2011. zh, pzh nagyságrendes feladatok

1. **[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?

   Megoldás: $\mathcal{A}(n)$-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).

2. **[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!

   Megoldás: $\mathcal{A}$-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 $1 = \frac{n}{2^{\log n}}$):

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