Titolo nota 16/03/2016

SORTING

$\Omega(n \log n)$ albero di decisione

Insertion Sort $O(n^2)$

Selection Sort $\Theta(n^2)$

Merge Sort $\Theta(n \log n)$ ottimo

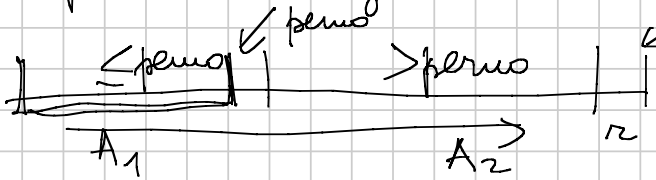
von Neumann (1945)

Quick_Sort Hoare (1962)

Divide et impera

1) **Dividi** : Scegli un elemento a caso perno

Dividi l'array in 2 parti : gli elementi < perno e gli elementi maggiori del perno



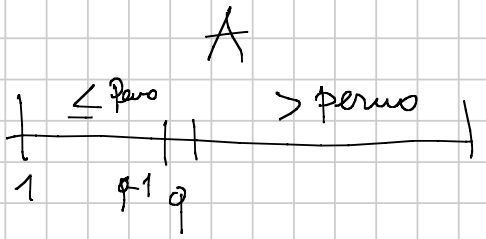
2) **Risolvi** direttamente se $n = 1$

ricorsivamente su A_1 e A_2

3) **Combina** : $\emptyset$

Titolo nota 16/03/2016

```
QUICKSORT (A, p, r)
if (p < r){
    q = PARTITION (A, p, r);
    QUICKSORT (A, p, q-1);
    QUICKSORT (A, q+1, r);
}
```



Se perno = max

Analisi

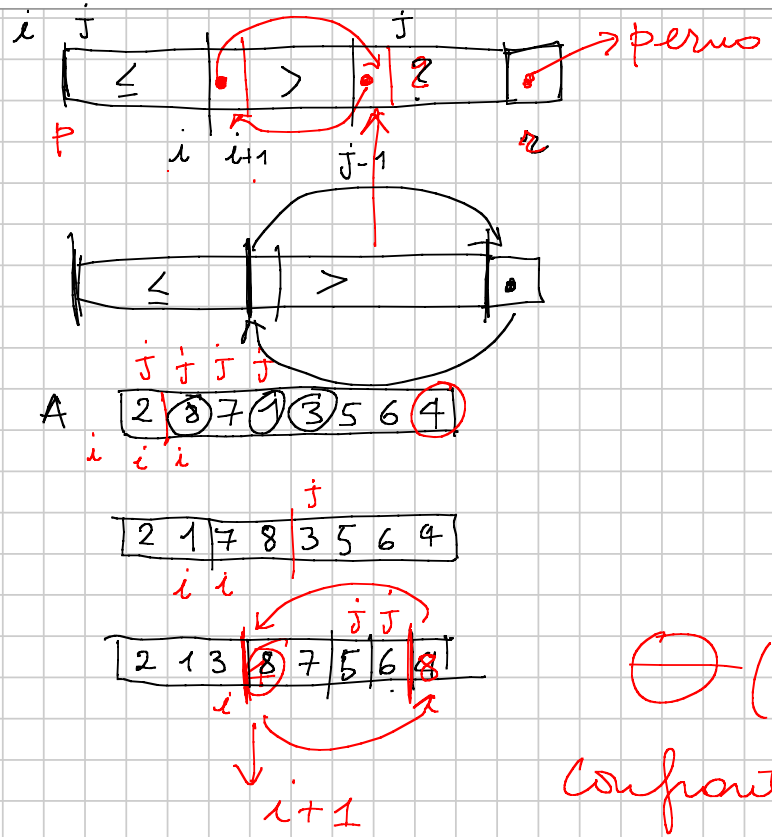
Partition

$$T(n) = T(i) + T(n-i-1) + \Theta(n)$$

$$0 \le i \le n-1$$

Titolo nota 16/03/2016

```
PARTITION (A, p, r);
x = A[r];
i = p-1;
for j = p to r-1 {
    if A[j] ≤ x {
        i++;
        scambia A[i] ↔ A[j];
    }
}
scambia A[i+1] con A[r];
return i+1;
```



$\Theta(n)$ confronti

Caso pessimo: il perno è il massimo o il minimo ad ogni livello di ricorrenza

QuickSort funziona male

$T(0) = T(1) = \Theta(1)$

$$T(n) = T(n-1) + \cancel{T(0)} + \overset{cn}{\cancel{\Theta(n)}}$$

$$T(n-1) = T(n-2) + c(n-1)$$

$$T(n) = T(n-2) + c(n-1) + cn$$

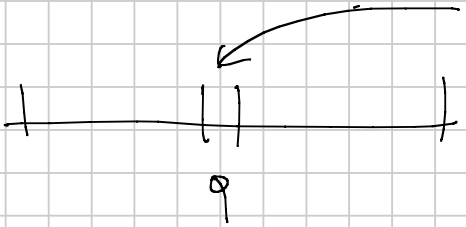
$$T(n) = T(n-3) + c(n-2) + c(n-1) + cn$$

$$\vdots$$

$$T(n) = \cancel{T(1)}^{c'} + c2 + c3 + \dots + c(n-2) + c(n-1) + cn$$

$$\leq c \sum_{i=1}^{n} i = c\,\frac{(n+1)(n)}{2} = \Theta(n^2)$$

caso ottimo: partizioni "abbastanza" bilanciate



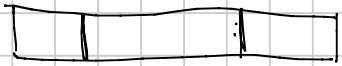
$$T(n) = 2T\left(\frac{n-1}{2}\right) + \Theta(n)$$

$$\leq 2T\left(\frac{n}{2}\right) + \Theta(n) = \text{MergeSort}$$

$a = 2 \quad b = 2 \quad n^{\log_b a} = n$

$f(n) \equiv n^{\log_b a}$ caso 2

$$T(n) = \Theta(n^{\log_b a} \cdot \log n) = \Theta(n \log n)$$

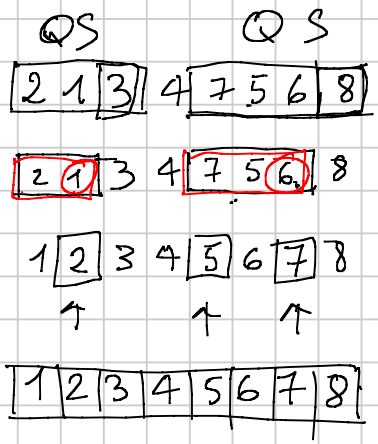


$$T(n) = T\left(\frac{2}{3}n\right) + T\left(\frac{n}{3}\right) + cn = \Theta(n \log n)$$

Titolo nota 16/03/2016

$$T(n) = T\left(\frac{9}{10}n\right) + T\left(\frac{n}{10}\right) + \Theta(n) \qquad \Theta(n \log n)$$

A



dopo Partition

=

=

1° chiamata

2° chiamate

3° e 4°

Titolo nota 16/03/2016

CASO MEDIO: ipotesi: distribuzione uniforme dei dati

PRATICA non è così: perché sottosequenze ordinate sono molto probabili.

Quando si applica a insieme ordinato $\Rightarrow \Theta(n^2)$

$X = A[r]$ $\qquad$ $X = \text{RANDOM}(p, r)$

↑ scelta casuale del perno in funzione della posizione

Algoritmo randomizzato o probabilistico

```
Randomized Partition (A, p, r):
    i ← RANDOM (p, r);
    scambiare A[i] ↔ A[r];
    return Partition (A, p, r);
```

scelta casuale del perno

```
Randomized_QuickSort (A, p, r):
    if p < r {
        q = Randomized_Partiton (A, p, r);
        Randomized_Quicksort (A, p, q-1);
        Randomized_Quicksort (A, q+1, r);
    }
```

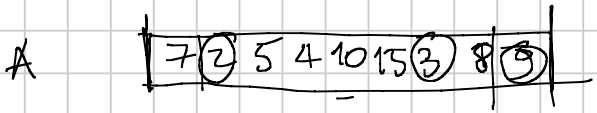
Randomized QUICKSORT — CASO MEDIO

A

A_i elemento di rango i in A

A_j elemento di rango j in A

$A[i]$ elemento in posizione i A_i i-esimo nell'ordinamento



$A[1] = 7$ $A_1 = 2$

$A[5] = 10$ $A_5 = 7$

$A_1 = 2$

$A_2 = 3$

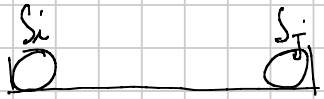
A_i e A_j

p_{ij} = probabilità che A_i si confronti A_j nella esecuzione di Q_S

se $j = i+1$ $p_{ij} = 1$ (si confrontano sicuramente)

non esiste elemento che li possa separare

Se $j \neq i+1$ e $j = i+1$



S_i e S_j si confrontano sse o S_i o S_j sono perno

$p_{ij} = \dfrac{\text{casi favorevoli}}{\text{tutti i casi}}$

$= \dfrac{2}{j-i+1}$

$k = j-i+1$

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} t_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} \leq \sum_{i=1}^{n-1} \sum_{k=2}^{n} \frac{2}{k}$$

$$= 2 \sum_{i=1}^{n-1} \sum_{k=2}^{n} \frac{1}{k} = 2n \sum_{k=1}^{n} \frac{1}{k} = 2n(\ln n + \Theta(1)) = \boxed{\Theta(n \log n)}$$

Selection $(A, p, r, k) \Rightarrow$ (Q. S. una modifica di)

Selezione del K-esimo elemento

Input: Array A non ordinato

Output: Trovare l'elemento di rango K (A_k)

$K = 4 \quad A_k = A[3]$

A 10 2 7 5 3 15 11 8

Soluzione banale: Ordino il K-esimo si trova in posizione A[K]

2 3 5 7 8 10 11 15