CURSUL 1

Ordinea de zi:

1. Administrative stuff
2. Algorithm. Complexity
3. Complexity of Divide & Conquer Algorithms. + Master Theorem

It IS a course on Fundamental Algorithms, learn (to):

- evaluate algorithm performance
- compare performance on different algorithms
- design efficient and optimal algorithms
- identify a solution to a given problem
- specific efficient algorithms on fundamental problems

nota: vreia heap-ul pentru cursul 2

Ce vrem în algoritmi?

- corectitudinea — se demonstrează formal
  MUST-HAVE
  incorectitudinea se exemplifică (cauți contraexemplu pe dimens. mici)
- eficienta
  VERY-GOOD-TO-HAVE
  NICE-TO-HAVE

Cu ajutorul complexității exprimăm (estimăm) numărul de resurse necesare

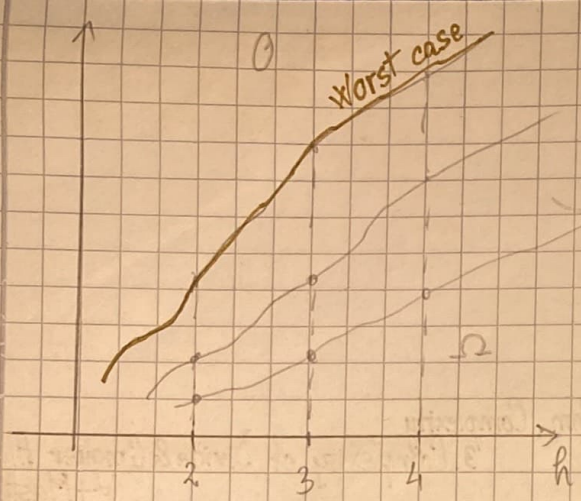
RESURSE: → timp !
→ memorie
→ altele: operații

NU e o idee bună să măsurăm the actual time, fiindcă depinde mult de specificațiile tehnice (hardware) a calculatorului folosit.

Cazuri de luat în considerare: best

average Meh

WORST Uh-oh



Ne interesează comportamentul la infinit, asimptotic. Îi bai pt. input mare?
Încercăm să limităm / mărginim superior.

În teorie: studiu caz rău
În practică: studiu caz mediu

notă: studiază pt. Seminarul 1, sortarea prin inserare

Problemă de complexitate: care este cea mai MICĂ cantitate de resurse necesare ORICĂRUI posibil (<ne>cunoscut) algoritm care ar putea rezolve problema dată?

Eficiența:

- relativă: comparații între algoritmi (de cele mai multe ori) în cazul mediu statistic
- absolută → algoritmul este/nu este optim?

Ce înseamnă complexitatea dpdv practic?

O - expresses the asymptotic upper bound of a function
Ω - ——— lower bound
Comparăm O cu Ω

Un algoritm este OPTIM dacă timpul de rulare (running time) al algoritmului-soluție în cazul worst este egal cu cel în cazul bun, iar adițional algoritmul folosește memorie adițională constantă: $O = \Omega$

Reguli de calcul

1. const se ignoră
2. $O(... + ...) = O(...) + O(...)$
3. $O(... * ...) = O(...) \times O(...)$

Values of $\Omega()$ for some problems:

- searching : $\Omega(\log n)$
- selection : $\Omega(n)$
- sorting : $\Omega(n \log n)$

< Remarks >

o) $O(1)$ = constant time

o) Ω caracterizează problema, iar O algoritmul

Complexity vs Computational Power

· avem 2 mașini, una cu o viteză superioară (let it be M_2, ailaltă M_1)

· deci în același timp (unitate de timp), cele 2 mașini vor avea dimensiune diferită.

dimens_M2 > dimens_M1, adică în timpul T, mașina M_2 poate face mai multe operații.

e.g. ai un vector, 2 mașini. M_2 are viteză mai mare (număr de instrucțiuni per sec.) va fi capabil să sorteze un vector mai mare decât M_1 (deci bigger dimension)

Complexity of Divide & Conquer Algorithms. Master Theorem

```
DIVIDE_ET_IMPERA (n, I, O)
1  if n ≤ n0
2      DIRECT_SOLUTION (n, I, O)
3  else
4      DIVIDE (n, I1, I2, ..., Ia)
5      DIVIDE_ET_IMPERA (n/b, I1, O1)
6      ... // un număr de „a" apeluri recursive în total
7      DIVIDE_ET_IMPERA (n/b, Ia, Oa)
8      COMBINE (O1, O2, ..., Oa, O)
```

$$T(n) = \begin{cases} T_0 & \text{dacă } n \le n_0 \\ a \cdot T\left(\frac{n}{b}\right) + \underbrace{n^c}_{f(n)} \end{cases}$$

- a — numărul de apeluri recursive
- b — factorul de divizare
- c — gradul polinomială care descrie