Algel I. gyakorlat

Barátkozás a tárggyal és egymással

2012. február 6.

2. **Bizonyítsuk be, hogy**

   (a) $\log_2 f(n) = \Theta(\log_{100} f(n)) \quad (f(n) > 0)$
   $\log_2 f(n) = \frac{\log_{100} f(n)}{\log_{100}(2)} = c \cdot \log_{100} f(n)$. Innen kiadódik mindkettő.

   (b) $f(x) = a_k x^k + a_{k-1}x^{k-1} + \cdots + a_0 \quad (a_k \neq 0) \quad \Rightarrow \quad f(n) = \Theta(n^k)$
   O: $\sum a_i n^i \leq c \cdot n^k$ kell. $n^k (> 0)$ osztással $a_k + \sum a_i \frac{1}{n^{i-k}} \leq c$, amiből már könnyű látni (analízis).
   Ω: teljesen hasonlóan.

   (c) $2^{n+1} = O(2^n)$, de $2^{2n} \neq O(2^n)$
   $2^{n+1} = 2 \cdot 2^n$, ahonnan az első triviális.
   Tfh $2^{2n} = O(2^n)$. Ekkor $n > n_0$ esetén $2^{2n} \leq c \cdot 2^n$, ahonnan egy osztással $2^n \leq c$. Ez nyilvánvalóan nem lehetséges.

   (d) $\max(f(n), g(n)) = \Theta(f(n) + g(n)) \quad (f(n), g(n) > 0)$
   $\max(f(n), g(n)) \leq f(n) + g(n)$, ezért O triviálisan adódik. $\max(f(n), g(n)) \geq \frac{f(n)+g(n)}{2} = c \cdot (f(n) + g(n))$, ahonnan Ω adódik.

3. **[Vizsga: 2007. június 12.] Egy $\mathcal{A}$ algoritmusról azt tudjuk, hogy n hosszú bemeneteken a lépésszáma $O(n \log n)$. Lehetséges-e, hogy**

   (a) **van olyan x bemenet, amin a lépésszáma x^3?**
   Természetesen lehet ilyen, pl. ha $|x| < n_0$.

   (b) **minden x bemeneten legfeljebb $2007|x|$ lépést használ?**
   Lehet, hiszen $2007|x| = O(|x|) = O(|x| \log |x|)$. **(Szokás szerint $|x|$ az x szó hosszát jelöli.)**

4. **Jelöljük $T(n)$-nel egy algoritmus legnagyobb lehetséges lépésszámát az n méretű inputokon. Tudjuk, hogy $T(n) \leq 10$, ha $n \leq 5$ és $T(n) \leq T(n-1) + n/3$, ha $n > 5$. Ekkor mit tudunk mondani $T(n) = O(n)$, $T(n) = O(n^2)$ és $T(n) = O(n^3)$ egyenlőségek helyességéről?**
   A „favágó” megoldás:

$$\begin{aligned} T(n) &\leq T(n-1) + \frac{n}{3} \leq T(n-2) + \frac{n}{3} + \frac{n-1}{3} \\ &\leq \cdots \leq T(5) + \frac{1}{3}(n + (n-1) + \cdots + (n - (n-6))) \\ &= T(5) + \frac{1}{3}(\sum_{i=1}^{n} i - (1+2+3+4+5)) = T(5) + \frac{n(n+1)}{6} - 5 \\ &\leq 10 - 5 + \frac{n(n+1)}{6} = 5 + \frac{n(n+1)}{6} \end{aligned}$$

   Innen egyszerű bizonyítani $O(n^2)$-et, ahonnan $O(n^3)$ is rögtön adódik. $O(n)$-ről nem tudunk semmit, nincs kizárva, de nem is bizonyítható (Vigyázat! Ha $\leq$ helyett $=$ lenne, akkor már nem ez lenne a helyzet!).

   Az „elegáns” megoldás (Vigyázat! Sok veszély!):
   $O(n^2)$-et bizonyítjuk. $n_0 = 5$-re és $c = 1$-re igaz, mert $T(5) \leq 10 \leq 1 \cdot 5^2 = 25$. Indukcióval tfh valamilyen $n >= 5$-re igaz (azaz $T(n) \leq c \cdot n^2 = 1 \cdot n^2$), belátandó, hogy $n+1$-re is.

$$T(n+1) \leq T(n) + \frac{n+1}{3} \leq c \cdot n^2 + \frac{n+1}{3} = 1 \cdot n^2 + \frac{n+1}{3} \leq n^2 + n + 1 \leq n^2 + 2n + 1 = (n+1)^2 = c \cdot (n+1)^2,$$

pont amit bizonyítani akartunk. **Figyelem!** Itt a konstanst rögzíteni kell előre, nem bánhatunk vele olyan lazán, mint a normál bizonyításoknál! Végig **ugyanaz** a c szerepel (jelen esetben 1-re rögzítve, más feladatnál lehet más). Az egyszerűséget az adta, hogy felső becsléseket simán adhatunk (n helyett $2n$-t írunk, stb.). Ezzel viszont pl. $O(n)$-t akkor sem lehetne ilyen simán kizárni, ha $T(n)$ definícióban $\leq$ helyett $=$ lenne.

5. **[ZH: 2011. március 28.] Egy problémára két algoritmusunk van.**

   Az $\mathcal{A}$ algoritmus az $n \geq 2$ méretű problémából 10 lépéssel 2 db $n-1$ méretűt készít és ezeket oldja meg rekurzívan.

   A $\mathcal{B}$ az $n \geq 2$ problémából 3 lépéssel 4 db $n-1$ méretűt készít és ezeket oldja meg rekurzívan. Az $n = 1$ esetben mindkét eljárás 1 lépést használ.

   Melyik algoritmus lesz nagy n értékekre gyorsabb?
   $\mathcal{A}(n)$-et és $\mathcal{B}(n)$-et felírjuk először rekurzívan, majd átírjuk zárt alakra, ahonnan ki fog jönni (felhasználva, hogy $1 = n - (n-1)$, valamint $\sum_{i=0}^{k} c^i = \frac{c^{k+1}-1}{c-1}$).

$$\begin{aligned}
\mathcal{A}(n) &= 10 + 2\mathcal{A}(n-1) = 10 + 2(10 + 2\mathcal{A}(n-2)) = 10 + 10 \cdot 2 + 2^2\mathcal{A}(n-2) = \\
&= 10 + 10 \cdot 2 + 2^2(10 + 2\mathcal{A}(n-3)) = 10 + 10 \cdot 2 + 10 \cdot 2^2 + 2^3\mathcal{A}(n-3) = \\
&= \cdots = 10(2^0 + 2^1 + 2^2 + \cdots + 2^{n-2}) + 2^{n-1}\mathcal{A}(1) = 10\sum_{i=0}^{n-2} 2^i + 2^{n-1} = \\
&= 10(2^{n-1} - 1) + 2^{n-1} = 11 \cdot 2^{n-1} - 10 = O(2^n) \\
\mathcal{B}(n) &= 3 + 4\mathcal{B}(n-1) = 3 + 4(3 + 4\mathcal{B}(n-2)) = 3 + 3 \cdot 4 + 4^2\mathcal{B}(n-2) = \\
&= 3 + 3 \cdot 4 + 4^2(3 + 4\mathcal{B}(n-3)) = 3 + 3 \cdot 4 + 3 \cdot 4^2 + 4^3\mathcal{B}(n-3) = \\
&= \cdots = 3(4^0 + 4^1 + 4^2 + \cdots + 4^{n-2}) + 4^{n-1}\mathcal{B}(1) = 3\sum_{i=0}^{n-2} 4^i + 4^{n-1} = \\
&= 3(\frac{4^{n-1}}{3} - \frac{1}{3}) + 4^{n-1} = 2 \cdot 4^{n-1} - 1 = \Omega(4^n)
\end{aligned}$$

   Fentiekből látszik, hogy $\mathcal{A}$ a gyorsabb (fontos, hogy az egyikre O-t számoltunk, a másikra Ω-t; persze itt minkettő Θ, ezért nem kellett különbözőképpen számolnunk).

---

6. **Igaz-e, hogy**

   (a) **ha $f = O(g)$ és $g = O(h)$, akkor $f = O(h)$**
   $f \leq c_f \cdot g \leq c_f \cdot (c_g \cdot h) = c' \cdot h$ ($n > \max(n_f, n_g)$ esetén), tehát igaz.

   (b) **ha $f = \Omega(g)$ és $g = \Omega(h)$, akkor $f = \Omega(h)$**
   $f \geq c_f \cdot g \geq c_f \cdot (c_g \cdot h) = c' \cdot h$ ($n > \max(n_f, n_g)$ esetén), tehát igaz.

7. **Az alábbi függvényeket rendezzük olyan sorozatba, hogy ha f_i után közvetlenül f_j következik a sorban, akkor $f_i(n) = O(f_j(n))$ teljesüljön!**

$$f_1(n) = 8n^{2.5}, \quad f_2(n) = 5\sqrt{n} + 1000n, \quad f_3(n) = 2^{\log^2 n}, \quad f_4(n) = 2008n^2 \log n$$

   $f_2 = O(f_4) = O(f_1) = O(f_3)$. Ez három számolás, nem írom le. Amit bizonyítás nélkül fel lehet hozzá használni: $\log n \leq c \cdot \sqrt{n}$.

8. **Az $\mathcal{A}$ algoritmusról azt tudjuk, hogy n hosszú bemeneteken a lépésszáma $O(n^2)$. Lehetséges-e, hogy**

   (a) **$\forall n$ hosszú bemeneten $O(n)$ lépést használ?**
   Természetesen, hiszen a definíciónak nem mond ellent.

   (b) **$\exists x$, hogy az x bemeneten az algoritmus lépésszáma $10|x|^2 \log |x| - 800$ (ahol $|x|$ az x bemenet hosszát jelöli)?**
   Természetesen, hiszen ha $|x| < n_0$, akkor bármi lehet.

9. **[Vizsga: 2007. június 19.] Az alábbi függvényeket rendezze olyan sorozatba, hogy ha f_i után közvetlenül f_j következik a sorban, akkor $f_i(n) = O(f_j(n))$ teljesüljön!**

$$f_1(n) = 2^{100n} - 2^{50n}, \quad f_2(n) = 2007n^3, \quad f_3(n) = 3^{3n}$$

Megsejtjük, hogy $f_2(n) = O(f_3(n)) = O(f_1(n))$. Most bebizonyítjuk, definíció szerint.

$f_2(n) = O(f_3(n))$: $\exists c > 0, n_0 > 0$, hogy $\forall n > n_0$ $2007n^3 \leq c(3^{3n})$. Ezt átalakítva, konstanst hozzávéve c-hez $n^3 \leq c \cdot 27^n$, aminek az igazságát már nem kell bizonyítani, triviálisnak tekinthető.

$f_3(n) = O(f_1(n))$: $\exists c > 0, n_0 > 0$, hogy $\forall n > n_0$ $3^{3n} \leq c(2^{100n} - 2^{50n})$. Írhatjuk, hogy $3^{3n} \leq c2^{50n}(2^{50n} - 1)$, és mondjuk logaritmust vonva $n \log 27 \leq c' + n \log 2^{50} + \log(2^{50n} - 1) = c' + 50n + \log(2^{50n} - 1)$. A kifejtetlen tag biztos pozitív, így ez biztosan igaz bármely $n > 0$ és $c \geq 0$ esetén.

Számolás nélkül, nem definíció szerint nem lehet max. pontot kapni!

10. **[Vizsga: 2007. június 5.] Jelölje egy algoritmus maximális lépésszámát az n hosszú bemeneteken $L(n)$. Azt tudjuk, hogy minden $n = 2k > 4$ páros számra $L(2k) \leq L(2k-2) + 1$ teljesül, és hogy $L(4) = 10$. Következik-e ebből, hogy az algoritmus lépésszáma $O(n)$?**
Nem következik, hiszen a páratlan n-ekről semmit sem tudunk, ott lehet akár 2^n is.

11. **Jelöljük $T(n)$-nel egy algoritmus legnagyobb lehetséges lépésszámát az n méretű inputokon. Tudjuk, hogy $T(n) \leq 10$, ha $n \leq 5$ és $T(n) \leq T(n-1) + n/3$, ha $n > 5$. Ekkor mit tudunk mondani $T(n) = O(n)$, $T(n) = O(n^2)$ és $T(n) = O(n^3)$ egyenlőségek helyességéről? (Most szépen kiszámolva!)**
Lásd feljebb (a „favágó" verzió).

12. **[PZH: 2011. április 22.] Egy problémára két algoritmusunk van.**

Az $\mathcal{A}$ algoritmus az $n \geq 2$ méretű problémából 5 lépéssel 2 db legfeljebb $n/2$ méretűt készít és ezeket oldja meg rekurzívan.

A $\mathcal{B}$ algoritmusról azt tudjuk, hogy lépésszáma az n méretű problémákon $O(n^2)$.

Ha ennyiből lehetséges, határozza meg, melyik algoritmus lesz nagy n értékekre gyorsabb! Ha ennyi információból még nem következik, hogy $\mathcal{A}$ vagy $\mathcal{B}$ lesz a gyorsabb, akkor indokolja meg, miért nem!
$\mathcal{A}$-ra felső becslésünk van, így kiszámolva $O(\dots)$ lenne. Az O felső becslés, így mindkét algoritmus lépésszámára csak felső becslésünk van, vagyis nem tudjuk eldönteni, melyik a gyorsabb (lehet mindkettő akár pl. konstans lépésszámú is).

Megjegyzés: itt nem volt rá szükség, de ha a rekurzív képletet fel akarnánk írni és ki akarnánk számolni, akkor (feltéve, hogy $\mathcal{A}(1) = c$, valamint néhány egészrész jelet nagyvonalúan elhanyagolva) azt így lehet (felhasználva, hogy $1 = \frac{n}{2^{\log n}}$):

$$\begin{aligned}
\mathcal{A}(n) &\leq 5 + 2\mathcal{A}(\frac{n}{2}) \leq 5 + 2(5 + 2\mathcal{A}(\frac{n}{4})) = 5 + 5 \cdot 2 + 2^2\mathcal{A}(\frac{n}{4}) \leq \\
&\leq 5 + 5 \cdot 2 + 2^2(5 + 2\mathcal{A}(\frac{n}{8})) = 5 + 5 \cdot 2 + 5 \cdot 2^2 + 2^3\mathcal{A}(\frac{n}{16}) \leq \\
&\leq \cdots \leq 5(2^0 + 2^1 + 2^2 + \cdots + 2^{\log n}) + 2^{\log n}\mathcal{A}(1) = 5 \sum_{i=0}^{\log n - 1} 2^i + c \cdot n = \\
&= 5(2^{\log n} - 1) + c \cdot n = (5 + c)n - 5 = O(n)
\end{aligned}$$

13. **Tudjuk, hogy $f(x) = O(h(x))$ és $g(x) = O(h(x))$. Igaz-e, hogy**

(a) **ha** $h(x) = 3x$**, akkor** $f(g(x)) = O(h(x))$
$x > \max(n_f, n_g)$ esetén $f(g(x)) \leq c_f \cdot 3g(x) \leq c_f \cdot 3 \cdot c_g \cdot 3x \leq c'3x = c'h(x)$.

(b) $f(g(x)) = O(h(x))\ \forall h$ **függvényre**
Ellenpélda: $f(x) = x^2, g(x) = x^3, h(x) = x^4$, ebből $f(g(x)) = (x^3)^2 = x^6$. Látszik, hogy a feltétel teljesül, viszont $x^6 \neq O(x^4)$.