1r 22/09

Algorisme (teoria)

Amalia Duch

duch@cs.upc.edu

I Anàlisi d'Algorismes

Eficiència d'un algorisme: quantitat de recursos (temps, espai) que fan servir (de l'ordinador)

L'anàlisi de l'eficiència pot ser experimental, però depen: tipus màquina, llenguatge prog. ---

→ per això fem un anàlisi matemàtic, que ens permet preveure quin serà el comportament d'un determinat algorisme sense haver-lo d'implementar

- Definim:

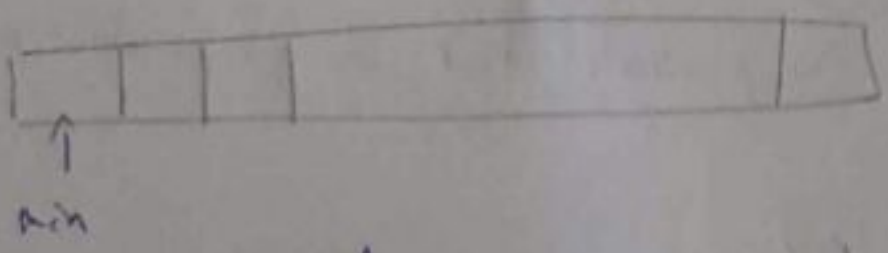
- cost en temps: nombre d'operacions elementals que es requereixen

" multiplicacions/divisions/asignacions; increments; pas de paràmetres, etc ---. dels tipus bàsics

no bàsics: vector, string bàsics: booleans, caràcters, int, double

- cost en espai: quantitat de memòria que es faria servir

Ordenació per selecció



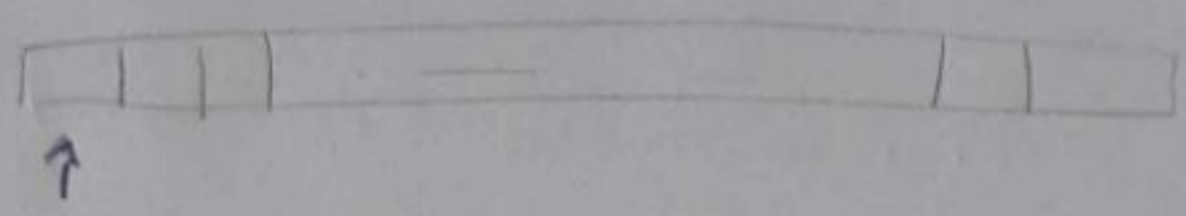
{ programar pel proper dia

busques el min de la resta i canvies les posicions

comparacions: $\frac{n(n-1)}{2} \simeq n^2 \simeq O_{n^2}$

intercanvis: $n-1$

Ordenació per inserció



Suposem que el 1r ja està ordenat → agafem el segon element i el comparem amb l'altre
(si és més gran el deixem a la dreta, sino el canviem a la dreta i seguim comparant...)

comparacions: —— $n-1$ si el vector està ordenat → cas millor
$\simeq n$ → cas mitjà $\frac{(n-1)(n-2)}{2}$
$\frac{n(n-1)}{2} \simeq n^2$ (si el vector està ordenat decreixentment) → cas pitjor

intercanvis:

Def: El cost (a temps, espai, # operacions entrada/sortida) d'un algorisme és una funció $T_n \quad A_n \longrightarrow \mathbb{R}^+$
$\alpha \longrightarrow T_n(\alpha)$

on A_n és el conjunt de totes les possibles entrades de mida n (de la mateixa mida)

- Cost en cas millor: $T_{millor}(n) = \min\{T_n, \alpha \in A_n\}$
- Cost en cas pitjor $T_{pitjor}(n) = \max\{T_n, \alpha \in A_n\}$
- Cost en cas mitjà: $T_{mitjà}(n) = \sum_{\alpha \in B_n} P_n(\alpha) \cdot T(\alpha)$
  ↳ probabilitat de que es doni α

Prop: $T_{millor}(n) \leq T_{mitjà}(n) \leq T_{pitjor}(n)$

Notació asintòtica: O, Ω, Θ

lletra O gran — omega — teta

Def: Sigui $f: \mathbb{N} \longrightarrow \mathbb{R}^+$

$O(f) = \{ g: \mathbb{N} \longrightarrow \mathbb{R}^+ \mid \exists c>0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0 \; f(n) \geq c \cdot g(n) \}$

↳ "el creixement de la f és més gran que el de g"

exemples: $O(n) = \{ 53, \frac{1}{n}, \log(n), \sqrt{n}, 1024n+230, \ldots \}$

i qualsevol constant

$O(n^2) = \{ n\sqrt{n}, n\log(n), \ldots \}$; $O(n) \subset O(n^2)$

El cost dels algorismes d'inserció i selecció és $O(n^2)$

Def: $\Omega(f) = \{ g: \mathbb{N} \to \mathbb{R}^+ \mid \exists c>0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0 \; f(n) \leq c \cdot g(n) \}$

$\Theta(f) = O(f) \cap \Omega(f)$

exemple: El cost de l'algorisme d'inserció és $\Omega(n)$ (no n^2)

selecció és $\Omega(n^2)$ (també (n))

Prop: Si $\lim_{n \to \infty} \frac{g(n)}{f(n)} < \infty$ llavors $g \in O(f) \xrightarrow[\text{notació}]{\text{abús de}} g = O(f)$

Si $\lim_{n \to \infty} \frac{g(n)}{f(n)} > 0$ llavors $g = \Omega(f)$

Si $\lim_{n \to \infty} 0 < \frac{g(n)}{f(n)} < \infty$ llavors $g = \Theta(f)$ i $f = \Theta(g)$

ex: El cost de l'algorisme de selecció és $\Theta(n^2)$

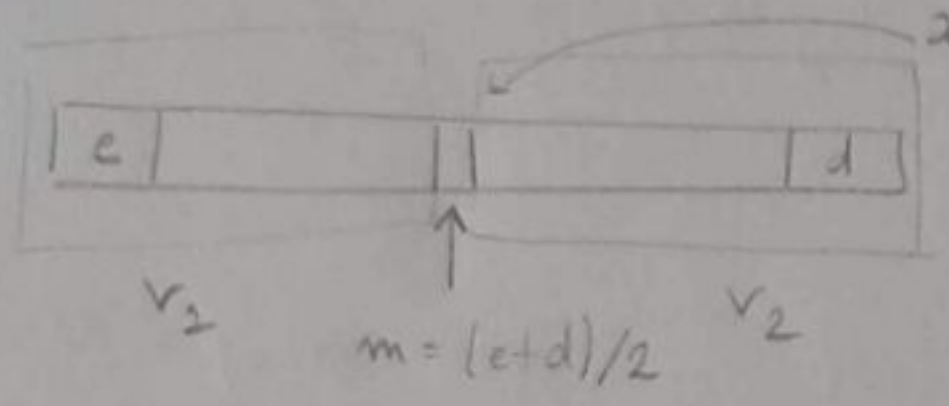
inserció en el cas millor és $\Theta(n)$

i en el cas pitjor és $\Theta(n^2)$

Cerca dicotòmica

Input: Vector ordenat v; element x

Output: Determinar si x és o no a v



cas millor: $\Theta(1)$ <-> "és constant"

cas pitjor $\Theta(\log(n))$

si $x == v[m]$ → jo estem

sino si $x < v[m]$

return cerca x a v_1 → fem la meitat amb la primera meitat → recursiu

si $x > v[m]$ (o altre)

↳ fem la meitat amb la segona meitat

return cerca x a v_2