# Algoritmusok bonyolultsága

1. előadás

http://www.ms.sapientia.ro/~kasa/komplex.htm

# Könyvészet


- Sara Baase: *Computer Algorithms. Intoduction to Design and Analysis*, Addison-Wesley Publ. Co. 1978, 1983, 1988. http://www2.latech.edu/~choi/Bens/Teaching/Csc520/
- Michael R. Garey, David S.Johnson: *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co. 1979.
- Michael Sipser: *Introduction to the Theory of Computation*, PWS Publ. Co. 1997. http://www.cin.ufpe.br/~jjss/Introcuction%20to%20Theory%20of%20computation%20by%20Micheal%20Sipser%20Ist%20Ed..pdf
- C. H. Papadimitriou: *Számítások bonyolultsága*, Novadat Kiadó, 1999.
- Lovász László: *Algoritmusok bonyolultsága*, Egyetemi jegyzet, 2013. http://etananyag.ttk.elte.hu/FiLeS/downloads/16_LOVASZ_Algor_bonyol.pdf
- T. H. Cormen, C. E. Leiserson, R. R. Rivest, X. Stein: *Új algoritmusok*, Scolar Kiadó, Budapest, 2003.

# A $O(f(n))$ jelölés

$f, g : \mathbf{N} \to \mathbf{R}^+$

Ha létezik egy $C > 0$ valós szám és egy $n_0$ természetes szám úgy, hogy

$$f(n) \leq Cg(n) \quad \text{ha} \quad n \geq n_0,$$

akkor az $f$ függvény *rendje* kisebb vagy egyenlő, mint a $g$ rendje, és ezt így írjuk: $f(n) = O(g(n))$.

Ha $f(n) = O(g(n))$ és $g(n) = O(f(n))$, akkor az $f$ és $g$ függvények azonos rendűek, és ennek jelölése: $f(n) = \Theta(g(n))$.

Ha $f(n) = O(g(n))$ akkor $g(n) = \Omega(f(n))$.

# Példák

$n^2 = O(n^3)$, de $n^3 \neq O(n^2)$

$n + 4 = O(n^2 + n + 3)$, $n + 4 = O(n + 3)$, $n + 4 = \Theta(n + 3)$

$n = O(n^2 + n + 3)$, $n^2 = O(n^2 + n - 3)$ és $n^2 = \Theta(n^2 + n + 3)$

$n^2 = \Omega(n - 3)$

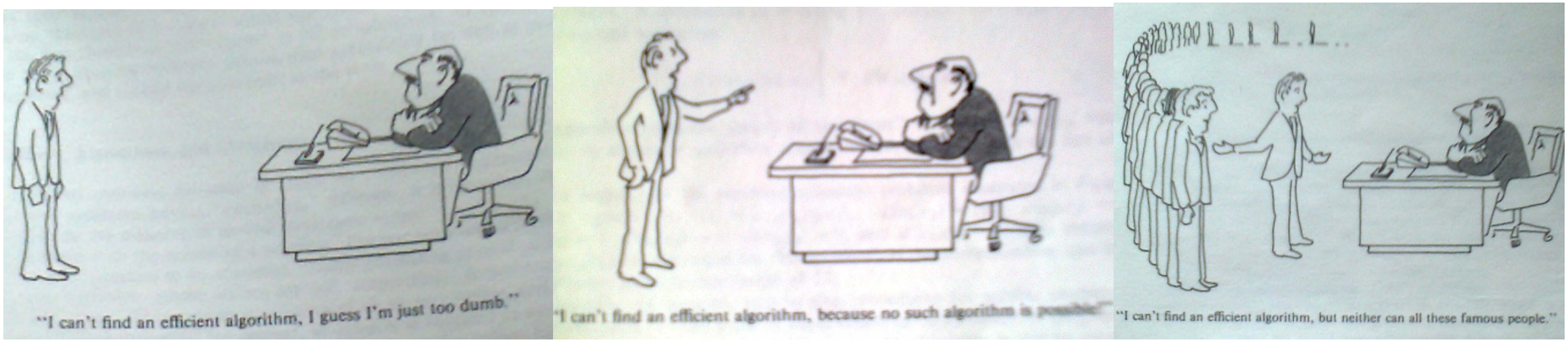
$\log n = O(n)$

$O(2^n)$ nem sokat mond, nem érdemes használni

$\Omega(2^n)$ biztos, hogy nem polinomiális

# Bonyolultság mérése

miért fontos?

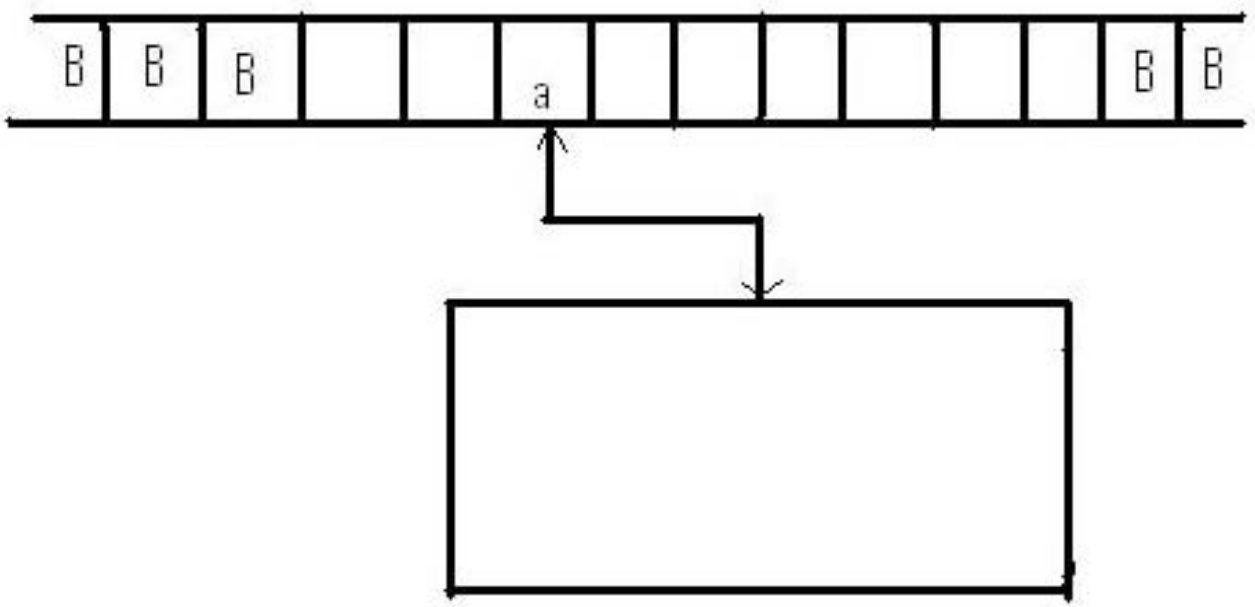



*Garey–Johnson könyvéből*

bonyolultság:

- időbonyolultság
- tárbonyolultság

# Bonyolultság mérése




Turing-gép

Alan Turing
(1912–1954)

# Bonyolultság mérése – szekvenciális keresés

*n*-elemű lista esetében a szekvenciális keresés bonyolultsága $O(n)$

Szekvenciális_keresés($A, x$)

1. $j := hossz[A]$
2. **while** $(j > 0)$ és $(A_j \neq x)$
3. **do** $j := j - 1$
4. **return** $j$

# Bonyolultság mérése – bináris keresés

bináris keresés

$A_1 \leq A_2 \leq \ldots \leq A_n, \quad x$

Bináris_keresés$(A, x)$

1. $i := 1$, $j := hossz[A]$
2. **while** $i \leq j$
3. **do** $k := \left\lfloor \frac{i+j}{2} \right\rfloor$
4. **if** $x = A_k$
5. **then return** $k$
6. **if** $x < A_k$
7. **then** $j := k - 1$
8. **else** $i := k + 1$
9. **return** 0

$$\begin{cases} W(n) = 1 + W\left(\left\lfloor \frac{n}{2} \right\rfloor\right), \text{ha } n > 1, \\ W(1) = 1. \end{cases}$$

megoldása: $W(n) = \lfloor \log n \rfloor + 1$, ha $n \geq 1$.

Itt a log kettes alapú logaritmust jelent.

# Bonyolultság mérése – mátrixszorzat

Mátrix_szorzat($A, B$)

1. **for** $i := 1$ **to** $sorszám[A]$
2. **do for** $j := 1$ **to** $oszlopszám[B]$
3. **do** $C_{ij} := 0$
4. **for** $k := 1$ **to** $oszlopszám[A]$
5. **do** $C_{ij} := C_{ij} + A_{ik} B_{kj}$
6. **return** $C$

Ha $sorszám[A] = oszlopszám[A] = oszlopszám[B] = n$, akkor $W(n) = n^3$ (alapművelet a szorzás).
Összeadások száma: $n^2$.

Lehet-e kevesebb szorzással?
Strassen, 1969

# Bonyolultság mérése

- polinomiális feladatok

$\mathcal{P}$ osztály: polinomiális feladatok (problémák) osztálya
létezik polinomiális algoritmus a feladat megoldására

$\mathcal{NP}$ osztály: nemdeterminisztikusan polinomiális feladatok osztálya

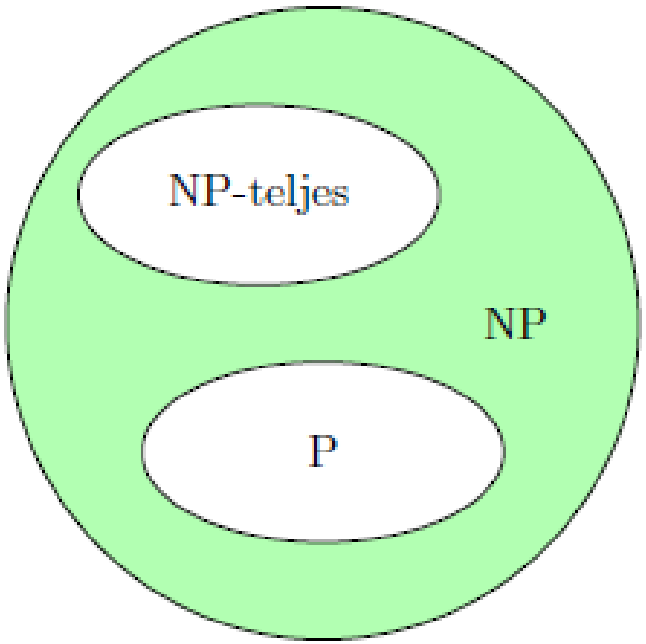
$\mathcal{P} \subseteq \mathcal{NP}$

$\mathcal{NP}$-teljes feladatok: lehető legnehezebbek

Ha egyszer egy $\mathcal{NP}$-teljes feladatról kiderül, hogy polinomiális, akkor $\mathcal{P} = \mathcal{NP}$.

# Bonyolultság mérése

- polinomiális feladat ($\mathcal{P}$)
- nemdeterminisztikusan polinomiális feladat ($\mathcal{NP}$)




$\mathcal{P} = \mathcal{NP}$ vagy $\mathcal{P} \neq \mathcal{NP}$?

LP lineáris programozási feladat
Dantzig 1945, Kacsiján (Khachiyan) 1979, Karmarkar 1984

# Bonyolultság mérése

$\mathcal{NP}$-teljes feladatok:

- logikai formulák kielégíthetősége
- hátizsák probléma
- csomagolási probléma (bin packing)
- Hamilton-út keresése
- lefedőszó probléma

## Példák lineáris programozásra

https://www.uni-miskolc.hu/~matha/linearis_programozas.pdf