Gráfok és algoritmusok

Minimális vágások keresése

2025. április 8.

Minimális vágás keresése

Def: $G = (V, E)$ irányítatlan gráfban $X, Y \subseteq V$ esetén $E(X, Y)$ az $X - Y$ és $Y - X$ között futó élek halmaza. Adott $c : E \to \mathbb{R}_+$ kapacitásfüggvény mellett
$d_c(X, Y) = \tilde{c}(E(X, Y)) := \sum\{c(e) : e \in E(X, Y)\}$,
$d_c(X) := d_c(X, V - X)$, $\lambda_c(x, y) = \min\{d_c(X) : x \in X \not\ni y\}$ és
$\lambda_c(G) = \min\{\lambda_c(x, y) : x, y \in V(G)\}$.
($c \equiv 1$ esetén a jelölés egyszerűen $d(X, Y)$, $d(X)$, $\lambda(x, y)$, stb.)
A G **minimális vágása** olyan $X \subset V$, amire $d_c(X) = \lambda_c(G)$.
Cél: Hatékony algoritmus minimális vágás keresésére.

Minimális vágás keresése

Def: $G = (V, E)$ irányítatlan gráfban $X, Y \subseteq V$ esetén $E(X, Y)$ az $X - Y$ és $Y - X$ között futó élek halmaza. Adott $c : E \to \mathbb{R}_+$ kapacitásfüggvény mellett
$d_c(X, Y) = \tilde{c}(E(X, Y)) := \sum\{c(e) : e \in E(X, Y)\}$,
$d_c(X) := d_c(X, V - X)$, $\lambda_c(x, y) = \min\{d_c(X) : x \in X \not\ni y\}$ és
$\lambda_c(G) = \min\{\lambda_c(x, y) : x, y \in V(G)\}$.
($c \equiv 1$ esetén a jelölés egyszerűen $d(X, Y)$, $d(X)$, $\lambda(x, y)$, stb.)
A G **minimális vágása** olyan $X \subset V$, amire $d_c(X) = \lambda_c(G)$.
Cél: Hatékony algoritmus minimális vágás keresésére.
Első ötlet: Folyamalgoritmus. Bármely s, t csúcspárhoz találunk minimális st-vágást. A globális minimumhoz minden (s, t) párra meghatározzuk $\lambda_c(s, t)$-t, azaz $\binom{|V|}{2}$-szer futtatjuk az algoritmust.

Minimális vágás keresése

Def: $G = (V, E)$ irányítatlan gráfban $X, Y \subseteq V$ esetén $E(X, Y)$ az $X - Y$ és $Y - X$ között futó élek halmaza. Adott $c : E \to \mathbb{R}_+$ kapacitásfüggvény mellett
$d_c(X, Y) = \tilde{c}(E(X, Y)) := \sum\{c(e) : e \in E(X, Y)\}$,
$d_c(X) := d_c(X, V - X)$, $\lambda_c(x, y) = \min\{d_c(X) : x \in X \not\ni y\}$ és
$\lambda_c(G) = \min\{\lambda_c(x, y) : x, y \in V(G)\}$.
($c \equiv 1$ esetén a jelölés egyszerűen $d(X, Y)$, $d(X)$, $\lambda(x, y)$, stb.)
A G **minimális vágása** olyan $X \subset V$, amire $d_c(X) = \lambda_c(G)$.
Cél: Hatékony algoritmus minimális vágás keresésére.
Első ötlet: Folyamalgoritmus. Bármely s, t csúcspárhoz találunk minimális st-vágást. A globális minimumhoz minden (s, t) párra meghatározzuk $\lambda_c(s, t)$-t, azaz $\binom{|V|}{2}$-szer futtatjuk az algoritmust.
Második ötlet: A folyamalgoritmust elég csupán $(|V| - 1)$-szer lefuttatni, mert s-et rögzíthetjük.

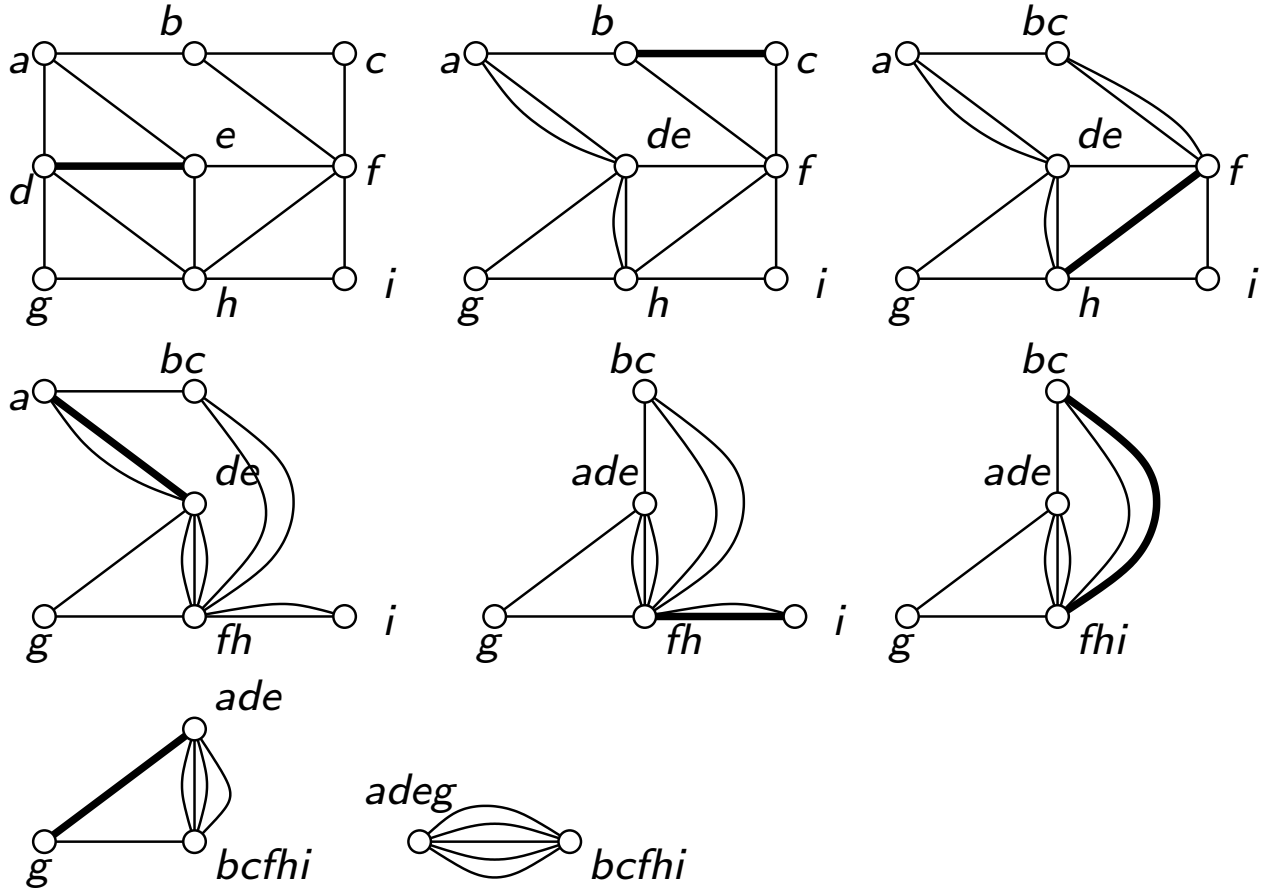
Minimális vágás keresése

Def: $G = (V, E)$ irányítatlan gráfban $X, Y \subseteq V$ esetén $E(X, Y)$ az $X - Y$ és $Y - X$ között futó élek halmaza. Adott $c : E \to \mathbb{R}_+$ kapacitásfüggvény mellett
$d_c(X, Y) = \tilde{c}(E(X, Y)) := \sum\{c(e) : e \in E(X, Y)\}$,
$d_c(X) := d_c(X, V - X)$, $\lambda_c(x, y) = \min\{d_c(X) : x \in X \not\ni y\}$ és
$\lambda_c(G) = \min\{\lambda_c(x, y) : x, y \in V(G)\}$.
($c \equiv 1$ esetén a jelölés egyszerűen $d(X, Y)$, $d(X)$, $\lambda(x, y)$, stb.)
A G **minimális vágása** olyan $X \subset V$, amire $d_c(X) = \lambda_c(G)$.
Cél: Hatékony algoritmus minimális vágás keresésére.
Első ötlet: Folyamalgoritmus. Bármely s, t csúcspárhoz találunk minimális st-vágást. A globális minimumhoz minden (s, t) párra meghatározzuk $\lambda_c(s, t)$-t, azaz $\binom{|V|}{2}$-szer futtatjuk az algoritmust.
Második ötlet: A folyamalgoritmust elég csupán $(|V| - 1)$-szer lefuttatni, mert s-et rögzíthetjük.

Ez mind szép és jó, de mi ennél hatékonyabb eljárást szeretnénk. Azt lehetne pl. kihasználni, hogy G irányítatlan.

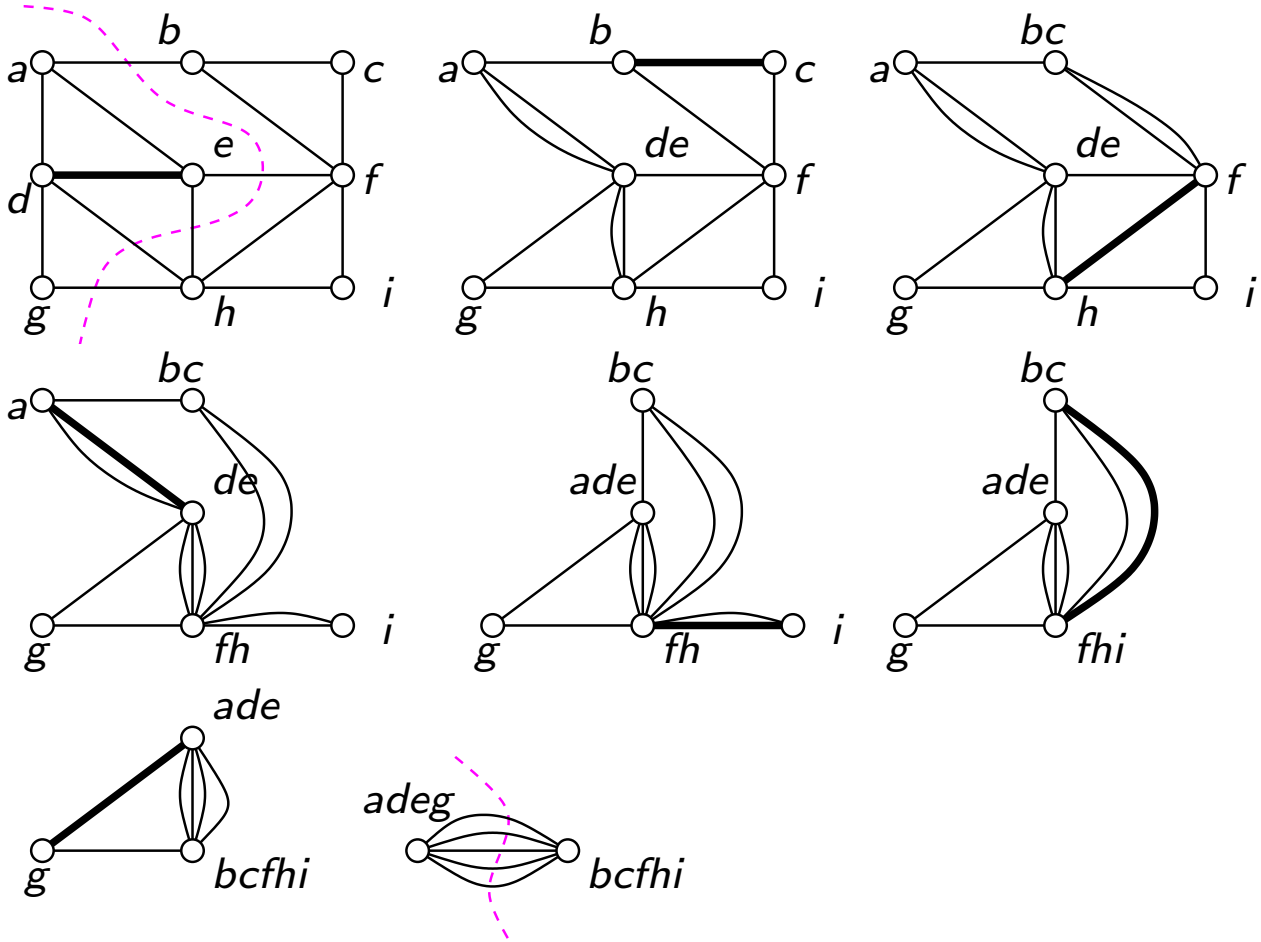
Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.



Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.



Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.

Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.
Tétel: Legyen X a G egy minimális vágása. Karger fenti eljárása legalább $1/\binom{n}{2}$ valószínűséggel X-et találja meg, ahol $n = |V(G)|$.
Köv: (1) $P(kn^2$ futtatás után sosem X az output$) \leq e^{-2k}$.
(2) n csúcsú gráfnak legfeljebb $\binom{n}{2}$ minimális vágása van.
(A korlát éles: a körnek pl. épp ennyi van.)

Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.
Tétel: Legyen X a G egy minimális vágása. Karger fenti eljárása legalább $1/\binom{n}{2}$ valószínűséggel X-et találja meg, ahol $n = |V(G)|$.

Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.
Tétel: Legyen X a G egy minimális vágása. Karger fenti eljárása legalább $1/\binom{n}{2}$ valószínűséggel X-et találja meg, ahol $n = |V(G)|$.
Biz: k csúcsú gráfban az egypontú vágások minden élt kétszer használnak, így $kd_c(X) \leq \sum_{\forall v} \tilde{c}(E(v)) = 2\tilde{c}(E)$ teljesül. Annak a valószínűsége, hogy X a random élösszehúzást túléli

$$\frac{\tilde{c}(E) - d_c(X)}{\tilde{c}(E)} = \frac{k\tilde{c}(E) - kd_c(X)}{k\tilde{c}(E)} \geq \frac{k\tilde{c}(E) - 2\tilde{c}(E)}{k\tilde{c}(E)} = \frac{k-2}{k} .$$

Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.
Tétel: Legyen X a G egy minimális vágása. Karger fenti eljárása legalább $1/\binom{n}{2}$ valószínűséggel X-et találja meg, ahol $n = |V(G)|$.
Biz: k csúcsú gráfban az egypontú vágások minden élt kétszer használnak, így $kd_c(X) \leq \sum_{\forall v} \tilde{c}(E(v)) = 2\tilde{c}(E)$ teljesül. Annak a valószínűsége, hogy X a random élösszehúzást túléli

$$\frac{\tilde{c}(E) - d_c(X)}{\tilde{c}(E)} = \frac{k\tilde{c}(E) - kd_c(X)}{k\tilde{c}(E)} \geq \frac{k\tilde{c}(E) - 2\tilde{c}(E)}{k\tilde{c}(E)} = \frac{k-2}{k} .$$

Ezért

$$P(X \text{ az output}) \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \ldots \cdot \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)} .$$

Mi pedig pontosan ezt akartuk bizonyítani. □

Karger algoritmusa

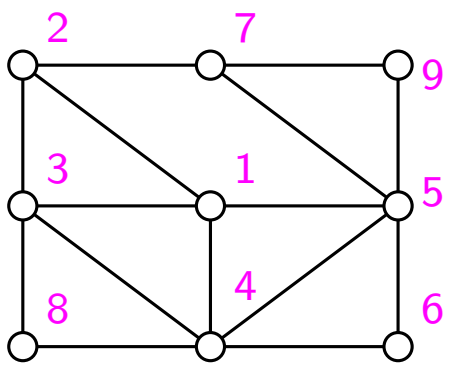
Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.
Tétel: Legyen X a G egy minimális vágása. Karger fenti eljárása legalább $1/\binom{n}{2}$ valószínűséggel X-et találja meg, ahol $n = |V(G)|$.

Karger algoritmusa

Hajtsuk végre a következő eljárást.
Amíg $|V| > 2$, válasszunk az aktuális c szerinti eloszlással random élt, és húzzuk össze. Ha csak két csúcs marad, megállunk. Ez a két csúcs G egy minimális vágás jelöltjét határozza meg.
Tétel: Legyen X a G egy minimális vágása. Karger fenti eljárása legalább $1/\binom{n}{2}$ valószínűséggel X-et találja meg, ahol $n = |V(G)|$.
Megj: (1) Karger algoritmusának hatékonyságát Stein ötlete egy nagyságrenddel megjavítja. Ugyanis míg nagy a gráf, drága egy él összehúzása. Kis gráfra már olcsó, de ekkor könnyen elveszhet a minvágás. Az értékes munkát jobban kihasználjuk, ha egy önmagát meghívó, rekurzív algoritmussal dolgozunk:
A random élösszehúzással nem 2, hanem $\frac{n}{\sqrt{2}}$ csúcsnál állunk le. A kapott kisebb gráfra kétszer meghívjuk ugyanezt az algoritmust.
Így csak konstanszor többet dolgozunk, mégis $1/\binom{n}{2}$ helyett legalább $\frac{1}{n}$ valószínűséggel találunk minvágást.
(2) Karger algoritmusa bár igen szellemes, nem determinisztikus.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Megf: G egy maxvissza sorrendje úgy kapható, hogy tetszőleges csúcsból kiindulva mindig azt a csúcsot választjuk következőnek, amelyik a legnagyobb összsúllyal kapcsolódik a korábban már kiválasztott csúcsokhoz.
Példa:

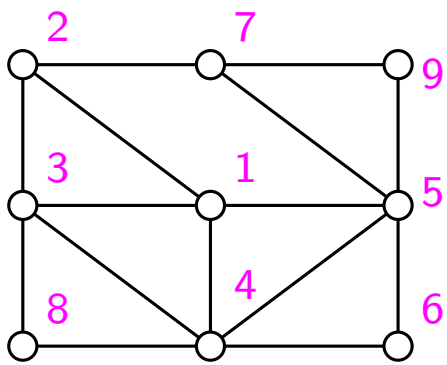


Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális. (Később bizonyítjuk.)



Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.
Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.

($G/v_n v_{n-1}$ itt azt a gráfot jelöli, amit G-ből úgy kapunk, hogy a v_{n-1} és v_n csúcsokat egybeolvasztjuk.)

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.

Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.
Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.
Biz: Ekkor $\lambda_c(G) < d_c(v_n) = \lambda_c(v_{n-1} v_n)$, azaz G egyetlen minvágása sem szeperálja a v_n és v_{n-1} csúcsokat. Ezért G minden minvágása túléli e két csúcs egybeolvasztását. Mivel $G/v_n v_{n-1}$ minden vágása megtalálható G-ben is, ezért e két gráf minimális vágásai megegyeznek. □

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.

Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.

Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.

Köv: G minvágása vagy $\{v_n\}$ vagy $G/v_n v_{n-1}$ egy minvágása.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.
Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.
Köv: G minvágása vagy $\{v_n\}$ vagy $G/v_n v_{n-1}$ egy minvágása.
Nagamochi és Ibaraki algoritmusa Input: $G = (V, E), c \in \mathbb{R}_+^E$
Output: $\lambda_c(G)$ értéke és G egy X minvágása.
Működés 1. Elkészítjük G egy $v_1, v_2, \ldots, v_n$ maxvissza sorrendjét.
2. Rekurzívan meghatározzuk $G' = G/v_{n-1}v_n$ egy X' minvágását.
3. Ha $\lambda_c(G') < d_c(v_n)$, akkor az output $\lambda_c(G')$ és X' G-beli őse.
4. Különben az output $\lambda_c(G) = d_c(v_n)$ és $X = \{v_n\}$.

Minvágás maxvissza sorrenddel

Lássunk egy determinisztikus algoritmust ugyanerre a problémára.
Def: A $G = (V, E)$ gráfnak a c élsúlyozás mellett $v_1, v_2, \ldots, v_n$ egy **maxvissza sorrendje**, ha $d_c(v_{i+1}, V_i) \geq d_c(v_j, V_i)$ teljesül minden $1 \leq i < j \leq n$ esetén, ahol $V_i = \{v_1, v_2, \ldots, v_i\}$.
Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.
Köv: Ha $v_1, v_2, \ldots, v_n$ a G maxvissza sorrendje, és $\{v_n\}$ nem minimális vágás akkor G és $G/v_n v_{n-1}$ minvágásai megegyeznek.
Köv: G minvágása vagy $\{v_n\}$ vagy $G/v_n v_{n-1}$ egy minvágása.
Nagamochi és Ibaraki algoritmusa Input: $G = (V, E), c \in \mathbb{R}_+^E$
Output: $\lambda_c(G)$ értéke és G egy X minvágása.
Működés 1. Elkészítjük G egy $v_1, v_2, \ldots, v_n$ maxvissza sorrendjét.
2. Rekurzívan meghatározzuk $G' = G/v_{n-1}v_n$ egy X' minvágását.
3. Ha $\lambda_c(G') < d_c(v_n)$, akkor az output $\lambda_c(G')$ és X' G-beli őse.
4. Különben az output $\lambda_c(G) = d_c(v_n)$ és $X = \{v_n\}$.
Alg helyessége: A lemmán múlik. Ennek alaposan nekifutunk.

Scan First Search bejárások és folytonos sorrendek

Egy bejárási algoritmus elején az input gráf csúcsai eléretlenek. Az algoritmus lefutása során minden csúcs az eléretlen-elért-befejezett evolúción megy keresztül. Általános lépés esetek szerint:

1. Van elért csúcs, mondjuk u.

a Ha $\exists uv \in E$, v eléretlen, akkor az uv él mentén v elértté válik.

b Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs a Van eléretlen csúcs, v. Ekkor v elértté válik.

b Eléretlen csúcs sincs. Ekkor minden csúcs befejezett, STOP.

Scan First Search bejárások és folytonos sorrendek

Egy bejárási algoritmus elején az input gráf csúcsai eléretlenek. Az algoritmus lefutása során minden csúcs az eléretlen-elért-befejezett evolúción megy keresztül. Általános lépés esetek szerint:

1. Van elért csúcs, mondjuk u.

a Ha $\exists uv \in E$, v eléretlen, akkor az uv él mentén v elértté válik.

b Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs a Van eléretlen csúcs, v. Ekkor v elértté válik.

b Eléretlen csúcs sincs. Ekkor minden csúcs befejezett, STOP.

DFS: 1-ben u a legkésőbb elért csúcs.

BFS: 1-ben u a legkorábban elért csúcs.

Scan First Search bejárások és folytonos sorrendek

Egy bejárási algoritmus elején az input gráf csúcsai eléretlenek. Az algoritmus lefutása során minden csúcs az eléretlen-elért-befejezett evolúción megy keresztül. Általános lépés esetek szerint:
1. Van elért csúcs, mondjuk u.
a Ha $\exists uv \in E$, v eléretlen, akkor az uv él mentén v elértté válik.
b Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs a Van eléretlen csúcs, v. Ekkor v elértté válik.
b Eléretlen csúcs sincs. Ekkor minden csúcs befejezett, STOP.
DFS: 1-ben u a legkésőbb elért csúcs.
BFS: 1-ben u a legkorábban elért csúcs.
SFS: 1' Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

Scan First Search bejárások és folytonos sorrendek

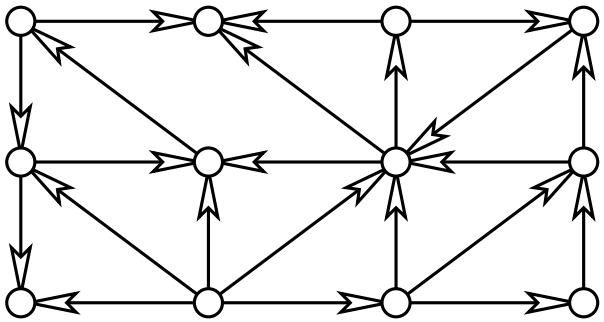
Egy bejárási algoritmus elején az input gráf csúcsai eléretlenek. Az algoritmus lefutása során minden csúcs az eléretlen-elért-befejezett evolúción megy keresztül. Általános lépés esetek szerint:
1. Van elért csúcs, mondjuk u.
a Ha $\exists uv \in E$, v eléretlen, akkor az uv él mentén v elértté válik.
b Ha nincs ilyen uv él, akkor u befejezetté válik.
2. Nincs elért csúcs a Van eléretlen csúcs, v. Ekkor v elértté válik.
b Eléretlen csúcs sincs. Ekkor minden csúcs befejezett, STOP.
DFS: 1-ben u a legkésőbb elért csúcs.
BFS: 1-ben u a legkorábban elért csúcs.
SFS: 1' Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.
Megj: (1) A BFS az SFS-nek az a speciális esete, amikor 1'-ben mindig a legkorábban elért csúcsot választjuk.
(2) Az SFS bejárás magyar neve **lokális szélességi keresés.**

Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

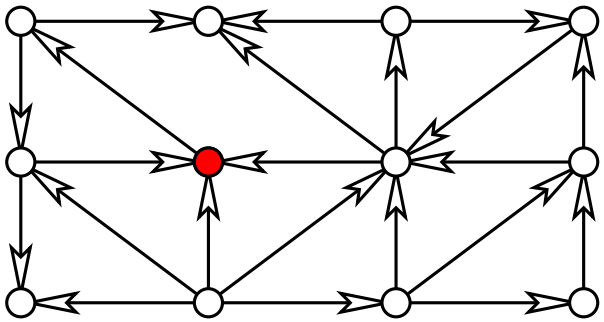
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



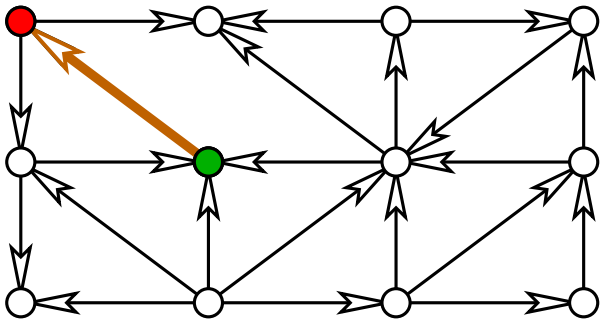
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



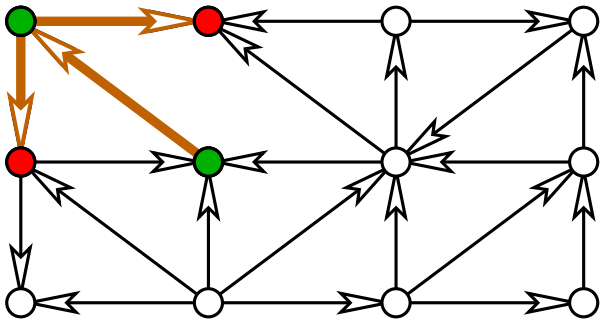
Scan First Search bejárások és folytonos sorrendek

SFS: 1' Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



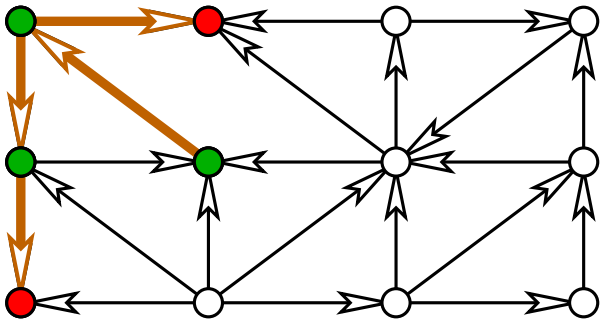
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



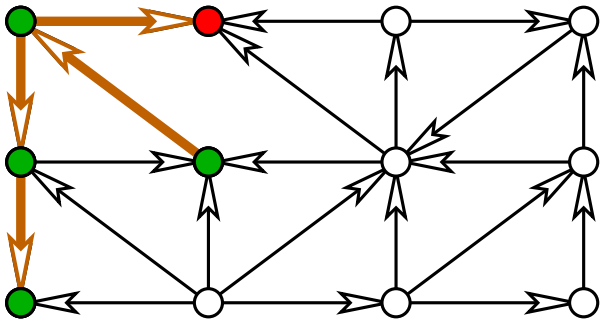
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



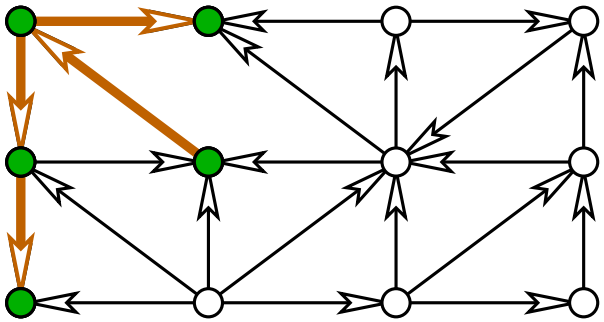
Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



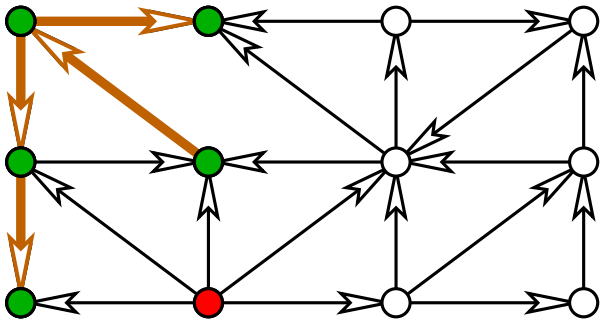
Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



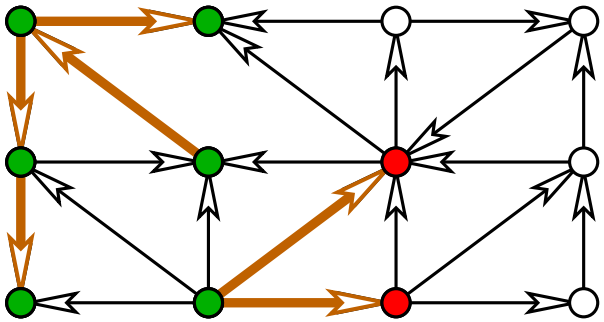
Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



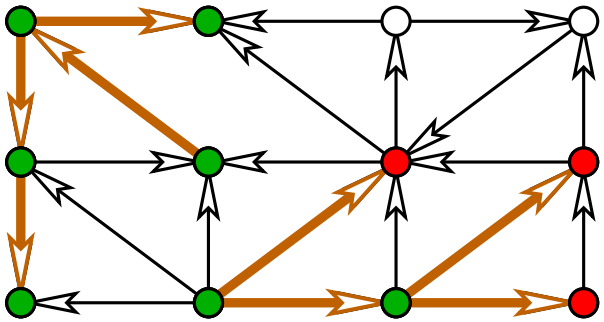
Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



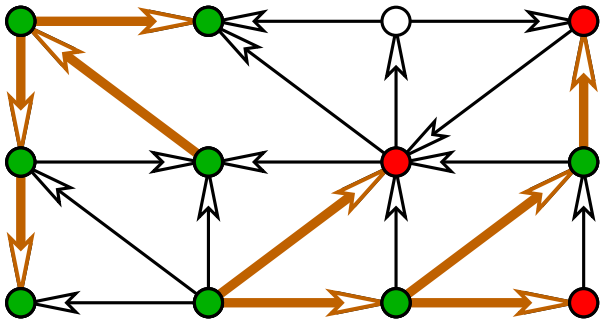
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



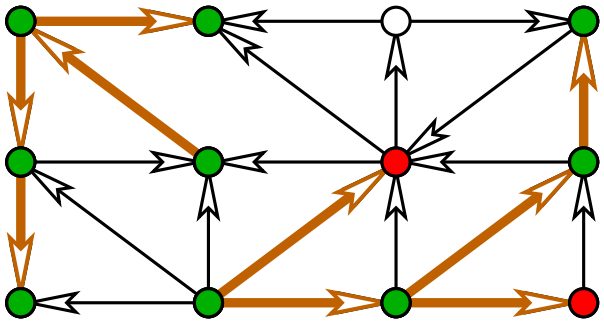
Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



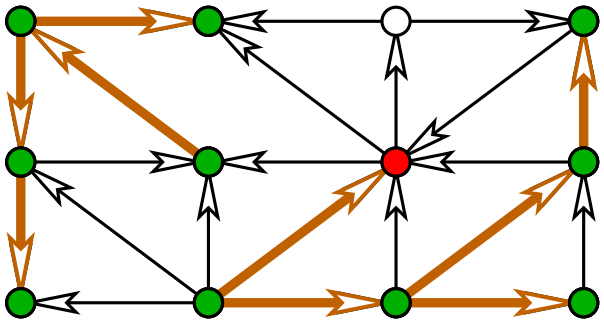
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



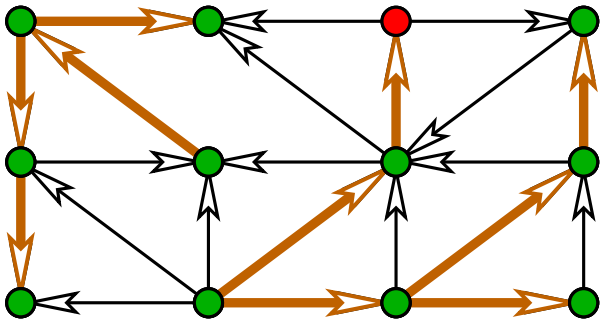
Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



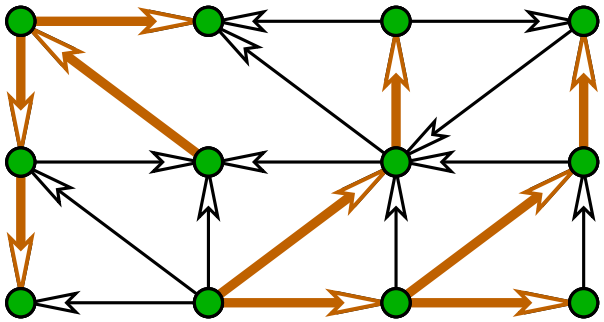
Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



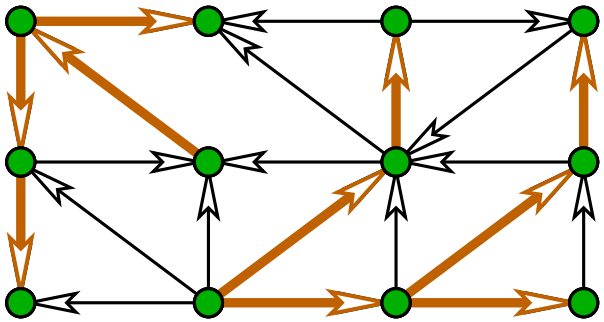
Scan First Search bejárások és folytonos sorrendek

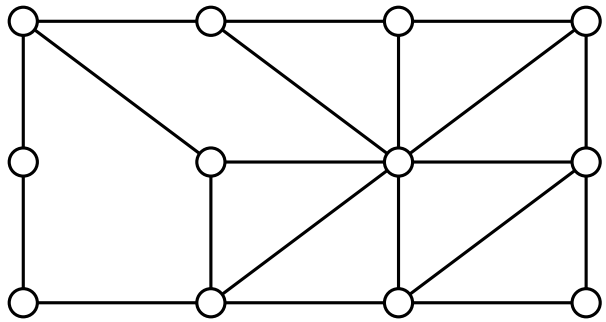
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.



Scan First Search bejárások és folytonos sorrendek

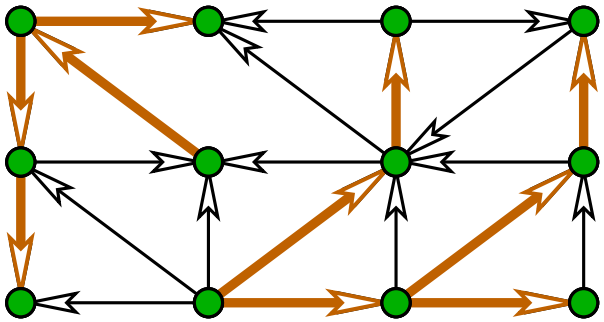
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

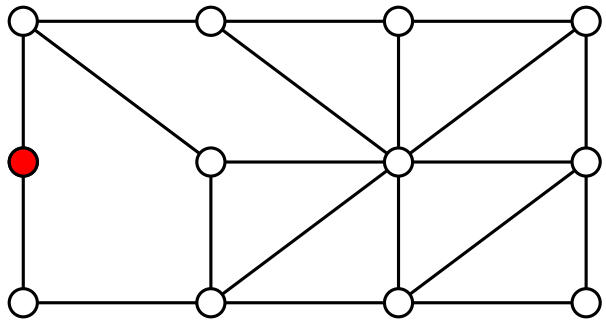




Scan First Search bejárások és folytonos sorrendek

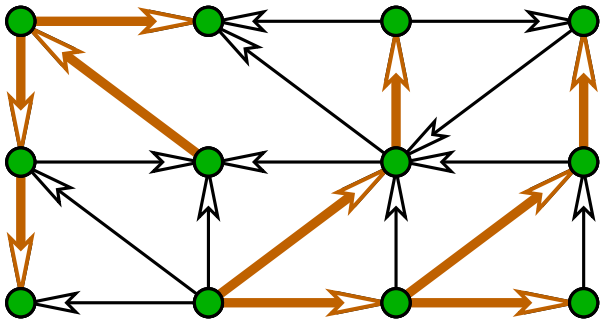
SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

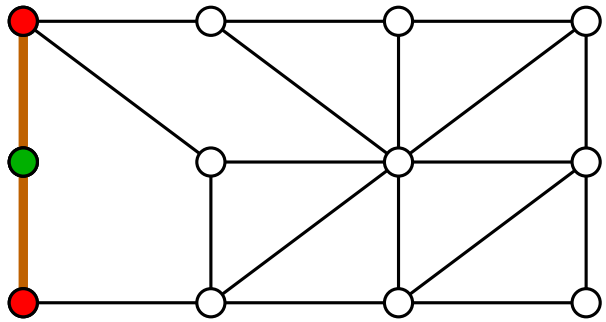




Scan First Search bejárások és folytonos sorrendek

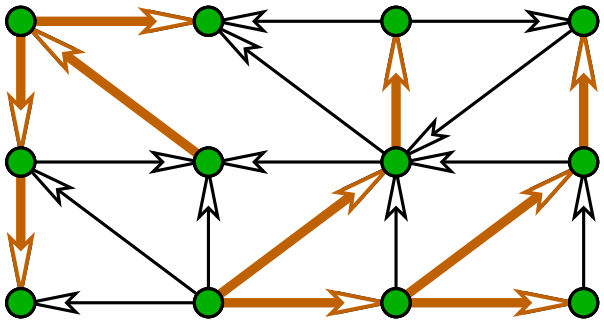
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

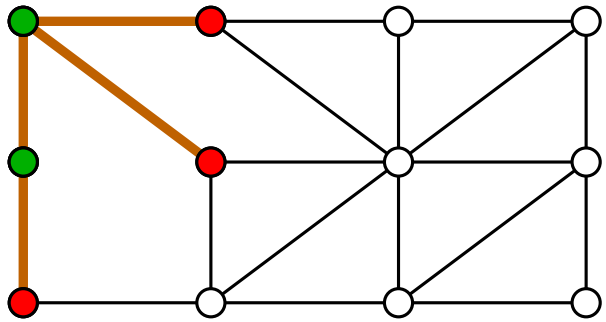




Scan First Search bejárások és folytonos sorrendek

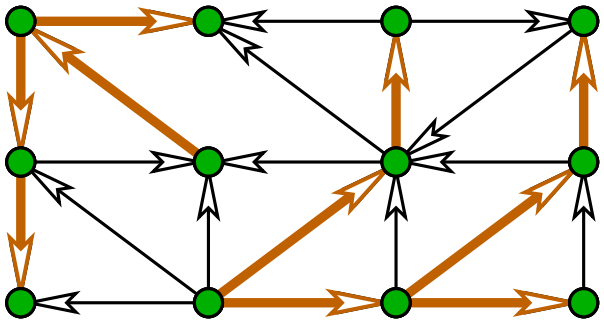
SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

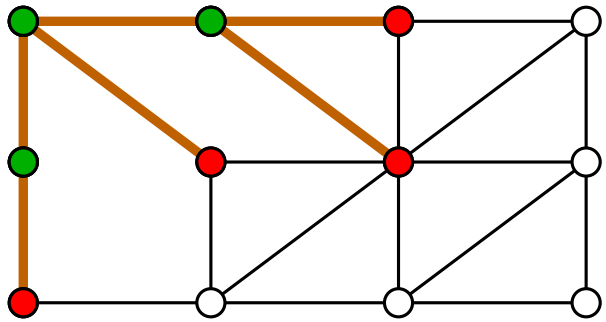




Scan First Search bejárások és folytonos sorrendek

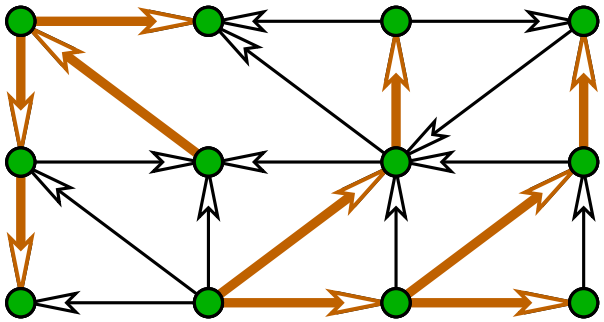
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

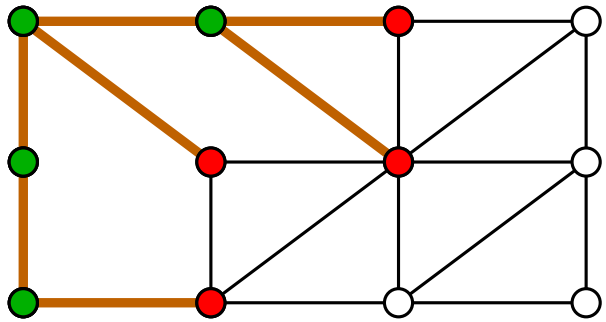




Scan First Search bejárások és folytonos sorrendek

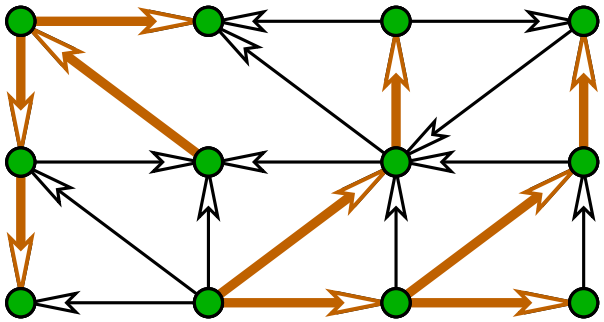
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

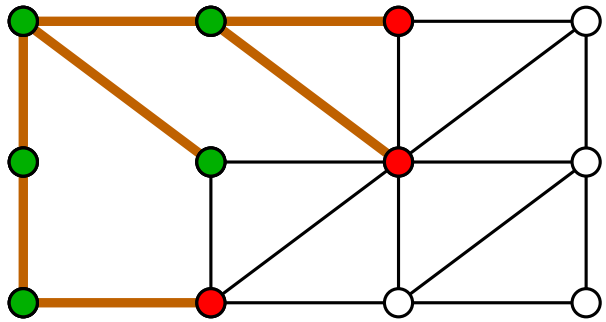




Scan First Search bejárások és folytonos sorrendek

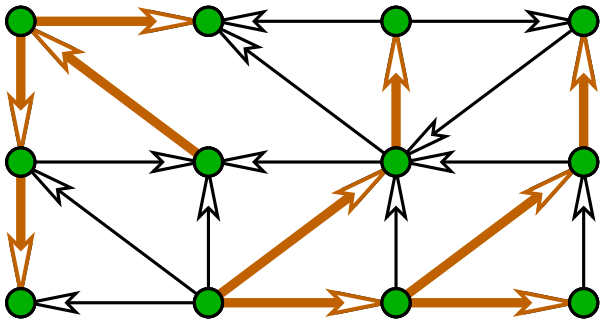
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

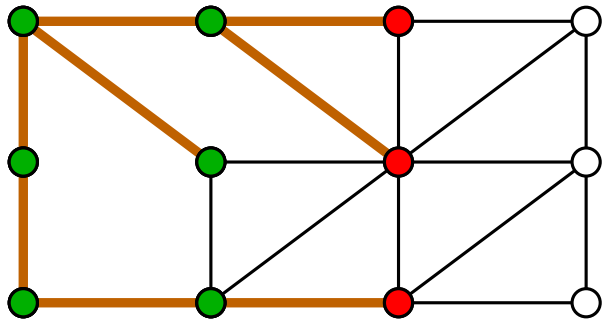




Scan First Search bejárások és folytonos sorrendek

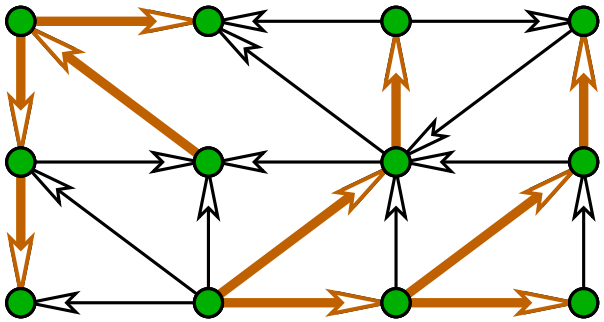
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

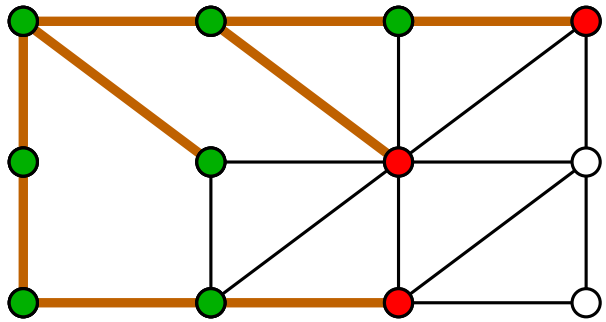




Scan First Search bejárások és folytonos sorrendek

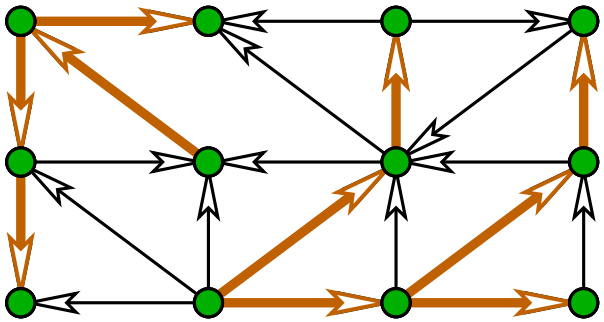
SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

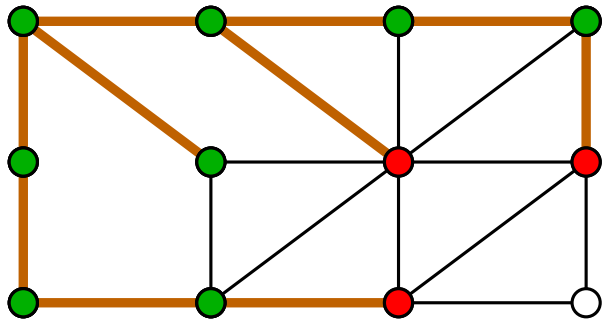




Scan First Search bejárások és folytonos sorrendek

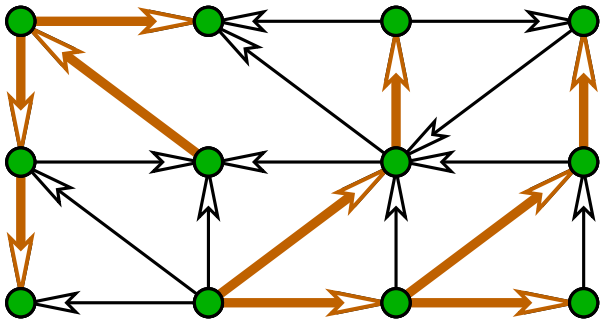
SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

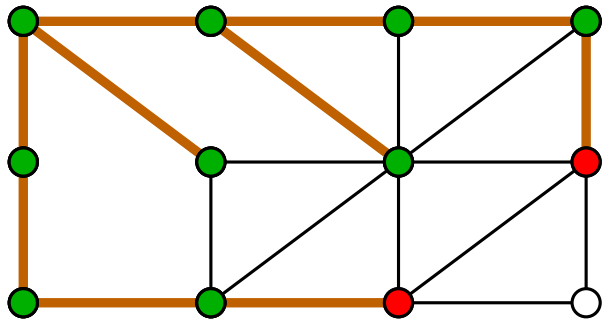




Scan First Search bejárások és folytonos sorrendek

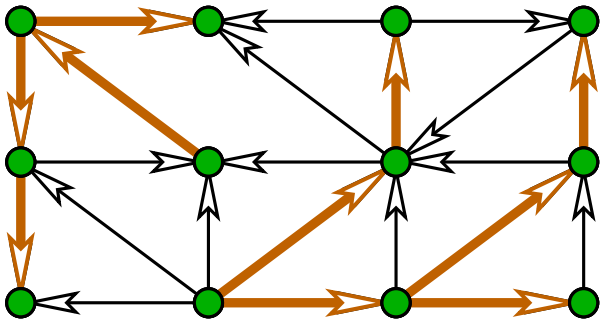
SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

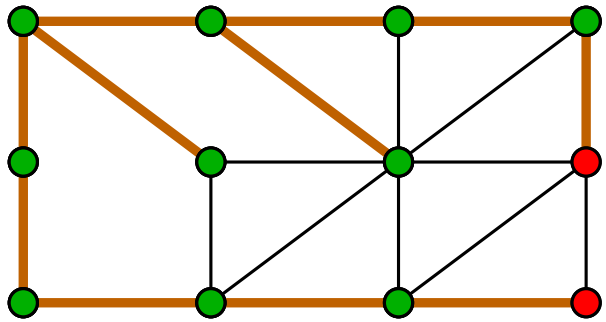




Scan First Search bejárások és folytonos sorrendek

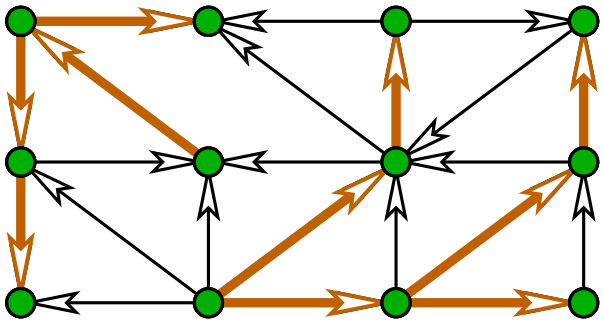
SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

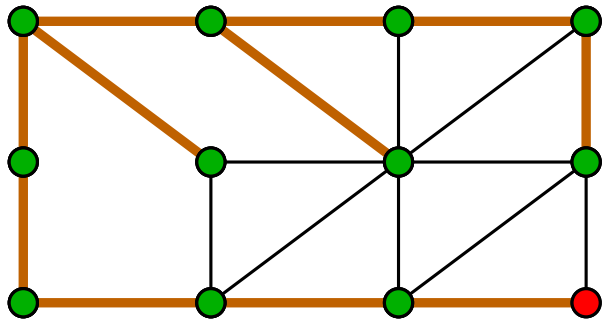




Scan First Search bejárások és folytonos sorrendek

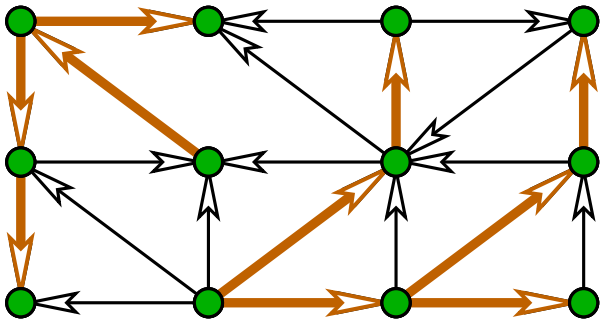
SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

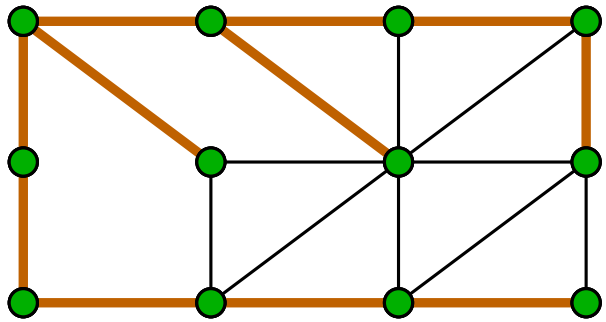




Scan First Search bejárások és folytonos sorrendek

SFS: 1′ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.





Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.

Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik. Az SFS bejárás befejezési sorrendje jól karakterizálható.

Def: A G gráf csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje **folytonos**, ha G minden K komponensére igaz, hogy K csúcsai intervallumot alkotnak ebben a sorrendben és K legelső csúcsa kivételével K minden csúcsának van a sorrendben korábbi szomszédja. A **folytonos sorrendhez tartozó erdőt** minden csúcsból az őt megelőző, legkorábbi szomszédjába futó élek alkotják.

Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik. Az SFS bejárás befejezési sorrendje jól karakterizálható.

Def: A G gráf csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje **folytonos**, ha G minden K komponensére igaz, hogy K csúcsai intervallumot alkotnak ebben a sorrendben és K legelső csúcsa kivételével K minden csúcsának van a sorrendben korábbi szomszédja. A **folytonos sorrendhez tartozó erdőt** minden csúcsból az őt megelőző, legkorábbi szomszédjába futó élek alkotják.

Tétel: (1) A G csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje pontosan akkor folytonos, ha G egy SFS bejárásának befejezési sorrendje.

Scan First Search bejárások és folytonos sorrendek

SFS: 1' Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.
Az SFS bejárás befejezési sorrendje jól karakterizálható.
Def: A G gráf csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje **folytonos**, ha G minden K komponensére igaz, hogy K csúcsai intervallumot alkotnak ebben a sorrendben és K legelső csúcsa kivételével K minden csúcsának van a sorrendben korábbi szomszédja. A **folytonos sorrendhez tartozó erdőt** minden csúcsból az őt megelőző, legkorábbi szomszédjába futó élek alkotják.
Tétel: (1) A G csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje pontosan akkor folytonos, ha G egy SFS bejárásának befejezési sorrendje.
Biz: A folytonos sorrenden végighaladva végrehajtható az SFS bejárás, mindig a folytonos sorrend soron következő csúcsát választva u-nak 1'-ben.
Az SFS befejezési sorrendjében G komponensei intervallumok, és minden nemgyökér csúcsot megelőzi az a szomszédja, ahonnan az SFS bejárás elérte. Így az SFS befejezési sorrend folytonos. □

Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik. Az SFS bejárás befejezési sorrendje jól karakterizálható.

Def: A G gráf csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje **folytonos**, ha G minden K komponensére igaz, hogy K csúcsai intervallumot alkotnak ebben a sorrendben és K legelső csúcsa kivételével K minden csúcsának van a sorrendben korábbi szomszédja. A **folytonos sorrendhez tartozó erdőt** minden csúcsból az őt megelőző, legkorábbi szomszédjába futó élek alkotják.

Tétel: (1) A G csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje pontosan akkor folytonos, ha G egy SFS bejárásának befejezési sorrendje.

Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik.
Az SFS bejárás befejezési sorrendje jól karakterizálható.
Def: A G gráf csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje **folytonos**, ha G minden K komponensére igaz, hogy K csúcsai intervallumot alkotnak ebben a sorrendben és K legelső csúcsa kivételével K minden csúcsának van a sorrendben korábbi szomszédja. A **folytonos sorrendhez tartozó erdőt** minden csúcsból az őt megelőző, legkorábbi szomszédjába futó élek alkotják.
Tétel: (1) A G csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje pontosan akkor folytonos, ha G egy SFS bejárásának befejezési sorrendje.
(2) Az SFS bejáráshoz tartozó erdő megegyezik a befejezési sorrendjéhez (mint folytonos sorrendhez) tartozó erdővel.

Scan First Search bejárások és folytonos sorrendek

SFS: $\boxed{1'}$ Van elért csúcs, mondjuk u. **Minden** uv élre, amire v eléretlen, v elértté válik az uv mentén, majd u befejezetté válik. Az SFS bejárás befejezési sorrendje jól karakterizálható.

Def: A G gráf csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje **folytonos**, ha G minden K komponensére igaz, hogy K csúcsai intervallumot alkotnak ebben a sorrendben és K legelső csúcsa kivételével K minden csúcsának van a sorrendben korábbi szomszédja. A **folytonos sorrendhez tartozó erdőt** minden csúcsból az őt megelőző, legkorábbi szomszédjába futó élek alkotják.

Tétel: (1) A G csúcsainak $v_1, v_2, \ldots, v_n$ sorrendje pontosan akkor folytonos, ha G egy SFS bejárásának befejezési sorrendje.
(2) Az SFS bejáráshoz tartozó erdő megegyezik a befejezési sorrendjéhez (mint folytonos sorrendhez) tartozó erdővel.

Biz: Az SFS erdőben minden (nemgyökér) csúcs őse az a szomszédja, amelyiket először fejeztük be az SFS bejárás során. □

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.

(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.

(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

Biz: (1) A maxvissza sorrendben minden komponens intervallum, és minden komponens elsőtől különböző csúcsának van korábbi szomszédja a komponensben.

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.
(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

Biz: (1) A maxvissza sorrendben minden komponens intervallum, és minden komponens elsőtől különböző csúcsának van korábbi szomszédja a komponensben.
(2) Minden V_i-hez G-ben csatlakozó, $u \notin V_i$ csúcsra $d_{G-F_1}(u, V_i) = d(u, V_i) - 1$ az F_1 definíciója alapján. Ha pedig u nem csatlakozik V_i-hez, akkor $d_{G-F_1}(u, V_i) = d(u, V_i) = 0$. Ezért $G - F_1$ maxvissza sorrendjének készítésekor tudunk $v_1, v_2, \ldots, v_n$ sorrendben haladni. □

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.

(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.

(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.
(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

Megj: Itt csak a $c \equiv \mathbf{1}$ esetet bizonyítjuk. Ha c egész, akkor minden e él helyett bevezethetünk $c(e)$ párhuzamos élt, és az így kapott a gráfban dolgozunk. Ha pedig c racionális, akkor a közös nevezővel történő végigszorzással a minvágások ugyanazok maradnak, és c egész lesz.
(Más trükk is működik, ami a fentieken túl az irracionális esetet is kezeli, de ezzel itt nem foglalkozunk.)

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.

(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

A maxvissza sorrend folytonossága

Állítás: (1) Minden maxvissza sorrend folytonos.
(2) Ha F_1 a $v_1, v_2, \ldots, v_n$ maxvissza sorrendhez tartozó erdő, akkor $v_1, v_2, \ldots, v_n$ a $G - F_1$ gráfnak is maxvissza sorrendje.

Lemma: Ha $v_1, v_2, \ldots, v_n$ a G c élsúlyozás melletti maxvissza sorrendje, akkor $\lambda_c(v_{n-1}, v_n) = d_c(v_n)$, azaz a v_{n-1}-et és v_n-t szeparáló vágások között $\{v_n\}$ minimális.

Biz: Mivel $\lambda(v_{n-1}, v_n) \leq d(v_n) =: k$, ezért elegendő megmutatni, hogy $\lambda(v_{n-1}, v_n) \geq k$, azaz v_{n-1} és v_n között van k éldiszjunkt út. Legyen a $v_i v_j$ él címkéje p, ha $i < j$ és v_i a p-dik a v_j szomszédai között a maxvissza sorrendben. $F_p := \{p \text{ címkéjű élek}\}$. Ekkor a maxvissza sorrendhez tartozó erdő G-ben F_1, $G - F_1$-ben F_2, sít. A $G - F_1 - F_2 - \ldots - F_\ell$ gráfnak $v_1, v_2, \ldots, v_n$ minden ℓ-re folytonos sorrendje, így v_n és v_{n-1} F_k-nak egyazon komponensébe esik. Ezért v_n és v_{n-1} komponense az $F_1, F_2, \ldots, F_{k-1}$ erdők mindegyikében közös. Vagyis az $F_1, F_2, \ldots, F_k$ erdők mindegyike tartalmaz egy-egy $v_{n-1}v_n$-útat. Ezek az utak pként éldiszjunktak, így $d(v_n) = k \leq \lambda(v_{n-1}, v_n) \leq d(v_n)$. □

Mit tanultunk ma?

- Nemnegatív élkapacitásokkal ellátott gráf globális és lokális minimális vágásai.
- Globális minimális vágás keresése n^2 ill. n folyammal.
- Karger algoritmusa súlyozás szerint véletlenül választott él összehúzásával.
- Nagamochi-Ibaraki-algoritmus maxvissza sorrendel.
- Maxvissza sorrend utolsó két csúcsának különös tulajdonsága.
- SFS bejárások, folytonos sorrendek.
- SFS bejáráshoz tartozó F_i erdők, $G - F_1$ maxvissza sorrendje.
- A NI-algoritmus kulcslemmájának igazolása SFS bejárások erdeivel.

Mit tanultunk ma?

- Nemnegatív élkapacitásokkal ellátott gráf globális és lokális minimális vágásai.
- Globális minimális vágás keresése n^2 ill. n folyammal.
- Karger algoritmusa súlyozás szerint véletlenül választott él összehúzásával.
- Nagamochi-Ibaraki-algoritmus maxvissza sorrendel.
- Maxvissza sorrend utolsó két csúcsának különös tulajdonsága.
- SFS bejárások, folytonos sorrendek.
- SFS bejáráshoz tartozó F_i erdők, $G - F_1$ maxvissza sorrendje.
- A NI-algoritmus kulcslemmájának igazolása SFS bejárások erdeivel.

Ennyit mára a természettudományok újdonságaiból, érdekességeiből