Algoritmi e Strutture Dati (19/10/2021)

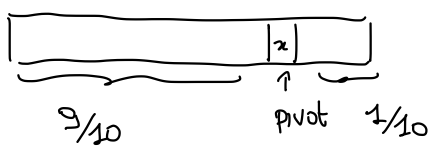
* Quick Sort

$$T^{QS}_{max}(n) = \Theta(n^2)$$

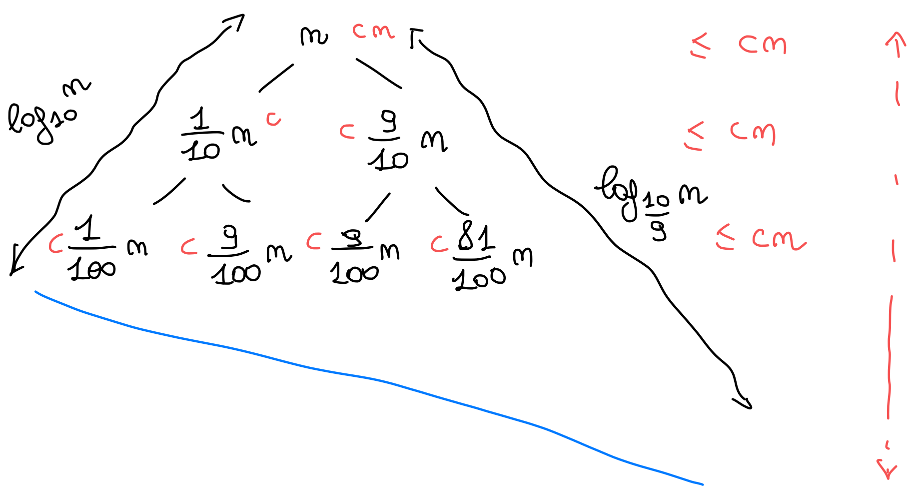
$$T^{QS}_{min}(n) = \Theta(n \log n)$$

$$T^{QS}_{med}(n) = \Theta(n \log n)$$

$\boxed{\text{OSS1}}$ Portizionamento proporzionale



$$T(n) = T\left(\frac{9}{10}n\right) + T\left(\frac{1}{10}n\right) + \underbrace{\Theta(n)}_{\leq cn}$$

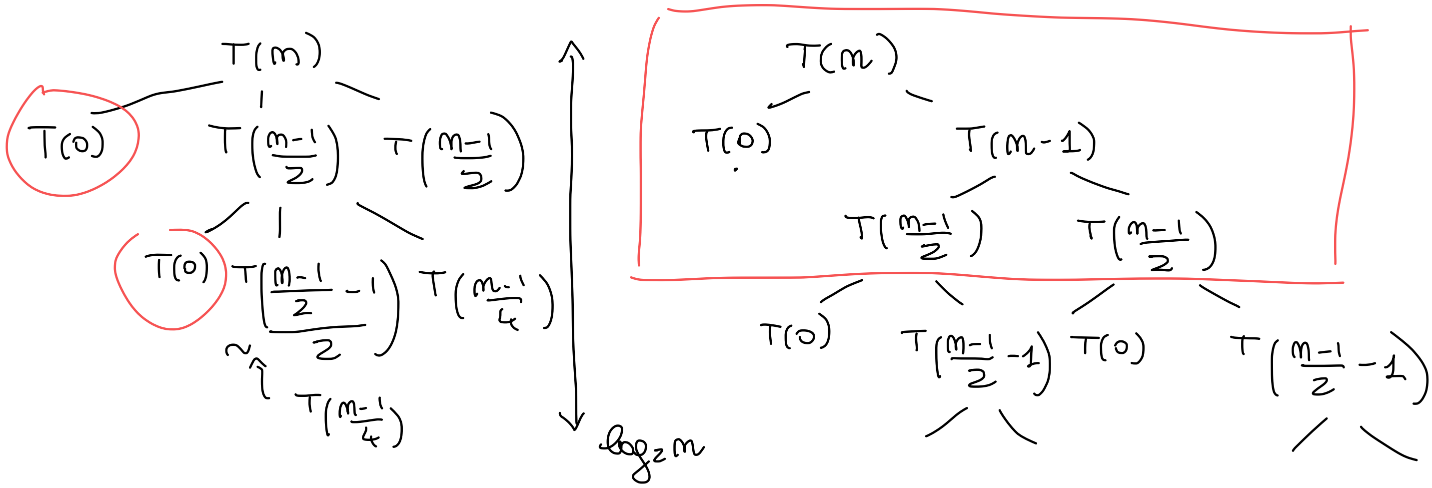


$$T(n) \leq c n \log_{\frac{10}{9}} n$$

$$= O(n \log n)$$

$$\log_{\frac{10}{9}} n = \underbrace{\log_{\frac{10}{9}} 2}_{2,7} \cdot \log_2 n$$

$\boxed{\text{OSS 2}}$ alternanze tra partizionamenti pessimi/ottimi



$$T(n) = O(n \log n)$$

$\leadsto$ $$T^{qs}_{med}(n) = \Theta(n \log n)$$

* se l'input non ha distribuzione uniforme?

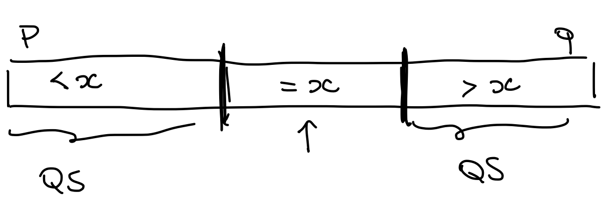
es: se gli array di input sono "quasi ordinati"

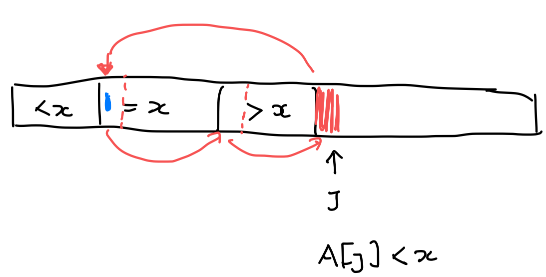
```
Randomized Partition (A, p, r)
    i = Random (p, r)
    A[r] <-> A[i]
    return Partition (A, p, r)
```

* non va comunque bene se tutti gli elementi sono uguali

Tripartion per Quicksort

x pivot





ESERCIZIO

Limite inferiore per l'ordinamento

problema:

input: $a_1, \dots, a_n$

output: $a'_1, \dots, a'_n$ permutazione ordinata di $a_1, \dots, a_n$

ordinamento basato su confronti (e assegnamenti)

$$\Omega(n \log n)$$

$\Rightarrow$ $\boxed{\text{limite inferiore per le operazioni di confronto}}$

$\Rightarrow$ $\boxed{\text{assumiamo che tutti gli elementi siano distinti} \quad \forall i,j \;\; i \neq j \quad a_i \neq a_j}$

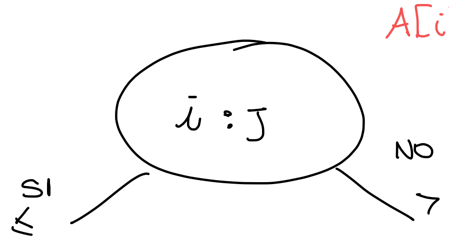
$\leq$ $\quad \cancel{\neq}$ $\quad <$ $\quad \geq$ $\quad >$

* **Alberi di decisione**

rappresentazione (astratta) dell'esecuzione di un algoritmo su input di dimensione prefissata n

$\rightarrow$ nodi interni

$A[i] \leq A[j]$?



$\rightarrow$ foglie : output permutazione dell'input

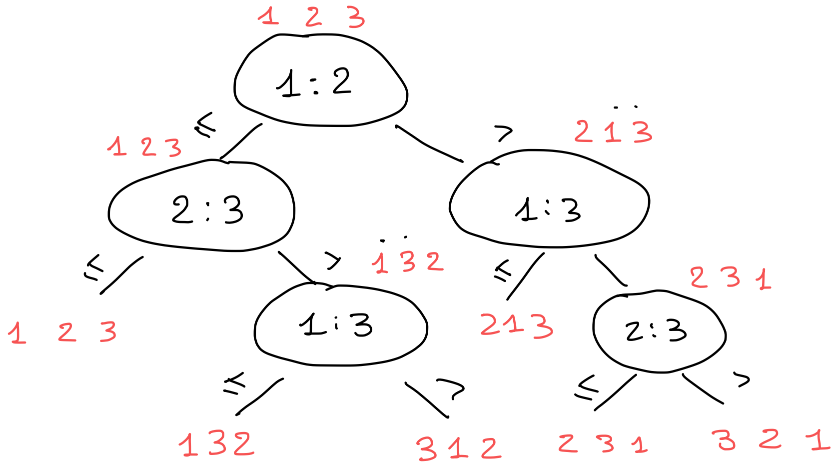
Esempio : InsertionSort

input $n = 3$

A 1 2 3

1	2	3

```
InsSort(A, n)
  for j = 2 to n
    i = j-1
    while (i > 0) and (A[i] > A[i+1])
      A[i] <-> A[i+1]
      i = i-1
```

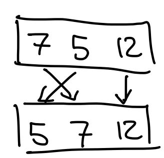


* altezza dell'albero di decisione (per input di dim n) dà il numero di confronti necessario (nel caso peggiore) con input di dim. n

2 OSSERVAZIONI

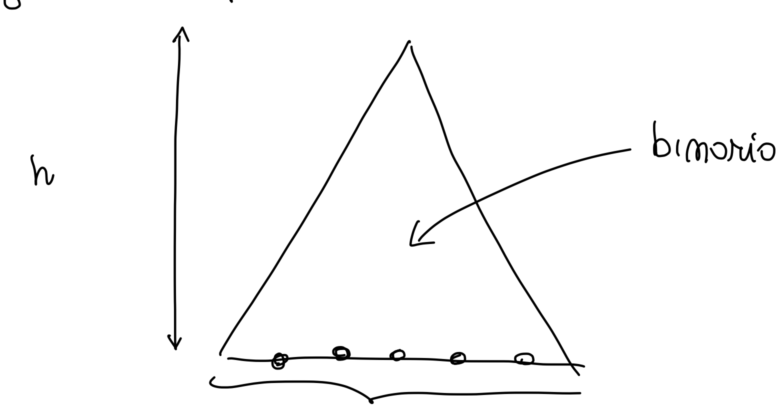
→ $\boxed{\text{ogni permutazione deve comparire in almeno una foglia}}$

$n = 3$ nessuna foglia contiene 2 1 3



→ $\boxed{\text{ogni foglia contiene una sola permutazione}}$

$\Downarrow$ # foglie $\geq$ # permutazioni $= n!$



foglie $\geq n!$

foglie $\leq 2^h$

$h \geq \log_2$ (# foglie)

$$\#\text{operazioni su input dim } n \geq h \geq \log_2(\#\text{foglie}) \geq \log_2 n!$$

$$= \log_2 n(n-1)(n-2)\cdots\left(\frac{n}{2}\right)\underbrace{\left(\frac{n}{2}-1\right)\cdots 2 \; 1}$$

$$\geq \log_2 \underset{\frac{n}{2}}{\underset{\vee\!|}{n}}\;\underset{\frac{n}{2}}{\underset{\vee\!|}{(n-1)}}\;\underset{\frac{n}{2}}{\underset{\vee\!|}{(n-2)}}\cdots\frac{n}{2}$$

$$\geq \log_2\left(\frac{n}{2}\right)^{\frac{n}{2}} = \frac{n}{2}\log_2\frac{n}{2} = \frac{n}{2}(\log_2 n - 1)$$

$$= \Theta(n \log n)$$

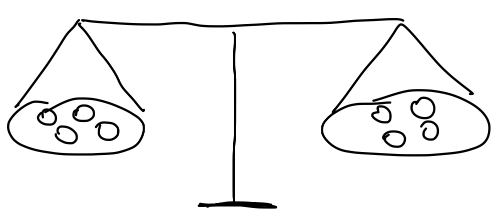
$$\#\text{operazioni su input di dim } n = \Omega(n \log n)$$

problema dell'ordinamento ha complessità $\Omega(n \log n)$

→ Merge Sort, Heap Sort $O(n \log n)$ ⇝ $O(n \log n)$

$\Theta(n \log n)$

ESERCIZIO: 10 monete (n) una falsa (peso minore)



quante pesate per trovare la moneta falsa

$O(\log_{\cancel{3}} n)$

problema $\Omega(\log n)$

Select (A, i) un elemento in posizione i nell'array ordinato

$A[1..n]$

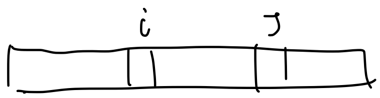
select ($\boxed{3|7|1|2|5}$, 4) = 5

1 2 3 5 7

↑

1. $O(n \log n)$
2. $O(n + i \log n)$ } ESERCIZIO
3. $O(n \log i)$
4. $O(n)$

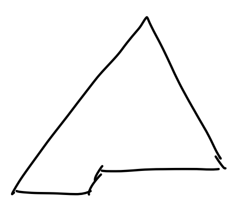
* **Inversioni**

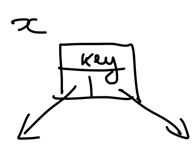


$i < j$ $\quad A[i] > A[j]$

$Inv(A) = \#$ inversioni (⇝ ~ mergesort)

* Max Heap come linked structures

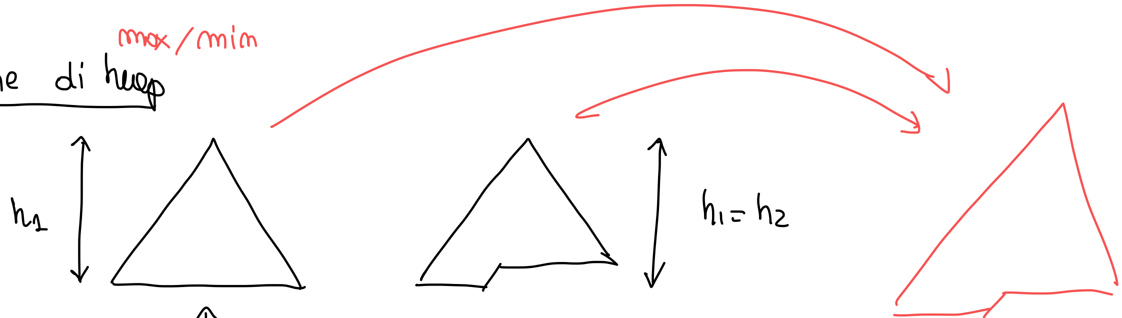




x.key
x.left
x.right

heap { radice + informazioni (es size) + operazioni

* **Fusione di heap** max/min



↑ completo

→ array : $O(n)$ ←

→ linked : $O(\log n)$