad). Az előbbi tehát sorra az (s,v,t,w,y,x,s) csúcsokat érinti, az utóbbi pedig az (u,w,z,y,w,t,u) csúcsokat (de persze mindkét esetben a kör, illetve a körséta bármelyik csúcsát választhattuk volna indulási és egyben érkezési csúcsnak).



1.9. ábra

Az 1.20. Definíció szerint egy egyetlen csúcsból és nulla darab élből álló élsorozat körséta, de nem kör, mert egy kör hossza (vagyis az éleinek a száma) pozitív kell legyen. Ha viszont a v csúcsra illeszkedik egy e hurokél, akkor a (v,e,v) élsorozat már kör, aminek a hossza egy. Ugyancsak kör a kettő hosszúságú (v,e_1,w,e_2,v) élsorozat, ha e_1 és e_2 párhuzamos élek a v és a w csúcsok között. Egyszerű gráfokban azonban nem létezhet háromnál rövidebb kör (hiszen hurokélek, illetve párhuzamos élek híján lehetetlen egy, illetve kettő hosszúságú kört alkotni).

A kör fogalmának a birtokában már pontosan definiálhatjuk a fát is.

1.21. Definíció. *Az F gráf* fa*, ha F összefüggő és nem tartalmaz kört.*

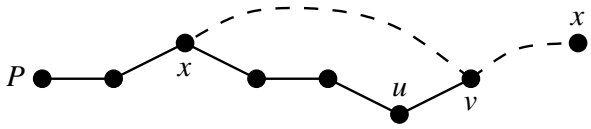
Az 1.8. ábra két gráfja valóban megfelel ennek a definíciónak, így ezek fák. Ugyancsak fa az egy csúcsú, él nélküli gráf is, valamint az utak is (pontosabban: azok a gráfok, amik egyetlen út csúcsaiból és éleiből állnak). A definíciónak azt a feltételét, hogy F nem tartalmaz kört, röviden úgy is szokás kifejezni, hogy F *körmentes*. Ebből a tulajdonságból az is következik, hogy F egyszerű gráf; valóban, ha F tartalmazna hurokélt vagy párhuzamos éleket, akkor volna benne egy, illetve kettő hosszúságú kör is.

Az 1.8. ábra gráfjai tartalmaznak egy fokú csúcsokat, a második csúcsainak több, mint a fele ilyen. Ha egy fa egy (tetszőleges, de pozitív hosszúságú) útból áll, akkor abban az egy fokú csúcsok száma csak kettő – de az alábbi tételből ki fog derülni, hogy ennél kevesebb nem is lehetne. Az egyetlen olyan fa, ami nem tartalmaz egy fokú csúcsot az egy csúcsú, él nélküli fa (vagyis a nulla hosszúságú út).

1.22. Tétel. *Ha F legalább két csúcsú fa, akkor F tartalmaz legalább két olyan csúcsot, amiknek a foka egy.*

Bizonyítás: Válasszunk F-ben egy P leghosszabb utat – vagyis egy olyat, aminél hosszabb út már nincs F-ben. (Az tehát előfordulhat, hogy P-vel azonos hosszúságú

út még több is van F-ben, de ez a bizonyítás szempontjából érdektelen.) Mivel F legalább két csúcsú és így tartalmaz élt (különben nem volna összefüggő), ezért P pozitív hosszúságú, így két különböző végpontja van. Azt állítjuk, hogy ezeknek a foka egy.



1.10. ábra

Legyen ugyanis v a P egyik végpontja (lásd az 1.10. ábrát). Biztosan szomszédja v-nek a P mentén v-t megelőző (vagy követő) u csúcs. Tegyük fel indirekt, hogy v-nek van egy u-tól különböző x szomszédja is. Ha x nincs rajta P-n, akkor P-t megtoldhatjuk a $\{v,x\}$ éllel és az x csúccsal; mivel így egy P-nél hosszabb utat kapunk, ez P választása miatt lehetetlen. Ha viszont x rajta van P-n, akkor P-nek a v-től x-ig tartó szakasza a $\{v,x\}$ éllel együtt kört alkotna F-ben, ami szintén ellentmondás.

Mivel P két végpontjának csak egy-egy szomszédja van (amikkel csak egy-egy él mentén lehetnek szomszédosak, hiszen a körmentesség miatt F egyszerű gráf), ezért ezzel a tételt beláttuk. □

Érdemes megfigyelni, hogy a fenti bizonyításban F összefüggőségéből csak annyit használtunk ki, hogy F-nek van éle. Így igaz az az erősebb állítás is, hogy minden élt tartalmazó, körmentes gráfnak van legalább két elsőfokú csúcsa. (Az *elsőfokú*, *másodfokú*, stb. kifejezések is elterjedtek a gráfelméleti nyelvhasználatban, a jelentésük azonos azzal, hogy egy fokú, kettő fokú, stb. A fák elsőfokú csúcsait pedig – maradva a botanikai ihletettségű elnevezéseknél – *levélnek* is szokták hívni.)

Számoljuk meg az 1.8. ábra fáinak csúcsait és éleit: az elsőnek 5 csúcsa és 4 éle, a másodiknak 20 csúcsa és 19 éle van. Az utakra is nyilván teljesül, hogy eggyel több csúcsuk van, mint élük. Az alábbi tételből kiderül, hogy ez minden fára igaz.

1.23. Tétel. *Legyen G egy n csúcsú és m élű gráf. Ekkor*
(i) ha G összefüggő, akkor $m \geq n-1$;
(ii) ha G körmentes, akkor $m \leq n-1$;
(iii) ha G fa, akkor $m = n-1$.

Bizonyítás: A *(iii)* állítás nyilván közvetlen következménye *(i)*-nek és *(ii)*-nek. Az utóbbiak bizonyításához pedig képzeljük el, hogy G-t az n csúcsú üres (vagyis él nélküli) gráfból fokozatosan építjük fel úgy, hogy (tetszőleges sorrendben) egymás után adjuk hozzá G éleit és gondoljuk meg, hogy eközben hogyan változnak a gráf komponensei.

Ha már néhány élt felvettünk a gráf élhalmazába és most éppen az $e = \{u,v\}$ élt adjuk hozzá, akkor két eset lehetséges. Ha u és v közös komponensébe tartozik az aktuális gráfnak, vagyis összeköti őket egy P út, akkor bármely $x,y \in V(G)$ esetén e hozzávétele nem változtat azon, hogy x és y elérhető-e egymásból úton, így a gráf

komponensei sem változnak. Valóban, ha x és y között volt út e hozzávétele előtt, akkor nyilván utána is lesz. Megfordítva, ha e hozzávétele után van egy Q út x és y között és Q nem tartalmazza e-t, akkor persze Q az e hozzávétele előtt is létezett. Ha viszont Q tartalmazza e-t, akkor e helyére beszúrhatjuk Q-ba a P utat (hiszen annak a végpontjai azonosak e végpontjaival); a beszúrás eredménye egy x és y közötti élsorozat, de ebből az 1.11. Állítás szerint x és y közötti út létezése is következik.

Gondoljuk meg most azt az esetet, ha az újonnan felvett $e = \{u, v\}$ él végpontjai különböző komponensekben vannak: u a K_u, v a K_v komponensben. Ekkor K_u és K_v egyesül, vagyis a $V(K_u) \cup V(K_v)$ csúcshalmaz feszít egy új komponenst, a többi komponens viszont nem változik. Valóban, e hozzávétele után u-ból v-n keresztül elérhetővé válnak K_v csúcsai és K_u csúcsai is nyilván elérhetők maradnak; továbbá a $V(K_u) \cup V(K_v)$ halmazba nem tartozó csúcsok nyilván nem válnak u-ból elérhetővé. Ezért (például) u komponensét a $V(K_u) \cup V(K_v)$ halmaz feszíti. A K_u-tól és K_v-től különböző komponenseket pedig természetesen nem érinti e felvétele, hiszen nem változik az ezeknek a csúcsaiból úton elérhető csúcsok halmaza.

Azt kaptuk tehát, hogy egy új él felvétele a komponensek számát vagy nem változtatja, vagy eggyel csökkenti. Ebből a tétel *(i)* állítása azonnal következik: mivel kezdetben (amikor még nem volt éle a gráfnak) n komponens volt és végül csak egy van (mert G összefüggő), ezért valóban legalább $n-1$ él hozzávételére volt szükség.

A tétel *(ii)* állításának a bizonyításához pedig csak azt kell észrevennünk, hogy ha G körmentes, akkor a bizonyítás második bekezdésében írt eset nem fordulhat elő: az aktuálisan hozzávett $e = \{u, v\}$ él végpontjai nem tartozhatnak közös komponensbe. Valóban, különben már e hozzávétele előtt is volna út u és v között, ami e-vel együtt körré záródna. Mivel tehát minden egyes él hozzávétele eggyel csökkenti a komponensek számát, ezért ennek a folyamatnak legföljebb $n-1$ él után meg kell állnia (hiszen az n-edik él már egy összefüggő gráf két csúcsát kötné össze és így kört hozna létre). □

Megjegyezzük, hogy bár az utóbbi két tételre egymástól független bizonyításokat adtunk, az 1.22. Tétel, illetve az 1.23. Tétel *(iii)* állítása közül bármelyik viszonylag könnyen bebizonyítható a másikból.

Ha ugyanis egy F fából töröljük egy v elsőfokú csúcsát (annak az egyetlen élével együtt), akkor megint egy fát kapunk. Valóban, v törlésével a körmentesség nyilván nem sérül és könnyű megmutatni, hogy az összefüggőség sem (lásd az 1.27. Feladat megoldását). Így ha F legalább két csúcsú fa, akkor F-ből indulva és az 1.22. Tételre támaszkodva addig törölhetünk elsőfokú csúcsokat, míg végül az $n = 1$ csúcsú és $m = 0$ élű fához érkezünk. Mivel minden törlésnél n és m is eggyel csökkent, ebből $m = n-1$ valóban következik.

Ahhoz pedig, hogy az 1.23. Tétel *(iii)* állításából belássuk az 1.22. Tételt, figyeljük meg, hogy ha F-nek kettőnél kevesebb elsőfokú csúcsa lenne, akkor a csúcsainak a fokszámösszege legalább $2n-1$ volna. Valóban, $n-1$ darab legalább 2-es és egy darab legalább 1-es fokszám összege tényleg legalább ennyi (és nulla fokú csúcs létezése kizárt, mert $n \geq 2$ és F összefüggő). Ebből az 1.3. Tétel miatt $m \geq n$ következne, ellentmondás.

1.24. Feladat. Egy F fa csúcsainak fokszámai között pontosan kétféle érték fordul elő és mindkettő pontosan ugyanannyiszor. Adjuk meg F-et.

Megoldás: Mivel a kétféle fokszám ugyanannyiszor fordul elő, ezért F csúcsainak száma páros. Legyen ezért a csúcsok száma $n = 2k$. Az 1.22. Tétel szerint az egyik előforduló fokszám az 1, a másikat jelölje d.

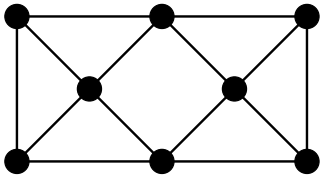
Ekkor F-ben a fokszámok összege $k \cdot d + k \cdot 1 = k(d+1)$. Az 1.3. Tétel szerint ez F élszámának kétszerese: $2m = k(d+1)$, ahol m jelöli F éleinek számát. Másrészt az 1.23. Tétel miatt $m = n-1 = 2k-1$. Ezek összevetéséből: $k(d+1) = 4k-2$. Ezt az egyenletet d-re rendezve: $d = 3 - \frac{2}{k}$.

Mivel d és k is nyilván pozitív egész, ezért csak a $k = 1, 2$ értékek jönnek szóba. Azonban $k = 1$ esetén $d = 3-2 = 1$ adódna, ami lehetetlen, mert így F-ben nem volna kétféle fokszám. Marad a $k = 2$ eset, amiből $d = 3-1 = 2$ adódik.

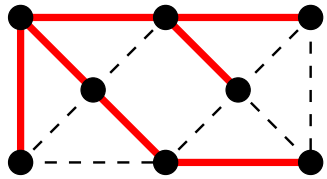
Ezek szerint F egy $n = 2k = 4$ csúcsú fa, amiben két darab elsőfokú és két darab másodfokú csúcs van. Így F csak egy három hosszúságú út lehet. □

1.5.1. Feszítőfa

Tegyük fel, hogy az 1.11a ábra gráfja valamilyen tervezés alatt álló hálózat (például egy számítógép-hálózat vagy úthálózat) csomópontjait és a köztük létrehozható kapcsolatokat ábrázolja. A feladat annak az eldöntése, hogy a gráf melyik éleit vegyük be a készülő hálózatba úgy, hogy bármely két csomópont között legyen összeköttetés a hálózatban (nem feltétlen közvetlenül, hanem esetleg további csomópontokon keresztül) és a hálózatnak (például gazdaságossági megfontolások miatt) a lehető legkevesebb éle legyen.



1.11a ábra



1.11b ábra

A feladat egyik lehetséges megoldását az 1.11b ábra hét piros éle mutatja. Ezek (a gráf nyolc csúcsával együtt) fát alkotnak, így megfelelnek a feladat fenti leírásából fakadó összefüggőségi feltételnek. Továbbá az 1.23. Tétel *(i)* állítása szerint egy nyolc csúcsú, összefüggő gráfnak nem is lehetne hétnél kevesebb éle, így a piros élek választása ennek a feltételnek is megfelel. Természetesen számos más jó megoldása is volna ugyanennek a feladatnak: minden olyan élhalmaz alkalmas volna, ami a gráf összes csúcsával együtt fát alkot. Az ilyen fáknak számtalan alkalmazása van a gráfelméletben, ezeknek ad nevet az alábbi definíció.

1.25. Definíció. *A G gráfnak az F részgráfja* feszítőfa*, ha F fa és $V(G) = V(F)$ (vagyis F a G összes csúcsát tartalmazza).*

A *feszítőfa* és a *feszített részgráf* kifejezéseket nem szabad összekeverni, mert a hasonlóságuk ellenére egészen más – sőt, részben ellentétes – jelentéssel bírnak. Egy feszítőfa a gráf *éleinek* egy részhalmazával adható meg, hiszen a gráf minden csúcsát tartalmaznia kell. Ezzel szemben egy feszített részgráfot (az 1.5. Definíció szerint) a gráf *csúcsainak* egy részhalmaza határoz meg, de ennek ismeretében az élei már adottak: a kiválasztott csúcsok között futó összes élt tartalmazza.

Az 1.11a ábra gráfjának a fentiek szerint van feszítőfája: az 1.11b ábrán látható egy ilyen (a sok közül). Azonban ez nem minden gráfra igaz: ha egy gráf nem összefüggő, akkor nyilván feszítőfája sem lehet. Az alábbi tételből kiderül, hogy más akadálya viszont már nem lehet feszítőfa létezésének.

1.26. Tétel. *A G gráfnak akkor és csak akkor létezik feszítőfája, ha G összefüggő.*

Bizonyítás: Ha G-nek létezik egy F feszítőfája, akkor (mivel F fa és így összefüggő) bármely $x, y \in V(G) = V(F)$ csúcsok között létezik F-beli út, ami persze G-beli is. Így G valóban összefüggő.

A fordított irányhoz legyen G egy tetszőleges, összefüggő gráf. Annak megmutatásához, hogy G-ben van feszítőfa, az 1.23. Tétel bizonyításának a gondolatmenetét hívjuk segítségül. Ott megmutattuk, hogy ha egy H gráfnak K_u és K_v két különböző komponense és H-hoz hozzáadunk egy új $e = \{u, v\}$ élt, amire $u \in V(K_u)$ és $v \in V(K_v)$, akkor ezzel K_u és K_v egyesül: az új H' gráf egyik komponensét a $V(K_u) \cup V(K_v)$ csúcshalmaz feszíti, a többi komponenst pedig e hozzávétele nem érinti. Továbbá az is nyilvánvaló, hogy ha H nem tartalmazott kört, akkor H' sem tartalmazhat: valóban, ha H'-ben volna egy C kör, akkor ez (H körmentessége miatt) tartalmazná e-t; így C-ből e elhagyásával egy H-beli utat kapnánk u és v között, ami lehetetlen, mert u és v H különböző komponenseibe tartoznak.

A fentiekre alapozva az üres halmaztól indulva fokozatosan építjük fel a keresett feszítőfa élhalmazát, mindig egy olyan éllel bővítve azt, aminek a végpontjai az aktuális gráf különböző komponenseibe tartoznak. A fentiek szerint így sosem hozunk létre kört és a komponensek száma folyamatosan csökken. Így amint ez a szám eléri az egyet, valóban összefüggő és körmentes gráfot, vagyis feszítőfát kapunk.

Azt kell ezért csak megmutatnunk, hogy ha az aktuális H gráfnak még legalább két komponense van (vagyis nem összefüggő), akkor mindig található G-ben egy olyan él, aminek a végpontjai H különböző komponenseibe tartoznak. Legyenek ezért K_x és K_y a H különböző komponensei és $x \in V(K_x)$, $y \in V(K_y)$ tetszőleges csúcsok. Mivel G összefüggő, ezért létezik G-ben egy P út x-ből y-ba. Induljunk el P mentén x-ből y-ba és álljunk meg az első olyan csúcsnál, ami nem $V(K_x)$-beli. Ilyennek kell léteznie, mert $y \notin V(K_x)$; jelölje ezt v, a v-t megelőző csúcsot P-n pedig u. Ekkor az $e = \{u, v\}$ él megfelel a feltételeknek: $e \in E(G)$ (hiszen e a G-beli P út éle) és $u \in V(K_x)$, $v \notin V(K_x)$ miatt e végpontjai valóban H különböző komponenseibe tartoznak. □

A fenti bizonyításból egyben egy módszer is kiolvasható feszítőfa keresésére egy G összefüggő gráfban: az üres halmaztól indulva mindig olyan éllel bővítjük a készülő feszítőfa élhalmazát, aminek a végpontjai a korábban már kiválasztott élek (és a gráf összes csúcsa) által alkotott részgráf különböző komponenseibe tartoznak. A bizonyítás utolsó bekezdéséből az is kiderült, hogy ha K az aktuális gráf egy tetszőleges komponense, akkor mindig találunk olyan $e = \{u, v\}$ élt $E(G)$-ben, amire $u \in V(K)$ és $v \notin V(K)$. Ebből következik, hogy a feszítőfa élhalmazának fokozatos építése megvalósítható úgy is, hogy minden közbülső pillanatban csak egyetlen olyan komponens van, aminek egynél több csúcsa van, az összes többi komponens egy (izolált) csúcsból áll; valóban, ha K jelöli az egyetlen, egynél több csúcsú komponenst, akkor a fentiek szerinti $e = \{u, v\}$ kiválasztása után (amire tehát $u \in V(K)$ és $v \notin V(K)$) K bővül v-vel és e-vel, de a többi komponens továbbra is egy csúcsú. Ha tehát a bizonyításban leírt módszernek eszerint a speciális esete szerint járunk el, akkor az aktuális gráf mindig egy (abban) elsőfokú csúccsal bővül.

Bemutatunk az 1.26. Tételre egy másik bizonyítást is, ami a fent leírttal ellentétes stratégiát követ: nem az üres halmaz felől indulva, a körmentesség fenntartásával építi fel a feszítőfát, hanem a gráf teljes élhalmaza felől indulva, az összefüggőséget fenntartó ismételt törlésekkel éri el azt. Az alábbi bizonyítás még valamivel rövidebb is a fentinél; a fenti előnye viszont, hogy később több alkalmazásban is az abban látott módszerrel fogunk feszítőfát nyerni és ezekben fontos lesz tudnunk, hogy valóban feszítőfát kaptunk.

Az 1.26. Tétel egy alternatív bizonyítása: Először azt mutatjuk meg, hogy ha egy G összefüggő gráf egy tetszőleges C körének bármelyik $e = \{u, v\}$ élét töröljük, akkor a kapott G' gráf szintén összefüggő. Legyen ugyanis x és y két tetszőleges csúcs G-ben és válasszunk ezek között egy P utat (ami G összefüggősége miatt létezik). Ha P nem tartalmazza e-t, akkor persze P G'-ben is út x és y között. Ha viszont P átmegy e-n, akkor szúrjuk be P-be e helyére azt az u-ból v-be vezető utat, amit C-ből e törlésével kapunk. Ezzel egy x és y közötti élsorozatot kapunk, de ebből az 1.11. Állítás szerint x és y közötti út létezése is következik. Mivel tehát G'-ben bármely két csúcs között létezik út, ezért az valóban összefüggő.

Rátérve a tétel bizonyítására, elég azt megmutatnunk, hogy minden G összefüggő gráfnak van feszítőfája (mert a másik irány a fentiek szerint nagyon könnyen adódik). Ha G körmentes, akkor fa is, így van feszítőfája: saját maga. Ha nem, akkor töröljük egy tetszőleges körének bármelyik élét, a kapott gráfot jelölje G_1; ez a fentiek szerint összefüggő. Ha G_1 már körmentes, akkor az feszítőfája G-nek. Ha nem, akkor ismét törölhető egy körének egy éle és a kapott G_2 gráf megint összefüggő lesz. Ezt az eljárást folytatva véges sok lépés után körmentes gráfot kell kapnunk (hiszen az élszám fokozatosan csökken). Mivel az eljárás során a gráf összefüggősége végig fennmarad (és a csúcshalmaz sem változik), ezért annak a leállása után valóban feszítőfát kapunk. □

1.27. Feladat. Létezik-e olyan (legalább két csúcsú) összefüggő gráf, aminek bármelyik csúcsát törölve (az éleivel együtt) a kapott gráf már nem összefüggő?

Megoldás: Megmutatjuk, hogy ilyen gráf nem létezik: minden összefüggő G gráfból törölhető egy alkalmasan választott v csúcs úgy, hogy ezzel összefüggő gráfot

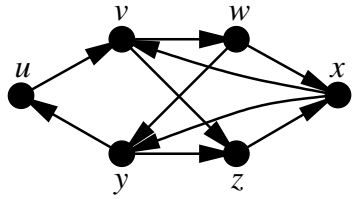
kapjunk. Vegyünk ugyanis az 1.26. Tétel szerint egy F feszítőfát G-ben, abban pedig az 1.22. Tétel szerint egy v elsőfokú csúcsot. Ekkor a v törlése után maradó G' gráf valóban összefüggő, mert bármely x és y csúcsai között még az F-ből megmaradó gráfban is létezik út. Valóban, az x és y közötti, F-beli P utat (ami F összefüggősége miatt létezett) nem érinti v törlése, hiszen v elsőfokú, így P nem haladhatott át rajta. □

1.6. Irányított gráfok

Gráfokkal – a kifejezés eddig használt értelmében – bizonyos elemek közötti szimmetrikus viszonyokat modellezhetünk. Gyakoriak azonban az olyan alkalmazások is, amikben az elemek közötti kapcsolatok nem szimmetrikusak. Például egy társaság tagjai között lehetnek nem kölcsönös ismeretségek (ha mondjuk jelen van egy közismert ember), egy térképen lehetnek egyirányú útszakaszok és egy számítógéphálózatban is előfordulhat, hogy két csomópont között csak egyirányú kommunikáció lehetséges. Ezeket a helyzeteket *irányított gráfokkal* modellezhetjük.

Egy irányított gráf szintén csúcsokból és élekből áll, azonban minden élnek iránya is van: nem elég információ, hogy az e él az u és a v csúcsok között halad, tudnunk kell azt is, hogy e az u-ból a v-be, vagy a v-ből az u-ba mutat. Egy G irányított gráf csúcshalmazát és élhalmazát szintén $V(G)$, illetve $E(G)$ jelöli, az $e = (u,v)$ jelölés pedig azt fejezi ki, hogy az $e \in E(G)$ él az $u \in V(G)$ csúcsból a $v \in V(G)$ csúcsba vezet. (Természetesen nincs akadálya annak sem, hogy egy irányított gráf az (u,v) és a (v,u) éleket is tartalmazza.)

Az irányított gráfok ábrázolásakor az irányított éleket egyszerűen nyilakkal szokás jelölni; így például az 1.12. ábrán egy irányított gráf látható.

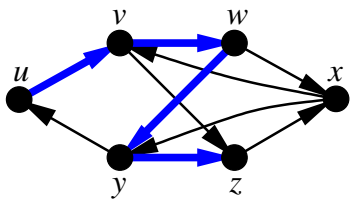


1.12. ábra

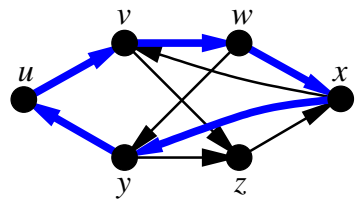
Ha nem tesszük ki az „irányított” jelzőt, akkor a gráf kifejezést továbbra is a korábban bevezetett értelemben használjuk (amiben tehát az éleknek nincs iránya). Ahol ez a szövegkörnyezetből mégsem derül ki egyértelműen, ott az *irányítatlan gráf* kifejezéssel utalunk arra, hogy nem irányított gráfról van szó. Az irányított gráf fogalmára is lehetne az 1.1-hez hasonló, precíz definíciót alkotni, de ettől ebben a jegyzetben eltekintünk. Továbbá a fejezetben eddig bevezetett fogalmak túlnyomó részének meg lehetne alkotni az irányított gráfokra vonatkozó, analóg változatát is. Az esetek egy részében ez szinte magától értetődő módosításokat jelentene

(mint például a részgráf 1.4. Definíciója, vagy az izomorfia 1.8. Definíciója esetében), más fogalmaknál azonban komolyabb nehézségekbe ütköznénk. Így például az összefüggőség, illetve még inkább az összefüggő komponens fogalmával analóg, irányított gráfokra vonatkozó fogalmak és tételek már komplikáltabbak az irányítatlan eseteknél (mégpedig azért, mert könnyen előfordulhat, hogy a v csúcs elérhető u-ból, de ez fordítva nem teljesül); ezekkel ebben a jegyzetben nem foglalkozunk.

Szükségünk lesz viszont az *irányított út*, illetve az *irányított kör* fogalmára, amik az út, illetve a kör korábban (az 1.10., illetve az 1.20. Definícióban) bevezetett fogalmainak az irányított megfelelői. Ez a két fogalom pontosan megfelel az irányítatlan eseteknek, az egyetlen különbség, hogy az egymás után következő csúcsok közül mindig az utóbbiba vezet irányított él az előtte állóból. Az irányított út tehát azt jelenti, hogy a gráf egy csúcsából indulva az irányított élek mentén (és természetesen az irányítások figyelembe vételével, tehát sosem a „nyíllal szemben") haladunk úgy, hogy minden csúcsot legföljebb egyszer érintünk; egy irányított kör pedig csak annyiban különbözik ettől, hogy az utolsó csúcsa azonos az elsővel, de ettől eltekintve szintén nem érint egy csúcsot kétszer. Az 1.13a ábrán egy irányított út, az 1.13b ábrán pedig egy irányított kör látható az 1.12. ábra irányított gráfjában.



1.13a ábra



1.13b ábra

1.28. Feladat. Legkevesebb hány éle kell legyen egy 100 csúcsú G irányított gráfnak, ha

a) van olyan csúcsa, amiből az összes többi csúcs elérhető irányított úton;

b) minden csúcsából az összes többi csúcs elérhető irányított úton?

Megoldás: a) 99 élű gráfot könnyen lehet készíteni ezzel a tulajdonsággal: például ha G egyetlen, 99 hosszúságú irányított útból áll. Megmutatjuk, hogy 99-nél kevesebb éle nem lehet G-nek. Legyen ugyanis H az az irányítatlan gráf, amit G-ből kapunk az élek irányításának a figyelmen kívül hagyásával. Ha G-nek a v csúcsából az összes többi elérhető irányított úton, akkor nyilván H-ban is elérhető v-ből minden csúcs irányítatlan úton. Ezért H összefüggő (hiszen v komponense az összes csúcsot tartalmazza). Így az 1.23. Tétel *(i)* állítása miatt H legalább 99 élű, ezért G-nek is legalább ennyi éle van.

b) Ha G egyetlen, 100 hosszúságú irányított körből áll, akkor teljesül rá ez a tulajdonság. Ez mutatja, hogy van ilyen tulajdonságú 100 élű gráf; megmutatjuk, hogy ennél kevesebb élű nincs. Ugyanis egy ilyen G-nek minden v csúcsából legalább egy irányított élnek ki kell indulnia (különben v-ből nem lehetne más csúcsot

elérni irányított úton). Így G minden csúcsához tartozik egy onnan kilépő él – és persze minden csúcshoz másik (hiszen egy irányított él csak egy csúcsból léphet ki). Tehát G valóban legalább 100 élű. □

1.7. Gráfelméleti algoritmusok hatékonysága

A gráfelmélet gyakorlati alkalmazásai többségükben algoritmikus feladatok. Ezekben tehát a bemenet egy gráfból (és esetleg további adatokból) áll, amiben valamilyen, az adott alkalmazás szempontjából releváns kérdésre kell válaszolni – ami jelentheti valaminek a kiszámítását, egy részgráf meghatározását, stb. Például egy ilyen, számtalan alkalmazással bíró feladat egy adott gráf két adott csúcsa között egy legrövidebb út meghatározása. (Ennek a feladatnak több változata is van, így ebben a jegyzetben is több különböző algoritmust fogunk megismerni a megoldására.)

Természetesen a gráfelméleti algoritmusok esetében is nagyon fontos kérdés a futásidő, illetve a hatékonyság; nagyon könnyen előfordulhat, hogy egy gráfelméleti problémára készített programkód ugyan elvileg helyes eredményt ad, de az alkalmazásban felmerülő méretű bemenetekre még a legerősebb hardvereken is évezredekig, vagy akár évmilliókig futna. Korábbi tanulmányainkból tudjuk, hogy egy algoritmust akkor szokás hatékonynak tekinteni, ha az *polinomiális futásidejű*, vagyis ha vannak olyan c és k fix konstansok, hogy az algoritmus minden s méretű (azaz s biten tárolható) inputra legföljebb $c \cdot s^k$ lépés után megáll. Ahhoz, hogy ezt az alapvető fogalmat gráfelméleti algoritmusokra is alkalmazni tudjuk, előtte foglalkoznunk kell a gráfok tárolásának a kérdésével – hiszen enélkül nincs értelme a bemenet méretéről beszélni, ha a bemenet egy gráfból (is) áll.

Előrebocsátjuk, hogy gráfok tárolására számtalan adatszerkezet (vagyis „tárolási mód") ismert és a gráfelméleti algoritmusok implementációinak a hatékonyságát messze nem elhanyagolható mértékben befolyásolhatja, hogy ezek közül az adott alkalmazáshoz mennyire jól illeszkedőt választunk. Bár az adatszerkezet megválasztása ritkán (vagy soha) nem befolyásolja, hogy az algoritmus polinomiális futásidejű-e vagy sem (mert az egyes tárolási módok polinomiális algoritmussal egymásba konvertálhatók), de az könnyen előfordulhat, hogy például pár száz helyett pár ezer vagy tízezer csúcsú gráfokra is elfogadható futásidőt tudunk elérni pusztán egy megfelelő adatszerkezet segítségével. Ezzel a kérdéssel ez a jegyzet nem foglalkozik bővebben, csak a két legalapvetőbb gráfelméleti adatszerkezetet említjük meg.

1.7.1. Szomszédsági mátrix

Talán a legkézenfekvőbb gondolat egy n csúcsú gráfot egy $n \times n$-es mátrixban tárolni, amiben a sorok és az oszlopok is a csúcsoknak felelnek meg és két csúcs egymáshoz való viszonyát (vagyis a köztük futó élek számát) a mátrixban a megfelelő sor és oszlop kereszteződésében álló elem rögzíti.

1.29. Definíció. *Legyen G egy tetszőleges, irányított vagy irányítatlan gráf és legyen $V(G) = \{v_1, v_2, \ldots, v_n\}$. Ekkor G-nek a* szomszédsági mátrixa *az az $n \times n$-es A mátrix, aminek az i-edik sorának és a j-edik oszlopának a kereszteződésében álló $a_{i,j}$ eleme a v_i-ből v_j-be menő G-beli élek száma minden $1 \leq i, j \leq n$ esetén. Ennek a jele: $A(G) = A$.*

Ha G irányítatlan gráf, akkor nyilván $a_{i,j} = a_{j,i}$ minden $1 \leq i, j \leq n$ esetén (vagyis $A(G)$ a v_i és v_j között futó élek számát két helyen is tárolja, ha $v_i \neq v_j$). A szomszédsági mátrix főátlójában álló elemek az egyes csúcsokra illeszkedő hurokélek számát mutatják (hiszen $i = j$ esetén így értendő a v_i-ből v_i-be menő élek száma). Így ha G egyszerű, irányítatlan gráf, akkor $A(G)$ egy bináris mátrix (vagyis minden eleme 0 vagy 1) és a főátlójának minden eleme 0. A definícióból látszik, hogy $A(G)$ nem csak a G gráftól függ, hanem a csúcsainak a számozásától is: ha $V(G)$ elemeit máshogyan számozzuk meg v_1-től v_n-ig, akkor $A(G)$ is megváltozik – mégpedig a sorai és az oszlopai az átszámozásnak megfelelően átrendeződnek. Ez azonban az alkalmazásokban nem okoz problémát, hiszen a csúcsok számozásától függetlenül $A(G)$ ugyanazt az információt hordozza.

Az 1.14a ábra az 1.2. ábra irányítatlan gráfjának, az 1.14b ábra pedig az 1.12. ábra irányított gráfjának a szomszédsági mátrixát mutatja; a csúcsokat mindkét esetben ábécé szerinti növekvő sorrendben számoztuk.

$$\begin{pmatrix} 0 & 1 & 0 & 3 & 0 \\ 1 & 1 & 1 & 0 & 2 \\ 0 & 1 & 2 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{pmatrix}$$

1.14a ábra

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

1.14b ábra

A szomszédsági mátrix nem csak egyszerű adatszerkezet gráfok tárolására, hanem érdekes elméleti alkalmazásai is vannak. Nem nehéz megmutatni például, hogy minden $k \geq 1$ egészre az $A(G)^k$ mátrix i-edik sorának és j-edik oszlopának a kereszteződésében álló elem a v_i-ből v_j-be vezető, pontosan k hosszúságú élsorozatok számával egyenlő (irányított és irányítatlan gráfokban is). Ennél jóval meglepőbb, hogy az irányítatlan, egyszerű gráfokról sok hasznos információt hordoznak a szomszédsági mátrixuknak a sajátértékei. Ezeknek a halmazát a gráf *spektrumának* nevezik és a gráfelmélet egy külön fejezete foglalkozik velük.

1.7.2. Szomszédsági lista

Bár a szomszédsági mátrix rendkívül egyszerű fogalom, sok hátránya is lehet annak, ha egy gráfot így tárolunk. Ha például egy navigációs szoftverhez használt G gráf egy térképet reprezentál, akkor könnyen elképzelhető, hogy a csúcsok száma sok ezer vagy tízezer, de az egy csúcsra illeszkedő élek száma általában nagyon kevés.

(Még Párizs híres terébe, a Place de l'Étoile-ba is „csak" tizenkét sugárút torkollik.) Így $A(G)$ elemeinek a túlnyomó többsége nulla, vagyis nem „éri meg" G-t így tárolni – és nem csak azért, mert az értékes információhoz képest hatalmas helyet foglalna a memóriában: azért sem, mert egy adott v csúcs szomszédainak a felderítéséhez végig kellene pásztázni $A(G)$ teljes, v-nek megfelelő sorát (vagy oszlopát), ami nagyon lelassítaná a program működését.

Ilyen, viszonylag „ritka" gráfok tárolásához (vagyis amikben az élek száma nem túl nagy a csúcsokéhoz képest) jó alternatívát jelenthet a *szomszédsági lista* (vagy másik nevén *éllista*). Ennek az alapgondolata szintén nagyon egyszerű: minden v csúcshoz egy listában eltároljuk v szomszédait (tetszőleges sorrendben); irányított gráf esetén pedig azokat a csúcsokat, amikbe v-ből vezet irányított él. Ha pedig v-ből w-be több, például k darab (irányított vagy irányítatlan) él is vezet, akkor a v-hez tartozó listában w-t k-szor is feltüntetjük. Itt *lista* alatt egy olyan, a legtöbb modern programnyelv által támogatott adattípust értünk, ami lehetővé teszi elemeknek egy sorozatban való tárolását úgy, hogy minden elemtől egy mutató segítségével jutunk a következőbe. A szomszédsági lista, mint gráfok tárolására szolgáló adatszerkezet tehát nem egyetlen ilyen listából áll, hanem annyiból, mint a gráf csúcsainak a száma.

Az 1.15 ábrán ismét az 1.2. ábra irányítatlan, illetve az 1.12. ábra irányított gráfjának a leírása látható, de most szomszédsági listákkal.

u: $v \to x \to x \to x$
v: $y \to y \to v \to w \to u$
w: $x \to w \to w \to v$
x: $u \to u \to u \to w$
y: $v \to v$

1.15a ábra

u: v
v: $w \to z$
w: $y \to x$
x: $v \to y$
y: $z \to u$
z: x

1.15b ábra

Hasonlóan a szomszédsági mátrixhoz, irányítatlan gráfok esetében minden e él (a hurokéleket kivéve) kétszer is hozzájárul a szomszédsági listához: e mindkét végpontjának a listájában megjelenik a másik. Irányított gráfok esetében pedig minden él egyszer járul hozzá a szomszédsági listához: a kezdőpontjának a listájában jelenik meg a végpontja. Ezért egy m élű irányított, illetve (hurokélmentes) irányítatlan gráf szomszédsági listájában a csúcsokhoz tartozó listák összhossza m, illetve $2m$. (Irányítatlan esetben ez tartalmilag azonos az 1.3. Tétel állításával, hiszen egy v csúcs listájának a hossza $d(v)$, a v fokszáma. Az egyetlen különbség, hogy a hurokélek a v listájának a hosszát csak eggyel növelik, míg a fokszámát kettővel.)

Fentebb már említettük, hogy a szomszédsági lista előnye a szomszédsági mátrixhoz képest például az, hogy gyorsabban kiolvashatók belőle egy adott csúcs szomszédai. Ugyanakkor hátránya is van: ha arra vagyunk kíváncsiak, hogy két adott csúcs szomszédos-e, akkor a szomszédsági mátrixból ezt egyetlen elem kiolvasásával meg tudjuk válaszolni, míg a szomszédsági listából csak úgy, ha az egyik csúcs listáján végigmegyünk és megnézzük, hogy a másik szerepel-e rajta. Ez az egyszerű példa is mutatja azt a fentebb már említett tényt, hogy a gráf tárolásához

használt adatszerkezet helyes, az alkalmazáshoz illeszkedő megválasztása nagyban befolyásolhatja az implementáció hatékonyságát. Szintén említettük már, hogy a szomszédsági mátrixon és a szomszédsági listán kívül számos további, ezeknél akár jóval komplikáltabb és ravaszabb adatszerkezetet is ismer a számítástudomány, amik egyes algoritmusok hatékony implementálásában fontos szerepet játszhatnak.

1.7.3. Súlyozott gráfok tárolása

Számos gráfelméleti algoritmus bemenete nem csak egy gráfból áll, hanem ahhoz tartozó számadatok is szerepelnek benne. Gyakori például, hogy a gráf éleihez tartozó számok vannak a bemenetben (amiknek a jelentése és szerepe az adott alkalmazástól függ); ilyenkor az e élhez tartozó számot elterjedt szóhasználat szerint szokás az él *súlyának* nevezni és $w(e)$-vel jelölni (az angol *weight* szó kezdőbetűjéből). Egyszerű példa erre a korábban már említett legrövidebb út feladat: számtalan alkalmazásban (például navigációs szoftverekben) egy P út hosszát nem egyszerűen a P-t alkotó élek száma fejezi ki, hanem ezeknek az éleknek az összsúlya (ahol tehát az élsúlyok a bemenet részét képezik és leírhatják például a megfelelő útszakasz hosszát, vagy a megtételéhez szükséges időt).

A szomszédsági mátrix és a szomszédsági lista is könnyen bővíthető az élsúlyok tárolásával. Szomszédsági mátrix esetén például megtehető, hogy ha a tárolt G gráf egyszerű és v_i és v_j szomszédosak G-ben, akkor $a_{i,j}$ értéke nem 1, hanem a v_i és v_j közötti e él $w(e)$ súlya; ebben az esetben viszont egy speciális, erre a célra fenntartott szimbólummal kell jelölni a mátrixban, ha v_i és v_j nem szomszédosak (hiszen a 0 elem egy v_i és v_j között futó, 0 súlyú él létére utalna). Szomszédsági lista esetén pedig az $e = \{u,v\}$ irányítatlan, vagy az $e = (u,v)$ irányított él $w(e)$ súlyát v-vel együtt lehet tárolni az u csúcs listájában. (Ilyenkor tehát a csúcsokhoz tartozó listák nem egyszerűen csúcsokat tartalmaznak, hanem olyan párokat, amik egy csúcsból és egy számból állnak.)

1.7.4. Gráfelméleti algoritmusok lépésszáma

Térjünk vissza a szakasz elején felvetett kérdésre: hogyan mérjük egy gráfelméleti algoritmus lépésszámát? A kérdés megválaszolását több olyan körülmény nehezíti, amik elméleti szempontból relevánsak ugyan, de a gyakorlatban szerencsére nem okoznak problémát. Az alábbiakban rávilágítunk ezekre a nehézségekre – de azzal a kimondott céllal, hogy meggyőződhessünk róla, hogy a jegyzet hátralévő részében (és általában gráfelméleti algoritmusok tervezésekor) nem kell foglalkozunk velük.

Az első ilyen nehézség a bemenet méretével, vagyis a tárolásához szükséges bitek számával kapcsolatos. Ha például szomszédsági listával tárolunk egy n csúcsú és m élű (egyelőre élsúlyok nélküli) gráfot, akkor mi lesz a bemenet mérete? Fentebb említettük, hogy a listák összhossza irányított, illetve irányítatlan gráf esetén m, illetve $2m$. A listák elemein kívül tárolni kell még, hogy azok melyik csúcshoz tartoznak (az 1.15 ábra bal szélső oszlopainak megfelelően), így ez összesen $m+n$, vagy $2m+n$ csúcs azonosítójának a tárolását jelenti. Felmerül azonban a kérdés: hány biten tárolható egy csúcs azonosítója? Ha ezeket k biten tároljuk, akkor $2^k \geq n$

kell teljesüljön, mert a k hosszú bitsorozatok száma 2^k (és minden csúcshoz különböző, azt azonosító bitsorozat kell tartozzon). Ebből $k \geq \log_2 n$ adódik, így teljes precizitást feltételezve azt kellene mondanunk, hogy a bemenet mérete – vagyis a szomszédsági lista tárolásához szükséges bitek száma – $c \cdot (m+n) \cdot \log n$ valamilyen c konstansra. Ez azonban egyrészt rendkívül kényelmetlenné tenné az algoritmusok lépésszámának a mérését a bemenet méretéhez viszonyítva, másrészt az így adódó bonyodalmak a gyakorlat szempontjából érdektelenek és fölöslegesek volnának. Ugyanis az alkalmazásokban felmerülő gráfok csúcsszáma jellemzően legföljebb ezres vagy tízezres nagyságrendű, ritka az ennél nagyobb n; de még ha tízmillió csúcsot feltételezünk, akkor is $\log_2 n \leq 24$, vagyis egy csúcs azonosítója kényelmesen eltárolható három bájton. Mivel ez a mai számítógépek kapacitását figyelembe véve elhanyagolható, nyugodtan tekinthetjük úgy, hogy egy csúcs azonosítója mindig eltárolható egy előre rögzített, konstans méretű memória helyen. Ennek megfelelően mondhatjuk azt is, hogy szomszédsági listával tárolva a gráfot a bemenet mérete $c \cdot (m+n)$ valamilyen c konstansra.

Szomszédsági mátrixszal való tárolást feltételezve még elméleti bonyodalomba sem ütközünk – legalábbis akkor, ha a gráf egyszerű: egy $n \times n$-es, bináris mátrix tárolása nyilván n^2 biten lehetséges. Ha azonban a gráf nem feltétlen egyszerű, akkor itt is fellépnek zavaró, elméleti kérdéseket felvető körülmények: elvileg még egy egyetlen csúcsú gráf is tartalmazhat olyan csillagászati mennyiségű hurokélt, hogy a szomszédsági mátrixának az egyetlen eleméhez, vagyis a hurokélek számának a tárolásához szükséges bitek száma óriási. Ezzel kapcsolatban hasonlót mondhatunk, mint az előző bekezdésben: bár a gyakorlati alkalmazásokban felmerülő gráfok élszáma nagyságrendekkel több lehet, mint a csúcsszámuk, de azt bátran feltételezhetjük, hogy a két csúcs között futó párhuzamos élek száma, illetve az egy csúcsra illeszkedő hurokélek száma észszerű korlátok alatt marad. Így az ezeknek a számoknak a leírásához szükséges bitek száma csekélynek, egy fix konstanssal felülről becsülhetőnek tekinthető. Ezt elfogadva pedig mondhatjuk, hogy a bemenet mérete szomszédsági mátrixot feltételezve $c \cdot n^2$ valamilyen c konstansra.

A következő elméleti nehézség a súlyozott gráfokkal, illetve a bemenet részét képező számok tárolásával és az ezeken végzett alapműveletekkel kapcsolatos. Képzeljük el például, hogy a fentebb már említett legrövidebb út feladatot az 1.16. ábra pofonegyszerű gráfján kell megoldanunk: u-ból v-be vezető legrövidebb utat keresünk, ahol x és y a megfelelő élek súlyát (vagyis hosszát) jelölik. Az eredmény nyilván $x+y$ lesz – ami azt mutatja, hogy a feladatot megoldó, tetszőleges algoritmusnak képesnek kell lenni a bemenet részét képező x és y számok összeadására. Korábbi tanulmányainkban, számelméleti algoritmusok kapcsán megtanultuk, hogy az N (természetes) szám tárolásához szükséges bitek száma $\lfloor \log_2 N \rfloor + 1$, vagyis N ennyivel járul hozzá a bemenet méretéhez; továbbá, hogy két szám összeadása lineáris futásidejű algoritmussal (az „írásbeli” összeadással) lehetséges. Mindezekből annak kellene következnie, hogy a legrövidebb út feladatot megoldó, tetszőleges algoritmus futásideje még $n = 3$ csúcs és $m = 2$ él esetén is tetszőlegesen nagy lehet – hiszen az speciális esetként tartalmazza az összeadás feladatát.

A valóság ezzel szemben az, hogy a gráfelméleti alkalmazásokban a bemenet ré-

1.16. ábra

szét képező számokat másképp tárolják és rajtuk az alapműveleteket is máshogyan végzik el, mint a korábban tanult számelméleti algoritmusok esetében. A számelméleti algoritmusok hatalmas, több száz vagy ezer számjegyű számokkal dolgoznak és az eredményt tökéletes pontossággal, számjegyek sorozataként adják meg. Ezzel szemben a gráfelméleti algoritmusok törtekkel és negatív számokkal is dolgoznak, de minden – akár bemenetként kapott, akár kiszámított – számadatot egy előre rögzített, konstans méretű memóriahelyen tárolnak. Ez persze azzal is jár, hogy a tárolt számokat szükség esetén kerekítik, csak egy bizonyos pontossággal veszik figyelembe. (A számítástechnikában elterjedt megoldás erre a *lebegőpontos számábrázolás*.) Bár ez a megközelítés nyilván veszteséggel járhat, de nagy előnye, hogy a számokon végzett alapműveletek mind gyorsan, egy fix konstanssal felülről becsülhető időben elvégezhetők. Így a gráfelméleti algoritmusok lépésszámának a becslésekor nem kell az alapműveletek elvégzésének az idejére külön odafigyelnünk; csak az a fontos (az viszont nagyon), hogy az algoritmus a működése során hányszor végez a tárolt adatokon valamilyen alapműveletet.

A fentiekből az is következik, hogy az élsúlyozott gráfok tárolására is érvényes, hogy a bemenet mérete szomszédsági lista, illetve szomszédsági mátrix esetén $c \cdot (m+n)$, illetve $c \cdot n^2$ (valamilyen c konstansra). Valóban: a szomszédsági listában a csúcsokkal együtt tárolt élsúlyok miatt egy él hozzájárulása a bemenet méretéhez ugyan megnő, de továbbra is egy (másik) rögzített konstans alatt marad; így (attól függően, hogy a gráf irányított-e) továbbra is $2m+n$ vagy $m+n$ darab, fix méretű azonosítót kell tárolnunk (ahol az azonosító most a megfelelő csúcsba vezető él súlyát is tartalmazza). Hasonlóan, az $n \times n$-es szomszédsági mátrixban n^2 darab értéket (vagy az él hiányára utaló szimbólumot) tárolunk, de mindegyik konstans számú bitet igényel, így ezek összesen továbbra is $c \cdot n^2$ bitet jelentenek.

Mikor lesz tehát egy gráfelméleti algoritmus polinomiális futásidejű? Annak érdekében, hogy a különböző tárolási módokat közös keretben tudjuk kezelni, rögzítsük le, hogy a fentiek szerint a bemenet s mérete legföljebb $c_0 \cdot (m+n)^2$ valamilyen c_0 konstansra. (Ez tehát szomszédsági mátrix és lista esetén is igaz – de egyébként az ebben a jegyzetben nem tárgyalt, a gráfok tárolására alkalmas további adatszerkezetekre is.) A polinomiális algoritmus definíciója szerint ezért az algoritmus lépésszámára a $c_1 \cdot s^{k_1} = c_1 \cdot \left((c_0 \cdot (m+n)^2\right)^{k_1} = c_0 \cdot c_1 \cdot (m+n)^{2k_1}$ felső becslésnek kell teljesülnie minden bemenet esetén. Azt mondhatjuk ezért, hogy egy gráfelméleti algoritmus akkor polinomiális, ha a lépésszáma legföljebb $c \cdot (m+n)^k$ valamilyen rögzített c és k konstansokra (amiket a fenti becslésben a $c = c_0 \cdot c_1$ és a $k = 2k_1$ helyettesítésekkel kapunk).

A gráfelméleti algoritmusok hatékonysága kapcsán azonban általában nem elégszünk meg a polinomialitás tényének az ismeretével, hanem ennél finomabb skálán is értékeljük azokat. Előfordulhat például, hogy valamilyen problémára már ismerünk egy $c \cdot n^3$ futásidejű algoritmust (ami persze polinomiális), de helyette később

sikerül találni egy $c \cdot n^2$ lépésszámút; ez komoly fejleménynek számít, mert lehetővé teszi, hogy sokkal nagyobb gráfokra is elfogadható futásidőt garantáljunk. Általában a legjobbnak a lineáris, vagyis $c \cdot (m+n)$ futásidejű algoritmusok számítanak (egy ilyet már a következő fejezetben megismerünk). Ennél jobb ugyanis nem lehet egy algoritmus, ha legalább annyit feltételezünk róla, hogy a bemenetét végigolvassa.

A különböző, n-től és m-től függő lépésszámok összehasonlítása érdekében érdemes meggondolni ezeknek a paramétereknek az egymáshoz való viszonyát. Ha G egyszerű gráf, akkor m felülről becsülhető a csúcsokból alkotható párok számával, vagyis $m \leq \binom{n}{2} = \frac{n(n-1)}{2}$. Hasonlóan, ha G olyan irányított gráf, ami minden (u,v) élből legföljebb egy példányt tartalmaz és nincs benne hurokél, akkor $m \leq n(n-1)$ (mert egy él kezdőpontját n-féleképpen, utána a végpontját $(n-1)$-féleképpen választhatjuk meg). Másik oldalról, ha egy gráfról csak annyit feltételezünk, hogy nem tartalmaz izolált pontot, akkor az irányítatlan és az irányított esetben is az $m \geq \frac{n}{2}$ becslést kapjuk (mert minden csúcsra illeszkedik legalább egy él, de egy él két csúcsot köt össze). Így ezekkel a – gyakorlati alkalmazásokban legtöbbször észszerű – feltételezésekkel a $c_1 \cdot n \leq m \leq c_2 \cdot n^2$ becslést kapjuk (alkalmas c_1 és c_2 konstansokra). Így például a lineáris, vagyis $(n+m)$-mel arányos futásidejű algoritmusok mindig jobbak (de legalábbis nem rosszabbak), mint a $c \cdot n^2$ futásidejűek.

Összefoglalva a fentieket:

- Gráfelméleti algoritmusok tervezésekor és vizsgálatakor feltételezhetjük, hogy a csúcsok azonosítói és a bemenet részét képező számadatok is csak fix konstansnyi helyet foglalnak el a bemenetben, továbbá hogy a számokon végzett alapműveletekhez is csak fix konstansnyi időre van szükség.
- Szomszédsági listával, illetve szomszédsági mátrixszal tárolva egy n csúcsú és m élű (esetleg élsúlyozott) gráfot, a bemenet mérete $c \cdot (n+m)$, illetve $c \cdot n^2$, ahol c valamilyen rögzített konstans.
- Egy gráfelméleti algoritmus akkor polinomiális, ha a lépésszáma legföljebb $c \cdot (n+m)^k$ valamilyen c és k konstansokra. A gyakorlatban a k kitevő pontos értékének komoly jelentősége lehet, ezért annál jobbnak tekintünk egy algoritmust, minél kisebb a lépésszámára vonatkozó felső becslésben a k értéke.

A valósághoz tartozik sajnos az is, hogy számos, a gyakorlatban egyébként igen fontos gráfelméleti problémára a számítástudomány nem ismer polinomiális algoritmust; ilyenekkel is fogunk találkozni ebben a jegyzetben. Ezeknek a problémáknak egy jelentős részében jó okunk van feltételezni – bár ez nem bizonyított –, hogy nem csak jelenleg nem ismert rájuk polinomiális algoritmus, hanem nem is létezik ilyen. Erről a kérdésről az *Algoritmuselmélet* tárgyban lesz bővebben szó.

2. fejezet

Szélességi keresés

Már a gráfokkal kapcsolatban eddig megismert alapfogalmak is számos algoritmikus kérdést vetnek fel. Hogyan lehet hatékonyan eldönteni, hogy egy (például szomszédsági mátrixával vagy szomszédsági listával) adott G gráf összefüggő-e? Ha nem, hogyan lehet meghatározni egy csúcsának a komponensét? Ha igen, hogyan lehet megadni egy feszítőfáját? A *Szélességi keresés*, vagy az angol neve, a *Breadth First Search* rövidítéseként széles körben elterjedt nevén a *BFS algoritmus* többek között ezeket a problémákat is megoldja. Továbbá a fentiek mellett meghatározza egy adott s csúcsból az összes v csúcs távolságát – vagyis az s-ből v-be vezető legrövidebb út hosszát (ha ilyen út egyáltalán van). Itt egy út hossza alatt továbbra is egyszerűen az utat alkotó élek számát értjük (vagyis a BFS nem alkalmas a legrövidebb út feladat általánosabb, élsúlyokat is figyelembe vevő esetének a megoldására). Ez a feladat sok alkalmazásban felmerül: példa erre, ha egy számítógép-hálózatban úgy szeretnénk információt továbbítani, hogy az adatcsomagok a lehető legkevesebb közbülső csomóponton haladjanak át (vagyis a *hopok* számát minimalizáljuk). A BFS-t számos összetettebb gráfelméleti algoritmus is használja szubrutinként (erre több példát is fogunk látni később).

A BFS algoritmus működésének alapötlete rendkívül egyszerű: ha már megtaláltuk az s-től $0, 1, \ldots, t-1$ távolságra lévő csúcsokat, akkor a t távolságra lévők éppen azok, amik ezek között nem szerepelnek, de szomszédosak egy $t-1$ távolságra lévővel. Az s-től 0 távolságra nyilván csak maga s van. Így az eljárás először megkeresi az 1 távolságra lévő, vagyis s-sel szomszédos csúcsokat. Majd az ezek közül valamelyikkel, de s-sel nem szomszédos csúcsokat 2 távolságra lévőnek nyilvánítja. Ezek után a 2 távolságra lévők valamelyikével szomszédos, de eddig nem érintett csúcsokat 3 távolságra lévőnek nyilvánítja, stb.

Persze az algoritmus az s-től azonos távolságra lévő csúcsokat nem tudja egyszerre bejárni, így ezek között is kialakít egy sorrendet. Ez a sorrend az eljárás végrehajtásától függően többféle is lehet, de nem véletlenszerű, mert az algoritmus olyan sorrendben foglalkozik a csúcsokkal és próbál továbblépni belőlük, amilyen sorrendben megtalálta azokat. Vagyis ha a $t-1$ távolságra lévő csúcsok között x megelőzi y-t, akkor a t távolságra lévő csúcsok között is előbbre kerülnek azok a

(korábban még nem bejárt) csúcsok, amiket az eljárás x-ből ért el azokhoz képest, amiket y-ból.

Az alábbi pszeudokóddal a fenti, szemléletes képnél pontosabban is leírjuk a BFS algoritmust. Az eljárás a működése során az alábbi adatokat tartja nyilván:

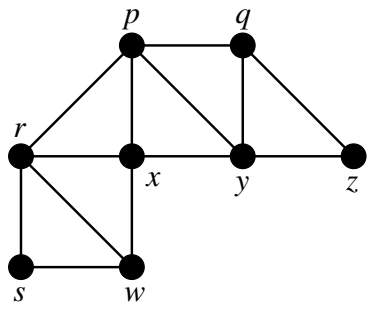
- táv(v) $(v \in V)$: v távolsága s-től
- *aktív_csúcs*: a jelenleg aktív csúcs, aminek a szomszédait éppen vizsgáljuk
- **L**: lista azokból a már bejárt csúcsokból, amiknek a szomszédait az eljárás még nem vizsgálta végig
- előző(v) $(v \in V, v \neq s)$: a v-t megelőző csúcs az algoritmus által megtalált, s-ből v-be vezető legrövidebb úton

BFS ALGORITMUS

Bemenet: Egy $G = (V,E)$ gráf és egy $s \in V$ csúcs

```
1   L ← (s); aktív_csúcs ← *
2   táv(s) ← 0; minden v ∈ V, v ≠ s-re: táv(v) ← ∞
3   minden v ∈ V-re: előző(v) ← *
4   ciklus: amíg L nem üres
5       aktív_csúcs ← L első tagja; aktív_csúcs törlése L-ből
6       ciklus: v végigfut aktív_csúcs minden szomszédján
7           ha táv(v) = ∞, akkor:
8               v hozzáfűzése L végére
9               táv(v) ← táv(aktív_csúcs) + 1
10              előző(v) ← aktív_csúcs
11      ciklus vége
12  ciklus vége
```

A fenti kódban ∞, illetve $*$ annak jelzésére szolgáló szimbólumok, hogy táv(v), illetve előző(v) még nem kapott értéket. Az eljárás működését a 2.1. ábra gráfján mutatjuk be.



2.1. ábra

Lefuttatva az algoritmust az alábbi adatok keletkeznek:

v	s	p	q	r	x	y	z	w
táv(v)	0	2	3	1	2	3	4	1
előző(v)	$*$	r	p	s	r	p	q	s

Természetesen ez csak az egyik lehetséges helyes futása az algoritmusnak: attól függően, hogy a 6-11. sorokban futó belső ciklus milyen sorrendben veszi sorra *aktív_csúcs* szomszédait, a BFS-nek számos más helyes futása is elképzelhető. Ebben a futásban az eljárás a csúcsokat s,r,w,p,x,q,y,z sorrendben járta be, vagyis *aktív_csúcs* sorra ezeket az értékeket vette fel. Az **L** lista tartalma a 4-12. sorokban futó külső ciklus magjának az első végrehajtása után $\mathbf{L} = (r,w)$ volt, majd a ciklusmag további végrehajtásai után sorra $\mathbf{L} = (w,p,x)$, $\mathbf{L} = (p,x)$, $\mathbf{L} = (x,q,y)$, $\mathbf{L} = (q,y)$, $\mathbf{L} = (y,z)$, $\mathbf{L} = (z)$ és legvégül **L** kiürült.

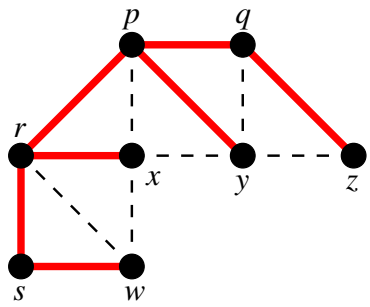
Az eljárásban az előző(v) változók szerepe az, hogy a kimenetből ne csak az s-ből az egyes csúcsokba vezető legrövidebb utak hossza legyen kiolvasható, hanem az ilyen utak egyike is. Valóban: egy tetszőleges v_0 csúcsból az előző$(v_0) = v_1$, előző$(v_1) = v_2, \ldots$ sorozat mentén haladva mindig eggyel csökken az s-től vett távolság, így végül táv(v_0) lépésben elérünk s-be. Például a fenti gráf esetében a z csúcsból az előző$(z) = q$, előző$(q) = p$, előző$(p) = r$, előző$(r) = s$ lépésekkel valóban táv$(z) = 4$ hosszú úton jutunk el s-be.

Könnyű végiggondolni, hogy a szélességi keresés helyesen működik, vagyis az eljárás végén táv$(v) = t$ pontosan azokra a v csúcsokra teljesül, amiknek az s-től való távolsága t. Ezt legegyszerűbb teljes indukcióval megmutatni. Az állítás $t = 0$-ra nyilván igaz, hiszen s-től 0 távolságra csak saját maga van. Ha pedig az állítás a $0,1,\ldots,t-1$ értékekre már igaz valamilyen $t \geq 1$ esetén, akkor az s-től $t-1$ távolságra lévő u csúcsok az indukciós feltevés szerint valóban megkapják a táv$(u) = t-1$ értéket az algoritmus futása során. Mivel ekkor ezek bekerülnek az **L** listába, így az eljárás egy későbbi pontján aktívvá kell válniuk. Következik, hogy pontosan azok a v csúcsok kapják a táv$(v) = t$ értéket, amik szomszédosak egy s-től $t-1$ távolságra lévő csúccsal, de a megfelelő $t-1$ távolságra lévő csúcs aktívvá válása előtt még táv$(v) = \infty$; ez utóbbi pedig ismét az indukciós feltevés szerint azt jelenti, hogy v távolsága s-től legalább t (mert táv(v) nem kapta meg a t-nél kisebb értékek egyikét sem). Más szóval: táv$(v) = t$ pontosan akkor következik be v-re, ha v szomszédos egy s-től $t-1$ távolságra lévő csúccsal és v távolsága s-től legalább t; vagyis ha v távolsága s-től éppen t.

Ugyanez a gondolatmenet mutatja azt is, hogy az algoritmus leállása után táv$(v) = \infty$ pontosan azokra a csúcsokra teljesül, amik nem érhetők el s-ből úton – vagyis amelyek nincsenek s-sel egy komponensben. Így a G gráf pontosan akkor összefüggő, ha az eljárás végén táv$(v) = \infty$ egyetlen v csúcsra sem igaz.

2.1. A BFS-fa

A BFS algoritmus fenti példafuttatásában külön figyelmet érdemel a táblázat második sora – általában pedig azok az élek, amik minden $v \neq s$ esetén v-t előző(v)-vel kötik össze. A 2.1. ábra gráfjára ezek a 2.2. ábrán, pirossal láthatók.



2.2. ábra

Megfigyelhető, hogy a szóban forgó élek egy fát, mégpedig a G s-et tartalmazó komponensének egy F feszítőfáját alkotják. Ez általában is így van, ami közvetlenül adódik az 1.26. Tétel bizonyításából, illetve az utána írt megjegyzésből. Valóban, amikor az eljárás a v csúcsot eléri (vagyis táv(v) és előző(v) értéket kap), akkor v elsőfokú lesz az azokból az $\{u, \text{előző}(u)\}$ élekből (és G összes csúcsából) álló gráfban, amikre az eljárás u-t v-nél korábban elérte (vagyis előző(u) már hamarabb értéket kapott). Így az $\{\text{előző}(v), v\}$ él hozzáadása valóban két különböző komponenst köt össze (amik közül az egyik csak a v csúcsból áll).

Az így kapott F fát *BFS-fának* nevezzük; ez tehát összefüggő G esetén feszítőfája G-nek és bármely v csúcsra az s-et v-vel összekötő F-beli út a legrövidebbek egyike az s-ből v-be vezető G-beli utak közül.

2.2. Szélességi keresés irányított gráfban

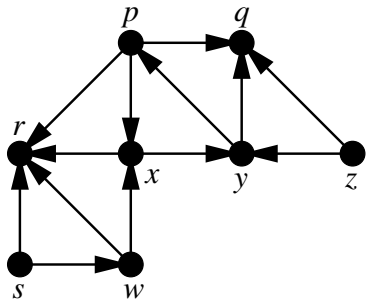
Az adott csúcsból induló legrövidebb (vagyis legkevesebb élből álló) irányított utak megtalálásának feladata irányított gráfokban is gyakran felmerülő probléma. Szerencsére ennek a feladatnak a megoldására nem kell vadonatúj algoritmust kidolgoznunk, a BFS eljárás ezt is megoldja. Ehhez a fentebb látott pszeudokódnak csak a 6. sorát kell minimálisan módosítani:

6 **ciklus:** v **végigfut** azokon a csúcsokon, amikbe *aktív_csúcs*-ból vezet él

Más szóval: az eljárás mindig csak az éppen aktív csúcsból kiinduló irányított éleken továbblépve próbál addig még el nem ért v csúcsot találni.

Például a 2.3. ábra irányított gráfjára az eljárást (az egyik lehetséges módon) lefuttatva az a csúcsokat s, r, w, x, y, p, q sorrendben járja be és az alábbi adatok keletkeznek:

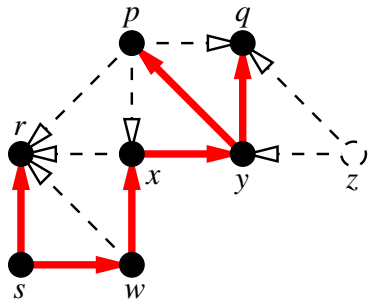
v	s	p	q	r	x	y	z	w
táv(v)	0	4	4	1	2	3	∞	1
előző(v)	$*$	y	y	s	w	x	$*$	s



2.3. ábra

Látható, hogy itt már a z csúcs nem elérhető s-ből irányított úton. Általában is igaz, hogy pontosan azok a v csúcsok érhetők el s-ből irányított úton, amikre a BFS eljárás leállásakor $\text{táv}(v) \neq \infty$. Érdemes megfigyelni azt is, hogy a BFS-t ugyancsak a fenti irányított gráfra, de az x csúcsból indítva x-en kívül csak a p, q, r és y csúcsok volnának elérhetők – vagyis az elérhető csúcsok halmaza még akkor is nagyban függ a kiindulópont választásától, ha a gráf az élek irányítását figyelmen kívül hagyva egyébként összefüggő. (Ez a példa is azt érzékelteti, amit az 1.6. szakaszban már említettünk: az összefüggőség és a komponens fogalma irányított gráfokra összetettebb.)

Az előző(v) változók szerepe a BFS eljárás irányított változatában azonos a korábbival: tetszőleges v csúcsra (legalábbis azokra, amikre $\text{táv}(v) \neq \infty$ és így előző(v) is értéket kapott) ezeket használva lépegethetünk vissza v-től s-ig, hogy egy s-ből v-be vezető legrövidebb irányított utat is megkapjunk. Ismét hasznos külön kiemelni azokat az irányított éleket, amik előző(v)-ből v-be vezetnek valamilyen v-re: ezek (az élek irányításától most eltekintve) megint egy F fát alkotnak, aminek csúcshalmaza az s-ből elérhető csúcsokból áll és F élei mentén s-ből eljuthatunk az összes többi csúcsba a lehető legrövidebb úton (legalábbis az ilyen utak egyikén). A BFS algoritmus fenti futásából származó BFS-fát a 2.4. ábra mutatja.



2.4. ábra

2.3. Újraindított szélességi keresés

A fentiek szerint a BFS algoritmus (az eddig tárgyalt formájában) nem deríti fel a gráf teljes csúcshalmazát, csak az s-ből elérhető csúcsokat (ami irányítatlan gráf esetén s komponensének a csúcsait jelenti). Bizonyos alkalmazásokban azonban szükség lehet a teljes csúcshalmaz bejárására. Ezt a fentebb bemutatott pszeudokód egyszerű kiegészítésével lehet elérni: ha a 4-12. sorokban zajló ciklus leállása után még van a gráfnak olyan v csúcsa, amire táv$(v) = \infty$, akkor választunk egy tetszőleges ilyen v csúcsot és ebből újraindítjuk az eljárást. Ez tehát pontosabban azt jelenti, hogy a 4-12. sorokban szereplő ciklust beágyazzuk egy másik ciklusba, ami annak a leállása után keres egy, a táv$(v) = \infty$ feltételt kielégítő v csúcsot, majd a táv$(v) \leftarrow 0$, $\mathbf{L} = (v)$ beállítások után újraindítja a 4. sorban induló ciklust; ha pedig ilyen v-t már nem talál, akkor megáll. Ezzel a módosítással tehát az algoritmus bejárja a gráf teljes csúcshalmazát.

Irányítatlan G gráf esetén ezzel a kiegészítéssel meg lehet határozni a G komponenseit – például úgy, hogy a komponensek számozására egy c változót használunk, amit kezdetben 1-nek inicializálunk és a (4. sorban induló) ciklus minden újraindításakor 1-gyel megnövelünk. Ha az algoritmus futása során egy $C(v)$ értéket is fenntartunk minden csúcshoz, amit a v csúcs elérésekor (vagyis például a pszeudokód 10. sora után) a c aktuális értékére állítunk be, akkor az eljárás leállása után az azonos $C(v)$ értéket kapott csúcsok tartoznak közös komponensbe és c végső értéke mutatja a gráf komponenseinek a számát.

2.4. A BFS algoritmus lépésszáma

Minden algoritmus, így a BFS esetében is fontos meggondolni annak a futásidejét. Mivel a BFS eljárás szempontjából a párhuzamos és hurokélek érdektelenek, ezért feltehetjük, hogy a bemenetként kapott G gráf egyszerű. Jelölje továbbra is G csúcsainak, illetve éleinek a számát n, illetve m.

Tegyük fel először, hogy G szomszédsági listával van megadva. Ekkor a 6-11. sorokban zajló belső ciklus nagyon egyszerűen és hatékonyan végrehajtható: ehhez csak végig kell lépkedni a v-hez tartozó listán (az irányított és az irányítatlan esetben is). Mivel ezeknek a listáknak az összhossza az irányított, illetve irányítatlan esetben m, illetve $2m$ és az algoritmus tejes futása során mindegyiken csak egyszer kell végigmenni, ezért ebből összesen $c_1 \cdot m$ lépésszám adódik (ahol c_1 valamilyen rögzített konstans). Ezen felül minden csúcs aktívvá válása (vagyis a pszeudokód 5. sorának a végrehajtása) is igényel valamilyen c_2 konstansnyi lépést, ez tehát összesen további $c_2 \cdot n$-nel járul hozzá a lépésszámhoz. Ezért azt mondhatjuk, hogy a teljes lépésszám legföljebb $c \cdot (n+m)$ valamilyen c konstansra – ami azt jelenti, hogy az eljárásnak ez az implementációja nem csak polinomiális lépésszámú, de még azon belül is rendkívül hatékony.

Érdemes végiggondolni azt az esetet is, ha G szomszédsági mátrixszal van megadva. Említettük már, hogy ebben az esetben egy v csúcs szomszédai sokkal kevésbé könnyen elérhető módon állnak rendelkezésre: végig kell értük pásztáznunk az $A(G)$

teljes, v-nek megfelelő sorát (vagy oszlopát). Bár a fentebb írtakhoz hasonlóan itt is elég ezt minden v-re egyszer megtenni, de akkor sem elég $d(v)$-vel arányos időt fordítanunk az éppen aktív v csúcsra, hanem csúcsonként $c \cdot n$, összesen pedig $c \cdot n^2$ lépésszámot kapunk. Bár ez is polinomiális futásidőt jelent, de rosszabb a szomszédsági lista esetén elért futásidőnél.

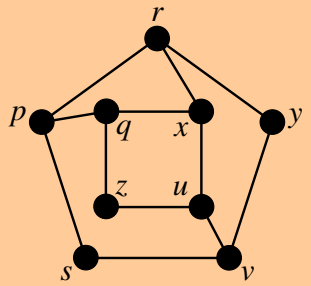
Egy algoritmus definíció szerint akkor lineáris futásidejű, ha minden s méretű bemeneten legföljebb $c \cdot s$ lépés után megáll valamilyen c konstansra. A fentiek szerint a BFS lépésszáma $c \cdot (n+m)$, illetve $c \cdot n^2$ (valamilyen c konstansokra), ha a bemenetet szomszédsági listával, illetve szomszédsági mátrixszal tároljuk. Azonban a bemenet s mérete is más a két esetben: szomszédsági lista esetén $s = c \cdot (n+m)$, szomszédsági mátrix esetén pedig $s = c \cdot n^2$ (ahol a c konstansok rögzítettek, de egymástól és a fentebb használtaktól is eltérhetnek). Ebből az következik, hogy a BFS algoritmus valójában mindkét esetben lineáris futásidejű: a lépésszáma a bemenet méretének egy konstansszorosával felülről becsülhető.

Bár ez az állítás kétségtelenül igaz, de inkább elfedi a lényeget, mint megvilágítja azt. A BFS algoritmust a fentiek szerint valóban érdemesebb szomszédsági listával implementálni, mert így a tárhellyel és a futásidővel is spórolhatunk a szomszédsági mátrixos implementációhoz képest. Más megfogalmazásban: a BFS csak azért lineáris futásidejű szomszédsági mátrixos implementáció esetén is, mert a fölöslegesen nagy futásidő fölöslegesen nagy tárhellyel párosul.

2.1. Feladat. Az alábbi ábráról véletlenül lemaradt a gráf egy éle (de a csúcsok mind rajta vannak). Megállapítható-e biztosan, hogy a hiányzó él melyik két csúcsot kötötte össze, ha tudjuk, hogy az s-ből indított BFS eljárás a gráf csúcsait az alábbi sorrendben járta be (vagyis *aktív_csúcs* sorra ezeket az értékeket vette fel)?

a) $s, p, v, q, r, z, u, y, x$

b) $s, v, p, y, z, u, q, r, x$



Megoldás: a) A hiányzó él lehetett akár $\{p,z\}$, akár $\{v,z\}$. Valóban: az első esetben s szomszédainak, vagyis p-nek és v-nek a bejárása után az algoritmus p szomszédait járja be, ezek közül q és r után épp az utolsóként z-t; a másik esetben p szomszédai után a v szomszédainak felsorolását kezdi az eljárás z-vel. Így ebben az esetben nem állapítható meg biztosan a hiányzó él (csak annyi mondható róla, hogy $\{p,z\}$ és $\{v,z\}$ közül az egyik).

b) Ha s-nek volna más szomszédja is az ábrán is látható p-n és v-n kívül, akkor az csak y lehetne, hiszen a bejárás sorrendjében ez következik s, p és v után. Ez

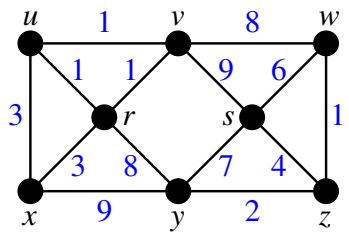
azonban lehetetlen: így y után nem következhetne z, mert az sem s-nek, sem v-nek nem lehetne már szomszédja (hiszen az ábráról csak egy él hiányzik). Így s-nek csak az ábrán is látható két szomszédja van, amiknek a bejárása után tehát v (további) szomszédainak kell jönnie. Mivel itt y, z, majd u következik, ezért csak a $\{v,z\}$ él hiányozhat a gráfból, más lehetőség itt nincs. □

3. fejezet

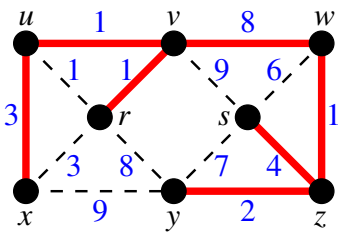
Minimális összsúlyú feszítőfa

Térjünk vissza az 1.5. szakaszban már említett problémára: a G összefüggő gráf egy tervezés alatt álló hálózatot reprezentál és azt kell eldöntenünk, hogy G melyik élei kerüljenek be a hálózatba úgy, hogy a lehető legkisebb költséggel kapjunk összefüggő gráfot a $V(G)$ csúcshalmazon. Akkor költség alatt egyszerűen a kiválasztott élek számát értettük és így a probléma megoldásait G feszítőfái adták. Most ennek a problémának egy jóval általánosabb és az alkalmazások szempontjából sokkal fontosabb változatáról lesz szó: a probléma bemenete G-n kívül tartalmaz G minden e éléhez egy $w(e)$ súlyt is és a feladatunk egy olyan F feszítőfa meghatározása, amire a $\sum\{w(e) : e \in E(F)\}$ összeg, vagyis F éleinek az összsúlya a lehető legkisebb. Ez a probléma számtalan gyakorlati alkalmazásban felmerül, például számítógépes, telekommunikációs vagy közlekedési hálózatok tervezésekor; ezekben $w(e)$ az e által reprezentált hálózati elem megépítésének a költségét fejezi ki.

A 3.1a ábrán a probléma egy lehetséges bemenete látható, a kék számok mutatják az élek $w(e)$ súlyát. Az alább bemutatásra kerülő algoritmus segítségével meg fogjuk tudni mutatni, hogy a 3.1b ábrán látható, 20 összsúlyú feszítőfa minimális összsúlyú. (Ez nem az egyetlen 20, vagyis minimális összsúlyú feszítőfa ebben a gráfban, de 20-nál kisebb összsúlyú nincs.)



3.1a ábra



3.1b ábra

Az alább bemutatandó algoritmus Joseph Kruskal (1929 – 2010) amerikai matematikustól származik 1956-ból (bár a cseh Otakar Borůvka már 1926-ban pub-

likált egy ehhez mind működésében, mind hatékonyságában nagyon hasonló algoritmust). Kruskal algoritmusát *mohó algoritmusnak* is szokták nevezni, mert a működése közben mindig az éppen legjobbnak tűnő döntést hozza meg és ezeket a döntéseket később sem bírálja felül. Ilyen szemléletű algoritmusból nagyon sokat ismer a számítástudomány; ezek egyszerű működésű és kedvező futásidejű eljárások – de éppen az egyszerűségük miatt nagyon ritkán adnak optimális eredményt. A minimális összsúlyú feszítőfa problémája azonban kivételesen szerencsés ebből a szempontból: még a mohó megközelítés is mindig optimális eredményre vezet.

Kruskal algoritmusa az 1.26. Tétel bizonyításában látott eljárással készíti el G egy feszítőfáját: folyamatosan karban tart egy F részgráfot, aminek a csúcshalmaza végig azonos G csúcshalmazával, az élhalmazát pedig az üres halmaztól indítva mindig egy olyan éllel bővíti, ami az aktuális részgráf két különböző komponensébe tartozó csúcsokat köt össze (és így minden él hozzávétele eggyel csökkenti a komponensek számát). Kruskal algoritmusa ezt az eljárást mindössze azzal egészíti ki, hogy az F-be aktuálisan beválasztható élek közül (vagyis azok közül, amik F különböző komponensei között futnak) mindig a legkisebb súlyúak egyikét választja. Ez a választási szabály úgy is megfogalmazható, hogy az eljárás mindig a gráf körmentességét megőrző élek közül a legkisebb súlyúak egyikét választja. (Valóban, az $e = \{u,v\}$ él hozzávétele pontosan akkor nem hoz létre kört, ha e hozzávétele előtt nincs út u és v között, vagyis ha u és v nem ugyanabba a komponensbe tartoznak.)

Az alábbi táblázat az algoritmus egy lehetséges futtatását mutatja a 3.1a ábrán látható bemenetre. Kezdetben minden csúcs külön komponenst alkot, így az eljárás a legkisebb súlyú élek egyikét, $\{w,z\}$-t választja (de ehelyett bármelyik 1 súlyút választhatná). Ezután még két 1 súlyú élt választ, amik mindkét esetben különböző komponensek között futnak az aktuális gráfban. Bár $\{u,r\}$ is 1 súlyú, de ez ezen a ponton már nem választható, mert u és r az aktuális, három élű gráfban közös komponensbe tartoznak (vagyis $\{u,r\}$ hozzávétele kört hozna létre). Így most a 2 súlyú $\{y,z\}$-t, majd a 3 súlyú $\{u,x\}$-et választja. Itt $\{u,x\}$ helyett $\{r,x\}$ is jó választás lett volna, de $\{u,x\}$ kiválasztása után $\{r,x\}$ már nem választható, mert r és x között van út az aktuális gráfban. A most következő, 4 súlyú $\{s,z\}$ kiválasztása után a gráfnak már csak két komponense van, így az eljárás végül kénytelen az ezek között futó 8 súlyú élek egyikét választani.

F újonnan választott éle	az újonnan választott él súlya	az aktuális gráf komponenseinek csúcshalmazai
		$\{u\}, \{v\}, \{w\}, \{r\}, \{s\}, \{x\}, \{y\}, \{z\}$
$\{w,z\}$	1	$\{u\}, \{v\}, \{r\}, \{s\}, \{x\}, \{y\}, \{w,z\}$
$\{u,v\}$	1	$\{u,v\}, \{r\}, \{s\}, \{x\}, \{y\}, \{w,z\}$
$\{v,r\}$	1	$\{u,v,r\}, \{s\}, \{x\}, \{y\}, \{w,z\}$
$\{y,z\}$	2	$\{u,v,r\}, \{s\}, \{x\}, \{w,y,z\}$
$\{u,x\}$	3	$\{u,v,r,x\}, \{s\}, \{w,y,z\}$
$\{s,z\}$	4	$\{u,v,r,x\}, \{w,s,y,z\}$
$\{v,w\}$	8	$\{u,v,w,r,s,x,y,z\}$

Természetesen be kell még bizonyítanunk, hogy Kruskal algoritmusa valóban

mindig minimális összsúlyú feszítőfát eredményez.

3.1. Tétel. *Ha a G összefüggő gráf $F_{mohó}$ feszítőfáját úgy készítjük el, hogy az üres halmaztól indulva mindig az aktuális részgráf különböző komponenseit összekötő, legkisebb súlyú élek egyikével bővítjük $F_{mohó}$ élhalmazát, akkor $F_{mohó}$ minimális összsúlyú feszítőfa G-ben.*

Bizonyítás: Azt már korábban, az 1.26. Tétel bizonyításában beláttuk, hogy $F_{\text{mohó}}$ feszítőfa G-ben; azt kell tehát megmutatnunk, hogy minimális összsúlyú.

Számozzuk meg G éleit e_1-től e_m-ig úgy, hogy $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$ teljesüljön. Ekkor az algoritmus működése felfogható úgy is, hogy az G éleit ebben a sorrendben végigvizsgálja és minden e_i él $F_{\text{mohó}}$-ba való felvételéről aszerint dönt igennel vagy nemmel, hogy e_i végpontjai a korábban már felvett élek gráfjának különböző komponenseibe tartoznak-e (hiszen így a szóba jövő élek közül valóban mindig a legkisebb súlyúak egyikét választja). A bizonyítás leírása érdekében feleltessünk meg az eljárás működésének egy m hosszúságú $\mu_1, \mu_2, \ldots, \mu_m$ bináris sorozatot: ennek az i-edik helyén álló μ_i elem aszerint legyen 1 vagy 0 minden $1 \leq i \leq m$ esetén, hogy az algoritmus e_i-ről igen vagy nem döntést hozott-e. Sőt, ne csak $F_{\text{mohó}}$-nak feleltessünk meg egy ilyen sorozatot: minden F feszítőfa leírható azzal a hozzá tartozó bináris sorozattal, aminek az i-edik eleme aszerint 1 vagy 0 minden $1 \leq i \leq m$ esetén, hogy $e_i \in E(F)$ teljesül-e vagy sem.

Indirekt fogunk bizonyítani: tegyük fel, hogy $F_{\text{mohó}}$ nem minimális összsúlyú. Válasszunk egy optimális, vagyis minimális összsúlyú F_{opt} feszítőfát – de ha ilyenből több is van, akkor ne tetszőlegesen: az optimális feszítőfák között F_{opt} olyan legyen, hogy a neki (a fenti értelemben) megfelelő $\omega_1, \omega_2, \ldots, \omega_m$ bináris sorozat a lehető legtovább azonos legyen az $\mu_1, \mu_2, \ldots, \mu_m$ sorozattal. Legyen k a legkisebb olyan index, amire $\mu_k \neq \omega_k$ és legyen $e_k = \{u, v\}$; ilyen k-nak kell lennie, mert $F_{\text{mohó}}$ nem optimális. Ekkor tehát (ahogyan az az alábbi táblázatból is látszik) az $e_1, e_2, \ldots, e_{k-1}$ élek közül ugyanazok tartoznak $E(F_{\text{mohó}})$-ba, mint $E(F_{\text{opt}})$-ba, de e_k-ra ez már nem igaz, ez pontosan az egyikben van benne.

	e_1	e_2	$\ldots$	e_{k-1}	e_k	$\ldots$	e_ℓ	$\ldots$
$F_{\text{mohó}}$:	μ_1	μ_2	$\ldots$	μ_{k-1}	μ_k	$\ldots$		
	$\|\|$	$\|\|$	$\ldots$	$\|\|$	$\nparallel$			
F_{opt}:	ω_1	ω_2	$\ldots$	ω_{k-1}	ω_k	$\ldots$		

Jelölje F_0 azt a gráfot, amire $V(F_0) = V(G)$ és $E(F_0) = \{e_i : 1 \leq i < k \text{ és } \mu_i = 1\}$; vagyis F_0 a G összes csúcsát tartalmazza, az élei közül pedig azokat, amikről az algoritmus az e_k vizsgálata előtt igennel döntött. Ekkor F_0 részgráfja $F_{\text{mohó}}$-nak és a fentiek szerint F_{opt}-nak is. Két esetet kell megvizsgálnunk: vagy $\mu_k = 0$ és $\omega_k = 1$, vagy $\mu_k = 1$ és $\omega_k = 0$.

Először tegyük fel, hogy $\mu_k = 0$ és $\omega_k = 1$. Ez tehát azt jelenti, hogy az eljárás e_k-ról nemleges döntést hozott, vagyis u és v között vezet út F_0-ban (mert ugyanabba a komponensébe tartoznak). Más szóval: ha az e_k élt F_0-hoz adjuk, akkor a kapott

F' gráf tartalmaz kört. Ez azonban ellentmondás, hiszen $\omega_k = 1$ miatt F' részgráfja F_{opt}-nak, márpedig F_{opt} körmentes (hiszen fa).

Vizsgáljuk meg ezért a másik esetet: tegyük fel, hogy $\mu_k = 1$ és $\omega_k = 0$. Ekkor tehát az eljárás e_k-ról igennel döntött. Mivel F_{opt} összefüggő, ezért van benne egy P út u-ból v-be. P-nek kell legyen egy F_0-ba nem tartozó e_ℓ éle, különben u és v között volna út F_0-ban, így az algoritmus nem dönthetett volna e_k-ról igennel. Mivel $e_\ell \in E(F_{\text{opt}})$, de $e_\ell \notin E(F_0)$ és $e_\ell \neq e_k$ (mert $\omega_k = 0$), ezért $e_\ell \notin \{e_1, e_2, \ldots, e_k\}$. Vagyis $k < \ell$, amiből $w(e_k) \leq w(e_\ell)$ adódik. Legyen most F^* az a gráf, amit F_{opt}-ból nyerünk e_k hozzávételével, majd e_ℓ törlésével; vagyis $E(F^*) = E(F_{\text{opt}}) \cup \{e_k\} \setminus \{e_\ell\}$. Alább megmutatjuk, hogy F^* is feszítőfája G-nek; ebből pedig következni fog a kívánt ellentmondás. Valóban, $w(e_k) \leq w(e_\ell)$ miatt F^* összsúlya legföljebb annyi, mint F_{opt}-é; kisebb viszont nem lehet (mert F_{opt} minimális összsúlyú), így egyenlő azzal. Ez azt jelenti, hogy F^* is egy minimális összsúlyú feszítőfa, de a neki megfelelő bináris sorozatban az első $k-1$ tagon kívül még a k-adik is azonos μ_k-val, hiszen $e_k \in E(F^*)$ és az $e_1, e_2, \ldots, e_{k-1}$ élek közül ugyanazok tartoznak $E(F_{\text{opt}})$-ba, mint $E(F^*)$-ba. Ez pedig valóban ellentmond annak, ahogyan F_{opt}-ot választottuk a minimális összsúlyú feszítőfák közül.

Annak megmutatása van tehát már csak hátra, hogy F^* feszítőfája G-nek. Ha az $e_k = \{u, v\}$ élt hozzáadjuk F_{opt}-hoz, az az F_{opt}-beli, u és v között vezető P útból egy C kört hoz létre. Ezért F^*-ban van út e_ℓ végpontjai között: az a P' út, amit C-ből az e_ℓ törlésével kapunk. Korábban, az 1.23. Tétel bizonyításában láttuk, hogy egy gráf komponenseinek a csúcshalmazai nem változnak meg, ha a gráfhoz azonos komponensben lévő csúcsokat összekötő élt adunk (mert ettől semelyik két csúcsra nem változik meg azoknak az egymásból való elérhetősége). Ezt használva tehát F^* komponenseinek a csúcshalmazai sem változnak, ha e_ℓ-t hozzáadjuk (mert e_ℓ végpontjai P' létezése miatt F^*-nak ugyanabban a komponensében vannak). Azonban tudjuk, hogy így összefüggő gráfot kapunk, mert F_{opt} összefüggő és ezt nyilván e_k hozzáadása sem ronthatta el. Következik tehát, hogy F^* is összefüggő. Másrészt azt is tudjuk, hogy F^* $n-1$ élű, mert az élszáma azonos F_{opt}-éval. Ebből pedig az 1.26. Tétel miatt következik, hogy F^* feszítőfa: valóban, F^* az összefüggősége miatt tartalmaz feszítőfát, de mivel az is $n-1$ élű, ezért ez csak saját maga lehet. Ezzel tehát a bizonyítás teljes. □

3.1. Kruskal algoritmusának implementációja

G éleinek az élsúlyok szerinti növekvő (vagy pontosabban: nem csökkenő) sorbarendezése nem csak a fenti bizonyításban volt hasznos: Kruskal algoritmusának az implementációit is érdemes így kezdeni. Így ebben a sorrendben az éleken végighaladva az eljárásnak mindig csak azt kell tudnia eldönteni, hogy a soron következő él különböző komponensbe tartozó csúcsokat köt-e össze. Az algoritmus az alábbi pszeudokóddal írható le; ebben k jelöli az aktuális gráf élszámát (vagyis az eddig kiválasztott élek számát), j pedig az éppen vizsgált él sorszámát az élek sorozatában.

KRUSKAL ALGORITMUSA

Bemenet: Egy n csúcsú és m élű $G = (V,E)$ összefüggő gráf és egy $w : E \to \mathbb{R}$ súlyfüggvény

```
1  RENDEZZÜK AZ ÉLEKET a w(e) súlyok szerinti növekvő sorrendbe:
   legyen w(e_1) ≤ w(e_2) ≤ ... ≤ w(e_m)
2  k ← 0; j ← 1; E(F) ← ∅
3  ciklus: amíg k < n − 1
4      VIZSGÁLJUK MEG, HOGY e_j VÉGPONTJAI F KÜLÖNBÖZŐ KOM-
       PONENSEIBE TARTOZNAK-E
5      ha e_j végpontjai különböző komponensbe tartoznak, akkor:
6          E(F) ← E(F) ∪ {e_j}
7          k ← k + 1
8      j ← j + 1
9  ciklus vége
```

Az eljárás 3. sorában a $k < n-1$ feltételt az indokolja, hogy ha $E(F)$ mérete eléri az $(n-1)$-et, akkor F már feszítőfa (hiszen a komponenseinek a száma 1-re csökkent), így az eljárás megállhat. (Elvileg ugyan folytatódhatna a hátralévő élek megvizsgálásával is, de ez fölösleges volna, mert e_j végpontjai innentől mindig közös komponensbe tartoznának, ezért $E(F)$ már nem változna.) A fenti pszeudokód épít G összefüggőségére, különben hibás futást eredményez: ha G nem volna összefüggő, akkor k értéke sosem érné el az $(n-1)$-et, így j értéke túlfutna m-en (és a nem létező e_{m+1} végpontjait kezdené vizsgálni). Ezért ha nincs előzetes információnk G összefüggőségéről, akkor a 3. sor feltételét érdemes „amíg $k < n-1$ és $j \leq m$” alakúra változtatni; ha az ebben a formában futtatott eljárás leállása után $k < n-1$, akkor G nem összefüggő.

Az eljárás 1. sorának a végrehajtásához egy rendező algoritmusra van szükség; ilyenből sokat ismer a számítástudomány és az adott célra legalkalmasabbnak a megválasztása függhet attól, hogy a $w(e)$ súlyokkal kapcsolatban milyen előzetes információkkal rendelkezünk. Ezzel a kérdéssel ebben a jegyzetben nem foglalkozunk, az *Algoritmuselmélet* tárgyban lesz róla szó. Ennél is érdekesebb kérdés, hogy hogyan hajtsuk végre az eljárás 4. sorában írt vizsgálatot. Mivel itt azt kell eldönteni, hogy van-e út F-ben e_j végpontjai között, ezért ennek a megválaszolására akár a 2. fejezetben látott BFS algoritmust is használhatnánk. Ha így tennénk, akkor ezzel összesen legföljebb m-szer futtatnánk a BFS-t, de mindig egy legföljebb $n-2$ élű gráfon, így Kruskal algoritmusának a teljes futásideje (figyelembe véve, hogy a BFS lineáris futásidejű) $c \cdot n \cdot m$ volna valamilyen c konstansra (amibe az 1. sorbeli rendezés is bőven belefér). Ez a lépésszám ugyan polinomiális, de ennél sokkal jobbat lehet elérni már a következő, nagyon egyszerű gondolattal is: hasonlóan ahhoz, ahogyan az algoritmust fentebb a 3.1. ábra bemenetére futtattuk, érdemes inkább az aktuális gráf komponenseinek a csúcshalmazait tárolni. Ez megtehető akár azon a kézenfekvő módon is, hogy minden v csúcshoz fenntartunk egy $C(v)$ értéket, ami a komponensét azonosítja (abban az értelemben, hogy a közös $C(v)$ értékkel rendelkező csúcsok tartoznak ugyanabba a komponensbe). Kezdetben minden csúcs külön

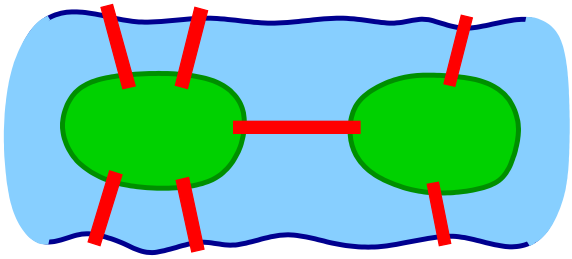
komponenst alkot, így minden $v \in V(G)$-re a $C(v) = v$ értékkel indíthatjuk az eljárást. Később az algoritmus futása közben a 4. sor végrehajtása egyszerűen abból áll, hogy ha $e_j = \{u, v\}$, akkor összehasonlítjuk $C(u)$ és $C(v)$ értékét. Ez ugyan nagyon gyors, de cserébe a 6. sor minden végrehajtása után gondoskodni kell u és v komponensének az egyesítéséről; ez megtehető úgy, hogy minden olyan x csúcsra, amire $C(x) = C(u)$, $C(x)$ értékét $C(v)$-re változtatjuk (vagy fordítva). Mivel minden ilyen egyesítéshez végig kell pásztáznunk mind az n csúcsot és ezt összesen $(n-1)$-szer kell megtennünk (hiszen ennyiszer történik komponensek egyesítése az algoritmus futása során), ezzel $c \cdot n^2$ futásidőt kapunk. Ez tehát általában sokkal jobb a fentebb írt $c \cdot m \cdot n$ futásidőnél – de itt már előfordulhat, hogy az 1. sorban szereplő rendezés futásideje ennél nagyobb és ezért az algoritmus teljes futásidejét az határozza meg (akkor, ha G nagyon „sűrű", vagyis m az egyszerű gráfokra vonatkozó $c' \cdot n^2$-es felső becslést megközelíti és a $w(e)$ élsúlyok is tetszőleges értéket felvehetnek).

Bár erről ebben a jegyzetben nem lesz szó, de érdemes megemlíteni, hogy létezik Kruskal algoritmusának egy, a fentinél jóval ravaszabb implementációja is: ha a komponensek csúcshalmazainak a tárolását és egyesítését az úgynevezett *unió-holvan adatszerkezettel* valósítjuk meg, akkor ezzel $c \cdot m \cdot \log m$ lépésszám érhető el – vagy akár még ennél is kedvezőbb, az 1. sorban írt rendezés lépésszámigényétől függően. Ez pedig jelentősen jobb lehet a fentebb írt $c \cdot n^2$-nél, ha G a fenti értelemben nem túl sűrű.

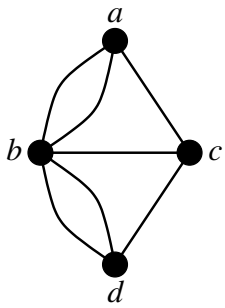
4. fejezet

Euler-séták és Euler-körséták

A gráfelmélet alapvetően a 20. században, annak is jórészt a második felében bontakozott ki és vált a matematika jelentős ágává. A történetileg első gráfelméleti eredménynek mégis egy 18. századi felfedezést szoktak tartani, aminek a kiindulópontja egy fejtörő, a *Königsbergi hidak problémája* volt. Königsberg városához (ami akkor Poroszország része volt, ma pedig Kalinyingrád néven Oroszországé) két sziget tartozott a várost átszelő Pregel folyón. A két szigetet a folyó két partjával, illetve egymással összesen hét híd kötötte össze, ahogyan az a 4.1a ábrán látható. A königsbergi polgárok a fejükbe vették, hogy úgy szeretnének a városban sétálni, hogy eközben minden hídon pontosan egyszer haladjanak át; sőt, ha lehet, a séta végén ráadásul a kiindulópontjukra érkezzenek vissza. Mivel ez sehogyan sem sikerült, megkeresték a kor híres matematikusát, a svájci Leonhard Eulert (1707 – 1783). Euler bebizonyította, hogy ilyen séta nem létezik – még akkor sem, ha a kezdőpont és a végpont lehet különböző.



4.1a ábra



4.1b ábra

Euler megfejtésének bemutatásához fogalmazzuk át a problémát gráfelméleti feladattá. Mivel a keresett sétában csak a hidak érintése fontos, ezért megtehetjük, hogy a folyó két partját, illetve a két szigetet egy gráf csúcsainak tekintjük, a hidak pedig a gráf élei lesznek, ahogyan az a 4.1b ábrán látható. Ekkor a feladatunk

az, hogy ebben a gráfban olyan sétát keressünk, ami minden élt pontosan egyszer tartalmaz; az ilyen sétákat azóta Eulerről nevezték el.

4.1. Definíció. *A G gráf egy P sétáját* Euler-sétának *nevezzük, ha az G minden élét tartalmazza. Ha P ráadásul körséta, akkor P-t* Euler-körsétának *mondjuk.*

Mivel a sétákban és a körsétákban az élek nem ismétlődhetnek (lásd az 1.10., illetve az 1.20. Definíciót), ezért az Euler-séták (és így az Euler-körséták is) a gráf minden élét pontosan egyszer tartalmazzák.

Egyes tankönyvek *Euler-útnak*, illetve *Euler-körnek* hívják a fenti definícióban bevezetett fogalmakat. Ezek hagyományos elnevezések, de nem illeszkednek az ebben a jegyzetben (az 1.10 és az 1.20. Definíciókban) bevezetett terminológiához, hiszen az Euler-séták nem utak és az Euler-körséták nem körök (érdektelen esetektől eltekintve). Ezért mi a továbbiakban maradunk a 4.1. Definíció elnevezéseinél.

Euler-séták és körséták keresése (illetve a létezésük eldöntése) a fenti mellett sok más rejtvényben is előkerül, például amikor különféle rajzokat egy vonallal, a ceruza felemelése nélkül kell lerajzolni. Emellett számos gyakorlati alkalmazása is van a feladatnak. Ha például egy város teljes úthálózatát kell bejárnunk – mondjuk azért, hogy felmérjük az utak állapotát vagy hogy utcaképeket készítsünk –, akkor a várost reprezentáló összefüggő gráfban egy olyan élsorozatot keresünk, ami a gráf minden élét tartalmazza; ha ezen belül ráadásul a bejárás összhosszát is minimalizálni szeretnénk (hogy az üzemanyaggal takarékoskodjunk), akkor a legjobban akkor járunk, ha a gráfban Euler-sétát találunk és így minden útszakaszon csak egyszer kell áthaladnunk.

> Ha egy összefüggő gráfban nincs Euler-séta, akkor is nyilván létezik a gráf összes élét tartalmazó élsorozatok között legrövidebb. Egy ilyennek a meghatározása már jóval nehezebb feladat egy Euler-séta létezésének az eldöntésénél; bár ismert rá hatékony algoritmus, de ebben a jegyzetben nem foglalkozunk vele. Ennek a feladatnak a neve *Kínai postás probléma* és a megoldása nagyban épít az Euler-sétákkal kapcsolatos ismeretekre, így a 4.2. Tételre.

Térjünk vissza a Königsbergi hidak problémájára: hogyan mutatta meg Euler, hogy nem létezik a városi polgárok igényeinek megfelelő séta – vagyis hogy a 4.1b ábra gráfjában nincs Euler-séta? Tegyük fel, hogy egy tetszőleges (hurokélt nem tartalmazó) G gráfban adott egy P Euler-séta és v olyan csúcsa G-nek, ami nem egyezik meg P-nek sem a kezdőpontjával, sem a végpontjával. Ekkor P párokba rendezi a v-re illeszkedő éleket: valahányszor P áthalad v-n, egy élen belép v-be, egy másikon pedig kilép v-ből. Mivel minden áthaladás kettőt használ el a v-re illeszkedő élek közül és P tartalmazza v minden élét, ezért v foka páros kell legyen. Következik, hogy ha G-ben van Euler-séta, akkor legföljebb két csúcs kivételével minden csúcs foka páros kell legyen G-ben (ahol tehát a két kivételes csúcs a séta kezdő- és végpontja). Mivel azonban a 4.1b ábra gráfjában mind a négy csúcs foka páratlan, ezért valóban nem lehet benne Euler-séta.

A fenti egyszerű, de ravasz gondolatmenet persze nem csak a 4.1b ábra gráfjára használható és Euler sem csak a 4.1a ábrára oldotta meg a feladatot, hanem „szigetek és köztük vezető hidak tetszőleges rendszerére” – vagyis mai megfogalmazás

szerint minden gráfra. Így az alábbi, Leonhard Eulertől, 1735-ből származó tétel is csak szövegezésében tér el az eredetitől. Ennek az erejét és a jelentőségét az adja, hogy kiderül belőle: a csúcsok fokának a paritására vonatkozó feltétel nem csak szükséges, hanem (összefüggő gráfokra) elégséges is.

4.2. Tétel. *Legyen G összefüggő gráf. Ekkor*

(i) G-ben akkor és csak akkor van Euler-körséta, ha G minden csúcsának a foka páros;

(ii) G-ben akkor és csak akkor van Euler-séta, ha G-nek legföljebb két páratlan fokú csúcsa van.

Bizonyítás: Ha G tartalmaz hurokélt, akkor ezeket töröljük G-ből; ezzel sem G csúcsai fokának a paritását nem változtattuk (mert a hurokélek kettővel növelik a fokszámot), sem azt, hogy G-ben van-e Euler-séta vagy körséta (hiszen egy hurokélt beszúrhatunk bármilyen sétába, amikor az éppen a megfelelő csúcsnál tart). Ezért a továbbiakban feltehetjük, hogy G nem tartalmaz hurokélt.

Fentebb már megmutattuk, hogy ha G-ben van egy P Euler-séta, ami nem a v csúcsból indul és nem is oda érkezik, akkor v foka páros; valóban, P-nek a v-n való áthaladásai párosával fogyasztják a v-re illeszkedő éleket és P mindegyiket pontosan egyszer tartalmazza. Ez a gondolat könnyen kiegészíthető azzal, hogy ha P egy Euler-körséta, aminek a kezdő- és végpontja is az u csúcs, akkor u foka is páros. Valóban, ha P első, illetve utolsó éle e, illetve f (vagyis P u-ból e-n indul el és a körséta legvégén f-en érkezik vissza u-ba), akkor továbbra is elmondható, hogy P párba rendezi az u-ra illeszkedő, e-től és f-től különböző éleket, mert minden u-n való áthaladásakor kettőt használ fel ezek közül. Így u-ra páros sok, e-től és f-től különböző él illeszkedik, vagyis a rá illeszkedő élek teljes száma – más szóval: u foka – is páros. Ezzel tehát megmutattuk a tétel *(i)* és *(ii)* állításából a „csak akkor” irányt: ha G-ben van Euler-körséta, illetve séta, akkor teljesül a csúcsok fokának paritására tett állítás.

Az *(i)* állításból a fordított irány bizonyításához leírunk egy algoritmust, ami minden olyan összefüggő gráfban, amiben minden csúcs foka páros, megad egy Euler-körsétát. Az eljárás induláskor választ egy tetszőleges $u \in V(G)$ csúcsot, majd a működése során végig karbantart egy u-ból u-ba vezető P körsétát G-ben. Az algoritmus lényegi része egy olyan lépés lesz, ami P hosszát (vagyis az éleinek a számát) megnöveli. Ennek a lépésnek a (véges sokszori) ismétlésével P előbb-utóbb G minden élét tartalmazni fogja, vagyis Euler-körséta lesz.

A növelő lépés kulcsa egy pofonegyszerű eszköz (amivel az 1.12. Feladat megoldásában már találkoztunk is): egy gráf egy v csúcsából indulva „vaktában, elakadásig sétálunk”. Ez alatt azt értjük, hogy v-ből indulva bejárunk egy Q sétát úgy, hogy mindig egy tetszőlegesen (avagy „vaktában”) választott, olyan élen lépünk tovább az aktuális csúcsból, ami még nem szerepelt Q-ban; ha pedig ilyen él már nincs (vagyis „elakadtunk”), akkor megállunk. Az eljárás helyes működéséhez szükségünk lesz az alábbi lemmára.

4.3. Lemma. *Tegyük fel, hogy a H gráfban minden csúcs foka páros és legyen $v \in V(H)$ tetszőleges csúcs. Ha v-ből indulva vaktában, elakadásig sétálunk, akkor az így kapott Q séta végpontja szintén v.*

A Lemma bizonyítása: Hasonlóan az 1.12. Feladat megoldásához, most is elmondhatjuk, hogy Q párosával fogyasztja az általa érintett, v-től különböző csúcsok éleit; vagyis egy csúcson való minden áthaladása kettőt használ fel az arra illeszkedő élek közül. Ha tehát Q éppen az $x \neq v$ csúcsnál tart, akkor x élei közül korábban páros sokat használt fel (pontosan kétszer annyit, mint ahányszor eddig x-en áthaladt) és még egyet most, amikor x-be legutóbb megérkezett; összesen tehát páratlan sokat. Mivel azonban x foka páros, ezért a rá illeszkedő élek között kell legyen Q által nem tartalmazott is. Így Q valóban nem akadhat el x-ben. ◇

Az alábbi algoritmus a fenti lemmára alapozva, vaktában bejárt körséták ismételt keresésével és azoknak az egyre bővülő P körsétába való befűzésével állít elő egy Euler-körsétát. Ehhez mindig egy olyan v csúcsot választ a következő, vaktában bejárandó séta kezdőpontjának, aminek az élei között van P-beli és nem P-beli is. Ezután v-ből indulva a P-be nem tartozó élek gráfjában keresi a következő, P-be befűzendő körsétát. Az eljárás működéséhez így P-n kívül szükség van ennek a v kiinduló csúcsnak a tárolására is.

ALGORITMUS EULER-KÖRSÉTA KERESÉSÉRE

Bemenet: Egy összefüggő $G = (V,E)$ gráf, amiben minden csúcs foka páros

1 $u \in V$ legyen tetszőleges csúcs
2 $P \leftarrow (u)$ (egy csúcsú, nulla hosszúságú séta); $v \leftarrow u$
3 **ciklus**
4 SÉTÁLJUNK VAKTÁBAN, ELAKADÁSIG v-ből indulva a P-be nem tartozó élek (és G összes csúcsa) által alkotott gráfban, a kapott séta: Q
5 P-ben v egy előfordulásának a helyére szúrjuk be Q-t
6 **ha** P tartalmazza G minden élét, **akkor**: **stop**
7 v legyen olyan csúcs, amire illeszkedik P-beli és nem P-beli él is
8 **ciklus vége**

Tegyük fel, hogy az algoritmus éppen a 4. sort hajtja végre és jelölje H, illetve G_P a P-be nem tartozó, illetve a P-be tartozó élek (és G összes csúcsa) által alkotott részgráfot. Mivel G_P-ben van Euler-körséta (hiszen P nyilván az), ezért G_P-ben minden csúcs foka páros (ezt fentebb, a bizonyítás elején már beláttuk). Ebből következik, hogy H-ban is minden csúcs foka páros, mert minden x csúcs esetén az x-re illeszkedő, összesen páros sok G-beli él közül a páros sok G_P-belit elhagyva kapjuk az x-re illeszkedő, H-beli éleket. Ebből a 4.3. Lemma szerint következik, hogy Q valóban körséta, így amikor azt az 5. sor végrehajtásakor beszúrjuk P-be, akkor P továbbra is séta marad (és persze körséta is).

Meg kell még mutatnunk, hogy az eljárás a 7. sor végrehajtásakor mindig talál olyan csúcsot, amire illeszkedik P-beli és nem P-beli él is. Mivel a 7. sor minden

végrehajtásakor P-nek legalább egy éle már van, de még nem Euler-körséta, ezért választhatunk egy P-be tartozó e és egy P-be nem tartozó f élt. Legyen továbbá x, illetve y az e, illetve az f egyik végpontja. Ha most $x = y$, akkor készen vagyunk. Ha nem, akkor G-ben van egy x-ből y-ba vezető R út, mert G összefüggő (ezt ugyanis a tétel kimondásakor feltettük). Ha R minden éle P-beli, akkor y jó választás, mert van P-beli és nem P-beli éle is. Ha nem, akkor R-en x-től y felé haladva álljunk meg az első, P-be nem tartozó élnél; ennek a kezdőpontja (vagyis R mentén x-hez közelebbi végpontja – ami akár x is lehet) valóban olyan csúcs kell legyen, aminek van P-beli és nem P-beli éle is.

Megmutattuk tehát, hogy az algoritmus helyesen működik: az általa karbantartott P mindig egy u-ból u-ba vezető körséta, aminek a hossza a ciklusmag minden végrehajtásakor megnő (ahol az utóbbi állítás azért igaz, mert a 4. sorban meghatározott Q pozitív hosszúságú, hiszen v-re illeszkedik P-be nem tartozó él is). Ezzel tehát a tétel *(i)* állításának a bizonyítása teljes.

Hátravan még a *(ii)* állításból az „akkor” irány megmutatása. Tegyük fel ezért, hogy G összefüggő és legföljebb két páratlan fokú csúcsa van. Ha nincs páratlan fokú csúcsa, akkor a fentiek szerint van benne Euler-körséta (ami nyilván egyben Euler-séta is). A páratlan fokú csúcsok száma nem lehet egy, mert ez ellentmondana az 1.3. Tételnek (ugyanis a fokszámok összege G-ben páratlan volna). Marad tehát az az eset, amikor G-nek pontosan két páratlan fokú csúcsa van, jelölje ezeket x és y. Adjunk hozzá G-hez egy új, x és y közötti e élt (akkor is, ha esetleg x és y már szomszédosak voltak G-ben). A kapott G' gráfban már minden csúcs foka páros, hiszen x és y foka is megnőtt eggyel. Továbbá G' összefüggő is, hiszen G az volt és e hozzáadása ezt nyilván nem ronthatta el. Ezért a tételnek a fentebb már bebizonyított *(i)* állítása miatt G'-ben van Euler-körséta. Ebből pedig e-t törölve egy (x-ből y-ba vezető) Euler-sétát kapunk G-ben. (Valóban: ha adott egy P körséta és annak egy e éle, akkor P könnyen átalakítható úgy, hogy az e-n induljon el annak az egyik végpontjából és ugyanebben a csúcsban is érjen véget; ehhez csak annyit kell tenni, hogy P-t a megfelelő ponttól indulva járjuk be, majd a végére érve az elejétől folytatjuk ugyaneddig a pontig. Ha ezt az átalakítást elvégezzük a G'-beli Euler-körsétára, akkor ebből az első élt levágva kapjuk a megfelelő Euler-sétát.) □

> Az érdekesség kedvéért megemlítjük, hogy Euler eredeti, 1735-ös (latin nyelvű) cikkében nincs benne a 4.2. Tétel teljes bizonyítása: Euler csak a fokszámra vonatkozó feltételek szükségességét látta be, az elégségességet bizonyítás nélkül mondta ki. Az első teljes értékű bizonyítást több, mint egy évszázaddal később adta Carl Hierholzer (1840 – 1871) német matematikus – de ez már csak a halála után, 1873-ban jelent meg nyomtatásban. Hierholzertől származik a fenti bizonyításban szereplő algoritmus is.

Felhívjuk rá a figyelmet, hogy a fenti tételben G összefüggősége is fontos feltétel, anélkül a tétel állítása hamis volna. Ha például a G gráf több különálló körből áll (pontosabban: mindegyik komponense kör), akkor minden csúcsának a foka kettő, még sincs benne Euler-körséta (hiszen a komponensek között nincs átjárás). Az viszont nem teljesen igaz, hogy Euler-séta csak összefüggő gráfban lehet – csak majdnem. Ugyanis izolált csúcsok hozzáadása (vagy törlése) nem befolyásolja egy

Euler-séta (vagy Euler-körséta) létezését. Valóban, egy Euler-sétának nem kell eljutnia az izolált csúcsokhoz (hiszen a gráf minden élét enélkül is tartalmazhatja). Így ha egy Euler-sétát tartalmazó gráfhoz hozzáadunk néhány izolált pontot, akkor olyan gráfot kapunk, ami továbbra is tartalmaz Euler-sétát, de már nem összefüggő. Az viszont már valóban igaz (és nyilvánvaló), hogy ha G-ben van Euler-séta, akkor legföljebb egy olyan komponense lehet, ami nem egyetlen izolált pontból áll.

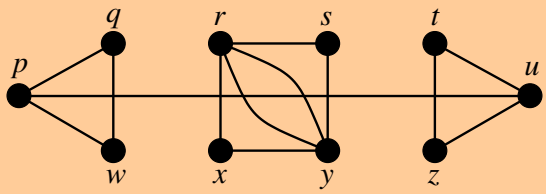
Fontos kiemelni, hogy a 4.2. Tétel miatt tetszőleges gráfban hatékonyan eldönthető Euler-séta, illetve Euler-körséta létezésének a kérdése: ehhez a fokszámok paritásának az ellenőrzésén kívül csak a gráf összefüggőségét kell megvizsgálni (az esetleg létező izolált pontok törlése után), ami megtehető például a 2. fejezetben látott BFS algoritmussal. Ráadásul ha az derül ki, hogy a gráf tartalmaz Euler-sétát, akkor egy ilyen hatékonyan meg is határozható, ugyanis a 4.2. Tétel bizonyításában látott algoritmus polinomiális futásidejű eljárás.

Ha pontosabbat szeretnénk mondani ennek az algoritmusnak a futásidejéről, akkor először is figyeljük meg, hogy itt ismét érdemes a bemenetet adó gráfot szomszédsági listával tárolni, mert ez jól támogatja a vaktában bejárandó séták gyors keresését. Ezzel az algoritmus hatékonyan implementálható úgy, hogy a futása során végig minden v csúcsra a v-hez tartozó lista első néhány tagja a v-re illeszkedő, P-beli élek végpontjait sorolja fel és ezután következnek a v-re illeszkedő, P-be nem tartozó élek végpontjai. Így minden csúcshoz elég fenntartani egy mutatót, ami a csúcs listájában az utolsó P-beli él végpontjára mutat (kezdetben pedig kitöltetlen). Ezekkel a mutatókkal az eljárás 4. sorának a végrehajtása, vagyis a P-be nem tartozó élek gráfjában a Q séta vaktában való bejárása nagyon egyszerűvé válik: a séta során az aktuális csúcs mutatóját mindig a lista következő tagjára állítjuk, a megfelelő éllel bővítjük Q-t, majd az él végpontjára lépve folytatjuk; ha pedig olyan csúcshoz érünk, aminek a mutatója már a listájának az utolsó elemére mutat, akkor a séta elakadt. Ezzel az algoritmus teljes futása alatt mindegyik csúcs listáján egyszer léptetjük végig ezt a mutatót, ezért n csúcsú és m élű gráfra a Q séták keresése összességében $c_1 \cdot (n+m)$-mel járul hozzá a lépésszámhoz (valamilyen c_1 konstansra).

Megoldást kell még találni arra is, hogy ebben az implementációban a 7. sor végrehajtásakor hogyan lehet hatékonyan megtalálni egy olyan csúcsot, amire illeszkedik P-beli és nem P-beli él is. Megtehetnénk, hogy minden ilyen esetben végigpásztázzuk a gráf csúcsait és mindegyikről a megfelelő mutató kiolvasásával döntünk (hiszen egy csúcs akkor alkalmas választás, ha a mutatója kitöltött és nem a listájának az utolsó elemére mutat), de ezzel az algoritmus lépésszámát n^2-tel arányosra ronthatnánk. Ennél jóval hatékonyabb, ha az aktuálisan tárolt P körséta mentén haladva keressük meg az első olyan csúcsot, amire illeszkedik P-be nem tartozó él is. Ehhez egyetlen további mutatót kell fenntartanunk, ami P-nek arra a pontjára mutat, ahonnan a legutóbbi Q körséta vaktában való bejárását kezdtük. Így a mutató által kijelölt csúcs előtt P-ben csak olyan csúcsok lehetnek, amiknek már minden éle P-beli – vagyis P-nek ez a kezdőszakasza már végleges, ide később sem szúródhat be további körséta. Ezért a 7. sor legközelebbi végrehajtásakor elég a mutató által kijelölt csúcstól folytatni a (közben kibővült) P bejárását. Mivel P teljes hossza az algoritmus futásának végére is csak a gráf élszámáig növekszik, ezért a 7. sor végrehajtásai összesen szintén csak $c_2 \cdot (n+m)$-mel járulnak hozzá a lépésszámhoz (egy c_2 konstansra). Ezzel tehát az eljárás teljes lépésszáma is $c \cdot (n+m)$-mel felülről

becsülhető (valamilyen c konstansra), így az nagyon hatékony, lineáris futásidejű algoritmus.

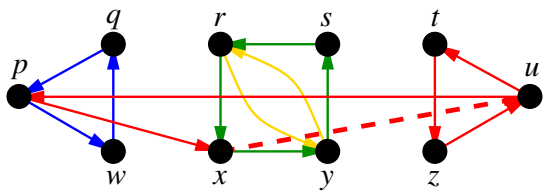
4.4. Feladat.
a) Tartalmaz-e az alábbi G gráf Euler-sétát, illetve Euler-körsétát? Ha a válasz igen, akkor adjunk meg egyet.
b) Adjuk hozzá a gráfhoz a $\{p,x\}$ élt és oldjuk meg a feladatot az így kapott gráfra is.



Megoldás: G-ben a p és u csúcsok foka 3, az összes többi csúcs foka viszont páros (2 vagy 4). Mivel G-nek van páratlan fokú csúcsa, ezért Euler-körséta nincs benne. Az ugyan teljesül G-re, hogy legföljebb két páratlan fokú csúcsa van – ennek ellenére nincs benne Euler-séta sem, ugyanis G nem összefüggő. Valóban, az $\{r,s,x,y\}$, illetve a $\{p,q,t,u,w,z\}$ csúcshalmazok által feszített részgráfok két külön komponenst alkotnak. Mivel mindkét komponens tartalmaz élt (vagyis egyik sem csupán egyetlen izolált pontból áll), ezért G-ben nincs Euler-séta.

A $\{p,x\}$ él hozzáadása már összefüggővé teszi G-t (hiszen kapcsolatot teremt a fenti két komponens között). Továbbá p és x foka is eggyel nő, így a kapott gráfban u és x foka lesz páratlan (mindkettő 3), a többi csúcs foka páros. Ezért a 4.2. Tétel szerint a gráf továbbra sem tartalmaz Euler-körsétát, de Euler-sétát már igen. Egy ilyennek a meghatározásához a fenti bizonyításban leírt algoritmust hívjuk segítségül. Ehhez először (a bizonyítás utolsó bekezdésében írtak szerint) hozzáadjuk a gráfhoz az $\{x,u\}$ élt (a 4.2. ábrán szaggatott vonallal), majd a kapott gráfban – amiben már minden csúcs foka páros – futtatjuk az algoritmust.

Indítsuk az eljárást az u csúcsból, vagyis az 1-2. sorokban írtak szerint P kezdetben a csak az u csúcsból álló, él nélküli séta és $v = u$. A 3. sorban írtak szerint u-ból vaktában bejárunk egy Q_1 sétát; ez persze többféleképpen történhet, az egyik lehetőség az, hogy Q_1 sorra az (u,t,z,u,p,x,u) csúcsokat érinti (és a bizonyításban látottak szerint valóban u-ban akad el); ezt látjuk a 4.2. ábrán pirossal.



4.2. ábra

Ezt a Q_1 sétát P-be u helyére beszúrva nyilván Q_1-et kapjuk, így most $P = Q_1$. Mivel P nem tartalmaz minden élt, ezért az eljárás 7. sorában írtak szerint választunk egy olyan v csúcsot, amire illeszkedik P-beli és nem P-beli él is: például lehet $v = p$. Most p-ből indulva a P-be nem tartozó (vagyis a 4.2. ábrán nem piros) éleken sétálunk vaktában, elakadásig; így kaphatjuk például az ábrán kékkel látható Q_2 körsétát, ami sorra a (p,w,q,p) csúcsokat érinti. Ezt a p csúcs helyére P-be beszúrva az most sorra az (u,t,z,u,p,w,q,p,x,u) csúcsokat érinti. Tovább folytatva, most választhatjuk például a $v = x$ csúcsot, mert erre illeszkedik P-beli és nem P-beli él is. x-ből a P-be nem tartozó (vagyis nem piros vagy kék) éleken vaktában bejárhatjuk azt a Q_3 körsétát, ami sorra az (x,y,s,r,x) csúcsokat érinti (az ábrán zölddel). Ezt beszúrva P-be x helyére az most sorra az $(u,t,z,u,p,w,q,p,x,y,s,r,x,u)$ csúcsokat érinti. Most P-ből még mindig hiányzik a két r és y közötti él, amiket a $v = y$ csúcsból indított, az ábrán sárgával jelölt Q_4 sétával járhatunk be, ami tehát sorra az (y,r,y) csúcsokat érinti. Q_4-nek az y helyére való beszúrása után P már minden élt tartalmaz, így Euler-körséta; P tehát végül sorra a $(u,t,z,u,p,w,q,p,x,y,r,y,s,r,x,u)$ csúcsok érintésével járja be a gráf összes élét. Ebből a körsétából az utolsó, $\{x,u\}$ él törlésével kapjuk a feladatban szereplő gráf Euler-sétáját. □

4.5. Feladat. Egy salsa találkozón 10 fiú és 10 lány vesz részt, mindenki pontosan 6 embert ismer az ellenkező neműek közül (az ismeretségek kölcsönösek). A résztvevők a következőt játsszák: egy fiú kiválasztja egy lányismerősét és felkéri salsázni; az illető lány a tánc után kiválasztja egy másik fiúismerősét és felkéri salsázni, stb. A szabály tehát az, hogy akit legutóbb felkértek, az az ellenkező nemű ismerősei közül egy olyat kell felkérjen, akivel (ebben a játékban) még nem táncolt. (Akit felkérnek, az mindig el is fogadja a felkérést.) A társaság célja az, hogy végül mindenki elmondhassa magáról, hogy a játék során minden (ellenkező nemű) ismerősével pontosan egyszer salsázott. Mutassuk meg, hogy ez a cél megvalósítható.

Megoldás: A találkozó résztvevői alkossák a G gráf csúcshalmazát és két csúcs akkor legyen szomszédos G-ben, ha a két megfelelő táncos ellenkező nemű és ismerik egymást. A feladat állítása nem más, mint hogy G-ben van Euler-séta; valóban, salsázásra való felkéréseknek egy, a játék szabályainak megfelelő sorozata épp egy G-beli Euler-sétának felel meg.

Mivel mindenki 6 ellenkező nemű résztvevőt ismer, ezért G-ben minden csúcs foka 6, vagyis páros. Meg kell még mutatnunk, hogy G összefüggő; legyen ezért K egy tetszőleges komponens G-ben és v annak egy csúcsa. Ha mondjuk v fiú, akkor a 6 lányismerőse K-ba tartozik (hiszen elérhetők v-ből); ha ezek közül a lányok közül az egyik w, akkor az ő 6 fiúismerőse (köztük v) szintén K-ba tartozik, hiszen ezek is elérhetők v-ből. Azt kaptuk tehát, hogy K-nak legalább 12 csúcsa van. Mivel ez G minden komponenséről elmondható, de G-nek összesen csak 20 csúcsa van, ezért nem lehet egynél több komponense, vagyis valóban összefüggő.

Mivel G összefüggő és minden csúcsának a foka páros, ezért a 4.2. Tétel szerint valóban van benne Euler-séta (sőt, Euler-körséta is). □

A fejezet zárásául röviden megemlítjük, hogy Euler-séták és Euler-körséták létezésének a problémája irányított gráfokra is felvethető és ez is felmerül alkalmazásokban. Itt tehát egy irányított gráf éleit kell úgy bejárni, hogy közben minden élen pontosan egyszer haladunk át (természetesen az él irányításának megfelelően). Szerencsére ez a probléma sem nehezebb az irányítatlan esetnél: a 4.2. Tétel állítása és bizonyítása – beleértve az abban leírt algoritmust is – könnyen módosítható az irányított változatra is. Nem nehéz végiggondolni például, hogy a 4.2. Tétel *(i)* állításában szereplő feltételnek az irányított Euler-körsétákra vonatkozó változata azt mondja, hogy minden csúcs kifoka meg kell egyezzen a befokával (vagyis a csúcsból kilépő élek száma egyenlő kell legyen az oda belépő élek számával). Ehhez hasonló az irányított Euler-sétákra vonatkozó fokszámfeltétel is (de ezt itt nem részletezzük). Végül az összefüggőségi feltétel sem okozna problémát az irányított esetben: bár az 1.6. szakaszban említettük, hogy az összefüggőség fogalmának az irányított megfelelője nem magától értetődő, de a 4.2. Tétel irányított változatában elég volna azt kikötni, hogy a szóban forgó gráfból az élek irányításának figyelmen kívül hagyásával kapott irányítatlan gráf összefüggő legyen.

5. fejezet

Hamilton-körök és Hamilton-utak

Az előző fejezetben tárgyalt Euler-séták és -körséták problémáját olyan alkalmazások motiválták, amikben minden él pontosan egyszeri érintésével kell bejárni egy gráfot. Az alkalmazások szempontjából még fontosabbak az olyan bejárások, amik során minden csúcsot kell pontosan egyszer érinteni; ezek az ír William Rowan Hamiltonról (1805 – 1865) kapták a nevüket.

5.1. Definíció. *A G gráf egy C körét* Hamilton-körnek *nevezzük, ha C G minden csúcsát tartalmazza. Hasonlóan, egy P utat* Hamilton-útnak *nevezünk G-ben, ha P G minden csúcsát tartalmazza.*

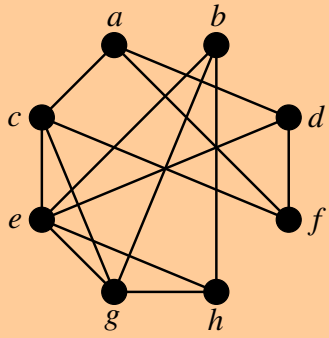
Mivel a kör és az út definíciója (lásd az 1.10., illetve az 1.20. Definíciót) magában foglalja, hogy ezekben a csúcsok nem ismétlődhetnek (eltekintve persze körök esetében az első és utolsó csúcs egybeesésétől), ezért a Hamilton-körök és -utak valóban pontosan egyszer érintik a gráf minden csúcsát.

Hamilton-körök vagy -utak keresése számtalan alkalmazásban merül fel. Tegyük fel például, hogy egy üzletkötőnek a munkája miatt végig kell látogatnia néhány várost, majd visszatérni az otthonába. A városok érintésének a sorrendje érdektelen, de az útiköltség persze nem. Ha ismert bármely két város esetén a köztük való utazás költsége, akkor az ügynök olyan Hamilton-kört keres a városok és a köztük lévő kapcsolatok gráfjában, aminek az összköltsége (vagyis a Hamilton-körben szereplő élek útiköltségeinek az összege) minimális. Ez a feladat *Utazóügynök probléma* néven vált közismertté.

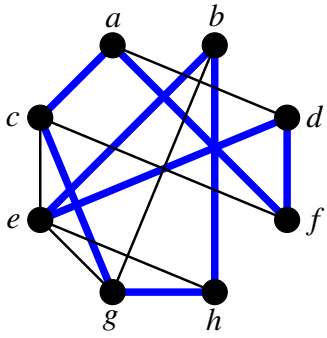
A Hamilton-körök és -utak problémája sokkal nehezebb az ezekhez hasonlónak tűnő Euler-körséták és -séták feladatánál. Míg az utóbbiak létezésére a 4. fejezetben könnyen ellenőrizhető szükséges és elégséges feltételt adtunk, a keresésükre pedig hatékony algoritmust mutattunk, addig a Hamilton-körök és -utak esetében egyiket sem tudjuk megtenni. Nem ismert polinomiális futásidejű algoritmus annak az

eldöntésére, hogy egy adott gráfban létezik-e Hamilton-kör vagy Hamilton-út; sőt, nem csak nem ismert, hanem nagyon valószínű (bár nem bizonyított), hogy nem is létezik ilyen. (Ennek az állításnak a pontos tartalmáról az informatikus képzés későbbi tárgyaiban esik szó.) Így azokban az alkalmazásokban, amikben Hamilton-körök vagy -utak keresése válik szükségessé (mint például az előző bekezdésben említett utazóügynök probléma), vagy csak viszonylag kis méretű gráfokra tudják megoldani a feladatot, vagy olyan eljárásokat használnak, amik csak egy bizonyos (és pontosan nem is feltétlen ismert) hibahatáron belül és esetleg csak a lehetséges bemenetek egy részére oldják meg a feladatot elfogadható futásidőn belül.

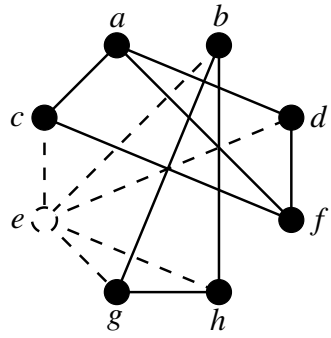
5.2. Feladat.
a) Van-e Hamilton-kör, illetve Hamilton-út az alábbi G gráfban?
b) Töröljük G-ből a $\{c,g\}$ élt és oldjuk meg a feladatot az így kapott gráfra is.



Megoldás: Némi kísérletezés árán találhatunk G-ben Hamilton-kört: például ha a csúcsokat (c,a,f,d,e,b,h,g,c) sorrendben járjuk be (lásd az 5.1a ábrát). Mivel egy Hamilton-kör bármelyik élét törölve Hamilton-utat kapunk, ezért G nyilván Hamilton-utat is tartalmaz.



5.1a ábra



5.1b ábra

Ugyanez mutatja, hogy a $\{c,g\}$ él törlésével kapott gráf tartalmaz Hamilton-utat: a fenti Hamilton-körből a $\{c,g\}$ él elhagyásával ilyet kapunk. Hamilton-

kört azonban ez a gráf már nem tartalmaz. Ennek a bizonyítására használhatjuk az 5.1b ábra által mutatott ötletet: ha töröljük a gráfból az e csúcsot, akkor a maradék gráf már nem összefüggő, az $\{a,c,d,f\}$ és a $\{b,g,h\}$ csúcshalmazok két külön komponenst feszítenek. Ez a megfigyelés valóban mutatja, hogy a gráfban nincs Hamilton-kör: ha volna, akkor abból az e csúcs törlésével a maradék gráf egy Hamilton-útját kapnánk. Ez azonban lehetetlen, mert egy Hamilton-utat tartalmazó gráfnak nyilván összefüggőnek kellene lennie. □

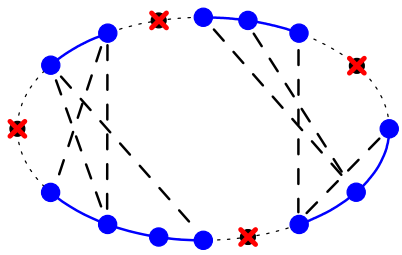
A fenti megoldásban egy leleményes ötlet vezetett annak az indoklásához, hogy a szóban forgó gráfban nincs Hamilton-kör: ha volna, akkor bármelyik csúcsot törölve összefüggő gráfot kapnánk (mert a maradék gráfban volna Hamilton-út). Ennek az ötletnek a továbbfejlesztését tartalmazza az alábbi tétel, ami bizonyos, szerencsés esetekben – de egyáltalán nem mindig – annak a bizonyítására használható, hogy egy gráfban nincs Hamilton-kör vagy Hamilton-út.

5.3. Tétel. *Legyen G tetszőleges gráf és $k \geq 1$ egész szám.*

(i) Ha G-ben van Hamilton-kör, akkor G-ből bárhogyan k csúcsot törölve a kapott gráf komponenseinek száma legföljebb k.

(ii) Ha G-ben van Hamilton-út, akkor G-ből bárhogyan k csúcsot törölve a kapott gráf komponenseinek száma legföljebb $k+1$.

Bizonyítás: (i) bizonyításához legyen C Hamilton-kör G-ben és töröljünk G-ből k tetszőleges csúcsot. Ekkor a törölt csúcsok C-t legföljebb k ívre bontják – vagyis C-ből legföljebb k olyan út keletkezik, amik a törlések előtt C egyes szakaszait alkották. Ezt illusztrálja az 5.2. ábra, amin négy csúcs törlése a Hamilton-kört négy ívre bontja (az ábrán kékkel). (A keletkező ívek száma pontosan k, ha a törölt csúcsok közül semelyik kettő nem volt C mentén szomszédos, mint az 5.2. ábrán; ha viszont voltak köztük C mentén szomszédosak, akkor az ívek száma kisebb k-nál.)



5.2. ábra

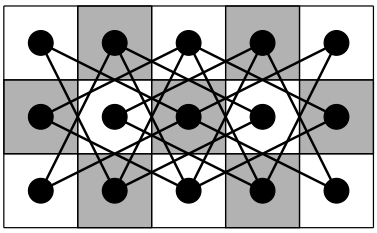
A törlések után kapott G' gráf nyilván tartalmazza a C-ből megmaradt éleket (valamint G összes többi olyan élét is, amik nem illeszkednek az elhagyott csúcsokra – ezeket jelzik az 5.2. ábrán a szaggatott vonalak). Így G'-ben az egy ívre eső csúcsok közös komponensbe tartoznak, hiszen az ív mentén vezet köztük út. Ezért G' komponenseinek száma nem lehet több az ívek számánál, így k-nál sem.

(ii) bizonyítása ehhez nagyon hasonló: egy G-beli P Hamilton-utat k csúcs törlése legföljebb $k+1$ szakaszra bont. Mivel az egy szakaszra eső csúcsok közös komponensbe tartoznak a törlések utáni gráfban, ezért annak nem lehet $(k+1)$-nél több komponense. □

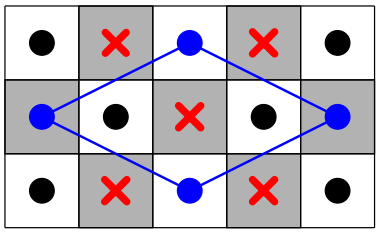
Fontos hangsúlyozni, hogy az 5.3. Tétel egyik állítása sem megfordítható: létezik például olyan gráf, amiből bárhogyan néhány csúcsot törölve a kapott gráf komponenseinek száma nem több a törölt csúcsok számánál, de a gráfban még sincs Hamilton-kör. (Bizonyítás nélkül megemlítjük, hogy ilyen például az 1.9. Feladatban, annak is az első két ábráján látott Petersen-gráf.) Így az 5.3. Tétel nem alkalmas arra, hogy a segítségével Hamilton-kör vagy Hamilton-út létezését mutassuk ki, csak ezeknek a nemlétét lehet – szerencsés esetekben – bizonyítani vele: ha egy G gráfból sikerül k csúcsot törölni úgy, hogy a kapott gráf komponenseinek száma k-nál több, akkor G-ben biztosan nincs Hamilton-kör; ha pedig a kapott komponensek száma $(k+1)$-nél is több lehet, akkor Hamilton-út sincs G-ben. Ez történt az 5.2. Feladat b) részében is: ott $k=1$ csúcs elhagyása után a komponsensek száma egynél több volt, amiből megtudtuk, hogy a gráfban nincs Hamilton-kör.

5.4. Feladat. Bejárható-e egy 3×5-ös sakktábla lóval úgy, hogy közben minden mezőre pontosan egyszer lépünk rá?

Megoldás: Fogalmazzuk át a kérdést gráfelméleti feladattá: legyenek a G gráf csúcsai a 3×5-ös sakktábla mezői és két csúcs akkor legyen szomszédos G-ben, ha a megfelelő mezők egymásból egyetlen lólépéssel elérhetők (lásd az 5.3a ábrát); a kérdés ekkor az, hogy G-ben van-e Hamilton-út.



5.3a ábra



5.3b ábra

Hagyjuk el G-ből az 5.3b ábrán piros X-szel jelölt mezőknek megfelelő öt csúcsot. A keletkező gráfnak hét komponense van: egy négy csúcsú kör (az ábrán kékkel) és hat izolált pont. Ezért az 5.3. Tétel *(ii)* állítása szerint G-ben nincs Hamilton-út; valóban, ha volna, akkor G-ből $k=5$ csúcsot elhagyva a kapott gráfnak legföljebb $k+1=6$ komponense lehetne. Így a 3×5-ös sakktáblát nem lehet a feladat által írt módon bejárni. □

Az 5.3. Tétel szükséges feltételt ad Hamilton-kör, illetve Hamilton-út létezésére – de, mint azt fentebb már említettük, ezek nem elégségesek; sőt, nem is ismert jól kezelhető szükséges és elégséges feltétel Hamilton-kör vagy út létezésére. Ismertek

viszont olyan tételek, amik elégséges (de nem szükséges) feltételt adnak Hamilton-kör létezésére; ilyen az alábbi tétel is, amit a magyar születésű brit matematikus, Gabriel Andrew Dirac (1925 – 1984) bizonyított be 1952-ben.

5.5. Tétel. (Dirac tétele)
Ha az $n \geq 3$ csúcsú G egyszerű gráfban minden csúcs foka legalább $\frac{n}{2}$, akkor G-ben van Hamilton-kör.

Ha tehát például egy 100 csúcsú egyszerű gráf minden csúcsa legalább 50 másikkal szomszédos, akkor Dirac tétele szerint biztosan van benne Hamilton-kör. Érdemes megfigyelni, hogy ugyanez 50 helyett 49-cel már nem volna igaz: ha a G gráf két 50 csúcsú teljes gráf egyesítéséből áll (pontosabban: mindkét komponense 50 csúcsú teljes gráf), akkor G-ben minden fokszám 49, de nyilván nincs benne Hamilton-kör (hiszen nem összefüggő). Ez tehát azt mutatja, hogy a csúcsok fokára vonatkozó alsó korlátot 1-gyel csökkentve a tétel már nem volna igaz – ennek ellenére, amint azt fentebb említettük, a Dirac tételében szereplő feltétel valóban csak elégséges, de nem szükséges: ha például a G gráf egy legalább 5 csúcsú kör, akkor nyilván van benne Hamilton-kör, de bármely csúcsának a foka csak 2, vagyis kisebb a csúcsszám felénél.

Dirac tétele helyett egy annál általánosabb tételt bizonyítunk be (amiből tehát Dirac tétele is következni fog); ezt a norvég Øystein Ore fedezte fel 1960-ban.

5.6. Tétel. (Ore tétele)
Tegyük fel, hogy az $n \geq 3$ csúcsú G egyszerű gráfra teljesül a következő feltétel: minden $u, v \in V(G)$, $\{u,v\} \notin E(G)$ esetén $d(u) + d(v) \geq n$; vagyis bármely két nemszomszédos csúcs fokszámának az összege legalább G csúcsainak a száma. Ekkor G-ben van Hamilton-kör.

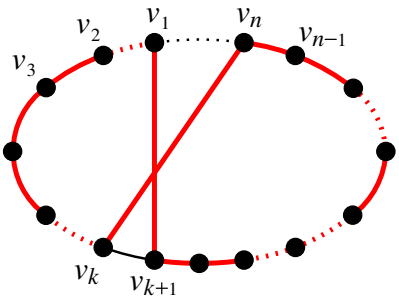
Részletesebben tehát Ore tétele a következőt állítja: ha az $n \geq 3$ csúcsú G egyszerű gráfban minden nemszomszédos $\{u,v\}$ csúcspárt végigvizsgálva mindig azt tapasztaljuk, hogy $d(u) + d(v) \geq n$, akkor G-ben biztosan van Hamilton-kör. Ha viszont G-ben van olyan nemszomszédos $\{u,v\}$ csúcspár, amire $d(u) + d(v) < n$, akkor a tétel semmit nem mond arról, hogy G-ben van-e Hamilton-kör vagy sem. Más szóval: ennek a tételnek az állítása sem megfordítható, a benne szereplő feltétel csak elégséges, de nem szükséges. Valóban, erre is jó példa egy legalább 5 csúcsú kör: van benne Hamilton-kör, de bármely két csúcsának a fokát összeadva csak 4-et, vagyis a csúcsszámnál kisebbet kapunk.

Az 5.6. Tétel bizonyítása: Tegyük fel, hogy a G gráf teljesíti a tételben írt feltételt, vagyis $d(u) + d(v) \geq n$ igaz minden nemszomszédos $\{u,v\}$ csúcspárra (és G csúcsszáma $n \geq 3$). Megadunk egy eljárást, ami előállít G-ben egy Hamilton-kört és ezáltal a tételt igazolja. (Ez az eljárás csak a bizonyítás céljait szolgálja, a gyakorlatban közvetlenül nem alkalmazzák, ezért pszeudokód formájában nem írjuk le.)

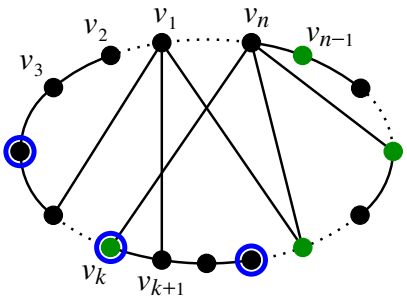
Az eljárás a futása során végig karbantartja G csúcsainak egy $\pi = (v_1, v_2, \ldots, v_n)$ sorrendjét, vagy más néven permutációját. Minden ilyen π permutáció esetén jelöljük $e(\pi)$-vel azt, hogy a $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$ párok között hány

olyan van, amik szomszédosak G-ben. Ha $e(\pi) = n$, vagyis a párok közül mindegyik éle G-nek, akkor a π által kijelölt sorrendben bejárva a csúcsokat nyilván Hamilton-kört kapunk G-ben. Ha viszont $e(\pi) < n$, akkor az eljárás úgy módosítja π-t, hogy ezáltal $e(\pi)$ értéke növekedjen. Egy tetszőleges kezdeti permutációból indulva és ezt a növelő lépést legföljebb n-szer alkalmazva végül $e(\pi) = n$ lesz, így valóban Hamilton-kört kapunk.

Tegyük fel ezért, hogy az aktuális π permutációra $e(\pi) < n$, vagyis a $\{v_1, v_2\}$, $\{v_2, v_3\}$, …, $\{v_{n-1}, v_n\}$, $\{v_n, v_1\}$ párok között még van nemszomszédos. Feltehetjük, hogy például $\{v_n, v_1\} \notin E(G)$, mert nem okoz tartalmi változást (így $e(\pi)$ sem változik), ha a csúcsok indexeit π mentén körben néhányszor egymás után „eltoljuk” – vagyis v_n-ből v_1 lesz és az összes többi v_i indexét eggyel növeljük. Ekkor az $e(\pi)$ növelésére szolgáló lépés alapötlete az 5.4a ábrán látható: keresünk egy olyan k-t – és alább megmutatjuk, hogy ilyet mindig találunk is –, amire $\{v_k, v_n\} \in E(G)$ és $\{v_1, v_{k+1}\} \in E(G)$ egyszerre teljesülnek; egy ilyen k-ra pedig π-t úgy módosítjuk, hogy annak a v_{k+1}-től v_n-ig tartó szakaszát megfordítjuk, vagyis az új permutáció $\pi' = (v_1, v_2, \ldots, v_k, v_n, v_{n-1}, \ldots, v_{k+1})$ lesz.



5.4a ábra



5.4b ábra

Az 5.4a ábra pirossal mutatja, hogy π' szerint haladva milyen sorrendben járjuk körbe a csúcsokat. Látható, hogy π-ről π'-re áttérve a körbejárás mentén egymást követő csúcspárok listája csak két helyen változik: a $\{v_1, v_n\}$ és a $\{v_k, v_{k+1}\}$ párok már nem követik egymást, helyettük a $\{v_1, v_{k+1}\}$ és a $\{v_k, v_n\}$ párok igen. Mivel az utóbbi két csúcspár szomszédos G-ben (hiszen k-t így választottuk meg), viszont a fentiek szerint $\{v_1, v_n\} \notin E(G)$, ezért $e(\pi') > e(\pi)$ valóban igaz. (Pontosabban: ha $\{v_k, v_{k+1}\} \in E(G)$, akkor $e(\pi')$ 1-gyel nagyobb $e(\pi)$-nél, egyébként 2-vel.)

Azt kell tehát még megmutatnunk, hogy $\{v_1, v_{k+1}\} \in E(G)$ és $\{v_k, v_n\} \in E(G)$ teljesül valamilyen k-ra. Ehhez (képzeletben) fessük zöldre v_n összes szomszédját és jelöljük meg kékkel az összes olyan csúcsot, ami v_1 valamelyik szomszédját π-ben eggyel megelőzi (lásd az 5.4b ábrát); más szóval, a v_j csúcs akkor zöld, ha $\{v_j, v_n\} \in E(G)$ és v_j akkor kék, ha $\{v_1, v_{j+1}\} \in E(G)$. Ekkor a zöld csúcsok száma $d(v_n)$, hiszen ez v_n szomszédainak a száma (mert G egyszerű gráf). Hasonlóan, a kék csúcsok száma $d(v_1)$, mert v_1 szomszédai között saját maga nem szerepel (ismét azért, mert G egyszerű gráf), így ezeknek a száma ugyanannyi, mint az ezeket π-ben eggyel megelőző csúcsok száma. Mivel $\{v_1, v_n\} \notin E(G)$, ezért a tétel feltéte-

le szerint $d(v_1)+d(v_n) \geq n$ teljesül; így a kék és zöld színt kapott csúcsok száma együtt legalább n. Másrészt v_n nyilván se nem zöld, se nem kék, mert egyrészt v_n nem lehet saját maga szomszédja (hiszen G egyszerű), másrészt π utolsó tagja nem előzheti meg eggyel v_1 egy szomszédját. Azt kaptuk tehát, hogy legföljebb $n-1$ csúcs közül legalább n-re jutott valamilyen szín, így kell lennie köztük olyannak, ami mindkét színt megkapta. Ha v_k-t ilyennek választjuk, akkor valóban megfelel a feltételeknek: $\{v_1, v_{k+1}\} \in E(G)$, mert v_k kék és $\{v_k, v_n\} \in E(G)$, mert v_k zöld is. Ezzel a tétel bizonyítása teljes. □

Amint azt fentebb ígértük, Ore tételéből Dirac tétele (5.5. Tétel) már valóban könnyen következik.

Az 5.5. Tétel bizonyítása: Ha G-ben minden csúcs foka legalább $\frac{n}{2}$, akkor bármely nemszomszédos $\{u,v\}$ csúcspárra $d(u)+d(v) \geq \frac{n}{2}+\frac{n}{2} = n$. (Sőt, ez a szomszédos csúcspárokra is teljesül, csak ennek itt nincs jelentősége.) Így az 5.6. Tétel szerint G-ben valóban van Hamilton-kör. □

5.7. Feladat. Az ötöslottó játékban egy szelvényen 5 különböző számot kell megjelölni az $1, 2, \ldots, 90$ számok közül. Mutassuk meg, hogy ha az összes lehetséges módon kitöltenénk lottószelvényeket (úgy, hogy minden lehetséges kitöltés pontosan egy szelvényen szerepeljen), akkor az így kapott szelvények elrendezhetők volnának egy sorban úgy, hogy az egymás mellé kerülő szelvényeken soha ne legyen egy közös szám sem megjelölve.

Megoldás: Kezdjük megint azzal, hogy a feladatot átfogalmazzuk gráfelméleti állítássá: a G gráf csúcsai legyenek a lehetséges lottószelvények és két csúcs akkor legyen szomszédos G-ben, ha a megfelelő szelvényeken nincs közös szám megjelölve; a feladat állítása ekkor az, hogy G-ben van Hamilton-út. Ezt az 5.5. Tétel segítségével fogjuk igazolni: megmutatjuk, hogy G-ben minden csúcs foka legalább G csúcsszámának a fele; ebből az fog kiderülni, hogy G-ben nem csak Hamilton-út, de még Hamilton-kör is van.

G-nek $n = \binom{90}{5}$ csúcsa van, mert ennyi az $\{1,2,\ldots,90\}$ halmaz 5 elemű részhalmazainak, vagyis a lehetséges lottószelvényeknek a száma. Ennek az értéke $n = \frac{90\cdot 89\cdot 88\cdot 87\cdot 86}{5!} = 43\,949\,268$. G egy tetszőleges v csúcsát kiválasztva v foka $d(v) = \binom{85}{5}$, mert ennyiféleképpen lehet a v-nek megfelelő szelvényen szereplő 5 számtól különböző 85 szám közül 5 különbözőt kiválasztani; vagyis ennyi azoknak a szelvényeknek a száma, amiken nincs a v-nek megfelelőn szereplő szám megjelölve. Ennek az értéke pedig $d(v) = \frac{85\cdot 84\cdot 83\cdot 82\cdot 81}{5!} = 32\,801\,517$. Látható, hogy $d(v) \geq \frac{n}{2}$ valóban minden csúcsra (bőven) teljesül (és G egyszerű gráf), így Dirac tétele szerint G-ben valóban van Hamilton-kör. □

6. fejezet

Gráfok színezése

Képzeljük el a következő problémát: adott számítási feladatok egy listája és mindegyikhez egy előre kijelölt időintervallum, amikor azt el kell végezni. A feladatokat processzorokhoz kell rendelnünk, de ezek minden időpillanatban egyszerre csak egy feladaton dolgozhatnak. A célunk az, hogy a számítási feladatoknak egy olyan ütemezését készítsük el – vagyis úgy rendeljük őket processzorokhoz –, hogy ehhez a lehető legkevesebb processzorra legyen szükség. A problémát egy G gráffal ábrázolhatjuk: a csúcsok a számítási feladatok és két csúcs akkor szomszédos, ha a megfelelő feladatok időintervallumai metszők – vagyis ha a két feladat között ütközés van, nem kerülhetnek azonos processzorra. A probléma egy lehetséges megoldását, vagyis a feladatok egy lehetséges ütemezését úgy tehetjük szemléletessé, hogy G minden csúcsát kiszínezzük valamilyen színnel és az ugyanolyan színű csúcsoknak megfelelő feladatokat rendeljük közös processzorhoz. Ekkor egy színezés akkor megfelelő, ha a G-ben szomszédos csúcsok mindig különböző színt kapnak – hiszen ez felel meg annak, hogy ha ütközés van két feladat között, akkor azok különböző processzorra kerülnek. A felhasznált processzorok száma pedig a színezésnél használt összes színek számának felel meg, így ezt kell minimalizálnunk.

A fentihez hasonló problémák, amikben egy gráf csúcsainak egy, a fenti értelemben helyes színezését kell megtalálnunk, számos fontos gyakorlati alkalmazásban merülnek fel; leginkább akkor, ha a gráf élei valamilyen fajta ütközést, elkerülendő konfliktust reprezentálnak. Az alábbi definíció az ennek megfelelő alapfogalmakat vezeti be.

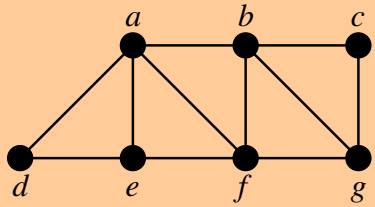
6.1. Definíció. *Legyen G egy gráf és $k \geq 1$ egész szám. A G gráf k* színnel színezhető*, ha G minden csúcsa kiszínezhető k adott (tetszőleges) szín valamelyikével úgy, hogy G bármely két szomszédos csúcsának a színe különböző. A G* kromatikus száma *k, ha G k színnel színezhető, de $(k-1)$-gyel már nem. G kromatikus számának a jele: $\chi(G)$.*

(A χ görög betű, a szín görög megfelelőjének, a $\chi\rho\omega\mu\alpha$ szónak az első betűje; a kiejtése: „khí”.)

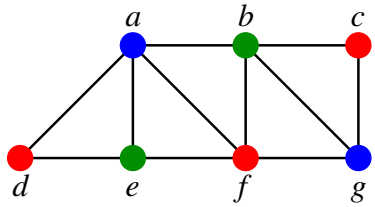
A kromatikus szám vizsgálatakor általában csak egyszerű gráfokkal foglalkozunk: hurokélek eleve nem lehetnek a gráfban, hiszen egy csúcs nyilván nem lehet saját magától különböző színű; párhuzamos éleket ugyan tartalmazhat a gráf, de két csúcs között futó több párhuzamos él közül elég volna mindig csak egyet megtartani annak garantálására, hogy a két csúcs különböző színt kapjon.

Egy gráf kromatikus száma mindig egy pozitív egész szám és könnyű látni, hogy $\chi(G)$ tetszőlegesen nagy értéket is felvehet. Legyen ugyanis $n \geq 1$ egész és tekintsük azt az n csúcsú, egyszerű gráfot, amiben bármely két különböző csúcs szomszédos; ezt a gráfot *n csúcsú teljes gráfnak* nevezzük és K_n-nel jelöljük. Erre például nyilván $\chi(K_n) = n$, mert semelyik két csúcsának nem adhatjuk ugyanazt a színt, hiszen szomszédosak.

6.2. Feladat. Határozzuk meg az alábbi G gráf kromatikus számát, $\chi(G)$-t.



Megoldás: Három színnel könnyen megszínezhetők a gráf csúcsai például úgy, ahogyan a 6.1. ábrán látható.



6.1. ábra

Két szín viszont biztosan nem volna elegendő: mivel (például) az a, d és e csúcsok közül bármely kettő szomszédos, ezért már ezekre is három különböző színt kell használnunk.

Így tehát $\chi(G) = 3$. □

A gráfok színezésének, illetve a kromatikus szám fogalmának a története a 19. század közepéig nyúlik vissza. Ekkorról származik az a megfigyelés, hogy a térképek országai mindig megszínezhetők legföljebb négy színnel úgy, hogy a szomszédos – vagyis közös határszakasszal rendelkező – országokat különböző színűre festjük. Ekkor keletkezett a híres *négyszín-sejtés* is, ami azt állította, hogy ez minden térképre valóban igaz. Ezt a sejtést csak több, mint egy évszázaddal később, 1976-ban sikerült igazolnia Appelnek és Hakennek, de a bizonyításuk sok száz oldalnyi terjedelmű volt és közel kétezer konkrét eset ellenőrzésére is szükség volt

hozzá. Ez utóbbit számítógéppel végezték el, ami akkoriban szokatlan újdonságnak számított és számos kételyt is felvetett a bizonyítás megbízhatóságával kapcsolatban. Bár azóta születtek rövidebb bizonyítások is a négyszín-tételre, ezek is reménytelenül hosszúak és szintén használnak számítógépet.

A térképszínezés feladatában egy olyan gráf csúcsainak a színezéséről van szó, aminek a csúcsai a térkép országai és két csúcs akkor szomszédos, ha a megfelelő két országnak van közös határszakasza. Szemléletesen könnyű látni, hogy minden ilyen gráf *síkbarajzolható*, vagyis lerajzolható a síkba úgy, hogy az éleket reprezentáló vonalak csúcsokon kívül ne keresztezzék egymást. Valóban, ha a térképen minden ország belsejében valahol elhelyezünk egy annak megfelelő pöttyöt és két ilyet mindig a két megfelelő ország közös határszakaszán keresztül kötünk össze, akkor könnyen elérhető, hogy ezek a vonalak ne keresztezzék egymást.

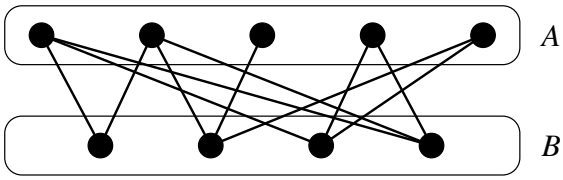
A négyszín-tétel pontosan tehát annyit állít, hogy ha G síkbarajzolható (és hurokélmentes) gráf, akkor $\chi(G) \leq 4$. A tételre több bizonyítás is született a 19. században, ám ezekről később kiderült, hogy hibásak. Ez is hozzájárult ahhoz, hogy a négyszín-tétel a gráfelmélet és a modern matematika egyik legközismertebb problémájává vált és a megoldására tett erőfeszítések a gráfelmélet jelentős fejlődését és több új ágának a kialakulását hozták.

6.1. Páros gráfok

Azok a G gráfok, amikre $\chi(G) = 1$, teljesen érdektelenek: ezek az üres (vagyis él nélküli) gráfok, hiszen egy él két végpontjára máris két különböző színt kell használnunk. Azok a gráfok viszont, amik két színnel színezhetők, már nagyon széles és alkalmazásokban gyakran felmerülő osztályát adják a gráfoknak: ezek a páros gráfok.

6.3. Definíció. *A G gráfot* páros gráfnak *nevezzük, ha a $V(G)$ csúcshalmaza felbontható az A és B diszjunkt halmazok egyesítésére úgy, hogy G minden éle egy A-beli csúcsot köt össze egy B-belivel. Ilyenkor a szokásos $G = (V,E)$ jelölés helyett a $G = (A,B;E)$ jelölést is használjuk.*

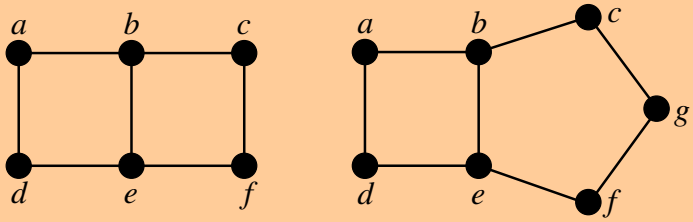
A fenti definíció csak átfogalmazása annak, hogy a G gráfra $\chi(G) \leq 2$ teljesül (vagyis hogy G két színnel színezhető): valóban, ha a $G = (A,B;E)$ páros gráfban az A-beli csúcsokat pirosra, a B-belieket kékre festjük, akkor helyes színezést kapunk. Egy páros gráf látható például a 6.2. ábrán.



6.2. ábra

Páros gráfok sokszor kerülnek elő gyakorlati alkalmazásokban is: olyankor, amikor két különböző típusú dolog közötti kapcsolatokat kell gráffal reprezentálni. Például: egy munkahelyen adottak dolgozók és elvégzendő munkák; ha A jelöli a dolgozók, B pedig a munkák halmazát és egy $a \in A$ dolgozót akkor kötünk össze egy $b \in B$ munkával, ha a el tudja végezni b-t, akkor ezzel nyilván egy $G = (A, B; E)$ páros gráfot adtunk meg.

6.4. Feladat. Döntsük el az alábbi gráfokról, hogy párosak-e.



Megoldás: A bal oldali gráf páros: ha az a, e és c csúcsokat tesszük az A pontosztályába, a maradék b, d és f csúcsokat pedig a B-be, akkor valóban teljesül, hogy a gráf minden éle A-beli csúcsot köt össze B-belivel. Ugyanezt úgy is megfogalmazhatjuk, hogy a gráfot megszíneztük két színnel: a, e és c piros, b, d és f pedig kék színt kapott.

A jobb oldali gráf viszont nem páros: ennek az oka a b, c, g, f és e csúcsok által meghatározott öt hosszú körben rejlik. Valóban: ha két színnel próbálnánk színezni a gráfot és b-t például pirosra festjük, akkor c-t kénytelenek volnánk kékre, g-t megint pirosra, f-et kékre – és így e-nek már egyik színt sem adhatjuk, mert piros és kék szomszédja is van. □

A fenti megoldás második felében látott jelenség általánosítható is: ha G páros gráf, akkor nem tartalmazhat páratlan hosszú kört. Az alábbi tétel azt mondja ki, hogy ennek a megfordítása is igaz.

6.5. Tétel. *A G gráf akkor és csak akkor páros gráf, ha nem tartalmaz páratlan hosszú kört.*

Bizonyítás: Már fentebb is említettük, hogy ha $G = (A, B; E)$ páros gráf, akkor nem tartalmazhat páratlan kört: valóban, ha minden él A és B között vezet, akkor például egy $a \in A$ csúcsból indulva páratlan sok lépés után (vagyis egy páratlan sok élű utat bejárva) nyilván B-beli csúcsnál tartunk, vagyis nem érhetünk vissza a-ba.

Annak megmutatásához, hogy ha G nem tartalmaz páratlan kört, akkor páros gráf, a 2. fejezetben látott BFS algoritmust hívjuk segítségül (és az ott bevezetett jelöléseket használjuk). Tegyük fel először, hogy G összefüggő és futtassuk le a BFS eljárást egy tetszőleges s csúcsából indítva. Alkossák most az A pontosztályt G-nek azok a v csúcsai, amikre táv(v) (vagyis v-nek az s-től való távolsága) páros, a B-be pedig azok a csúcsok kerüljenek, amikre táv(v) páratlan. Megmutatjuk, hogy

a csúcshalmaznak ez a felbontása megfelel a 6.3. Definíciónak; más szóval, hogy G minden $e = \{u,v\}$ élére táv(u) és táv(v) paritása különböző.

Ehhez először figyeljük meg, hogy a BFS algoritmust tetszőleges (irányítatlan) gráfban lefuttatva a gráf minden $e = \{u,v\}$ élére $|\text{táv}(u) - \text{táv}(v)| \leq 1$ teljesül. Ennek megmutatásához feltehetjük, hogy az eljárás végrehajtása során u korábban volt aktív csúcs v-nél (mert u és v szerepe szimmetrikus). Az u csúcs aktívvá válásakor az eljárás az $\{u,v\}$ élt is vizsgálja. Ha ebben a pillanatban táv$(v) = \infty$ akkor az algoritmus a táv$(v) \leftarrow$ táv$(u)+1$ értékadást hajtja végre, így $|\text{táv}(u) - \text{táv}(v)| = 1$. Ha viszont az $\{u,v\}$ él vizsgálatának a pillanatában táv$(v) \neq \infty$, akkor a $w =$ előző(v) csúcs az u-nál is korábban volt aktív csúcs. Ekkor egyrészt táv$(v) =$ táv$(w)+1$, másrészt táv$(w) \leq$ táv$(u) \leq$ táv(v) következik abból, hogy a w, u és v csúcsok ebben a sorrendben voltak aktívak (nem feltétlen közvetlenül egymás után). Mindezekből táv$(u) \leq$ táv$(v) =$ táv$(w)+1 \leq$ táv$(u)+1$ adódik, vagyis $|\text{táv}(u) - \text{táv}(v)| \leq 1$ most is igaz.

Válasszunk most egy tetszőleges $e = \{u,v\}$ élt; az előző bekezdésben írtak szerint tehát táv(u) és táv(v) különbsége 0 vagy 1. Ha ez a különbség 1, akkor készen vagyunk, táv(u) és táv(v) paritása eltér. Azt kell tehát kizárnunk, hogy táv$(u) =$ táv(v) teljesüljön. Tegyük fel ezért, hogy táv$(u) =$ táv(v) mégis igaz és induljunk el a BFS-fában u-ból és v-ből is visszafelé, s irányába (vagyis tekintsük az $u_0 = u$, $u_1 =$ előző(u_0), $u_2 =$ előző(u_1), … és a $v_0 = v$, $v_1 =$ előző(v_0), $v_2 =$ előző(v_1), … csúcsokon áthaladó utakat). Ez a két út egy ponton találkozni fog, hiszen legkésőbb táv$(u) =$ táv(v) lépés után mindkét csúcsból s-be érkezünk vissza – de lehet, hogy már ennél kevesebb lépés után is közös csúcshoz jutunk. (Ahhoz hasonlítható ez, mint amikor két, egy generációba tartozó rokon a családfájukon a legközelebbi közös ősüket keresi.) Legyen a két út első találkozási pontja a w csúcsban. Ekkor a w-ből az u-ba és a v-be vezető út a BFS-fában azonos, mondjuk k hosszúságú. (Látható, hogy $k =$ táv$(u) -$ táv(w), de ennek a bizonyításban nincs jelentősége.) Ekkor ezt a két utat egyesítve és még az $e = \{u,v\}$ éllel kiegészítve egy $2k+1$ hosszú, vagyis páratlan kört kapnánk, ami ellentmondás.

Ezzel tehát beláttuk, hogy ha G összefüggő és nem tartalmaz páratlan kört, akkor páros gráf. Ebből viszont nyilván következik ugyanez abban az esetben is, ha G nem összefüggő: mivel a fentiek szerint G minden K_i komponense páros gráf, vagyis $V(K_i)$ felbontható az A_i és B_i csúcshalmazokra a 6.3. Definíciónak megfelelően, ezért az $A = \cup_i A_i$ és $B = \cup_i B_i$ felbontás bizonyítja, hogy G páros gráf. □

A fenti bizonyításból az a fontos tény is kiderült, hogy egy adott G gráfról nagyon hatékonyan, a BFS algoritmussal eldönthető, hogy páros-e: ha a BFS eljárás a futása során talál olyan $e = \{u,v\}$ élt, amire táv$(u) =$ táv(v), akkor a G-ben van páratlan kör és így nem páros; ha viszont nem talál ilyen élt, akkor a táv(v) paritásán alapuló felbontás igazolja, hogy G páros.

A 6.5. Tételnek egyszerű következménye, hogy a fák is páros gráfok: valóban, mivel ezek körmentesek, ezért páratlan hosszú kört sem tartalmaznak.

6.2. A mohó színezés

A fentiek szerint tehát hatékony, polinomiális algoritmussal eldönthető egy G gráfról, hogy rá $\chi(G) \leq 2$ teljesül-e: ez akkor igaz, ha G páros gráf. Ehhez képest meglepő, hogy annak az eldöntése, hogy $\chi(G) \leq 3$ teljesül-e egy G gráfra (vagyis hogy G kiszínezhető-e három színnel) már sokkal nehezebb probléma: nem ismert rá polinomiális algoritmus; sőt, jó okunk van azt hinni (bár ez nem bizonyított tény), hogy ilyen nem is létezik. (Erről az informatikus képzés későbbi tárgyaiban lesz még szó.) Ebből persze az is következik, hogy $\chi(G)$ kiszámítása egy tetszőleges G gráf esetén szintén algoritmikusan nehéz probléma.

Leírunk viszont egy olyan algoritmust, ami minden G gráfot nagyon hatékonyan megszínez valahány színnel – de a felhasznált színek száma sajnos sokkal több is lehet $\chi(G)$-nél. Az eljárást *mohó színezésnek* nevezzük, mert a 3. fejezetben látott Kruskal-algoritmushoz hasonlóan mindig az adott pillanatban legjobbnak tűnő döntést hozza és ezen később sem változtat. Az algoritmus működése nagyon egyszerű: valamilyen sorrendben végighalad a csúcsokon és mindegyiket kiszínezi az első olyan színre, amilyen színű szomszédja még nincs; ha pedig ilyen színt már nem talál az eddig használt színek között, akkor bővíti a palettát egy új színnel.

Az eljárás leírásában a színeket azonosítani fogjuk a pozitív egészekkel: piros, kék, zöld, stb. helyett 1., 2., 3., stb. színekről fogunk beszélni. A v csúcs színét $c(v)$, az eddig használt legnagyobb sorszámú szín számát pedig C jelöli majd.

MOHÓ SZÍNEZÉS

Bemenet: Egy $G = (V,E)$ gráf és a csúcsainak egy $v_1, v_2, \ldots, v_n$ sorrendje.

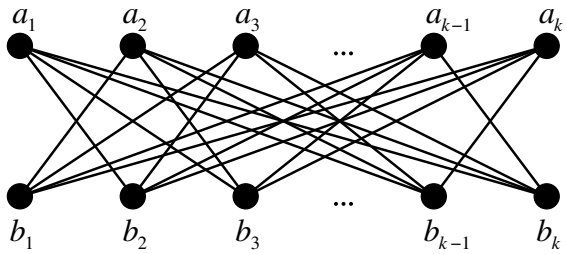
1 $C \leftarrow 1$; $c(v_1) \leftarrow 1$
2 **ciklus:** i **fut** 2-**től** n-**ig**
3 Legyen $t \in \{1,2,\ldots,C,C+1\}$ a legkisebb olyan szín, amire v_i-nek nincs t színű szomszédja.
4 $c(v_i) \leftarrow t$
5 **ha** $t = C+1$, **akkor**: $C \leftarrow C+1$
6 **ciklus vége**

Az eljárás 3. sorában $t = C+1$ csak akkor lehetséges, ha v_i-nek már minden eddig használt szín esetén (vagyis az $1,2,\ldots,C$ mindegyikére) van ilyen színre festett szomszédja. Ebben az esetben v_i elsőként kapja a $(C+1)$-es színt és az 5. sorban C értéke eggyel növekszik.

A 3. fejezetben, Kruskal algoritmusa kapcsán már említettük, hogy a mohó eljárások ugyan nagyon hatékonyak, de általában nem adnak optimális eredményt. A mohó színezés esetében is ez a helyzet: ez sokkal több színt is használhat annál, mint amennyi feltétlen szükséges volna (vagyis $\chi(G)$-nél). Az alábbi állításban példát mutatunk arra, hogy ez a különbség akár tetszőlegesen nagy is lehet.

6.6. Állítás. *Minden $k \geq 1$ egész esetén létezik olyan ($2k$ csúcsú) G gráf és a G csúcsainak egy olyan sorrendje, hogy $\chi(G) = 2$, de a mohó színezést a G-re a csúcsoknak ebben a sorrendjében futtatva az eljárás k színt használ.*

Bizonyítás: Készítsük el azt a páros gráfot, aminek a két pontosztálya $A = \{a_1, a_2, \ldots, a_k\}$ és $B = \{b_1, b_2, \ldots, b_k\}$. Minden $1 \leq i \leq k$ esetén az a_i-t kössük össze a b_i-n kívül minden más $b_j \in B$ ($j \neq i$) csúccsal, de b_i-vel ne; lásd a 6.3. ábrát.



6.3. ábra

Hajtsuk most végre a mohó színezést a csúcsoknak az $a_1, b_1, a_2, b_2, \ldots, a_k, b_k$ sorrendjében. Ekkor a_1 és b_1 az 1. színt kapják (mert nem szomszédosak). Mivel a_2 és b_2 már szomszédos b_1-gyel, illetve a_1-gyel, de egymással nem, ezért a 2. színt kapják. Hasonlóan: minden $1 \leq i \leq k$ esetén a_i-nek és b_i-nek az eljárás az i. színt adja, mert mindketten szomszédosak 1., 2., …, $(i-1)$. színt kapot csúccsal (de egymással nem). Így a mohó színezés végül valóban k színnel színezi ki a gráfot.

Emellett $\chi(G) = 2$ valóban igaz: mivel G páros gráf, ezért ha A csúcsait pirosra, B csúcsait kékre festjük, helyes színezést kapunk. □

Érdemes megjegyezni, hogy az, hogy a mohó algoritmus k színnel színezte a fenti gráfot, nagyon erősen múlt a csúcsok sorrendjén: ha például az $a_1, \ldots, a_k, b_1, \ldots, b_k$ sorrendre hajtottuk volna végre, akkor csak két színt használt volna, ami már optimális. Általában sem nehéz belátni, hogy tetszőleges G gráf esetén létezik a csúcsoknak olyan sorrendje, amit használva a mohó színezés optimális, vagyis $\chi(G)$ színnel színez. A problémát az jelenti, hogy egy ilyen sorrend megtalálása általában nagyon nehéz feladat – illetve pontosan ugyanolyan nehéz, mint a gráf egy optimális színezését meghatározni. Később látni fogjuk, hogy egy fontos speciális esetben ennél lényegesen jobb a helyzet.

Bár a mohó színezés a fentiek szerint általában nagyon gyenge teljesítményt nyújt, azért valamilyen felső becslést mégis lehet adni a felhasznált színek számára.

6.7. Állítás. *Legyen G (hurokélmentes) gráf és jelölje $\Delta(G)$ a G-beli maximális fokszámot (vagyis a G-beli csúcsok fokszámai közül a legnagyobbat). Ekkor a mohó színezést a csúcsok tetszőleges sorrendjében végrehajtva az legföljebb $\Delta(G)+1$ színt használ.*

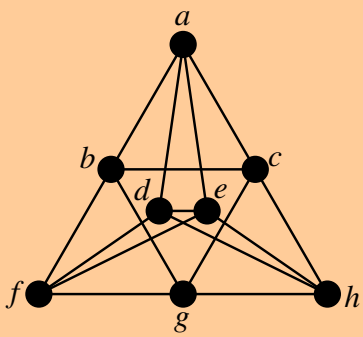
Bizonyítás: Azt kell tehát megmutatni, hogy a mohó színezést a fenti pszeudokód szerint lefuttatva C értéke nem növekszik $\Delta(G)+1$ fölé. Tegyük fel ezért, hogy C már elérte a $C=\Delta(G)+1$ értéket és a soron következő csúcs a v_i. Mivel $d(v_i)\leq\Delta(G)$, ezért legföljebb $\Delta(G)$ olyan szín létezhet, amit a v_i csúcs nem kaphat meg. (Ez is csak abban az esetben lehetséges, ha v_i minden szomszédja kapott már színt és mindegyikük csupa különbözőt.) Mivel tehát $C>\Delta(G)$, ezért kell létezzen olyan $t\in\{1,\ldots,C\}$ szín, amit a mohó eljárás (a paletta bővítése nélkül) v_i-nek adhat. □

6.8. Következmény. *Minden (hurokélmentes) G gráfra $\chi(G)\leq\Delta(G)+1$.*

Bizonyítás: Mivel a 6.7. Állítás szerint a mohó színezés $\Delta(G)+1$ színnel kiszínezi G-t, $\chi(G)$ értéke ennél több nem lehet. □

Természetesen a $\chi(G)\leq\Delta(G)+1$ becslés is nagyon „gyenge”, vagyis tetszőlegesen nagy különbség előfordulhat az egyenlőtlenség két oldala között. Erre nagyon egyszerű példát mutatni: ha G egy n csúcsú „csillag” (vagyis egy olyan fa, aminek az egyik csúcsa $n-1$ fokú és az összes többi 1 fokú), akkor $\chi(G)=2$ (hiszen az 1 fokúakat színezhetjük azonos színűre), de $\Delta(G)+1=n$.

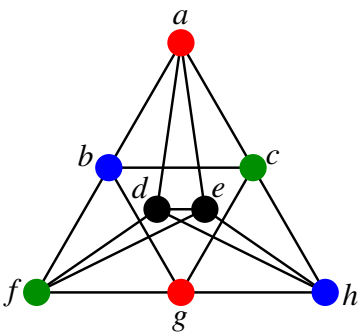
6.9. Feladat. Határozzuk meg az alábbi G gráf kromatikus számát, $\chi(G)$-t.



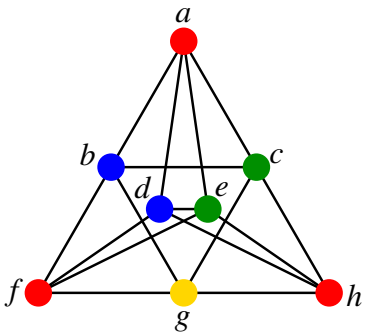
Megoldás: Válasszunk ki G-ben egy tetszőleges háromszöget: például az a, b és c csúcsokat. Ezeket muszáj három különböző színnel megfesteni: legyen a piros, b kék és c zöld. Ekkor (mivel „spórolunk a színekkel”) g lehet megint piros, h kék és f zöld, ahogyan azt a 6.4a ábra mutatja. Ezt a színezést folytatva d-nek és e-nek két további új színt kell adnunk, hiszen mindkettő szomszédos piros, kék és zöld csúccsal is, valamint egymással is. Mindebből azt hihetnénk, hogy $\chi(G)=5$; azonban fontos hangsúlyozni, hogy ez a gondolatmenet és az eredménye is teljesen rossz!

Valóban, a 6.4b ábra G-nek egy helyes színezését mutatja 4 színnel. Mi volt tehát a hiba a fenti gondolatmenetben? Az, hogy észrevétlen is a mohó színezést alkalmaztuk: csak akkor vezettünk be új színt, amikor az feltétlen szükségessé vált. Láttuk azonban, hogy a mohó színezés messze nem optimális, a szükségesnél sokkal

több színt is használhat. És valóban: ebben az esetben például jobban járunk, ha a fenti színezés sorrendjét követve g-nél nem „spórolunk”, hanem annak ellenére is bevezetünk egy új, negyedik színt, hogy „mohón” haladva arra még nem feltétlen volna szükség.



6.4a ábra



6.4b ábra

De akkor végül mennyi $\chi(G)$ értéke? Egy négy színnel való színezést már láttunk és most megmutatjuk, hogy G nem színezhető ki három színnel. A valóság az, hogy miközben a megoldás elején leírt gondolatmenet arra valóban alkalmatlan, hogy $\chi(G)$-t meghatározza, arra szerencsére tökéletesen alkalmas, hogy a három színnel való színezés lehetetlenségét kimutassa. Tegyük fel ugyanis indirekt, hogy G kiszínezhető három színnel. Ekkor ezt a három színt fel kell használnunk az a, b és c csúcsok alkotta háromszögön, így feltételezhetjük, hogy a piros, b kék és c zöld. Mivel egy három színnel való színezést feltételeztünk, ezért nincs választásunk: g-nek pirosnak kell lennie (hiszen kék és zöld szomszédja már van) és hasonlóan f zöld, h pedig kék kell legyen. Így épp ahhoz a szituációhoz érkezünk, amit a 6.4a ábra mutat. Most azonban azt látjuk, hogy a d csúcsnak már van piros, kék és zöld szomszédja is, így egyik színt sem kaphatja ebből a háromból. Ez az ellentmondás mutatja, hogy mégsem létezhet három színnel való színezés.

Megmutattuk tehát, hogy G megszínezhető négy színnel, de hárommal nem, így $\chi(G) = 4$. □

6.3. Klikkszám és kromatikus szám

A 6.2. feladat megoldásában úgy érveltünk amellett, hogy háromnál kevesebb szín nem lehet elég a színezéshez, hogy mutattunk három csúcsot, amelyek közül bármely kettő szomszédos volt; ebből azonnal következett, hogy már ehhez a három csúcshoz is szükség volt három színre. Ugyanez a gondolat három helyett tetszőleges, nagyobb számokra is működhet: ha G-ben találunk k csúcsot úgy, hogy ezek közül bármely kettő szomszédos, akkor biztosak lehetünk benne, hogy legalább k színre szükség van G színezéséhez.

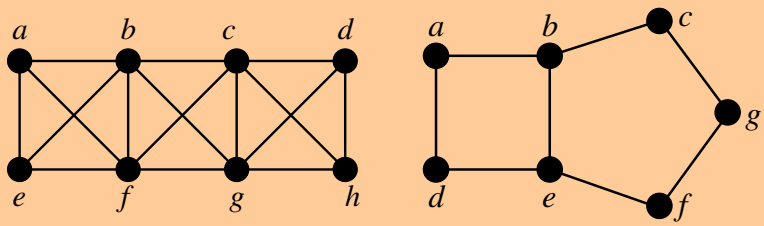
6.10. Definíció. *A G gráf* klikkszáma *k, ha G-ben található k darab csúcs úgy, hogy ezek közül bármely kettő szomszédos, de $k+1$ ilyen csúcs már nem található. G klikkszámának a jele: $\omega(G)$.*

A fenti elnevezés onnan ered, hogy G-nek azokat a részgráfjait, amik teljes gráfot alkotnak (vagyis bármely két csúcsuk szomszédos), *klikknek* is szokták nevezni. $\omega(G)$ tehát a G klikkjei közül az egyik legnagyobbnak a csúcsszáma.

6.11. Állítás. *Minden (hurokélmentes) G gráfra $\omega(G) \leq \chi(G)$ teljesül.*

Bizonyítás: G-ben található $\omega(G)$ darab csúcs, amik közül bármely kettő szomszédos. Ezeknek a csúcsoknak tehát G tetszőleges színezésében csupa különböző színt kell kapnia. Így G minden színezése legalább $\omega(G)$ színt használ, amiből $\omega(G) \leq \chi(G)$ valóban következik. □

6.12. Feladat. Határozzuk meg az alábbi G gráfok $\omega(G)$ klikkszámát és $\chi(G)$ kromatikus számát.



Megoldás: A bal oldali gráfban könnyű találni négy csúcsot, amik közül bármely kettő szomszédos: például a, b, e és f. Ebből $\omega(G) \geq 4$ következik. Bár a gráf elegendően kicsi ahhoz, hogy kevés próbálgatással is beláthassuk, hogy öt méretű klikk már nincs benne, ezt rövidebben is meg lehet indokolni. Ugyanis a gráf kiszínezhető négy színnel: például ha a és c piros, b és d kék, e és g zöld, f és h pedig sárga színt kap. Ebből tehát $\chi(G) \leq 4$ következik. Összevetve ezeket a 6.11. Állítással: $4 \leq \omega(G) \leq \chi(G) \leq 4$, amiből tehát $\omega(G) = \chi(G) = 4$ következik.

A jobb oldali gráfról már a 6.4. Feladatban láttuk, hogy nem páros gráf (mert tartalmaz páratlan kört), így rá $\chi(G) > 2$. Három színnel viszont könnyű kiszínezni: például ha a, e és c piros, b, d és f kék, g pedig zöld színt kap. Így tehát $\chi(G) = 3$. Másrészt ránézésre is jól látszik, hogy ebben a gráfban nincs háromszög (vagyis három csúcsú klikk). Két csúcsú klikk viszont még nyilván van: bármelyik él két végpontja ilyet alkot. Ezért $\omega(G) = 2$. □

6.13. Feladat. A G gráf csúcsai legyenek az 1 és 1000 közé eső természetes számok. Két különböző csúcsot akkor kössünk össze, ha a különbségük legfeljebb 9. Mennyi G kromatikus száma?

Megoldás: Mivel minden 1 és 1000 közötti egész a nála legföljebb 9-cel kisebb és 9-cel nagyobb számokkal van összekötve, ezért könnyen megadható G egy helyes színezése tíz színnel: az 1-re végződő (vagyis 10-zel osztva 1 maradékot adó) számokat 1-es színűre, a 2-re végződőket 2-es színűre, stb., a 0-ra végződőket 10-es színűre színezzük. Mivel azonos számjegyre végződő számok különbsége mindig legalább 10, ezért valóban nem festettünk szomszédos csúcsokat azonos színűre.

Vegyük észre azt is, hogy G-ben például az $\{1,2,\ldots,10\}$ csúcshalmaz klikket alkot: valóban, közülük bármely kettőnek a különbsége legföljebb 9, ezért szomszédosak. A tíz csúcsú klikk létezése pedig bizonyítja, hogy G nem színezhető meg kilenc színnel (hiszen már erre a tíz csúcsra is tíz különböző színt kell használnunk).

Mivel tehát G megszínezhető tíz színnel, de kilenccel nem, ezért $\chi(G) = 10$.

Más megfogalmazásban: a tíz színnel való helyes színezéssel azt mutattuk meg, hogy $\chi(G) \leq 10$; a tíz csúcsú klikk létezéséből pedig az következik, hogy $\omega(G) \geq 10$. Ezekből és a a 6.11. Állításból $10 \leq \omega(G) \leq \chi(G) \leq 10$ következik, amiből tehát $\omega(G) = \chi(G) = 10$. □

A fenti feladatokból is látszik, hogy színezés szempontjából azok a G gráfok „szerencsések”, amikre $\chi(G) = \omega(G)$ teljesül: ha egy ilyen gráfban mutatunk egy k csúcsú klikket és megszínezzük a csúcsait k színnel, akkor $k \leq \omega(G) \leq \chi(G) \leq k$ miatt $\omega(G) = \chi(G) = k$ teljesül; vagyis egyszerre megtudjuk $\chi(G)$ és $\omega(G)$ értékét. Sajnos azonban vannak olyan gráfok is, amikre $\omega(G) < \chi(G)$; ilyen volt például a 6.12. Feladatban a jobb oldali gráf és a 6.9. Feladat gráfja is (bár az utóbbiban $\omega(G)$-t nem határoztuk meg); de minden páratlan hosszú körre is teljesül, hogy $\chi = 3$ és $\omega = 2$.

Ezekben a példákban tehát $\omega(G)$ kisebb volt $\chi(G)$-nél, de mindig csak eggyel. Felmerül a kérdés: vajon az $\omega(G) \leq \chi(G)$ becslés is ugyanolyan „rossz-e”, mint amilyen például a $\chi(G) \leq \Delta(G)+1$ volt; vagyis előfordulhat-e tetszőlegesen nagy különbség $\omega(G)$ és $\chi(G)$ között? A válasz itt is igenlő – sőt, még az $\omega(G) = 2$ feltétel mellett (vagyis a háromszöget nem tartalmazó gráfok körében) is elérhető tetszőlegesen nagy kromatikus szám. Ennek megmutatásához azonban már sokkal ravaszabb konstrukcióra van szükség.

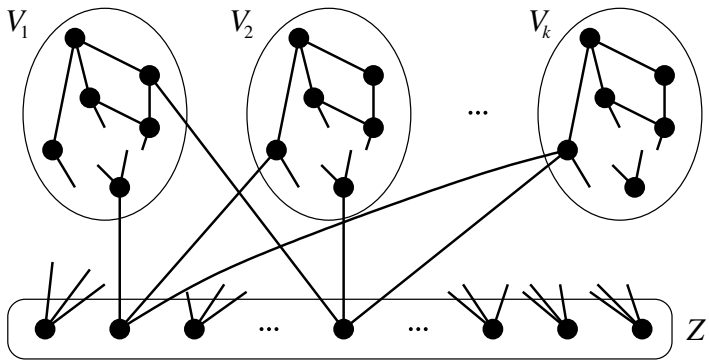
6.14. Tétel. *Minden $k \geq 2$ esetén létezik olyan G_k gráf, amire $\omega(G_k) = 2$ és $\chi(G_k) = k$.*

Bizonyítás: $k = 2$-re és $k = 3$-ra még könnyű ilyen gráfokat mutatni: G_2 lehet a K_2 teljes gráf (vagyis két csúcs és köztük egy él), G_3 pedig egy öt csúcsú (vagy bármilyen páratlan hosszú) kör.

A bizonyítás első fontos gondolata az lesz, hogy a G_k gráfokat nem közvetlenül, hanem rekurzióval adjuk meg: megadunk egy eljárást, ami egy, a feltételeknek már megfelelő G_k gráfból (amire tehát $\omega(G_k) = 2$ és $\chi(G_k) = k$ igaz) elkészíti a G_{k+1} gráfot (amire tehát $\omega(G_{k+1}) = 2$, de $\chi(G_{k+1}) = k+1$). Ennek az eljárásnak az ismételgetésével (például) G_3-ból (egy öt pontú körből) indulva tetszőleges k-ra megkaphatjuk G_k-t. (Úgy is fogalmazhatunk, hogy a tételt k-ra vonatkozó teljes indukcióval látjuk be: $k = 2$-re $k = 3$-ra az állítás igaz, az indukciós lépés bizonyítása

pedig abból fog állni, hogy ha G_k létezését már tudjuk valamilyen k-ra, akkor ebből megmutatjuk G_{k+1} létezését is.)

Tegyük fel tehát, hogy a $G_k = (V,E)$ gráf már adott valamilyen $k \geq 3$ esetén, amire tehát $\omega(G_k) = 2$ és $\chi(G_k) = k$. Vegyük most fel G_k-nak k darab diszjunkt példányát; vagyis készítsük el a V csúcshalmaz k darab diszjunkt „kópiáját", a $V_1, V_2, \ldots, V_k$ halmazokat és minden V_i-n belül ugyanazokat a csúcspárokat kössük össze éllel, mint amik V-ben szomszédosak voltak (lásd a 6.5. ábrát). Ezek után az összes lehetséges módon válasszunk ki a $V_1, V_2, \ldots, V_k$ halmazokból egy-egy csúcsot és a kijelölt csúcs-k-ast kössük össze egy további, újonnan felvett csúccsal – mégpedig minden csúcs-k-as esetén egy-egy másikkal. Az így kapott gráf pedig legyen G_{k+1}. (Ezek szerint tehát ha G_k-nak n csúcsa volt, akkor G_{k+1}-nek $k \cdot n + n^k$ csúcsa lesz: a V_i halmazoknak ugyanis együttesen $k \cdot n$ csúcsa van, továbbá n^k-féleképpen lehet kiválasztani ezek mindegyikéből egy-egy csúcsot és minden ilyen kiválasztásból egy-egy új csúcs születik G_{k+1}-ben.)



6.5. ábra

Megmutatjuk, hogy G_{k+1} megfelel a kívánt feltételeknek. Jelölje ehhez a V_i-kből kiválasztott csúcs-k-asokból létrejött G_{k+1}-beli csúcsok halmazát Z (aminek tehát n^k eleme van).

Kezdjük az $\omega(G_{k+1}) = 2$ állítás indoklásával. A V_i halmazokon belül nyilván nem találunk háromszöget (vagyis három csúcsú klikket), hiszen az ezek által feszített részgráfok G_k-val izomorfak, amiről tudjuk, hogy $\omega(G_k) = 2$. Másrészt olyan háromszög sem lehet G_{k+1}-ben, ami tartalmaz Z-beli csúcsot. Először is Z-n belül nem fut G_{k+1}-nek éle, ezért egy ilyen háromszög legföljebb egy Z-beli csúcsot tartalmazhatna. De mivel minden Z-beli csúcs összes szomszédja különböző V_i-kben található, így ezek között nem fut él. Vagyis olyan háromszög sincs G_{k+1}-ben, ami csak egy Z-beli csúcsot tartalmaz.

Most azt látjuk be, hogy G_{k+1} megszínezhető $k+1$ színnel. A V_i halmazok által feszített részgráfok izomorfak G_k-val, így ezek megszínezhetők k színnel (hiszen tudjuk, hogy $\chi(G_k) = k$). Ráadásul a V_i-k színezéséhez használhatjuk ugyanazt a k színt, hiszen a V_i-k között nem futnak élek. Végül bevezethetünk egy új, $(k+1)$-edik színt és Z minden csúcsát megszínezhetjük ezzel a színnel. Mivel Z-n belül nem

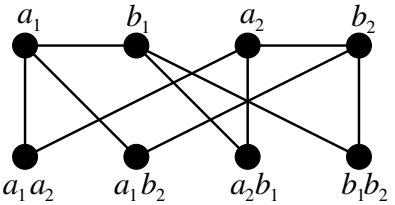
futnak élek, ezzel valóban helyes színezést kapunk.

Végül azt mutatjuk meg, hogy G_{k+1} nem színezhető meg k színnel. Tegyük fel indirekt, hogy mégis létezik G_{k+1}-nek egy helyes színezése k színnel és rögzítsünk le képzeletben egy ilyen színezést. Ekkor egy tetszőleges V_i-t kiválasztva annak a csúcsai között mind a k színnek elő kell fordulnia; valóban, ha nem így volna, akkor a V_i által feszített, G_k-val izomorf részgráf $k-1$ színnel volna színezve, ami ellentmond annak, hogy $\chi(G_k) = k$ (és így G_k nem színezhető meg k-nál kevesebb színnel). Válasszunk ki ezért V_1-ből egy 1. színű v_1 csúcsot; majd válasszunk V_2-ből egy 2. színű v_2 csúcsot, stb. Vagyis minden $1 \leq i \leq k$ esetén egy olyan $v_i \in V_i$ csúcsot választunk, ami az i-edik színt kapta a feltételezett színezésben. A $(v_1, v_2, \ldots, v_k)$ csúcs-k-ashoz a G_{k+1} konstrukciója szerint tartozik egy $z \in Z$ csúcs, ami épp ezzel a k csúccsal szomszédos. Ám a feltételezett színezésünk z-nek is adott színt; ha ez a szín a j-edik, akkor találtunk a gráfban két szomszédos és azonos színűre festett csúcsot: z szomszédos v_j-vel és mindketten j-edik színűek. Ez az ellentmondás mutatja, hogy G_{k+1}-nek valóban nem létezhet helyes színezése k színnel.

Beláttuk tehát, hogy G_{k+1} megszínezhető $k+1$ színnel, de k színnel már nem. Így $\chi(G_{k+1}) = k+1$ valóban igaz. □

A G_k gráfnak a fenti bizonyításban bemutatott konstrukciója Alexander Zykov (1922 – 2013) orosz matematikustól származik 1949-ből. Ha például el akarnánk vele készíteni egy olyan gráfot, amire $\omega = 2$ és $\chi = 4$, akkor alkalmazhatnánk az öt pontú körre, mint G_3-ra; az így kapott G_4 gráfnak tehát már $3 \cdot 5 + 5^3 = 140$ csúcsa volna. Ha pedig ebből a G_4-ből a G_5-öt akarnánk megkapni, annak már $4 \cdot 140 + 140^4$ csúcsa volna, ami 384 milliónál több. A G_k gráfok csúcsszáma tehát robbanásszerűen nő, az exponenciálisnál is gyorsabban. Ismertek olyan konstrukciók is a fenti tétel bizonyítására, amiknek a csúcsszáma „csak” exponenciális tempóban növekszik, de ezek bonyolultabbak a fentinél.

A fenti bizonyítás konstrukcióját a 6.6. ábrán illusztráljuk. Mivel már G_4 ábrázolására sincs esélyünk (hiszen 140 csúcsú), ezért azt mutatjuk be, hogy ha a konstrukciót az a és b csúcsokból és a köztük futó élből álló gráfra, mint G_2-re alkalmazzuk, akkor ebből milyen G_3 keletkezik. A $V(G_2)$ két kópiáját a $V_1 = \{a_1, b_1\}$ és $V_2 = \{a_2, b_2\}$ csúcshalmazok adják; minden V_1-ből és V_2-ből választott $u \in V_1$ és $v \in V_2$ csúcspár esetén uv jelöli a belőlük keletkezett új csúcsot, amit tehát u-val és v-vel kötünk össze. A kapott G_3 gráf valóban nem tartalmaz háromszöget, de mivel öt csúcsú kört többet is, ezért tényleg nem lehet két színnel megszínezni.



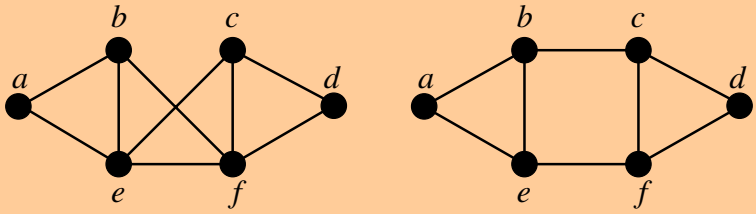
6.6. ábra

6.4. Intervallumgráfok kromatikus száma

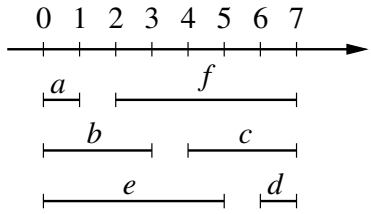
Idézzük fel a gráfok színezésének gyakorlati alkalmazására a 6. fejezet elején említett példát: előre adott időintervallumokban végrehajtandó számítási feladatokat kell processzorokhoz rendelnünk úgy, hogy a közös processzorra kerülő feladatok időintervallumai között nem lehet átfedés. Ekkor az ehhez szükséges processzorok számának minimalizálása egy gráfszínezési feladatra vezetett: G csúcsai a feladatok és két feladat akkor szomszédos G-ben, ha az intervallumaik metszők. Az ebben a problémában is felmerülő gráfokat vezeti be az alábbi definíció.

6.15. Definíció. *A G egyszerű gráfot akkor nevezzük* intervallumgráfnak*, ha léteznek a számegyenesen olyan $I_1, I_2, \ldots, I_n \subseteq \mathbb{R}$ (korlátos és zárt) intervallumok, hogy G ezekből megkapható a következő módon: G csúcsai megfelelnek az intervallumoknak és két különböző csúcs pontosan akkor szomszédos G-ben, ha a két megfelelő intervallumnak van közös pontja. Ilyenkor azt mondjuk, hogy az $\{I_1, \ldots, I_n\}$ intervallumrendszer* reprezentálja *a G intervallumgráfot.*

6.16. Feladat. Döntsük el az alábbi gráfokról, hogy intervallumgráfok-e.



Megoldás: A bal oldali gráf intervallumgráf, mert reprezentálja (például) a következő intervallumrendszer: $a = [0,1]$, $b = [0,3]$, $c = [4,7]$, $d = [6,7]$, $e = [0,5]$ és $f = [2,7]$. Valóban, a 6.7. ábrát a gráf fenti rajzával összevetve látszik, hogy bármely két csúcs pontosan akkor szomszédos, ha a megfelelő intervallumok metszők.



6.7. ábra

A jobb oldali ábra gráfja viszont nem intervallumgráf; ennek az oka pedig a b, e, c és f csúcsok alkotta négy pontú körben rejlik. Tegyük fel ugyanis, hogy ez a gráf

mégis intervallumgráf és egy ezt reprezentáló intervallumrendszerben ennek a négy csúcsnak sorra az $I_b = [b_1, b_2]$, $I_e = [e_1, e_2]$, $I_c = [c_1, c_2]$ és $I_f = [f_1, f_2]$ intervallumok felelnek meg. Mivel b és f nem szomszédosak a gráfban, ezért I_b és I_f diszjunktak kell legyenek; mivel b és f szerepe teljesen szimmetrikus, feltehetjük például, hogy I_b balra van I_f-től, vagyis $b_2 < f_1$. Mivel c szomszédos b-vel és f-fel is, ezért I_c-nek metszenie kell I_b-t és I_f-et is; ebből $c_1 \leq b_2$ és $c_2 \geq f_1$ következik. Ugyanez elmondható e-ről is, amiből $e_1 \leq b_2$ és $e_2 \geq f_1$ adódik. Ebből viszont az következik, hogy I_c és I_e metszők, hiszen a $[b_2, f_1]$ intervallumot mindkettő tartalmazza. Ez viszont ellentmond annak, hogy c és e nem szomszédos a gráfban. □

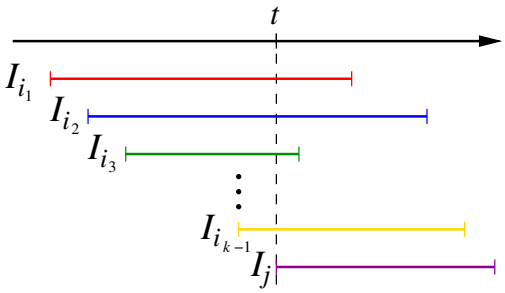
Fentebb már kiderült a számítási feladatok ütemezésének problémája kapcsán, hogy az intervallumgráfok színezése gyakorlati alkalmazásokban is felmerülő, fontos feladat. Korábban arról is szóltunk már, hogy $\chi(G)$ meghatározása egy tetszőleges G gráfra algoritmikusan nagyon nehéz probléma. Intervallumgráfokra viszont szerencsére hatékony algoritmussal megoldható; ezt mondja ki az alábbi tétel.

6.17. Tétel. *Legyen G intervallumgráf és tegyük fel, hogy G-t az $\{I_1, I_2, \ldots, I_n\}$ intervallumrendszer reprezentálja. Ekkor ha az $\{I_1, \ldots, I_n\}$ intervallumokat a baloldali végpontjuk szerinti növekvő sorrendbe rendezzük és G csúcsainak erre a sorrendjére hajtjuk végre a mohó színezést, akkor az eljárás G-t optimális számú, $\chi(G)$ színnel színezi meg.*

Bizonyítás: Legyen $I_i = [a_i, b_i]$ minden $i = 1, \ldots, n$ esetén. Mivel az intervallumok indexelésének nincs jelentősége, ezért az egyszerűség kedvéért feltehetjük, hogy az $I_1, I_2, \ldots, I_n$ intervallumok ebben a sorrendben eleve a baloldali végpontjuk szerinti növekvő sorrendben állnak, vagyis $a_1 \leq a_2 \leq \ldots \leq a_n$ teljesül. Jelölje a mohó színezés által használt színek számát k; vagyis a mohó eljárást a 6.2. szakaszban írt pszeudokód szerint futtatva annak a leállásakor $C = k$. Meg fogjuk mutatni, hogy G-ben létezik k csúcsú klikk; ebből a 6.3. szakaszban írtak szerint valóban következni fog, hogy G-t nem lehet k-nál kevesebb színnel megszínezni (mert már a klikk csúcsaira is k különböző színt kell használni).

Legyen I_j az az intervallum, amit a mohó eljárás elsőnek színezett a k-adik színűre (vagyis I_j-nél érte el a C értéke a k-t) és legyen $t = a_j$ ennek a bal oldali végpontja (lásd a 6.8. ábrát). Mivel a mohó színezés I_j-t nem színezte 1-es színűre, ezért I_j-nek kell létezzen G-ben egy olyan szomszédja, amit az eljárás már korábban 1-es színre színezett; jelölje ezt az intervallumot I_{i_1}. Mivel I_{i_1}-et az eljárás korábban vizsgálta, ezért annak a bal oldali végpontjára $a_{i_1} \leq t = a_j$. Másrészt viszont I_{i_1} metszi I_j-t (hiszen a gráfban szomszédosak), így ebből az I_{i_1} jobb oldali végpontjára $b_{i_1} \geq t = a_j$ adódik. Mindebből tehát $t \in I_{i_1}$ következik. Hasonlóan folytatva: mivel a mohó eljárás I_j-t 2-es színűre sem színezte, ezért kell létezzen egy olyan I_{i_2} intervallum, ami már I_j vizsgálata előtt a 2-es színt kapta és amelyre $t \in I_{i_2}$. Hasonlóan kapjuk az $I_{i_3}, \ldots, I_{i_{k-1}}$ intervallumokat is.

Összefoglalva a fentieket: az $I_{i_1}, I_{i_2}, \ldots, I_{i_{k-1}}$ intervallumok mindegyike tartalmazza t-t és ugyanez még I_j-ről is elmondható. Az ennek a k darab intervallumnak



6.8. ábra

megfelelő csúcsok tehát klikket alkotnak G-ben, hiszen az intervallumok közül bármelyik kettő metsző (mert tartalmazzák t-t). A k csúcsú klikk (és a k színnel való színezés) létezéséből pedig a fentiek szerint valóban következik, hogy $\chi(G) = k$. □

Fontos hangsúlyozni, hogy a fenti tételben szereplő algoritmus futásához nem elegendő a megszínezendő intervallumgráf ismerete, hanem egy azt reprezentáló intervallumrendszer is szükséges. Ha például az algoritmust a 6.16. Feladat bal oldali gráfjára, illetve az azt reprezentáló, a 6.7. ábrán látható intervallumrendszerre futtatjuk, akkor az eljárás az intervallumokat $a = [0,1]$, $b = [0,3]$, $e = [0,5]$, $f = [2,7]$, $c = [4,7]$, $d = [6,7]$ sorrendben vizsgálhatja (itt az első három intervallum sorrendje közömbös). Az algoritmus először a-t 1-es, majd b-t 2-es és e-t 3-as színűre festi, hiszen ezek közül bármely kettő metsző. Most a soron következő f ismét az 1-es színt kapja, mert nem szomszédos a-val. Ezután c a 2-es színt kapja, mert 1-es színű szomszédja már van (mégpedig f), de 2-es nincs. Végül d a 3-as színt kapja, mert az 1-es, illetve 2-es színt kapott f-fel, illetve c-vel szomszédos, de 3-as színű szomszédja nincs. Így az algoritmus ezt az intervallumgráfot három színnel színezi ki (és a színosztályok, vagyis az azonos színnel színezett csúcshalmazok a 6.7. ábra sorainak felelnek meg). A legnagyobb sorszámú, 3-as színt az eljárás futása során elsőként az e kapta; ennek a bal oldali végpontja a $t = 0$ és a fenti bizonyításnak megfelelően a 0-t tartalmazó intervallumok (vagyis a, b és e) valóban három csúcsú klikket alkotnak a gráfban – ezzel igazolva, hogy a kapott színezés optimális számú színt használ.

6.18. Következmény. *Ha G intervallumgráf, akkor rá $\omega(G) = \chi(G)$ teljesül.*

Bizonyítás: A 6.17. Tétel bizonyításban megmutattuk, hogy minden intervallumgráf esetén létezik olyan k szám, amire G megszínezhető k színnel és G tartalmaz k csúcsú klikket. Az előbbiből $\chi(G) \leq k$, az utóbbiból $\omega(G) \geq k$ következik. Ezekből és a 6.11. Állításból $k \leq \omega(G) \leq \chi(G) \leq k$ adódik, amiből tehát $\omega(G) = \chi(G)$ valóban következik. □

7. fejezet

Párosítások

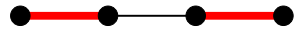
Egy munkahelyeken a dolgozókat két fős csapatokba kell beosztani, mert a munka természetéből fakadóan csak párosával tudják ellátni a feladataikat. Ez a beosztás azonban nem tetszőleges, bizonyos párok nem volnának alkalmasak közös munkavégzésre. A helyzetet egy G egyszerű gráffal ábrázolhatjuk: a csúcsok a dolgozók és két csúcs akkor szomszédos, ha a megfelelő két dolgozó kerülhet egy csapatba. Ekkor a munkahely vezetősége számára természetesen adódó feladat, hogy a lehető legtöbb megengedett párt hozzák létre a dolgozóik közül. A következő definíció az ennek a feladatnak megfelelő alapfogalmakat vezeti be.

7.1. Definíció.
- *A G gráfban az $M \subseteq E(G)$ élhalmazt* párosításnak *vagy* független élhalmaznak *nevezzük, ha (M nem tartalmaz hurokélt és) M semelyik két élének nincs közös végpontja.*
- *M* maximális párosítás*, ha G-nek nincs $|M|$-nél nagyobb méretű párosítása.*
- *A G-beli maximális párosítások méretét $\nu(G)$ jelöli.*
- *Az M független élhalmaz* teljes párosítás*, ha G minden csúcsára illeszkedik M-beli él.*

A fenti definíció szerint tehát a „párosítás” és a „független élhalmaz” kifejezések szinonimák, a jelentésük azonos. A definíciót fogalmazhatjuk úgy is, hogy M akkor párosítás, ha (nincs benne hurokél és) G minden csúcsára legfeljebb egy M-beli él illeszkedik; ha pedig ebben a mondatban a *legfeljebb* egyet *pontosan* egyre cseréljük, akkor a teljes párosítás definícióját kapjuk. Fontos kiemelni, hogy a maximális párosítás fogalma *nem azt jelenti*, hogy M-hez nem lehet egy további élt hozzávenni úgy, hogy az párosítás maradjon. Valóban, például a 7.1a ábra G gráfjában (ami egy három hosszú út) pirossal jelölt, egy élű M párosításhoz nem lehet hozzávenni a maradék két él közül egyet sem, de M mégsem maximális, mert a 7.1b ábrán egy ennél nagyobb párosítás látható ugyanebben a gráfban. Ha tehát egy G gráfra azt állítjuk, hogy $\nu(G) = k$, az azt jelenti, hogy G-ben van k élű független élhalmaz, de

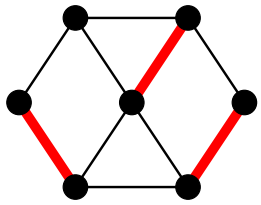
$k+1$ élű már nincs. A ν görög betű, a kiejtése „nű”; $\nu(G)$-t a G-beli „független élek maximális számának” is szokták hívni.

7.1a ábra

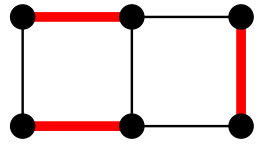


7.1b ábra

Párosítást alkotnak például a 7.2 ábrán pirossal jelölt élhalmazok. A 7.2b ábrán a párosítás teljes, hiszen minden csúcsra illeszkedik párosításbeli él. A 7.2a ábrán a párosítás nem teljes, de maximális, hiszen hét csúcsból nyilván nem lehet háromnál több párt létrehozni. Ezért mindkét gráfról elmondható, hogy rájuk $\nu(G) = 3$.

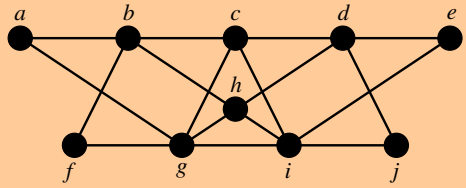


7.2a ábra

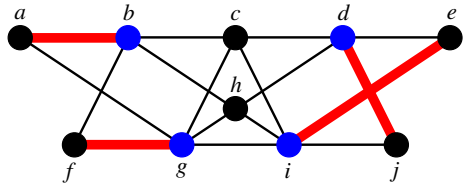


7.2b ábra

7.2. Feladat. Határozzuk meg az alábbi G gráfban a független élek maximális számát, $\nu(G)$-t.



Megoldás: G-ben nagyon könnyű négy élű párosítást találni, a számos lehetőség közül egy például a 7.3. ábrán látható $M = \{\{a,b\},\{f,g\},\{d,j\},\{i,e\}\}$ élhalmaz.



7.3. ábra

Megmutatjuk, hogy G-ben nincs öt élű párosítás – de ehhez már egy ravasz ötletet fogunk bevetni. Tekintsük ugyanis a 7.3. ábrán kékkel jelölt $X = \{b,d,g,i\}$

csúcshalmazt. Ez rendelkezik azzal a speciális tulajdonsággal, hogy G *minden* élének (tehát nem csak az M-be tartozó, pirossal kiemelteknek) legalább az egyik végpontját tartalmazza. Valóban: X a $\{g,i\}$ él mindkét végpontját tartalmazza és G összes többi (szám szerint 16) élének pontosan az egyik végpontját.

Miért hasznos X-nek ez a tulajdonsága? Tegyük fel, hogy G-ben van egy öt élű M' párosítás. Ekkor M' éleire is teljesül, hogy legalább az egyik végpontjuk X-beli. Mivel M' öt elemű és X csak négy, ezért X-nek kell legyen olyan eleme, amire két M'-beli él is illeszkedik. Ez ellentmond annak, hogy M' párosítás (hiszen két M'-beli él közös csúcsra illeszkedne) és így bizonyítja, hogy nem létezhet G-ben öt élű párosítás.

Megmutattuk tehát, hogy G-ben van négy élű párosítás, de öt élű nincs, így $\nu(G) = 4$. □

A fenti megoldásban kulcsszerepet játszó X halmaznak megfelelő fogalmat vezeti be a következő definíció.

7.3. Definíció.
- *A G gráf csúcsainak egy $X \subseteq V(G)$ halmazát* lefogó ponthalmaznak *nevezzük, ha G minden élének legalább az egyik végpontja X-beli.*
- *X* minimális lefogó ponthalmaz*, ha G-ben nincs $|X|$-nél kisebb lefogó ponthalmaz.*
- *A G-beli minimális lefogó ponthalmazok méretét $\tau(G)$ jelöli.*

A 7.1. Definíció után írtakhoz hasonlókat mondhatunk most is: egy lefogó ponthalmaz minimalitása *nem azt jelenti*, hogy egy csúcsát sem lehet elhagyni anélkül, hogy a lefogó tulajdonság sérülne; hanem azt, hogy nincs ennél kisebb méretű lefogó ponthalmaz. Erre példa a 7.4 ábrán látható kettő hosszú út: ha X az út két végpontjából áll, akkor ez egy olyan lefogó ponthalmaz, aminek egyik tagját sem lehetne elhagyni; de X mégsem minimális, mert az út középső pontja önmagában is lefogó. Így ha egy G gráfról azt állítjuk, hogy $\tau(G) = k$, az azt jelenti, hogy G-ben van k csúcsú lefogó ponthalmaz, de $k-1$ csúcsú nincs. A τ is görög betű, a kiejtése: „tau”; $\tau(G)$-t pedig a G-beli „lefogó pontok minimális számának” is szokták nevezni.

7.4a ábra

7.4b ábra

A 7.2. Feladat megoldásában láttuk, hogy a lefogó ponthalmazok és a párosítások mérete között fontos kapcsolat van; ezt általánosítja az alábbi állítás.

7.4. Állítás. *Minden G gráfra $\nu(G) \leq \tau(G)$ teljesül.*

Bizonyítás: Legyen G-ben M egy maximális párosítás és X egy minimális lefogó ponthalmaz; ekkor tehát $|M| = \nu(G)$ és $|X| = \tau(G)$. X minden élnek legalább az

egyik végpontját tartalmazza, így ez nyilván M éleire is teljesül. Azonban ugyanaz az X-beli csúcs nem illeszkedhet két M-beli élre is, mert M párosítás. Más szóval: minden M-beli élre másik X-beli csúcsnak kell illeszkednie. Ebből tehát $|M| \leq |X|$ és így $\nu(G) \leq \tau(G)$ valóban következik. □

7.5. Feladat. Határozzuk meg a 7.2. Feladat gráfjában a lefogó pontok minimális számát, $\tau(G)$-t.

Megoldás: A 7.3. ábrán mutattunk G-ben egy négy élű párosítást és egy négy pontú lefogó ponthalmazt. Az előbbiből $\nu(G) \geq 4$, az utóbbiból $\tau(G) \leq 4$ következik. Ezeket összevetve a 7.4. Állítással kapjuk, hogy $4 \leq \nu(G) \leq \tau(G) \leq 4$, amiből pedig $\nu(G) = \tau(G) = 4$ adódik. □

A 7.2. Feladat gráfjára tehát $\nu(G) = \tau(G)$ teljesül, de ugyanez sajnos már nem minden gráfra igaz. A legegyszerűbb példa erre, ha G egy három pontú kör: ekkor $\nu(G) = 1$ (hiszen bármely él párosítást alkot, de három csúcson két élű párosítás nyilván nem létezhet) és $\tau(G) = 2$ (mert bármely két csúcs lefogó ponthalmazt alkot, de egyetlen csúcs még nyilván nem fogná le a háromszög mindhárom oldalát). De olyan gráfra is könnyű példát mutatni, amire $\nu(G)$ és $\tau(G)$ között a különbség tetszőlegesen nagy: az n csúcsú teljes gráfra $\nu(K_n) = \lfloor \frac{n}{2} \rfloor$ (mert ennyi élű párosítást a csúcsok tetszőleges párbaállításával lehet kapni, de ennél nagyobb párosításhoz már nincs elég csúcsa a gráfnak) és $\tau(K_n) = n - 1$ (mert a gráf tetszőleges $n-1$ csúcsa lefogó ponthalmazt alkot, de ha ennél kisebb csúcshalmazt veszünk, akkor az abból kimaradó bármely két csúcs között futó él nincs lefogva).

Hasonlóan ahhoz, amit a 6.3. szakaszban $\omega(G)$ és $\chi(G)$ viszonyáról írtunk, megint azt mondhatjuk, hogy a párosítások és lefogó ponthalmazok szempontjából azok a G gráfok „szerencsések”, amikre $\nu(G) = \tau(G)$ teljesül. Ez pontosabban azt jelenti, hogy ha egy G gráfban sikerül találni egy k élű párosítást és egy k csúcsú lefogó ponthalmazt, akkor ebből (a 7.5. Feladat megoldásában látotthoz hasonlóan) $k \leq \nu(G) \leq \tau(G) \leq k$, vagyis $\nu(G) = \tau(G) = k$ következik. Más szóval: egy azonos méretű párosítás és lefogó ponthalmaz bizonyítják egymás optimalitását (hasonlóan ahhoz, ahogyan egy k színnel való színezés és egy k csúcsú klikk esetén történt). Ha viszont egy gráfban nem lehet azonos méretű párosítást és lefogó ponthalmazt találni (mert $\nu(G) < \tau(G)$), akkor ravaszabb indoklásokra lehet szükség $\nu(G)$ és $\tau(G)$ meghatározásához.

7.1. Független ponthalmaz és lefogó élhalmaz

Ebben a szakaszban két olyan fogalomról lesz szó, amik szoros kapcsolatban állnak a fentiekben bevezetett független élhalmaz és lefogó ponthalmaz fogalmával – mégpedig több értelemben is: egyrészt az értelmezésük is részben analóg, másrészt látni fogjuk, hogy fontos összefüggések kapcsolják össze őket.

7.6. Definíció. *A G gráf csúcsainak egy $Y \subseteq V(G)$ halmazát* független ponthalmaznak *nevezzük, ha Y-nak semelyik két eleme nem szomszédos G-ben (és Y-beli csúcsra hurokél sem illeszkedik). A G-beli független ponthalmazok közül a maximálisaknak a méretét $\alpha(G)$ jelöli.*

A független ponthalmaz fogalmával érintőlegesen már találkoztunk (bár nem neveztük így) a gráfok színezése kapcsán: ha G csúcsait a 6.1. Definíciónak megfelelően kiszínezzük, akkor az azonos színűre festett csúcsok független ponthalmazt alkotnak (hiszen azonos színű csúcsok között nem futhat él). A maximális független ponthalmaz keresésének a feladata is olyan gyakorlati alkalmazásokban merülhet fel, amikor a gráf élei valamilyen fajta ütközéseket fejeznek ki és a cél egy lehető legnagyobb ütközésmentes csúcshalmaz keresése. Ha például a G gráf csúcsai egy céghez befutó megrendelések és két csúcs között akkor vezet él, ha a megfelelő két megrendelés kizárja egymást (abban az értelemben, hogy erőforrás korlátok miatt nem tudják mindkettőt elvállalni), akkor egy maximális független ponthalmaz a lehető legtöbb megrendelésnek felel meg, amiket a cég még el tud vállalni.

A független ponthalmaz és $\alpha(G)$ kapcsán is megismételjük, amit $\nu(G)$ és $\tau(G)$ kapcsán is hangsúlyoztunk: a független ponthalmaz maximalitása sem azt jelenti, hogy ne lehetne egy további csúcsot hozzávenni a függetlenség megtartásával, hanem azt, hogy nincs nála nagyobb; így $\alpha(G) = k$ azt jelenti, hogy G-ben van k csúcsú független ponthalmaz, de $k+1$ csúcsú már nincs. $\alpha(G)$-t pedig a G-beli „független pontok maximális számának” is szokták hívni.

Érdemes megemlíteni, hogy a független ponthalmazok, illetve $\alpha(G)$ szoros kapcsolatban állnak a 6. fejezetben tárgyalt másik fogalompárral, a klikkel, illetve $\omega(G)$-vel is. Ha ugyanis a G egyszerű gráfról áttérünk annak a $\overline{G}$ komplementerére, akkor egy olyan G-beli Y csúcshalmazból, aminek semelyik két tagja nem szomszédos, $\overline{G}$-ben olyan csúcshalmazt kapunk, aminek bármely két tagja szomszédos. Más szóval: a G-beli független ponthalmazok a $\overline{G}$-beli klikkeknek felelnek meg és viszont; ebből következően $\alpha(G) = \omega(\overline{G})$ és $\omega(G) = \alpha(\overline{G})$ is igaz.

7.7. Definíció. *Az izolált pontot nem tartalmazó G gráf éleinek egy $Z \subseteq E(G)$ halmazát* lefogó élhalmaznak *nevezzük, ha G-nek minden csúcsára illeszkedik legalább egy Z-beli él. A G-beli lefogó élhalmazok közül a minimálisaknak a méretét $\rho(G)$ jelöli.*

$\rho(G) = k$ tehát azt jelenti, hogy G-nek van k élű lefogó élhalmaza, de $k-1$ élű már nincs; $\rho(G)$-t pedig a G-beli „lefogó élek minimális számának” is hívják. A definícióban nyilván azért kellett kikötni az izolált pont mentességet, mert különben G-nek egyáltalán nem létezhetne lefogó élhalmaza. A ρ görög betű, a kiejtése: „ró”.

Minimális lefogó élhalmaz keresése is alkalmazásokban felmerülő probléma; a következő példa ugyan kevésbé gyakorlati jellegű, de illusztrációnak alkalmas. Képzeljük el, hogy egy csoport gyerek valamilyen páros játékot szeretne játszani,

például libikókázni. A G gráf csúcsai legyenek a gyerekek és két csúcs akkor legyen szomszédos, ha a megfelelő két gyerek szívesen libikókázna egymással (mert például nem túl nagy köztük a súlybeli különbség). Ekkor egy G-beli lefogó élhalmaz annak felel meg, hogy a gyerekeket úgy állítjuk párba, hogy mindenki legalább egyszer libikókázzon; $\rho(G)$ pedig az ehhez szükséges libikókázások számának a minimumát adja meg.

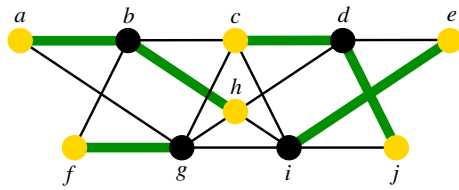
Az alábbi állítás egy, a független ponthalmazok és a lefogó élhalmazok mérete közötti fontos kapcsolatról szól – analóg módon azzal, amit a 7.4. Állítás mondott a független élhalmazokról és a lefogó ponthalmazokról.

7.8. Állítás. *Minden (izolált pontot nem tartalmazó) G gráfra $\alpha(G) \leq \rho(G)$ teljesül.*

Bizonyítás: Legyen G-ben Y egy maximális független ponthalmaz és Z egy minimális lefogó élhalmaz; ekkor tehát $|Y| = \alpha(G)$ és $|Z| = \rho(G)$. Mivel Z lefogó élhalmaz, ezért minden csúcsra illeszkedik Z-beli él, így ez nyilván Y csúcsaira is igaz. Azonban ugyanaz a Z-beli él nem illeszkedhet két Y-beli csúcsra is, mert Y független ponthalmaz. Más szóval: minden Y-beli csúcsra másik Z-beli élnek kell illeszkednie. Ebből tehát $|Y| \leq |Z|$ és így $\alpha(G) \leq \rho(G)$ valóban következik. □

7.9. Feladat. Határozzuk meg a 7.2. Feladat gráfjában a független pontok maximális számát, $\alpha(G)$-t és a lefogó élek minimális számát, $\rho(G)$-t.

Megoldás: A 7.5. ábrán sárgával jelölt $Y = \{a,c,e,f,h,j\}$ csúcshalmaz független (mert semelyik két tagja nem szomszédos) és az ábrán zölddel kiemelt $Z = \big\{\{a,b\},\{b,h\},\{c,d\},\{d,j\},\{e,i\},\{f,g\}\big\}$ élhalmaz lefogó (mert G minden csúcsára illeszkedik legalább egy Z-beli él).



7.5. ábra

Mivel létezik hat elemű független ponthalmaz, ezért $\alpha(G) \geq 6$. Hasonlóan, a hat elemű lefogó élhalmaz létezéséből $\rho(G) \leq 6$ következik. Ezeket összevetve a 7.8. Állítással kapjuk, hogy $6 \leq \alpha(G) \leq \rho(G) \leq 6$. Ebből tehát $\alpha(G) = \rho(G) = 6$ adódik. □

A 7.5 és a 7.9. Feladatok megoldását összevetve feltűnhet egy fontos kapcsolat az ugyanabban a gráfban talált $X = \{b,d,g,i\}$ minimális lefogó ponthalmaz és az

$Y = \{a,c,e,f,h,j\}$ maximális független ponthalmaz között: ezek egymás komplementerei (vagyis pontosan azok a csúcsok kerültek Y-ba, amelyek X-ből hiányoztak). Az alábbi állítás azt mondja ki, hogy ez a jelenség egyáltalán nem véletlen.

7.10. Állítás. *Legyen G tetszőleges gráf, $X \subseteq V(G)$ csúcshalmaz és $Y = V(G) \setminus X$ az X komplementere. Ekkor az X pontosan akkor minimális lefogó ponthalmaz G-ben, ha Y maximális független ponthalmaz G-ben.*

Bizonyítás: Egy tetszőleges $H \subseteq V(G)$ csúcshalmaz esetén jelölje $\overline{H} = V(G) \setminus H$ a H komplementer halmazát. Először azt mutatjuk meg, hogy H pontosan akkor lefogó ponthalmaz, ha $\overline{H}$ független ponthalmaz. Az, hogy H lefogó ponthalmaz azt jelenti, hogy G-nek minden $\{u,v\}$ élére $u \in H$ vagy $v \in H$ (vagy mindkettő) teljesül. Más szóval: G-nek nincs olyan $\{u,v\}$ éle, hogy $u \in \overline{H}$ és $v \in \overline{H}$. Ez pedig pontosan azt jelenti, hogy $\overline{H}$ független ponthalmaz.

Rátérve az állítás bizonyítására, tegyük fel először, hogy X minimális lefogó ponthalmaz. Az előző bekezdésben írtakból következik, hogy $Y = \overline{X}$ független ponthalmaz. Ha Y nem volna maximális független ponthalmaz, akkor létezne egy olyan Y_1 független ponthalmaz, amire $|Y_1| > |Y|$. Ekkor viszont ismét az előző bekezdésben írtak miatt $\overline{Y_1}$ az X-nél kisebb lefogó ponthalmaz volna, ami ellentmondás. Így Y valóban maximális független ponthalmaz.

Megfordítva, most azt tegyük fel, hogy Y maximális független ponthalmaz. Ekkor, hasonlóan a fentiekhez, $X = \overline{Y}$ lefogó ponthalmaz; és ha nem volna minimális, akkor egy X-nél kisebb lefogó ponthalmaz komplementere Y-nál nagyobb független ponthalmazt adna, ami ellentmondás. □

A fenti állításnak fontos következménye, hogy a minimális lefogó ponthalmaz és a maximális független ponthalmaz keresésének problémái algoritmikusan ekvivalens feladatok. Más szóval: ha a két probléma közül bármelyikre megadunk egy algoritmust, akkor az (minimális módosítással) egyben a másik megoldására is használható. Az alábbiakból az fog kiderülni, hogy hasonló kapcsolat a maximális párosítás és a minimális lefogó élhalmaz keresésének problémái között is létezik – bár ez kicsit bonyolultabb; ehhez szükségünk lesz az alábbi lemmára.

7.11. Lemma. *Legyen G egy n csúcsú, izolált pontot nem tartalmazó gráf, k pedig tetszőleges nemnegatív egész. Ekkor*

(i) ha G-ben van k élű párosítás, akkor G-ben van legföljebb $n-k$ élű lefogó élhalmaz;

(ii) ha G-ben van k élű lefogó élhalmaz, akkor G-ben van legalább $n-k$ élű párosítás.

Bizonyítás: Az *(i)* bizonyításához legyen M egy k élű párosítás G-ben. Minden olyan v csúcs esetén, amire nem illeszkedik M-beli él, válasszunk egy tetszőleges v-re illeszkedő élt és egészítsük ki M-et az összes így kiválasztott éllel; a kapott élhalmazt jelölje Z. Ekkor Z nyilván lefogó élhalmaz, hiszen úgy készült, hogy minden csúcsra tartalmazzon egy arra illeszkedő élt. Mivel M éleinek összesen $2k$ végpontja van,

így az M által le nem fedett csúcsok száma $n-2k$; a Z készítésekor tehát ennyi csúcs esetén választottunk egy-egy arra illeszkedő élt. Ugyan előfordulhat, hogy két különböző ilyen csúcshoz ugyanazt az élt választottuk (ha ezek szomszédosak), de akkor is legföljebb $n-2k$ éllel egészítettük ki M-et. Így $|Z| \leq k+(n-2k) = n-k$, amivel *(i)*-et beláttuk.

Rátérve *(ii)* bizonyítására, legyen Z egy k élű lefogó élhalmaz G-ben és $H = (V(G), Z)$ az a gráf, ami G összes csúcsából és Z éleiből áll. Ha H-nak van egy csúcsú komponense, akkor minden ilyen csúcsra kell illeszkedjen Z-beli hurokél, különben Z nem volna lefogó. Jelöljük H legalább két csúcsú komponenseinek a számát c-vel. Mivel a komponensek nyilván összefüggők, ezért az 1.23. Tétel szerint mindegyiknek legalább annyi éle van, mint a csúcsainak a száma mínusz egy. Ezért a legalább két csúcsú komponensek együttes élszáma legalább annyi, mint az együttes csúcsszámuk mínusz c. Az egy csúcsból álló komponensek együttes élszáma pedig legalább az együttes csúcsszámuk, hiszen a fentiek szerint mindegyik tartalmaz legalább egy hurokélt. Mindebből tehát az következik, hogy H összes élszáma legalább az összes csúcsszáma mínusz c; mivel H-nak n csúcsa és k éle van, kapjuk, hogy $k \geq n-c$. Átrendezve: $c \geq n-k$.

Válasszunk most ki H minden legalább két csúcsú komponenséből egy-egy tetszőleges élt, ami nem hurokél; a kapott élhalmazt jelölje M. Ekkor $|M| = c \geq n-k$, hiszen c a legalább két csúcsú komponensek száma. Másrészt M nyilván párosítás, hiszen különböző komponensekből vett éleknek nem lehet közös végpontja. Ezzel tehát *(ii)* bizonyítása teljes. □

A 7.10. Állításnak és a 7.11. Lemmának együtt a szakaszban eddig szereplő fogalmakat összekapcsoló érdekes következményei vannak. A következő tétel *(ii)* állítása Gallai Tibor (1912 – 1992) magyar matematikustól származik – és bár a tétel *(i)* állítása történetileg nem köthető hozzá (sem más matematikushoz, mert ez a benne szereplő fogalmak definíciójának egyszerű következménye), a két állítást mégis szokás közös keretben kimondani az általuk mutatott szimmetria miatt.

7.12. Tétel. (Gallai tétele)
Minden n csúcsú G gráfra fennállnak az alábbiak:
(i) $\alpha(G)+\tau(G) = n$;
(ii) $\nu(G)+\rho(G) = n$, *ha G-nek nincs izolált pontja.*

Bizonyítás: Ha X egy minimális lefogó ponthalmaz G-ben, akkor a 7.10. Állítás szerint $Y = V(G) \setminus X$ maximális független ponthalmaz. Mivel $|X| = \tau(G)$, $|Y| = \alpha(G)$ és $|X|+|Y| = n$, ebből az *(i)* állítás rögtön következik.

A *(ii)* bizonyításához legyen M egy maximális, $\nu(G)$ élű párosítás G-ben. Ekkor a 7.11. Lemma *(i)* állítása szerint G-ben van legföljebb $n-\nu(G)$ élű lefogó élhalmaz. Ebből tehát $\rho(G) \leq n-\nu(G)$ és így $\nu(G)+\rho(G) \leq n$ adódik.

Megfordítva, legyen most Z egy minimális, $\rho(G)$ élű lefogó élhalmaz G-ben. Ekkor a 7.11. Lemma *(ii)* állítása szerint G-ben van legalább $n-\rho(G)$ élű párosítás. Ebből tehát $\nu(G) \geq n-\rho(G)$ és így $\nu(G)+\rho(G) \geq n$ adódik.

Összevetve a kapott egyenlőtlenségeket $\nu(G)+\rho(G) = n$ valóban következik. □

Ezt a tételt használva például a 7.9. Feladatot sokkal gyorsabban is megoldhattuk volna: mivel a 7.5. Feladatban már ugyanarról az $n = 10$ csúcsú G gráfról beláttuk, hogy rá $\nu(G) = \tau(G) = 4$, ezért a tétel két állításából sorra $\alpha(G) = n - \tau(G) = 6$ és $\rho(G) = n - \nu(G) = 6$ adódhatott volna.

Fontos kiemelni, hogy a 7.12. Tétel *(ii)* állításából és a 7.11. Lemma bizonyításából együtt valóban következik az, amit a 7.11. Lemma kimondása előtt állítottunk: a maximális párosítás és a minimális lefogó élhalmaz keresésének problémái algoritmikusan ekvivalensek. Valóban, ha a 7.11. Lemma *(i)* állításának bizonyítását egy $\nu(G)$ élű párosításra alkalmazzuk, akkor egy $n - \nu(G) = \rho(G)$ élű, vagyis minimális lefogó élhalmazt kapunk; ha pedig a 7.11. Lemma *(ii)* állításának a bizonyítását alkalmazzuk egy $\rho(G)$ élű lefogó élhalmazra, akkor egy $n - \rho(G) = \nu(G)$ élű párosítást kapunk. Ezek közül az előbbire láttunk is már példát a 7.9. Feladat megoldásában: a 7.5. ábrán látható Z lefogó élhalmaz pontosan úgy keletkezett a 7.3. ábrán látható M párosításából, ahogyan az a 7.11. Lemma bizonyításának az első bekezdésében történt: az M fedetlenül hagyta a c és a h csúcsokat (vagyis ezekre nem illeszkedett M-beli él) és Z éppen M éleiből, valamint a c-re, illetve a h-ra illeszkedő $\{c,d\}$, illetve $\{b,h\}$ élekből állt. De ugyanez fordítva is működik: a 7.5. ábrán látható Z lefogó élhalmaznak (a gráf összes csúcsával együtt) négy komponense van, amik mind legalább két csúcsúak; ha mindegyikből egy-egy élt választunk, akkor valóban négy élű, vagyis maximális párosítást kapunk. Megkapható így például a 7.3. ábrán látható M párosítás is: ha a három csúcsú komponensekből a $\{d,j\}$ és az $\{e,i\}$ éleket választjuk.

Érdemes megfigyelni, hogy a 7.12. Tétel a 7.4. és a 7.8. Állítások között is kapcsolatot teremt: mivel a tétel szerint (izolált pontot nem tartalmazó gráfokban) $\nu(G) + \rho(G) = \tau(G) + \alpha(G)$, ezért a $\nu(G) \leq \tau(G)$ és a $\rho(G) \geq \alpha(G)$ egyenlőtlenségek közül bármelyikből következik a másik.

Az ebben a szakaszban vizsgált két problémáról, a maximális független ponthalmaz, illetve a minimális lefogó élhalmaz kereséséről kiderült, hogy ezek algoritmikusan ekvivalensek a minimális lefogó ponthalmaz, illetve a maximális független élhalmaz keresésének problémáival. Az érdekesség kedvéért megemlítjük, hogy e közül a négy feladat közül az élhalmazokról szóló két feladat hatékonyan megoldható: ismert olyan (az amerikai Jack Edmonds (1934 –) által 1965-ben publikált) polinomiális futásidejű algoritmus, ami egy tetszőleges gráfban meghatároz egy maximális párosítást; ezt használva tehát a fentiek szerint minimális lefogó élhalmazt is lehet kapni. Ez az algoritmus azonban igen bonyolult, ebben a jegyzetben nem foglalkozunk vele. A maximális független ponthalmaz és a minimális lefogó ponthalmaz meghatározása (tetszőleges gráfban) azonban algoritmikusan nagyon nehéz feladatok: ezekre nem ismert polinomiális algoritmus és jó okunk van feltételezni, hogy ilyen nem is létezik.

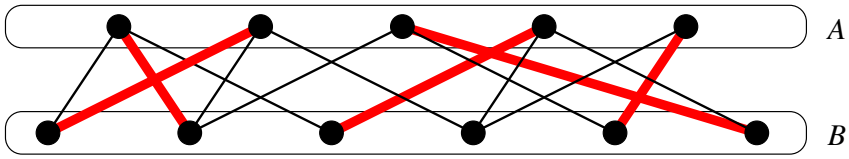
7.2. Párosítások páros gráfban

Egy céghez különböző megrendelések futnak be, amiket a cég vezetése dolgozókhoz szeretne hozzárendelni. Minden munkát egyetlen dolgozó végezhet el és a dol-

gozók egyszerre csak egy feladaton tudnak dolgozni. Azonban egy adott megrendelést nem minden dolgozó tud elvégezni (mert a munkák jellege és az egyes dolgozók képzettségei között különbségek vannak). A helyzetet egy $G = (A, B; E)$ páros gráffal ábrázolhatjuk: az A csúcshalmazt a munkák, a B-t pedig a dolgozók alkotják és egy $a \in A$ munkát akkor kötünk össze egy $b \in B$ dolgozóval, ha b el tudja végezni a-t. Tegyük fel, hogy a cégvezetés célja az, hogy a lehető legtöbb megrendelést vállalják el (mert ezáltal tudják maximalizálni a profitot). Ekkor a feladatuk nem más, mint hogy G-ben egy maximális párosítást kell találniuk.

Ebben a szakaszban páros gráfok párosításairól lesz szó; a fenti példa azt illusztrálja, hogy ez a probléma gyakorlati alkalmazásokban is felvetődő, fontos feladat. Meg fogunk ismerni egy hatékony algoritmust, ami minden páros gráfban meghatároz egy maximális párosítást. Emellett fontos elméleti eredményekre is jutunk: választ kapunk például arra a kérdésre, hogy mitől függ az, hogy egy G páros gráfban van-e teljes párosítás.

Emellett egy ennél kicsit általánosabb kérdést is vizsgálni fogunk, aminek már speciálisan csak páros gráfok esetén van értelme: mitől függ az, hogy egy $G = (A, B; E)$ páros gráfban létezik-e A-t lefedő párosítás. Általában ha egy G gráfban adott egy M párosítás és egy $X \subseteq V(G)$ csúcshalmaz, akkor azt mondjuk, hogy *M lefedi X-et*, ha minden X-beli csúcsra illeszkedik M-beli él. A céghez befutó megrendelésekről szóló fenti példa mutatja, hogy egy $G = (A, B; E)$ páros gráfban A-t lefedő párosítás keresése (amikor tehát minden A-beli csúcsra illeszkedik M-beli él, de B csúcsaira ez már nem követelmény) gyakorlati alkalmazásokban is felmerülhet: ha a cég minden megrendelést el akar vállalni (de az nem baj, ha egyes dolgozók nem jutnak munkához). A 7.6. ábra például egy A-t lefedő párosítást mutat egy $G = (A, B; E)$ páros gráfban, ami azonban nem teljes párosítás, hiszen B-nek van a párosítás által nem fedett csúcsa. Ebben a gráfban teljes párosítás nyilván nem is létezhet, hiszen B-nek több csúcsa van, mint A-nak. Általában is elmondható, hogy egy A-t lefedő párosítás pontosan akkor teljes párosítás, ha $|A| = |B|$.



7.6. ábra

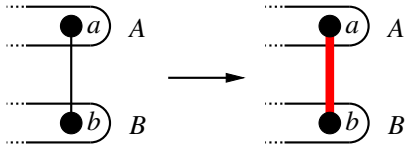
A későbbi félreértések elkerülése érdekében fontos hangsúlyozni, hogy a „páros gráf” és a „párosítás” kifejezések ugyan hasonlítanak egymásra, de a jelentésük egészen más, nem szabad összekeverni őket. Fentebb már kiderült, hogy a párosítás fogalma tetszőleges, nem feltétlen páros gráfokban is értelmes; és a páros gráfoknak is vannak egészen más, a párosításoktól különböző alkalmazási területei.

7.2.1. A javítóutas algoritmus

Az ebben a pontban bemutatásra kerülő algoritmus Kőnig Dénes (1884 – 1944) magyar matematikustól származik 1931-ből.

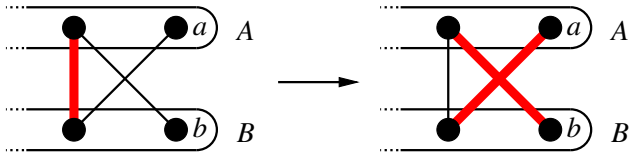
Az algoritmus alapgondolata az, hogy folyamatosan nyilvántart egy aktuális M párosítást, amit egy növelő lépés ismételgetésével addig javít, amíg csak lehetséges. Az eljárás „lelke” tehát egy olyan lépés, amivel egy adott párosításból egy annál eggyel több élt tartalmazó másik párosítást lehet előállítani.

Milyen helyzet teszi lehetővé azt, hogy az aktuális M méretét növeljük? A legszerencsésebb ilyen helyzet az, ha találunk a gráfban egy olyan $\{a,b\}$ élt, aminek egyik végpontját sem fedi az aktuális párosítás (vagyis sem a-ra, sem b-re nem illeszkedik M-beli él). Ilyenkor ezt az élt egyszerűen hozzávehetjük M-hez, ahogyan az a 7.7. ábrán látható.



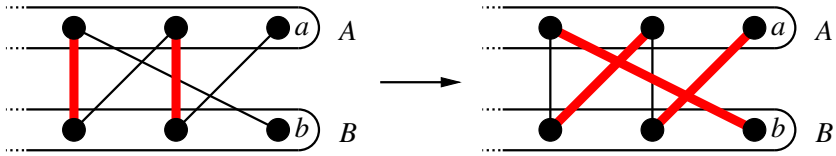
7.7. ábra

Sajnos azonban nagyon is könnyen előfordulhat, hogy M-hez nem lehet egyszerűen egy további élt hozzávenni, pedig van M-nél nagyobb párosítás: erre már láttunk példát a 7.1. ábrán is (hiszen az azon látható három hosszú út nyilván páros gráf). Lehetséges tehát, hogy M növelése érdekében „fel kell bontanunk egy aktuális párt” (vagyis egy élt ki kell hagynunk M-ből). Ez akkor segít, ha olyan szerencsés helyzetet találunk, amilyet a 7.8. ábra bal oldala mutat: ha az $a \in A$ és $b \in B$ csúcsokra nem illeszkedik M-beli él, de A-ból B-be vezet egy olyan három élű út, aminek a középső éle M-beli. Ekkor ezt az élt kihagyhatjuk M-ből és helyette az út két szélső élét bevehetjük M-be (ahogyan azt a 7.8. ábra jobb oldala mutatja).



7.8. ábra

Sajnálatos módon azonban még az is előfordulhat, hogy a 7.8. ábra szerint sem tudjuk M-et növelni, pedig az nem maximális. Lehetséges, hogy M-ből két élt is ki kell hagynunk a növelés érdekében: ha az M által nem fedett $a \in A$ és $b \in B$ csúcsok között egy olyan öt élű út van, aminek a második és negyedik éle M-beli, akkor a 7.9. ábra szerint ezeket kihagyva és helyettük az út másik három élét M-hez véve növelhető a párosítás mérete.



7.9. ábra

A fenti három ábra a legegyszerűbb speciális eseteit mutatja az algoritmus működéséhez szükséges alábbi kulcsfogalomnak.

7.13. Definíció. *Legyen $G = (A, B; E)$ páros gráf és M egy párosítás G-ben. Ekkor egy G-beli P út* javítóút *M-re nézve, ha rá az alábbiak teljesülnek:*
(1) P egy M által nem fedett A-beli csúcsból indul;
(2) P egy M által nem fedett B-beli csúcsban ér véget;
(3) P-nek minden páros sorszámú éle (tehát a második, negyedik stb.) M-beli.

A javítóutas algoritmus úgy működik, hogy egy tetszőleges párosításból kiindulva javítóutak ismételt keresésével addig növeli a párosítás méretét, amíg ez lehetséges. A kezdeti párosítás akár az üres halmaz is lehet, de bármely más párosításból is indítható az eljárás.

JAVÍTÓUTAS ALGORITMUS (MAXIMÁLIS PÁROSÍTÁS KERESÉSÉRE PÁROS GRÁFBAN)

Bemenet: Egy $G = (A, B; E)$ páros gráf

1 $M \leftarrow \emptyset$ (vagy ehelyett M lehet tetszőleges kezdeti párosítás is)
2 **ciklus**
3 KERESSÜNK EGY P JAVÍTÓUTAT az M párosításra nézve
4 **ha** nem létezik M-re nézve javítóút, **akkor**: **stop**
5 **különben**:
6 $M \leftarrow M \setminus \{P$ páros sorszámú élei$\} \cup \{P$ páratlan sorszámú élei$\}$
7 **ciklus vége**

A 7.13. Definícióból következik, hogy egy P javítóútnak a hossza (éleinek a száma) páratlan kell legyen, hiszen P egyik végpontja A-ban, a másik B-ben van. Így P-nek pontosan eggyel kevesebb M-beli éle van, mint M-be nem tartozó. Ezért a 6. sor végrehajtásakor M mérete mindig pontosan eggyel nő.

A fenti pszeudokódból látszik, hogy az ebben a formájában még nem tekinthető valódi algoritmusnak, hiszen egyáltalán nem nyilvánvaló, hogy a 3. sorban a javítóút keresését (illetve a létezésének az eldöntését) hogyan kell megvalósítani. Messze nem világos, hogy ezt hogyan lehet megoldani, ha például G egy sok ezer csúcsú páros gráf és abban egy néhány ezer élű M párosítás adott. Szerencsére azonban a 2. fejezetben látott BFS algoritmus minimális módosításával ez a feladat könnyen

megoldható. Ehhez két ponton kell változtatni a BFS eljáráson. Az első, hogy a BFS-t nem egyetlen kezdőpontból indítjuk, hanem az összes, M által nem fedett A-beli csúcsból egyszerre. Ha tehát A_1 jelöli ezeknek a csúcsoknak a halmazát, akkor a BFS eljárásnak a 2. fejezetben írt pszeudokódja szerint az $\mathbf{L}$ listát kezdetben A_1 elemeivel töltjük fel (tetszőleges sorrendben) és minden $a \in A_1$ csúcs táv(a) értékét 0-nak inicializáljuk. A BFS eljárás második módosítása pedig ahhoz szükséges, hogy a javítóút minden második éle M-beli legyen. Ez nagyon egyszerűen elérhető: az eljárás 6-11. soraiban zajló belső ciklusban a v csúcs nem *aktív_csúcs* minden szomszédján fut végig, hanem csak az M-beli, vagy M-be nem tartozó élek mentén vett szomszédjain táv$(\textit{aktív_csúcs})$ paritásától (vagyis az aktív csúcsnak az A_1-beliektől való távolságától) függően. Pontosabban: ha táv$(\textit{aktív_csúcs})$ páratlan és így $\textit{aktív\_csúcs} \in B$, akkor v *aktív_csúcs*-nak csak az egyetlen M szerinti párja lehet (ha van ilyen); ha viszont táv$(\textit{aktív_csúcs})$ páros és így $\textit{aktív\_csúcs} \in A$, akkor v *aktív_csúcs* összes szomszédján végigfut. Ha pedig az utóbbi esetben *aktív_csúcs*-nak olyan $v \in B$ szomszédját találjuk meg, amire táv$(v) = \infty$ és v-t M nem fedi le, akkor javítóutat találtunk: ezt a $v_0 = v$, $v_1 =$ előző(v_0), $v_2 =$ előző(v_1), ... sorozaton egy A_1-beli csúcsig visszafelé lépegetve kapjuk. Ha pedig az eljárás lezajlik anélkül, hogy ez bekövetkezne, akkor nincs javítóút G-ben M-re nézve.

Megjegyezzük, hogy a BFS algoritmus fenti két módosítása kiváltható azzal is, hogy az eljárást egy erre a célra átalakított irányított gráfon futtatjuk. Vegyünk fel ugyanis egy új s és egy t pontot; s-ből vezessünk egy-egy irányított élt minden A_1-beli csúcsba és hasonlóan, B minden, M által le nem fedett csúcsából vezessünk egy irányított élt t-be. Továbbá G-nek az M-beli éleit irányítsuk a B-beli végpontjuktól az A-beli felé, az M-be nem tartozó éleket pedig az A-beli végpontjuktól a B-beli felé. Ha az így kapott irányított gráfban találunk a BFS algoritmussal s-ből t-be egy irányított utat, akkor az az eredeti gráfban épp egy javítóútnak felel meg; ha viszont ilyen irányított út nincs, akkor javítóút sincs G-ben.

Ezzel a visszavezetéssel tehát a BFS eredeti (irányított gráfokra vonatkozó) változatát is alkalmassá tehetjük javítóút keresésére, de a gyakorlatban ezeknek az átalakításoknak az elvégzésénél hatékonyabb, ha az eredeti gráfon futtatjuk a fentiek szerint módosított BFS eljárást.

Anélkül, hogy az (implementációtól is erősen függő) részletekben elmélyednénk, fontos kiemelni, hogy a javítóutas algoritmus nagyon hatékony, polinomiális algoritmus. Valóban: a fentiekből kiderült, hogy tulajdonképpen a csúcsok számával arányos számban ismételt BFS algoritmusról van szó (hiszen minden javító lépés eggyel növeli a párosítás élszámát, de az nem nőhet nagyobbra, mint a csúcsok számának a fele). Így a javítóutas algoritmus hatékonysága következik abból, hogy a BFS algoritmus polinomiális (sőt: a gráfot szomszédsági listával tárolva az élszám és a csúcsszám összegével arányos) lépésszámú.

7.14. Feladat. Egy dzsungel mélyén élő törzsben kilenc fiú (a,b, …, i) és tíz lány (1-től 10-ig) ért házasulandó korba. A törzsfőnök felmérte, hogy ki kivel hajlandó frigyre lépni, az eredményei az alábbi táblázatban láthatók. A törzsfőnök szeretne a fiatalokból a lehető legtöbb házaspárt összeállítani (úgy, hogy mindenki olyannal házasodjon, akivel hajlandó).

a) A javítóutas algoritmus segítségével oldjuk meg a törzsfőnök feladatát.

b) Sajnos e és 1 összeveszett, többé már nem hajlandók egymáséi lenni. Oldjuk meg a feladatot erre az esetre is.

	1	2	3	4	5	6	7	8	9	10
a	♥						♥		♥	♥
b		♥			♥					
c			♥		♥	♥		♥	♥	
d				♥			♥	♥	♥	
e	♥			♥	♥					
f						♥	♥	♥		♥
g		♥	♥	♥						
h		♥	♥							
i		♥	♥							

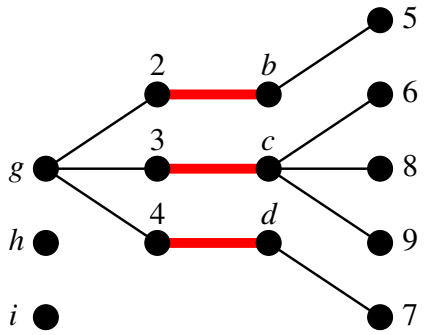
Megoldás: Álljon az A halmaz a fiúkból, B a lányokból és a $G = (A,B;E)$ páros gráfban egy $x \in A$ fiú akkor legyen szomszédos egy $y \in B$ lánnyal, ha a törzsfőnök táblázata szerint alkothatnak házaspárt. A javítóutas algoritmust G-ben fogjuk futtatni. Természetesen az eljárásnak sok helyes futása van (mert az aktuális M párosításra nézve több javítóút is létezhet), alább ezek közül bemutatunk egyet.

Az eljárást az üres párosítással indítjuk, majd az hatszor egymás után egy élű javítóutakat talál: először az $\{a,1\}$ élből állót, amit az $M = \emptyset$ párosításhoz véve egy élű párosítást kap (a 7.7. ábrának megfelelően). Majd sorra a $\{b,2\}$, $\{c,3\}$, $\{d,4\}$, $\{e,5\}$ és $\{f,6\}$ élekből álló egy élű javítóutakat találja és ezeket az éleket egymás után a párosításhoz veszi. Ekkor tehát az aktuális M párosítás hat élű:

a	b	c	d	e	f	g	h	i
1	2	3	4	5	6			

Az M által le nem fedett A-beli, illetve B-beli csúcsok halmaza most $A_1 = \{g,h,i\}$, illetve $B_1 = \{7,8,9,10\}$. Mivel G-nek már nincs A_1 és B_1 között futó éle, így egy élű javítóút ezen a ponton már nem lesz. Ezért a BFS eljárás fent leírt, módosított változatával keresünk javítóutat; ez látható a 7.10. ábrán.

Az ábrán a függőlegesen egymás fölött elhelyezkedő, egy oszlopban álló csúcsok kapnak azonos táv(v) értéket; egy oszlopon belül pedig felülről lefelé haladunk a csúcsok vizsgálatával. Így tehát az A_1-beli csúcsok táv(v) értéke 0, ezek szomszédainak, a 2, 3, és 4 csúcsoknak pedig 1. (Itt h-ból és i-ből nem ért el új csúcsot az eljárás, de ez csak azért történt így, mert a működését g-vel kezdte. Ha például g helyett h-val kezdte volna, akkor 2-t és 3-at h-ból érte volna el és g-ből csak a

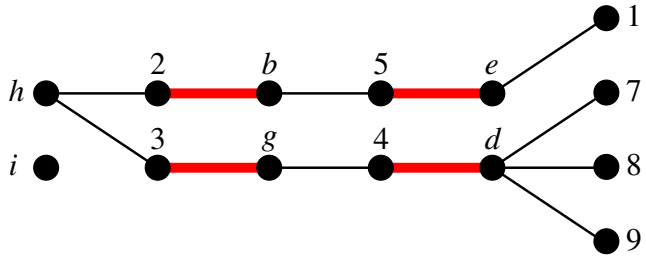


7.10. ábra

4-et.) Most a 2, 3 és 4 csúcsokból csak a rájuk illeszkedő M-beli éleken mehetünk tovább, így érjük el sorra a b, c és d csúcsokat (amelyeknek a táv(v) értéke tehát 2). Ezekből megint G tetszőleges élein haladhatunk tovább, így kapjuk az ábra utolsó oszlopában álló csúcsokat. Ezek között azonban már van B_1-beli: 7, 8 és 9 is ilyen; így bármelyiküktől visszafelé lépegetve nyerhetünk javítóutat. Válasszuk például a 8-at (mert az eljárás először ezt találta meg), így a $g,3,c,8$ csúcsokból álló javítóutat kapjuk. E mentén javítva tehát a $\{c,3\}$ él kikerül M-ből, de helyette a $\{g,3\}$ és $\{c,8\}$ élek bekerülnek. Így az aktuális M párosítás az alábbi:

a	b	c	d	e	f	g	h	i
1	2	8	4	5	6	3		

Az algoritmus tehát most erre az M-re nézve keres javítóutat a módosított BFS segítségével. Amint az a 7.11. ábrán látszik, először a táv$(v) = 5$ értéket kapott csúcsok között talál az M által nem fedett csúcsot: a 7-est. Ebből (az előző(v) mutatók segítségével visszafelé lépegetve) a $h,3,g,4,d,7$ csúcsokból álló javítóutat kapjuk.

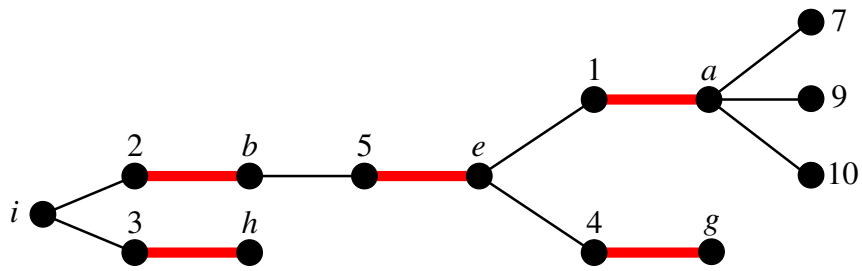


7.11. ábra

A javítóutat használva tehát M-ből kikerülnek a $\{g,3\}$ és $\{d,4\}$ élek és helyettük bekerülnek a $\{h,3\}$, $\{g,4\}$ és $\{d,7\}$ élek. Így az aktuális párosítás az alábbi:

a	b	c	d	e	f	g	h	i
1	2	8	7	5	6	4	3	

A-ban már csak egy M által le nem fedett csúcs van: az i. Így most ebből indítjuk a javítóút keresést a 7.12. ábra szerint.



7.12. ábra

Először a táv$(v) = 7$ értéket kapott csúcsok között találunk az M által nem fedettet: a 9-et. Így a hét hosszúságú, az $i, 2, b, 5, e, 1, a, 9$ csúcsokból álló javítóutat kapjuk. Ennek a három páros sorszámú élét kihagyjuk M-ből és helyettük a négy páratlan sorszámút bevesszük:

a	b	c	d	e	f	g	h	i
9	5	8	7	1	6	4	3	2

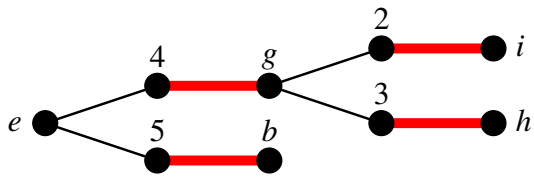
Ezzel tehát egy A-t lefedő párosítást kaptunk. Erre nézve nyilván nincs javítóút (hiszen nem létezik az M által nem fedett A-beli csúcs), ezért az algoritmus ezen a ponton megáll.

A b) feladatban e és 1 sajnálatos összeveszése nyilván azt jelenti, hogy az $\{e, 1\}$ élt ki kell hagyni a gráfból és az így kapott új gráfban kell maximális párosítást keresni. Külön balszerencse, hogy az $\{e, 1\}$ él benne van a fentebb kapott, A-t lefedő párosításban, így ugyanaz az M a b) feladat gráfjának már nem párosítása. Persze megtehetnénk, hogy ebben a gráfban újraindítjuk a javítóutas algoritmust az üres párosítástól kezdve. De mivel az algoritmus tetszőleges kezdeti párosítástól indítható, ennél sokkal jobban járunk, ha a fent kapott párosításból kihagyjuk az $\{e, 1\}$ élt és ettől a párosítástól indítjuk az eljárást. Így tehát a javítóutas algoritmust most az alábbi párosítástól indítjuk:

a	b	c	d	e	f	g	h	i
9	5	8	7		6	4	3	2

A 7.13. ábra mutatja a javítóút keresésére szolgáló módosított BFS eljárás futását.

Látható, hogy az M által le nem fedett B-beli csúcsok, 1 és 10 közül egyik sem érhető el, a javítóút keresés az ábra által mutatott ponton megáll. Így erre az M-re nézve nincs javítóút, ezért az algoritmus futása itt ér véget. □



7.13. ábra

Egyelőre természetesen nem tudjuk, hogy a fenti megoldás végén kapott párosítás valóban maximális-e a b) feladat gráfjában. Annak bizonyításában, hogy a javítóutas algoritmus mindig maximális párosítást ad, kulcsszerepet fog játszani az alábbi fogalom és a hozzá kapcsolódó lemma.

7.15. Definíció. *Legyen $G = (A,B;E)$ páros gráf és M egy párosítás G-ben. A G-beli P utat* alternáló útnak *hívjuk, ha rá a 7.13. Definíció (1) és (3) követelményei teljesülnek (de a (2) nem feltétlenül). Más szóval: alternáló útnak az olyan utakat nevezzük, amik párosítás által nem fedett A-beli csúcsból indulnak és minden második élük M-beli.*

7.16. Lemma. *Tegyük fel, hogy a $G = (A,B;E)$ páros gráf M párosítására nézve nincs javítóút G-ben. Vezessük be az alábbi jelöléseket:*

(1) jelölje A_1, illetve B_1 az M által nem fedett A, illetve B-beli csúcsok halmazát;

(2) jelölje A_2 azoknak az (M által fedett) A-beli csúcsoknak a halmazát, amikbe vezet alternáló út;

(3) jelölje A_3 a maradék A-beli csúcsoknak a halmazát (amik tehát M által lefedettek, de nem vezet hozzájuk alternáló út).

(4) Jelölje B_2, illetve B_3 az A_2, illetve A_3 csúcsainak M szerinti párjaiból álló B-beli csúcsok halmazait.

Ekkor G-nek nincs olyan éle, ami $A_1 \cup A_2$ és $B_1 \cup B_3$ között vezet.

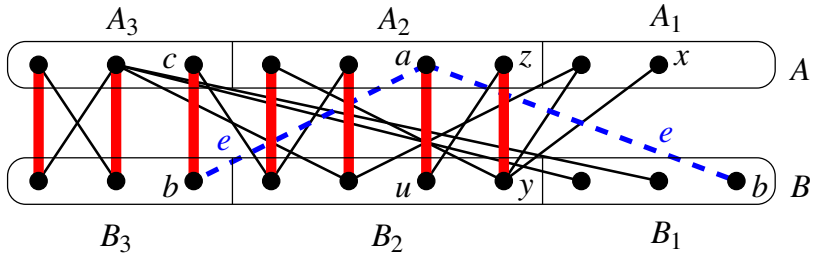
A lemma állítását illusztrálhatjuk a 7.14. Feladat b) részének a megoldásán. Az ott vizsgált, nyolc élű M párosításra nézve nem volt G-ben javítóút. Itt $A_1 = \{e\}$, illetve $B_1 = \{1,10\}$ voltak az M által fedetlen A, illetve B-beli csúcsok. Az M-re vonatkozó javítóút keresésekor a módosított BFS eljárás éppen az A_2-be, illetve B_2-be tartozó csúcsokat találta meg: $A_2 = \{b,g,h,i\}$ és $B_2 = \{2,3,4,5\}$ (lásd a 7.13. ábrát). Valóban láthatjuk, hogy az $A_1 \cup A_2 = \{b,e,g,h,i\}$ halmaz csúcsainak minden szomszédja a $B_2 = \{2,3,4,5\}$ csúcsok közül kerül ki. Ez nem is meglepő: ha bármelyik $(A_1 \cup A_2)$-beli csúcsnak lenne más szomszédja is, akkor azt a módosított BFS eljárásnak meg kellett volna találnia; lényegében ennek a gondolatnak a kifejtésén alapul a fenti lemma bizonyítása.

A 7.16. Lemma bizonyítása: Tegyük fel indirekt, hogy G-nek mégis létezik egy olyan $e = \{a,b\}$ éle, amire $a \in A_1 \cup A_2$ és $b \in B_1 \cup B_3$. Ez összesen négy esetet jelent, amiket sorban kizárunk.

Ha $a \in A_1$ és $b \in B_1$, akkor a-ból b-be egyetlen élű javítóút vezetne; ez lehetetlen, hiszen feltettük, hogy G-ben nincs javítóút.

Ha $a \in A_1$ és $b \in B_3$, akkor a-ból e-n át b-be, majd onnan a b-re illeszkedő M-beli élen át továbblépve egy A_3-beli csúcsba jutnánk egy két élű alternáló úton. Ez is lehetetlen, mert A_3-ba épp az alternáló úton nem elérhető A-beli csúcsok kerültek.

Tegyük most fel, hogy $a \in A_2$ és $b \in B_1$ (lásd a 7.14. ábrát). Ekkor a-ba vezet egy P alternáló út. Ennek az utolsó éle nyilván M-beli, hiszen P A-ból indul és A-ba is érkezik, így páros sok éle van (és minden második éle M-beli). Mivel P csúcsai nyilván alternáló úton elérhetők, ezért az első (A_1-beli) csúcsán kívül az A-beli csúcsai A_2-beliek, a B-beli csúcsai pedig ezeknek az M szerinti párjai, vagyis B_2-beliek. Így b biztosan nincs rajta P-n. Ezért P-t folytathatjuk az e élen át b-ig, amivel egy javítóutat kapunk. Ez ellentmondás, mert feltettük, hogy ilyen nem létezik. A 7.14. ábrán például az a-ba vezető egyik alternáló út sorra az x, y, z, u, a csúcsokon át haladt; ezt e-n át b-ig meghosszabbítva valóban javítóutat kapnánk.



7.14. ábra

Végül tegyük fel, hogy $a \in A_2$ és $b \in B_3$ (ismét lásd a 7.14. ábrát). Ekkor a fenti bekezdés gondolatmenetét megismételve az a-ig vezető P alternáló út utolsó éle M-beli. Ezért P-t először e-n át b-ig, majd a b-re illeszkedő M-beli éllel tovább meghosszabbítva egy P-nél kettővel hosszabb P' alternáló utat kapunk. Mivel azonban $b \in B_3$, ezért a P' út vége (amit a 7.14. ábrán c jelöl) A_3-beli. Ez ellentmondás, mert A_3-beli csúcsba nem vezethet alternáló út. □

A fenti lemma segítségével már be tudjuk bizonyítani az alábbi, Kőnig Dénestől, 1931-ből származó tételt, amiből következik, hogy a javítóutas algoritmus (tetszőleges kezdeti párosításból indítva) valóban mindig maximális párosítást ad.

7.17. Tétel. *Ha a $G = (A, B; E)$ páros gráf M párosítására nézve nincs javítóút, akkor M maximális párosítás G-ben.*

Bizonyítás: Jelölje M élszámát $|M| = k$. Annak bizonyításához, hogy M maximális, a lefogó ponthalmaz fogalmát (lásd a 7.3. Definíciót) és a 7.4. Állítást hívjuk segítségül: meg fogjuk mutatni, hogy létezik G-ben k pontú lefogó ponthalmaz. Ebből a korábban (a 7.5. Feladatban, illetve utána) már látott gondolatmenet szerint valóban következni fog, hogy M maximális: a k méretű M párosítás létezéséből $\nu(G) \geq k$, a

k pontú lefogó ponthalmaz létezéséből $\tau(G) \leq k$, ezekből és a 7.4. Állításból együtt pedig $k \leq \nu(G) \leq \tau(G) \leq k$ adódik. Következik, hogy $\nu(G) = k$, ami valóban azt jelenti, hogy M maximális párosítás.

Azt állítjuk, hogy a 7.16. Lemma jelöléseit használva $X = A_3 \cup B_2$ lefogó ponthalmaz G-ben. Legyen ugyanis $e = \{a,b\}$ tetszőleges éle G-nek, amire $a \in A$ és $b \in B$. Ha $a \in A_3$, akkor e-nek az A-beli végpontja X-beli, így e-t X lefogja. Ha viszont $a \in A_1 \cup A_2$, akkor a 7.16. Lemmából következik, hogy $b \in B_2$; így X ebben az esetben is lefogja e-t.

Megmutatjuk még, hogy $|X| = k$. Ez abból adódik, hogy X minden M-beli élnek pontosan az egyik végpontját tartalmazza: az A_2 és B_2 között futóknak a B-beli végpontját, az A_3 és B_3 között futóknak pedig az A-beli végpontját. Mivel $|M| = k$ és X minden M-belinek egy végpontját tartalmazza, ezért $|X| = k$ is igaz.

Megmutattuk tehát, hogy X k elemű lefogó ponthalmaz, amivel a bizonyítás teljes. □

7.2.2. A javítóutas algoritmus elméleti következményei

Korábban (a 7.4. Állítás után) láttuk, hogy léteznek olyan G gráfok, amikre $\nu(G) < \tau(G)$; sőt, a különbség a két oldal között tetszőlegesen nagy is lehet. Ehhez képest fontos tény, hogy páros gráfok esetén ez nem fordulhat elő.

7.18. Következmény. (Kőnig tétele)
Minden G páros gráfra $\nu(G) = \tau(G)$ teljesül.

Bizonyítás: A 7.17. Tétel bizonyításában láttuk, hogy minden G páros gráf esetén egy alkalmas k pozitív egészre (a javítóutas algoritmus által talált párosítás élszámára) $k \leq \nu(G) \leq \tau(G) \leq k$ teljesül. Ebből (a fenti bizonyításban kihasznált $\nu(G) = k$ mellett) $\nu(G) = \tau(G)$ valóban következik. □

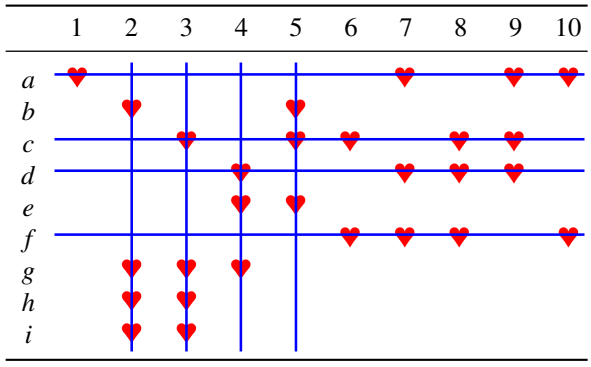
A 7.17. Tétel bizonyításában szereplő $k \leq \nu(G) \leq \tau(G) \leq k$ összefüggésből a már említett $\nu(G) = k$ és $\nu(G) = \tau(G)$ következmények mellett $\tau(G) = k$ is adódik, aminek szintén fontos gyakorlati alkalmazásai vannak: ebből ugyanis kiderül, hogy a 7.17. Tétel bizonyításában látott $X = A_3 \cup B_2$ lefogó ponthalmaz minimális. Vannak olyan, a gyakorlati élet által felvetett problémák, amik épp egy minimális lefogó ponthalmaz meghatározását teszik szükségessé egy G gráfban. A 7.1. szakasz végén említettük, hogy ez általában algoritmikusan nagyon nehéz feladat, nem ismert és nem is várható rá polinomiális algoritmus. A fentiekből viszont következik, hogy a javítóutas algoritmus minimális kiegészítésével (pontosabban: annak a leállása után az $X = A_3 \cup B_2$ halmaz meghatározásával) ez a feladat páros gráfokra hatékonyan megoldhatóvá válik. Sőt, a 7.1. szakaszban írtakból az is következik, hogy a javítóutas algoritmussal ezek mellett még maximális független ponthalmaz és minimális lefogó élhalmaz is hatékonyan kereshető páros gráfokban.

7.19. Feladat. Határozzuk meg a 7.14. Feladat b) részének páros gráfjában $\tau(G)$, $\alpha(G)$ és $\rho(G)$ értékét, valamint adjunk meg egy minimális lefogó ponthalmazt, egy maximális független ponthalmazt és egy minimális lefogó élhalmazt.

Megoldás: A szóban forgó $G = (A, B; E)$ páros gráfot tehát a törzsfőnök táblázata adja meg, de az $\{e, 1\}$ él már hiányzik belőle.

A 7.14. Feladat megoldásának a végén a javítóutas algoritmus G-ben egy nyolc élű M párosítással állt meg. Később, a 7.17. Tételből megtudtuk, hogy ez maximális párosítás, így $\nu(G) = 8$. Kőnig tételéből (a 7.18. Következményből) tehát következik, hogy $\tau(G) = 8$, hiszen G páros gráf. Mivel a G csúcsainak száma $n = 19$, ezért ezekből és Gallai tételének (7.12. Tétel) két állításából következik, hogy $\alpha(G) = n - \tau(G) = 11$ és $\rho(G) = n - \nu(G) = 11$.

A 7.16. Lemma kimondása után a 7.14. Feladat megoldásának végén látható ábrából már kiolvastuk, hogy (a lemma jelöléseit használva) $A_1 = \{e\}$, $B_1 = \{1, 10\}$, $A_2 = \{b, g, h, i\}$ és $B_2 = \{2, 3, 4, 5\}$. Így a maradék (alternáló úton el nem érhető) A-beli csúcsok halmaza $A_3 = \{a, c, d, f\}$, ezeknek a párjaiból álló halmaz pedig $B_3 = \{6, 7, 8, 9\}$. A 7.17. Tétel bizonyításában megmutattuk, hogy $X = A_3 \cup B_2$ lefogó ponthalmaz; ebben a konkrét esetben tehát az $X = \{a, c, d, f, 2, 3, 4, 5\}$ lefogó ponthalmazt kapjuk, amire $|X| = \tau(G) = 8$. Erről könnyen ellenőrizhető, hogy valóban lefogó: ha a törzsfőnök táblázatában (amiből az $\{e, 1\}$ élnek megfelelő szívet már töröltük) az ezeknek a csúcsoknak megfelelő sorokat, illetve oszlopokat áthúzzuk, ahogyan az a 7.15. ábrán látható, akkor a gráf éleinek megfelelő minden szív legalább az egyik irányból át van húzva; ez éppen azt jelenti, hogy minden élnek legalább egy végpontja X-beli.



7.15. ábra

A kapott maximális párosítás és minimális lefogó ponthalmaz birtokában pedig a 7.1. szakaszban írtak szerint készíthető maximális független ponthalmaz, illetve minimális lefogó élhalmaz. Az előbbi esetben tehát (a 7.10. Állításnak megfelelően) az X-ben nem szereplő csúcsok halmaza, $Y = \{b, e, g, h, i, 1, 6, 7, 8, 9, 10\}$ független ponthalmaz, amire $|Y| = \alpha(G) = 11$. Az utóbbi esetben pedig (a 7.11. Lemma *(i)* állításának bizonyítása, illetve a 7.12. Tétel után írtak szerint) az M által le nem

fogott csúcsok halmaza $\{e,1,10\}$; így ezekből egy-egy tetszőleges élt M-hez véve minimális lefogó élhalmazt kapunk. Így például a $Z = M \cup \big\{\{e,4\},\{a,1\},\{a,10\}\big\}$ élhalmaz lefogó, amelyre $|Z| = \rho(G) = 11$. □

A fenti feladatban vizsgált páros gráfban tehát nem csak $\nu(G)$ és $\tau(G)$ értéke egyezett meg (amit Kőnig tételéből, vagyis a 7.18. Következményből már általában is tudunk), hanem $\alpha(G)$ és $\rho(G)$ értéke is. Figyelmet érdemel, hogy ez sem véletlen: páros gráfokban a 7.8. Állításban látott $\alpha(G) \leq \rho(G)$ egyenlőtlenség is egyenlőséggel teljesül.

7.20. Következmény. *Ha a G páros gráf nem tartalmaz izolált pontot, akkor rá $\alpha(G) = \rho(G)$ teljesül.*

Bizonyítás: Kőnig tételéből (7.18. Következmény) tudjuk, hogy $\nu(G) = \tau(G)$, hiszen G páros gráf. Gallai 7.12. Tételéből pedig azt tudjuk, hogy $\alpha(G) + \tau(G) = n$ és $\nu(G) + \rho(G) = n$ igazak, ahol n a G csúcsainak a száma. Ezekből együtt tehát $\alpha(G) = n - \tau(G) = n - \nu(G) = \rho(G)$ valóban következik. □

A szakasz elején, a hozzá befutó megrendeléseket teljesíteni igyekvő cég példája kapcsán már láttuk, hogy gyakorlati alkalmazásokban felvetődő probléma lehet, hogy egy $G = (A,B;E)$ páros gráfban létezik-e A-t lefedő párosítás. Erre a kérdésre persze bizonyos értelemben már ismerjük a választ: mivel $|A|$-nál nagyobb párosítás nyilván nem lehet G-ben, ezért A-t lefedő párosítás pontosan akkor létezik, ha $\nu(G) = |A|$; ez pedig a javítóutas algoritmus futtatásával eldönthető. Az alább következő tétel mégis nagyon fontos – elsősorban olyan helyzetekben, amikor nem egy konkrét, adott G gráfra kell eldöntenünk ezt a kérdést, hanem csak bizonyos információk állnak rendelkezésünkre G-ről.

Idézzük fel ismét a 7.14. Feladat b) részét: a 7.16. Lemma kimondása után már láttuk, hogy az $A_1 \cup A_2 = \{b,e,g,h,i\}$ halmaz csúcsainak minden szomszédja a $B_2 = \{2,3,4,5\}$ csúcsok közül kerül ki. Ez a megfigyelés pedig már önmagában is mutatja, hogy ebben a gráfban nincs A-t lefedő párosítás: ez nyilvánvaló abból, hogy az $A_1 \cup A_2$ halmazt alkotó öt fiú mindegyike a B_2-t alkotó négy lány között keresi a párját, így legalább az egyiküknek nyilván nem juthat. Az alábbi, Philip Hall brit matematikustól, 1935-ből származó tétel ezt a megfigyelést általánosítja.

7.21. Definíció. *Legyen $G = (A,B;E)$ páros gráf és $X \subseteq A$ egy tetszőleges részhalmaza A-nak. Ekkor az X* szomszédságának *nevezzük és $N(X)$-szel jelöljük a B-nek azt a részhalmazát, ami azokból a B-beli csúcsokból áll, amiknek van (legalább egy) szomszédja X-ben. Képletben:*

$$N(X) = \big\{b \in B : \exists a \in X, \{a,b\} \in E(G)\big\}.$$

Így például a 7.14. Feladat b) részének gráfjában az $X = \{b,e,g,h,i\}$ halmazra $N(X) = \{2,3,4,5\}$.

7.22. Tétel. (Hall tétele)
A $G = (A, B; E)$ páros gráfban akkor és csak akkor létezik A-t lefedő párosítás, ha minden $X \subseteq A$ részhalmazra $|N(X)| \geq |X|$ teljesül.

Bizonyítás: A feltétel szükségessége nyilvánvaló: ha létezik A-t lefedő párosítás, akkor $|N(X)| \geq |X|$ minden $X \subseteq A$ esetén. Valóban, ha M egy A-t lefedő párosítás, akkor az X elemeinek M szerinti szomszédai mind $N(X)$-ben vannak és páronként különbözők, így $N(X)$-nek legalább $|X|$ eleme van. (Vagy ugyanez más megfogalmazásban: ha egy $X \subseteq A$ részhalmazra $|N(X)| < |X|$ teljesülne, akkor – ahogyan azt a tétel kimondása előtti példában is láttuk – nem juthatna már az X minden elemének sem csupa különböző pár, így nem létezhetne A-t lefedő párosítás.)

Most belátjuk a feltétel elégségességét: ha $|N(X)| \geq |X|$ minden $X \subseteq A$ részhalmazra teljesül, akkor van A-t lefedő párosítás G-ben. Futtassuk ugyanis a javítóutas algoritmust a G gráfra (tetszőleges párosításból indítva). Megmutatjuk, hogy az algoritmus leállásakor kapott M párosítás lefedi A-t. Tegyük fel ugyanis indirekt, hogy nem ez a helyzet. Ekkor a 7.16. Lemma jelöléseit használva $N(A_1 \cup A_2) = B_2$ következik; valóban, a lemma állítása szerint $A_1 \cup A_2$ és $B_1 \cup B_3$ között nincs éle G-nek, ezért az $(A_1 \cup A_2)$-beliek minden szomszédja B_2-beli. Továbbá mivel $|A_2| = |B_2|$ (hiszen A_2 és B_2 elemeit az M megfelelő élei párba állítják) és $A_1 \neq \emptyset$ (mert az M nem fedi le A-t), ezért $|A_1 \cup A_2| > |B_2|$. Következik, hogy az $X = A_1 \cup A_2$ esetében sérül az $|N(X)| \geq |X|$ feltétel; ez az ellentmondás pedig bizonyítja a tétel állítását. □

7.23. Feladat. Egy kiránduláson n házaspár vesz részt. El kellene osztani közöttük $2n$ különböző fajta csokit úgy, hogy mindenki egyet kapjon. Tudjuk, hogy mindenki legalább n fajtát szeret a csokik közül. Továbbá minden emberre teljesül, hogy ha valamelyik fajta csokit nem szereti, akkor a házastársa ezt a fajtát biztosan szeretni fogja. Bizonyítsuk be, hogy a csokik szétoszthatók úgy, hogy mindenki olyat kapjon, amit szeret.

Megoldás: A $G = (A, B; E)$ páros gráf A pontosztályát alkossa a $2n$ résztvevő, a B-t pedig a $2n$ csoki; egy embert akkor kössünk össze egy csokival, ha az illető ezt a fajta csokit szereti. Azt kell megmutatnunk, hogy G-ben létezik A-t lefedő párosítás.

Ezért Hall tételét hívjuk segítségül. Legyen $X \subseteq A$ egy tetszőleges részhalmaza A-nak. Ha van olyan házaspár, aminek X mind a két tagját tartalmazza, akkor a feladat szövege szerint $N(X) = B$ (hiszen minden csoki szomszédos G-ben ennek a házaspárnak legalább az egyik tagjával). Így az ilyen X-ekre $2n = |N(X)| \geq |X|$ nyilván igaz. Ha viszont X minden házaspárnak legfőljebb csak az egyik tagját tartalmazza, akkor $|X| \leq n$, hiszen csak n házaspár van. De ha X legalább egy embert tartalmaz, akkor $|N(X)| \geq n$ következik abból, hogy minden ember legalább n csokit szeret. Így ilyenkor $|N(X)| \geq n \geq |X|$. Ha viszont $X = \emptyset$, akkor persze $N(X) = \emptyset$, így $0 = |N(X)| \geq |X| = 0$ ebben az esetben is igaz.

Megmutattuk tehát, hogy $|N(X)| \geq |X|$ az A minden X részhalmazára teljesül, így Hall tétele szerint G-ben valóban létezik A-t lefedő párosítás. □

Hall tételéből könnyen választ kaphatunk arra a kérdésre is, hogy teljes párosítás létezése mitől függ egy páros gráfban. Bár az erre vonatkozó tétel alig különbözik a fentitől és bármelyikből könnyen bebizonyítható a másik, mégis, ezt az állítást az alábbi formájában Georg Frobenius (1849 – 1917) német matematikusnak szokták tulajdonítani.

7.24. Következmény. (Frobenius tétele)
A $G = (A,B;E)$ páros gráfban akkor és csak akkor létezik teljes párosítás, ha $|A| = |B|$ és minden $X \subseteq A$ részhalmazra $|N(X)| \geq |X|$ teljesül.

Bizonyítás: A feltétel szükségessége ismét nyilvánvaló: ha létezik G-ben teljes párosítás, akkor $|A| = |B|$ természetesen igaz és mivel egy teljes párosítás egyben A-t lefedő párosítás is, így azt már beláttuk, hogy $|N(X)| \geq |X|$ minden $X \subseteq A$ részhalmazra teljesül.

Megfordítva, ha $|N(X)| \geq |X|$ minden $X \subseteq A$ részhalmazra teljesül, akkor Hall 7.22. Tétele miatt G-ben van A-t lefedő párosítás. De mivel $|A| = |B|$, ezért egy ilyen párosításnak nyilván B-t is fednie kell, így az teljes párosítás. Ezzel tehát a feltétel elégségességét is beláttuk. □

A szakasz végén még bemutatunk egy további olyan következményt is, ami sok alkalmazásban előkerül (és később a 8.6. Tételben is használni fogjuk). Egy G gráfot *d-regulárisnak* mondunk, ha G-ben minden pont foka d.

7.25. Következmény. *Ha a $G = (A,B;E)$ páros gráf d-reguláris, ahol $d \geq 1$ tetszőleges egész, akkor G-ben van teljes párosítás.*

Bizonyítás: Mivel G minden élének pontosan az egyik végpontja A-beli és minden A-beli csúcsra d él illeszkedik, ezért G élszámára $|E| = d \cdot |A|$ adódik. Hasonlóan kapjuk, hogy $|E| = d \cdot |B|$. Ezekből tehát $d \cdot |A| = d \cdot |B|$ és így $|A| = |B|$ adódik.

Legyen $X \subseteq A$ tetszőleges és jelölje E_X az X csúcsaira illeszkedő élek halmazát; ezeknek a B-beli végpontja tehát $N(X)$-beli. Hasonlóan a fentiekhez, mivel minden X-beli csúcsra pontosan d darab E_X-beli él illeszkedik, ezért $|E_X| = d \cdot |X|$. $N(X)$ csúcsaira nem csak E_X-beli élek illeszkedhetnek, de annyi ezekről is elmondható, hogy mindegyikre legföljebb d darab E_X-beli él illeszkedik; így $|E_X| \leq d \cdot |N(X)|$. Ezek összevetéséből $d \cdot |X| \leq d \cdot |N(X)|$ és így $|X| \leq |N(X)|$ adódik.

Megmutattuk tehát, hogy a 7.24. Következmény feltételei teljesülnek, így G-ben valóban létezik teljes párosítás. □

7.3. Teljes párosítás tetszőleges gráfban

A 7.1. szakasz végén már említettük, hogy maximális párosítás keresésére tetszőleges gráfban is ismert hatékony, polinomiális algoritmus, ez azonban jóval bonyolultabb a 7.2. szakaszban bemutatott javítóutas algoritmusnál. (Bár a javítóút 7.13. Definíciója könnyen általánosítható tetszőleges gráfokra és még a 7.17. Tétel megfelelője is igaz marad, a valódi problémát a javítóút keresése, illetve a létezésének

az eldöntése okozza: amit páros gráfokra a 7.2. szakaszban a BFS algoritmus egyszerű módosításával meg tudtunk oldani, az tetszőleges gráfok esetében már sokkal komplikáltabb feladat.)

Ebben a szakaszban csak a (nem feltétlen páros) gráfok párosításairól szóló legalapvetőbb tételt fogjuk bemutatni, ami Frobenius tételét (7.24. Következmény) általánosítja: arra a kérdésre válaszol, hogy mitől függ egy tetszőleges gráfban teljes párosítás létezése. Ez a tétel jól illusztrálja azt is, hogy a tetszőleges (vagyis nem feltétlen páros) gráfok párosításaival kapcsolatos problémák jóval komplikáltabbak, mint a páros gráfok esetében.

Idézzük fel a 7.2. Feladatot: megmutattuk, hogy az abban szereplő tíz csúcsú gráfban nincs teljes párosítás, mert $\nu(G) = 4$; ennek a kulcsa pedig a négy csúcsú $X = \{b,d,g,i\}$ lefogó ponthalmaz volt. Az X lefogó voltát fogalmazhatjuk úgy is, hogy ennek a négy csúcsnak az elhagyása esetén a maradék gráfnak már egyáltalán nem volna éle; vagyis hat egyetlen csúcsú komponens maradna. Látni fogjuk, hogy ez a jelenség egyben példa az alábbi tétel állítására is; a tétel kulcsgondolata ennél annyival általánosabb, hogy az elhagyott pontok számát nem csak a kapott gráf egyetlen csúcsú komponenseinek a számával hasonlítja össze, hanem az összes páratlan csúcsszámú komponensek számával.

A következő, William T. Tutte (1917 – 2002) brit matematikustól, 1947-ből származó tétel kimondásához két egyszerű jelölést vezetünk be. Ha adott a G gráf csúcsainak $X \subseteq V(G)$ részhalmaza, akkor $G-X$ jelöli azt a gráfot, amit G-ből az X csúcsainak (és az azokra illeszkedő éleknek) a törlésével kapunk; más megfogalmazásban: $G-X$ a $V(G) \setminus X$ csúcshalmaz által feszített részgráfja G-nek. Továbbá tetszőleges H gráf esetén $c_p(H)$ jelöli a H páratlan sok csúcsot tartalmazó komponenseinek a számát. (Például ha H összefüggő, akkor $c_p(H)$ értéke 0 vagy 1 attól függően, hogy H csúcsainak a száma páros vagy páratlan.)

7.26. Tétel. (Tutte tétele)
A G gráfban akkor és csak akkor létezik teljes párosítás, ha a csúcsok minden $X \subseteq V(G)$ részhalmazára $c_p(G-X) \leq |X|$ teljesül.

Más szóval: pontosan akkor létezik teljes párosítás G-ben, ha G-ből bárhogyan k darab csúcsot elhagyva a kapott gráf páratlan sok csúcsú komponenseinek a száma legföljebb k. Érdemes külön felhívni a figyelmet arra, hogy ennek a feltételnek $k = 0$ (vagyis a tétel fenti alakja szerint: $X = \emptyset$) esetén is teljesülnie kell; ez pedig éppen annyit jelent, hogy G-nek minden komponense páros sok csúcsú kell legyen (ami nyilván szükséges a teljes párosítás létezéséhez, de önmagában nem elégséges).

Visszatérve a 7.2. Feladat példájára: az ott látott G gráf és $X = \{b,d,g,i\}$ esetén $|X| = 4$ és $c_p(G-X) = 6$, mert X elhagyása után hat darab egy csúcsú komponens keletkezik; így a 7.26. Tétel is mutatja, hogy G-ben nincs teljes párosítás.

Tutte tételének a bizonyítása már kicsit nehezebb, itt csak a feltétel szükségességét igazoljuk.

A szükségesség bizonyítása a 7.26. Tételben: Azt kell tehát megmutatni, hogy ha G-ben van teljes párosítás, akkor minden $X \subseteq V(G)$ esetén $c_p(G-X) \leq |X|$ teljesül. Rögzítsünk le G-ben egy M teljes párosítást és válasszunk egy tetszőleges $X \subseteq V(G)$ részhalmazt. Tegyük fel, hogy X elhagyása után a kapott gráfnak ℓ darab páratlan sok csúcsú komponense van; jelölje ezeket $C_1, C_2 \ldots, C_\ell$.

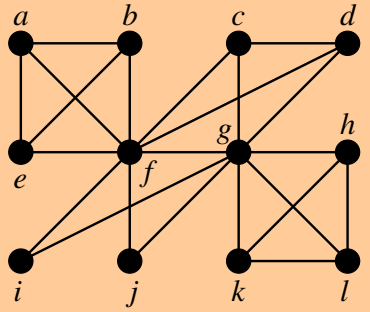
Válasszunk most ki egy tetszőleges C_i páratlan komponenst G-ben. Ekkor C_i minden csúcsára is illeszkedik M-beli él, hiszen M teljes párosítás. Mivel azonban

C_i-nek páratlan sok csúcsa van, ezért legalább egy C_i-beli csúcsnak az M szerinti párja C_i-n kívül kell legyen. Válasszunk ki minden C_i-hez egy-egy ilyen csúcsot (ami tehát egy C_i-beli csúcs C_i-n kívüli párja) és jelölje ezt v_i.

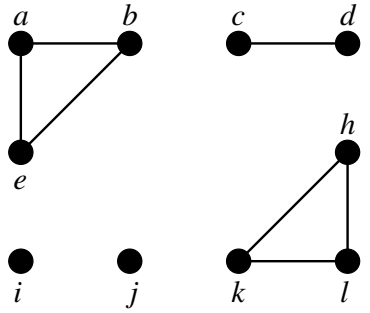
Ekkor $v_i \in X$ minden $i \in \{1,\dots,\ell\}$ esetén; valóban, mivel az X elhagyása után C_i külön komponenst alkot, ezért a C_i-beli csúcsok minden G-beli szomszédja csak C_i-ben vagy X-ben lehet (és v_i-ről tudjuk, hogy nem C_i-beli). Másrészt a v_i csúcsok nyilván mind különbözők, hiszen M egy párosítás.

Ebből tehát következik, hogy X-nek legalább $c_p(G-X) = \ell$ csúcsa van, amivel tehát a 7.26. Tétel feltételének a szükségességét beláttuk. □

7.27. Feladat. Létezik-e teljes párosítás az alábbi G gráfban?



Megoldás: Legyen $X = \{f,g\}$. Ekkor az X elhagyásával kapott $G-X$ gráf a 7.16. ábrán látható.



7.16. ábra

Látszik, hogy $(G-X)$-nek öt komponense van, amik közül négy (az a, b és e, illetve a h, k és l csúcsokból álló háromszögek, valamint az i és j izolált csúcsok) páratlan sok csúcsúak, egy pedig (a c és d csúcsokból, illetve a köztük futó élből álló) páros. Ezért $c_P(G-X) = 4$. Mivel tehát $4 = c_p(G-X) > |X| = 2$, ezért Tutte tétele szerint G-ben nincs teljes párosítás. □

8. fejezet

Gráfok élszínezése

Egy iskola órarendtervezési feladata (egyszerűsített formában) a következő lehet. Adottak az osztályok (vagy tanulói csoportok) és a tanárok, valamint ismert minden osztály és tanár esetén, hogy az adott tanár ennek az osztálynak tart-e órát és ha igen, akkor egy héten hányat. A feladatunk az, hogy beosszuk az iskola egy héten megtartandó összes óráját egy órás idősávokba úgy, hogy minden osztálynak és tanárnak egy idősávban legföljebb egy órája lehessen. (Ebben az egyszerűsített modellben tehát feltételeztük, hogy az idősávok felcserélhetők – ami egy valós helyzetben nyilván csak korlátozottan volna igaz.) A helyzetet egy $G = (A, B; E)$ páros gráffal ábrázolhatjuk, ahol A az osztályokból, B pedig a tanárokból áll és minden $a \in A$ osztály és $b \in B$ tanár esetén annyi párhuzamos él fut G-ben a és b között, ahány órája az a osztálynak a b tanárral egy héten van (ami lehet nulla is). Ekkor az iskola megtartandó óráit a G élei reprezentálják, így $E(G)$-t kell felosztani az idősávoknak megfelelő részhalmazokra. Ezt szemléletesebbé tehetjük azzal az átfogalmazással, hogy G éleit megszínezzük: például a piros színű élek felelnek meg a hétfőn 8-kor kezdődő idősávnak, a kék élek a hétfőn 9-kor kezdődőnek, stb. Az a követelmény pedig, hogy minden osztálynak és tanárnak egy idősávban csak egy órája lehessen annak felel meg, hogy minden $v \in A \cup B$ csúcs esetén a v-re illeszkedő élek között ne lehessen azonos színű. Az ezt kielégítő élszínezések között is nyilván olyanra van szükség, ami a lehető legkevesebb színt használja: ez felel meg annak, hogy a legkevesebb idősávot használva szeretnénk elkészíteni az iskola órarendjét.

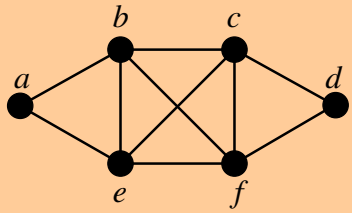
A fenti példa csak egy a számtalan olyan alkalmazás közül, ami gráfok éleinek a színezését kívánja meg az alábbi definíció szerint.

8.1. Definíció. *Legyen G egy gráf és $k \geq 1$ egész szám. A G gráf k* színnel élszínezhető*, ha G minden éle kiszínezhető k adott (tetszőleges) színnel úgy, hogy G bármely két szomszédos (vagyis közös csúcsra illeszkedő) élének a színe különböző. A G* élkromatikus száma *k, ha G k színnel élszínezhető, de $(k-1)$-gyel már nem. G élkromatikus számának a jele: $\chi_e(G)$.*

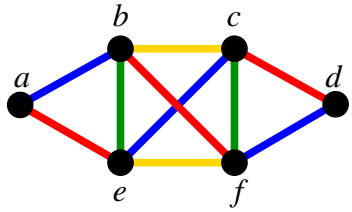
Látható, hogy ez a definíció analóg a 6.1. Definícióval: a különbség csak annyi,

hogy csúcsok helyett most az éleknek kell színt adni.

8.2. Feladat. Határozzuk meg az alábbi G gráf élkromatikus számát, $\chi_e(G)$-t.



Megoldás: G élei megszínezhetők négy színnel, például az alábbi ábra szerint.



Annak indoklásához, hogy ennél kevesebb szín nem elég, tekintsük például a b csúcsra illeszkedő négy élt: ezek közül bármely kettő szomszédos (hiszen a b-re illeszkednek), így közülük semelyik kettő nem színezhető azonos színűre. Így már ehhez a négy élhez is feltétlen szükség van négy különböző színre.

Megmutattuk tehát, hogy G élei megszínezhetők négy színnel, de hárommal nem, így $\chi_e(G) = 4$. □

A fenti megoldásban megfigyelhetjük, hogy mind a négy felhasznált szín esetén az ilyen színűre festett élek független élhalmazt alkotnak G-ben; például az $M = \{\{a,e\},\{b,f\},\{c,d\}\}$ párosítás a piros élek halmaza. Ez nyilván általában is igaz, hiszen az élszínezés 8.1. Definíciója éppen azt követeli meg, hogy azonos színű élek ne illeszkedjenek közös csúcsra – más szóval: ezek független élhalmazt alkotnak. Így egy gráf élszínezését felfoghatjuk úgy is, hogy a gráf élhalmazát párosításokra bontjuk fel. Ez a megfigyelésünk analóg azzal, amit a 7.6. Definíció után, a független ponthalmazokkal kapcsolatban írtunk: a gráfok csúcsainak színezésekor az azonos színnel színezett ponthalmazok függetlenek.

A gráfok csúcsainak színezésekor nagyon hasznosnak bizonyult a klikkszám fogalma (lásd a 6.3. szakaszt): ha található G-ben k csúcs úgy, hogy közülük bármely kettő szomszédos, akkor már ezekre is k különböző színt kell használni, így $\chi(G) \geq k$. Az ezzel analóg, élszínezésekre vonatkozó gondolatot már a 8.2. Feladat megoldásában is használtuk: mivel a b-re illeszkedő négy él közül bármely kettő szomszédos volt, ezért $\chi_e(G) \geq 4$. A csúcsok színezése kapcsán használt klikkszám fogalmának tehát élszínezések esetén a maximális fokszám felel meg: $\Delta(G)$-vel jelöljük egy G gráf csúcsainak fokszámai közül a legnagyobbat.

8.3. Állítás. *Minden (hurokélmentes) G gráfra $\Delta(G) \leq \chi_e(G)$ teljesül.*

Bizonyítás: Legyen v egy $\Delta(G)$ fokú csúcs G-ben. Ekkor a v-re illeszkedő $\Delta(G)$ darab élnek G tetszőleges élszínezésében csupa különböző színt kell kapnia. Így G minden élszínezése legalább $\Delta(G)$ színt használ, amiből $\Delta(G) \leq \chi_e(G)$ valóban következik. □

A fenti állításban a hurokélmentességet csak azért kellett kikötni, mert a csúcsok fokát korábban úgy definiáltuk, hogy a hurokélek a fokszámot kettővel növelik; élszínezés szempontjából viszont a hurokélek is csak egy új színt tennének szükségessé. A 6. szakasz elején elmondtuk, hogy a gráfok csúcsainak színezése kapcsán csak egyszerű gráfokkal érdemes foglalkozni. Élszínezések esetében azonban nem ez a helyzet: a szakaszt bevezető órarendtervezési feladat példája is mutatja, hogy gyakorlati alkalmazásokban is felmerül párhuzamos éleket tartalmazó gráfok élszínezése; alább látni fogjuk, hogy ebben a témában ez egyáltalán nem részletkérdés. Hurokélek azonban még az órarendtervezési feladat gráfjában sem voltak és könnyű végiggondolni, hogy ezeknek általában sincs jelentősége: egy hurokéleket is tartalmazó gráf élszínezési feladatát lényegében megoldjuk, ha az összes hurokél elhagyásával kapott gráf éleit színezzük meg a lehető legkevesebb színnel.

Fentebb már említettük, hogy a 8.3. Állítás az élszínezésekre vonatkozó megfelelője az $\omega(G) \leq \chi(G)$ összefüggésnek (6.11. Állítás). Az utóbbi esetében is felmerült a kérdés: lehet-e tetszőlegesen nagy a különbség a két oldal között? A válasz az $\omega(G) \leq \chi(G)$ egyenlőtlenség esetén igenlő volt: ez következett a 6.14. Tételből. A $\Delta(G) \leq \chi_e(G)$ összefüggés esetében már izgalmasabb a válasz — legalábbis egyszerű gráfok esetében. Ez következik az alábbi, Vagyim G. Vizing (1937 – 2017) szovjet matematikustól, 1964-ből származó, itt bizonyítás nélkül közölt tételből.

8.4. Tétel. (Vizing tétele)
Minden G egyszerű gráfra $\chi_e(G) \leq \Delta(G)+1$ teljesül.

Vizing tételéből és a 8.3. Állításból együtt tehát következik, hogy ha G egyszerű gráf, akkor az élkromatikus száma $\Delta(G)$ vagy $\Delta(G)+1$. Vizing tételének a bizonyításából ráadásul egy hatékony, polinomiális algoritmus is kiolvasható egy G egyszerű gráf $\Delta(G)+1$ színnel való élszínezésére. Az érdekesség kedvéért azonban megemlítjük, hogy annak az eldöntése, hogy egy G egyszerű gráf $\Delta(G)$ színnel is élszínezhető-e már nagyon nehéz feladat: ez is azok közé a problémák közé tartozik, amelyekre nem ismert és valószínűleg nem is létezik polinomiális algoritmus (noha ez nem bizonyított tény).

Fontos azonban hangsúlyozni, hogy Vizing tétele csak egyszerű gráfokra vonatkozik; egy nem egyszerű gráf élkromatikus száma lehet nagyobb, mint $\Delta(G)+1$. Legyen például G az a gráf, amit egy háromszögből nyerünk úgy, hogy annak mindhárom élét helyettesítjük k párhuzamos éllel, ahol $k \geq 2$ tetszőleges egész. Ekkor $\Delta(G) = 2k$, hiszen mindhárom csúcsra $2k$ él illeszkedik. Másrészt $\chi_e(G) = 3k$, mert G-nek összesen $3k$ éle van és ezek közül bármely kettő szomszédos (mert valamelyik csúcsban találkoznak), így minden élt csupa különböző színűre kell festeni. Ez

a példa egyben azt is mutatja, hogy a $\Delta(G) \leq \chi_e(G)$ egyenlőtlenség két oldala között bármekkora lehet a különbség, ha nem kötjük ki, hogy G egyszerű gráf.

A 6.14. Tételből valójában több is kiderül, mint hogy az $\omega(G) \leq \chi(G)$ egyenlőtlenség két oldala között tetszőlegesen nagy lehet a különbség: azt is megtudjuk belőle, hogy $\chi(G)$ az $\omega(G)$-nek semmilyen függvényével nem becsülhető felülről. Vagyis nem létezik olyan $f : \mathbb{N} \to \mathbb{N}$ függvény, amire $\chi(G) \leq f(\omega(G))$ mindig igaz volna. Valóban, ez következik abból, hogy $\chi(G)$ olyan gráfokra is tetszőlegesen nagy lehet, amikre $\omega(G) = 2$.

Ha ugyanezt a kérdést a $\Delta(G) \leq \chi_e(G)$ egyenlőtlenségre vizsgáljuk, akkor a válasz más lesz – még nem feltétlen egyszerű gráfok esetében is. Ha a 6.2. szakaszban látott mohó eljárást élszínezésekre adaptáljuk (ami könnyen megtehető), akkor a 6.7. Állítás bizonyításával analóg módon könnyen belátható, hogy $\chi_e(G) \leq 2 \cdot \Delta(G) - 1$. Ennél erősebb felső korlát következik Claude Shannon (1916 – 2001) amerikai matematikus 1949-ben bizonyított tételéből: $\chi_e(G) \leq \lfloor \frac{3}{2}\Delta(G) \rfloor$ igaz minden G gráfra (tehát nem csak az egyszerű gráfokra). Fentebb láttuk, hogy arra a G gráfra, amit egy háromszögből nyertünk minden élének k párhuzamos éllel való helyettesítésével $\Delta(G) = 2k$ és $\chi_e(G) = 3k$ teljesült; ez a példa tehát mutatja, hogy Shannon tételében a $\frac{3}{2}$-es szorzó már nem javítható.

8.5. Feladat. Határozzuk meg a tizenegy csúcsú teljes gráf élkromatikus számát, $\chi_e(K_{11})$-et.

Megoldás: K_{11} minden csúcsának a foka tíz, így $\Delta(K_{11}) = 10$. Meg fogjuk mutatni, hogy K_{11} élei nem színezhetők meg tíz színnel, vagyis $\chi_e(K_{11}) > 10$. Mivel Vizing 8.4. Tételéből $\chi_e(K_{11}) \leq 11$ következik, ezért ezekből $\chi_e(K_{11}) = 11$ adódni fog.

Tegyük fel ezért indirekt, hogy K_{11} éleit megszíneztük tíz színnel, legyen ezek közül egy a piros. Mivel minden csúcs foka tíz és a színek száma is tíz, ezért minden csúcsra kell illeszkedjen piros színű él (és persze ugyanez a többi színről is elmondható volna). Mivel egynél több piros él persze egy csúcsra sem illeszkedhet, ezért ebből következik, hogy a piros élek teljes párosítást alkotnak K_{11}-ben. Ez azonban nyilván lehetetlen: páratlan csúcsú gráfban nem létezhet teljes párosítás.

Így tehát K_{11} élei nem színezhetők meg tíz színnel, ezért $\chi_e(K_{11}) = 11$. □

A fenti megoldás tizenegy helyett természetesen tetszőleges páratlan számra is működne. A megoldás érdekessége, hogy úgy határoztuk meg K_{11} élkromatikus számát, hogy közben nem színeztük meg K_{11} éleit; ehelyett Vizing tételére hivatkoztunk. Persze meg lehetne adni K_{11} egy helyes élszínezését tizenegy színnel, de ez egyáltalán nem volna magától értetődő (bár nem is különösebben nehéz).

Többször találkoztunk már azzal a jelenséggel, hogy egy gráfelméleti probléma tetszőleges gráfokra nagyon nehéz, de bizonyos gyakorlati alkalmazásokban megjelenő speciális gráfokra hatékonyan megoldhatóvá válik. Így például polinomiális algoritmussal kiszámítható egy adott intervallumrendszer által reprezentált G intervallumgráfra $\chi(G)$ értéke (lásd a 6.17. Tételt) vagy $\tau(G)$ és $\alpha(G)$ értéke páros gráfokra (lásd a 7.2. szakaszt), miközben ezekre a feladatokra tetszőleges gráfok esetén nem ismert és nem is várható polinomiális algoritmus. Az alábbi, szintén Kőnig Dénestől, 1916-ból származó tétel szerint az élszínezési feladat is könnyebb

páros gráfokra; márpedig a szakasz elején látott órarendtervezési feladatból kiderült, hogy ez a kérdés alkalmazásokban is felvetődő, fontos probléma. Érdemes külön kiemelni, hogy az alábbi tétel nem csak egyszerű gráfokról szól, hanem tetszőleges, akár párhuzamos éleket is tartalmazó páros gráfokról is.

8.6. Tétel. (Kőnig élszínezési tétele)
Minden $G = (A, B; E)$ páros gráfra $\chi_e(G) = \Delta(G)$ teljesül.

Bizonyítás: Legyen $d = \Delta(G)$. A tételt először abban az esetben bizonyítjuk be, ha G-ben minden pont foka d, vagyis G d-reguláris. A $d = 0$ esetben az állítás igaz (bár érdektelen): ilyenkor G-nek nincs éle, így $\chi_e(G) = 0$. Ha $d \geq 1$, akkor a 7.25. Következmény szerint G-ben létezik egy M_1 teljes párosítás. Színezzük ki M_1 éleit 1-es színűre (mondjuk pirosra), majd hagyjuk el őket G-ből. A kapott G_1 gráf nyilván $(d-1)$-reguláris, hiszen minden csúcsára pontosan egy M_1-beli él illeszkedett, amit elhagytunk. Így ha $d - 1 \geq 1$, akkor G_1-re ismét alkalmazható a 7.25. Következmény: G_1-ben is létezik egy M_2 teljes párosítás. Színezzük M_2 éleit 2-es színűre (mondjuk kékre) és ezeket is hagyjuk el G_1-ből. A kapott G_2 gráf most $(d-2)$-reguláris, stb. Ennek az eljárásnak a folytatásával, a 7.25. Következmény d-szer való egymás után alkalmazásával valóban egy helyes, d színnel való élszínezését kapjuk G-nek: a d-edik teljes párosítás kiszínezése és elhagyása után már minden pont foka nulla, így ezen a ponton már valóban G minden éle színezett. Ráadásul mivel minden teljes párosítás esetén új színt használtunk, ezért minden csúcsra d különböző színű él illeszkedik.

Térjünk most rá arra az esetre, amikor G nem d-reguláris, van kisebb fokú csúcsa is. A bizonyítás kulcsa az lesz, hogy G-t kiegészítjük további élekkel és esetleg csúcsokkal úgy, hogy d-reguláris gráfot kapjunk. Ha $|A| \neq |B|$, akkor A és B közül a kisebbikhez vegyünk fel annyi új csúcsot, hogy azonos méretűek legyenek. A kapott gráfban válasszunk egy tetszőleges olyan $a \in A$, $b \in B$ csúcspárt, amikre $d(a) < d$ és $d(b) < d$ teljesül és kössük össze a-t és b-t egy új éllel (akkor is, ha esetleg már szomszédosak voltak); majd ezt folytassuk egészen addig, amíg létezik ilyen csúcspár. Az eljárás végén kapott gráfot jelölje G'.

Megmutatjuk, hogy G' már d-reguláris. Jelölje ehhez G' éleinek a számát m. Mivel a fenti bekezdésben írt, az élek hozzávételéből álló folyamat megállt, ezért A és B közül legalább az egyikben már minden pont foka d; tegyük fel, hogy ez például A-ra igaz. Ekkor tehát $m = d \cdot |A|$, hiszen minden élnek pontosan az egyik végpontja A-beli. A B csúcsai nyilván legfőljebb d fokúak, hiszen ez kezdetben igaz volt ($d = \Delta(G)$ miatt) és új éleket csak d-nél kisebb fokú csúcshoz vettünk hozzá. Ha tehát létezne B-ben d-nél kisebb fokú csúcs, akkor ebből $m < d \cdot |B|$ következne. Azonban $|A| = |B|$ miatt $d \cdot |A| < d \cdot |B|$ lehetetlen, így valóban nem létezhet B-ben sem d-nél kisebb fokú csúcs.

Tehát G' d-reguláris, így a bizonyítás első bekezdésében írtak szerint d színnel élszínezhető. Mivel azonban G minden éle szerepel G'-ben is, ezzel G-nek is megadtuk egy helyes élszínezését $d = \Delta(G)$ színnel. Ebből $\chi_e(G) = \Delta(G)$ a 8.3. Állítás szerint valóban következik. □

Megjegyezzük, hogy a fenti bizonyításban a 7.25. Következményt olyan páros gráfra alkalmaztuk, ami nem feltétlen egyszerű, lehetnek benne párhuzamos élek; ez azonban nem okoz problémát, mert a 7.25. Következmény nem csak egyszerű gráfokra vonatkozik (és ilyen feltevésre annak a bizonyításában sem volt szükség).

Fontos kiemelni, hogy a fenti tétel bizonyításából egy hatékony, polinomiális algoritmus is kiolvasható egy páros gráf $\Delta(G)$ (vagyis optimális számú) színnel való élszínezésére: ehhez először G-t d-reguláris gráffá kell kiegészíteni, majd a 7.2. szakaszban látott javítóutas algoritmussal d-szer egymás után teljes párosítást keresni.

Bemutatunk egy másik bizonyítást is a 8.6. Tételre, ami egyrészt nem épít a párosításokról korábban tanultakra, másrészt még hatékonyabb algoritmus olvasható ki belőle, mint a fenti bizonyításból.

A 8.6. Tétel egy alternatív bizonyítása: Leírunk egy algoritmust, ami G éleit megszínezi $\Delta(G)$ darab színnel; ebből a tétel a 8.3. Állítás szerint következik.

Jelölje G éleit $e_1, e_2, \ldots, e_m$. Az algoritmus egymás után foglalkozik G éleivel és a sorra kerülő élt mindig megszínezi a rendelkezésre álló $\Delta(G)$ szín valamelyikével (mégpedig helyesen, vagyis szomszédos élek különböző színt kapnak). Eközben előfordulhat, hogy a korábban már megszínezett élek egy részét át kell festeni más színűre, de semelyik él nem változik vissza színtelenné.

Tegyük fel tehát, hogy az $e_1, e_2, \ldots, e_{i-1}$ élek már kaptak valamilyen színt és legyen $e_i = \{u, v\}$, ahol $u \in A$ és $v \in B$. Mivel u-ra és v-re is legföljebb $\Delta(G)$ él illeszkedik, de ezek között még van színtelen (nevezetesen e_i), ezért mindkét csúcsnak van szabad színe – vagyis olyan szín, amilyen színű él még nem illeszkedik a csúcsra. Ha u-nak és v-nek van közös szabad színe, akkor persze e_i megkaphatja ezt a színt és készen vagyunk. Tegyük fel, hogy nem ez a helyzet.

Legyen u egy szabad színe a piros, v egy szabad színe a kék. Legyen C a G-nek az a részgráfja, ami (G összes csúcsából és) a jelenlegi színezés szerint pirosra vagy kékre színezett élekből áll. Ha létezne C-ben u és v között egy P út, akkor P éleinek színe nyilván váltakozva piros és kék volna, különben egy csúcsra két azonos színű él illeszkedne; továbbá P-nek az u-ra illeszkedő éle kék, a v-re illeszkedő éle pedig piros kellene legyen, mert u-nak a piros, v-nek a kék szabad színe. Ebből következően egy ilyen P út csak páros sok élű lehetne. Ez viszont lehetetlen, hiszen $u \in A$ és $v \in B$ miatt u és v között csak páratlan hosszú út létezhet.

Ebből tehát az következik, hogy u és v között nincs út C-ben, vagyis u és v C-nek két különböző komponensébe esik. Fessük át a C u-t tartalmazó komponensében a kék éleket pirosra, a piros éleket kékre. Ezzel a színezést nyilván nem rontottuk el, viszont mostantól az u-nak a kék szabad színe lesz. Mivel v nem esik ebbe a komponensbe, ezért a kék v-nek is szabad színe maradt. Így az e_i él kékre színezésével továbbra is helyes színezést kapunk. □

Megjegyezzük, hogy a fenti bizonyításban szereplő C gráf minden komponense út vagy kör kell legyen és u és v is egy-egy C-beli út végpontja; ez következik abból, hogy C-ben minden pont foka legföljebb kettő (hiszen minden csúcsra legföljebb egy piros és egy kék él illeszkedhet) és u és v egy fokú (mert u-ra piros, v-re kék él nem illeszkedik). Ezzel az egyszerű megfigyeléssel pedig jelentősen egyszerűsíthető az u-t tartalmazó komponensben zajló színcsere az algoritmus végrehajtásakor, hiszen csak az u végpontú C-beli út élein kell a két színt felcserélni.

9. fejezet

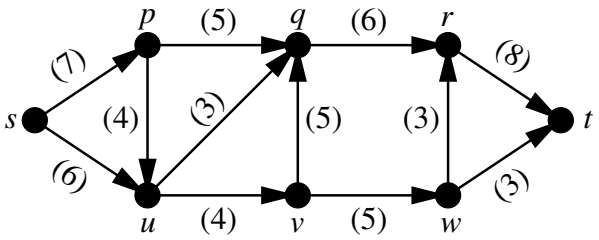
A maximális folyam feladat

Tegyük fel, hogy a G irányított gráf egy olyan hálózatot reprezentál, amin valamilyen termék szállítása zajlik. Lehet szó például úthálózatról, amin forgalom halad át; vagy egy csőhálózatról, amin olajat vagy vizet szállítanak; de akár egy számítógéphálózatról is, amin információt továbbítanak – hosszan lehetne még sorolni a példákat. Ismert a hálózat minden elemének, vagyis G minden e élének a *kapacitása*, ami azt mondja meg, hogy egy időegység alatt mennyi terméket lehet átjuttatni e-n. Ezen kívül adott még két kitüntetett csúcs: s a *termelő*, a szállítandó termék kiindulópontja és t a *fogyasztó*, a célállomás, ahová a terméket el kell juttatni. A megválaszolandó kérdés pedig az, hogy mi az a maximális termékmennyiség, amit a hálózaton (vagyis G-n) el lehet juttatni s-ből t-be úgy, hogy minden élre az azon átjutó termékmennyiség nem haladja meg az él kapacitását; illetve természetesen az is kérdés, hogy ezt a maximális mennyiséget hogyan lehet eljuttatni.

Ebben a fejezetben ennek a gyakorlati alkalmazások szempontjából igen fontos algoritmikus feladatnak a megoldásáról lesz szó. Nagyon lényeges azonban hangsúlyozni, hogy a fenti bekezdés ugyan ad egy intuitív képet a feladatról, de nem definiálja azt. Az alábbiakban ezt az intuitív képet kitöltjük precíz matematikai tartalommal és ezáltal pontosan definiáljuk a *maximális folyam* feladatot. A következő definíció a probléma bemenetét fogalmazza meg.

9.1. Definíció. *Legyen adott a $G = (V,E)$ irányított gráf, annak az $s,t \in V(G)$ egymástól különböző csúcsai és a $c : E \to \mathbb{R}^+ \cup \{0\}$ kapacitás függvény (ami tehát minden $e \in E$ élhez egy nemnegatív $c(e)$ kapacitás értéket rendel). Ekkor a (G,s,t,c) négyest* hálózatnak *nevezzük.*

A bevezetőben már említettük, hogy a hálózat kifejezést sokféle értelemben használják mind a köznyelvben, mind a szakirodalomban, gyakran a gráf fogalmával lényegében azonos jelentéssel; a fenti definíció szerint viszont a most tárgyalt témában speciális jelentéssel bír. A 9.1. ábra egy hálózatot mutat, az élek kapacitásait a zárójelben álló számok mutatják.



9.1. ábra

A maximális folyam feladat bemenete tehát egy hálózat a fenti definíció értelmében. Nehezebb kérdés, hogy hogyan adjuk meg a probléma kimenetét; vagyis hogyan írjuk le a szállítandó termékből egy adott mennyiség eljuttatásának a módját s-ből t-be? A válasz az, hogy minden e élre megadjuk az azon áthaladó termék $f(e)$ mennyiségét. Természetesen ezeknek az $f(e)$ értékeknek értelemszerű feltételeket kell kielégíteniük: egyrészt nemnegatívaknak kell lenniük és nem haladhatják meg az adott él kapacitását; másrészt pedig minden s-től és t-től különböző v csúcsra teljesülnie kell annak, hogy v-ben se nem keletkezik, se nem nyelődik el termék. Ehhez az utóbbi, *folyammegmaradási feltételnek* is nevezett követelményhez a v-be belépő és a v-ből kilépő éleken is össze kell adni az $f(e)$ értékeket és elő kell írni, hogy ezek mindig egyenlők legyenek. Mivel az alábbiakban ilyen összegek gyakran kerülnek majd elő, bevezetjük az alábbi jelöléseket:

$$\sum_{e:\, v\bullet\leftarrow} f(e) = \sum \{ f(e) : e \text{ végpontja } v \}$$

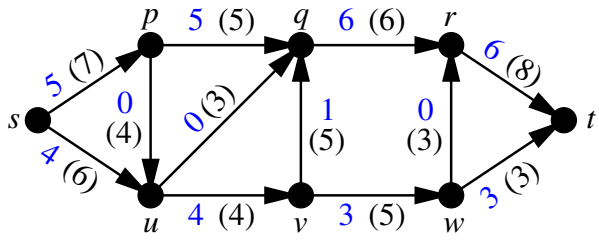
$$\sum_{e:\, v\bullet\rightarrow} f(e) = \sum \{ f(e) : e \text{ kezdőpontja } v \}$$

9.2. Definíció. *Egy adott (G,s,t,c) hálózat esetén* folyamnak *nevezzük az $f : E \to \mathbb{R}^+ \cup \{0\}$ függvényt, ha rá az alábbi feltételek teljesülnek:*

(1) $0 \le f(e) \le c(e)$ *minden $e \in E(G)$ él esetén;*

(2) $\displaystyle\sum_{e:\, v\bullet\rightarrow} f(e) = \sum_{e:\, v\bullet\leftarrow} f(e)$ *minden $v \in V(G)$, $v \neq s,t$ csúcs esetén.*

Bármely hálózatban folyam például az azonosan nulla függvény: ha $f(e) = 0$ minden $e \in E(G)$ élre, akkor nyilván teljesül a fenti definíció minden követelménye. A 9.1. ábra hálózatában pedig a 9.2. ábrán, a zárójelen kívül álló kék értékek adnak meg egy folyamot. Valóban, minden élen nulla és az adott él kapacitása közti értékek állnak és a folyammegmaradási feltételek is teljesülnek; az utóbbit például a q csúcsra ellenőrizve: $\displaystyle\sum_{e:\, q\bullet\rightarrow} f(e) = 6$ és $\displaystyle\sum_{e:\, q\bullet\leftarrow} f(e) = 5+0+1 = 6$.



9.2. ábra

A maximális folyam feladat kimenete tehát egy f folyam lesz a 9.2. Definíció értelmében – de természetesen nem mindegy, hogy milyen. Hogyan mérjük meg, hogy az f mennyi terméket szállít s-ből t-be? A válasz egyszerű: ez a mennyiség az s-et elhagyó termék nettó mennyisége; itt a „nettó" jelző arra utal, hogy ha valamilyen okból a hálózat s-be belépő éleket is tartalmaz és a folyamértékek összege ezeken pozitív, akkor ezt az összeget le kell vonni az s-ből kilépő összfolyammennyiségből. Valóban: ha például s része egy C körnek és C minden e élén $f(e) = 1$, akkor intuitíve világos, hogy az s-ből kilépő C-beli élen szállított egy egységnyi termék nem jut el t-be, hanem C mentén visszaérkezik s-be. Mivel azonban s-től és t-től különböző csúcsokban se nem keletkezik, se nem nyelődik el termék, ezért szintén intuitíve érezhető (de később ezt igazolni is fogjuk, lásd a 9.8. Állítást, illetve az utána következő megjegyzést), hogy az s-et elhagyó nettó mennyiség meg is érkezik t-be – vagyis egyenlő a t-be érkező nettó mennyiséggel (ahol a „nettó" jelző ismét arra utal, hogy a t-ből kilépő éleken vett összeget – ha ilyenek vannak – le kell vonni a t-be érkező éleken vett összegből).

9.3. Definíció. *A (G,s,t,c) hálózatban adott f folyam m_f-fel jelölt* értéke*:*

$$m_f = \sum_{e:\, s\bullet\rightarrow} f(e) - \sum_{e:\, s\bullet\leftarrow} f(e).$$

Így például a fenti ábrán látható f folyam értéke: $m_f = 5+4 = 9$. Látható az is, hogy ez valóban megegyezik a t-be érkező $6+3 = 9$ mennyiséggel.

Ezzel elérkeztünk arra a pontra, hogy precízen kitűzhetjük a *maximális folyam feladatot*: egy, a 9.1. Definíció szerint adott hálózat esetén keresünk egy, a 9.2. Definíciónak megfelelő f folyamot úgy, hogy annak a 9.3. Definíció szerinti m_f értéke maximális legyen.

Lényeges tehát hangsúlyozni: a maximális folyam feladat kapcsán (is) szét kell választanunk a bennünk élő intuitív képet a precíz matematikai megfogalmazástól. A fejezet elején írt, valamilyen fajta hálózaton zajló szállítást ábrázoló kép ugyan segíti a feladat megértését és érzékelteti annak a gyakorlati fontosságát, de annyiban mégis félrevezető, hogy hajlamosak lehetünk miatta a feladatra úgy gondolni, mint-

ha az a G gráfban s-ből t-be vezető utak kereséséről szólna. Ez tehát tévedés volna, a valóságot a 9.1-9.3. Definíciók írják le.

A teljesség kedvéért megemlítjük, hogy a szállított termék s-ből t-be való áramlását ábrázoló intuitív kép és a fenti definíciók között van pontosan leírható kapcsolat, de ez nem teljesen magától értetődő. Be lehet bizonyítani, hogy ha adott a (G,s,t,c) hálózatban az f folyam, akkor létezik s-ből t-be vezető G-beli irányított utaknak egy olyan $\{P_1,P_2,\dots,P_p\}$ halmaza és G-beli irányított köröknek egy olyan $\{C_1,C_2,\dots,C_q\}$ halmaza, valamint az ezekhez tartozó $\alpha_1,\alpha_2,\dots,\alpha_p$, illetve $\beta_1,\beta_2,\dots,\beta_q$ nemnegatív valós értékek, hogy minden e él esetén az e-t tartalmazó P_i-khez tartozó α_i-k összege plusz az e-t tartalmazó C_j-khez tartozó β_j-k összege $f(e)$-t adja; valamint $\sum_{i=1}^{p}\alpha_i = m_f$. Könnyű ugyanakkor példát mutatni arra, hogy ugyanez csak irányított utakkal, körök említése nélkül már nem volna igaz.

Megemlítünk ezen a ponton egy másik, a maximális folyam definíciójával kapcsolatos elméleti kérdést is: egyáltalán nem magától értetődő, hogy maximális folyamérték (vagyis a 9.3. Definíció szerinti m_f értékek maximuma) létezik. Bármely hálózat esetén a folyamértékek halmaza nyilván nemüres (mert láttuk, hogy $f\equiv 0$ folyam) és felülről korlátos (mert például az s-ből kilépő élek összkapacitása nyilván felső korlát m_f-re); de az általában nem igaz, hogy a valós számok minden nemüres és felülről korlátos részhalmazának van maximuma – gondoljunk például az 1-nél kisebb számok halmazára. Be lehet azonban bizonyítani (és a későbbiekből következni is fog), hogy maximális folyam valóban minden hálózatban létezik; itt csak azt kívántuk hangsúlyozni, hogy ez az állítás messze nem magától értetődő.

9.1. A javítóutas algoritmus maximális folyam keresésére

Az alább bemutatandó, Lester R. Ford (1927 – 2017) és Delbert R. Fulkerson (1924 – 1976) amerikai matematikusoktól, 1956-ból származó, maximális folyam keresésére szolgáló eljárást szintén javítóutas algoritmusnak szokták nevezni, hasonlóan a 7.2. szakaszban látott eljáráshoz. A két algoritmus természetesen egészen más, de azért a névrokonságnál szorosabb a kapcsolatuk: ennek az eljárásnak a magva is egy növelő lépés, amivel az aktuálisan nyilvántartott f folyamból egy annál nagyobb értékűt lehet előállítani. Továbbá ez a növelő lépés is javítóút keresésén alapul – de ez a fogalom most egészen mást fog jelenteni, mint a 7.2. szakaszban. (A két javítóutas algoritmus kapcsolatáról bővebben lesz még szó a 9.17. Lemma utáni apróbetűs részben.)

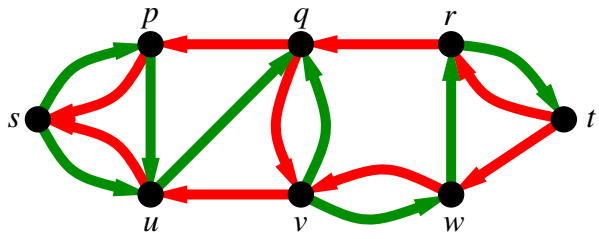
A kérdés tehát az, hogy hogyan lehetne az aktuális f folyamon javítani? Elég természetes gondolat, hogy ha találunk s-ből t-be egy olyan P irányított utat, aminek minden e élére $f(e)<c(e)$, akkor P mentén a folyam javítható: ha δ-val jelöljük a $c(e)-f(e)$ értékek P élein vett minimumát és P minden élére $f(e)$-t megnöveljük δ-val, akkor egy új folyamot kapunk, amire m_f is δ-val nagyobb. Valóban: egyrészt δ választása miatt az $f(e)$ értékek sehol sem nőnek $c(e)$ fölé; másrészt a folyammegmaradási feltételek is igazak maradnak, hiszen a P által érintett minden v csúcsra $\sum_{e:\,v\bullet\leftarrow} f(e)$ és $\sum_{e:\,v\bullet\rightarrow} f(e)$ is δ-val nő; végül m_f növekedését az okozza, hogy P első

e éle s-ből indul és $f(e)$ ezen is δ-val nő. Ha azonban erre az egyszerű gondolatra alapoznánk az algoritmus javító lépését, akkor az nem adna maximális folyamot; például a 9.2. ábrán látható folyamon ezzel a módszerrel már nem lehetne javítani, pedig rövidesen látni fogjuk, hogy az nem maximális. El kell ugyanis fogadnunk, hogy a folyam globális javításához lokálisan csökkentésekre is szükség lehet: előfordulhat, hogy bizonyos e élekre $f(e)$-t csökkenteni kell ahhoz, hogy m_f nőhessen. A fenti, s-ből t-be vezető irányított úton alapuló gondolatot éppen ezzel fogjuk kiegészíteni: nem csak azokra az élekre koncentrálunk, amiken $f(e) < c(e)$ (és így rajtuk $f(e)$ növelhető), hanem azokra is, amiken $f(e) > 0$ (és így rajtuk $f(e)$ csökkenthető); a ravasz ötlet az lesz, hogy az utóbbi típusú éleken az s-ből t-be vezető út keresésekor nem az eredeti irányukban, hanem fordítva haladunk át. Ez motiválja az alábbi, a folyamok elméletében alapvető fontosságú definíciót.

9.4. Definíció. *Legyen f folyam a (G,s,t,c) hálózatban. Ekkor az f-hez tartozó H_f* segédgráfot *a következőképpen definiáljuk. H_f irányított gráf, amire $V(H_f) = V(G)$, vagyis H_f csúcsainak halmaza azonos G csúcshalmazával. Továbbá H_f élhalmazába kétféle típusú él kerül:*

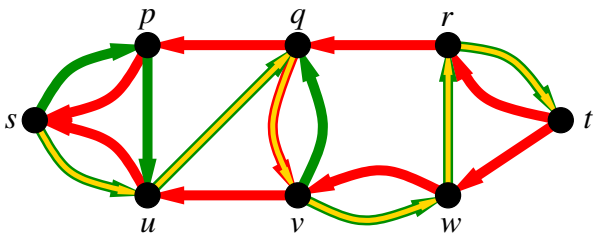
- *Ha $e = (u,v)$ olyan éle G-nek, amire $f(e) < c(e)$, akkor $e = (u,v)$ a H_f élhalmazába is bekerül; az ilyen élek neve* előreél.
- *Ha $e = (u,v)$ olyan éle G-nek, amire $f(e) > 0$, akkor e megfordítása, az $e' = (v,u)$ él kerül be H_f élhalmazába; az ilyen élek neve pedig* visszaél.

Például a 9.2. ábra f folyamához tartozó H_f segédgráf a 9.3. ábrán látható; ezen az előreéleket zölddel, a visszaéleket pirossal ábrázoltuk. Látható, hogy G minden éléből H_f-nek legalább egy éle keletkezett: ha $0 < f(e) < c(e)$, akkor e és a megfordítása is H_f-be került (zölddel, illetve pirossal), ha viszont $f(e) = 0$, illetve $f(e) = c(e)$, akkor csak e, illetve csak e megfordítása került H_f-be.



9.3. ábra

9.5. Definíció. *Legyen f folyam a (G,s,t,c) hálózatban. Ekkor a H_f-beli, s-ből t-be vezető irányított utakat* javítóútnak *nevezzük f-re nézve.*

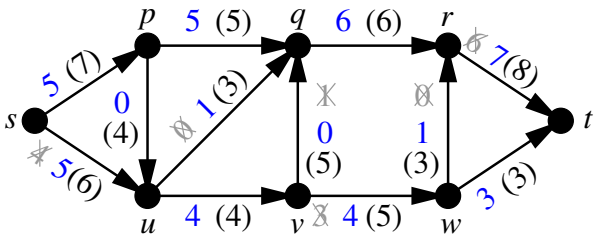


9.4. ábra

A 9.4. ábra például egy s-ből t-be vezető irányított utat mutat (sárgával) a 9.3. ábra segédgráfjában. Ez tehát javítóút a 9.2. ábrán látható folyamra nézve.

Hogyan használható javításra egy P javítóút? Egy alkalmasan választott δ értékre δ-val növeljük a folyam értékét a P előreélein és δ-val csökkentjük a folyam értékét a P visszaéleinek megfelelő G-beli éleken (vagyis azok megfordítottjain). δ megválasztásánál pedig csak arra figyelünk, hogy a folyam értéke minden e élen 0 és $c(e)$ között maradjon; ezen belül viszont a lehető legnagyobb δ-t választjuk.

A fenti ábra P javítóútjának hat éle közül öt előreél: (s,u), (u,q), (v,w), (w,r) és (r,t); ezeken sorra legföljebb 2-vel, 3-mal, 2-vel, 3-mal, illetve 2-vel lehet növelni a folyam értékét (ezek tehát a $c(e)-f(e)$ értékek ezen az öt élen). Az út élei közül viszont egy, a (q,v) visszaél; ennek a G-ben szereplő megfordítottján, a (v,q)-n legföljebb 1-gyel lehet csökkenteni a folyam értékét (hiszen $f\big((v,q)\big)=1$). Ha tehát az öt előreélen δ-val növelni, a (v,q)-n pedig δ-val csökkenteni szeretnénk a folyam értékét, akkor a fentiekből $\delta \leq \min\{2,3,2,3,2,1\}$ következik. Így a legnagyobb megfelelő érték a $\delta=1$, ezt fogjuk tehát választani. Elvégezve a megfelelő változtatásokat (vagyis (v,q)-n az 1-gyel való csökkentést, a P öt darab előreélén pedig az 1-gyel való növelést) a 9.5. ábrán látható eredményt kapjuk. (Szürkével, áthúzva az eredeti $f(e)$ értékek láthatók.)



9.5. ábra

Figyeljük meg, hogy a módosítások után szintén folyamot kaptunk: nem csak a kapacitás követelmények teljesülnek (amit δ választásával garantáltunk), hanem a folyammegmaradási feltételek is fennállnak minden s-től és t-től különböző csúcsra. A folyam értéke viszont δ-val nőtt: az s-ből kiinduló termékmennyiség 10. Később

megmutatjuk (lásd a 9.6. Állítást), hogy mindez általában is igaz.

A (maximális folyam keresésére szolgáló) javítóutas algoritmus a fenti példában látott növelő lépés ismételt alkalmazásából áll. Persze kérdés, hogy a javítóút keresése a segédgráfban hogyan történjen. Szerencsére a 2. fejezetben látott BFS algoritmus irányított gráfokra vonatkozó változata erre hatékony megoldást kínál.

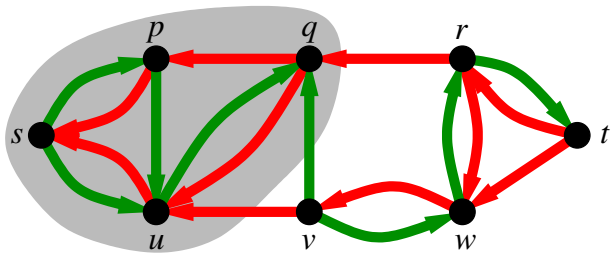
JAVÍTÓUTAS ALGORITMUS (MAXIMÁLIS FOLYAM KERESÉSÉRE)

Bemenet: egy (G,s,t,c) hálózat.

1 Minden $e \in E(G)$-re $f(e) \leftarrow 0$ (vagy f lehet tetszőleges kezdeti folyam).
2 **ciklus**
3 Készítsük el az f-hez tartozó H_f segédgráfot a 9.4. Definíció szerint.
4 FUTTASSUK A BFS ALGORITMUST P javítóút keresésére f-re nézve; vagyis keressünk H_f-ben egy P irányított utat s-ből t-be.
5 **ha** nem létezik f-re nézve javítóút, **akkor**: **stop**
6 **különben**:
$\delta_1 \leftarrow \min\{c(e) - f(e) : e$ előreéle P-nek$\}$
7 $\delta_2 \leftarrow \min\{f(e) : e$ megfordítottja visszaéle P-nek$\}$
$\delta \leftarrow \min\{\delta_1, \delta_2\}$
Minden $e \in E(G)$-re
8 $$f(e) \leftarrow \begin{cases} f(e)+\delta, & \text{ha } e \text{ előreéle } P\text{-nek;} \\ f(e)-\delta, & \text{ha } e \text{ megfordítottja visszaéle } P\text{-nek;} \\ f(e), & \text{egyébként.} \end{cases}$$
9 **ciklus vége**

Az eljárás 7. és 8. sorában írtak megfelelnek annak, ahogyan a fenti példában a folyamot változtattuk: δ_1 a P előreélein lehetséges növelések minimuma, δ_2 a P visszaéleinek megfordításain lehetséges csökkentések minimuma, δ pedig az összes ilyen érték minimuma; a 8. sorban pedig a megfelelő növelések és csökkentések valósulnak meg. A 3. sorban írtakat érdemes kiegészíteni azzal, hogy H_f-et szerencsére nem szükséges minden alkalommal teljesen újrageneránli: ennél hatékonyabb, ha az előző segédgráfot frissítjük. Valóban, mivel $f(e)$ értéke nem változott azokon a G-beli éleken, amiknek megfelelő H_f-beli élt az előző P javítóút nem tartalmazott, így az ezekből keletkezett H_f-beli élek sem változhattak. Ha például folytatni akarjuk az algoritmus futtatását a fejezetben fentebb látott példán, akkor a 9.3. ábrán látható segédgráfot kell frissítenünk – de csak annak a hat G-beli élnek megfelelő helyeken, amiket a javítóút érintett. Ezek közül is az (s,u), (v,w) és (r,t) éleken nincs változás, mert ugyan ezeken a folyam értéke nőtt 1-gyel, de a kapacitásnál (szigorúan) kisebb maradt, így a folyam továbbra is növelhető és csökkenthető is. Az (u,q) és (w,r) éleken viszont $f(e)$ értéke 0-ról 1-re nőtt, így ezeknek a megfordítottjait be kell venni H_f-be (de az eredeti irányukban is megmaradnak, mert mindkettőnek a kapacitása 1-nél nagyobb). Végül a (v,q) élen $f(e)$ 0-ra csökkent, így itt a (q,v) visszaél kikerül H_f-ből. A kapott segédgráf a 9.6. ábrán látható.

Látható, hogy ebben a segédgráfban már nincs irányított út s-ből t-be (vagyis



9.6. ábra

a 9.5. ábrán látható folyamra nézve már nincs javítóút), mert s-ből csak az $\{s,p,q,u\}$ halmaz csúcsaiba lehet eljutni, de H_f-nek nincs ebből a halmazból kilépő éle. Így ezen a ponton a javítóutas algoritmus megáll.

9.2. A javítóutas algoritmus vizsgálata

Ahhoz, hogy a fenti algoritmus működőképességéről meggyőződjünk, az alábbi kérdéseket kell feltennünk és megválaszolnunk.

1. Helyesen működik-e az eljárás? Vagyis: igaz-e, hogy a 8. sorban végrehajtott változtatások után is mindig folyamot kapunk?
2. Optimális eredményt ad-e az algoritmus? Vagyis: igaz-e, hogy ha már nincs f-re nézve javítóút (és ezért az eljárás megáll), akkor f maximális folyam?
3. Hatékony-e az algoritmus? Vagyis: igaz-e, hogy egyrészt az eljárás mindig véges időben megáll, másrészt a futásideje polinomiális?

Ebben a szakaszban ezeket a kérdéseket vizsgáljuk meg. Közülük az elsőt az alábbi állítás válaszolja meg.

9.6. Állítás. *Ha f folyam, akkor a javítóutas algoritmus 8. sorában végrehajtott változtatások után is az marad és m_f értéke δ-val nő.*

Bizonyítás: Ha e előreéle a P javítóútnak, akkor $\delta \leq \delta_1 \leq c(e) - f(e)$ miatt $f(e)$ értéke nem nő $c(e)$ fölé. Hasonlóan, ha e megfordítottja visszaéle P-nek, akkor $\delta \leq \delta_2 \leq f(e)$ miatt $f(e)$ értéke nem csökken nulla alá. Így $0 \leq f(e) \leq c(e)$ a változtatások után is igaz marad G minden élére.

Megmutatjuk, hogy a folyammegmaradási feltételek is igazak maradnak minden $v \in V(G)$, $v \neq s,t$ csúcsra. Ha P nem halad át v-n, akkor a folyam értéke a v-re illeszkedő éleken nem változik, így a folyammegmaradási feltétel is nyilván igaz marad v-re. (Példa erre a 9.4. ábrán a p csúcs.) Ha viszont P áthalad v-n, akkor jelölje e_1, illetve e_2 P-nek a v-be belépő, illetve v-ből kilépő élét. Mivel e_1 és e_2 is lehet előreél vagy visszaél, ezért négy esetet kell megvizsgálnunk.

1. Ha e_1 és e_2 is előreél, akkor mindketten azonos irányítással szerepelnek G-ben is és mindkettőn δ-val nő $f(e)$ értéke. Így $\sum_{e:v\bullet\leftarrow} f(e)$ és $\sum_{e:v\bullet\rightarrow} f(e)$ is δ-val nő,

ezért egyenlők maradnak. (A 9.4. ábrán példát látunk erre az esetre az u, w és r csúcsoknál.)

2. Ha e_1 és e_2 is visszaél, akkor mindketten fordított irányítással szerepelnek G-ben és mindkettő megfordítottján δ-val csökken $f(e)$ értéke. Így δ-val csökken $\sum_{e:\, v\bullet\leftarrow} f(e)$ és $\sum_{e:\, v\bullet\rightarrow} f(e)$ is, ezért egyenlők maradnak.
3. Ha e_1 előreél és e_2 visszaél, akkor e_1 azonos irányban, e_2 pedig fordított irányban szerepel G-ben (és így az utóbbi is belép v-be). A folyam értéke pedig e_1-en δ-val nő, e_2 megfordításán viszont δ-val csökken. Így $\sum_{e:\, v\bullet\leftarrow} f(e)$ nem változik (mert a δ-val való növekedés és csökkenés kiejti egymást), de persze nem változik $\sum_{e:\, v\bullet\rightarrow} f(e)$ sem, mert ezeken az éleken $f(e)$ nem változik. Így a folyammegmaradás v-nél igaz marad. (Erre az esetre példa a q csúcs a 9.4. ábrán.)
4. Végül ha e_1 visszaél, e_2 pedig előreél, akkor a helyzet az előbbivel analóg: a folyam értéke δ-val csökken e_1 megfordításán és δ-val nő e_2-n; így a v-ből kilépő éleken a δ-val való csökkenés és növekedés kiejti egymást, a v-be belépő éleken pedig nincs változás. A folyammegmaradás tehát ebben az esetben is igaz marad. (A 9.4. ábrán a v csúcs példa erre az esetre.)

Végül megmutatjuk, hogy m_f δ-val nő. Jelölje e_0 a P első, s-ből induló élét. Ha e_0 előreél, akkor $f(e_0)$ δ-val nő (és a többi s-re illeszkedő élen nincs változás), így az s-ből kilépő nettó folyam mennyisége (lásd a 9.3. Definíciót) valóban δ-val nő. (Ez történt például a 9.4. ábrán is.) Ha viszont e_0 visszaél, akkor e_0 megfordítottja szerepel G-ben, ami tehát s-be belép; mivel ezen $f(e)$ értéke δ-val csökken, ezért $\sum_{e:\, s\bullet\leftarrow} f(e)$ is δ-val csökken. Így m_f értéke ebben az esetben is δ-val nő (mert az s-ből kilépő összfolyammenyiségből δ-val kevesebbet vonunk le). □

A következő célunk annak megmutatása, hogy ha f-re nézve nincs javítóút, akkor m_f maximális; ehhez egy, a folyamok elméletében alapvető fontosságú segédeszközt kell megismernünk.

Képzeljük el, hogy (G,s,t,c) egy úthálózatot, az f folyam pedig ezen valamilyen titkos katonai eszközök szállítását reprezentálja. Szeretnénk megtudni a folyam m_f értékét, de s-et és t-t nem közelíthetjük meg, mert ezek védett katonai objektumok. Mit tehetünk? Szerencsére a terepet észak-déli irányban átszeli egy folyó, aminek s a nyugati, t pedig a keleti partján fekszik. Ha a folyón átívelő összes hídra (amelyek persze irányított élek) kémeket küldünk és ezek jelentik a hidakon folyó folyam értékét, akkor intuitíve világos, hogy ezekből m_f meghatározható: ez a nyugati partról a keletire átjutó nettó termékmennyiség kell legyen – vagyis a nyugati partról a keletire menő e hidakon vett $f(e)$ értékek összegének és a keleti partról a nyugatira menő e hidakon vett $f(e)$ értékek összegének a különbsége. Alább ezt az ötletet szeretnénk pontosan megfogalmazni és bebizonyítani, de ehhez először általánosítanunk kell a fenti kémtörténetben főszerepet játszó folyó fogalmát. Könnyű belegondolni, hogy a folyó szerepe csak annyi, hogy a terepet kettévágja – vagyis G csúcshalmazát két részre osztja: a nyugati, illetve a keleti parton fekvőkre. Ez a kettéosztás pedig tetszőleges lehet feltéve, hogy s a nyugati, t pedig a keleti parton

fekszik. Az alábbi, alapvető fogalom ennek a kettéosztásnak ad nevet.

9.7. Definíció. *A (G,s,t,c) hálózatban st-*vágásnak *nevezzük az $X \subseteq V(G)$ csúcshalmazt, ha rá $s \in X$ és $t \notin X$ teljesül. Ha a szövegkörnyezetből s és t szerepe egyértelmű, akkor X-et st-vágás helyett röviden* vágásnak *is hívjuk.*

Amint látható, a definíció a csúcsok kettéosztása helyett a csúcsok egy részhalmazát nevezi vágásnak – mégpedig X a nyugati parton fekvő csúcsok halmazának felel meg. Persze ennek a kijelölése tartalmilag azonos a csúcshalmaz kettévágásával (amennyiben az X-be nem tartozó csúcsok alkotják a keleti partot), ez a megfogalmazás azonban egyszerűbben kezelhető lesz. Érdemes megemlíteni, hogy több tankönyv az X és $V(G) \setminus X$ csúcshalmazok között futó élek halmazát (tehát a kémtörténet hidainak megfelelő éleket) nevezi vágásnak; tartalmilag ez is azonos a fenti definícióval, de körülményesebbé tenné a tárgyalást.

Az alábbi állítás a folyamérték kémek jelentésein alapuló meghatározásáról szóló fenti, intuitív leírást fogalmazza meg pontosan. Az állítás szövegében az $e = (u,v) \in E(G)$ élt *X-ből kilépőnek* nevezzük, ha $u \in X$ és $v \notin X$; hasonlóan, az *e belép X-be*, ha $u \notin X$ és $v \in X$.

9.8. Állítás. *Ha f tetszőleges folyam, X pedig tetszőleges vágás a (G,s,t,c) hálózatban, akkor*

$$m_f = \sum\{f(e) : e \text{ kilép } X\text{-ből}\} - \sum\{f(e) : e \text{ belép } X\text{-be}\}.$$

Alkalmazzuk például az állítást a 9.5. ábrán látható folyamra és az $X = \{s,p,q\}$ vágásra. Ekkor az X-ből kilépő élek (s,u), (p,u) és (q,r), ezeken a folyamértékek összege $5+0+6 = 11$; az X-be belépő élek pedig (u,q) és (v,q), amiken az összfolyamérték $1+0 = 1$. Valóban látható, hogy a kilépő és belépő összfolyamértékek különbsége $11-1 = 10$, ami egyenlő m_f-fel.

A 9.8. Állítás bizonyítása: Írjunk fel X minden csúcsára egy-egy egyenletet:

$$s\text{-re}: \quad m_f = \sum_{e:\, s\bullet\!\rightarrow} f(e) - \sum_{e:\, s\bullet\!\leftarrow} f(e)$$

$$\text{minden } x \in X, x \neq s \text{ csúcsra}: \quad 0 = \sum_{e:\, x\bullet\!\rightarrow} f(e) - \sum_{e:\, x\bullet\!\leftarrow} f(e)$$

Ezek az egyenletek persze mind igazak: az első a folyamérték 9.3. Definíciója, a többi pedig a folyam 9.2. Definíciójának folyammegmaradási követelményéből fakad (átrendezés után).

Adjuk most össze ezt az $|X|$ darab egyenletet és vizsgáljuk meg, hogy mit kapunk. A bal oldalon persze m_f áll (hiszen ehhez $(|X|-1)$-szer nullát adtunk). Az összeg jobb oldalát pedig úgy tekinthetjük át, ha minden e élre megnézzük, hogy $f(e)$ hányszor és milyen előjellel jelenik ott meg. Válasszunk ezért egy tetszőleges $e = (u,v)$ élt. Mivel u és v is lehet X-ben vagy azon kívül, ezért négy esetet kell megvizsgálnunk.

1. Ha $u \in X$ és $v \in X$, vagyis e X-en belül halad, akkor $f(e)$ kétszer is megjelenik az összeg egyenlet jobb oldalán: egyszer az $x = u$ csúcsra felírt egyenletből pozitív előjellel (hiszen e kilép u-ból) és egyszer az $x = v$-re felírt egyenletből negatív előjellel (hiszen e belép v-be); így ebben az esetben végül is $f(e)$ nem járul hozzá az összeg jobb oldalához, mert a két előfordulása kiejti egymást.
2. Ha $u \in X$ és $v \notin X$, vagyis e kilép X-ből, akkor $f(e)$ egyszer, az $x = u$ csúcsra felírt egyenlet jobb oldalán jelenik meg, mégpedig pozitív előjellel (mert e kilép u-ból). Így $f(e)$ az összeg jobb oldalán is megjelenik.
3. Ha $u \notin X$ és $v \in X$, vagyis e belép X-be, akkor $f(e)$ szintén egyszer jelenik meg az összeg jobb oldalán, de negatív előjellel: az $x = v$-re felírt egyenletből (mert e belép v-be).
4. Végül az $u \notin X$, $v \notin X$ esetben $f(e)$ egyáltalán nem jelenik meg az egyenletek jobb oldalán (és így persze az összegben sem).

Összefoglalva a fentieket: az összeg egyenlet jobb oldalán az X-ből kilépő élekre az $f(e)$ 1-szeres szorzóval, az X-be belépőkre pedig (-1)-szeres szorzóval jelenik meg. Más szóval: a felírt $|X|$ darab egyenlet összege éppen az állításbeli egyenlet, ami így nyilván szintén igaz. □

Érdemes alkalmazni a fenti állítást egy tetszőleges (G,s,t,c) hálózatbeli f folyamra és az $X = V(G) \setminus \{t\}$ vágásra: ekkor az X-ből kilépő, illetve X-be belépő összfolyamérték megegyezik a t-be belépő, illetve t-ből kilépő összfolyamértékkel; így azt kapjuk, hogy $m_f = \sum_{e:\, t\bullet\leftarrow} f(e) - \sum_{e:\, t\bullet\rightarrow} f(e)$. A folyamérték 9.3. Definíciója előtt említettük, hogy ez az állítás intuitíve természetes: mivel s-től és t-től különböző csúcsokban nem keletkezik és nem is nyelődik el termék, ezért az s-et elhagyó nettó mennyiségnek azonosnak kell lenni a t-be érkező nettó mennyiséggel. A fenti állításból tehát ennek a pontos bizonyítása is adódik. A 9.8. Állítás fő alkalmazása azonban a következő állítás bizonyításában lesz.

Mivel a célunk továbbra is annak megmutatása, hogy a javító utas algoritmus leállása után a folyam maximális, ezért szükségünk lesz egy olyan eszközre, amivel kimutatható a kapottnál nagyobb folyamérték lehetetlensége. Térjünk vissza a vágás fogalmát motiváló, a terepet észak-déli irányban kettévágó folyó példájára. Természetesnek érződik, hogy ha összeadjuk a folyón átívelő összes, a nyugati partról a keletire tartó híd kapacitását, akkor ezzel felső becslést kapunk a folyam értékére: valóban, ennél többet nyilván nem lehet „átpumpálni” a nyugati partról a keletire, így ennél több s-ből t-be sem juthat el. Ez a gondolat motiválja az alábbi definíciót.

9.9. Definíció. *A (G,s,t,c) hálózatban az X st-vágás $c(X)$-szel jelölt* kapacitása *a*

$$c(X) = \sum \{c(e) : e \text{ kilép } X\text{-ből}\}$$

összeg (ahol a fentieknek megfelelően $e = (u,v)$ az X-ből kilépő él, ha $u \in X$ és $v \notin X$). A vágás kapacitását a vágás értékének *is szokták nevezni.*

A vágás kapacitásával kapcsolatos tipikus félreértéseket elkerülendő fontos hangsúlyozni egyrészt azt, hogy $c(X)$ meghatározásakor az X-be belépő élek sem-

milyen szerepet nem játszanak (így ezeknek a kapacitása se nem adódik hozzá, se nem vonódik le a kilépő kapacitások összegéből); másrészt pedig azt, hogy $c(X)$ számításakor az élek *kapacitásait* adjuk össze, nem pedig valamilyen f folyamra az $f(e)$ folyamértékeket.

9.10. Feladat. Határozzuk meg a 9.1. ábrán látható hálózatban az $X = \{s, p, q\}$ vágás $c(X)$ kapacitását.

Megoldás: Az X-ből kilépő élek (amint azt fentebb már láttuk) (s,u), (p,u) és (q,r), ezeken a kapacitások összege pedig $6+4+6 = 16$. Így $c(X) = 16$. □

A folyón nyugatról keletre átívelő hidak összkapacitásáról szóló fenti, intuitív gondolatot az alábbi állítás fogalmazza meg pontosan.

9.11. Állítás. *Ha f tetszőleges folyam, X pedig tetszőleges vágás a (G,s,t,c) hálózatban, akkor $m_f \leq c(X)$.*

Bizonyítás:

$$m_f = \sum\{f(e) : e \text{ kilép } X\text{-ből}\} - \sum\{f(e) : e \text{ belép } X\text{-be}\} \leq$$
$$\leq \sum\{c(e) : e \text{ kilép } X\text{-ből}\} - 0 = c(X),$$

ahol az első egyenlőség a 9.8. Állításból, az utolsó a vágás kapacitásának fenti definíciójából fakad, az egyenlőtlenség pedig a folyam 9.2. Definíciójának kapacitás feltételéből: az $f(e) \leq c(e)$ felső becslést, illetve az $f(e) \geq 0$ alsó becslést alkalmaztuk az X-ből kilépő, illetve az X-be belépő élekre. □

Ennek az állításnak a segítségével már be tudjuk bizonyítani az alábbi, a szakasz elején említett Fordtól és Fulkersontól, 1956-ból származó tételt.

9.12. Tétel. *Ha a (G,s,t,c) hálózatban az f folyamra nézve nincs javítóút, akkor f maximális folyam (vagyis nincs m_f-nél nagyobb értékű folyam).*

Bizonyítás: Megmutatjuk, hogy létezik olyan X st-vágás, aminek a kapacitása $c(X) = m_f$. Ebből a 9.11. Állítás szerint következni fog, hogy minden f' folyamra $m_{f'} \leq c(X) = m_f$ teljesül és ezért f értéke valóban maximális.

Álljon X azokból a v csúcsokból, amikre létezik s-ből v-be vezető irányított út a H_f segédgráfban. Ekkor $t \notin X$, mert f-re nézve nem létezik javítóút. Másrészt $s \in X$ világos, így X valóban st-vágás.

Azt állítjuk, hogy ha $e = (u,v)$ tetszőleges, X-ből kilépő él, akkor rá $f(e) = c(e)$ teljesül. Ha ez nem volna igaz, akkor $f(e) < c(e)$ miatt (a 9.4. Definíció szerint) e bekerülne H_f élhalmazába (előreélként). Így egy H_f-ben s-ből u-ba vezető irányított út (ami $u \in X$ miatt létezik) megtoldható volna az $e = (u,v)$ éllel v-ig és így egy

s-ből v-be vezető irányított utat kapnánk; ez azonban ellentmond annak, hogy $v \notin X$ (hiszen e kilép X-ből).

Megmutatjuk azt is, hogy ha $e = (u,v)$ X-be belépő él, akkor rá $f(e) = 0$ teljesül. Ha ez nem volna igaz, akkor $f(e) > 0$ miatt $e' = (v,u)$, az e megfordítottja bekerülne H_f élhalmazába visszaélként. Így egy H_f-ben s-ből v-be vezető irányított út (ami $v \in X$ miatt létezik) megtoldható volna e'-vel u-ig és így egy s-ből u-ba vezető irányított utat kapnánk; ez is ellentmondás, hiszen e belép X-be és így $u \notin X$.

Mindezekből az következik, hogy f-re és X-re megismételhető a 9.11. Állítás bizonyításának a számolása – azzal a lényeges különbséggel, hogy egyenlőtlenség helyett egyenlőséget írhatunk:

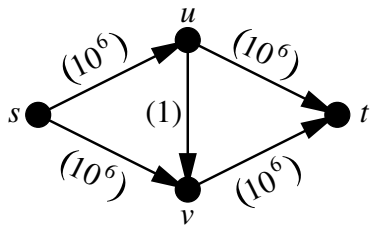
$$m_f = \sum\{f(e) : e \text{ kilép } X\text{-ből}\} - \sum\{f(e) : e \text{ belép } X\text{-be}\} = \\ = \sum\{c(e) : e \text{ kilép } X\text{-ből}\} - 0 = c(X).$$

Valóban: a fentiek szerint az X-ből kilépő, illetve az X-be belépő élekre $f(e) = c(e)$, illetve $f(e) = 0$ teljesül (ezeken kívül pedig ismét csak a 9.8. Állítást és a 9.2. Definíciót alkalmaztuk).

Beláttuk tehát, hogy $m_f = c(X)$, amivel a tétel bizonyítása teljes. □

Érdemes kipróbálni a fenti bizonyítás alapötletét a 9.5. ábrán látható 10 értékű folyamra. A 9.6. ábra mutatta, hogy erre nézve már nincs javítóút, az s-ből elérhető csúcsok halmaza pedig $X = \{s,p,q,u\}$. Erre $c(X) = 10$, mert az X-ből kilépő élek (q,r) és (u,v), ezek kapacitásösszege pedig $6+4 = 10$. Így X a 9.11. Állítás szerint valóban mutatja, hogy 10-nél nagyobb értékű folyam nem létezhet a hálózatban. A fenti bizonyításnak az X-ből kilépő, illetve X-be belépő élekre vonatkozó állításai is jól illusztrálhatók ezen a példán: a két X-ből kilépő élre $f\big((q,r)\big) = 6 = c\big((q,r)\big)$ és $f\big((u,v)\big) = 4 = c\big((u,v)\big)$, az egyetlen X-be belépő élre pedig $f\big((v,q)\big) = 0$ teljesül.

A javítóutas algoritmussal kapcsolatban a szakasz elején felsorolt kérdések közül tehát már csak az utolsó van hátra: igaz-e, hogy az eljárás mindig véges, sőt, polinomiális futásidejű? Erre már jóval nehezebb válaszolni – sőt, meglepő szempontok és tények is felmerülnek. Lehet ugyanis példát mutatni arra, hogy ha az algoritmus belsejében a javítóút keresését nem a BFS eljárással végezzük, hanem tetszőleges irányított utat választhatunk H_f-ben s-ből t-be, akkor (kellően szerencsétlen választások esetén) előfordulhat, hogy az algoritmus nem áll meg véges időben. Ez tehát azt jelenti, hogy egy ilyen esetben az eljárás a végtelenségig egymás után készíti az újabb f folyamokat, amikre (a korábban bizonyítottak szerint) m_f mindig nő – de sosem érkezik el egy olyan pillanat, amikor H_f-ben már nincs javítóút. (Könnyű megmutatni azonban – lásd a 9.17. Lemma utáni apróbetűs részt –, hogy ez csak akkor fordulhat elő, ha az élek kapacitásai között irracionális számok is vannak.) Erre itt nem mutatunk példát, mert nagyon körülményes volna, de a javítóutak választásának fontosságára a következő, jóval egyszerűbb példa is rámutat: ha a 9.7. ábra hálózatában úgy futtatjuk az algoritmust az azonosan nulla folyamtól indítva, hogy felváltva választjuk az $s \to u \to v \to t$ és az $s \to v \to u \to t$ javítóutat (és így az (u,v) élen a folyam értéke váltakozva 1, illetve 0 lesz), akkor az eljárás kétmillió javítás után áll meg; ha viszont az $s \to u \to t$ és $s \to v \to t$ javítóutakat választjuk, akkor már két javítás után is célhoz ér.



9.7. ábra

Az alábbi, itt bizonyítás nélkül közölt tételt egymástól függetlenül fedezte fel Yefim Dinitz (1951 –) szovjet matematikus 1970-ben és Jack Edmonds (1934 –) és Richard Karp (1935 –) amerikai matematikusok 1972-ben.

9.13. Tétel. *Ha a G gráf n csúcsú és m élű és a (G,s,t,c) hálózatban a maximális folyam keresésére szolgáló javítóutas algoritmus futása során H_f-ben mindig az egyik legrövidebb (vagyis legkevesebb élű) s-ből t-be vezető irányított utat választjuk javítóútnak, akkor az eljárás legfőljebb $n \cdot m$ javító lépés után megáll.*

A fenti tételből tehát következik, hogy ha a javítóutakat mindig a BFS algoritmus segítségével keressük (a 9.1. szakaszban látott pszeudokódnak megfelelően), akkor a javítóutas algoritmus igen hatékony, polinomiális futásidejű eljárás. Pontosabban: mivel a segédgráf élszáma mindig legfőljebb $2m$ (hiszen G minden éléből a segédgráfban legfőljebb két él keletkezik), ezért a javítóút keresésére szolgáló BFS eljárás lépésszáma felülről becsülhető m egy konstansszorosával. (Itt a BFS lépésszámára az általában érvényes $(n+m)$-mel arányos helyett m-mel arányos felső becslést is használhatunk, mert most csak az s-ből irányított úton elérhető csúcsok feltérképezése a cél, amiknek a száma az 1.23. Tétel *(i)* állításából következően legfőljebb $m+1$.) Nem nehéz végiggondolni azt sem (a részletektől itt eltekintünk), hogy a pszeudokód 7. és 8. sorában írt számítások és módosítások is elvégezhetők m-mel arányos lépésszámban. Mivel tehát legfőljebb $n \cdot m$-szer kell egy legfőljebb $c \cdot m$ futásidejű ciklusmagot végrehajtani, ezért a javítóutas algoritmus teljes lépésszáma legfőljebb $c \cdot n \cdot m^2$ egy alkalmas c konstansra.

A teljesség kedvéért megjegyezzük, hogy léteznek a a fenti lépésszámnál is jobb teljesítményt nyújtó algoritmusok a maximális folyam feladatra. Egyrészt a javítóutas algoritmus futása is felgyorsítható további ötletekkel és trükkökkel, másrészt egészen más megközelítésen alapuló algoritmusokat is felfedeztek. Ezek azonban jóval komplikáltabbak, ebben a jegyzetben nem foglalkozunk velük.

Megemlítjük még a fenti tétel egy elméleti szempontból fontos következményét is. Korábban, a maximális folyam feladat kitűzése kapcsán mondtuk, hogy egyáltalán nem magától értetődő, hogy maximális értékű folyam minden hálózatban létezik. A 9.13. Tételből viszont ez is következik; sőt, ehhez a tétel állításából annyit is elég felhasználni, hogy az algoritmus véges sok javító lépés után megáll. Valóban: az algoritmus leállása utáni folyamról a 9.12. Tétel szerint állíthatjuk, hogy az maximális.

A javítóutas algoritmus optimalitása és végessége a következő tétel bizonyításában is kulcsszerepet kap. Ehhez térjünk vissza a folyamértékek és a vágások kapacitásának a kapcsolatára. A 9.11. Állítást a következőképpen is megfogalmazhatjuk:

$$\max\{m_f : f \text{ folyam}\} \leq \min\{c(X) : X \text{ } st\text{-vágás}\};$$

valóban, ha $m_f \leq c(X)$ igaz bármely f folyamra és X vágásra, akkor nyilván igaz egy olyan f-re, illetve X-re is, amelyekre m_f maximális, illetve $c(X)$ minimális.

9.14. Feladat. Adjunk meg a 9.1. ábra hálózatában egy minimális kapacitású st-vágást.

Megoldás: A 9.12. Tétel után már láttuk, hogy az $X = \{s, p, q, u\}$ vágás kapacitása 10. Ebből tehát $\min\{c(X) : X \text{ } st\text{-vágás}\} \leq 10$ adódik. A 9.5. ábrán látott 10 értékű folyam létezéséből pedig $10 \leq \max\{m_f : f \text{ folyam}\}$ következik. Ezekből és a fenti, a 9.11. Állításból következő becslésből

$$10 \leq \max\{m_f : f \text{ folyam}\} \leq \min\{c(X) : X \text{ } st\text{-vágás}\} \leq 10$$

adódik, amiből következik, hogy $\min\{c(X) : X \text{ } st\text{-vágás}\} = 10$, vagyis X minimális kapacitású vágás. (Ugyanebből az is következik, hogy a 9.5. ábrán látható 10 értékű folyam maximális – amit már a 9.12. Tételből is tudtunk, hiszen láttuk, hogy f-re nézve nincs javítóút). □

A fenti megoldásból tehát kiderült, hogy a 9.1. ábra hálózatában a maximális folyam értéke megegyezik az st-vágások kapacitásának minimumával: mindkettő 10. Az alábbi, a folyamok elméletében alapvető fontosságú tétel azt állítja, hogy ez nem csak erre, hanem minden hálózatra igaz.

9.15. Tétel. (Ford-Fulkerson tétel)
Bármely (G, s, t, c) hálózatra

$$\max\{m_f : f \text{ folyam}\} = \min\{c(X) : X \text{ } st\text{-vágás}\},$$

vagyis a hálózatbeli maximális folyam értéke megegyezik az st-vágások kapacitásának minimumával.

Bizonyítás: Futtassuk a (G, s, t, c) hálózatban a javítóutas algoritmust (például az azonosan nulla folyamból indítva) leállásig – ami a 9.13. Tétel szerint véges sok javító lépés után bekövetkezik; a kapott folyamot jelölje f. A 9.12. Tétel bizonyításában láttuk, hogy ekkor a H_f-ben s-ből irányított úton elérhető csúcsok alkotta X vágásra $m_f = c(X)$ teljesül; jelölje ezt a közös értéket d. Ekkor a d értékű folyam létezéséből $d \leq \max\{m_f : f \text{ folyam}\}$, a d kapacitású vágás létezéséből pedig $\min\{c(X) : X \text{ } st\text{-vágás}\} \leq d$ következik. Összevetve ezt a fenti, a 9.11. Állításból következő becsléssel:

$$d \leq \max\{m_f : f \text{ folyam}\} \leq \min\{c(X) : X \text{ } st\text{-vágás}\} \leq d.$$

Ebből tehát $\max\{m_f : f \text{ folyam}\} = d = \min\{c(X) : X \text{ } st\text{-vágás}\}$ valóban adódik. □

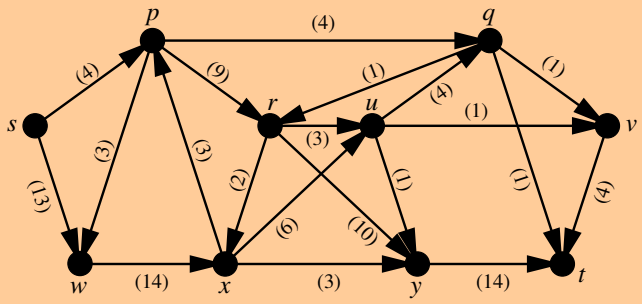
Fontos hangsúlyozni, hogy a fenti bizonyítás támaszkodik az (itt bizonyítás nélkül közölt) 9.13. Tételre – illetve abból annyit használ fel, hogy a javítóutas algoritmus véges sok lépés után megáll.

Később látni fogjuk (lásd a 9.17. Lemma utáni apróbetűs részt), hogy ha minden él kapacitása racionális szám, akkor az algoritmus végessége egyszerűbben is indokolható; így ezekben az esetekben a 9.13. Tételre való hivatkozás is kiváltható a fenti tétel bizonyításában.

A 9.15. Tétel teljes értékű, minden esetre kiterjedő bizonyításához valójában az algoritmus végességénél kevesebbet is elég volna belátni: azt, hogy maximális értékű folyam minden hálózatban létezik. Korábban már említettük, hogy egyrészt ez messze nem nyilvánvaló, másrészt a 9.13. Tételből következik. Ha azonban ezt máshogyan (például analízisbeli eszközökkel) igazolnánk, akkor ebből már valóban következne a 9.15. Tétel: mivel egy f maximális folyamra nézve nem létezhet javítóút (hiszen ekkor volna f-nél nagyobb értékű folyam), ezért az s-ből H_f-beli irányított úton elérhető csúcsok X halmaza (a 9.12. Tétel bizonyítása szerint) m_f-fel azonos kapacitású vágás, amiből a fenti bizonyítás gondolatmenete már akadálytalanul működik.

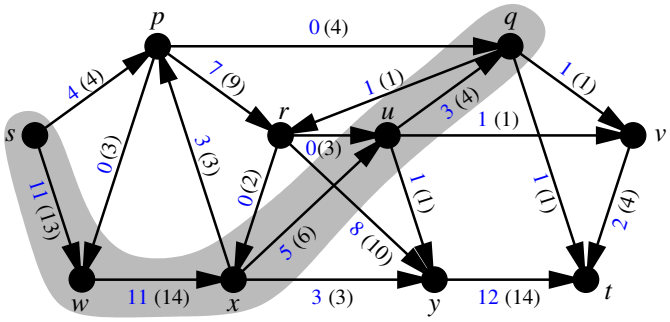
Érdemes kiemelni a 9.15. Tétel bizonyításának egy alkalmazásokban igen fontos következményét: a gondolatmenetből kiderült, hogy a javítóutas algoritmus leállása után az s-ből H_f-beli irányított úton elérhető csúcsok X halmaza minimális kapacitású vágást alkot. Léteznek olyan, a gyakorlat által felvetett problémák, amelyek épp egy ilyen vágás meghatározását igénylik; ezek szerint tehát a javítóutas algoritmus erre is hatékonyan alkalmazható. Erre láttunk példát a 9.14. Feladatban is: az abban szereplő $X = \{s, p, q, u\}$ vágás a 9.6. ábra segédgráfjában az s-ből irányított úton elérhető csúcsok halmaza volt.

9.16. Feladat. Adjunk meg az alábbi hálózatban egy maximális értékű folyamot (s-ből t-be) és egy minimális kapacitású st-vágást.



Megoldás: A 9.8. ábrán (kékkel, a zárójeleken kívül) látható értékek egy 15 értékű folyamot alkotnak; valóban, ez a folyam, illetve a folyamérték 9.2., illetve 9.3. Definíciója alapján könnyen ellenőrizhető.

Az ugyancsak a 9.8. ábrán látható $X = \{s, q, u, w, x\}$ vágás kapacitása is 15; valóban, az X-ből kilépő élek (s,p), (q,r), (q,v), (q,t), (u,v), (u,y), (x,y) és (x,p), ezeknek a kapacitásösszege pedig $4+1+1+1+1+1+3+3 = 15$.



9.8. ábra

A 15 értékű folyam létezéséből $15 \leq \max\{m_f : f \text{ folyam}\}$, a 15 kapacitású vágás létezéséből pedig $\min\{c(X) : X\ st\text{-vágás}\} \leq 15$ következik. Ezekből és a 9.11. Állításból adódó $\max\{m_f : f \text{ folyam}\} \leq \min\{c(X) : X\ st\text{-vágás}\}$ becslésből:

$$15 \leq \max\{m_f : f \text{ folyam}\} \leq \min\{c(X) : X\ st\text{-vágás}\} \leq 15.$$

Ebből tehát $\max\{m_f : f \text{ folyam}\} = 15 = \min\{c(X) : X\ st\text{-vágás}\}$, így a 9.8. ábrán látható folyam maximális értékű, az X vágás pedig minimális kapacitású. □

A fenti megoldáshoz két megjegyzést is fűzünk. Az első, hogy a megoldásban írt becslésben $\max\{m_f : f \text{ folyam}\}$ és $\min\{c(X) : X\ st\text{-vágás}\}$ közé a 9.15. Tétel szerint persze egyenlőségjelet is tehettünk volna; mivel azonban erre itt nem volt szükség, megelégedtünk a 9.11. Állításból adódó egyenlőtlenségjellel.

A második megjegyzésünk, hogy bár a megoldás gondolatmenete hiánytalan, de egyáltalán nem derül ki belőle, hogy a megadott folyam és vágás hogyan keletkezett – pedig ezeknek a megtalálása már egy ekkora feladatban is messze nem magától értetődő. A válasz az, hogy mindkettőt a javítóutas algoritmus futásából kaptuk (a vágás esetében úgy, hogy X-be az eljárás leállása után H_f-ben s-ből elérhető csúcsokat vettük be). Ez a megoldás azt mutatja meg, hogy az algoritmus futásából kapott folyam és vágás önmagában is teljes értékű indoklást tesznek lehetővé, ezek egymás optimalitását bizonyítják. A gondolatmenet persze attól volt működőképes, hogy a megadott folyam, illetve vágás értéke, illetve kapacitása egyenlő; a 9.15. Tétel éppen arra garancia, hogy ilyen folyam és vágás minden hálózatban létezik.

9.3. A folyamprobléma változatai

A fentiekben tárgyalt maximális folyam feladat csak egy a hálózatokról és folyamokról szóló, a gyakorlati élet által felvetett problémák sorában. Az alábbiakban megemlítünk néhány további változatot – elsősorban olyanokat, amik megoldhatók a javítóutas algoritmussal vagy annak egyszerű kiegészítésével.

9.3.1. Egészértékű maximális folyam

Számos gyakorlati alkalmazásban a hálózat által szállított termék nem korlátlanul osztható, hanem csak valamilyen alapegység többszörösei jöhetnek szóba. Miközben például egy csőhálózat vagy számítógép-hálózat által szállított folyadék, illetve információ korlátlanul osztható (legalábbis a feladat modellezhető így), addig egy úthálózaton megoldandó logisztikai feladat esetében nyilván nem volna értelme annak, hogy egy útszakaszon törtmennyiségű kamion haladjon át.

Az egészértékű maximális folyam feladat egyedül abban a lényeges feltételben tér el az eredetitől, hogy minden e élre $f(e)$ értéke egész szám kell legyen. Természetesen ebben az esetben feltételezhető, hogy a feladat bemenetében a $c(e)$ kapacitások is egészek (hiszen az $f(e) \in \mathbb{Z}$ feltevés miatt a törtértékű kapacitások helyettesíthetők volnának az egészrészükkel).

De még a kapacitások egészértékűségét feltételezve is messze nem magától értetődő, hogy a maximális folyam megvalósítható-e csupa egész értékekkel? Más szóval: nem fordulhat-e elő, hogy $\max\{m_f : f \text{ folyam}\} = k$, de olyan k értékű folyam már nem létezik, amire $f(e) \in \mathbb{Z}$ minden e élen igaz volna? Az alábbi lemma azt mondja, hogy erre a kérdésre szerencsére nemmel válaszolhatunk.

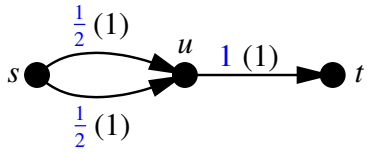
9.17. Lemma. (Egészértékűségi lemma)
Tegyük fel, hogy a (G,s,t,c) hálózatban minden $e \in E(G)$ élre $c(e) \in \mathbb{Z}$. Ekkor
(i) létezik olyan f maximális folyam a hálózatban, amelyre $f(e) \in \mathbb{Z}$ minden $e \in E(G)$ élre és
(ii) ilyen egészértékű maximális folyam a javítóutas algoritmussal található.

Bizonyítás: Indítsuk a javítóutas algoritmust egy tetszőleges egészértékű f folyamtól, például az azonosan nullától. Ekkor az algoritmus a működése során végig fenntartja f egészértékűségét. Valóban, ha az aktuális f folyamra $f(e) \in \mathbb{Z}$ minden e élre, akkor az algoritmus 9.1. szakaszban látott pszeudokódjának 7. sorában δ_1 és δ_2 számításakor is egész számok minimumát vesszük (hiszen $c(e) - f(e)$ és $f(e)$ is egész minden e élre, mert a $c(e)$-k egészértékűségét feltettük), így δ_1 és δ_2 is, ebből következően pedig δ is egész. Így a 8. sorban végrehajtott módosítások után is igaz marad $f(e) \in \mathbb{Z}$ minden e élre.

Következik, hogy f az algoritmus leállásakor is egészértékű; azt pedig a 9.12. Tételből tudjuk, hogy ekkor f maximális folyam. □

Az Egészértékűségi lemma az egyszerűsége dacára egyrészt alkalmazásokban igen fontos, másrészt abban az értelemben messze nem magától értetődő dolgot állít, hogy a javítóutas algoritmus ismerete nélkül a bizonyítása jóval nehezebb volna.

Felhívjuk rá a figyelmet, hogy a lemma csak azt állítja, hogy a maximális folyamok között *létezik* egészértékű – de azt nem, hogy *mindegyik* ilyen. Ez az állítás ugyanis nem is volna igaz: a 9.9. ábrán látható példában a kapacitások egészek, de a megadott maximális folyam mégsem egészértékű. (Persze könnyű ugyanebben a hálózatban egészértékű maximális folyamot is mutatni: ha az s-ből u-ba vezető két él közül az egyiken 0, a másikon 1 a folyam értéke.)



9.9. ábra

Az Egészértékűségi lemmának közvetlen következménye, hogy ha a kapacitások egészek, akkor a maximális folyamérték is egész; valóban, egy egészértékű f maximális folyamra a 9.3. Definíciót alkalmazva nyilván egész m_f értéket kapunk. Ez a következmény persze a Ford-Fulkerson tételből (9.15. Tétel) is könnyen adódik: a vágások kapacitásának minimuma nyilván egész, hiszen az élek kapacitásának egészértékűsége miatt $c(X)$ minden X vágásra egész (a 9.9. Definíció szerint); így a maximális folyamérték is egész.

Röviden kitérünk a 9.17. Egészértékűségi lemma néhány, elméleti szempontból érdekes vonatkozására.

A lemma egyszerű (és korábban már említett) következménye, hogy ha minden kapacitás egész és a javítóutas algoritmust az azonosan nulla folyamtól (vagy bármilyen más egészértékű folyamtól) indítjuk, akkor az eljárás véges sok lépés után megáll – még akkor is, ha a javítóutakat tetszőlegesen választjuk (és nem feltétlen mindig az egyik legrövidebbet a 9.13. Tétel szerint). Valóban: a lemma bizonyításából kiderült, hogy az $f(e)$ értékek az algoritmus futása során mindig egészek, így az m_f folyamérték is nyilván mindig az. Ezért m_f minden javító lépés során legalább eggyel nő. Így ha a maximális folyamérték M, akkor ezt az algoritmus legföljebb M javítás után el is éri. Sőt: ezzel a gondolatmenettel analóg módon az is belátható, hogy az algoritmus végessége már azt feltételezve is következik, hogy minden él kapacitása racionális szám (az azonosan nullától indítva, a javítóutak tetszőleges választása mellett). Ha ugyanis d jelöli a kapacitások nevezőinek (amik egész számok) a legkisebb közös többszörösét, akkor az algoritmus során keletkező minden számadat $\frac{1}{d}$-nek többszöröse, így a folyamérték minden javító lépés során legalább $\frac{1}{d}$-vel nő; ha tehát továbbra is M jelöli a maximális folyamértéket, akkor az eljárás legföljebb $d \cdot M$ javítás után megáll.

Fontos azonban hangsúlyozni, hogy ezen a módon csak az algoritmus végességét tudjuk megindokolni, a polinomiális futásidőt már nem. Valóban: ha a 9.7. ábrán látott példában a négy 10^6 értékű kapacitást egy tetszőleges (pozitív egész) k-ra cseréljük, akkor a javítóutak ott látott módon való választása esetén az algoritmus csak $2k$ javítás után áll meg. Márpedig a számelméleti algoritmusok futásidejének vizsgálata kapcsán szerzett korábbi tapasztalatokból tudjuk, hogy a bemenet részét képező számok értékével arányos lépésszám exponenciális futásidőt jelent. (Ha pedig a kapacitásokat az 1.7. szakaszban írtaknak megfelelően egy fix konstansnyi helyen tároljuk, az még tovább rontja a futásidőre vonatkozó becslést, hiszen kisebb méretű bemenethez változatlan lépésszám társul.)

Az Egészértékűségi lemma azt is lehetővé teszi, hogy felhívjuk a figyelmet a jegyzetben eddig tárgyalt két javítóutas algoritmus (vagyis a 7.2. szakaszban bemutatott, páros gráfokban maximális párosítást kereső eljárás és az ebben a fejezetben látott, maximális folyamot kereső algoritmus) közötti kapcsolatra. Mivel a két el-

járás látszólag egészen más feladatot old meg, ezért azt hihetnénk, hogy a névrokonságon túl nincs köztük szorosabb kapcsolat. A valóság ezzel szemben az, hogy az utóbbi, a maximális folyam problémára vonatkozó algoritmus a 7.2. szakaszban látott eljárás általánosításának is tekinthető.

Legyen ugyanis adott a $G = (A, B; E)$ páros gráf, amiben maximális párosítást kell találnunk. Alakítsuk át G-t irányított gráffá úgy, hogy mindegyik élét az A-beli végpontja felől a B-beli felé irányítjuk. Vegyünk fel ezen kívül két új csúcsot, s-et és t-t; majd s-ből vezessünk egy-egy irányított élt A minden csúcsába és hasonlóan, B minden csúcsából vezessünk egy-egy irányított élt t-be. Végül a kapott irányított gráfban minden él kapacitását definiáljuk 1-nek. Nem nehéz végiggondolni, hogy a kapott hálózat egészértékű folyamai (amik a kapacitások választása miatt minden élen csak 0 vagy 1 értéket vehetnek fel) megfelelnek az eredeti páros gráf párosításainak és viszont. Sőt: egy egészértékű f folyamot feltételezve a 9.5. Definíció szerinti javítóutak megfelelnek a 7.13. Definíció szerinti javítóutaknak. Ebből következően pedig a maximális (egészértékű) folyamot kereső javítóutas eljárás lényegében a 7.2. szakaszban látott javítóutas algoritmussá specializálódik.

9.3.2. Több termelő és fogyasztó

A maximális folyam feladatban s-ből t-be, vagyis egyetlen termelőtől egyetlen fogyasztóig kellett a lehető legtöbb terméket eljuttatni. Lehetnek azonban olyan alkalmazások is, ahol a termék a hálózat több pontján is keletkezhet és több célállomása is van – de a termelők is és a fogyasztók is egyenértékűek (vagyis teljesen mindegy, hogy egy adott termelő pontban keletkező termék melyik fogyasztóhoz jut el).

A feladat pontos kitűzése tehát abban különbözik az eredetitől, hogy s és t helyett a termelők $s_1, s_2, \ldots, s_k$, valamint a fogyasztók $t_1, t_2, \ldots, t_\ell$ listája adott (amelyek mind a G irányított gráf csúcsai). A folyam definíciója úgy módosul, hogy a folyammegmaradási feltételeket az s_i-ktől és t_j-ktől különböző csúcsokra követeljük meg és a maximalizálandó folyamérték $\sum_{i=1}^{k} \left(\sum_{e:\, s_i \bullet\!\!\rightarrow} f(e) - \sum_{e:\, s_i \bullet\!\!\leftarrow} f(e) \right)$ – ami tehát az s_i-ket összesen elhagyó nettó termékmennyiség. (Természetesen ebben az esetben is megmutatható, hogy ez egyenlő a t_j-kbe érkező összes nettó termékmennyiséggel.)

Ez a feladat szerencsére egy egyszerű fogással visszavezethető az eredetire (és így a javítóutas algoritmussal megoldhatóvá válik). Vegyünk fel ugyanis a hálózathoz két új csúcsot, jelölje ezeket S és T; majd S-ből vezessünk egy-egy új élt minden s_i termelőhöz és hasonlóan, T-be vezessünk egy-egy új élt minden t_j fogyasztóból. Persze az S-ből induló és a T-be érkező éleknek is kell valamilyen kapacitást adnunk, de a célunk az, hogy ez gyakorlatilag végtelen legyen – vagyis ezeken az e éleken nem szeretnénk $f(e)$-t felülről korlátozni. A javítóutas algoritmus működése könnyen módosítható úgy, hogy kezelni tudjon végtelen kapacitású éleket – de ezt kiválthatjuk azzal is, ha olyan nagy (véges) kapacitást adunk ezeknek az éleknek, hogy az már ne jelentsen valódi korlátozást. Például egy S-ből s_i-be vezető él kapacitásának választhatjuk az s_i-ből kilépő élek kapacitásának az összegét (és a T-be vezető élekre hasonlóan, az él kezdőpontjába érkező élek összkapacitását).

Könnyen végiggondolható, hogy ha az így módosított hálózatban S-ből T-be

keresünk egy maximális f folyamot, akkor ezzel (az S-re és T-re illeszkedő élek figyelmen kívül hagyása után) a több termelős és fogyasztós folyamfeladat optimális megoldását kapjuk. Valóban, az S-sel és T-vel kibővített hálózatban az s_i-kre vonatkozó folyammegmaradási feltételeket összeadva azt kapjuk, hogy a folyam értéke egyenlő az eredeti hálózatban az s_i-ket összesen elhagyó nettó termékmennyiséggel (a fentebb írt képlet értelmében). Az pedig a folyam definíciójából azonnal adódik, hogy az S-re és T-re illeszkedő élek elhagyása után az eredeti hálózatban minden s_i-től és t_j-től különböző csúcsban teljesül a folyammegmaradás, valamint minden e élre $0 \leq f(e) \leq c(e)$.

9.3.3. Irányítatlan élek

Szintén természetes gondolat, hogy egyes alkalmazásokban a hálózat irányítatlan éleket is tartalmazhat, amiken tetszőleges irányban haladhat a termék. Például egy csőhálózat csövein nyilván mindkét irányban folyhat folyadék és az úthálózatokban is vannak kétirányú utcák.

Ebben a feladatban tehát a bemenet, vagyis a hálózat definíciója úgy módosul, hogy a G gráf vegyesen tartalmazhat irányított és irányítatlan éleket is (de persze akár mindegyik éle is lehet irányítatlan). Ettől eltekintve azonban nincs változás: továbbra is adott az s termelő és a t fogyasztó, valamint minden e élnek a $c(e)$ kapacitása. A kimenet, vagyis a folyam definíciója pedig úgy változik, hogy nem csak az $f(e)$ értékeket kell megadni (amelyekre persze $0 \leq f(e) \leq c(e)$ változatlanul követelmény), hanem G minden $\{u,v\}$ irányítatlan élének meg kell adni egy irányítását – vagyis meg kell mondani, hogy azon (u,v) vagy (v,u) irányban halad a termék. A folyammegmaradási feltételeknek pedig ezeket az irányításokat figyelembe véve kell teljesülniük (és a folyam értékének a kiszámításában is szerepet játszik az s-re illeszkedő, eredetileg irányítatlan éleknek a kimenet részeként megadott irányítása).

Hasonlóan a több termelő és fogyasztó esetéhez, a maximális folyam feladatnak ez az általánosítása is könnyen visszavezethető az eredetire és így ez is hatékonyan megoldható a javítóutas algoritmussal. A visszavezetés ötlete nagyon természetes: G minden $e = \{u,v\}$ irányítatlan élét helyettesítjük az $e_1 = (u,v)$ és az $e_2 = (v,u)$ irányított élekkel és $c(e_1)$ és $c(e_2)$ értékét is $c(e)$-nek definiáljuk.

Tegyük fel, hogy az így kapott (csak irányított éleket tartalmazó) hálózatban meghatároztunk egy f maximális folyamot (például a javítóutas algoritmussal). Ekkor f-ből még nem feltétlen olvasható ki közvetlenül az eredeti feladat egy megoldása: előfordulhat ugyanis, hogy az e irányítatlan élből készített mindkét irányított élen pozitív a folyam értéke; márpedig e-n nyilván nem folyhat egyszerre mindkét irányba termék (hanem a fentiek szerint a megoldásban ki kell jelölni, hogy melyik irányba folyik). Ez a probléma azonban könnyen orvosolható: ha $f(e_1) > 0$ és $f(e_2) > 0$ az $e = \{u,v\}$ irányítatlan élből készült $e_1 = (u,v)$ és $e_2 = (v,u)$ irányított élekre, akkor a $\delta = \min\{f(e_1), f(e_2)\}$ értékkel csökkentsük $f(e_1)$ és $f(e_2)$ értékét is. (Vagyis ha például $f(e_1) \geq f(e_2) > 0$, akkor $f(e_1)$ új értéke $f(e_1) - f(e_2)$, $f(e_2)$ új értéke pedig 0 lesz.) Ezután a változtatás után f nyilván továbbra is folyam lesz (hiszen u-ban és v-ben is δ-val csökkent a belépő és a kilépő összmennyiség is, így a folyammegmaradás nem sérült; a kapacitás feltételek meg nyilván igazak

maradtak, hiszen $f(e_1)$ és $f(e_2)$ is csökkent), az m_f folyamérték pedig nem változik.

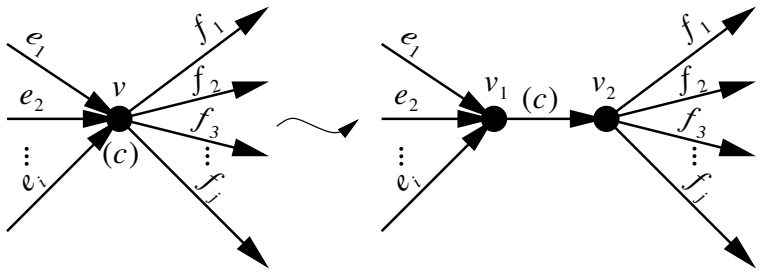
Ha a fenti módosítást elvégezzük minden olyan esetben, amikor az szükséges, akkor végül is egy olyan f maximális folyamot kapunk, amire $f(e_1)$ és $f(e_2)$ közül legalább az egyik nulla minden $e = \{u, v\}$ (eredetileg) irányítatlan él esetében. Ebből az f-ből pedig már könnyen kiolvasható az irányítatlan éleket is tartalmazó folyamprobléma egy optimális megoldása: ha az e irányítatlan élre például $f(e_1) > 0$ és $f(e_2) = 0$, akkor e-t az e_1 szerint irányítjuk és $f(e)$ értékét $f(e_1)$-nek definiáljuk (ha pedig $f(e_1) = f(e_2) = 0$, akkor e irányítása tetszőleges és $f(e) = 0$ lesz).

9.3.4. Pontkapacitás

Aki már közlekedett városban az tudja, hogy a forgalom általában a kereszteződésekben torlódik fel, nem a nyílt útszakaszokon. Egy úthálózat esetében magától értetődő, hogy maguknak a csomópontoknak is korlátos az áteresztőképessége, nem csak az ezek közötti kapcsolatoknak. Ugyanez persze másfajta hálózatok esetében is könnyen felmerülhet.

Módosítsuk tehát úgy a maximális folyam feladat bemenetét, hogy nem csak az éleknek, hanem bizonyos (s-től és t-től különböző) v csúcsoknak (akár mindegyiknek) is adott egy nemnegatív $c(v)$ kapacitása. A folyam definíciója pedig úgy módosul, hogy minden ilyen v csúcsra a $\sum_{e:\, v\bullet\rightarrow} f(e)$ és $\sum_{e:\, v\bullet\leftarrow} f(e)$ értékeknek nem csak egyenlőnek kell lenniük, hanem mindkettő legfőljebb $c(v)$ lehet.

A feladatnak ez a változata ismét visszavezethető az eredetire egy ravasz ötlettel. Ha a v csúcsnak adott a $c(v)$ kapacitása, akkor v-t a 9.10. ábrán látható módon „széthúzzuk” egy éllé. Vagyis v helyett felvesszük a v_1 és v_2 csúcsokat és minden, eredetileg a v-be érkező él végpontja v_1, a v-ből induló élek kezdőpontja pedig v_2 lesz. (Olyan ez, mintha v-t kettévágnánk a v_1 „érkezési oldalra” és a v_2 „indulási oldalra”.) Majd v_1-ből v_2-be felveszünk egy új élt, aminek a kapacitása $c(v)$ lesz.

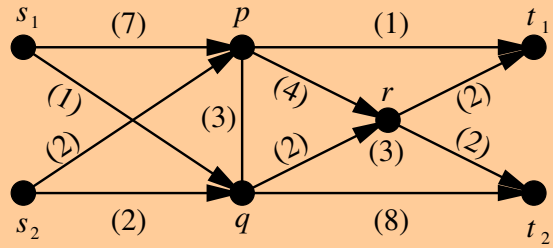


9.10. ábra

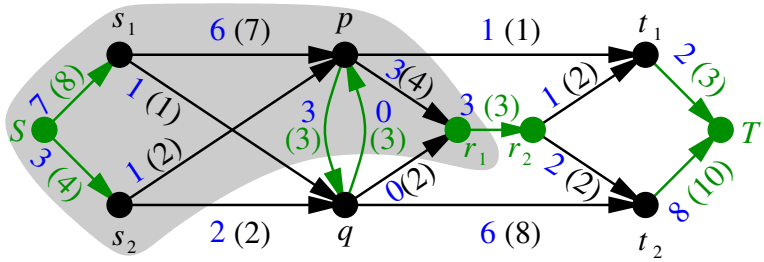
Az így kapott (már az eredeti definíciónak megfelelő) maximális folyam feladatot megoldva, majd a csúcsokból keletkezett éleken a folyamértékeket figyelmen kívül hagyva a pontkapacitásos feladat optimális megoldását kapjuk. Valóban: ha a v csúcsból keletkezett (v_1, v_2) élen a folyamérték $f\big((v_1, v_2)\big) = d$, akkor a v_1-re és v_2-re fennálló folyammegmaradási feltételek miatt $\sum_{e:\, v_1\bullet\leftarrow} f(e)$ és $\sum_{e:\, v_2\bullet\rightarrow} f(e)$ értéke

is d; így az eredeti hálózatban v-re szintén teljesül a folyammegmaradás, valamint a v-be belépő és a v-ből kilépő összefolyamérték is $d \le c\big((v_1,v_2)\big) = c(v)$.

9.18. Feladat. Adjunk meg egy maximális folyamot az alábbi ábrán látható hálózatban, amiben két termelő (s_1 és s_2) és két fogyasztó (t_1 és t_2) van, a $\{p,q\}$ él irányítatlan, az r csúcsnak pedig adott a $c(r) = 3$ kapacitása.



Megoldás: A fentiekben látott visszavezetéseket alkalmazva a 9.11. ábrán látható hálózatot kapjuk (a módosult részeket zölddel ábrázoltuk). Részletesebben: felvettük az S termelőt és ebből s_1-be és s_2-be vezettünk egy-egy élt; ezek kapacitásául pedig az s_1-ből, illetve s_2-ből kilépő élek összkapacitását választottuk. Analóg módon jártunk el a fogyasztók esetében is. Továbbá a $\{p,q\}$ élt helyettesítettük a (p,q) és (q,p) irányított élekkel, amelyeknek a kapacitása $c(\{p,q\}) = 3$ lett. Végül az r csúcsot széthúztuk az (r_1,r_2) éllé, aminek a kapacitása $c(r) = 3$ lett.

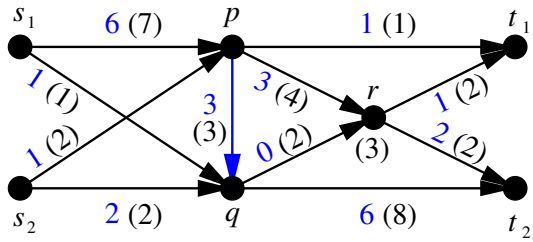


9.11. ábra

A kapott hálózatban szintén a 9.11. ábrán látható egy 10 értékű f folyam és az $X = \{S, s_1, s_2, p, r_1\}$ vágás, aminek a kapacitása könnyen ellenőrizhetően ugyancsak 10. Természetesen f-et és X-et is a javítóutas algoritmus futtatásával kaptuk, de ennek a részleteit mellőzve is következik a 9.16. Feladat megoldásában látott indoklás megismétlésével, hogy f maximális folyam, mert $m_f = c(X)$ itt is teljesül.

f-ből pedig az eredeti hálózatban a 9.12. ábrán látható maximális folyam olvasható ki.

Itt tehát az S-re és T-re illeszkedő éleket és (r_1,r_2)-t elhagytuk, továbbá a $\{p,q\}$ élt p-ből q-ba irányítottuk. (Itt nem volt szükség $f\big((p,q)\big)$ és $f\big((q,p)\big)$ közös δ értékkel való csökkentésére a fentebb írtak szerint, mert $f\big((q,p)\big)$ eleve nulla volt.)



9.12. ábra

A folyamérték (vagyis az s_1-et és s_2-t együttesen elhagyó nettó termékmennyiség) persze szintén 10. □

9.3.5. További folyamproblémák

Amint az a fenti feladatban látható, a maximális folyam probléma eddig említett általánosításai kombinálhatók is. Vannak azonban olyan további, a gyakorlati alkalmazások szempontjából igen fontos, hálózatok folyamaival kapcsolatos feladatok, amik már nem vezethetők vissza a fentiekben látotthoz hasonló, egyszerű trükkökkel a maximális folyam feladatra, hanem egészen más megközelítést igényelnek és akár sokkal nehezebbek is lehetnek. Ezek közül röviden megemlítünk kettőt, amiknek a részletesebb tárgyalása már messze meghaladná ennek a jegyzetnek a kereteit.

A *minimális költségű folyam* feladatban egy előre adott mennyiségű terméket kell eljuttatni s-ből t-be úgy, hogy a szállítás összköltsége (aminek a kiszámításához további bemenő adatok is szükségesek) a lehető legkisebb legyen. (Eközben természetesen az élek kapacitására ugyanúgy tekintettel kell lenni.) Ennél is nehezebb a *többtermékes folyam* probléma, amiben ugyanazt a hálózatot egyszerre több termék szállítására kell használni úgy, hogy minden terméknek adott a saját termelő és fogyasztó pontja, de az élek terheléséhez (ami a kapacitást sehol nem haladhatja meg) természetesen mindegyik termék hozzájárul. (Ez a feladat tehát lényegesen különbözik a fentebb tárgyalt több termelős és fogyasztós maximális folyam feladattól és sokkal nehezebb annál, mert itt már egyáltalán nem mindegy, hogy egy adott termelő pontban keletkező termék melyik fogyasztóhoz jut el.)

A minimális költségű folyam és a többtermékes folyam feladatok fenti, rövid leírása persze csak egy intuitív képet ad, ezek a fejezetben tárgyalt maximális folyam problémához hasonlóan pontosan definiálhatók. Ez a két feladat is megoldható hatékony, polinomiális algoritmussal, de ezek már jóval komplikáltabbak a fejezetben tárgyalt javítóutas algoritmusnál. A többtermékes folyam probléma egészértékű változata viszont már nagyon nehéz feladat, nem ismert és valószínűleg nem is létezik rá polinomiális algoritmus. Ezekről a kérdésekről az informatikus képzés felsőbbéves tárgyaiban lesz bővebben szó.

10. fejezet

Diszjunkt utak, többszörös összefüggőség

Tegyük fel, hogy egy számítógép-hálózaton egyszerre több, nagy méretű fájlt kell eljuttatnunk az s csomópontból a t-be. Ezért mindegyik fájlhoz kijelölünk egy s-ből t-be vezető utat a hálózatot reprezentáló gráfban. Azonban a linkek túlterhelésének elkerülése érdekében a gráf minden élén legföljebb csak egy ilyen út haladhat át. Ha tehát a fájlok száma k, akkor azt a kérdést kell megválaszolnunk, hogy létezik-e G-ben k darab olyan út s-ből t-be, amelyek közül semelyik kettőnek nincs közös éle; illetve ha a válasz igen, akkor természetesen meg is kell adnunk k ilyen utat.

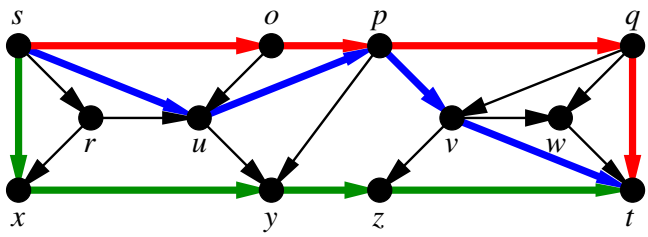
Egy ehhez hasonló, de ennél szigorúbb követelményt támasztó probléma is gyakran felmerül alkalmazásokban: itt azt várjuk el, hogy a keresett, s-ből t-be vezető utak közül semelyik kettőnek ne legyen közös csúcsa — természetesen s-et és t-t kivéve. Ez a feladat például akkor merülhet fel, ha biztonsági okokból szeretnénk elkerülni, hogy az utak közbülső csomópontjain egynél több fájl haladjon át (mert egyetlen fájlnak az illetéktelen kezekbe jutása még nem jelent kockázatot).

A G gráfban az s-ből t-be vezető $P_1, P_2, \ldots, P_k$ utakat *éldiszjunktnak* nevezzük, ha közülük semelyik kettőnek nincs közös éle. Ha pedig $P_1, P_2, \ldots, P_k$ közül semelyik kettőnek nincs s-től és t-től különböző közös csúcsa, akkor *pontdiszjunktnak (vagy csúcsdiszjunktnak)* nevezzük ezeket az utakat.

Megjegyezzük, hogy ezek az elnevezések valójában kicsit pontatlanok, helyesebb volna *páronként éldiszjunkt*, illetve *páronként pontdiszjunkt* utakról beszélni. A „páronként” jelző nélkül ugyanis érthetők úgy is ezek a kifejezések, mintha csak egy olyan él vagy közbülső csúcs létezését akarnánk kizárni, ami mindegyiken rajta van a szóban forgó utak közül. (Ugyanis k halmaz diszjunkt volta valóban csak azt jelenti, hogy nincs olyan elem, ami mindegyikben benne van.) Ehhez képest a fenti, sokkal szigorúbb értelemben használjuk az éldiszjunkt, illetve pontdiszjunkt kifejezéseket: olyan él vagy közbülső csúcs létezését is kizárjuk, ami az utak közül akár csak kettőn rajta van. Ennek ellenére, a „páronként” jelzőt a továbbiakban elhagyjuk, mert nagyon körülményessé tenné a megfogalmazásainkat. Egy másik

pontosítási lehetőség volna a pontdiszjunkt esetben *belsőleg pontdiszjunkt* utakról beszélni ezzel hangsúlyozva, hogy s és t persze az ilyen utaknak is közös csúcsa; a továbbiakban a „belsőleg” jelzőt is elhagyjuk, ez sem fog félreértéshez vezetni.

A 10.1. ábrán látható három, s-ből t-be vezető út éldiszjunkt (hiszen minden él legföljebb egy színt kapott), de nem pontdiszjunkt, hiszen a p csúcson a piros és a kék út is áthalad. A kék és a zöld út viszont például pontdiszjunkt, mert s-en és t-n kívül nem találkoznak.



10.1. ábra

Fentebb már említettük, hogy a pontdiszjunktság erősebb követelmény; vagyis ha a $P_1, P_2, \ldots, P_k$ utak pontdiszjunktak, akkor éldiszjunktak is. Valóban, ha az e élt egynél több út tartalmazná, akkor e-nek volna olyan, s-től és t-től különböző végpontja, amit szintén egynél több út tartalmazna. Az utak éldiszjunktságából viszont nem következik a pontdiszjunktságuk: példa erre a fenti ábra három útja.

Ebben a fejezetben diszjunkt utakkal kapcsolatos kérdéseket vizsgálunk: hatékony algoritmust adunk ilyenek keresésére és a létezésükre vonatkozó fontos elméleti eredményekre jutunk. A diszjunkt utak problémája négy különböző feladatot is jelent: nem csak azt kell eldöntenünk, hogy éldiszjunkt vagy pontdiszjunkt utakat keresünk, hanem azt is, hogy irányított vagy irányítatlan gráfban. Mind a négy változat gyakorlati alkalmazásokban felmerülő, fontos feladat.

10.1. Éldiszjunkt utak

Az éldiszjunkt utak problémája kapcsán a 10.1. ábrát nézve eszünkbe juthatnak a 9. fejezetben tárgyalt folyamok: ha mindhárom színes úton egy-egy egységnyi terméket szállítunk s-ből t-be (vagyis a színes e éleken a folyam értéke $f(e) = 1$, minden más élen pedig $f(e) = 0$), akkor ezzel egy három értékű folyamot kapunk. Az alábbi lemma, amit a fejezetben később többször is alkalmazni fogunk, ezt a gondolatot bontja ki.

10.1. Lemma. *Legyen f olyan, $m_f = d$ értékű folyam a G irányított gráfban az s termelőből a t fogyasztóba, amire $f(e) = 0$ vagy $f(e) = 1$ minden e élre. Ekkor G-ben létezik d olyan éldiszjunkt irányított út s-ből t-be, amiknek minden e élére $f(e) = 1$.*

Bizonyítás: Ha $d = 0$, akkor a lemma állítása magától értetődő, így a továbbiakban feltesszük, hogy $d \geq 1$. Jelölje G_f azt a gráfot, ami G összes csúcsából és azokból az e éleiből áll, amikre $f(e) = 1$. Azt kell megmutatnunk, hogy G_f-ben létezik d éldiszjunkt irányított út s-ből t-be.

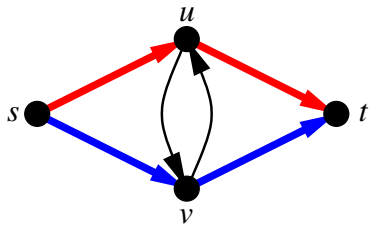
Ehhez először azt mutatjuk meg, hogy G_f-ben létezik irányított út s-ből t-be. s-ből indulva haladjunk „vaktában" G_f élei mentén csak arra vigyázva, hogy egy élen csak egyszer menjünk át; állítjuk, hogy az így kapott Q élsorozat t-ben akad el. Valóban: mivel f folyam, ezért minden $v \neq s,t$ csúcsra a v-be belépő és a v-ből kilépő G_f-beli élek száma egyenlő, s-ből pedig d-vel több G_f-beli él lép ki, mint ahány belép. Így valahányszor Q egy t-től különböző csúcsba belép, tovább is tud onnan lépni. Ezért Q valóban csak t-ben akadhat el. Persze Q nem feltétlen út, hiszen egy csúcsot többször is érinthet, de Q létezéséből egy s-ből t-be vezető irányított út létezése is következik: hasonlóan az (irányítatlan gráfokra vonatkozó) 1.11. Állítás bizonyításához, Q-ból az ismétlődő csúcsok előfordulásai közötti részeket sorra kivágva irányított úthoz jutunk.

Válasszunk ezért G_f-ben egy P_1 irányított utat s-ből t-be; majd P_1 élein változtassuk meg a folyam értékét $f(e) = 0$-ra. Ekkor f továbbra is folyam (hiszen P_1 s-től és t-től különböző csúcsain $\sum_{e:\, v\bullet\leftarrow} f(e)$ és $\sum_{e:\, v\bullet\rightarrow} f(e)$ is eggyel csökkent) és az értéke $m_f = d - 1$ (hiszen $\sum_{e:\, s\bullet\rightarrow} f(e)$ eggyel csökkent, $\sum_{e:\, s\bullet\leftarrow} f(e)$ pedig nem változott). Ha most $d - 1 \geq 0$, akkor újra alkalmazhatjuk az előző bekezdés állítását: választhatunk egy olyan P_2 irányított utat s-ből t-be, aminek minden élére $f(e) = 1$. Ezután P_2 minden e élén is megváltoztatjuk a folyam értékét $f(e) = 0$-ra, amivel $m_f = d - 2$ lesz, stb. Ezt az eljárást addig folytathatjuk, amíg végül $m_f = 0$ lesz és megkapjuk a $P_1, P_2, \ldots, P_d$ irányított utakat s-ből t-be. Mivel ezek az utak nyilván éldiszjunktak (hiszen egy-egy út kiválasztása után annak az e élein $f(e)$-t nullára állítottuk, így a későbbi utak ezeket már nem tartalmazhatták), ezért a lemmát beláttuk. □

Fontos hangsúlyozni, hogy a fenti bizonyításból hatékony algoritmus is következik a lemma által garantált éldiszjunkt utak előállítására. Ehhez az eljárás során az s-ből t-be vezető utakat kereshetjük például a BFS algoritmussal (persze mindig csak az azokból az élekből álló G_f gráfon futtatva a BFS eljárást, amikre $f(e) = 1$ az aktuális f folyam szerint).

Érdemes a fenti bizonyításhoz hozzáfűzni azt is, hogy ugyan a $P_1, P_2, \ldots, P_d$ utak mind G_f-beliek (ahol most f még az eredeti, d értékű folyamot jelöli), de az már nem feltétlen igaz, hogy G_f minden éle rajta van valamelyik úton. Más szóval: a $P_1, P_2, \ldots, P_d$ utak élhalmazainak az egyesítése nem feltétlen $E(G_f)$, hanem csak annyi igaz, hogy része annak. Ha például G_f a 10.2. ábrán látható gráf (ami lehetséges, mert ha minden e élére $f(e) = 1$, akkor ezzel $m_f = 2$ értékű folyamot kapunk s-ből t-be), akkor a bizonyításban írt eljárással kaphatjuk például a piros és a kék éldiszjunkt utakat; azonban az (u,v) és a (v,u) élek egyikben sincsenek benne. (Lásd még erről a kérdésről a 9.3. Definíció utáni apróbetűs részt is.)

A 10.1. ábra gráfjában láttuk, hogy s-ből t-be található három éldiszjunkt út; de vajon található-e négy? Egy egyszerű, de szellemes indoklással meggyőződhetünk róla, hogy a válasz nemleges: minden s-ből t-be vezető út tartalmaz egyet az (o,p),



10.2. ábra

(u,p) és (y,z) élek közül. Valóban, ha ezt a három élt kihagynánk a gráfból, akkor a kapott gráfban már nyilván nem volna s-ből t-be vezető út (hiszen nem volna az $\{s,o,r,u,x,y\}$ csúcshalmazból kivezető él), így tényleg nem lehet s-ből t-be eljutni anélkül, hogy e közül a három él közül legalább egyet ne érintenénk. Ebből következik, hogy bárhogyan is választunk s-ből t-be négy utat, az (o,p), (u,p) és (y,z) élek között lesz olyan, ami ezek közül kettőben is benne van, így ezek az utak biztosan nem lesznek éldiszjunktak. Az alábbi definíció az ebben a gondolatmenetben alkalmazott kulcsfogalmat vezeti be.

10.2. Definíció. *A G irányított gráfban a $Z \subseteq E(G)$ élhalmaz* lefogja az s-ből t-be vezető irányított utakat *(ahol $s,t \in V(G)$ a gráf két különböző csúcsa), ha minden s-ből t-be vezető, G-beli irányított út tartalmaz Z-beli élt. Hasonlóan értelmezzük ezt a fogalmat irányítatlan G gráfokra is: a $Z \subseteq E(G)$ élhalmaz* lefogja az s-ből t-be vezető irányítatlan utakat*, ha minden s és t végpontú, G-beli irányítatlan út tartalmaz Z-beli élt.*

A fenti gondolatmenet tehát azon alapult, hogy a $Z = \{(o,p),(u,p),(y,z)\}$ élhalmaz lefogja az s-ből t-be vezető irányított utakat a 10.1. ábrán látható ábra gráfjában. (Figyelem: ez a mondat *nem azt jelenti*, hogy Z a három, színessel kiemelt út mindegyikéből tartalmaz egy-egy élt, hanem ennél sokkal többet állít; azt, hogy Z *minden*, az s-ből t-be vezető irányított útból tartalmaz legalább egy élt.) A definíció fogalmazható úgy is, hogy Z akkor fogja le az s-ből t-be vezető (irányított, illetve irányítatlan) utakat, ha a Z elhagyása után kapott gráfban már nincs s-ből t-be (irányított, illetve irányítatlan) út.

Felhívjuk rá a figyelmet, hogy az elnevezések felületes hasonlósága ellenére a most bevezetett fogalomnak semmi köze sincs a 7.7. Definícióban értelmezett lefogó élhalmazhoz; a kettőt tehát nem szabad összekeverni.

Az alábbi tétel összeköti az éldiszjunkt utak problémájával kapcsolatban eddig látott két gondolatot: az s-ből t-be vezető utakat lefogó élhalmaz fogalmát, illetve a folyamokat; ezzel egyben meg is oldja ezt a problémát. A tétel három állítás ekvivalens voltát mondja ki; ez azt jelenti, hogy ha az állítások közül bármelyik igaz, akkor a másik kettő is az.

10.3. Tétel. *Legyen adott a G irányított gráf, annak az $s,t \in V(G)$ különböző csúcsai és a $k \geq 1$ egész. Ekkor az alábbi állítások ekvivalensek:*

(i) Létezik G-ben k éldiszjunkt irányított út s-ből t-be.

(ii) Nem létezik G-ben legföljebb $k-1$ élű, az s-ből t-be vezető irányított utakat lefogó élhalmaz.

(iii) A (G,s,t,c) hálózatban a maximális folyam értéke legalább k, ahol c az azonosan 1 kapacitásfüggvény.

Bizonyítás: A tételt úgy látjuk be, hogy igazoljuk az *(i)*⇒*(ii)*, a *(ii)*⇒*(iii)*, valamint a *(iii)*⇒*(i)* következtetések helyességét. Ezzel a bizonyítás valóban teljes lesz, hiszen bármelyik állításból helyes következtetések mentén eljuthatunk a másik kettőbe.

Tegyük fel ezért először, hogy *(i)* igaz: léteznek az s-ből t-be vezető $P_1, P_2, \ldots, P_k$ éldiszjunkt irányított utak. Legyen Z tetszőleges, az s-ből t-be vezető irányított utakat lefogó élhalmaz. Mivel Z minden s-ből t-be vezető útból tartalmaz élt, ez igaz a $P_1, P_2, \ldots, P_k$ utakra is; válasszunk ezért minden $1 \leq i \leq k$ esetén P_i-ből egy e_i élt, amire $e_i \in Z$. Ekkor az $e_1, e_2, \ldots, e_k$ élek mind különbözők, mert a P_i utak éldiszjunktak. Így Z tartalmaz k különböző élt, ezért $|Z| \geq k$, amivel *(ii)*-t beláttuk.

Most azt tegyük fel, hogy *(ii)* teljesül és legyen $X \subseteq V(G)$ tetszőleges vágás a (G,s,t,c) hálózatban. Ekkor az X-ből kilépő élek nyilván lefogják az s-ből t-be vezető irányított utakat, hiszen $s \in X$ és $t \notin X$ miatt minden s-ből t-be vezető útnak tartalmaznia kell X-ből kilépő élt. *(ii)*-ből tehát következik, hogy X-ből legalább k él lép ki. Ez fogalmazható úgy is, hogy $c(X) \geq k$, mert a 9.9. Definíció szerint $c(X)$ épp az X-ből kilépő élek száma (hiszen $c(e) = 1$ minden e élre). Mivel tehát minden vágás kapacitása legalább k, ezért $\min\{c(X) : X \text{ } st\text{-vágás}\} \geq k$. Így a 9.15. Ford-Fulkerson tételből $\max\{m_f : f \text{ folyam}\} \geq k$ következik, amivel *(iii)*-at beláttuk.

Végül azt tegyük fel, hogy *(iii)* igaz és legyen $\max\{m_f : f \text{ folyam}\} = d$; ekkor tehát *(iii)* szerint $d \geq k$. Mivel c egészértékű, ezért a 9.17. Lemmából következően létezik olyan f egészértékű folyam, amire $m_f = d$; rögzítsünk le egy ilyen f-et. Ekkor tehát minden e élre $f(e) = 0$ vagy $f(e) = 1$ (ismét azért, mert $c(e) = 1$ minden e élre). Így $d \geq k$ miatt *(i)* valóban következik a 10.1. Lemmából. □

Ismét fontos hangsúlyozni, hogy a fenti bizonyításból hatékony algoritmus is következik k darab, s-ből t-be vezető éldiszjunkt út létezésének az eldöntésére, illetve ilyenek keresésére. Ehhez tehát a $c \equiv 1$ kapacitásfüggvényre meg kell határoznunk egy egészértékű f maximális folyamot s-ből t-be (például a 9. fejezetben látott javítóutas algoritmussal); ha erre $m_f \geq k$, akkor a 10.1. Lemma bizonyításában írt eljárással kaphatunk k éldiszjunkt utat. Ha viszont $m_f < k$, akkor a 10.3. Tétel szerint nem léteznek ilyen utak; sőt, ebben az esetben egy ezt a tényt a tétel *(ii)* állításának megfelelően igazoló élhalmazt is meg tudunk adni: egy k-nál kisebb kapacitású X vágásból kilépő élek halmaza ilyen. (X-et pedig a 9.15. Tétel bizonyítása, illetve az utána írt megjegyzés szerint szintén a javítóutas algoritmussal kaphatjuk meg.)

A 10.3. Tétel tehát megoldja az s-ből t-be vezető éldiszjunkt utak problémáját abban az esetben, ha G irányított gráf. Az alábbi tételből az derül ki, hogy nagyon hasonlóan kezelhető a feladat irányítatlan gráfokra is; ehhez csak azt a 9.3. szakasz-

ban már látott ötletet kell alkalmazni, hogy minden irányítatlan élt helyettesítünk két irányítottal.

10.4. Tétel. *Legyen adott a G irányítatlan gráf, annak az $s,t \in V(G)$ különböző csúcsai és a $k \geq 1$ egész. Ekkor az alábbi állítások ekvivalensek:*

(i) Létezik G-ben k éldiszjunkt irányítatlan út s-ből t-be.

(ii) Nem létezik G-ben legföljebb $k-1$ élű, az s-ből t-be vezető irányítatlan utakat lefogó élhalmaz.

(iii) A (H,s,t,c) hálózatban a maximális folyam értéke legalább k, ahol a H irányított gráf úgy készül G-ből, hogy annak minden $\{u,v\}$ élét helyettesítjük az (u,v) és (v,u) irányított élekkel; c pedig az azonosan 1 kapacitásfüggvény.

Bizonyítás: Ismét az *(i)*⇒*(ii)*, *(ii)*⇒*(iii)* és *(iii)*⇒*(i)* következtetéseket látjuk be.

Az *(i)*⇒*(ii)* bizonyítása azonos a 10.3. Tétel bizonyításában írtakkal; az egyetlen különbség az, hogy a $P_1,P_2,\ldots,P_k$ utak most irányítatlanok.

A *(ii)*⇒*(iii)* bizonyításában sincs sok változás a 10.3. Tételhez képest: ha $X \subseteq V(H)$ a (H,s,t,c) hálózat tetszőleges vágása, akkor $c(X) \geq k$ továbbra is igaz, különben az X-ből kilépő éleknek megfelelő G-beli irányítatlan élek egy k-nál kisebb méretű, az s-ből t-be vezető utakat lefogó élhalmazt alkotnának. Így a 9.15. Tételből ismét következik, hogy $\max\{m_f : f \text{ folyam}\} \geq k$.

A *(iii)*⇒*(i)* bizonyítását megint kezdjük azzal, hogy a 9.17. Lemmára hivatkozással választunk egy f egészértékű maximális folyamot a (H,s,t,c) hálózatban; *(iii)* szerint tehát $m_f = d \geq k$. Ha G-nek van olyan $e = \{u,v\}$ éle, amire az e-ből készült $e_1 = (u,v)$ és $e_2 = (v,u)$ H-beli irányított élekre $f(e_1) = f(e_2) = 1$, akkor változtassuk meg $f(e_1)$ és $f(e_2)$ értékét is nullára; és ezt a módosítást hajtsuk végre minden szóba jövő esetben. Ezek a változtatások megfelelnek a 9.3. szakasz irányítatlan élekről szóló pontjában látott módosításoknak (ahol $f(e_1)$ és $f(e_2)$ értékét a kettejük minimumával csökkentettük). Így az ott írtak szerint f továbbra is folyam és m_f sem változott.

A 10.1. Lemmát f-re alkalmazva kapjuk a H-beli $P_1,P_2,\ldots,P_d$ éldiszjunkt irányított utakat s-ből t-be (ahol $d \geq k$). Mindegyik P_i útnak megfelel egy P_i' irányítatlan út G-ben s-ből t-be. A $P_1',P_2',\ldots,P_d'$ utak pedig éldiszjunktak, hiszen ha a P_i' és a P_j' út is tartalmazná az $e = \{u,v\}$ élt (ahol $i \neq j$), akkor P_i és P_j is tartalmazna egyet az $e_1 = (u,v)$ és $e_2 = (v,u)$ élek közül. Ez azonban lehetetlen: e_1 és e_2 közül legföljebb egy szerepelhet a $P_1,P_2,\ldots,P_d$ utak élhalmazainak egyesítésében, mert $f(e_1)$ és $f(e_2)$ közül legalább az egyik nulla, a 10.1. Lemmából kapott utak e éleire pedig $f(e) = 1$; azt pedig $P_1,P_2,\ldots,P_d$ éldiszjunktsága kizárja, hogy P_i és P_j ugyanazt az élt tartalmazza e_1 és e_2 közül. Ezzel tehát *(i)*-et beláttuk. □

Ismét hangsúlyozzuk, hogy a fenti tételből (illetve annak a bizonyításából) hatékony algoritmus is kövezkezik a tétel *(i)* állítása szerinti utak vagy a *(ii)* szerinti élhalmaz keresésére, illetve a létezésük eldöntésére.

A 10.3. és 10.4. Tételek *(i)* és *(ii)* állításai közötti ekvivelenciát először Karl Menger (1902 – 1985) osztrák matematikus bizonyította be 1927-ben. (Menger ehhez nem a folyamok elméletét használta, mert az akkor még nem volt ismert.) Men-

gernek ezt a (két) tételét kicsit más, úgynevezett „minimax” formában is szokás kimondani. (Minimax tételnek szokták nevezni azokat, amik egy fajta dolog maximuma és egy másik fajta dolog minimuma közti egyenlőséget mondanak ki. Ilyenből már többet is láttunk: a 6.17. Tétel szerint intervallumgráfokban a klikkek méretének maximuma egyenlő a színek számának minimumával egy jó színezésben; a 7.18. Következmény szerint páros gráfokban a maximális párosítás mérete egyenlő a minimális lefogó ponthalmaz méretével; a 8.6. Tétel szerint páros gráfokban a maximális fokszám egyenlő a színek számának minimumával egy jó élszínezésben; illetve a 9.15. Tétel szerint $\max\{m_f : f \text{ folyam}\} = \min\{c(X) : X \ st\text{-vágás}\}$.)

Egy adott G (irányított vagy irányítatlan) gráf és annak az s és t csúcsai esetén jelölje $\lambda_G(s,t)$ az s-ből t-be vezető éldiszjunkt (irányított vagy irányítatlan) utak maximális számát G-ben. ($\lambda_G(s,t) = k$ tehát azt jelenti, hogy k éldiszjunkt út létezik G-ben s-ből t-be, de $k+1$ már nem.) Jelölje továbbá $\lambda'_G(s,t)$ az s-ből t-be vezető (irányított vagy irányítatlan) utakat lefogó élhalmazok méreteinek a minimumát.

10.5. Következmény. (Menger tétele éldiszjunkt utakra)
Minden G (irányított vagy irányítatlan) gráfra és annak az $s,t \in V(G)$, $s \neq t$ csúcsaira $\lambda_G(s,t) = \lambda'_G(s,t)$ teljesül.

Bizonyítás: Legyen $k \geq 1$ tetszőleges egész. $\lambda_G(s,t) \geq k$ definíció szerint azt jelenti, hogy létezik G-ben k éldiszjunkt (irányított vagy irányítatlan) út s-ből t-be. $\lambda'_G(s,t) \geq k$ pedig azt jelenti, hogy az s-ből t-be vezető (irányított vagy irányítatlan) utakat lefogó élhalmazok minimális mérete legalább k; más szóval: nem létezik ilyen élhalmazból legföljebb $k-1$ méretű. Így a 10.3., illetve 10.4. Tételek *(i)* és *(ii)* állításai közötti ekvivelencia azt jelenti (az irányított, illetve irányítatlan esetben), hogy $\lambda_G(s,t) \geq k$ pontosan akkor igaz, ha $\lambda'_G(s,t) \geq k$. Ebből pedig $\lambda_G(s,t) = \lambda'_G(s,t)$ valóban következik (mert ugyanazoknál a k egészeknél nagyobb vagy egyenlők). □

Amint az a bizonyításból látható, ez a következmény valóban csak más formában ismétli meg a 10.3. és 10.4. Tételek *(i)* és *(ii)* állításai közötti ekvivelenciát, de tartalmilag azonos azokkal. Kicsit felületesen (legalábbis fontos részletek homályban hagyásával) akár mondhatjuk azt is, hogy a fenti következmény speciális esete a 9.15. Ford-Fulkerson tételnek; valóban, $\lambda_G(s,t)$ a maximális folyam értékével, $\lambda'_G(s,t)$ pedig a vágások kapacitásának minimumával egyenlő a 10.3., illetve a 10.4. Tételek *(iii)* állításaiban szereplő hálózatokban. (Megjegyezzük, hogy a $\lambda_G(s,t)$ jelölés általánosan elfogadott és használt, de $\lambda'_G(s,t)$ nem, azt csak a fenti tétel kimondásához vezettük be.)

10.6. Feladat.
a) Határozzuk meg a 10.1. ábrán látható ábra G gráfjára $\lambda_G(s,t)$ értékét.
b) Tekintsünk el ebben a gráfban az élek irányításától és oldjuk meg a feladatot az így kapott H irányítatlan gráfra is.

Megoldás: Az a) feladat esetében az ábrán látható három út mutatja, hogy $\lambda_G(s,t) \geq 3$. Másrészt a 10.1. szakasz elején már említett $Z = \{(o,p),(u,p),(y,z)\}$

élhalmaz lefogja az s-ből t-be vezető irányított utakat, amiből $\lambda'_G(s,t) \leq 3$ adódik. Így Menger fenti tétele szerint $3 \leq \lambda_G(s,t) = \lambda'_G(s,t) \leq 3$, amiből tehát $\lambda_G(s,t) = 3$. (Vegyük észre, hogy valójában ezt a gondolatmenetet a 10.1. szakasz elején kicsit más formában már láttuk: a Z élhalmaz mutatta, hogy nem létezik négy éldiszjunkt út s-ből t-be; mivel hármat az ábrán látunk, ebből $\lambda_G(s,t) = 3$ következik.)

A b) feladat esetében ugyanaz a három út (illetve ezeknek az irányítatlan megfelelője) ismét mutatja, hogy $\lambda_H(s,t) \geq 3$. A Z-nek megfelelő $\big\{\{o,p\},\{u,p\},\{y,z\}\big\}$ élhalmaz viszont itt már nem fogja le az s-ből t-be vezető utakat, mert a $\{p,y\}$ él irányítatlanná válásával eljuthatunk s-ből t-be a fenti három él használata nélkül is (például sorra az s,r,u,y,p,v,t csúcsok érintésével). Szerencsére azonban a $Z' = \big\{\{p,q\},\{p,v\},\{y,z\}\big\}$ élhalmaz lefogja H-ban az s-ből t-be vezető (irányítatlan) utakat; valóban, Z' elhagyása után H összefüggősége megszűnne és s és t különböző komponensbe kerülne. Így Z' indokolja a $\lambda'_H(s,t) \leq 3$ állítást, amiből a fentiekhez hasonlóan $3 \leq \lambda_H(s,t) = \lambda'_H(s,t) \leq 3$ miatt $\lambda_H(s,t) = 3$ következik. □

10.2. Pontdiszjunkt utak

Térjünk vissza ismét a 10.1. ábrán látható ábra gráfjára: az azon látható három útról már említettük, hogy nem pontdiszjunktak, de két pontdiszjunktat még találhatunk köztük: a kék és a zöld (vagy a piros és a zöld) ilyenek. A kérdés az, hogy található-e ebben a gráfban három pontdiszjunkt (irányított) út s-ből t-be (amelyek persze egészen másak is lehetnének, mint az ábrán látható színes utak)? A válasz nemleges, aminek az indoklásához egy, a 10.1. szakasz elején látotthoz hasonló ötlet vezet – de itt már az élek helyett a gráf csúcsaira kell fókuszálni. Figyeljük meg, hogy minden, az s-ből t-be vezető irányított út áthalad a p és a z csúcsok közül legalább egyen. Valóban, ha ezt a két csúcsot (nyilván az összes rájuk illeszkedő éllel együtt) töröl-nénk, akkor a kapott gráfban már nem volna s-ből t-be irányított út (hiszen megint nem volna az $\{s,o,r,u,x,y\}$ csúcshalmazból kilépő él); így valóban nem elkerülhető, hogy egy s-ből t-be vezető út áthaladjon p-n vagy z-n. Ezért bárhogyan is adnánk meg három irányított utat s-ből t-be, a p és z csúcsok közül legalább egyet két út is tartalmazna, így ezek nem lehetnének pontdiszjunktak. Az alábbi, a 10.2-vel analóg definíció az ezt a példát általánosító fogalmat vezeti be.

10.7. Definíció. *A G irányított gráfban az $Y \subseteq V(G)$ csúcshalmaz* lefogja az s-ből t-be vezető irányított utakat *(ahol $s,t \in V(G)$ a gráf két különböző csúcsa), ha $s \notin Y$, $t \notin Y$ és minden s-ből t-be vezető, G-beli irányított út tartalmaz Y-beli csúcsot. Illetve hasonlóan irányítatlan G gráfok esetében is: az $Y \subseteq V(G)$ csúcshalmaz* lefogja az s-ből t-be vezető irányítatlan utakat, *ha $s \notin Y$, $t \notin Y$ és minden s és t végpontú, G-beli irányítatlan út tartalmaz Y-beli csúcsot.*

A fentiek szerint például az $Y = \{p,z\}$ csúcshalmaz lefogja az s-ből t-be vezető irányított utakat a 10.1. ábrán látható gráfban. Ez a definíció is átfogalmazható úgy, hogy az s-et és t-t nem tartalmazó Y akkor fogja le az s-ből t-be vezető utakat, ha

Y csúcsainak a törlése után már nincs a gráfban s-ből t-be vezető út. Itt is hangsúlyozzuk, hogy a most bevezetett fogalomnak a névhasonlóságon túl nincs köze a 7.3. Definícióban bevezetett lefogó ponthalmazhoz.

A fenti definícióban azért kellett kikötni, hogy $s \notin Y$ és $t \notin Y$, mert egy s-et vagy t-t tartalmazó Y-ra nyilván teljesülne, hogy tartalmaz csúcsot minden s-ből t-be vezető útból (mégpedig s-et vagy t-t), de ebből semmi nem következne az s-ből t-be vezető pontdiszjunkt utak számára. Ennek viszont az a következménye, hogy ha a G irányított gráfban $(s,t) \in E(G)$ (vagyis G-nek van s-ből közvetlenül t-be vezető éle), akkor G-ben egyáltalán nem létezik az s-ből t-be vezető utakat lefogó csúcshalmaz; és hasonló a helyzet a G irányítatlan gráfban akkor, ha $\{s,t\} \in E(G)$ (vagyis s és t szomszédosak). Valóban: ezekben az esetekben még az $Y = V(G) \setminus \{s,t\}$ halmaz sem tartalmazna csúcsot az (s,t), illetve az $\{s,t\}$ élből álló egy hosszúságú útból. Ezért a pontdiszjunkt utakról szóló alábbi tételekben ki kell majd kötnünk, hogy $(s,t) \notin E(G)$, illetve $\{s,t\} \notin E(G)$.

A következő két tétel a 10.3. és a10.4. Tételeknek megfelelő eredményeket mondja ki pontdiszjunkt utakra. Ezekhez ismét használni fogunk egyet a 9.3. szakasz visszavezetései közül: a csúcsok kapacitásának kezelésére alkalmazott pontszéthúzás fogalmát (lásd a 9.10. ábrát): a v csúcs széthúzása a (v_1,v_2) irányított éllé tehát azt jelenti, hogy az eredetileg v-be érkező élek végpontját v_1-re, a v-ből induló élek kezdőpontját pedig v_2-re cseréljük és felvesszük a gráfba a (v_1,v_2) élt.

10.8. Tétel. *Legyen adott a G irányított gráf, az $s,t \in V(G)$ különböző csúcsok, amelyekre $(s,t) \notin E(G)$ és a $k \geq 1$ egész. Ekkor az alábbi állítások ekvivalensek:*

(i) Létezik G-ben k pontdiszjunkt irányított út s-ből t-be.

(ii) Nem létezik G-ben legföljebb $k-1$ csúcsú, az s-ből t-be vezető irányított utakat lefogó csúcshalmaz.

(iii) A (H,s,t,c) hálózatban a maximális folyam értéke legalább k, ahol a H irányított gráf úgy készül G-ből, hogy annak minden, s-től és t-től különböző v csúcsát széthúzzuk a (v_1,v_2) éllé; c pedig az azonosan 1 kapacitásfüggvény.

Bizonyítás: Megint az *(i)⇒(ii)*, *(ii)⇒(iii)* és *(iii)⇒(i)* következtetéseket igazoljuk.

Tegyük fel először, hogy *(i)* teljesül és az s-ből t-be vezető $P_1,P_2,\ldots,P_k$ utak pontdiszjunktak. Legyen Y tetszőleges, az s-ből t-be vezető irányított utakat lefogó csúcshalmaz. Mivel Y minden s-ből t-be vezető útból tartalmaz csúcsot, ez igaz a $P_1,P_2,\ldots,P_k$ utakra is; válasszunk ezért minden $1 \leq i \leq k$ esetén P_i-ből egy olyan v_i csúcsot, amire $v_i \in Y$. Ekkor a $v_1,v_2,\ldots,v_k$ csúcsok mind különbözők, mert a P_i utaknak s-en és t-n kívül nincs közös csúcsa, $v_i \in Y$ miatt pedig $v_i \neq s$ és $v_i \neq t$. Mivel tehát Y tartalmaz k különböző csúcsot, ezért $|Y| \geq k$, amivel *(ii)*-t beláttuk.

A *(ii)⇒(iii)* állítás bizonyításához legyen $X \subseteq V(H)$ a (H,s,t,c) hálózat egy tetszőleges vágása és legyen $c(X) = p$. Mivel H minden e élére $c(e) = 1$, ez azt jelenti, hogy X-ből p darab él lép ki; válasszunk ezek közül egy tetszőleges e élt. Ekkor e lehet a G egy $v \neq s,t$ csúcsából széthúzással keletkezett (v_1,v_2) él, vagy lehet egy olyan él is, ami megfelel G egy eredeti élének. Az utóbbi esetben tehát $e = (v_2,u_1)$, ahol v_2 egy $v \in V(G)$ csúcs „indulási oldala” H-ban, u_1 pedig egy $u \in V(G)$ csúcs „érkezési oldala” H-ban; illetve lehetséges még $v_2 = s$ vagy $u_1 = t$ is (mert s-et és t-t nem húztuk szét). Ha e az utóbbi típusú él és $v_2 \neq s$, akkor dobjuk ki v_2-t X-ből;

ezzel azt érjük el, hogy e már nem lép ki X-ből (és esetleg további, a v_2-ből kilépő H-beli élek sem) és cserébe kizárólag a (v_1,v_2) él kerülhet újként az X-ből kilépő élek közé (mert v_2-be nem lép be más él H-ban). Következésképp v_2 kidobásával az X-ből kilépő élek száma biztosan nem nőtt (és esetleg csökkent). Ugyanez történik akkor is, ha az X-ből kilépő $e=(v_2,u_1)$ élre $u_1\neq t$ és u_1-et bevesszük X-be (ahelyett, hogy a v_2-t kidobnánk): ekkor egyedül az (u_1,u_2) él változhat X-ből kilépővé és cserébe legalább egy él kikerül az X-ből kilépők közül (mégpedig éppen e). Mivel a tételben feltettük, hogy $(s,t)\notin E(G)$, ezért minden, az X-ből kilépő (v_2,u_1) típusú élre a fenti két művelet közül legalább az egyik végrehajtható: v_2-t kidobhatjuk X-ből vagy u_1-et bevehetjük X-be; tegyük is ezt meg minden lehetséges esetben. A végül kapott X' vágásból tehát már csak a G csúcsaiból széthúzással keletkezett, (v_1,v_2) típusú élek lépnek ki és ezeknek a száma legföljebb p.

A 10.3. és a 10.4. Tételek bizonyításában látotthoz hasonlóan most is elmondhatjuk, hogy az X'-ből kilépő élek lefogják H-ban az s-ből t-be vezető irányított utakat (hiszen $s\in X'$ és $t\notin X'$ továbbra is igaz). A pontszéthúzások hatására a G-beli, s-ből t-be vezető irányított utak megfelelnek a H-beli ilyen utaknak: minden olyan esetben, amikor egy G-beli út áthalad egy $v\neq s,t$ csúcson, akkor a megfelelő H-beli út v_1-be érkezve a (v_1,v_2) élen át v_2-be lép és innen hasonlóan megy tovább; ez a megfeleltetés pedig fordítva is ugyanígy működik. Ebből adódik, hogy az X'-ből kilépő (v_1,v_2) éleknek megfelelő G-beli v csúcsok lefogják az s-ből t-be vezető G-beli irányított utakat. A *(ii)* állításból ezért az következik, hogy X'-ből legalább k él lép ki; ebből tehát $k\leq p=c(X)$ is adódik (ahol X továbbra is a fenti bekezdésben írt módosítások előtti, tetszőlegesen választott vágást jelöli). Így a 9.15. Tételből ismét azt kapjuk, hogy $\max\{m_f : f \text{ folyam}\}\geq k$ teljesül a (H,s,t,c) hálózatban, amivel tehát *(iii)*-at beláttuk.

A *(iii)*⇒*(i)* következtetés bizonyítása pedig a korábbiakhoz hasonlóan megy: a 9.17. Lemma miatt létezik a (H,s,t,c) hálózatban egy olyan, $m_f=d\geq k$ értékű folyam, amire $f(e)=0$ vagy $f(e)=1$ teljesül minden e élre. Ebből a 10.1. Lemmát alkalmazva kapjuk a H-beli $P_1,P_2,\ldots,P_d$ éldiszjunkt irányított utakat s-ből t-be (ahol $d\geq k$). Az előző bekezdésben írtak szerint mindegyik P_i útnak megfelel egy P_i' út G-ben s-ből t-be. A $P_1',P_2',\ldots,P_d'$ utak pedig pontdiszjunktak, hiszen ha a P_i' és a P_j' út is tartalmazna egy $v\neq s,t$ csúcsot G-ben, akkor P_i és P_j is tartalmazná a (v_1,v_2) élt H-ban. Ezzel tehát *(i)*-et beláttuk. □

A következő, pontdiszjunkt irányítatlan utakról szóló tétel bizonyításában már nincs új gondolat, csak az eddig látottak kombinálására lesz szükség.

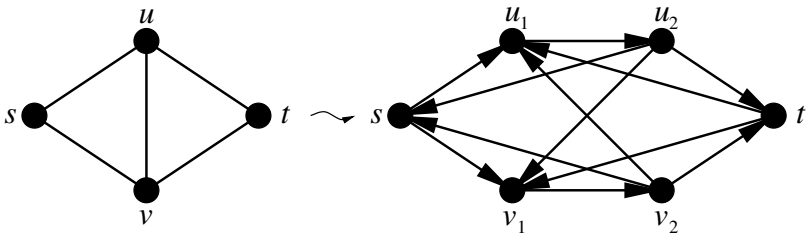
10.9. Tétel. *Legyen adott a G irányítatlan gráf, az $s,t\in V(G)$ különböző csúcsok, amikre $\{s,t\}\notin E(G)$ és a $k\geq 1$ egész. Ekkor az alábbi állítások ekvivalensek:*

(i) Létezik G-ben k pontdiszjunkt irányítatlan út s-ből t-be.

(ii) Nem létezik G-ben legföljebb $k-1$ csúcsú, az s-ből t-be vezető irányítatlan utakat lefogó csúcshalmaz.

(iii) A (H,s,t,c) hálózatban a maximális folyam értéke legalább k, ahol a H irányított gráf úgy készül G-ből, hogy először G minden $\{u,v\}$ élét helyettesítjük az (u,v) és (v,u) irányított élekkel, majd a kapott gráf minden, s-től és t-től különböző v csúcsát széthúzzuk a (v_1,v_2) éllé; c pedig az azonosan 1 kapacitásfüggvény.

A tétel *(iii)* állításában szereplő átalakításra a 10.3. ábra mutat példát: a bal oldalon látható G gráfból a jobbra látható H keletkezik.



10.3. ábra

A 10.9. Tétel bizonyítása: Az *(i)*⇒*(ii)* következtetés bizonyítása azonos a 10.8. Tétel bizonyításában írtakkal, csak a $P_1, P_2, \ldots, P_k$ utak most irányítatlanok.

A *(ii)*⇒*(iii)* állítás bizonyításához jelöle G' azt a gráfot, amit a G éleinek két irányítottal való helyettesítésével kaptunk (még a pontszéthúzások előtt). Ha *(ii)* teljesül, akkor G'-ben sem létezhet az s-ből t-be vezető irányított utakat lefogó, $k-1$ elemű csúcshalmaz, így a 10.8. Tétel szerint legalább k a maximális folyam értéke abban a hálózatban, amit G'-ből csúcsszéthúzással kaphatunk (a $c \equiv 1$ kapacitásfüggvény mellett). Mivel ez éppen a (H,s,t,c) hálózat, ezzel *(iii)*-at beláttuk.

Végül a *(iii)*⇒*(i)* bizonyítása is akadálytalan: egy, a 9.17. Lemma szerinti egész, vagyis minden élen 0 vagy 1 értékű maximális folyamból a 10.1. Lemma szolgáltatja a $d = m_f \geq k$ éldiszjunkt irányított utat s-ből t-be H-ban, amiből először G'-ben d darab pontdiszjunkt irányított, majd G-ben ugyanennyi pontdiszjunkt irányítatlan utat kapunk. □

A 10.8. és 10.9. tételek esetében is fontos hangsúlyozni az algoritmikus vonatkozásokat: mindkettőből a megfelelő feladat hatékony megoldhatósága is következik. Ennek a két tételnek az *(i)* és *(ii)* állításai közötti ekvivalenciát is a 10.1 szakaszban már említett Karl Menger bizonyította be először és hasonlóan az ott írtakhoz, ezek is kimondhatók „minimax” megfogalmazásban. Ehhez egy G (irányított vagy irányítatlan) gráf és annak az s és t csúcsai esetén jelölje $\kappa_G(s,t)$ az s-ből t-be vezető pontdiszjunkt (irányított vagy irányítatlan) utak maximális számát G-ben; továbbá jelölje $\kappa'_G(s,t)$ az s-ből t-be vezető (irányított vagy irányítatlan) utakat lefogó csúcshalmazok méreteinek a minimumát. (A κ görög betű, a kiejtése: „kappa”.)

10.10. Következmény. (Menger tétele pontdiszjunkt utakra)
Ha a G (irányított vagy irányítatlan) gráfra és annak az $s,t \in V(G)$, $s \neq t$ csúcsaira $(s,t) \notin E(G)$, illetve $\{s,t\} \notin E(G)$ teljesül (az irányított, illetve az irányítatlan esetben), akkor $\kappa_G(s,t) = \kappa'_G(s,t)$.

Bizonyítás: Analóg módon érvelhetünk a 10.5. Következmény bizonyításával. $\kappa_G(s,t) \geq k$ azt jelenti, hogy létezik G-ben k pontdiszjunkt út s-ből t-be. $\kappa'_G(s,t) \geq k$ pedig azt jelenti, hogy nem létezik G-ben legföljebb $k-1$ méretű, az s-ből t-be vezető utakat lefogó csúcshalmaz. Így a 10.8., illetve 10.9. Tételek *(i)* és *(ii)* állításai

közötti ekvivelencia miatt $\kappa_G(s,t) \geq k$ pontosan akkor igaz, ha $\kappa'_G(s,t) \geq k$. Mivel ez minden $k \geq 1$ egészre teljesül, ebből $\kappa_G(s,t) = \kappa'_G(s,t)$ valóban következik. □

Megint nem hazugság (legföljebb erősen leegyszerűsítő) azt állítani, hogy ez a következmény is speciális esete a 9.15. Ford-Fulkerson tételnek; annyi mindenképp igaz, hogy $\kappa_G(s,t)$ a maximális folyam értékével, $\kappa'_G(s,t)$ pedig a vágások minimális kapacitásával egyenlő a 10.8., illetve a 10.9. Tételek *(iii)* állításaiban szereplő hálózatokban (és e közül a két jelölés közül is csak $\kappa_G(s,t)$ általánosan elfogadott).

10.11. Feladat. Határozzuk meg $\kappa_G(s,t)$ és $\kappa_G(s,p)$ értékét a 10.1. ábrán látható ábra G gráfjára, illetve az abból az élek irányításának figyelmen kívül hagyásával kapott H irányítatlan gráfra.

Megoldás: Már többször említettük, hogy az ábrán látható utak közül például a kék és a zöld pontdiszjunkt. A 10.2 szakasz elején az is kiderült, hogy az $Y = \{p,z\}$ halmaz lefogja az s-ből t-be vezető utakat, így három pontdiszjunkt út már nem létezik s-ből t-be. Így tehát $\kappa_G(s,t) = 2$. Mindezek H-ra ugyanúgy elmondhatók, így $\kappa_H(s,t) = 2$ is adódik.

Könnyű megadni G-ben két pontdiszjunkt utat s-ből p-be (például az o, illetve u csúcsokon át). Ennél több viszont nyilván nem létezik, hiszen p-be csak két él lép be. (Ezt úgy is mondhatjuk, hogy az (o,p) és (u,p) élek lefogják az s-ből p-be vezető utakat, így még éldiszjunkt útból sem létezhet három s-ből p-be.) Így $\kappa_G(s,p) = 2$.

H-ban viszont már megadható három pontdiszjunkt út s-ből p-be: például az o-n, illetve u-n át vezető kettő hosszú utak mellé az x-en és y-on át vezető három hosszút is felvehetjük. Így $\kappa_H(s,p) \geq 3$. Másrészt az $Y = \{o,u,x\}$ csúcshalmaz lefogja az s-ből p-be vezető utakat H-ban (ez könnyen ellenőrizhető). Így $\kappa'_H(s,p) \leq 3$. Ezekből és a 10.10. Következményből tehát $3 \leq \kappa_H(s,p) = \kappa'_H(s,p) \leq 3$, és így $\kappa_H(s,p) = 3$ adódik. □

10.3. Többszörös összefüggőség

Egy hálózat tervezésekor (függetlenül attól, hogy az milyen célt szolgál) szinte mindig elvárás, hogy az azt reprezentáló gráf összefüggő legyen. Azonban számos alkalmazásban ez nem elegendő: ha például egy számítógép-hálózatban előfordulhatna az, hogy egyetlen link vagy csomópont meghibásodása (vagyis a gráfból való törlése) elrontja az összefüggőséget, az komoly kockázatot és egyben a hálózat alkalmatlanságát is jelentené. Sőt, az összefüggőség nem csak egyetlen linken vagy csomóponton nem múlhat: érdemes lehet megkövetelni, hogy egy adott, az alkalmazástól függő számú linken vagy csomóponton se múlhasson. Természetes gondolat egy hálózatot annál megbízhatóbbnak tekinteni, minél inkább ellenáll véletlen meghibásodásoknak vagy akár szándékos támadásoknak; más szóval: minél nagyobb számra teljesül, hogy ennyi él vagy csúcs elhagyása még nem ronthatja el az összefüggőséget. Az alábbi definíció az ebből a gondolatból fakadó, számos alkalmazásban kulcsfontosságú fogalmakat vezeti be.

10.12. Definíció. *Legyen G (irányítatlan) gráf és $k \geq 1$ tetszőleges egész.*

- *A G gráfot* k-szorosan élösszefüggőnek *mondjuk, ha az élei közül bárhogyan legföljebb $(k-1)$-et törölve mindig összefüggő gráfot kapunk.*
- *G-t* k-szorosan pontösszefüggőnek *(vagy röviden* k-szorosan összefüggőnek*) mondjuk, ha legalább $k+1$ csúcsa van és a csúcsai közül bárhogyan legföljebb $(k-1)$-et (az azokra illeszkedő élekkel együtt) törölve mindig összefüggő gráfot kapunk.*

Amint az a definícióból látható, ha nem deklaráljuk, hogy él- vagy pontösszefüggőségről van szó, hanem csak k-szorosan összefüggő gráfról beszélünk, akkor ez az utóbbit jelenti. A k-szoros (pont)összefüggőség definíciójában azért kellett kikötni, hogy a gráfnak legalább $k+1$ csúcsa legyen, mert különben egy „kicsi” gráf sokszorosan összefüggőnek értelmeződne (például egy három pontú kör százszorosan összefüggő volna), ami ellentmond a többszörös összefüggőség fogalmáról fentebb írt intuíciónak.

A fenti definíció csak irányítatlan gráfokról szól; bár létezik ennek a két fogalomnak irányított gráfokra vonatkozó megfelelője is, ezekkel itt nem foglalkozunk, a fejezet hátralévő részében csak irányítatlan gráfokat vizsgálunk.

Használjuk ismét példának a 10.1. ábrán látható ábra gráfjából az élek irányításának figyelmen kívül hagyásával kapható H gráfot. A 10.6. Feladat megoldásában kiderült, hogy H-ból a $Z' = \big\{\{p,q\},\{p,v\},\{y,z\}\big\}$ élhalmaz elhagyásával kapott gráf már nem összefüggő; ebből a most definiált fogalmakat használva az következik, hogy H nem négyszeresen élösszefüggő (mert ha az volna, akkor bármelyik legföljebb három élét elhagyva összefüggő gráfot kellene kapnunk). Hasonlóan, a 10.11. Feladat megoldásában pedig azt használtuk ki, hogy H-ból az $Y = \{p,z\}$ csúcshalmaz elhagyása után kapott gráf nem összefüggő; ebből most az következik, hogy H nem háromszorosan összefüggő. Persze ezzel nyitva hagytuk a kérdést, hogy H vajon háromszorosan élösszefüggő-e, illetve kétszeresen pontösszefüggő-e? A válasz mindkét esetben igen, de ezeket pusztán a fenti definíciók alapján hiánytalanul megindokolni elég fárasztó volna: az előbbi azt jelentenené, hogy H-ból az összes lehetséges módon elhagyunk két élt és mindig megvizsgáljuk, hogy a kapott gráf összefüggő-e; az utóbbihoz pedig a csúcsokat kellene egyesével elhagyni és mindig tesztelni az összefüggőséget. Később, több eszköz birtokában sokkal rövidebb indoklásokat is fogunk tudni adni ezekre (lásd a 10.17. Feladatot).

Ez a példa a most definiált két fogalom közti különbséget is mutatja: H háromszorosan élösszefüggő, de nem háromszorosan pontösszefüggő (bár az előbbit egyelőre nem indokoltuk meg). Könnyen lehet a két fogalom közti, még látványosabb különbséget mutató példát is megadni: legyen G az a 201 csúcsú gráf, amit két diszjunkt ponthalmazokon megadott 100 csúcsú teljes gráfból kapunk úgy, hogy felveszünk egy további v csúcsot és v-t összekötjük az összes többi (200 darab) csúccsal. Ekkor G nem kétszeresen összefüggő, hiszen v törlése elrontja az összefüggőséget; de nem nehéz végiggondolni (a részleteket mellőzzük), hogy G 100-szorosan élösszefüggő, mert bármely legföljebb 99 élének az elhagyása után összefüggő gráfot kapunk. Ezekből a példákból is érezhető, hogy a k-szoros összefüggőség erősebb

feltétel a k-szoros élösszefüggőségnél (vagyis az előbbiből következik az utóbbi), hiszen néhány csúcs törlése nagyobb „rombolást” végez a gráfban, mint ugyanennyi élé; ez valóban igaz és bár pusztán a definíciók alapján körülményes volna megindokolni, a későbbiekből ez is könnyen következni fog (lásd a 10.16. Következményt). A két fogalom közti különbség egyedül a $k = 1$ esetben tűnik el: az egyszeres élösszefüggőség és pontösszefüggőség is a gráf összefüggőségét jelenti (hiszen nulla darab él vagy csúcs elhagyása után követeli meg az összefüggőséget).

Természetesen ugyanaz a G gráf több különböző k értékre is lehet k-szorosan összefüggő vagy élösszefüggő. Például a fentebb említett H gráf nem csak háromszorosan élösszefüggő, hanem persze kétszeresen is (hiszen legföljebb egy élének az elhagyása sem ronthatja el az összefüggőséget, ha ugyanez legföljebb két élre igaz) és még egyszeresen is (hiszen H összefüggő). Általában is nyilván igaz, hogy ha G k-szorosan pontösszefüggő, illetve élösszefüggő, akkor minden $1 \leq \ell \leq k$-ra is ℓ-szeresen pontösszefüggő, illetve élösszefüggő. Ez az egyszerű megfigyelés adja a létjogosultságát az alábbi definíciónak.

10.13. Definíció. *A G gráfra $\lambda(G)$, illetve $\kappa(G)$ jelöli a legnagyobb olyan k egészt, amire G k-szorosan élösszefüggő, illetve k-szorosan pontösszefüggő.*

Például a $\lambda(G) = k$ állítás azt mondja G-ről, hogy abból bárhogyan legföljebb $k-1$ élt elhagyva mindig összefüggő gráfot kapunk, de k élt már el lehet hagyni úgy, hogy az összefüggőség megszűnjön. Így a 10.1. ábrán látható gráf irányítatlan H megfelelőjéről fentebb azt állítottuk, hogy $\lambda(H) = 3$ és $\kappa(H) = 2$ (de ezeket csak a 10.17. Feladatban fogjuk hiánytalanul megindokolni).

$\lambda(G)$-re, illetve $\kappa(G)$-re tekinthetünk úgy, mint a gráfok egyfajta biztonsági mértékére: minél nagyobbak ezek, annál több élt, illetve csúcsot ért sérülés vagy támadás esetén garantált a megmaradó gráf összefüggősége.

10.14. Feladat. A G gráf csúcshalmaza legyen $V(G) = \{10, 11, 12, \ldots, 39\}$ és két csúcs akkor legyen szomszédos G-ben, ha a megfelelő számok első számjegye különböző. Határozzuk meg $\lambda(G)$ és $\kappa(G)$ értékét.

Megoldás: Vezessük be a $V_1 = \{10, \ldots, 19\}$, $V_2 = \{20, \ldots, 29\}$ és $V_3 = \{30, \ldots, 39\}$ jelöléseket. Ekkor tehát két csúcs pontosan akkor szomszédos G-ben, ha V_1, V_2 és V_3 közül nem ugyanabba esnek.

G-ből el lehet hagyni 20 csúcsot úgy, hogy a kapott gráf ne legyen összefüggő: például $V_1 \cup V_2$ elhagyásával csak a V_3 csúcsaiból álló, él nélküli gráf marad. Így G nem 21-szeresen összefüggő.

Hagyjunk most el G-ből legföljebb 19 tetszőleges csúcsot és a kapott G' gráfból válasszunk két különböző csúcsot, x-et és y-t. Ha x és y nem szomszédosak, akkor ugyanabba a V_i halmazba tartoznak; mivel azonban G' legalább 11 csúcsú, kell létezzen G'-nek olyan z csúcsa is, ami nincs V_i-ben és így x-szel és y-nal is szomszédos. Következik, hogy G' összefüggő (hiszen bármely két csúcsa vagy szomszédos, vagy

van közös szomszédjuk). Ezzel tehát megmutattuk, hogy G 20-szorosan összefüggő (hiszen a definíciónak az a feltétele is teljesül rá, hogy legalább 21 csúcsa van).

Mivel G 20-szorosan összefüggő, de 21-szeresen már nem, ezért $\kappa(G) = 20$.

G-ben minden pont foka 20, hiszen ha $v \in V_i$, akkor v a V_i-n kívüli 20 csúccsal szomszédos. Ezért G-ből 20 élt is el lehet hagyni úgy, hogy az összefüggőség megszűnjön: bármelyik v csúcsra illeszkedő 20 él megfelelő. Így G nem 21-szeresen élösszefüggő.

Hagyjunk el G-ből tetszőlegesen legföljebb 19 élt, a kapott gráfot jelölje G' és legyen $x, y \in V(G)$ két különböző csúcs. Ha x és y ugyanabba a V_i-be tartoznak, akkor kis változtatással megismételhető a fenti gondolatmenet: ha x-nek és y-nak egyik $z \notin V_i$ csúcs sem volna közös szomszédja G'-ben, akkor minden ilyen z-re legalább egy olyat el kellett volna hagyni a z-re illeszkedő élek közül, aminek a másik végpontja x vagy y, így ez már legalább 20 különböző él elhagyását jelentené. Így x és y között van (2 hosszú) út G'-ben.

Vizsgáljuk most azt az esetet, amikor x és y különböző V_i-be tartoznak, például $x \in V_1$ és $y \in V_2$. Ha x és y szomszédosak G'-ben, akkor persze van köztük (1 hosszú) út. Ha nem, akkor $\{x, y\}$ az egyik a G-ből elhagyott élek közül. Ha x-nek és y-nak van közös, V_3-beli szomszédja, akkor megint van köztük (2 hosszú út) G'-ben. Ha nem, akkor minden V_3-beli csúcsra illeszkedik olyan az elhagyott élek közül, aminek a másik végpontja x vagy y. Ezzel tehát ($\{x, y\}$-nal együtt) már legalább 11-et „megtaláltunk" az elhagyott élek közül, így legföljebb 8 olyan elhagyott él lehet $\{x, y\}$-on kívül, ami V_1-beli csúcsot köt össze V_2-belivel. Ezért kell létezzen olyan $a \in V_1$ és $b \in V_2$ csúcs, amikre nem illeszkedik V_1 és V_2 között futó, elhagyott él. Következik, hogy a sorra az x, b, a, y csúcsokat érintő (3 hosszú) út összeköti x-et y-nal G'-ben. Mivel tehát bármely két csúcsa között vezet út, ezért G' összefüggő.

Így G 20-szorosan élösszefüggő, amiből az előző bekezdésben írtak miatt $\lambda(G) = 20$ következik. □

Az alábbi, ugyancsak Karl Mengertől származó tétel a k-szoros élösszefüggőségre és pontösszefüggőségre ad egy-egy szükséges és elégséges feltételt és ezzel egyben a téma legalapvetőbb eredményét mondja ki.

10.15. Tétel. (Menger tétele többszörös összefüggőségre)
Legyen G (irányítatlan) gráf és $k \geq 1$ egész szám. Ekkor

(i) G akkor és csak akkor k-szorosan élösszefüggő, ha bármely két különböző csúcsa között létezik k éldiszjunkt út;

(ii) G akkor és csak akkor k-szorosan pontösszefüggő, ha legalább $k+1$ csúcsú és bármely két különböző csúcsa között létezik k pontdiszjunkt út.

Bizonyítás: (i) bizonyításához tegyük fel először, hogy G bármely két különböző csúcsa között létezik k éldiszjunkt út és hagyjunk el G-ből tetszőlegesen legföljebb $k-1$ élt, a kapott gráfot jelölje G'. Ekkor bármely $s, t \in V(G)$, $s \neq t$ csúcsok között kell létezzen út G'-ben, hiszen a G-ben köztük létező $P_1, \ldots, P_k$ éldiszjunkt utak közül legalább egy G'-ben is érintetlenül megvan; valóban, egy él elhagyása a P_i-k éldiszjunktsága miatt legföljebb egyet tehetett közülük tönkre és a P_i-k számánál

kevesebb élt hagytunk el. Ez tehát definíció szerint azt jelenti, hogy G' összefüggő és így G k-szorosan élösszefüggő. Ezzel beláttuk *(i)*-ből a feltétel elégségességét (vagyis a „jobbról balra irányt").

A másik irány (vagyis a szükségesség) bizonyításához tegyük fel, hogy G k-szorosan élösszefüggő és legyen $s,t \in V(G)$, $s \neq t$ két tetszőleges csúcsa. Ha nem létezne s és t között k éldiszjunkt út G-ben, akkor a 10.4. Tétel miatt létezne egy, az s és t közti utakat lefogó, legföljebb $k-1$ élű Z élhalmaz. Ez azonban ellentmondana G k-szoros élösszefüggőségének: a Z elhagyásával kapott G' gráf nem volna összefüggő, mert s és t között nem volna út G'-ben. Ezzel tehát *(i)* bizonyítása teljes.

(ii) bizonyításából az elégségesség indoklása (a „jobbról balra irány") lényegében azonos a fenti, első bekezdésében írtakkal, csak élek helyett csúcsokra kell megismételni: a legföljebb $k-1$ csúcs elhagyásával kapott G' gráfban bármely $s,t \in V(G')$, $s \neq t$ csúcsok között létezik út, mert a G-ben köztük létező $P_1,\ldots,P_k$ pontdiszjunkt utak közül legföljebb $(k-1)$-et tehetett tönkre a csúcsok elhagyása. Így G' definíció szerint mindig összefüggő, G pedig k-szorosan összefüggő.

Végül *(ii)*-ből a szükségesség bizonyításához tegyük fel, hogy G k-szorosan összefüggő és legyen $s,t \in V(G)$, $s \neq t$ két tetszőleges csúcsa. Ha s és t nem szomszédosak, akkor a bizonyítás lényegében azonos módon működik az *(i)* szükségességének fenti indoklásával: ha nem létezne s és t között k pontdiszjunkt út G-ben, akkor a 10.9. Tétel szerint létező, az s és t közti utakat lefogó, legföljebb $k-1$ csúcs elhagyásával kapott G' gráf nem volna összefüggő (mert s és t között nem volna út G'-ben), ami ellentmondana G k-szoros összefüggőségének.

Ezzel azonban a bizonyítás egyelőre nem teljes: ha s és t szomszédosak G-ben, akkor a 10.9. Tétel nem alkalmazható, így ebben az esetben módosítani kell a fenti gondolatmenetet. Ilyenkor hagyjuk el G-ből az s és t közötti (összes) élt, a kapott gráfot jelölje H. Megmutatjuk, hogy H-ban létezik $k-1$ pontdiszjunkt út s és t között; ebből valóban következni fog, hogy G-ben létezik közöttük k pontdiszjunkt út, hiszen a H-beli $k-1$ úthoz hozzávehetünk egy egyetlen, s és t közti élből álló k-adikat. Tegyük fel ezért indirekt, hogy nem létezik H-ban $k-1$ pontdiszjunkt út s és t között. Ekkor alkalmazhatjuk a 10.9. Tételt H-ra (és k helyett $(k-1)$-re): létezik a legföljebb $k-2$ elemű Y csúcshalmaz, ami lefogja az s és t közti utakat H-ban. Ez tehát azt jelenti, hogy Y csúcsait (és az ezekre illeszkedő éleket) elhagyva H-ból a kapott H' gráf nem összefüggő és s és t H' különböző komponenseibe tartozik. H'-nek létezik s-től és t-től különböző v csúcsa, mert a k-szoros összefüggőség definíciója szerint G legalább $k+1$ csúcsú és ezek közül legföljebb $(k-2)$-t hagytunk el. Ekkor v nyilván az s-et és t-t tartalmazó H'-beli komponsensek közül legföljebb az egyikben van benne; tegyük fel például, hogy v nincs az s komponensében. Hagyjuk most el G-ből az $Y \cup \{t\}$ csúcshalmazt, a kapott gráfot jelölje G'. Mivel t elhagyása az s és t közti él (vagy élek) elhagyását is maga után vonja, ezért G' minden éle H'-ben is benne van. Így s és v G'-ben sem tartozik közös komponensbe, vagyis G' nem összefüggő. Ez azonban ellentmond G k-szoros összefüggőségének (hiszen $|Y \cup \{t\}| = k-1$), amivel a tétel bizonyítása teljes. □

Az utolsó bekezdés technikai bonyodalmaitól eltekintve látszik, hogy a fenti bizonyítás valójában csak a 10.4. és 10.9. Tételekre támaszkodik, illetve azokat két előre rögzített csúcs helyett az összes csúcspárra alkalmazza. A fenti tételnek közvetlen következménye az alábbi, korábban már említett állítás.

10.16. Következmény. *Ha a G gráf k-szorosan pontösszefüggő, akkor k-szorosan élösszefüggő is.*

Bizonyítás: Ha G k-szorosan pontösszefüggő, akkor a 10.15. Tétel *(ii)* állítása szerint bármely két csúcsa között létezik k pontdiszjunkt út. Mivel a pontdiszjunkt utak egyben éldiszjunktak is, ezért ebből a 10.15. Tétel *(i)* állítása szerint valóban következik, hogy G k-szorosan élösszefüggő. □

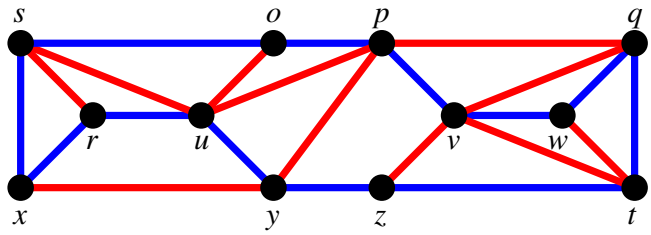
Ismét hangsúlyozzuk, hogy ennek az állításnak a megfordítása nem igaz, erre fentebb már több példát is láttunk. A 10.16. Következményt fogalmazhatjuk úgy is, hogy $\lambda(G) \geq \kappa(G)$ minden minden G gráfra igaz; valóban, ha $\kappa(G) = k$, akkor a következményből G k-szoros élösszefüggősége, vagyis $\lambda(G) \geq k$ adódik.

A 10.16. Következmény ismeretében jelentősen lerövidíthetjük például a 10.14. Feladat megoldását is: G 20-szoros pontösszefüggőségéből azonnal következik a 20-szoros élösszefüggősége is, amit pusztán a definícióra alapozva két bekezdésen át kellett indokolnunk. Az alábbi feladattal pedig a szakasz elejéről maradt adósságunkat törlesztjük.

10.17. Feladat. Legyen H a 10.1. ábrán látható gráfból az élek irányításának figyelmen kívül hagyásával kapott gráf. Mutassuk meg, hogy $\lambda(H) = 3$ és $\kappa(H) = 2$.

Megoldás: Korábban már láttuk, hogy H nem négyszeresen élösszefüggő és nem háromszorosan pontösszefüggő, mert a $\big\{\{p,q\},\{p,v\},\{y,z\}\big\}$ élhalmaz, illetve a $\{p,z\}$ csúcshalmaz elhagyásával kapott gráfok nem összefüggők.

A 10.4. ábrán H éleit megszíneztük két színnel: jól látható, hogy a kék élek egy C Hamilton-kört, a pirosak pedig egy F feszítőfát alkotnak H-ban.



10.4. ábra

Mivel H bármely két csúcsa között két pontdiszjunkt utat kapunk C két megfelelő íve mentén, ezért H kétszeres pontösszefüggősége következik a 10.15. Tétel *(ii)* állításából. Ráadásul ezt a két utat mindig kiegészíthetjük egy harmadikkal, aminek minden éle F-beli (hiszen a feszítőfa összefüggő); az így kapott három út pedig mindig éldiszjunkt, hiszen az F-beli út élei pirosak, míg a C-beli két út élei kékek. Így a 10.15. Tétel *(i)* állításából következően H háromszorosan élösszefüggő.

Ezzel tehát megindokoltuk a $\lambda(H) = 3$ és $\kappa(H) = 2$ állításokat is.

Megjegyezzük, hogy ez a megoldás erősen kihasználta H egy szerencsés tulajdonságát (nevezetesen az éldiszjunkt Hamilton-kör és feszítőfa létezését), de ez a gondolatmenet nem működött volna minden hasonló helyzetben. Így nem igaz például, hogy minden kétszeresen összefüggő gráf vagy minden háromszorosan élösszefüggő gráf tartalmaz Hamilton-kört. □

Vizsgáljuk meg végül a k-szoros összefüggőség és élösszefüggőség eldöntésének, illetve $\kappa(G)$ és $\lambda(G)$ kiszámításának a problémáját algoritmikus szempontból. Ha pusztán a definíciókból kiindulva próbálnánk algoritmust tervezni, az exponenciális futásidőhöz vezetne; valóban, ha az n csúcsú és m élű G-ből az összes, legföljebb $k-1$ elemű csúcshalmazt, illetve élhalmazt megpróbálnánk egyesével elhagyni és a kapott gráf összefüggőségét tesztelni, ez n-hez, illetve m-hez képest exponenciálisan sok tesztelést jelenthetne (például akkor, ha k értéke durván n, illetve m fele volna, vagy ha ezeknek legalább valamilyen rögzített konstansszorosa). A 10.15. Tételnek fontos következménye, hogy ezek a kérdések polinomiális futásidejű algoritmussal is megválaszolhatók. Azt ugyanis a 10.4., illetve 10.9. Tételekből tudjuk, hogy egy rögzített csúcspárra polinomiális futásidőben eldönthető, hogy van-e köztük k éldiszjunkt, illetve csúcsdiszjunkt út (például a folyamokra vonatkozó javítóutas algoritmussal). A 10.15. Tétel szerint tehát a k-szoros pont-, illetve élösszefüggőség eldöntéséhez megtehetjük, hogy ezt minden csúcspárra megvizsgáljuk (illetve a 10.15. Tétel bizonyításának utolsó bekezdése szerint a pontösszefüggőség esetében a szomszédos csúcspárok közti éleket előtte elhagyjuk és k értékét eggyel csökkentjük). Mivel ez egy n csúcsú gráf esetén $\binom{n}{2} < \frac{1}{2}n^2$ ilyen ellenőrzést jelent, ez polinomiális futásidőt eredményez.

A valósághoz tartozik, hogy bár a fenti bekezdésben vázolt algoritmusok valóban polinomiálisak, a gyakorlati alkalmazások jó részében már túl lassúak volnának. A k-szoros összefüggőség és élösszefüggőség tesztelésére, valamint $\kappa(G)$ és $\lambda(G)$ kiszámítására léteznek ennél jóval hatékonyabb algoritmusok is, de ezekkel ebben a jegyzetben nem foglalkozunk (viszont egy részük szerepel az informatikus MSc képzés felsőbb matematika anyagában).

Annyit azért megjegyzünk, hogy a k-szoros élösszefüggőség eldöntése egy egyszerű ötlettel jelentősen felgyorsítható a fentiekhez képest: nem szükséges minden $s \neq t$ csúcspárra megvizsgálni k éldiszjunkt út létezését s és t között, hanem elegendő egy tetszőlegesen rögzített s mellett az összes $t \neq s$ csúcsra megtenni ugyanezt; ezzel $\binom{n}{2}$ helyett csak $(n-1)$-szer kell futtatni a javítóutas algoritmust. Valóban, ha G nem k-szorosan élösszefüggő, mert a legföljebb $k-1$ élű Z élhalmazának elhagyásával kapható G' gráf nem összefüggő, akkor minden olyan esetben, amikor t a G'-nek egy s-étől különböző komponensében van, Z lefogja az s és t közötti utakat, így az $n-1$ próbálkozás során Z létezésére fény derül. Hasonló gondolat azonban a pontösszefüggőség esetében már nem működne: példa erre az a 10.12. Definíció után már említett 201 csúcsú G gráf, aminek a v csúcsa minden más csúccsal szomszédos, de a v törlése után keletkező gráf két diszjunkt 100 csúcsú teljes gráfból áll; ebben v és bármelyik másik csúcs között könnyen láthatóan létezik 100 pontdiszjunkt út, de G mégsem kétszeresen összefüggő.

11. fejezet

Legrövidebb utak

A gráfokra vonatkozó algoritmusok között a gyakorlati alkalmazásokban leggyakrabban felmerülők minden bizonnyal a legrövidebb utakat kereső eljárások. Ez érthető is: a globális műholdas helymeghatározó rendszerek korában számtalan térképes alkalmazásban, illetve járművek beépített navigációs szoftvereiben működnek ilyen algoritmusok. Emellett sok olyan alkalmazása is van ezeknek, amikben a minimalizálandó távolság nem a térbeli távolságból vagy akár a menetidőből származik, hanem egészen más típusú bemenő adatokból.

A *legrövidebb út* problémában adott egy $G = (V, E)$ irányított gráf, az élein egy $w : E \to \mathbb{R}$ súlyfüggvény, valamint egy $s \in V(G)$ csúcs. A feladat az, hogy minden $v \in V(G)$ csúcsra meghatározzuk egy s-ből v-be vezető legrövidebb irányított út hosszát, amit táv(v)-vel fogunk jelölni. Itt egy P út hossza alatt értelemszerűen a P-t alkotó e élek $w(e)$ súlyainak az összegét értjük. Természetesen előfordulhat, hogy egy v csúcs nem elérhető s-ből irányított úton; ilyenkor ezt a tényt a legrövidebb út feladatot megoldó algoritmusoknak nyilván ki kell tudni mutatni (noha erre a 2. fejezetben látott BFS eljárás is alkalmas volna). Ilyen esetekben kényelmes lesz (mert az algoritmusok leírását egyszerűsíti) táv(v)-t ∞-nek értelmezni. Ha viszont egy v csúcsra táv(v) véges (vagyis v elérhető s-ből irányított úton), akkor táv(v) kiszámítása mellett az is része a feladatnak, hogy az algoritmus kimenetéből kiolvasható legyen egy s-ből v-be vezető, táv(v) hosszúságú (vagyis legrövidebb) út is.

A legrövidebb út probléma nyilván irányítatlan gráfokra is felvethető, de ezzel a változattal nem kell külön foglalkoznunk, mert itt is működik a 9.3. és a 10.1. szakaszban már látott visszavezetés: ha minden $e = \{u, v\}$ irányítatlan élt helyettesítünk az $e_1 = (u, v)$ és $e_2 = (v, u)$ irányított élekkel és ezekhez a $w(e_1) = w(e_2) = w(e)$ súlyt rendeljük, akkor az így kapott irányított gráfban az s-ből a többi csúcsba vezető legrövidebb irányított utakat meghatározva az irányítatlan gráfra vonatkozó feladat megoldását kapjuk. Így az alábbiakban feltesszük, hogy G irányított gráf (és a G-beli irányított utakra is röviden csak útként fogunk hivatkozni).

A fejezetbeli algoritmusok leírását egyszerűsítendő azt is végig fel fogjuk tenni, hogy a bemenetet adó G (irányított) gráfban nincsenek hurokélek és bármely $u, v \in V(G)$ csúcsok esetén G-nek leföljebb egy u-ból v-be vezető éle van (miközben

az persze előfordulhat, hogy G-nek van egy u-ból v-be és egy v-ből u-ba vezető éle is – például akkor, ha G az előző bekezdésben írtak szerint egy irányítatlan gráfból, az élek megduplázásával készült). Valóban, mivel egy út nem tartalmazhat hurokélt, ezért az esetleges hurokélek a feladat érdemi megváltoztatása nélkül törölhetők a bemenetből. Továbbá ha a bemenetben több u-ból v-be vezető él is volna valamely u, v csúcsokra, akkor elég volna ezek közül a legkisebb súlyúak egyikét megtartani, a többi törölhető, mert azok úgysem lehetnének rajta egy legrövidebb úton.

A legrövidebb út probléma fenti kitűzésében a $w(e)$ élsúlyokról nem követeltük meg, hogy nemnegatívak legyenek, $w(e) < 0$ megengedett. Első látásra ez értelmetlennek tűnhet, hiszen akár a térbeli, fizikai távolságokra, akár a menetidőre gondolunk, ezek nyilván nem lehetnek negatívak. Ennek ellenére, több olyan alkalmazása is van a feladatnak, amiben negatív élsúlyok természetes módon merülnek fel. Tegyük fel például, hogy a G irányított gráf élei kémiai reakcióknak, a csúcsok a rendszer lehetséges állapotainak felelnek meg, az e él $w(e)$ súlya pedig az e reakcióhoz szükséges energia; az $e = (u, v)$ él tehát azt fejezi ki, hogy az u állapotból a v-be juthatunk az e reakcióval és ehhez $w(e)$ energia szükséges. Ekkor $w(e) < 0$ lehetséges akkor, ha az e reakció során energia szabadul fel. Ebben az alkalmazásban a legrövidebb út feladat akkor merül fel, ha arra vagyunk kíváncsiak, hogy a rendszer s állapotából mekkora minimális energiabefektetéssel lehet eljutni a többibe. Ehhez hasonló példa, ha $w(e)$ egy elektromos autó fogyasztását méri az e útszakaszon: ekkor a visszatöltés miatt $w(e)$ negatív is lehet, ha e meredeken lejt.

Egy fontos speciális esetben már adtunk hatékony algoritmust a legrövidebb út problémára: ha minden e élre $w(e) = 1$ (vagyis egy P út hossza egyszerűen a P éleinek a száma), akkor a 2. fejezetben látott BFS algoritmussal megoldható a feladat. Ráadásul a BFS nagyon hatékony is: ha G szomszédsági listával van megadva, akkor a lépésszáma legföljebb $c \cdot (n+m)$, ahol n, illetve m a G csúcsainak, illetve éleinek a száma és c egy alkalmas konstans.

Ebben a fejezetben a feladat általános esetével foglalkozunk, amiben w tetszőleges lehet. Látni fogjuk, hogy bár $c \cdot (n+m)$ futásidejű megoldást nem tudunk adni, de a feladat még az általános esetben is megoldható polinomiális algoritmussal – legalábbis akkor, ha a bemenetre egy alapvető és a gyakorlati alkalmazásokban rendre teljesülő feltétel fennáll. Ez az utóbbi kitétel valóban fontos: ha semmilyen kiegészítő feltételt nem szabunk a bemenetre, akkor a legrövidebb utak meghatározására nem ismert polinomiális algoritmus és valószínűleg nem is létezik ilyen.

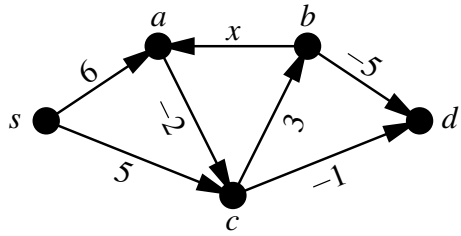
Megemlítünk két, szorosan ide kapcsolódó problémát. Az első a Hamilton-út keresése (vagy a létezésének az eldöntése) egy G gráfban, amiről már említettük, hogy nem ismert rá polinomiális algoritmus és – noha ez nem bizonyított, de – valószínűleg nem is létezik ilyen. A második pedig a leghosszabb út feladat, ami analóg a legrövidebb út feladattal: egy élsúlyozott irányított gráf adott s csúcsából leghosszabb utat kell keresni a v csúcsba.

A legrövidebb, illetve a leghosszabb utak keresése (teljesen általános feltételek mellett) ekvivalens feladatok, hiszen ha minden él $w(e)$ súlyát az ellentettjére változtatjuk, akkor a két feladat egymásba megy át. A Hamilton-út probléma pedig ennek a két feladatnak a speciális esetének tekinthető: ha minden él hossza 1, akkor a Hamilton-utak azonosak az $n-1$ hosszú utakkal (ahol n a csúcsok száma)

és persze ennél hosszabb út nem is létezhet a gráfban. (Vagy ekvivalens megfogalmazásban: ha minden élre $w(e) = -1$, akkor a Hamilton-utak az $1-n$ hosszú utak, aminél rövidebb biztosan nincs a gráfban.)

Így nem meglepő, hogy a leghosszabb, illetve (általános feltételek között) a legrövidebb út problémára nem várhatunk polinomiális algoritmust: ha ilyen létezne, akkor (azt minden csúcsból lefuttatva) a Hamilton-út feladatot is meg lehetne oldani hatékonyan – szemben az ezzel kapcsolatos, általánosan elfogadott sejtéssel.

Mi is tehát az a feltétel, amivel a legrövidebb út probléma hatékonyan megoldhatóvá válik? Fentebb említettük, hogy negatív élsúlyok előfordulhatnak a bemenetben (és ezt egyes gyakorlati alkalmazások ki is használják). Ez azonban felvet egy fontos, tisztázásra szoruló kérdést. Tekintsük például a 11.1. ábrán látható G irányított gráfot az élekre írt $w(e)$ élsúlyokkal és tegyük fel, hogy $x = -2$. Ekkor G-ben a (c,b,a,c) csúcsokat ebben a sorrendben érintő C irányított körben az élsúlyok összege -1, vagyis negatív. Ebből az következik, hogy s-ből bármelyik másik csúcsba tetszőlegesen kicsi (vagyis abszolút értékben nagy, de negatív) összsúlyú élsorozat létezik – ehhez csak kellően sokszor körbe kell járni C-t. Ha például s-ből d-be mínusz egymillió hosszú élsorozatot szeretnék, akkor s-ből c-be lépünk, onnan (10^6+4)-szer körbejárjuk C-t, majd c-ből d-be érkezünk; és mínusz egymillió helyett ez nyilván tetszőleges (4-nél kisebb) számra ugyanígy működne.



11.1. ábra

Első látásra talán azt hihetnénk, hogy ez a jelenség megkérdőjelezi a legrövidebb út feladat létjogosultságát – hiszen nincs értelme legkisebbet keresni egy alulról nem korlátos számhalmazban. Valójában azonban a C kört tartalmazó élsorozatok nem utak, hiszen egy P út minden csúcsot legföljebb egyszer érinthet. (Az élsorozat és az út definíciója irányított esetben is analóg azzal, ahogyan ezeket a fogalmakat irányítatlan gráfokra az 1.10. Definícióban értelmeztük.) Így az előző bekezdésben csak azt mutattuk meg, hogy az s-ből d-be vezető *élsorozatok* hosszai között nincs minimális; legrövidebb *útnak* viszont kell léteznie, hiszen a csúcsok számának végessége miatt az utak száma is az, így kell legyen köztük minimális hosszúságú. Ennek ellenére, a fenti példában látott jelenség rámutat arra a feltételre, ami a legrövidebb út problémát hatékonyan megoldhatóvá teszi.

11.1. Definíció. *A $G = (V,E)$ irányított gráf élein a $w : E \to \mathbb{R}$ súlyfüggvényt* konzervatívnak *nevezzük, ha G minden irányított körének éleire a $w(e)$ értékek összege nemnegatív.*

Így a fenti ábra gráfjában az $x = -2$ választással kapott w súlyfüggvény nem konzervatív (mert létezik -1 összsúlyú irányított kör). Ha viszont $x \geq -1$, akkor w konzervatív (mert a gráfban csak egyetlen irányított kör létezik, amiben az élek összhossza ebben az esetben már nemnegatív).

A továbbiakban a legrövidebb út probléma bemenetéről mindig fel fogjuk tenni, hogy w konzervatív – és látni fogjuk, hogy ezzel a feltevéssel a feladat már hatékonyan megoldhatóvá válik. A súlyfüggvény konzervativitása teljesül a két fentebb említett, kémiai reakciókról, illetve az elektromos autók fogyasztásáról szóló példában is: ez mindkét esetben az energiamegmaradás törvényéből következik. Általában is elmondható, hogy a legrövidebb út feladat gyakorlati alkalmazásai közül azokban, amikben negatív élhosszok előfordulnak, w konzervativitása legtöbbször a feltételekből természetesen adódik. Ha viszont $w(e) \geq 0$ minden e élre, akkor w konzervativitása magától értetődő; ezzel a fontos speciális esettel a 11.2. szakaszban foglalkozunk.

Érdemes röviden kitérni arra, hogy mit jelent w konzervativitása irányítatlan gráfok esetén. A fejezet elején említettük, hogy ilyenkor az élek két irányítottal való helyettesítésével a feladat visszavezethető az irányított esetre. Ez viszont azzal jár, hogy ha az $e = \{u, v\}$ irányítatlan élre $w(e) < 0$ teljesülne, akkor a belőle keletkezett, szintén $w(e)$ hosszúságú $e_1 = (u, v)$ és $e_2 = (v, u)$ irányítatlan élek egy $2 \cdot w(e) < 0$ összsúlyú irányítatlan kört alkotnának – vagyis az irányított változatban kapott w nem volna konzervatív. Következésképpen irányítatlan gráfok esetében a w konzervativitása egyszerűen azt jelenti, hogy minden e élre $w(e) \geq 0$ és a legrövidebb út problémára is csak ilyen esetekben adunk algoritmust (lásd a 11.2. szakaszt).

Térjünk vissza a fenti ábra példájára: láttuk, hogy az $x = -2$ esetben különbség volt az s-ből d-be vezető legrövidebb út és élsorozat hossza között – illetve élsorozatból nem is létezett legrövidebb. Az alábbi állítás azt mondja ki, hogy konzervatív w-re ez a különbség eltűnik.

11.2. Állítás. *Legyen adott a $G = (V, E)$ irányított gráf, az $s, v \in V(G)$, $s \neq v$ csúcsok és G élein a $w : E \to \mathbb{R}$ konzervatív súlyfüggvény. Ekkor ha létezik s-ből v-be egy t összsúlyú, k élből álló Q élsorozat, akkor létezik s-ből v-be egy legfeljebb t összsúlyú és legfeljebb k élű P út is.*

Bizonyítás: Ha Q eleve irányított út, akkor az állítás magától értetődően igaz. Ha nem, akkor létezik olyan csúcs, amit Q egynél többször érint. Keressünk Q-ban egy olyan ismétlődő csúcspárt, amik között Q mentén haladva az élek számában mért távolság a lehető legkisebb; jelölje ezt a csúcsot u és Q-nak az u legközelebbi előfordulásai közötti szakaszát C. Ekkor C persze egy u-ból u-ba vezető zárt élsorozat, de ennél több is igaz rá: irányított kör kell legyen. Valóban, ha nem az volna, akkor létezne olyan w csúcs, amit C egynél többször érint, így w előfordulásai között kisebb volna a távolság Q élei mentén haladva, mint C élszáma – ez pedig u és C választása miatt lehetetlen.

Vágjuk ki Q-ból a C irányított kört (abban az értelemben, hogy Q-nak ebből az u-tól u-ig tartó szakaszából csak maga az u csúcs marad meg), a kapott élsorozat legyen Q_1. Ekkor Q_1 is s-ből v-be vezet, az éleinek a száma k-nál kisebb, az összsúlya

pedig legföljebb t, mert w konzervativitása miatt C összsúlya (vagyis a kivágott e élek $w(e)$ súlyainak az összege) nemnegatív volt.

Ha Q_1 sem irányított út, akkor Q_1 is tartalmaz ismétlődő csúcsokat, így a fentiek szerint ismét kivágható belőle egy (nemnegatív összsúlyú) irányított kör. Ezt a műveletet ismételve végül a P irányított úthoz jutunk, hiszen az élsorozat élszáma folyamatosan csökken. Mivel az összsúlya a fentiek szerint (w konzervativitása miatt) nem nőhetett, ezzel az állítást beláttuk. □

Érdemes összevetni az 1.11. Állítást a fentivel: az utóbbi az előbbi irányított, súlyozott változatának tekinthető, így nem csoda, hogy a bizonyításuk is nagyon hasonló. A 11.2. Állításból valóban következik, hogy az s-ből v-be vezető legrövidebb út és élsorozat hossza egyenlő, hiszen minden út egyben élsorozat is, az élsorozatokból pedig lehet legföljebb ugyanolyan hosszú utat készíteni. Az állítás alábbi következménye pedig nélkülözhetetlen lesz a legrövidebb út problémára vonatkozó algoritmusok helyességének az igazolásához; ez azt állítja, hogy konzervatív súlyfüggvény esetén a legrövidebb utak kezdőszakaszai is mindig legrövidebb utak.

11.3. Következmény. *Legyen adott a $G = (V,E)$ irányított gráf, az $s,v \in V(G)$ csúcsok és G élein a $w : E \to \mathbb{R}$ konzervatív súlyfüggvény. Ha az s-ből v-be vezető P legrövidebb út áthalad az u csúcson, akkor P-nek az s-től u-ig tartó P_u szakasza egy s-ből u-ba vezető legrövidebb út.*

Bizonyítás: Tegyük fel indirekt, hogy létezik s-ből u-ba egy P_u-nál rövidebb P' út. Ekkor s-ből P' mentén u-ig haladva, majd onnan P-nek az u-tól v-ig tartó szakaszán tovább folytatva egy s-ből v-be vezető Q élsorozatot kapunk. (Q nem feltétlen út, mert P'-nek lehet közös éle vagy csúcsa P u-tól v-ig tartó szakaszával.) Mivel P_u-t a nála rövidebb P'-vel helyettesítettük, ezért Q összsúlya kisebb P-énél. Így a 11.2. Állítás miatt létezik a gráfban egy Q összsúlyánál nem hosszabb, vagyis P-nél rövidebb út is s-ből v-be, ami ellentmondás (mert P legrövidebb út s-ből v-be). □

Figyeljük meg, hogy a következmény állításában nélkülözhetetlen feltétel w konzervativitása, anélkül nem volna igaz. Legyen például a 11.1. ábra gráfjában most $x = -3$; ekkor persze a súlyfüggvény nem konzervatív (mert van -2 összhosszúságú irányított kör). Könnyű végiggondolni, hogy s-ből a-ba a legrövidebb út az 5 hosszúságú $s \to c \to b \to a$ út. Azonban ennek a c-ig tartó, 5 hosszúságú szakasza (vagyis az (s,c) élből álló út) nem legrövidebb út s-ből c-be, mert az $s \to a \to c$ út rövidebb ennél, csak 4 hosszúságú.

A legrövidebb út probléma kitűzésekor említettük, hogy a feladat természetesen nem csak a táv(v)-k (vagyis az s-ből v-be vezető legrövidebb út hosszának) meghatározásából áll, hanem minden v csúcsra egy legrövidebb utat is meg kell tudni adni. A 11.3. Következménynek nagyon hasznos folyománya, hogy ennek érdekében (konzervatív súlyfüggvény esetén) követhetjük azt a BFS eljárásnál már alkalmazott ötletet, hogy minden $v \neq s$ csúcsra megadunk egy olyan, előző(v)-vel jelölt csúcsot, ami közvetlenül megelőzi v-t egy s-ből v-be vezető P legrövidebb úton; az előző(v) mutatók ismeretében pedig azok mentén v-től s-ig visszafelé lépkedve kaphatunk egy legrövidebb utat s-ből v-be. Ennek az eljárásnak a helyes működéséhez

viszont szükséges, hogy P-nek az előző(v)-ig tartó szakasza is legrövidebb út legyen s-ből előző(v)-be – ami a 11.3. Következményből adódik. Ennek a fontosságát kiemelendő érdemes ismét az előző bekezdés példáját (tehát a 11.1. ábrát az $x = -3$ esetben) idézni: itt az s-ből c-be vezető (egyetlen) legrövidebb út $s \to a \to c$, így előző$(c) = a$ volna; így ha az s-ből a-ba vezető $s \to c \to b \to a$ legrövidebb utat a-tól visszafelé lépkedve az előző(v) mutatók mentén próbálnánk felderíteni, akkor c-nél „eltévednénk”, a mutató s helyett a-ba vinne vissza. A fentiek szerint tehát konzervatív súlyfüggvény esetén ilyen probléma szerencsére nem adódhat.

11.1. A Bellman-Ford algoritmus

Ebben a szakaszban bemutatunk egy Richard E. Bellman (1920 – 1984) és (a 9. fejezetben már említett) Lester R. Ford (1927 – 2017) amerikai matematikusokról elnevezett (bár röviddel előttük, 1955-ben Alfonso Shimbel (1923 – 1987), szintén amerikai matematikus által már publikált) algoritmust, ami hatékony megoldást ad a legrövidebb út problémára abban az esetben, ha a w hosszfüggvény konzervatív.

Az eljárás alapja egy sok más helyzetben is alkalmazható algoritmustervezési technika: a megoldandó feladatot egy problémasorozat végpontjaként fogjuk fel, ami több, kisebb feladatból áll. Ahhoz hasonlítható ez, mint amikor egy magas pontra (például a háztetőre) való feljutás érdekében nem helyből próbálunk nagyot ugrani, hanem lépcsőt építünk: a lépcsőfokok felelnek meg a kisebb feladatoknak, amik egymás utáni elvégzésével az eredetileg kitűzött probléma megoldásához jutunk. Persze egyáltalán nem magától értetődő, hogy ezeket a részfeladatokat hogyan alkossuk meg: ezt úgy kell megtenni, hogy mindegyik részfeladat viszonylag könnyen megoldható legyen az előző megoldásának a birtokában – vagyis bármelyik lépcsőfokról fel tudjunk lépni a következőre. A részfeladatok megalkotása gyakran szép ötleteket, kreativitást igényel; de ha sikerül őket jól megválasztani, akkor ez a probléma hatékony megoldásához vezethet. (Ezt az általános technikát *dinamikus programozásnak* nevezik és az *Algoritmuselmélet* tárgyban bővebben is esik róla szó.)

A legrövidebb út probléma esetében a következő egyszerű ötlet adja a fenti bekezdés szerinti lépcsőfokokat: a feladat egy adott bemenete esetén jelölje minden $v \in V(G)$ csúcsra és $k \geq 0$ egészre táv$_k(v)$ a legrövidebb olyan, s-ből v-be vezető irányított út hosszát, ami legfeljebb k élből áll; ha pedig ilyen út nincs, akkor legyen táv$_k(v) = \infty$. Ekkor a táv$_0(v)$ értékek könnyen megkaphatók, hiszen ezek a nulla élű (vagyis csak az s csúcsból álló) utaknak felelnek meg: táv$_0(s) = 0$ és minden $v \neq s$ csúcsra táv$_0(v) = \infty$. A legrövidebb út feladat megoldását adó táv(v) értékek pedig minden v csúcsra táv$_{n-1}(v)$-vel egyenlők (ahol n a gráf csúcsainak a száma), hiszen minden út legfeljebb $n-1$ élből állhat (mert az utak minden csúcsot csak egyszer érinthetnek). Az alábbi állításból az derül ki, hogy minden $k \geq 1$ esetén a táv$_k(v)$ értékek könnyen kiszámíthatók a táv$_{k-1}(v)$ értékekből – vagyis az előző bekezdés hasonlatát használva a lépcsősort jól sikerült megépíteni, mert mindegyik fokról könnyű fellépni a következőre, a végpont pedig az eredetileg megcélzott háztető.

Mivel a táv$_k(v)$ értékek a fentiek szerint felvehetik a végtelen értéket is, ezért kényelmi okokból fogadjuk el a következő, a ∞ elemmel való alapműveletekre vo-

natkozó egyszerű megállapodásokat, amiket a legrövidebb út feladatra vonatkozó további algoritmusokban is alkalmazni fogunk: $\infty + x = \infty$ minden $x \in \mathbb{R}$ esetén, $\min \emptyset = \infty$ (vagyis az üres halmaz minimuma végtelen), és egy (véges elemszámú) $H \subseteq \mathbb{R} \cup \{\infty\}$ halmaz minimumát a ∞ elemek nem befolyásolják, vagyis H minimuma egyenlő a H ∞-től különböző elemeinek a minimumával. (Mindebből az is következik, hogy $\min\{\infty\} = \infty$, hiszen $\min\{\infty\} = \min \emptyset = \infty$.)

11.4. Állítás. *Legyen adott a $G = (V,E)$ irányított gráf, az $s, v \in V(G)$, $v \neq s$ csúcsok és G élein a $w : E \to \mathbb{R}$ konzervatív súlyfüggvény. Ekkor minden $k \geq 1$ esetén*

$$\textit{táv}_k(v) = \min\{\textit{táv}_{k-1}(u) + w(e) : e = (u,v), e \in E(G)\}.$$

Az állítás szerint tehát $\text{táv}_k(v)$-t úgy kaphatjuk meg, hogy minden v-be belépő $e = (u,v)$ élre kiszámítjuk a $\text{táv}_{k-1}(u) + w(e)$ összeget és az így kapott értékeknek a minimumát vesszük; ha pedig G-nek nincs v-be belépő éle (ami lehetséges, bár az alkalmazások szempontjából érdektelen), akkor $\text{táv}_k(v) = \infty$. Az állítás a $\text{táv}_k(s)$ értékekről nem szól, de erre nincs is szükség: nyilván $\text{táv}_k(s) = 0$ minden $k \geq 0$ esetén, mert s-ből s-be csak az egyedül s-ből álló (egy csúcsú és él nélküli) út vezet.

A 11.4. Állítás bizonyítása: Jelölje minden $v \neq s$ csúcsra $\text{táv}'_k(v)$ az s-ből v-be vezető, legfeljebb k élből álló élsorozatok közül a legrövidebbnek a hosszát (vagyis az éleinek az összsúlyát); illetve legyen $\text{táv}'_k(v) = \infty$, ha ilyen élsorozat nem létezik. Más szóval: $\text{táv}_k(v)$ és $\text{táv}'_k(v)$ értelmezése között az egyetlen különbség az, hogy az utóbbiban élsorozatokat is figyelembe veszünk, míg az előbbiben csak utakat. A bizonyítás alapgondolata annak a megmutatása, hogy az állításban szereplő egyenlet a $\text{táv}_k(v)$ értékek helyett a $\text{táv}'_k(v)$ értékekre mindig teljesül – még akkor is, ha a w súlyfüggvény nem feltétlen konzervatív. Ha azonban w konzervatív, akkor a 11.2. Állításból következően $\text{táv}_k(v) = \text{táv}'_k(v)$ minden v csúcsra, így ebből az állítás következni fog.

Tegyük fel először, hogy s-ből v-be nem vezet legfeljebb k élű élsorozat, vagyis $\text{táv}'_k(v) = \infty$. Ekkor a v-be belépő $e = (u,v)$ élekre $\text{táv}'_{k-1}(u) = \infty$ kell teljesüljön, mert ha létezne s-ből u-ba legfeljebb $k-1$ élű élsorozat, akkor ezt e-vel megtoldva egy v-be vezető, legfeljebb k élű élsorozatot kapnánk. Ezért ebben az esetben az állításbeli egyenlet jobb oldalán a $\infty + w(e) = \infty$ értékek minimumát kell venni (vagy az üres halmazét, ha G-nek nincs a v-be belépő éle), így az állítás igaz.

A továbbiakban tehát feltehetjük, hogy s-ből v-be vezet legfeljebb k élű élsorozat. Osztályozzuk ezeket az utolsó élük szerint: minden $e = (u,v)$ élre az S_e halmaz álljon azokból a legfeljebb k élű, s-ből v-be vezető élsorozatokból, amiknek az utolsó éle e. Ha egy $e = (u,v)$ élre Q az S_e-beli élsorozatok között a legrövidebbek egyike, akkor Q-nak az s-től u-ig tartó Q' szakasza egy s-ből u-ba vezető, legfeljebb $k-1$ élű és legrövidebb, vagyis $\text{táv}'_{k-1}(u)$ hosszúságú élsorozat. Valóban, Q' nyilván legfeljebb $k-1$ élű és ha létezne s-ből u-ba egy Q'-nél rövidebb és legfeljebb $k-1$ élű élsorozat, akkor ezt e-vel megtoldva egy Q-nál rövidebb és legfeljebb k élű, S_e-beli élsorozatot kapnánk. Következésképp Q hossza $\text{táv}'_{k-1}(u) + w(e)$. Ebből pedig az állítás már adódik, mert ha minden $e = (u,v)$ élre az S_e-beli élsorozatok

közül a legrövidebb hosszát meghatározzuk és a kapott értékek minimumát vesszük, akkor nyilván $\text{táv}'_k(v)$-t kapjuk. □

Ebből az állításból, illetve az előtte írtakból az alábbi algoritmus adódik a legrövidebb út feladatra. Az eljárást úgy érdemes elképzelni, hogy sorról sorra kitöltünk egy táblázatot, aminek az oszlopait a csúcsokkal indexeltük, a sorait pedig az egyre növekvő k értékekkel: a k-adik sorban (amennyiben a sorok számozását nullától kezdjük) a v csúcsnak megfelelő oszlopban $\text{táv}_k(v)$ áll. A k-adik sor elemeinek kiszámításához mindig csak a $(k-1)$-edik sor ismeretére van szükség, az $(n-1)$-edik sorban pedig már a kimenetet, a $\text{táv}(v)$ értékeket kapjuk.

BELLMAN-FORD ALGORITMUS

Bemenet: Egy n csúcsú $G=(V,E)$ irányított gráf, egy $w : E \to \mathbb{R}$ súlyfüggvény és egy $s \in V$ csúcs

```
1   táv_0(s) ← 0; minden v ∈ V, v ≠ s-re táv_0(v) ← ∞
2   minden v ∈ V, v ≠ s-re előző_0(v) ← *
3   ciklus: k fut 1-től (n−1)-ig
4       táv_k(s) ← 0
5       ciklus: v végigfut a V(G) \ {s} csúcshalmazon
6           táv_k(v) ← táv_{k−1}(v)
7           előző_k(v) ← előző_{k−1}(v)
8           ciklus: e = (u,v) végigfut a v-be belépő éleken
9               ha táv_k(v) > táv_{k−1}(u) + w(e), akkor:
10                  táv_k(v) ← táv_{k−1}(u) + w(e)
11                  előző_k(v) ← u
12          ciklus vége
13      ciklus vége
14  ciklus vége
15  Minden v ∈ V(G) csúcsra: táv(v) ← táv_{n−1}(v), előző(v) ← előző_{n−1}(v)
```

Említettük már, hogy az algoritmusnak a $\text{táv}(v)$-k mellett egy $\text{előző}(v)$-vel jelölt csúcsot is meg kell határoznia minden $v \neq s$ esetén, ami közvetlenül megelőzi v-t egy s-ből v-be vezető legrövidebb úton. Ennek érdekében a fenti pszeudokódot kiegészítettük az $\text{előző}(v)$-knek megfelelő adattípussal: minden $v \neq s$ esetén $\text{előző}_k(v)$ jelöli a v-t megelőző csúcsot egy s-ből v-be vezető olyan, legföljebb k élű úton, ami legrövidebb az ilyen utak között; ha pedig ilyen út nincs, akkor $\text{előző}_k(v) = *$. Azonban az $\text{előző}_k(v)$-k meghatározása igényel némi körültekintést: hiba volna azt feltételezni, hogy $\text{előző}_k(v)$ lehet bármelyik olyan u csúcs, amire a 11.4. Állításbeli minimum felvétetik (vagyis amire $\text{táv}_k(v) = \text{táv}_{k-1}(u) + w(e)$ az $e=(u,v)$ élre). Ugyanis a 11.4. Állítás bizonyításából csak az derül ki, hogy egy ilyen esetben u lehet a v-t megelőző csúcsa az s-ből v-be vezető, legföljebb k élű élsorozatok közül a legrövidebbek egyikének – de az előfordulhat, hogy ez az élsorozat nem út (erre a 11.5. Feladatban látunk is majd példát). Ezt a pszeudokódban úgy kerültük ki, hogy abban az esetben, ha $\text{táv}_k(v) = \text{táv}_{k-1}(v)$ – vagyis a k élű utak között nincs

rövidebb a legföljebb $k-1$ élűek közötti legrövidebbnél –, akkor előző$_k(v)$-n sem változtattunk előző$_{k-1}(v)$-hez képest. Ha viszont táv$_k(v) <$ táv$_{k-1}(v)$ – vagyis a k élű utak között van rövidebb a legföljebb $k-1$ élűeknél – akkor előző$_k(v)$ már lehet bármelyik olyan $e=(u,v)$ él kezdőpontja, amire a 11.4. Állításban írt minimum felvétetik. Az utóbbi esetben ugyanis az s-ből v-be vezető, k élű és táv$_k(v)$ hosszúságú élsorozatok mind utak kell legyenek, különben a 11.2. Állítás (illetve annak a bizonyítása) szerint mégis létezne k-nál kevesebb élű és táv$_k(v)$ hosszú út s-ből v-be és így táv$_k(v)=$ táv$_{k-1}(v)$ következne.

A fenti pszeudokódban táv$_k(v)$ értékének a 11.4. Állítás szerinti kiszámítása a 6. sorbeli értékadással és a 8-12. sorokban zajló ciklussal történik. A 6. sorban táv$_k(v)$-t akár ∞-nek is inicializálhatnánk, a 8-12. sorban zajló ciklus végére táv$_k(v)$ értéke így is helyes volna. Két okból nem teszünk mégsem így: egyrészt táv$_k(v) \leq$ táv$_{k-1}(v)$ úgyis igaz (hiszen a legföljebb $k-1$ élű utak egyben legföljebb k élűek is); másrészt ezzel elronthatnánk az előző$_k(v)$ csúcsok meghatározását. Ugyanis ezeket az előző bekezdésben írtaknak megfelelően határozzuk meg: előző$_k(v)$-t a 7. sorban előző$_{k-1}(v)$-nek inicializáljuk, majd a 8-12. sorokban zajló ciklusban csak akkor változtatjuk meg, ha van olyan v-be belépő $e=(u,v)$ él, amire táv$_{k-1}(u)+w(e)$ kisebb táv$_{k-1}(v)$-nél; ebben az esetben pedig előző$_k(v)$ egy olyan $e=(u,v)$ él kezdőpontja lesz, amire táv$_{k-1}(u)+w(e)$ minimális (és ez a minimum egyben táv$_k(v)$ értéke is lesz).

Megjegyezzük, hogy az algoritmus futtatása során előfordulhat, hogy valamilyen $k<n-1$ értékre táv$_k(v)=$ táv$_{k-1}(v)$ teljesül minden v csúcsra. Ha ez a helyzet előáll, akkor az eljárás megállítható a 3-14. sorokban zajló ciklus k-nak megfelelő végrehajtása után, hiszen a táv$_k(v)$ és előző$_k(v)$ értékek nyilván a későbbiekben sem változhatnának. Ezzel a kiegészítéssel tehát egyes, szerencsés bemenetekre gyorsítható az algoritmus.

11.5. Feladat. Határozzuk meg a Bellman-Ford algoritmussal a 11.1. ábrán látható gráfban az $x=-1$ választással minden v csúcsra az s-ből v-be vezető legrövidebb út táv(v) hosszát és adjunk meg egy s-ből d-be vezető legrövidebb utat.

Megoldás: Lefuttatva az algoritmust az alábbi adatok keletkeznek:

		$v \mapsto$ táv$_k(v)$				$v \mapsto$ előző$_k(v)$			
k	s	a	b	c	d	a	b	c	d
0	0	∞	∞	∞	∞	$*$	$*$	$*$	$*$
1	0	6	∞	5	∞	s	$*$	s	$*$
2	0	6	8	4	4	s	c	a	c
3	0	6	7	4	3	s	c	a	c
4	0	6	7	4	2	s	c	a	b

A $k=0$-nak megfelelő sor kitöltése az eljárás 1-2. sorai szerint történt. A $k=1$-nek megfelelő sorban az eljárás 4. sora szerint táv$_1(s)=0$, majd táv$_1(a)$ meghatározása (a 6. sorbeli táv$_1(a)=$ táv$_0(a)=\infty$ inicializálás után) a táv$_1(a)=\min\{0+6,\infty-1\}=6$ minimumszámítással történik, ami a 8-12. sorok

között zajlik. Ebből előző$_1(a) = s$ is adódik, ami az (s,a) él vizsgálatakor, a 11. sorban állítódik be. A további számolásokat nem részletezzük, de egy pontra még érdemes kitérni: táv$_4(a)$ számításakor először a táv$_4(a) =$ táv$_3(a) = 6$ értéket állítjuk be, majd a $\min\{0+6, 7-1\} = 6$ számítást végezzük el, amiből táv$_4(a)$ értéke 6 marad és így előző$_4(a) =$ előző$_3(a) = s$. Ez a példa mutatja a fentebb említett körültekintés szükségességét az előző$_k(v)$ mutatókkal kapcsolatban: bár az $e = (b,a)$ élre is táv$_4(a) =$ táv$_3(b) + w(e)$ teljesül, mégis hiba volna az előző$_4(a) = b$ választás, mert így b-ből az előző$_4(v)$ mutatók mentén visszafelé lépkedve az (a,b,c,a) csúcsokat érintő (nulla összhosszúságú) C irányított kört járnánk be. Ebben az esetben tehát b nem lehet az a-t megelőző csúcs egy s-ből a-ba vezető legrövidebb úton – csak egy legrövidebb élsorozaton (ami az (s,a) él után bejárja a C irányított kört).

A táblázat utolsó sorában látható táv$_4(v)$, illetve előző$_4(v)$ értékek az eljárás 15. sora szerint egyenlők a kimenetet alkotó táv(v), illetve előző(v) értékekkel (noha ezt külön nem írtuk ki). Az előző(v) mutatók mentén d-ből visszafelé haladva sorra a b, c, a és s csúcsokhoz jutunk, amiből az $s \to a \to c \to b \to d$ utat kapjuk; ennek a hossza pedig valóban táv$(d) = 2$. □

11.1.1. A Bellman-Ford algoritmus lépésszáma

Jelölje a bemenetet adó G gráf csúcsainak, illetve éleinek a számát továbbra is n, illetve m. Mivel az eljárás 8-12. sorában zajló ciklus a v-be belépő éleket vizsgálja végig, ezért a Bellman-Ford algoritmus hatékony implementálásához érdemes G-t az 1.7. szakaszban látott szomszédsági listának egy olyan változatával tárolni, amiben a v csúcshoz tartozó lista a v-be belépő éleket sorolja fel. (Ugyanis az 1.7. szakaszban leírt eredeti változatban v listája a v-ből kilépő éleket tartalmazta – de egyrészt ez a különbség nyilván jelentéktelen, másrészt a kétféle szomszédsági lista közötti konverzió m-mel arányos lépésszámban könnyen elvégezhető.) Ekkor a 8-12. sorokban zajló ciklus végrehajtásához csak végig kell lépkedni a v-hez tartozó listán; mivel a lista minden eleménél, vagyis a v-be belépő élek mindegyikénél konstansnyi időt kell csak eltölteni a szükséges számításokkal, ezért ez v listájának a hosszával arányos lépésszámban megtehető. Így az 5-13. sorokban zajló ciklus az (s-től különböző) csúcsok listáinak az összhosszával, vagyis legföljebb m-mel arányos lépésszámot igényel. Mivel ezt a ciklust a 3-14. sorokban zajló ciklus belsejében $(n-1)$-szer kell lefuttatni, adódik, hogy a Bellman-Ford algoritmus teljes lépésszáma legföljebb $c \cdot n \cdot m$ valamilyen c konstansra – ami azt jelenti, hogy ez egy hatékony, polinomiális algoritmus.

11.1.2. A súlyfüggvény konzervativitása

A Bellman-Ford algoritmusról megtudtuk tehát, hogy hatékony és helyes eredményt is ad – legalábbis akkor, ha a w súlyfüggvény konzervatív. Ez utóbbi feltétel pedig valóban fontos, hiszen ez feltétele volt a 11.4. Állításnak és könnyű példát mutatni arra, hogy nem konzervatív w-re az eljárás nem is működik helyesen. Ha például a 11.1. ábra gráfjában az $x = -2$ választással (amivel persze elrontjuk w konzer-

vativitását) futtatjuk az algoritmust, akkor az a 11.5. Feladatban látotthoz képest annyiban módosul, hogy a táblázat utolsó sorában $\text{táv}_4(a) = \min\{0+6, 7-2\} = 5$ lesz és így $\text{előző}_4(a)$ értéke b-re módosul; ennek ellenére, $\text{táv}_4(a)$ valódi értéke továbbra is 6, mert a gráfban az $s \to a$ és $s \to c \to b \to a$ utakon kívül nincs más út s-ből a-ba, ezeknek a hossza pedig 6. Az is látszik ebből a példából, hogy a hibát az (a,c,b,a) csúcsokat sorra érintő, negatív összhosszúságú kör okozta, hiszen az $\text{előző}_4(v)$ mutatók mentén a-ból visszafelé lépegetve épp ezt járjuk be.

Ez a jelenség azonban felvet egy nagyon lényeges kérdést: honnan tudhatjuk egy tetszőleges bemenet esetében, hogy megbízhatunk-e a Bellman-Ford algoritmus futásából kapott végeredményben? Ha w nem konzervatív, akkor a fenti példa szerint a kimenet lehet hibás. G összes irányított körét végigpróbálgatni (és rendre megvizsgálni, hogy az összsúlyuk negatív-e) pedig reménytelen volna, mert ezeknek a száma exponenciális lehet a gráf csúcsszámában mérve. Szerencsére azonban az eljárás minimális kiegészítésével ezt a kérdést is kezelni tudjuk; így végül is a Bellman-Ford algoritmus fogja megválaszolni azt a kérdést is, hogy a saját kimenetének a helyessége garantálható-e.

Ehhez először is vegyük észre, hogy nem minden negatív összsúlyú irányított kör okozhat a fenti példában látotthoz hasonló gondot – csak azoké, amiknek a csúcsai elérhetők s-ből irányított úton. Ha ugyanis egy $v \neq s$ csúcsba nem vezet s-ből út, akkor az algoritmus futtatásakor $\text{táv}_k(v) = \infty$ lesz minden $k \geq 0$ esetén. Valóban, a pszeudokód 1. sora szerint $\text{táv}_0(v) = \infty$ és az eljárás során végig öröklődik $(k-1)$-ről k-ra a $\text{táv}_k(v) = \infty$ érték: mivel a v-be belépő élek kezdőpontjai sem lehetnek s-ből elérhetők (különben v is az volna), ezért ezt a (6. sorban beállított) értéket a 8-12. sorban zajló ciklus sosem írja felül. Így az s-ből nem elérhető v csúcsok $\text{táv}(v) = \text{táv}_{n-1}(v) = \infty$ távolságát nyilván helyesen határozza meg az algoritmus a w súlyfüggvénytől függetlenül. Ezt úgy is fogalmazhatjuk, hogy az s-ből nem elérhető csúcsokra illeszkedő e élek $w(e)$ súlyai érdektelenek, ezek sem az algoritmus futását, sem a helyes kimenetet nem befolyásolják.

Egy másik speciális esetben sem okozhat gondot egy negatív összsúlyú irányított kör: akkor, ha ez tartalmazza s-et. Megfigyelhetjük ugyanis, hogy az s-be belépő élek az algoritmus futásában semmilyen szerepet nem játszanak, ezeket az eljárás soha nem veszi figyelembe. Ez nem is meglepő, hiszen egy ilyen él nem lehet rajta egy s-ből induló úton. Ezért tekinthetjük úgy is, hogy az algoritmust azon a G' gráfon futtatjuk, amiből előzőleg töröltük G összes, s-be belépő élét: ez az eljárás futását és a kimenetét nem befolyásolja. Mivel G'-ben nyilván nem lehetnek s-et tartalmazó irányított körök, ezért a 11.4. Állítás G'-re való alkalmazásából még akkor is következik az algoritmus kimenetének a helyessége, ha G-ben vannak s-et tartalmazó, negatív összsúlyú irányított körök.

Ahhoz tehát, hogy a Bellman-Ford algoritmus kimenetének a helyességében megbízhassunk, arra van szükség, hogy a w súlyfüggvény konzervatív legyen a bemenetként kapott G gráfnak abban a részgráfjában, amit az s-ből elérhető csúcsok és a köztük vezető, de s-től különböző végpontú élek alkotnak. Az alábbi állítás éppen ezt a kérdést válaszolja meg. Felhívjuk rá a figyelmet, hogy mivel ebben w konzervativitását nem feltételezhetjük (hiszen a kérdés éppen az, hogy ez G egy

részgráfjában teljesül-e), ezért itt $\text{táv}_{n-1}(v)$ már nem (feltétlen) az s-ből v-be vezető legrövidebb út hosszát jelöli, hanem az algoritmus által számolt értékeket.

11.6. Állítás. *Legyen adott a Bellman-Ford algoritmus egy bemenete: az n csúcsú G irányított gráf, a $w : E(G) \to \mathbb{R}$ súlyfüggvény és az $s \in V(G)$ csúcs. Jelölje G_s azt a gráfot, amit G-ből kapunk az s-ből irányított úton nem elérhető csúcsok és az s-be belépő élek törlésével. Módosítsuk az algoritmust annyiban, hogy a 3-14. sorokban zajló ciklus magját a $k = n$ értékre is lefuttatjuk (vagyis a 3. sorban k 1-től n-ig fut). Ekkor w akkor és csak akkor konzervatív G_s-en, ha minden $v \neq s$ csúcsra $\text{táv}_n(v) = \text{táv}_{n-1}(v)$ teljesül.*

Bizonyítás: A 11.4. Állítás bizonyításából kiderült, hogy minden w súlyfüggvényre (annak a konzervativitásától függetlenül) az eljárás által számított $\text{táv}_k(v)$ értékek az s-ből v-be vezető, legföljebb k élű élsorozatok hosszainak a minimumával egyenlők minden $v \neq s$ csúcs esetén. Ha w konzervatív G_s-en, akkor a 11.2. Állításból (azt G_s-re alkalmazva) következik, hogy ezek egyben az s-ből v-be vezető, legföljebb k élű utak hosszainak a minimumával is egyenlők. Mivel egy út éleinek a száma legföljebb $n-1$ lehet, ebből $\text{táv}_n(v) = \text{táv}_{n-1}(v)$ valóban következik minden v-re.

Ha viszont w nem konzervatív G_s-en, akkor létezik G_s-ben egy negatív összsúlyú C kör. Ekkor (mint azt korábban a 11.1. ábra gráfjának példáján már láttuk) C csúcsaiba tetszőlegesen kis összsúlyú, s-ből induló élsorozat is készíthető: ha s-ből elmegyünk C egy csúcsáig, majd C-t kellően sokszor körbejárjuk. Ezért ha az algoritmus futtatását képzeletben k tetszőlegesen nagy értékeire is tovább folytatnánk, akkor a $\text{táv}_k(v)$ értékek sosem stabilizálódhatnának. Valóban, ha valamilyen k-ra $\text{táv}_k(v) = \text{táv}_{k+1}(v)$ teljesülne minden $v \neq s$ csúcsra, akkor a $\text{táv}_k(v)$ értékek már később (vagyis k nagyobb értékeire) sem változhatnának. Ez viszont ellentmondana annak, hogy a $\text{táv}_k(v)$ értékek C csúcsaira alulról nem korlátosak. Így ha $\text{táv}_n(v) = \text{táv}_{n-1}(v)$ minden $v \neq s$ csúcsra, akkor w konzervatív kell legyen. □

Ebből az állításból valóban következik, hogy a Bellman-Ford algoritmus kisebb kiegészítésével eldönthetjük, hogy megbízhatunk-e annak a kimenetében. Ehhez csak arra van szükség, hogy a 3-14. sorokban zajló ciklus magját még egyszer, a $k = n$ értékre is lefuttassuk, majd összehasonlítsuk a $\text{táv}_{n-1}(v)$ és a $\text{táv}_n(v)$ értékeket: ha $\text{táv}_n(v) = \text{táv}_{n-1}(v)$ minden v csúcsra igaz, akkor megnyugodhatunk, mert w konzervatív G_s-en és így az eljárás kimenete helyes. Ha viszont legalább egy v-re $\text{táv}_n(v) \neq \text{táv}_{n-1}(v)$, akkor w nem konzervatív G_s-en és így az eljárás által kiszámított $\text{táv}_{n-1}(v)$ értékek nem (feltétlen) egyeznek meg az s-ből v-be vezető legrövidebb utak hosszaival.

11.2. Dijkstra algoritmusa

Bemutatunk egy másik algoritmust is a legrövidebb út problémára. Ez Edsger W. Dijkstra (1930 - 2002) holland matematikustól származik és csak abban az esetben működik helyesen, ha minden él súlya nemnegatív; cserébe viszont a futásideje

jobb a Bellman-Ford algoritmusénál. Bár említettük, hogy negatív élsúlyok is előfordulnak alkalmazásokban, de a gyakorlati élet által produkált helyzetek többsége azért nem ilyen, az élsúlyok nemnegativitása természetes következménye a feladat feltételeinek. Ilyenkor tehát az alább bemutatandó algoritmust érdemes használni.

Dijkstra algoritmusa bizonyos vonásaiban hasonlít a Bellman-Ford algoritmusra: ez is minden $v \in V(G)$ csúcsra állandóan karban tart egy folyamatosan csökkenő táv(v) értéket, amire végig teljesül, hogy ilyen hosszúságú út biztosan létezik s-ből v-be – még ha ez az eljárás leállása előtt nem is feltétlen a legrövidebb. Szemben azonban a Bellman-Ford algoritmus stratégiájával, itt nem a legrövidebb utak élszámának növekedése jelenti az előrelépést, hanem az eljárás keretét adó ciklus magjának minden egyes végrehajtásakor egy (alkalmasan választott) v csúcs táv(v) értéke véglegesedik, így a ciklusmag n-szeri végrehajtása után ez a folyamat megáll. Ezért az algoritmus állandóan karban tart és növel egy (a „kész” szó kezdőbetűje alapján) K-val jelölt halmazt, aminek az elemeire táv(v) már végleges. Az eljárás működőképességének a kulcsát az alábbi állítás adja, ami lehetővé teszi K bővítését. Az állítás mondanivalója valójában egyszerű: ha minden K-n kívüli v csúcsra kiszámítjuk a K-ból v-be érkező $e = (u,v)$ élekre a táv$(u) + w(e)$ értékek $t(v)$-vel jelölt minimumát, akkor egy olyan K-n kívüli a csúcsra, amire ez minimális, $t(a)$ helyesen adja meg táv(a) értékét.

11.7. Állítás. *Legyen adott a $G = (V,E)$ irányított gráf, az $s \in V(G)$ csúcs és a $w : E \to \mathbb{R}^+ \cup \{0\}$ nemnegatív súlyfüggvény.*

- *Jelölje továbbra is minden $v \in V(G)$ csúcsra táv(v) az s-ből v-be vezető legrövidebb út hosszát.*
- *Legyen $K \subseteq V$ egy tetszőleges olyan csúcshalmaz, amire $s \in K$ és $K \neq V$.*
- *Minden olyan $v \notin K$ csúcsra, amibe vezet él K-beli csúcsból, legyen $t(v) = \min\{\text{táv}(u) + w(e) : e = (u,v), u \in K\}$.*
- *Ha a $v \notin K$ csúcsba nem vezet él K-beli csúcsból, akkor legyen $t(v) = \infty$.*
- *Legyen $a \notin K$ olyan csúcs, amire $t(a) = \min\{t(v) : v \notin K\}$.*

Ekkor $t(a) = \text{táv}(a)$.

Bizonyítás: $t(a) = \infty$ csak akkor lehetséges, ha K-ból csak olyan $e = (u,v)$ él lép ki, amire táv$(u) = \infty$, vagyis u nem érhető el s-ből irányított úton. Ebben az esetben nyilván nem létezhet s-ből a-ba vezető út G-ben, így $t(a) = \text{táv}(a) = \infty$ valóban igaz.

Ha viszont $t(a) \neq \infty$, akkor legyen $e = (u,a)$ egy olyan él, amire $u \in K$ és $t(a) = \text{táv}(u) + w(e)$. Ekkor egy s-ből u-ba vezető táv(u) hosszúságú (vagyis legrövidebb) P_u utat e-vel megtoldva egy $t(a)$ hosszúságú Q élsorozatot kapunk a-ba. Bár Q nem feltétlen út (mert a rajta lehet P_u-n), de Q létezéséből a 11.2. Állítás szerint következik egy s-ből a-ba vezető, legföljebb $t(a)$ hosszúságú út létezése is (hiszen az élsúlyok nemnegativitása miatt a konzervativitás nyilván teljesül).

Azt kell még megmutatnunk, hogy a-ba nem vezet $t(a)$-nál rövidebb út. Legyen ezért P egy tetszőleges, s-ből a-ba vezető út és legyen $e = (u,v)$ az első olyan éle P-nek, ami kilép K-ból (vagyis $u \in K$ és $v \notin K$). Ilyen élnek persze kell léteznie,

hiszen $s \in K$ és $a \notin K$. Ekkor P-nek az s-től u-ig tartó szakasza legalább táv(u) hosszúságú (hiszen táv(u) a legrövidebb ilyen út hossza). Ezért P-nek az s-től v-ig tartó szakasza már legalább táv$(u)+w(e)$ hosszúságú, amiből a w nemnegatív értékűsége miatt következik, hogy P is legalább táv$(u)+w(e)$ hosszúságú. Azonban $t(v)$ definíciója miatt táv$(u)+w(e) \geq t(v)$, a választása miatt pedig $t(v) \geq t(a)$. Mindezekből tehát P legalább $t(a)$ hosszúságú, amivel az állítást beláttuk. □

Ebből az állításból közvetlenül kiolvasható egy algoritmus a táv(v)-k meghatározására. Eszerint kezdetben a „kész" csúcsok halmaza lehet $K = \{s\}$ (és persze táv$(s) = 0$), majd az eljárás a K növelésére szolgáló lépést az állításnak megfelelően ismétli egészen addig, amíg $K = V$ nem lesz: K-ból és annak az u elemeire már ismert táv(u) értékekből mindig kiszámítja a $t(v)$ értékeket minden $v \notin K$ csúcsra, majd ezek minimumát veszi és ha ez az a csúcson vétetik fel, akkor a-t áthelyezi K-ba és táv$(a) = t(a)$ lesz. Bár ez az eljárás helyesen működne, de nagyon lassú volna (a futásideje n csúcsú gráfra n^3-bel arányos volna, így lassabb volna például a Bellman-Ford algoritmusnál is). Szerencsére egy egyszerű ötlettel jelentősen felgyorsítható: a $t(v)$ értékek számítása jóval hatékonyabbá tehető azáltal, hogy azokat nem mindig a semmiből számítjuk újra, hanem a ciklusmag eggyel korábbi végrehajtása során kapott értékeik frissítésével nyerjük. Ugyanis a $v \notin K$ csúcsokra szükségtelen az összes K-beli csúcsból v-be érkező élt figyelembe venni $t(v)$ következő számításakor: ha utoljára az a csúcs került K-ba (vagyis a ciklusmag előző végrehajtásakor $t(a)$ értéke volt minimális), akkor elég az a-ból v-be menő élt megvizsgálni. Valóban, ha az a-ból v-be vezet egy $e = (a,v)$ él és $t(v) >$ táv$(a)+w(e)$, akkor $t(v)$ új értéke táv$(a)+w(e)$ lesz. Az így kapott $t(v)$ pedig nyilván megfelel a 11.7. Állítás szerinti értékének. Ha pedig a $v \notin K$ csúcsba nem vezet a-ból él, akkor $t(v)$ értéke nem változik. Ennek az ötletnek az implementálásához csak arra van szükség, hogy nyilvántartsuk azt az $a \in V(G)$ „aktív" csúcsot is, ami utoljára került K-ba.

Dijsktra algoritmusának az alábbi pszeudokódja a fentieken kívül még az előző(v) mutatókat is tartalmazza. Ezeknek a szerepe azonos a korábban (legutóbb a Bellman-Ford algoritmusnál) látottal: egy s-ből v-be vezető legrövidebb utat úgy kaphatunk meg a kimenetből, hogy v-ből az előző(v)-k mentén visszafelé lépkedünk s-ig. Az előző(v)-k számítása a fentieknek megfelelően értelemszerű: minden $v \notin K$ csúcsra előző$(v) = u$ akkor jó választás, ha $u \in K$ és $e = (u,v)$-re $t(v) =$ táv$(u)+w(e)$. Ha pedig a $v \notin K$ csúcsok közül $t(a)$ értéke minimális és ezért a kerül át K-ba, akkor táv(a)-val együtt előző(a) is véglegesedik.

Az alábbi pszeudokódban az egyszerűség kedvéért nem különböztetjük meg a táv(v) és $t(v)$ jelöléseket. Az aktuális K halmaz elemeire táv(v) nyilván mindig a „valódi" értéket jelöli (vagyis egy s-ből v-be vezető legrövidebb út hosszát), a $v \notin K$ csúcsokra pedig a 11.7. Állításbeli $t(v)$ értéknek felel meg (és így felső becslést ad a végleges táv(v)-re).

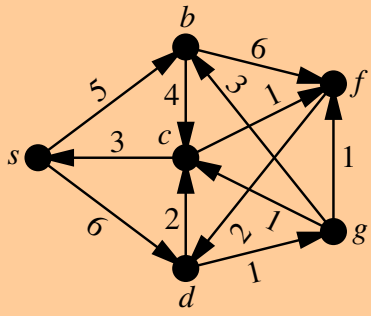
DIJKSTRA ALGORITMUSA

Bemenet: Egy n csúcsú $G=(V,E)$ irányított gráf, egy $w : E \to \mathbb{R}^+ \cup \{0\}$ nemnegatív súlyfüggvény és egy $s \in V$ csúcs

```
1   táv(s) ← 0; minden v ∈ V, v ≠ s-re táv(v) ← ∞
2   minden v ∈ V, v ≠ s-re előző(v) ← *
3   K ← {s}; a ← s
4   ciklus: amíg K ≠ V
5       ciklus: e végigfut az e = (a,v), v ∉ K éleken
6           ha táv(v) > táv(a) + w(e), akkor:
7               táv(v) ← táv(a) + w(e)
8               előző(v) ← a
9       ciklus vége
10      a legyen egy olyan a ∉ K csúcs, amire táv(a) = min{táv(v) : v ∉ K}
11      K ← K ∪ {a}
12  ciklus vége
```

Megjegyezzük, hogy az 5. sor végrehajtásakor előfordulhat, hogy nem létezik olyan $e=(a,v)$ él, amire $v \notin K$; ilyenkor a 6-8. sorokban álló ciklusmagot nyilván egyszer sem kell végrehajtani, az eljárás a 10. sornál folytatódik.

11.8. Feladat. Határozzuk meg Dijkstra algoritmusával az alábbi ábrán látható gráfban minden v csúcsra az s-ből v-be vezető legrövidebb út táv(v) hosszát és adjunk meg egy s-ből f-be vezető legrövidebb utat.



Megoldás: Az algoritmus futása során keletkező adatokat az alábbi táblázat mutatja. A táblázat minden sora a 4-12. sorokban álló ciklus magja egyszeri végrehajtásának felel meg; így az utolsó oszlopban álló, a-val jelölt aktív csúcs ennek a végén változik meg (az eljárás 10. sorának megfelelően), ezért a táblázat következő sorának megfelelő végrehajtás közben ez lesz aktív. A K halmaz aktuális elemeit viszont nem soroljuk fel, mert ez mindig azokból a csúcsokból áll, amik korábban aktív csúcsok voltak (vagyis az utolsó oszlopban az épp aktuális sorig tartó elemekből).

$v \mapsto$ táv(v)						$v \mapsto$ előző(v)					
s	b	c	d	f	g	b	c	d	f	g	a
0	∞	∞	∞	∞	∞	$*$	$*$	$*$	$*$	$*$	s
0	5	∞	6	∞	∞	s	$*$	s	$*$	$*$	b
0	5	9	6	11	∞	s	b	s	b	$*$	d
0	5	8	6	11	7	s	d	s	b	d	g
0	5	8	6	8	7	s	d	s	g	d	c
0	5	8	6	8	7	s	d	s	g	d	f

A ciklusmag utolsó előtti végrehajtásának a végén a két, K-ba nem tartozó csúcsra táv$(c) =$ táv$(f) = 8$, így az eljárás 10. sorában bármelyiket választhatjuk a-nak (a fenti táblázat szerinti végrehajtáskor c-t választottuk).

Figyeljük meg, hogy az eljárás során annak a 6. sorában írt esetvizsgálatot nem minden élre végeztük el: például a táblázat utolsó előtti sorában, amikor $a = g$ az aktív csúcs, az $e = (g,b)$ él nem lép ki K-ból (mert ezen a ponton $K = \{s,b,d,g\}$), így erre nem kell megvizsgálni a táv$(b) >$ táv$(g) + w(e)$ feltételt.

Az előző(v) mutatók mentén f-ből visszafelé lépkedve sorra a g, d és s csúcsokhoz jutunk, amiből a táv$(f) = 8$ hosszúságú $s \to d \to g \to f$ utat kapjuk s-ből f-be. □

Fontos hangsúlyozni, hogy Dijkstra algoritmusa csak nemnegatív élhosszok esetén működik helyesen; ha a gráfnak vannak negatív hosszúságú élei, akkor még abban az esetben is adhat hibás eredményt, ha a súlyfüggvény konzervatív. Ha például a 11.1. ábra gráfjára futtatnánk az eljárást (x tetszőleges értékére), akkor a ciklusmag első végrehajtásakor táv$(a) = 6$, táv$(c) = 5$ és táv$(b) =$ táv$(d) = \infty$ lenne, így c kerülne át K-ba; következésképp az algoritmus kimenetében is táv$(c) = 5$ volna, ami hibás, hiszen s-ből c-be a 4 hosszúságú $s \to a \to c$ út a legrövidebb.

Érdemes röviden kitérni Dijkstra algoritmusának az irányítatlan gráfokra vonatkozó esetére – annak ellenére is, hogy (mint azt már a fejezet elején említettük) megtehetjük, hogy minden irányítatlan élt két irányítottal helyettesítünk és az így kapott bemenetre a fenti algoritmust futtatjuk. A bemenet átalakításánál (és ezzel az élszám megduplázásánál) azonban egyszerűbb és hatékonyabb, ha a fenti pszeudokód szerinti eljárást az eredeti, irányítatlan gráfra futtatjuk azzal az apró különbséggel, hogy az 5. sorban e nem az $e = (a,v)$ irányított éleken fut végig, hanem az $e = \{a,v\}$ irányítatlanokon. Könnyen látható, hogy ezzel az algoritmus futása azonos azzal, ami az élek megduplázása utáni irányított gráfon zajlana.

11.2.1. Dijkstra algoritmusának lépésszáma

Jelölje továbbra is n, illetve m G csúcsainak, illetve éleinek a számát. Mivel a fejezet elején írtak szerint feltételezhetjük, hogy G-nek nincs hurokéle és minden $u,v \in V(G)$, $u \neq v$ csúcspár esetén legfőljebb egy u-ból v-be vezető éle van, ezért $m \leq n(n-1) < n^2$.

Tegyük fel ismét, hogy G szomszédsági listával van megadva (annak is az eredeti, az 1.7. szakaszban látott változatával, amiben a v csúcshoz tartozó lista a v-ből

kilépő éleket sorolja fel). Ekkor az eljárás 5-9. sorában álló ciklus végrehajtásakor csak a a-hoz tartozó listán kell végiglépkedni (és minden egyéb teendő nélkül továbblépni, ha egy olyan $e = (a, v)$ élnél járunk, amire $v \in K$). Mivel az algoritmus teljes futása során minden csúcs legföljebb egyszer tölti be a szerepét (pontosabban: az utolsóként K-ba kerülő csúcs kivételével mindegyik pontosan egyszer), ezért az 5-9. sorokban zajló ciklus végrehajtása a teljes lépésszámhoz összesen csak $c \cdot m$-mel járul hozzá (valamilyen c konstansra), mert a szomszédsági listában a listák összhosszúsága m.

Sajnos azonban a 10. sor végrehajtásaiból származó lépésszám ennél több lehet. Ismert, hogy egy k értéket tartalmazó tömb elemei közül a minimum kiválasztása k-val arányos lépésszámot igényel, mert végig kell vizsgálni mind a k elemet (és közben karban tartani az aktuális minimumot). Mivel a K-ba nem tartozó csúcsok száma kezdetben $n-1$ és ez az algoritmus futása során (a 11. sor végrehajtásakor) mindig eggyel csökken, ezért a minimumkeresések összesen az $(n-1)+(n-2)+\ldots+1 = \frac{n(n-1)}{2}$ összeggel arányos, vagyis n-től négyzetesen függő lépésszámot jelentenek. Összességében tehát azt mondhatjuk (felhasználva azt is, hogy a fentiek szerint $m < n^2$), hogy Dijkstra algoritmusának a lépésszáma legföljebb $c \cdot n^2$ valamilyen c konstansra.

Ez szerencsére jelentősen jobb is lehet a $c \cdot m \cdot n$ futásidejű Bellman-Ford algoritmusnál: ennek a lépésszáma akár n^3-bel arányos is lehet, ha az m élszám eléri az $n(n-1)$-es felső becslésnek például az 1%-át (vagy ehelyett annak tetszőleges rögzített konstansszorosát). Ez pedig nagyobb méretű gráfokra már jelentős mértékben rosszabb lehet Dijkstra algoritmusának az n^2-tel arányos futásidejénél.

Megjegyezzük, hogy Dijkstra algoritmusának a futásideje a $c \cdot n^2$-es felső becsléshez képest is tovább javítható, ha a táv(v) értékeket nem egyszerűen egy, a gráf csúcsaival indexelt tömbben tároljuk, hanem egy olyan, ennél sokkal ravaszabb adatstruktúrát használunk, amivel az eljárás 10. sorában szereplő minimumkiválasztások felgyorsíthatók. Egy ilyen adatszerkezetről (a *kupacról*) és Dijkstra algoritmusának az ezzel való, hatékonyabb implementálásáról az *Algoritmuselmélet* tárgyban esik szó.

A legrövidebb út probléma jól illusztrálja azt, hogy egy algoritmikus probléma megoldásánál alapvető fontosságú az adott célra legmegfelelőbb algoritmus, illetve ehhez a legalkalmasabb adatstruktúra megválasztása. A feladat megoldására írt program futásidejének jelentős mértékű, fölösleges növekedéséhez vezethet például az, ha a Bellman-Ford algoritmust implementáljuk egy olyan esetben, amikor az élsúlyok nemnegatívak. A következő fejezetben pedig meg fogunk ismerni egy olyan eljárást, ami még Dijkstra algoritmusánál is jóval hatékonyabban oldja meg a legrövidebb út problémát abban a – gyakorlati alkalmazások szempontjából – fontos speciális esetben, ha a bemenetet adó G irányított gráf nem tartalmaz irányított kört.

12. fejezet

Mélységi keresés, aciklikus irányított gráfok

A 2. fejezetben megismerkedtünk a szélességi keresés (BFS) algoritmussal, ami bejárja egy adott G gráf csúcsait egy adott s csúcsból indulva és közben több feladatot is nagyon hatékonyan megold: meghatározza a csúcsok távolságát s-től (amennyiben minden él hosszát egynek tekintjük), eldönti, hogy G összefüggő-e és meghatározza az s-et tartalmazó komponens egy feszítőfáját. Ezt a feszítőfát – vagyis a BFS-fát – szemléletesen a lehető „legszélesebbnek” érezhetjük: s-nél a lehető legtöbb irányba ágazik, majd minden ág ismét a lehető legtöbb felé ágazik tovább, stb.

Létezik azonban a gráfok bejárására egy másik, szintén nagyon sok alkalmazással bíró megközelítés is: ez a *mélységi keresés*, vagy az angol *Depth First Search* rövidítéseként elterjedt nevén *DFS algoritmus*. Ez bizonyos értelemben épp a BFS-sel ellentétes stratégiát alkalmaz: s-ből indulva addig halad „előre”, amíg el nem akad; ekkor visszalép egyet és ismét elakadásig megy, stb. A DFS által adott feszítőfa tehát nem széles, hanem inkább „mély” lesz. Ösztönösen is ezt a megközelítést választja mindenki egy ismeretlen terep, például egy épületet felderítésekor: senkinek nem jutna eszébe a BFS logikája szerint először az s „bejárat” melletti szobákat végiglátogatni (és közben folyton visszarohanni s-hez), majd csak ezután merészkedni a bejárattól két szobányi távolságra. Ehelyett természetesnek tűnik mindig új és új szobákba továbblépve addig bolyongani az épületben, amíg további, még meg nem látogatott szobába már nem nyílik ajtó és csak ekkor visszafordulni az eggyel korábbi szobába.

A BFS algoritmust irányítatlan és irányított gráfokra is leírtuk és hasonló a helyzet a DFS-sel is: mindkét fajta gráfra alkalmazható. Ennek ellenére, alább irányított gráfokra adjuk meg az algoritmus részletes leírását (mert a gyakorlati alkalmazásokban ez fordul elő többször) és csak röviden ejtünk szót az irányítatlan esetről.

A DFS algoritmus alkalmazásaiban fontos lesz, hogy az eljárás a gráf minden csúcsát bejárja – azokat is, amik nem elérhetők s-ből (irányított) úton. Ezért a DFS is alkalmazza a 2.3. szakaszban a BFS-sel kapcsolatban már leírt stratégiát: ha már

bejárta az s-ből elérhető csúcsokat és mégsem érte el a gráf összes csúcsát, akkor egy tetszőleges, eléretlen pontból újraindítja a bejárást és ezt egészen addig ismétli, amíg minden csúcs bejárttá nem válik.

Az algoritmus a működése során kétféleképpen is megszámozza a csúcsokat: a *mélységi számozás* azt mutatja meg, hogy az eljárás hányadjára ért el egy csúcsot; a *befejezési számozás* viszont azt írja le, hogy hányadjára fejezte be egy csúcsnak és „leszármazottainak" a végigvizsgálását. Az előbbit mszám(v), az utóbbit bszám(v) fogja jelölni.

Így az algoritmus a működése során az alábbi adatokat tartja nyilván:

- mszám(v) $(v \in V)$: a v csúcs mélységi száma
- bszám(v) $(v \in V)$: a v csúcs befejezési száma
- előző(v) $(v \in V)$: a v-t megelőző csúcs – vagyis az, amiből a bejárás v-t elérte
- *aktív_csúcs*: a jelenleg aktív csúcs
- *MSZÁM*: az eddigi legnagyobb mélységi szám
- *BSZÁM*: az eddigi legnagyobb befejezési szám

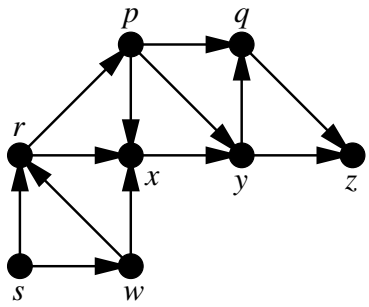
DFS ALGORITMUS

Bemenet: Egy $G = (V,E)$ irányított gráf és egy $s \in V$ csúcs

```
1   mszám(s) ← 1; minden v ∈ V, v ≠ s-re mszám(v) ← *
2   minden v ∈ V-re bszám(v) ← *
3   minden v ∈ V-re előző(v) ← *
4   MSZÁM ← 1; BSZÁM ← 0; aktív_csúcs ← s
5   ciklus
6     ha létezik olyan e = (aktív_csúcs, v) él, amire mszám(v) = *, akkor:
7       MSZÁM ← MSZÁM + 1; mszám(v) ← MSZÁM
8       előző(v) ← aktív_csúcs
9       aktív_csúcs ← v
10    különben:
11      BSZÁM ← BSZÁM + 1; bszám(aktív_csúcs) ← BSZÁM
12      ha előző(aktív_csúcs) ≠ *, akkor:
13        aktív_csúcs ← előző(aktív_csúcs)
14      különben:
15        ha van olyan v csúcs, amire mszám(v) = *, akkor:
16          aktív_csúcs legyen egy ilyen v csúcs
17          MSZÁM ← MSZÁM + 1; mszám(aktív_csúcs) ← MSZÁM
18        különben:
19          stop
20  ciklus vége
```

A 6-9. sorokban tehát az eljárás az aktuális *aktív_csúcs* csúcsból továbblép a még bejáratlan v-be, v-nek adja a soron következő mélységi számot, feljegyzi, hogy v-t *aktív_csúcs*-ból érte el és az aktív csúcsot v-re változtatja. Ha viszont ilyen v csúcs már nincs, akkor (a 11-13. sorok szerint) *aktív_csúcs* megkapja a soron következő befejezési számot és előző(*aktív_csúcs*) lesz az új aktív csúcs, amiből az

eljárás *aktív_csúcs*-ot elérte; az előző(*aktív_csúcs*) $= *$ esetben pedig (a 15-17. sorok szerint) új „gyökérpontot” választ, ami megkapja a soron következő mélységi számot.



12.1. ábra

Az algoritmus működését a 12.1. ábra gráfján illusztráljuk. Erre lefuttatva az eljárást az alábbi adatok keletkeznek:

v	s	p	q	r	x	y	z	w
mszám(v)	1	8	6	7	3	4	5	2
bszám(v)	8	5	2	6	4	3	1	7
előző(v)	$*$	r	y	w	w	x	y	s

Természetesen ez az algoritmusnak csak az egyik lehetséges helyes futása, mert nincs arra vonatkozó megkötés, hogy *aktív_csúcs* melyik, még bejáratlan szomszédjába lépjen tovább.

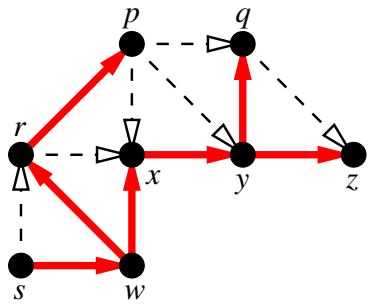
A fenti példában egyszer sem volt szükség új gyökérpont választására (vagyis a 16-17. sorok végrehajtására): a 12. sor végrehajtásakor előző(*aktív_csúcs*) $= *$ először *aktív_csúcs* $= s$-re fordult elő, amikor már minden v csúcsra mszám(v) $\neq *$ volt, így az eljárás megállt. Említettük azonban, hogy ez nem mindig van így, csak ha s-ből G minden csúcsa elérhető irányított úton (lásd a 12.1. Lemmát, illetve az utána írt megjegyzést). Az általános esetben az algoritmus leállásakor azokra a v csúcsokra lesz előző(v) $= *$, amik az eljárás során valamikor gyökérpontok voltak.

12.1. A DFS algoritmus működésének vizsgálata

A DFS működése elég szemléletes ahhoz, hogy magától értetődőnek tekintsünk olyan állításokat, amik valójában átgondolásra szorulnak. Ilyen állítást tettünk az előző bekezdésben is (s-en kívüli gyökérpont keletkezéséről) – de még az sem magától értetődő, hogy az eljárás végére minden csúcs kap befejezési számot. (Ugyanez a kérdés a mélységi számokkal kapcsolatban persze nem vetődik fel, hiszen a 15. sorban szereplő esetvizsgálat miatt az eljárás csak akkor állhat meg, ha már minden v csúcsra mszám(v) $\neq *$.) Az alábbiakban a DFS működésére vonatkozó egyéb ismeretek mellett ezeket a hiányosságokat is pótoljuk.

12.1.1. A DFS-erdő

Hasonlóan a BFS algoritmushoz, a DFS esetében is külön figyelmet érdemelnek az előző(v)-ből v-be mutató élek azokra a v csúcsokra, amikre előző(v) $\neq *$ (tehát amik nem voltak gyökérpontok az eljárás futása során). A fenti, a 12.1. ábra gráfjára mutatott példafuttatás esetén ezek a 12.2. ábrán pirossal kiemelve láthatók. Itt a piros élek fát alkotnak (ha eltekintünk az élek irányításától), de ugyanez általában nem igaz (csak akkor, ha s-en kívül nem volt több gyökérpont). Az viszont könnyen láthatóan igaz, hogy az előző(v)-ből v-be mutató élek (irányítatlan értelemben) olyan erdőt (vagyis körmentes részgráfot) alkotnak, ami G minden csúcsát tartalmazza. Ennek az erdőnek a neve *DFS-erdő*.



12.2. ábra

12.1.2. A DFS algoritmus lépésszáma

A DFS algoritmus hatékony implementálásához is érdemes feltenni, hogy a bemenetet adó G gráf szomszédsági listával van megadva. Ezzel ugyanis elérhető, hogy a pszeudokód 6. sorában zajló esetvizsgálatok során G minden élével a teljes futásidő alatt csak egyszer kelljen foglalkozni. Ehhez azonban szükség van némi körültekintésre: mivel G egy v csúcsára $v = aktív\_csúcs$ az eljárás során többször is előfordulhat, hiba lenne az algoritmust úgy implementálni, hogy az minden ilyen alkalommal elölről kezdje a v-ből kiinduló élek végigvizsgálását. Ennek érdekében minden v csúcsra érdemes fenntartani egy mutatót, amiben eltároljuk, hogy a v-hez tartozó listában (vagyis a v-ből kiinduló élek sorozatában) éppen hol tartunk. Így amikor $v = aktív\_csúcs$ legközelebb bekövetkezik, elég ettől a ponttól folytatni a 6. sorban zajló esetvizsgálatot.

A teljesség kedvéért megemlítjük, hogy egy hasonló probléma a 15. sorban, az új gyökérpont keresése kapcsán is felmerülhet: ha a futás során nagyon sokszor kell ilyet tenni (vagyis a 15. sort végrehajtani) és ehhez mindig elölről kezdjük a gráf csúcsainak átfésülését, akkor ez is fölöslegesen nagy, n^2-tel arányos lépésszámhoz vezethetne (ahol továbbra is n, illetve m jelöli a gráf csúcsainak, illetve éleinek a számát). Ezt is a fentihez hasonló módon kerülhetjük el: egyetlen további mutatót fenntartunk annak az eltárolására, hogy a 15. sor legutóbbi végrehajtásakor hol

hagytuk abba a csúcsok vizsgálatát és legközelebb innen folytatjuk.

A fentiek szerint tehát elmondható, hogy a DFS algoritmus a teljes futása alatt minden v csúcs listáján egyszer lépked végig (bár több menetben), ezért a lépésszáma alkalmas c konstansra $c \cdot (n+m)$. Így a DFS algoritmus is polinomiális lépésszámú, sőt, még azok között is a leghatékonyabbak közé tartozik.

12.1.3. Az élek osztályozása

A DFS algoritmus elemzésekor érdemes képzeletben kiegészíteni annak a működését egy T változóval, ami az időt méri: T értéke az eljárás indulásakor 1, majd minden alkalommal, amikor egy v csúcs $\text{mszám}(v)$ mélységi száma vagy $\text{bszám}(v)$ befejezési száma értéket kap (vagyis a pszeudokód 7., 11. vagy 17. sora végrehajtásra kerül), T értéke is nő eggyel. (Ezt felfoghatjuk úgy is, hogy $T = MSZÁM + BSZÁM$, ahol *MSZÁM*, illetve *BSZÁM* a pszeudokód szerint az eddigi legnagyobb mélységi, illetve befejezési szám.) Így az algoritmus futása során minden v csúcshoz tartozik egy $\text{kezd}(v)$ kezdési idő: ez T-nek az az értéke, amikor $\text{mszám}(v)$ értéket kap (és $\text{kezd}(s) = 1$). Az alábbi állításból pedig következni fog, hogy v-hez egy $\text{bef}(v)$ befejezési idő is tartozik, amikor $\text{bszám}(v)$ kap értéket. Továbbá beszélhetünk a v-hez tartozó $I(v)$ időintervallumról is, ami $\text{kezd}(v)$-től $\text{bef}(v)$-ig tart; más szóval, $T \in I(v)$ azt jelenti, hogy $\text{kezd}(v) \leq T \leq \text{bef}(v)$. Végül még értelmezzük pontosan a következő, fentebb szemléletesen (és idézőjelek között) már használt fogalmat: egy y csúcsot az x *leszármazottjának* nevezünk, ha az (adott futáshoz tartozó) DFS-erdőben létezik x-ből y-ba irányított út; más szóval, ha y-ból az előző(v) mutatók mentén visszafelé lépegetve véges sok lépés után x-be jutunk (vagy $y = x$).

12.1. Lemma. *Tegyük fel, hogy a G irányított gráfra a DFS algoritmust futtatva $\text{kezd}(x) = T_0$ (vagyis T_0 az első időpont, amikor $\text{mszám}(x) \neq *$). Ekkor*

(i) egy későbbi időpontban $\text{bszám}(x)$ is kap értéket (vagyis $\text{bef}(x)$ létezik);

(ii) ha egy v csúcsra $\text{kezd}(v) \in I(x)$, akkor $\text{bef}(v) \in I(x)$ is igaz és v leszármazottja x-nek;

(iii) ha P egy olyan, x-ből induló irányított út G-ben, aminek minden v csúcsára $\text{kezd}(x) \leq \text{kezd}(v)$, akkor P minden v csúcsára $\text{bef}(v) \leq \text{bef}(x)$ (vagyis $I(x)$ tartalmazza $I(v)$-t).

A lemma állításainak érzékeltetésére ismét hívjuk segítségül a fejezet elején már használt, az algoritmus működését illusztráló hasonlatot: egy ismeretlen épületet derítünk fel úgy, hogy abból a szobából, amiben éppen állunk (vagyis *aktív_csúcs*ból) mindig egy újabb, még meg nem látogatott szobába lépünk; ha pedig ez már nem lehetséges, akkor visszavonulunk abba a szobába, ahonnan érkeztünk. Bár ez az illusztráció életszerűségét rontja, az élek irányításának figyelembe vétele érdekében feltételezzük, hogy az ajtók csak az egyik irányba nyílnak és azok az ajtók, amiken új szobába léptünk, a visszajutásunk érdekében nyitva maradnak mögöttünk.

Ekkor bármelyik x szobába léptünk is be éppen, az innen induló felfedezőutunk – talán számtalan további elágazás és visszalépés után – egyszer lezárul, kimerülnek az x-ből elérhető újabb szobák és végül vissza fogunk vonulni azon az ajtón át,

amin x-be érkeztünk; pontosan ezt állítja a lemma *(i)* állítása. A *(ii)* állítás pedig azt mondja, hogy ha az x-be való első megérkezésünk és az x-ből való visszavonulásunk között egy v szobába jutunk, akkor a v-ből induló teljes, az onnan való visszavonulásunkig tartó barangolásunk után egyrészt még visszajutunk x-be, másrészt az x-ből való visszavonulás csak ezután következhet. Végül a *(iii)* állítás azt mondja, hogy ha az x-be érkezésünk pillanatában egy, az x-ből induló séta szobái még mind bejáratlanok, akkor az x-ből induló felfedezőutunk során ezekbe a szobákba mind el is jutunk (még ha nem is feltétlen a szóban forgó séta nyomvonalán).

Bár ezek az állítások kézenfekvőnek tűnhetnek, a szemlélet helyett az algoritmus pszeudokódjára építő, pontos bizonyításuk már kevésbé magától értetődő és kicsit körülményes.

Bizonyítás: Az *(i)* állítás bizonyításához megmutatjuk, hogy ha egy $T \geq T_0$ pillanatban még $\text{bszám}(x) = *$, akkor az épp aktuális *aktív_csúcs* csúcs leszármazottja x-nek. Ez a T_0 időpontban nyilván teljesül (mert *aktív_csúcs* $= x$), így elég lesz megmutatni, hogy ha ez a tulajdonság egy T időpontban igaz, akkor igaz marad $(T+1)$-ben is. Mivel a T időpontban $\text{előző}(\textit{aktív\_csúcs}) \neq *$ (hiszen *aktív_csúcs* leszármazottja x-nek), ezért T értéke csak a 7. vagy a 11. sor végrehajtása miatt nőhet $(T+1)$-re. Az első esetben egy új v csúcs kap $\text{mszám}(v)$ értéket, amire $\text{előző}(v) = \textit{aktív\_csúcs}$ és *aktív_csúcs* értéke v lesz; így ha *aktív_csúcs* leszármazottja volt x-nek, akkor ugyanez nyilván v-re is igaz lesz. Ha viszont T értéke a 11. sor végrehajtása miatt növekszik $(T+1)$-re, akkor a T időpontban már nincs olyan $(\textit{aktív\_csúcs}, v)$ él, amire $\text{mszám}(v) = *$. Ha most *aktív_csúcs* $\neq x$, akkor (miután $\text{bszám}(\textit{aktív\_csúcs})$ értéket kap) *aktív_csúcs* értéke $\text{előző}(\textit{aktív\_csúcs})$-ra változik; de mivel *aktív_csúcs* leszármazottja volt x-nek, nyilván $\text{előző}(\textit{aktív\_csúcs})$ is az lesz. Ha viszont *aktív_csúcs* $= x$, akkor ebben a pillanatban $\text{bszám}(x)$ értéket kap (és $\text{bef}(x) = T+1$).

Mivel az eljárás csak akkor állhat meg, ha $\text{előző}(\textit{aktív\_csúcs}) = *$ (hiszen a 19. sor végrehajtásához ez szükséges, de nem elégséges feltétel), ezért a fenti bekezdésben belátott állítás szerint ez nem történhet meg addig, amíg $\text{bszám}(x) = *$. Valóban, $\text{bszám}(x) = *$ (és $T \geq \text{kezd}(x)$) esetén *aktív_csúcs* leszármazottja x-nek, így $\text{előző}(\textit{aktív\_csúcs}) \neq *$. Mivel a DFS eljárás véges sok lépés után biztosan megáll (hiszen a csúcsok száma véges és a ciklusmag minden végrehajtásakor csökken a kitöltetlen $\text{mszám}(v)$ vagy $\text{bszám}(v)$ értékek száma), ezért az algoritmus megállásakor $\text{bszám}(x) \neq *$ valóban teljesül.

A *(ii)* állítás a $v = x$ esetben nyilván igaz, ezért feltehetjük, hogy $v \neq x$. Fentebb beláttuk, hogy $I(x)$ teljes időtartama alatt *aktív_csúcs* leszármazottja x-nek. Mivel a $\text{kezd}(v)$ időpontban *aktív_csúcs* $= v$, ezért v valóban leszármazottja x-nek. Másrészt ugyanezt az állítást v-re is alkalmazhatjuk: $I(v)$ alatt az aktív csúcs v-nek leszármazottja. Mivel x nem leszármazottja v-nek (hiszen $v \neq x$ miatt nem lehetnek egymás leszármazottai), ezért $I(v)$ alatt $x = \textit{aktív\_csúcs}$ lehetetlen. Mivel azonban a $\text{bef}(x)$ pillanatban $x = \textit{aktív\_csúcs}$ kell legyen, ebből $\text{bef}(x) > \text{bef}(v)$ és így $\text{bef}(v) \in I(x)$ valóban következik.

Következésképp *(iii)* igazolásához azt kell csak megmutatnunk, hogy ha egy x-ből induló P irányított út minden v csúcsára $\text{kezd}(x) \leq \text{kezd}(v)$, akkor ezekre a

csúcsokra kezd$(v) \in I(x)$ (más szóval, hogy kezd$(v) <$ bef(x)). Ez x-re nyilván igaz, így elég lesz belátni, hogy ha ez P valamely y csúcsára teljesül, akkor a P-n utána következő z csúcsra is. Mivel kezd$(y) \in I(x)$, ezért *(ii)* miatt bef$(y) \in I(x)$ is igaz. A bef(y) időpontban minden $e = (y,v)$ él v végpontjára mszám$(v) \neq *$, így ez az (y,z) élre és a z csúcsra is igaz. Ezért kezd$(z) <$ bef(y), amiből bef$(y) \in I(x)$ miatt kezd$(z) <$ bef(x) valóban adódik. □

A 12.1. Lemmából valóban következik az a fentebb már említett állítás, hogy ha G minden csúcsa elérhető s-ből irányított úton, akkor a DFS algoritmus futása során s lesz az egyetlen gyökérpont (vagyis az eljárás leállásakor minden $v \neq s$ csúcsra előző$(v) \neq *$). Valóban, mivel kezd$(v) \geq 1$ nyilván minden v csúcsra igaz, ezért a lemma *(iii)* állítása szerint (azt az $x = s$ csúcsra alkalmazva) s kapja a legnagyobb befejezési számot. Így a bef(s) időpontban már nem létezhet olyan v csúcs, amire mszám$(v) = *$, ezért az eljárás megáll.

A DFS-erdő, illetve a leszármazotti viszony fentebb bevezetett fogalma lehetőséget teremt G éleinek az alábbi osztályozására, ami a DFS algoritmus számos alkalmazásában szerephez jut.

12.2. Definíció. *Tegyük fel, hogy az s csúcsból indítva lefuttattuk a DFS algoritmust a G irányított gráfban. Jelölje a futáshoz tartozó DFS-erdőt F. Legyen $e = (u,v)$ a G-nek egy tetszőleges éle. Ekkor*

(i) *e-t* faélnek *nevezzük, ha $e \in E(F)$;*

(ii) *e-t* előreélnek *nevezzük, ha nem faél, de v leszármazottja u-nak (vagyis van F-ben u-ból v-be irányított út);*

(iii) *e-t* visszaélnek *nevezzük, ha u leszármazottja v-nek (vagyis van F-ben v-ből u-ba irányított út);*

(iv) *e-t* keresztélnek *nevezzük, ha u és v közül egyik sem leszármazottja a másiknak (vagyis F-ben egyikből sincs a másikba irányított út).*

Fontos kiemelni, hogy az itt bevezetett fogalmaknak csak a DFS egy konkrét futását feltételezve van értelme: ugyanaz az e él könnyen tartozhat a fenti négy osztály közül egy másikba is, ha a DFS-nek egy másik (helyes) futását feltételezzük. Például a 12.1. ábra gráfjára fentebb látott futtatás esetén az (s,r) él előreél és az összes többi, a DFS-erdőbe nem tartozó él keresztél. Visszaélt tehát ebben a példában nem találunk; később látni fogjuk, hogy ennek a ténynek fontos következményei vannak.

Fentebb már említettük, hogy a DFS eljárás (helyes implementáció esetén) G minden e élével egyszer foglalkozik. Ezért a gyakorlati alkalmazások számára nagyon hasznos, hogy ebben a pillanatban az is megállapítható (és akár eltárolható), hogy e a fenti négy kategória közül melyikbe tartozik – annak ellenére is, hogy ekkor még nem ismert a teljes DFS-erdő. Így tehát a DFS eljárás minimális kiegészítésével az is elérhető, hogy az (a futásidő érdemi növekedése nélkül) G minden élét besorolja a fenti definíció szerinti osztályok valamelyikébe. Ennek a részleteit mondja ki az alábbi tétel.

12.3. Tétel. *Tegyük fel, hogy a G irányított gráfra a DFS algoritmust futtatva a pszeudokód 6. sorának egyik végrehajtásakor aktív_csúcs $= a$ és az eljárás éppen az $e = (a, v)$ élt vizsgálja. Ekkor e erre a DFS bejárásra vonatkozóan akkor és csak akkor lesz*

(i) faél, ha mszám$(v) = *$;
(ii) előreél, ha mszám$(v) >$ mszám(a);
(iii) visszaél, ha mszám$(v) <$ mszám(a) *és* bszám$(v) = *$;
(iv) keresztél, ha mszám$(v) <$ mszám(a) *és* bszám$(v) \neq *$.

Bizonyítás: Mind a négy esetben elég lesz a „balról jobbra” irányt belátni – vagyis hogy a mélységi és befejezési számokra írt feltételek következnek e megfelelő osztályba való tartozásából. Valóban, mivel ezek a feltételek kölcsönösen kizárják egymást és minden él a 12.2. Definíció szerinti osztályok közül pontosan egybe tartozik, ezért ebből a „jobbról balra” irányok is következnek.

Ha e faél, akkor előző$(v) = a$. Mivel előző(v) csak az eljárás 8. sorában kaphat értéket, így ez a 6. sorban zajló esetvizsgálat miatt valóban csak a mszám$(v) = *$ esetben történhet meg.

Ha e előreél, akkor v leszármazottja a-nak, amiből mszám$(v) >$ mszám(a) azonnal adódik (hiszen a leszármazottai nyilván később kapnak mélységi számot a-nál, amik ezért nem lehetnek kisebbek mszám(a)-nál).

Ha e visszaél, akkor a helyzet fordított: a leszármazottja v-nek és így mszám$(a) >$ mszám(v). Továbbá mivel a v-ből a-ba vezető F-beli irányított út minden u csúcsa is leszármazottja v-nek, ezért ezekre kezd$(u) \geq$ kezd(v). Így a 12.1. Lemma *(iii)* állítása szerint ezekre a csúcsokra bef$(u) \leq$ bef(v). Ebből tehát bef$(a) <$ bef(v) is adódik, ami azt jelenti, hogy a bef(a) időpontban még bszám$(v) = *$. Mivel az $e = (a, v)$ él vizsgálata legkésőbb a bef(a) időpontban történik, ebből valóban következik, hogy bszám$(v) = *$ ekkor is igaz.

Végül ha e keresztél, akkor kezd$(v) \notin I(a)$, különben a 12.1. Lemma *(ii)* állítása szerint v leszármazottja volna a-nak. Továbbá kezd$(v) >$ bef(a) is lehetetlen, mert akkor e vizsgálatának pillanatában (ami nyilván $I(a)$-n belül van) mszám$(v) = *$ volna és így e faél volna. Ezekből tehát kezd$(v) <$ kezd(a). Ebből viszont bef$(v) <$ kezd(a) is következik, különben kezd$(a) \in I(v)$ miatt (ismét a 12.1. Lemma *(ii)* állítása szerint) a leszármazottja volna v-nek. Ez tehát azt jelenti, hogy a kezd(a) időpontban már bszám$(v) \neq *$, így ugyanez e vizsgálatának pillanatában is igaz. □

12.1.4. DFS algoritmus irányítatlan gráfon

Ahhoz, hogy a DFS eljárást egy G irányítatlan gráfra futtassuk, csak egy apró változtatásra van szükség a pszeudokód 6. sorában:

6 **ha** létezik olyan $e = \{$*aktív_csúcs*$, v\}$ él, amire mszám$(v) = *$, **akkor:**

Vagyis az *aktív_csúcs*-ból induló irányított élek helyett minden, *aktív_csúcs*-ra illeszkedő irányítatlan élen megpróbálunk továbblépni *aktív_csúcs*-ból.

A G irányítatlan gráfon futtatott DFS elképzelhető úgy is, mintha azon a H irányított gráfon hajtanánk végre, amit G-ből kapunk úgy, hogy annak minden $e = \{u,v\}$ élét helyettesítjük az $e_1 = (u,v)$ és $e_2 = (v,u)$ irányított élekkel (noha a gyakorlatban persze fölösleges volna az élhalmaz megduplázása). Ebből kiindulva könnyű végiggondolni, hogy hogyan alakul át az élek 12.2. Definíció szerinti osztályozása az irányítatlan esetben. Faélek nyilván továbbra is vannak, azonban a DFS-erdőbe nem tartozó élek sokkal egyszerűbben leírhatók. Egyrészt az előreélek egybeesnek a visszaélekkel (ezeket az irányítatlan esetben visszaélnek szokták hívni), hiszen ha H-ban $e_1 = (u,v)$ előreél, akkor $e_2 = (v,u)$ visszaél (és fordítva) mert u-ból v-be pontosan akkor létezik irányított út, ha v-ből u-ba létezik. Másrészt könnyű látni, hogy G-ben nem lehetnek keresztélek: ha H-ban $e_1 = (u,v)$ keresztél volna, akkor $e_2 = (v,u)$-nak is annak kellene lennie (mert mindkét állítás azt fejezi ki, hogy u és v között nincs irányított út a H-beli DFS-erdőben), így a 12.3. Tétel szerint mszám$(u) <$ mszám(v) és mszám$(v) <$ mszám(u) is teljesülne, ami nyilván lehetetlen. Következésképp az irányítatlan gráfban futtatott DFS algoritmus esetében minden él végpontjai közül az egyik leszármazottja a másiknak.

Ebből az is következik, hogy ha G irányítatlan és összefüggő, akkor a DFS-erdője feszítőfa: valóban, mivel ekkor H-ban minden csúcs elérhető s-ből irányított úton, fentebb már beláttuk, hogy H DFS-erdője (irányítatlan értelemben) feszítőfa, így ugyanez G-re is teljesül. Ezért a BFS-hez hasonlóan a DFS is használható az összefüggőség eldöntésére, illetve egy feszítőfa meghatározására.

12.2. Aciklikus irányított gráfok, topologikus rendezés

Egy G irányított gráfot akkor nevezünk *aciklikusnak*, ha nem tartalmaz irányított kört. (Elterjedt elnevezés az ilyen gráfokra a *DAG* is, ami az angol *Directed Acyclic Graph* kifejezés rövidítése.) Rengeteg gyakorlati alkalmazásban merülnek fel olyan irányított gráfok, amik a feladat természetéből fakadóan aciklikusak. Ennek a ténynek a jelentősége abban rejlik, hogy sok algoritmikus feladat jóval hatékonyabban oldható meg aciklikus irányított gráfokra, mint az általános esetben.

Tekintsük például azt a G irányított gráfot, aminek a csúcsai a Földön élő emberek és x-ből akkor vezet él y-ba, ha y gyermeke x-nek. G nyilván aciklikus, de ezt a tényt különösen látványossá tehetjük a következő módon: képzeletben állítsuk fel egyetlen sorba a Föld összes emberét, mégpedig balról jobbra születési idő szerinti növekvő sorrendben (feltételezve, hogy semelyik két ember nem született hajszálra azonos pillanatban). Ekkor G minden éle balról jobbra mutat (hiszen mindenki idősebb a saját gyerekeinél); így G-ben az élek mentén haladva egyre inkább jobbra kerülünk, vagyis valóban lehetetlen irányított kört bezárni. Ezt az egyszerű gondolatot általánosítja az alábbi definíció.

12.4. Definíció. *Legyen $G = (V,E)$ irányított gráf és $(v_1, v_2, \ldots, v_n)$ a G csúcsainak egy felsorolása. A $(v_1, v_2, \ldots, v_n)$ sorozatot* topologikus rendezésnek *(vagy* topologikus sorrendnek*) nevezzük, ha G minden (x,y) élére x előbb van a sorozatban, mint y (vagyis ha $x = v_i$ és $y = v_j$, akkor $i < j$).*

Például a 12.1. ábra irányított gráfjának az (s,w,r,p,x,y,q,z) sorozat topologikus rendezése. De természetesen nem minden irányított gráfnak van topologikus rendezése: ha a gráfban van irányított kör, akkor a csúcsok egyetlen sorbaállítása sem lehet jó, mert a kör mentén haladva olyan élt is kell találnunk, ami alacsonyabb sorszámú csúcsba visz minket vissza. Így topologikus rendezése csak aciklikus irányított gráfoknak lehet. Ezért érdekes az alábbi, egyszerű tétel.

12.5. Tétel. *A G irányított gráfnak akkor és csak akkor van topologikus rendezése, ha aciklikus.*

Bizonyítás: A feltétel szükségessége magától értetődő (és az imént láttuk be). Az elégségesség bizonyításához először azt mutatjuk meg, hogy minden aciklikus irányított gráf tartalmaz *nyelőt* – vagyis olyan csúcsot, amiből nem indul ki él. Ehhez vegyünk a gráfban egy P leghosszabb irányított utat és legyen v ennek a végpontja. Ekkor v szükségképpen nyelő: v-ből nem vezethet irányított él sem P egy korábbi pontjába (mert akkor irányított kör keletkezne), sem egy P-n kívüli csúcsba (mert akkor P-nél hosszabb irányított út keletkezne).

Válasszunk tehát G-ben egy nyelőt, legyen ez v_n. Most töröljük G-ből v_n-et (és a rá illeszkedő éleket). Nyilván a kapott G' gráf is aciklikus, így ebben is választhatunk egy nyelőt, ez legyen v_{n-1}, stb. Az eljárást hasonlóan folytatva kapjuk (jobbról balra) a $(v_1, v_2, \ldots, v_n)$ sorozatot, ami nyilván topologikus rendezése G-nek. □

A fenti bizonyítás azon alapult, hogy minden aciklikus irányított gráf tartalmaz nyelőt – de természetesen hasonlóan megmutatható, hogy olyan csúcsot, is tartalmaznak, amibe nem lép be él; ezeket *forrásnak* szokták nevezni. Topologikus rendezés létezése pedig nyelők ismételt törlése helyett források ismételt törlésével is megmutatható: ebben az esetben a sorrend balról jobbra alakul ki.

A topologikus rendezés léte a fő oka annak a fentebb már említett jelenségnek, hogy számos algoritmikus feladat sokkal hatékonyabban oldható meg aciklikus irányított gráfokra. Ilyen például a 11. fejezetben tárgyalt legrövidebb út probléma is: bár az erre megismert algoritmusok – Bellman és Ford, illetve Dijkstra algoritmusa – hatékony, polinomiális futásidejű eljárások, de a lépésszámuk azért rosszabb a (többek között a) BFS és DFS algoritmusnál is elérhető $c \cdot (n+m)$-nél (ahol n, illetve m továbbra is a gráf csúcs-, illetve élszámát jelöli és c alkalmas konstans), így nagyon nagy méretű gráfokra még ezeknek a lépésszáma is túl nagynak bizonyulhat. Ezért nagyon hasznos, hogy aciklikus irányított gráfok esetében jóval egyszerűbben és hatékonyabban, $c \cdot (n+m)$ futásidőben is megoldható a feladat.

Az alábbi algoritmus a legrövidebb út feladatot oldja meg (tetszőleges, valós élsúlyokra) abban az esetben, ha a bemenet részeként adott egy topologikus rendezés

is (és így a gráf aciklikus). Azt is feltételezni fogjuk, hogy a topologikus sorrend első csúcsa s, amitől a többi csúcs távolságának a meghatározása a feladat. Ezzel azonban az általános esetet is kezeljük: már a 11. fejezetben is említettük azt a nyilvánvaló tényt, hogy mivel s-be belépő élek nem lehetnek rajta s-ből induló irányított úton, ezért ezeknek a legrövidebb út problémában semmilyen szerepe nincs, a feladat megváltoztatása nélkül törölhetők a gráfból. Ha pedig s-be nem lép be irányított él, akkor tetszőleges topologikus sorrendben s előrehozható az első helyre anélkül, hogy a 12.4. Definíció feltételét elrontanánk. Az algoritmus leírásában ismét feltételezni fogjuk a ∞ szimbólummal való számolással kapcsolatban a 11.4. Állítás előtt írt megállapodásokat.

LEGRÖVIDEBB ÚT ACIKLIKUS IRÁNYÍTOTT GRÁFBAN

Bemenet: Egy $G = (V,E)$ aciklikus irányított gráf, egy $w : E(G) \to \mathbb{R}$ súlyfüggvény és G csúcsainak egy $(s = v_1, v_2, \ldots, v_n)$ topologikus rendezése

```
1   táv(v_1) ← 0, minden i > 1-re táv(v_i) ← ∞
2   ciklus: i fut 2-től n-ig
3       táv(v_i) ← min{táv(v_j) + w(e) : e = (v_j, v_i)}
4   ciklus vége
```

A fenti, pofonegyszerű eljárás helyesen határozza meg az $s = v_1$ csúcsból az összes többi v csúcs táv(v) távolságát. Valóban: ha $j < i$, akkor egy s-ből v_j-be vezető legrövidebb (vagy bármilyen) úton v_i nyilván nem lehet rajta. Ezért egy s-ből v_i-be vezető legrövidebb út egy s-ből v_j-be vezető legrövidebb útnak a (v_j, v_i) éllel való megtoldásából áll valamely $j < i$ értékre (legalábbis akkor, ha s-ből létezik v_i-be irányított út). Ha tehát feltételezzük, hogy az eljárás már helyesen meghatározta s-ből az $s = v_1, v_2, \ldots, v_{i-1}$ csúcsokba vezető legrövidebb utak hosszát (és kezdetben az $i = 1$ értékre, vagyis s-re nyilván ez a helyzet), akkor a ciklus magjában számolt minimum ezt v_i-re is helyesen teszi meg. Ha viszont minden (v_j, v_i) él esetén táv$(v_j) = \infty$ (vagy ha ilyen él egyáltalán nincs), akkor nyilván v_i-be sem vezet s-ből irányított út, így táv$(v_i) = \infty$ szintén helyes.

Érdemes megjegyezni, hogy bár a fenti eljárás ebben a formájában csak a legrövidebb utak hosszát számolja ki, nagyon könnyen lehetne azt úgy módosítani, hogy a futásából egy legrövidebb út is kiolvasható legyen s-ből minden más csúcsba. Ehhez a 11. fejezetben már látott módon minden $v \neq s$ csúcshoz meg kellene határozni egy előző(v) csúcsot is, ami megelőzi v-t egy s-ből v-be vezető legrövidebb úton. Az előző(v) csúcsokat pedig a 3. sorban, a táv(v_i)-kkel párhuzamosan kaphatjuk meg úgy, hogy előző(v_i) értéke egy olyan $e = (v_j, v_i)$ él kezdőpontja, amire a 3. sorban írt minimum felvétetik (feltéve, hogy táv$(v_i) \neq \infty$).

A fenti algoritmust tekinthetjük úgy is, mintha a Bellman-Ford algoritmus belső (a 11.1. szakaszban látott pszeudokód szerint az 5-13. sorokban zajló) ciklusát csak egyszer kellene végrehajtani. Ezért ennek az algoritmusnak az implementálásakor is érdemes a gráfot a szomszédsági listának azzal a változatával tárolni, amiben a v csúcshoz tartozó lista a v-be belépő éleket sorolja fel. Így az eljárás futtatásakor minden csúcs listáján csak egyszer kell végiglépkeni, így a teljes lépésszám (n csúcsú

és m élű gráfra) $c \cdot (n+m)$ lesz alkalmas c konstansra, vagyis az algoritmus megint nem csupán polinomiális futásidejű, hanem még azon belül is rendkívül hatékony.

Fontos megemlíteni azt is, hogy egy tetszőleges irányított gráfban egy *leghosszabb út* meghatározása s-ből a többi csúcsba nagyon nehéz probléma, nem ismert polinomiális algoritmus (lásd erről a 11. fejezet elején az apróbetűs részt). Ha azonban a gráf aciklikus, akkor a feladat hatékonyan megoldhatóvá válik: ehhez a fenti algoritmust csak annyiban kell módosítani, hogy a 3. sorban szereplő minimumot maximumra cseréljük (és a módszer helyességének a bizonyítása is azonos). Az így módosított eljárást is számos gyakorlati alkalmazásban használják.

Röviden megemlítjük az aciklikus irányított gráfok leghosszabb út problémájának a legfontosabb alkalmazását, ami egy ütemezési feladat. Itt a G irányított gráf csúcsai egy nagyobb projekt részfeladatait, G élei pedig megelőzési viszonyokat reprezentálnak: az $e = (u,v)$ él azt fejezi ki, hogy v-t csak u után lehet elvégezni. Ráadásul nem is azonnal: minden $e = (u,v)$ élnek adott egy $w(e)$ hossza, ami azt fejezi ki, hogy u kezdési időpontja után legalább $w(e)$ időt kell várni v elkezdéséig. Ekkor G nyilván aciklikus (vagy ha nem az, akkor a projekt nem megvalósítható). Ha s jelöli az egész projekt indításának, mint részfeladatnak megfelelő csúcsot és s-ből v-be vezet egy t hosszúságú irányított út, akkor a projekt indulása után v elvégzése előtt nyilván legalább t időt kell várni. Ezért ha s-et a nulla időpontban végezzük el, akkor minden $v \neq s$ csúcs esetén az s-ből v-be vezető leghosszabb irányított út hossza adja meg azt a legkorábbi időpillanatot, amikor v elkezdhető. Így a fenti bekezdésben írt algoritmussal olyan optimális ütemezést kaphatunk, amiben minden részfeladatot (és így az egész projektet is) a lehető leghamarabb végezzük el. Az ezt az ütemezési feladatot megoldó eljárás (ami tehát lényegében azonos a fenti bekezdésben írttal, illetve annak minimális kiegészítésével kapható) *PERT módszer* néven közismert (az angol *Program Evaluation and Review Technique* rövidítéséből).

12.3. Topologikus rendezés a DFS algoritmussal

A fenti, aciklikus irányított gráfban legrövidebb (vagy leghosszabb) irányított utakat kereső algoritmus feltételezte, hogy G-nek eleve adott egy topologikus rendezése. De mi van akkor, ha nem ez a helyzet: csak G adott és a topologikus rendezés meghatározása is a feladat része? A 12.5. Tétel bizonyításában látott módszer (vagyis nyelők ismételt keresése és törlése a gráfból) alkalmas egy topologikus rendezés algoritmikus meghatározására is, de n csúcsú gráfra $c \cdot n^2$ lépésszámú eljárást eredményezne. Ez azért kellemetlen, mert ezzel az imént látott legrövidebb (vagy leghosszabb) utakat kereső algoritmus lépéssszáma is $c \cdot n^2$-re romlana, ha azt a topologikus rendezés meghatározásával kellene kezdenünk. Ezért nagyon hasznos, hogy G aciklikussága eldönthető, illetve aciklikus G-re egy topologikus rendezés meghatározható $c \cdot (n+m)$ futásidőben is, mégpedig a DFS algoritmus segítségével.

12.6. Tétel. *Futtassuk le a DFS algoritmust a G irányított gráfban az s csúcsból indítva. G akkor és csak akkor aciklikus, ha az eljárás során nem keletkezik visszaél és ebben az esetben a befejezési számozás szerinti fordított sorrendben felsorolva G csúcsait topologikus rendezést kapunk.*

Bizonyítás: Ha keletkezik visszaél és $e = (a,v)$ ilyen, akkor G nyilván tartalmaz irányított kört: az F DFS-erdő tartalmaz v-ből a-ba egy P irányított utat (mert e visszaél), így P-t e-vel kiegészítve irányított körré zárhatjuk.

Tegyük fel ezért, hogy nem keletkezik visszaél. Ha megmutatjuk a tétel második állítását, vagyis hogy a csúcsoknak a befejezési számozás szerinti fordított sorrendje topologikus rendezés, akkor ebből nyilván G aciklikussága is következni fog, így a bizonyítás teljes lesz. Azt kell tehát megmutatnunk, hogy G minden $e = (a,v)$ élére $\text{bszám}(v) < \text{bszám}(a)$.

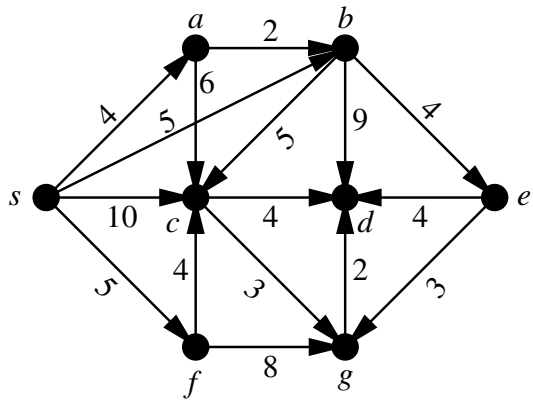
Mivel feltettük, hogy visszaél nem keletkezett, ezért e lehet faél, előreél vagy keresztél. Ha e faél vagy előreél, akkor v leszármazottja a-nak így a-ból v-be vezet olyan irányított út, aminek az u csúcsaira $\text{kezd}(u) \geq \text{kezd}(a)$. Így a 12.1. Lemma *(iii)* állítása szerint minden ilyen u csúcsra $\text{bef}(u) \leq \text{bef}(a)$. Ebből tehát $\text{bef}(v) < \text{bef}(a)$ és így $\text{bszám}(v) < \text{bszám}(a)$ is következik (hiszen egy korábban kapott befejezési szám nyilván kisebb is). Ha viszont e keresztél, akkor a 12.3. Tétel szerint $\text{bszám}(v) \neq *$ abban a pillanatban, amikor az eljárás e-t vizsgálja. Ebből következik, hogy $\text{bef}(v)$ ennél korábban volt; $\text{bef}(a)$ viszont nem lehetett korábban, hiszen épp egy a-ból induló él vizsgálata zajlik. Ebből tehát $\text{bef}(v) < \text{bef}(a)$ és így $\text{bszám}(v) < \text{bszám}(a)$ is adódik. □

A fenti tétel szerint tehát a DFS algoritmussal könnyen eldönthető, hogy G aciklikus-e és ha igen, megkapható G egy topologikus rendezése. Valóban, korábban már említettük, hogy a 12.3. Tételt használva az eljárás futásával párhuzamosan besorolható minden él a 12.2. Definíció szerinti osztályokba. Így eközben az is kiderül, hogy keletkezik-e visszaél – ami a fenti tétel szerint eldönti, hogy G aciklikus-e. Ha igen, akkor pedig a csúcsok $\text{bszám}(v)$ értékek szerinti csökkenő sorrendje topologikus rendezést ad. Ráadásul ezt a csökkenő sorrendet nem kell külön meghatározni, a DFS futásával párhuzamosan tárolhatjuk a csúcsok $\text{bszám}(v)$ értékek szerinti növekvő sorrendjét is (amit utána könnyen megfordíthatunk).

Ebből tehát következik, hogy aciklikus irányított gráfokban akkor is megoldható a legrövidebb út (és a leghosszabb út) probléma $c \cdot (n+m)$ futásidőben, ha G egy topologikus rendezése nem része a bemenetnek. A DFS algoritmusnak emellett számos további alkalmazása is ismert, de ezekre itt nem térünk ki.

12.7. Feladat.
a) Futtassuk a DFS algoritmust a 12.3. ábra gráfjára (az s csúcsból indítva) és határozzuk meg a futáshoz tartozó mélységi és befejezési számokat, a DFS-erdőt, valamint az abba nem tartozó élek 12.2. Definíció szerinti besorolását.
b) Döntsük el, hogy a gráf aciklikus-e és ha igen, adjuk meg egy topologikus rendezését.
c) Számítsuk ki az s csúcsból a többi csúcsba menő legrövidebb és leghosszabb utak hosszát, ahol a $w(e)$ élsúlyok az éleken látható számok.

Megoldás: A DFS eljárásnak nyilván számos helyes futása van, hiszen *aktív_csúcs*-ból mindig bármelyik, addig még bejáratlan élen továbbléphet. Az alábbi táblázatban látható adatokat eredményező futtatás során az ilyen esetekben mindig az ábécé

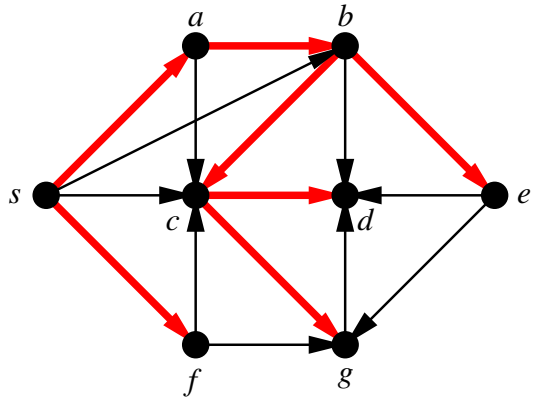


12.3. ábra

szerinti első csúcsba léptünk tovább.

v	s	a	b	c	d	e	f	g
mszám(v)	1	2	3	4	5	7	8	6
bszám(v)	8	6	5	3	1	4	7	2
előző(v)	$*$	s	a	b	c	b	s	c

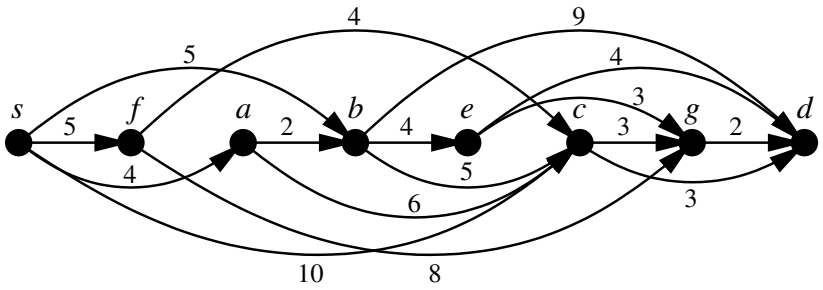
Az $(\text{előző}(v), v)$ élekből álló, a 12.4. ábrán pirossal látható DFS-erdő ismét egy fa, mert s-ből minden csúcs elérhető volt irányított úton.



12.4. ábra

A DFS-fába nem tartozó élek közül (s,b), (s,c), (a,c) és (b,d) előreélek, hiszen mindegyik esetben van irányított út az él kezdőpontjától a végpontjáig a DFS-fában. Az (e,d), (e,g), (g,d), (f,c) és (f,g) élek viszont keresztélek, mert egyik végpontjukból sem vezet irányított út a másikba a DFS-fában.

Mivel ezek szerint visszaél nincs, ezért a 12.6. Tétel szerint a gráf aciklikus és a csúcsok befejezési számozás szerint csökkenő sorrendje topologikus rendezést ad; ebben az esetben ez tehát az (s,f,a,b,e,c,g,d) sorrend. A 12.5. ábrán átrajzoltuk a gráfot úgy, hogy a csúcsok balról jobbra ebben a sorrendben következnek és valóban látható, hogy a gráf minden éle a topologikus rendezés definíciójának megfelelően balról jobbra vezet.



12.5. ábra

Minden v csúcsra jelölje táv(v), illetve maxút(v) az s-ből v-be vezető legrövidebb, illetve leghosszabb út hosszát. Ezeket a 12.2. szakaszban látott algoritmust követve, a topologikus rendezés sorrendje szerint haladva számítjuk: táv$(s)=$ maxút$(s)=0$, majd táv$(f)=$ maxút$(f)=5$ és táv$(a)=$ maxút$(a)=4$, mert f-be és a-ba is csak egy-egy él lép be. b-nél viszont már nincs egyenlőség: táv$(b)=\min\{0+5,4+2\}=5$ és maxút$(b)=\max\{0+5,4+2\}=6$. e-nél megint csak egy érték minimumát, illetve maximumát vesszük: táv$(e)=5+4=9$ és maxút$(e)=6+4=10$. Hasonlóan folytatva:

$$\begin{aligned}
\text{táv}(c) &= \min\{0+10,5+4,4+6,5+5\}=9,\\
\text{táv}(g) &= \min\{5+8,9+3,9+3\}=12 \text{ és}\\
\text{táv}(d) &= \min\{5+9,9+4,9+3,12+2\}=13, \text{ illetve}\\
\text{maxút}(c) &= \max\{0+10,5+4,4+6,6+5\}=11,\\
\text{maxút}(g) &= \max\{5+8,10+3,11+3\}=14 \text{ és}\\
\text{maxút}(d) &= \max\{6+9,10+4,11+3,14+2\}=16.
\end{aligned}$$

□