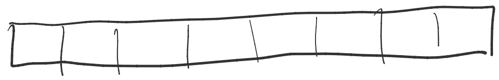
Lezione 12 marzo

Thursday, March 12, 2020 4:01 PM

multiinsieme
n elementi
con ripetizioni

a 

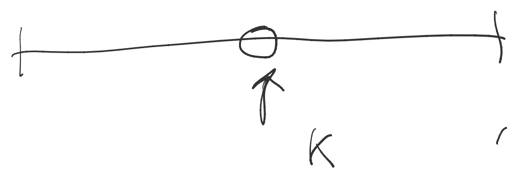
K

Contare le occorrenze

ContaOccorrenze(a, K):

Ordina l'array a $\Theta(n \log n)$

RICERCABINARIA 2 si arresta non appena trova K



return

? Ricerca binaria $O(\log n)$

occ = # numero di occorrenze di K

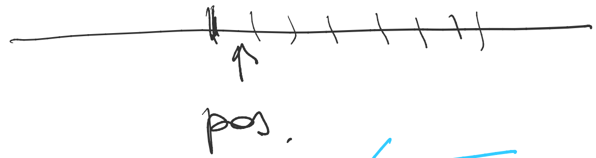
$O(\log n) + \Theta(K)$

$\boxed{occ = \text{costante} : O(\log n)}$

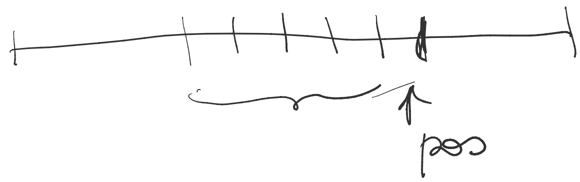
$occ = O(n) \rightarrow O(n)$ caso pessimo

$occ = O(n) \to O(n)$ caso pessimo

RICERCA BINARIA 3



RICERCA BINARIA 4 ← esercizio




```
ContaOcc(a);
  primaocc = RicercaBinaria3(a,K,1,n);
  if (primaocc == -1) return 0;
  ultimaocc = RicercaBinaria4(a,K,1,n);
                                  primaocc
  return (ultimaocc - primaocc + 1);
```

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

Limiti inferiori

Ordinamento   Sorting

IS $O(n^2)$

SS $\Theta(n^2)$

MS $\Theta(n \log n)$ time $\Theta(n)$ spazio

operazione base : confronto tra elementi

Ragionamento come per il problema delle 12 monete.

albero delle decisioni per det. un limite inferiore al numero di pesate

a b c

4 3 7

3 4 7

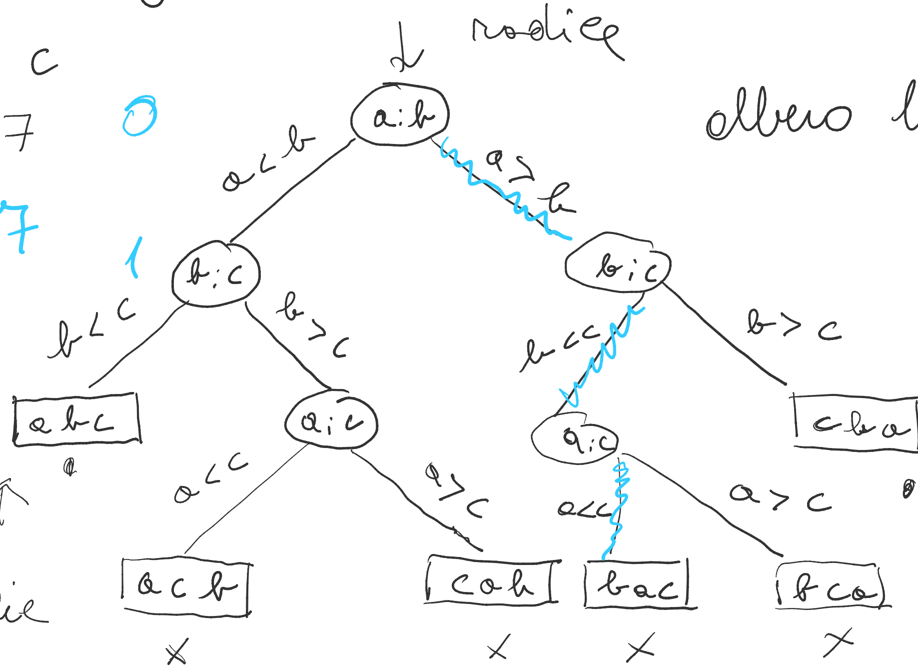
7 3 4

0

1

2

3 foglie



albero binario

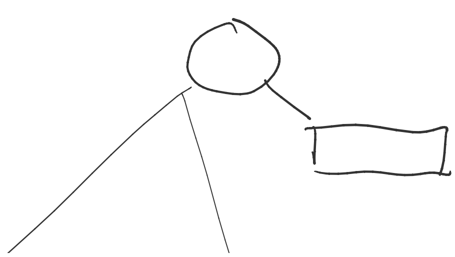
6 pl.

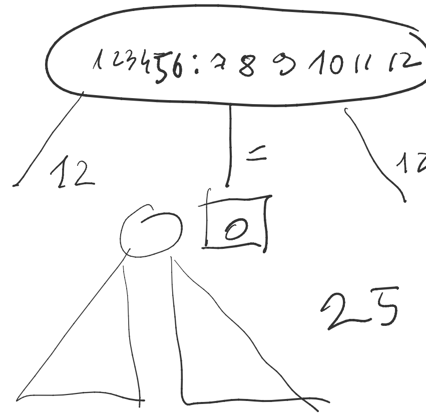
2 confronti

3 confronti

n elementi

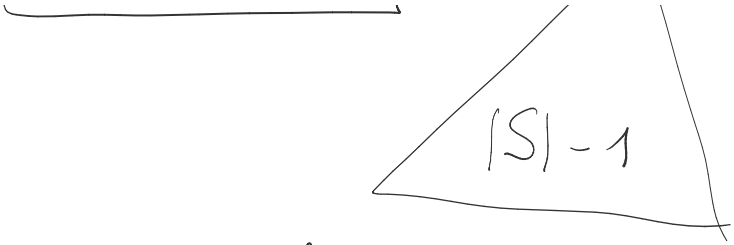
albero bilanciato



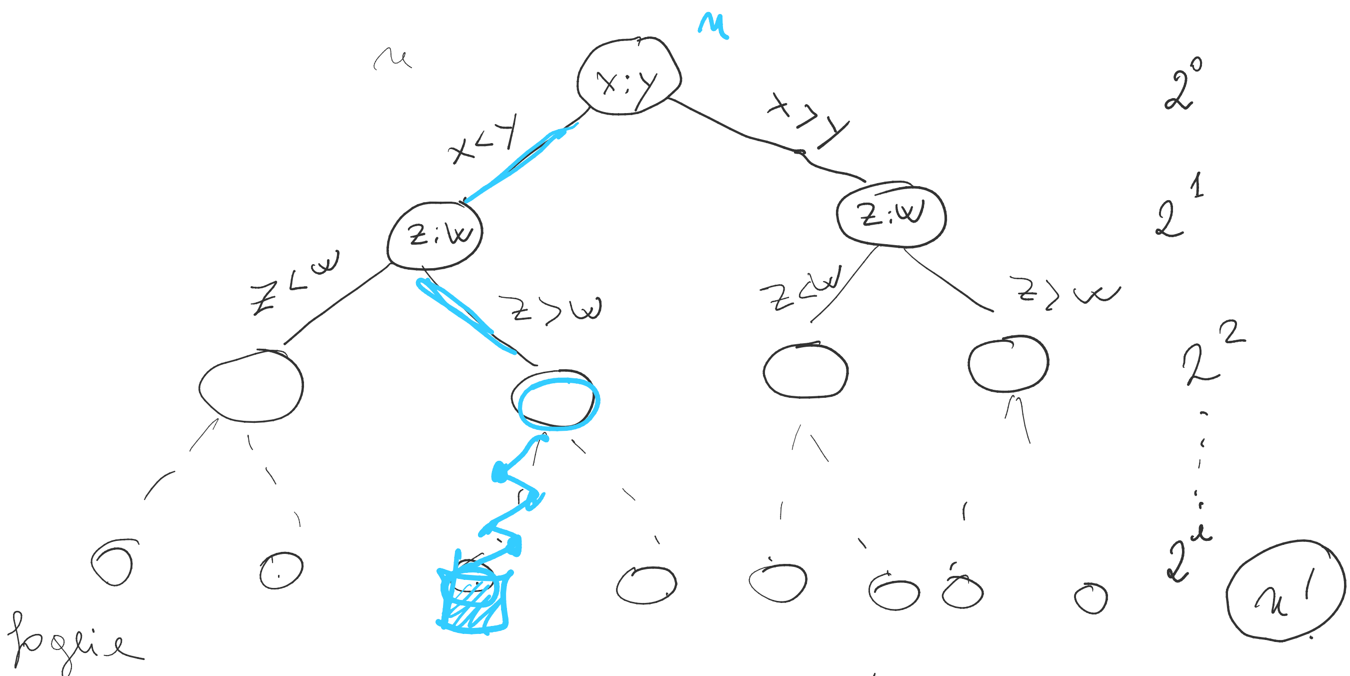


25

$\boxed{3! = 2 \cdot 3 = 6}$



n elementi distinti



Contare il n° di soluzioni $|S|$

$|S| = 25$ (1L1P2L2P... ... 12L 1P 0 problema monete)

n elementi da ordinare

quante sono ?

$|S| = n!$

$$\boxed{3^i \geq 25}$$

$$2^i \geq n!$$

l'albero di decisione deve avere tante foglie da contenere tutte soluzioni

$$\log_2 2^i \geq \log_2 (n!)$$

$$\log_2 2^i \geq \log_2(n!)$$

$$i \geq \log(n!)$$

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

APPROSS. DI STIRLING

$$i \geq \Theta(n \log n)$$

cosa rappresenta i: è il livello delle foglie nell'albero di decisione

Occorrono almeno i confronti per arrivare alla permutazione ordinata

Sorting è $\Omega(n \log n)$

Merge Sort $\Theta(n \log n)$ time $\Theta(n)$ spazio

limite inferiore $\Omega(n \log n)$

Merge Sort è ottimo (in tempo)

Limiti inferiori

1) Dimensione dell'input/output

SORTING: tutti gli elementi vanno esaminati

limite inf.: $\Omega(n)$ non è significativo

$\Omega(n \log n)$

Stampa tutte permutazioni di un array di

n elementi. output: n! permutazioni

Stampa tutte permutazioni di un array di n elementi. <u>output</u>: n! permutazioni

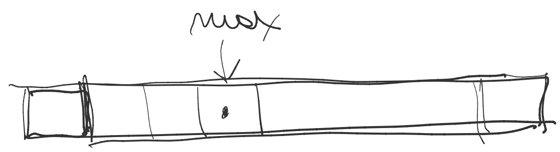
$$\Omega(n!) = \Omega(n^n) \quad \underline{\text{esponenziale}}$$

2) Eventi contabili

for i = 2 to n
{

<u>Trova max</u>

<u>banale</u>



SelSort
min

max = a[1]

$C(n) = n - 1$

$\Omega(n)$

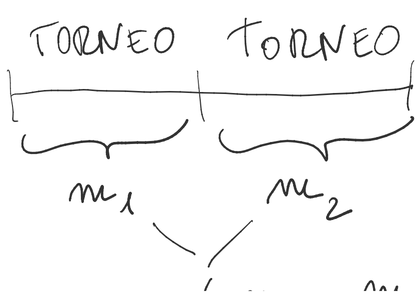
valutare esattamente quanti sono necessari.

<u>Evento</u>: per essere giudicato non max un elemento dell'array deve perdere in almeno un confronto

in totale gli el. non massimi sono n-1. <u>Occorrono n-1 confronti</u>

Divide et Impera

TORNEO



a

Esercizio

$\boxed{n = 2^k}$

$m_1 \quad m_2$

$\max(m_1, m_2)$

$\boxed{n = 2^k}$

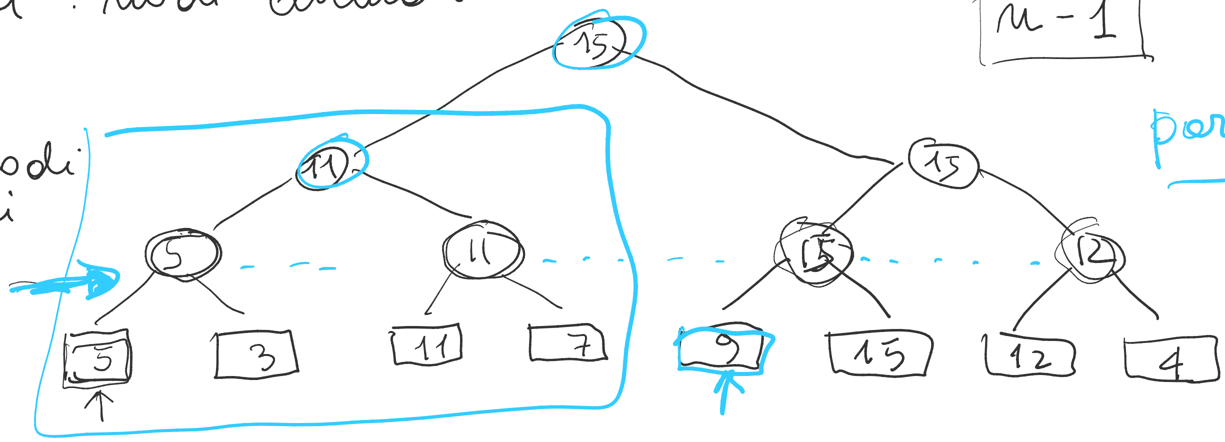
Confronti : nodi cerchiati

$\boxed{n-1}$

parallelo

n-1 nodi interni

n foglie



Si determina il max con n-1 confronti. TORNEO è ottimo!

$$C(n) = \begin{cases} 0 & \text{se } n = 1 \\ 2C\left(\frac{n}{2}\right) + 1 \end{cases}$$

senza teorema dell'esperto.
col metodo iterativo.

$\boxed{\text{TORNEO A ELIMINAZIONE DIRETTA}}$

determinare PRIMO
determinare SECONDO

Francia : $\boxed{\text{Croazia}}$

↓ ↓

Campione seconda

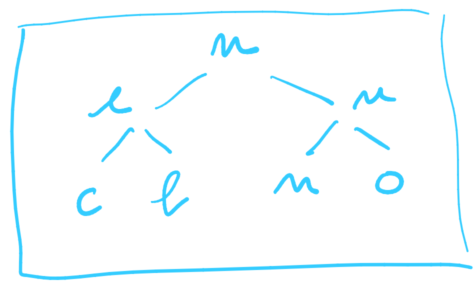
... la seconda ... ?

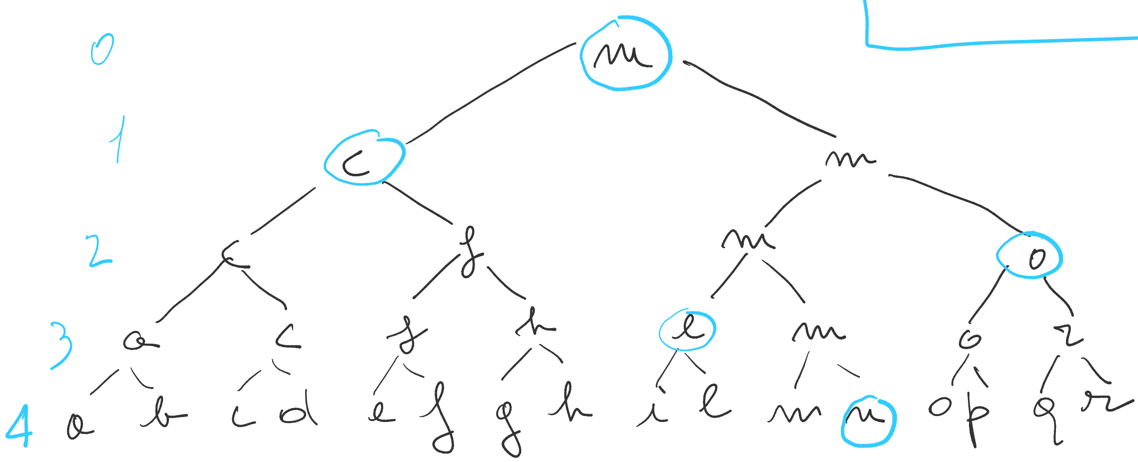
È proprio il secondo più bravo?

pr (secondo più bravo risulti secondo) = $\frac{n}{2(n-1)} \simeq \frac{1}{2}$

Lewis Carrol: Alice nel paese delle meraviglie

pubblicò un articolo: ingiustizia dei tornei di tennis





$n = 2^4$

Alg. DOPPIO TORNEO

1) TORNEO su n giocatori $\quad n-1$
2) TORNEO su $\log_2 n$ giocatori $\quad \log n - 1$

$$C = n - 1 + \log n - 1 = n + \log n - 2$$

$n + \log n - 2$

$\boxed{\text{è OTTIMO}}$ $\boxed{n = 2^k}$

Il problema di stabilire PRIMO e SECONDO richiede $n + \log n - 2$ confronti

tecnica dell'oracolo/avversario

pone sempre l'alternativa peggiore

$x : y$

minimizzare l'informazioni che posso ottenere per transitività.