# Algel VII. gyakorlat

## Hurrá, ZH!

## 2013. március 25.

1. **Az $\mathcal{A}$ algoritmus egy $n \geq 3$ problémából 8 lépéssel 3 darab $n-1$ méretűt készít, és ezeket oldja meg rekurzívan, egyébként pedig 15 lépést használ. Tudjuk, hogy a $\mathcal{B}$ algoritmus lépésszáma ugyanerre a problémára $g(n) = \Omega(n^{\frac{2n}{\log n}} + 4 \log n!)$. Igaz-e, hogy az $\mathcal{A}$ algoritmus gyorsabb $\mathcal{B}$-nél?**

   Az $\mathcal{A}$ algoritmus lépésszámát jelöljük $f(n)$-nel. A feladat alapján így tudjuk felírni:

   $$f(n) = \begin{cases} 3f(n-1)+8 & n \geq 3 \\ 15 & n < 3 \end{cases}$$

   Az elegáns megoldás: ránézésre azt állítom, hogy $f(n) \leq 3^{n+2} - 4$. Indukcióval $f(1) = 15 \leq 3^{1+2}-4$ és $f(2) = 15 \leq 3^{2+2}-4$, tfh valamely $n$-re igaz, ekkor $n+1$ eset: $f(n+1) \leq 3f(n)+8 \leq 3 \cdot (3^{n+2}-4)+8 = 3^{n+3}-4$; pont, amit szerettünk volna. Tehát $f(n) \leq 3^{n+2}-4 = c \cdot 3^n - 4 = O(3^n)$. *Érdemes megfigyelni, hogy egy nagyon durva felső becslést adtunk, de az indukcióhoz így volt kényelmes. Felesleges lett volna „szép" korlátot adni, mert akkor az indukcióval sokkal többet kellett volna szórakozni.*
   Persze megcsinálhatjuk a kifejtős megoldást is (*itt sokkal pontosabb felső korlátot kapunk, de ez nem érdekel minket*):

   $$\begin{aligned} f(n) &\leq 3f(n-1)+8 \leq 3(3f(n-2)+8)+8 = 3^2 f(n-2) + 3 \cdot 8 + 8 \leq \\ &\leq \cdots \leq 3^{n-2} f(2) + 8(3^{n-3} + 3^{n-2} + \cdots + 3^0) = 15 \cdot 3^{n-2} + 8\frac{3^{n-2}+1}{3-1} = \\ &= 19 \cdot 3^{n-2} + 4 = O(3^n) \end{aligned}$$

   $g(n)$ kiszámolásához felhasználunk néhány logaritmusazonosságot, valamint azt, hogy $\log(n!) \approx n \log n$. Tehát

   $$g(n) = \Omega(n^{\frac{2n}{\log n}} + 4 \log n!) = \Omega(n^{n\frac{\log 4}{\log n}} + 4n \log n) = \Omega((n^{\log_n 4})^n + 4n \log n) = \Omega(4^n + 4n \log n) = \Omega(4^n)$$

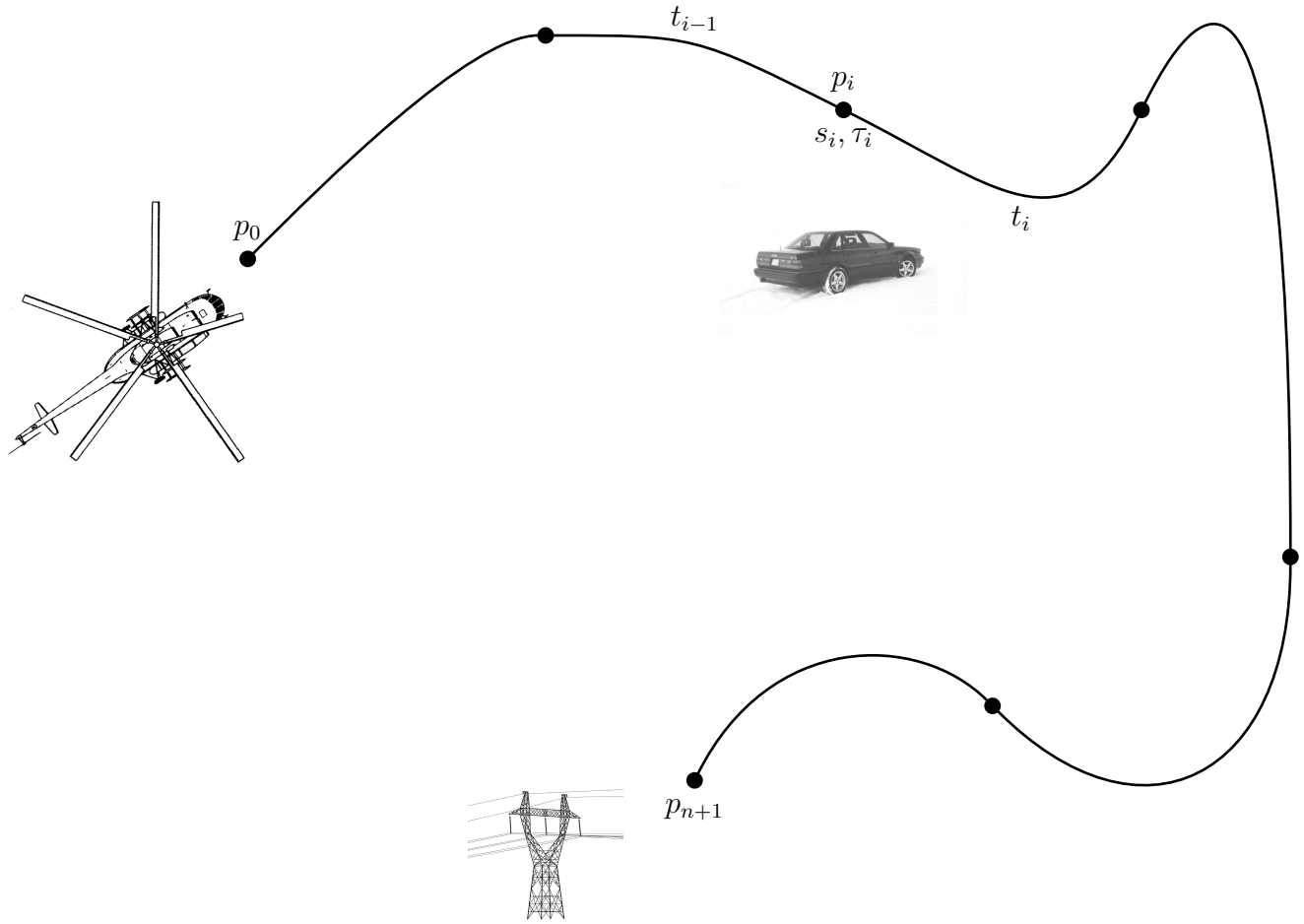
   Tehát $f(n)$-re a felső becslésünk kisebb, mint $g(n)$-re az alsó becslésünk, tehát az $\mathcal{A}$ algoritmus gyorsabb.

2. **A hóhelyzetben egy helikopterrel a $p_0$ kiindulási pontból el kell jutnunk a $p_{n+1}$ helyen található kidőlt villanyoszlopokat megvizsgálni, és minderre a következő hóvihar érkezéséig $T$ időnk van. Az előre meghatározott útvonalon a $p_i$ $(i = 1 \ldots n)$ waypointokat kell érintenünk, amelyek közül valahánynál elakadt autók találhatók. Helikopterünkön segélycsomagokat is szállítunk, egy ilyen csomag célbajuttatása a $p_i$ waypointnál (ha vannak ott autók) $s_i$ segítséget jelent, viszont $\tau_i$ időt ezzel elvesztünk. Tudjuk továbbá, hogy a $(p_i, p_{i+1})$ szakasz végigrepüléséhez $t_i$ időre van szükség. Adjunk algoritmust, ami $O(nT)$ lépésben meghatározza, hogy mely waypointoknál álló autósoknak adjunk segélycsomagot ahhoz, hogy a lehető legnagyobb segítséget nyújtsuk!**

   Figyeljük meg, hogy $\sum_{i=0}^{n} t_i$ időt mindenképp eltöltünk az utazással, tehát a segélycsomagok osztására $T' = T - \sum_{i=0}^{n} t_i$ időnk van. Készítsünk egy $S$ táblázatot, ahol $S[i,t]$ jelentése az, hogy amennyiben legfeljebb az első $i$ helyszínt vesszük figyelembe és $t$ időkorlátunk van, akkor mi a legnagyobb nyújtható segítség. Ez alapján a keresett érték $S[n, T']$ mezőben lesz. A táblázat kitöltése:

   $$S[i,t] = \max(S[i-1,t], S[i-1, t-\tau_i] + s_i),$$

vagyis vagy nem állunk meg $p_i$-ben, vagy megállunk, de akkor az előzőekre ennyivel kevesebb időnk van. A táblázat egy celláját konstans idő kitölteni, és $nT'$ cella van, tehát a lépésszám (a $t_i$ értékek összegzését is számolva) $O(n + nT') = O(nT)$.
*Egyébként az is egy helyes megoldás lenne, hogy ez a feladat megegyezik a hátizsákproblémával, ha a súlyokat $\tau_i$-re, az értékeket $s_i$-re, a súlykorlátot pedig $T'$-re állítjuk. Ekkor futtatva az ismert algoritmust $O(nT')$ lépésben kész vagyunk.*




3. **Egy kórházban $n$ darab új vesére váró házisurjancsunk van, valamint $v$ átültethető veséről tudunk. Egy surjancsvesének $k$ paramétere van, ezekből legalább $k-5$ egyezésére van szükség ahhoz, hogy az átültetés sikeres lehessen. Adjunk algoritmust, amely $O(nv(k + n + v))$ lépésben megmondja, hogy mely átültetések elvégzésével tudjuk a legtöbb surjancsot megmenteni!**

   Készítsünk egy páros gráfot úgy, hogy az $A$ ponthalmaz feleljen meg a veséknek, a $B$ pedig a surjancsoknak. Egy vese és egy surjancs pontosan akkor legyen egy éllel összekötve, ha az adott átültetés megengedett. Ebben a gráfban egy párosítás megfelel egy engedélyezett átültetéshalmaznak, valamint minden lehetséges átültetéshalmazhoz tartozik egy párosítás. Vagyis a gráfban egy maximális párosítás pont megfelel azoknak az átültetéseknek, amikkel a legtöbb surjancsot megmenthetjük. A gráfnak $|V| = n + v$ pontja van, és legfeljebb $|E| = nv$ éle. Annak eldöntése, hogy egy él behúzható-e, $k$ lépés (végignézzük a surjancs és a vese mind a $k$ paraméterét, és meghatározzuk, hogy mennyi egyezik). A párosítást a tanult alternálóutas algoritmussal megkereshetjük, aminek lépésszáma $O(|V||E|)$. Tehát a teljes lépésszám $O(nvk + nv(n + v)) = O(nv(k + n + v))$.

4. **Egy irányított gráf csúcshalmaza $\{A, B, C, D, E, F\}$, az élek és súlyaik pedig az alábbiak: $s(A, B) = 2$, $s(A, C) = 7$, $s(A, D) = 3$, $s(A, F) = 6$, $s(C, E) = 3$, $s(D, B) = -2$, $s(D, C) = -4$, $s(D, E) = -2$, $s(E, F) = 4$.**
**Futtassuk ezen a gráfon a Bellman-Ford algoritmust az $A$ csúcsból vett legrövidebb utak hosszának meghatározására!**

| $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|
| 0 | 2 | 7 | 3 | $\infty$ | 6 |
| 0 | 1 | $-1$ | 3 | 1 | 6 |
| 0 | 1 | $-1$ | 3 | 1 | 5 |
| 0 | 1 | $-1$ | 3 | 1 | 5 |

Mivel az utolsó két sor megegyezik, ezért megállhatunk itt. (*Remélem nem számoltam el semmit. Ha mégis, akkor elnézést.*)

5. **Adott $T_1, \ldots, T_n$, egyenként $p_1, \ldots p_n$ hosszú munkákat kell egymás után elvégeznünk úgy, hogy amennyiben $T_i$-t $C_i$ időpontban fejezzük be, akkor a**

$$\sum_{i=1}^{n} C_i$$

**érték minimális legyen. Adjunk $O(n \log n)$ lépésszámú algoritmust az optimális sorrend meghatározására!**

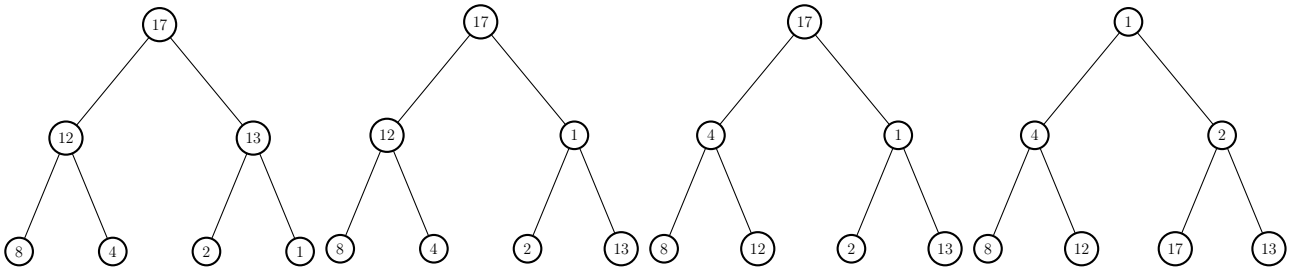
Némi rajzolgatás után megsejtjük, hogy a munkákat $p_i$ szerint növekvő sorrendbe rendezve optimális megoldást kapunk. Ezt be is bizonyítjuk. Tfh van egy olyan sorrendünk, ahol nem $p_i$ szerint növekvő sorrendben vannak a munkák, vagyis egymás után következik $p_j$ és $p_k$ úgy, hogy $p_j > p_k$. Ekkor ezt a kettőt megcserélve a célfüggvény értéke így változik (ha $C, C_i$ az eredeti célfüggvény értéke és az eredeti befejezési időpontok, $C', C_i'$ pedig a csere utáni):

$$C' = \sum_{i=1}^{n} C_i' = \sum_{i \neq \{j,k\}} C_i + (C_j + p_k) + (C_k - p_j) = \sum_{i=1}^{n} C_i + p_k - p_j < \sum_{i=1}^{n} C_i < C,$$

mert $C_j$ és $C_k$ befejezési időpontokon kívül mást nem változtottunk, továbbá $k$ munka pontosan $p_j$ idővel korábban, $j$ munka pedig $p_k$ idővel később fejeződik be. Az egyenlőtlenség a $p_j > p_k$ feltételezésből következik. Vagyis megmutattuk, hogy amennyiben a munkák nem $p_i$ szerint növekvő sorrendben vannak, akkor lehet javítani a célfüggvény értékén, vagyis úgy nem lehet optimális a megoldás. Tehát megtehetjük, hogy a munkákat $O(n \log n)$ lépésben rendezzük például kupacos rendezéssel.

6. **Építsünk kupacot a tanult lineáris idejű algoritmussal a következő tömbből: $[17, 12, 13, 8, 4, 2, 1]$!**

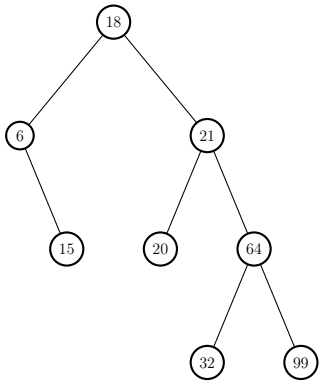
Felrajzuljuk a tömbhöz tartozó teljes bináris fát, majd jobbról balra, lentről felfele megcsináljuk a kupacolást.




7. **Egy bináris keresőfa posztorder bejárása a következő sorozatot adja: $15, 6, 20, 32, 99, 64, 21, 18$. Rajzoljuk fel a fát!**

A posztorder bejárásról tudjuk, hogy a gyökeret fogja utoljára kiírni, tehát a fa gyökere 18. A keresőfa tulajdonság miatt tudjuk, hogy a bal részfában a nála kisebb elemek szerepelnek, a

jobb oldaliban pedig a nagyobbak. Ezekre ugyanezt az érvelést rekurzívan használjuk. Tehát a fa:




8. **Létezik-e olyan piros-fekete fa, ahol a gyökér bal részfája pontosan 3-mal több elemet tárol, mint a jobb?**

   Mi az hogy, nagyon is! Lásd a példát.