Pár félévnyi algel kurzus széljegyzete

avagy tanácsok a számonkérésekre

v0.7 „prebéta" (2012.12.13.) – **fenntartásokkal kezelendő!!**

ha valaki hibát vél felfedezni, vagy ki tudja egészíteni, jelezze: kazi@cs.bme.hu

1. Általánosságban

- amikor önmagában az „eredményből" nem látszanak a „lépések", írjatok indoklást!
  - pl: rajzolja fel a fát/kupacot (kell, hogy hogy jött ki)
  - egyébként sem haszontalan, mert ha valamit elnéztek, akkor menthet pár pontot
    - ha nincs, nehéz azt mondani, hogy igazából jó, csak sikerült elírni
- szokás szerint (lásd: BSZ) fontos odafigyelni arra, hogy a feladatot ne szűkítsétek le, bővítsétek ki egy-egy példára
  - ha egy kis részlet a „legrosszabb"/„legjobb" eset, és már ott is/sem működik egy adott dolog, akkor legyen ott annak a leírása, indoklása, hogy miért is a „legrosszabb"/„legjobb", miért következik abból, hogy mindig/sohasem működik valami
  - ha vannak olyan részek, amit ki akartok zárni a feladatból, akkor legyen megindokolva, hogy miért lehet megtenni
  - ha szerintetek valamilyen átalakítással a feladat lényegében ugyanaz maradt, akkor (amennyiben ez nem teljesen triviális) indokoljátok meg, hogy miért ekvivalens az eredetivel
- van, amit meg kell tanulni, nem elég érteni... ez van...
- van, amit meg kell érteni, nem elég megtanulni... ez (is) van...
- a vélemények többsége szerint kevésbé „intuitívak" a feladatok, mint a BSZ-ben voltak
  - a legtöbb feladathoz 1-2 elméleti pont/fogás szükséges, többnyire legfeljebb egy, de leginkább nulla feladatspecifikus „ötlet"
- sokszor nem mindegy, hogy O, Ω vagy Θ van egy-egy műveletnél
  - általában minket az O érdekel
  - ha Θ van (például legnagyobb elem kikeresése min-kupacból, keresés B-fában, stb.), akkor az arra utal, hogy az adott struktúrával nem is tudunk szorosabb felső becslést adni, hiszen tudunk egy ugyanilyen alsót
- gráfos feladatoknál (illetve máshol is) gyakori, hogy az ember (új) gráfot próbál konstruálni – tartsátok mindig szem előtt, hogy ez az új gráf éleinek és csúcsainak megadását is a lépésszámhoz adja, tehát innentől a lépésszámhoz a gráfmegadás is hozzá tartozik
  - éllistás esetben $O(|V(G)| + |E(G)|)$
  - szomszédossági mátrixos megadásnál: $O(|V(G)|^2)$
- NEM érdemes random algoritmusneveket egymás után hányva próbálkozni, hogy hátha jár rá pont (nem fog), ezzel csak rossz benyomást keltetek, és az nem biztos, hogy jó
- Abból, hogy mi a kérdés, valamelyest következik, hogy mi az eljárás ha a válasz igen, illetve ha nem (példa tipikusan 0 pontot ér, ha épp bizonyítani kellene)

	Igen	**Nem**
Lehetséges-e	példa (a feltételek bizonyításával)	bizonyítás (sokszor jó az indirekt)
Következik-e	bizonyítás	ellenpélda (a feltételek bizonyításával)

- Ha valamiről be kell látni, hogy nem lehetséges, akkor a próbálkozás sikertelensége nem bizonyítás (viszont nagyon hasonló, ám helyes módszer, ha indirekt módszerrel jutsz arra az állandó ellentmondásra, amibe a próbálgatással beleakadnál folyton)!
- Van-e olyan kupac/piros-fekete fa/akármi, hogy... vagy sorolj fel minden olyan...
  - ha a feladatban nem szerepel, hogy egy konkrét tanult algoritmussal hoztuk létre, akkor pusztán a tanult algoritmusok alkalmazásával készített példányok 0 vagy kevés pontot érnek
  - azok az esetek is érdekesek számunkra, amik az adott struktúra feltételeit kielégítik, de akár nem is állíthatók elő a tanult algoritmusokkal

2. Nagyságrendek

- a próbálgatás nem kifizetődő, ugyanis semmit nem jelent, ha kipróbáljátok 2-3 számra, mert a küszöböt választhatnánk annál nagyobbra (és felette simán lehet, hogy az ellenkezője teljesül), emiatt azt sejteti, hogy a nagyságrendbecslés legalapvetőbb dolgaival sem vagytok tisztában – szóval ezt NE csináljátok, mert ez 0 pontot ér! (sőt, leginkább -10-et)
- Ha nincs tippetek, hogy melyik függvény a nagyobb nagyságrendű, akkor jó szolgálatot tehet a L'Hospital-szabály (már ha megy még a deriválás) a sejtés megfogalmazásához – pl $a(n)$ és $b(n)$ esetén:
  - a/b-n alkalmazható (∞/∞ alakú) – a határérték végtelenben megegyezik a deriváltak hányadosának határértékével
  - ha az a/b kifejezés határértéke konstans, akkor jó eséllyel hasonló nagyságrendűek, ha végtelen, akkor esélyes, hogy a nő gyorsabban, ha nulla, akkor b
  - faktoriális esetén nem sokat segít :(– valószínűleg nem akartok a Γ-függvénnyel és deriváltjaival megismerkedni csak emiatt
- több függvény összehasonlításánál érdemes lehet előbb Θ szerint átalakítani, mert az alsó és felső becslésnél is használható, lásd:
  - $\frac{2012 \cdot n^5+n^3}{n^4} = \Theta(n)$, mert $\frac{2012 \cdot n^5+n^3}{n^4} \leq \frac{2012 \cdot n^5+n^5}{n^4} = 2013n$ és $\frac{2012 \cdot n^5+n^3}{n^4} \geq \frac{2012 \cdot n^5}{n^4} = 2012n$
    - ha $n \geq 2$
  - relevánsan könnyebben kezelhető tud lenni az n, mint a $\frac{2012 \cdot n^5+n^3}{n^4} = \Theta(n)$
  - amilyen (O, Ω, Θ) nagyságrendi becslést belátunk erre, az a Θ tulajdonságai miatt visszafelé is igaz
- $O(n+e) \Leftrightarrow O(\max\{n, e\})$

3. Dinamikus prorgramozás

- bár nem egyszerű, érdemes a DinProgba energiát fektetni
  - ilyen többnyire a ZH-n és a vizsgákon is van
  - sok másik, a félévben tanult algoritmus, lényegében DinProg
  - nagyon sok valós feladatnál az egyik leghatékonyabb algoritmus ilyen
- van több fontos dolog, aminek egy ilyen megoldásnál szerepelni kell

  - hogyan építkezünk a kisebb feladatokból
  - hogyan indulunk el (hogyan kapjuk meg az értéket azokra a feladatokra, amelyiknél nincs kisebb)
  - mikor vagyunk kész
  - hol lesz a megoldás
  - miért helyes a kapott megoldás (ha valami a legjobb megoldás, miért kapjuk meg azt, ha megkapunk valamit, az miért a legjobb)
  - lépésszám indoklása

- ha „gyanúsan alacsony" a lépésszámkorlát a feladat méretéhez képest, akkor gyakran dinamikus programozásos típuspéldáról van szó

- ha nem tudjátok, hogyan induljatok el, az alábbiakat érdemes véggigondolni:
  - az utolsó lépésben milyen információk kellenek kisebb feladatokról, amit fel tudunk használni
  - ez az információ kijön-e kisebb feladatok hasonló infóiból
  - ezt az infót számontartani, frissíteni mennyibe kerülne, ahogy haladunk előre
  - ez belefér-e a lépésszámkorlátba

4. Keresés és rendezés

- megkülönböztetünk összehasonlítás alapú és kulcsmanipulációs (láda-, radix) rendezéseket
  - összehasonlítás alapú rendezésnél csak a determinisztikus hasonlíthatóságra építünk
  - kulcsmanipulációs esetben van olyan plusz információnk, ami az adatok lehetséges értékeire vonatkozik (a lehetséges értékek száma n-hez képest „nem túl nagy")

- összehasonlítás alapú alapú rendezéssel n elem rendezésére $\Theta(n \log n)$ – tudunk ilyen korláttal rendezni (összefésülés, beszúrásos, stb.) O – ennél „gyorsabban" nem tudunk rendezni, ha nincs releváns plusz információnk Ω

- a gyorsrendezés csak átlagos lépésszámban gyors (átlagosan $O(n \log n)$), rossz esetben még lassabb, mint a versenytársak (általánosságban csak $O(n^2)$-et mondhatunk) – ha átlagos lépésszámról szól a feladat, akkor kellhet gyorsrendezés, ha felső korlátot kell biztosítani, akkor ez nem valószínű – **!** ez csak összehasonlítások számának tekintetében igaz így, a gyorsrendezés tényleg gyors, ha tömböt akarunk rendezni és nem csak az összehasonlítások számára optimalizálunk

- sok feladatban lehet szükség rendezésre, ha csak a feladat részeként adni kell egy $O(n \log n)$-es rendezőalgoritmust, akkor mindegy melyiket írjátok (ami ilyen), de írjatok egyet

- a legnagyobb/legkisebb elem kiválasztása egy n-elemű listából $\Theta(n)$

- ha van egy $5n$ elemű listánk, amit rendezni szeretnénk, de van benne $\Theta(n)$ elem, amelyeknek az egymás közötti viszonyáról nincs (SEMMILYEN) információnk, de ezzel ők is rendeződnek, akkor teljes rendezést sem tudjuk $n \log n$-es felső korlátnál lejjebb nyomni, hiszen akkor ezt a szakaszt is így rendeztük volna, ami nem lehetséges; ugyanez igaz a maximum-/minimumkeresésre, azaz egy ilyen helyzetben nem tudjuk ezeknél a műveletigény felső becslését n-es alá nyomni

- az összefésüléses rendezés egy speciális eszköz is lehet több (konstans sok) már rendezett lista „összefésülésére", az együttes rendezett lista előállítási költsége ilyenkor $O(n)$ – ha nem konstans sok van, hanem mondjuk n darab, akkor már NEM tudunk ilyen szoros felső becslést mondani.

5. BFS és DFS

- egy csomó mindenre jók, nagyon sok feladatnál kerülhetnek elő (de nyilván főleg gráfoknál)
- az $O(n+e)$ lépésszám árulkodó lehet, de ráadásul az itt tanult kb összes lépésszámkorlátba belefér ez is
- egy mélységi bejárásban van-e visszaél $\Leftrightarrow$ van-e a gráfban kör

6. Legrövidebb utak gráfokban

- a három algo biztosan alkalmazható, ha nincs negatív él (Dijkstra) illetve nincs negatív összsúlyú kör (B-F, Floyd)
  - DE körültekintéssel vannak más esetek is, amelyekben használhatóak – nem kell egyből kizárni őket, mert nem teljesül az ELÉGSÉGES feltétel
  - pl: ha nem érhetjük el a negatív éleket, mindegyik használható
  - pl2: ha csak egy negatív él van, akkor néhány Dijkstra (negatív él nélkül, annak végpontjaiba illetve végpontjaiból indítva) is tud legrövidebb utat számolni
  - ha lenne negatív kör a gráfban és el lehetne érni, akkor nem lenne értelme a feladat – bármilyen kicsi összköltség elérhető
- Miért is jobb a Bellman-Ford algo, mint a Floyd, ha csak egy csúcsból kell nézni, mikor mindkettő $O(n^3)$
  - A Bellman-Ford megáll, ha van két azonos sor egymás után – a Floydot végig kell csinálni
  - A Bellman-Ford esetében egy tömböt (n) tartunk számon – a Floyd esetében egy mátrixot $(n \times n)$
- élsúlyozás nélküli, vagy egyenletesen (pozitív súlyokkal) súlyozott gráfban $O(n+e)$-ben is tudunk legrövidebb utat keresni – BFS
- a csúcsfelbontás egy hasznos trükk: ha a gráfban minden csúcsnak konstans sok, meghatározott szerepe van, akkor egy-egy csúcs helyett felvehetjük ezt a konstans sok csúcsot, illetve konstansszoros élet – ezáltal továbbrais egy $O(n)$ csúcsú és $O(e)$ élű gráfunk van, de sokszor könnyebben kezelhet

7. Keresőfák

- Tudjátok az összesről, ami ide tartozik, hogy ide tartozik
- a (2,3)-fa lényegében a B_3 fa
- a bináris fa alapvetően rosszabb, mint a B vagy a P-F
  - bináris keresőfánál lehet $O(n)$-es ág
  - P-F fában $\log n$-es műveleti korlátokat garantálunk – ehhez kell az, hogy ne teljes bináris fát akarjunk, hanem engedjünk egy két ágat (a piros csúcsok által) megnyúlni
  - B-fa esetén hasonló a helyzet, csak itt széltében engedjük meg ugyanezt a megnyúlást – emellet a keresés lépésszámára alsó korlát is van, mert az elemek a levelekben vannak (tehát mindig pontosan ugyanannyi lépés és hasonlítás elérni egy elemet
- konstans sok keresőfából kilistázható az összes elem rendezve $O(n)$ lépésben – inorder listázással és összefésüléssel
- a B-fákat talán egy picit jobban szeretjük

  - minden levél ugyanolyan mélyen van, tehát kiszámítható hozzáférési idők vannak (mindig ugyanannyi), tehát jól költségbecsülhető
  - ha m-et jól választjuk, akkor a blokkméretre jól passzol, tehát nagy méret esetén tudjuk a háttértárról értelmesen csúcsonként olvasni
  - ilyesmik vannak adatbázisokban indexelésre

8. Kupac

- ne keverjük a keresőfákkal!!!
- a minimumon kívül nem sokmindent tudunk az elemekről
- a kupacépítés $O(n)$-ben megtehető, ha először bepakoljuk az elemeket, aztán berendezzük (lineáris idejű kupacépítés) - **!** a beszúrások egymásutánjából álló rendező eljárásra csak $O(n \log n)$-es felső korlátot tudunk adni (tehát ez NEM a kupacépítés algoritmusa, csak beszúrások)
- a k-adik legkisebb a felső k szint valamelyikén lehet – ott $2^k - 1$ elem van.
- a legnagyobb elem levél, amiből $\Theta(n)$ darab van (közelítőleg $n/2$)
- a középső elem szinte bárhol lehet a gyökéren kívül

9. Hash

- ha azt kell belátni, hogy az egyik esetben több az ütközés, mint a másikban, a beszúrandó elemektől függetlenül
  - mivel mindegy mit szúrunk be, ezért biztosan az egyikben való ütközésből következik, hogy az a másikban ütközni fog!
  - be kell látni, hogy ahol „kevesebb" az ütközés, ott bármely elem beszúrásakor fellépő k ütközés esetén legalább k-t ütközik a másik esetben is
  - a fenti tipikusan akkor szokott teljesülni, ha egy jól meghatározható csoport minden tagja páronként egyszer találkozik és ez igaz a másik esetben is ugyanerre a csoportra – de persze nem csak akkor lehet
- érdemes lehet a hash értékeket előre (táblázatosan) kiszámolni, hogy ha el is számoltok valamit, csak egyszer kelljen visszaellenőrizni
  - illetve, ha még ez sem ment meg a pontatlanságtól, akkor legalább látszik, hogy csak elszámolásról van szó

10. Minimális feszítőfák keresése

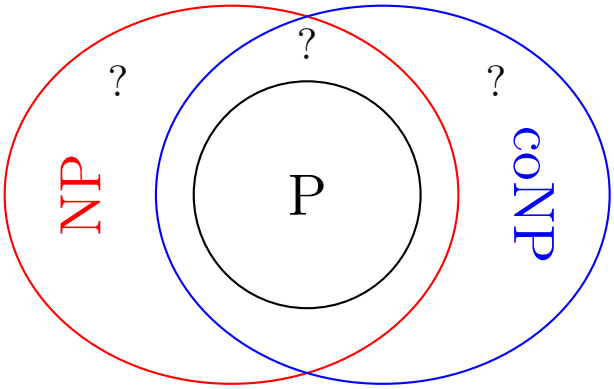
- sok ilyen feladatnál, ami nem az órán tanult algoritmusok alkalmazását jelenti, a piros szabály alkalmazása a kulcs.
- bármely gráfban takaros színezést kapunk, ha az olyan éleket, amelyek minden rajtuk átmenő körön a legnagyobb súlyúak, pirosra színezzük

11. Bonyolultságelmélet

- az első állításból a többi következik (néha a fordított irány is igaz, DE NEM MINDIG)
  - $A \prec B$
  - A polinom időben visszavezethető (Karp-redukálható) B-re
  - bonyolultságelméleti szempontból B legalább olyan bonyolult, mint A – azaz, ha $B \in P$, akkor $A \in P$ ha $A \notin P$, akkor $B \notin P$
  - létezik egy polinomiális algoritmus, amellyel bármely A-hoz tartozó konkrét bemenethez készíthetünk egy B-hez tartozó ekvivalens feladatot – azaz, ha $B \in P$, akkor $A \in P$ ha $A \notin P$, akkor $B \notin P$
  - A részproblémája a B feladatnak
  - B általánosítása az A feladatnak
- ha P-beliséget akarunk belátni, akkor egy polinomiális algoritmus kell (az eldöntési problémára vagy a komplementerére)
- ha NP-beliséget szeretnénk belátni, akkor elég adni egy jó tanút, és meg tudjuk mutatni, hogy hatékony
  - tanú: igen válasz esetén mindig létezik, nem válasz esetén soha
  - hatékony: ha megmutatja valaki, akkor polinom időben le tudjuk ellenőrizni
- ha A NP-teljességét akarjuk belátni, akkor két dologra van szükség
  - NP-beli (hatékony tanúsítvány)
  - egy NP-teljes X probléma polinom időben visszavezethető rá $X \prec A$
  - **!** a másik irányú Karp-redukció ebből a szempontból nem bizonyít semmit (ha X NP-teljes)
    - ((pl a 2-SZÍN probléma P-beli, pedig visszavezethető a 3-SZÍNre (új csúcs, mindegyikkel összekötve) – ezt gyorsan felejtsétek is el, ez a rossz irány))
    - ha X P-beli akkor természetesen más a helyzet, ezzel éppen belátjuk, hogy A is P-beli.
- ha egy problémáról azt akarjuk belátni, hogy coNP-beli (illetve -teljes), akkor sok esetben természetesebb (és mivel NP-beli példát többet ismertek, jó eséllyel sikeresebb) választás a komplementeréről belátni, hogy NP-beli (illetve -teljes)
- következtetés bizonyítására gyakran alkalmas az indirekt módszer
- **!** NE felejtsétek el, hogy olyan állítást jelenleg NEM tudunk tenni (vagy legalábbis nem a tárgy keretein belül szeretnénk ilyet bizonyíttatni bárkivel is), hogy valami $\notin P$

NP-teljes (emlékeztető) (matematikailag nem feltétlenül korrekt):

3-SZÍN	G: G egy irányítatlan egyszerű gráf, aminek csúcsai kiszínezhetők 3 színnel
MAXFTLEN	(G, k): G egy gráf, amiben van k darab független pont
MAXKLIKK	(G, k): G egy gráf, amiben van k pontú teljes részgráf
H	(G): G irányítatlan gráf, van benne Hamilton-kör
Hút	(G): G irányítatlan gráf, van benne Hamilton-út
stHút	(G, s, t): G irányítatlan gráf, van benne Hamilton-út s-ből t-be
3DH	$(A, B, C; F_1, \ldots, F_n)$: létezik olyan F-sorozat, hogy $A \cup B \cup C$ minden elemét pontosan egyszer fedik le
X3C	$(A; F_1, F_2, \ldots, F_n)$: A lefedhető pontosan egyszeresen egyes F_i halmazokat használva.
PARTÍCIÓ	$(s_1, s_2, \ldots, s_n)$: két részre bontható, hogy a két rész összege egyenlő
RH	$(s_1, s_2, \ldots, s_n; b)$: van részhalmaz, aminek az összege b
HÁTIZSÁK	$(s_1, \ldots, s_n; v_1, \ldots, v_n; b, k)$: van olyan indexhalmaz, hogy $\sum s \leq b$ és $\sum v \geq k$
RÉSZGRÁFIZO	(G_1, G_2): amelyre $\exists G' \leq G_2$, és $G' \simeq G_1$
EP	(x, A, b, c, d): van olyan x egész vektor, amelyre igaz $Ax \leq b$ és $cx \geq d$



A ?-es részekbe nem tudjuk, hogy tartozik-e valami is. Ha üres az NP\P vagy a coNP\P részek (mindkettő két kérdőjeles részt foglal magába) bármelyike, akkor a három halmaz egybeesik.

2012. december 13.

KS.