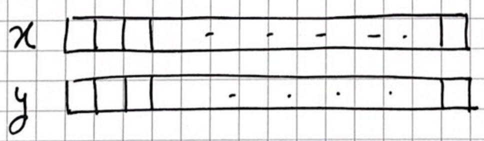
DiV / DaC

Multiplication | Karatsuba.

Carl Friederich Gauss (1777-1855) devise how to multiply complex numbers with only 3 multiplications (instead of 4) by noticing that:

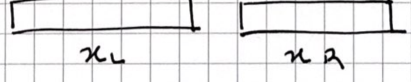
$$(a+bi)(c+di) = ac - bd + \underbrace{(bc+ad)}_{\substack{\shortparallel \\ \left((a+b)(c+d) - ac - bd\right)}} i$$

↳ Seems modest but when applied recursively becomes very significant.

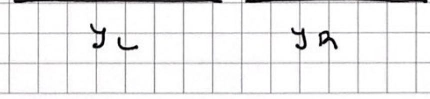


n bit numbers.
integers.

n power of 2

$x =$ 

$= x_L * 2^{n/2} + x_R$

$y =$ 

$= y_L * 2^{n/2} + y_R$

$$xy = (2^{n/2} x_L + x_R)(2^{n/2} y_L + y_R)$$
$$= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \Rightarrow T(n) = \Theta(n^2).$$

with Gauss trick:

Since $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n) \Rightarrow T(n) = \Theta(n^{1.59}).$$

1

everis

an NTT DATA Company

Strassen.

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = Y \qquad n \times n$$

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$$T(n^2) = 8T\left(\frac{n^2}{4}\right) + \cancel{\theta(n^2)}\ \theta(n^2)$$

$$T(N) = 8T\left(\frac{N}{2}\right) + \theta(N^2) \Rightarrow T(N) = \theta(N^3)$$

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E$

$P_5 = (A + D)(E + H)$

$P_6 = (B - D)(G + H)$

$P_7 = (A - C)(E + F)$

$$T(N) = 7T\left(\frac{N}{2}\right) + \theta(N)$$

$$\Rightarrow T(N) = \theta(N^{\log_2(7)}) = \theta(N^{2.81})$$

UNIVERSITAT POLITÈCNICA DE CATALUNYA

1

- [ ] E.T.S. d'Enginyeria de Telecomunicació de Barcelona
- [ ] E.T.S. d'Enginyers de Camins, Canals i Ports de Barcelona
- [ ] Facultat d'Informàtica de Barcelona

Titulació: Merge-Sort

Assignatura

Cognoms

Nom

DNI

Pàgina ______ de ______

Merge-Sort.

* Alg.

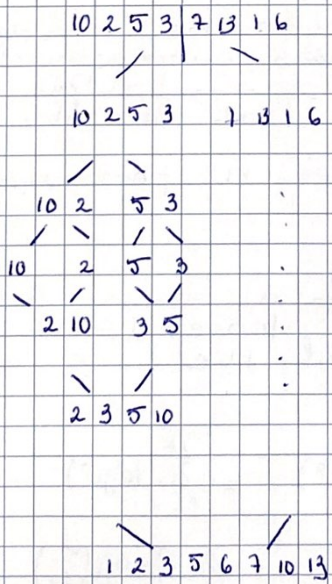
Input: vector de n elements $V[0, \ldots, n-1]$

Output: V amb els elements ordenats creixentment

```
mergesort(v[0,...,n-1])
  if n>1
      ret. merge(mergesort(v[0,...,n/2]), mergesort(v[n/2+1,...,n-1]));
  else
      ret. v;
```

* Cost: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \Rightarrow T(n) = \Theta(n \log n)$.

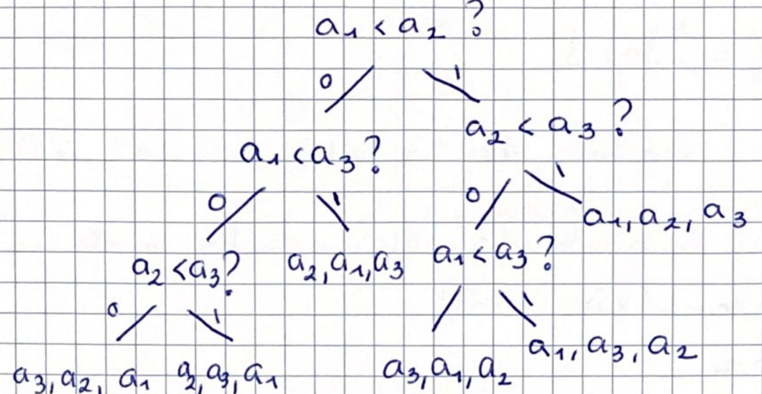
* Exemple:



An nlogn lower bound for sorting.

- Sorting algorithms based in comparisons can be depicted as trees.

  The one in the following picture sorts an array of three elements a_1, a_2, a_3



- The depth of the tree - the number of comparisons on the longest path from root to leaf, in this case 3, is exactly the worst-time complexity of the algorithm.

- This way to looking at sorting algorithms is useful because it allows one to argue that mergesort is optimal, in the sense that $\Omega(n\log n)$ comparisons are necessary for sorting n elements.

- Here is the argument: Consider such tree. Each of its leaves corresponds to a permutation. In fact, each permutation must appear as the label of a leaf for the alg. to be correct.
  Since there are $n!$ permutations, we need at least $n!$ leaves.

- Recall now that a binary tree of depth d has at most 2^d leaves (proof by induction).
  So the depth of the tree must be at least $\log(n!) = \Omega(n\log n)$.

$$\log(n!) = \sum_{i=1}^{n} \log(i) = \sum_{i=1}^{n/2} \log(i) + \sum_{i=n/2}^{n} \log(i)$$

$$\geq \frac{n}{2}\log\left(\frac{n}{2}\right) = \Omega(n\log n)$$

+

UNIVERSITAT POLITÈCNICA DE CATALUNYA

1

Titulació FIBONACCI

Assignatura

Cognoms

Nom

DNI

E.T.S. d'Enginyeria de Telecomunicació de Barcelona

E.T.S. d'Enginyers de Camins, Canals i Ports de Barcelona

Facultat d'Informàtica de Barcelona

Pàgina ______ de ______

FIBONACCI

Leonardo Fibonacci (of Pisa)
S XIII 1170-1250

* Seq. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

* $$F_n = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ F_{n-1} + F_{n-2} & n>1 \end{cases}$$

* Creix quasi tan ràpid com les potències de 2.

  $F_{30} > 1$ milió

  $F_{100} \rightarrow$ 21 digits de llargada!

  ⋮

  $F_n \approx 2^{0.694n}$

* Implementació → * Directa de la recurrència
  cost exponencial

  → * Guardant els dos últims valors
  cost lineal

  → $$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

  $$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

→ OBS:

We have been too liberal with what we consider a basic step.

It is reasonable to treat addition as a single computer step if small numbers are being added, 32-bit numbers say.

But the nth Fibonacci number is about 0,694 n bits long, and this can far exceed 32 as n grows.

Arithmetic operations on arbitrarily large numbers cannot possibly be performed in a single, constant time step

1

everis
an NTT DATA Company

Selecció en temps lineal
(Floyd - Rivest).

INPUT: Seqüència S de n elements
$0 \leq k < n$

OUTPUT: k-èsim element + petit de S

$k = 0 \rightarrow$ mínim
$k = n-1 \rightarrow$ màxim
$k = \lfloor \frac{n}{2} \rfloor \rightarrow$ mediana

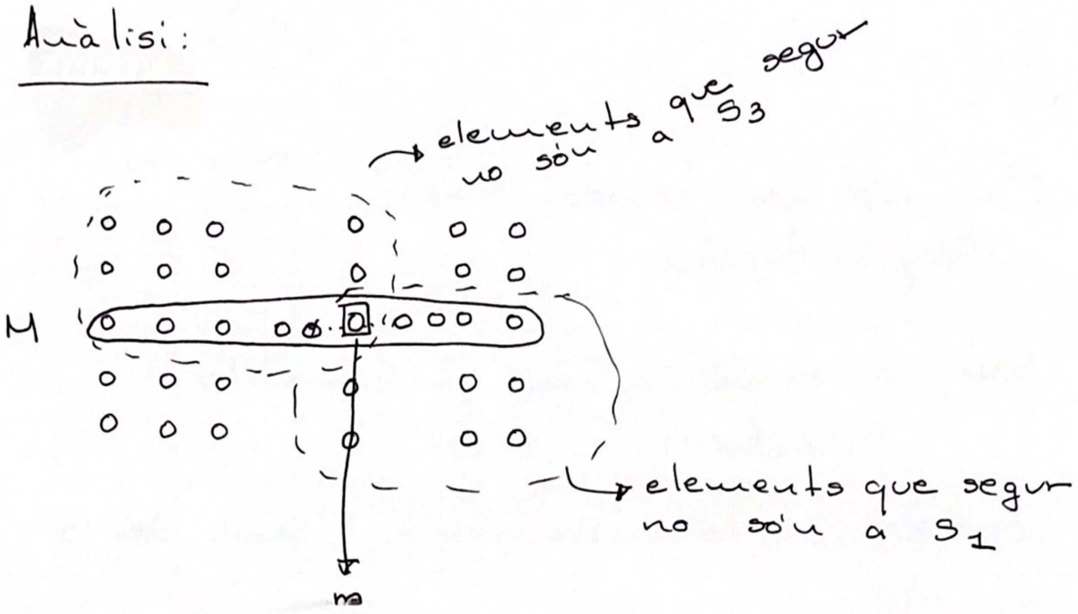
⋮

Algorisme:

SI $n < 50 \rightarrow$ directe.

SI NO
1) Dividir S en subconjunts de 5 elts.
2) Calcular la mediana de cada subc.
3) Calcular m = mediana de "medianas" cjto M (recursivament)
4) Separar S en S_1: $x \in S \mid x < m$
   S_2: $x \in S \mid x = m$
   S_3: $x \in S \mid x > m$
5) Si $|S_1| \geq k$ Buscar k rec. en S_1.
   sino:
   Si $|S_1| + |S_2| \geq k \rightarrow$ mediana m
   sino
   Calcular $k - |S_1| - |S_2|$ en S_3
   Buscar recursiv.

Anàlisi:



$\approx$ 1/4 elts són $\leq m$
i 1/4 elts són $\geq m$

$$T(n) \leq \begin{cases} c\,n & n \leq 50 \\ T\left(\frac{n}{5}\right) + T\left(\frac{3}{4}n\right) + \underbrace{\Theta(n)}_{c\cdot n} & n > 50 \end{cases}$$

claim:

$T(n) \leq 20\,c\cdot n$

Proof.

H.I. Cert per $1, \ldots, n-1$.

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3}{4}n\right) + cn \leq \frac{20cn}{5} + \frac{3\cdot 20\cdot c\cdot n}{4} + c\cdot n$$

$$= 4\cdot c\cdot n + 15\cdot c\cdot n + c\cdot n$$

$$= 20\,c\cdot n.$$