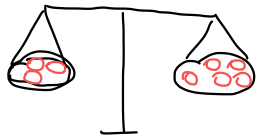


# Algoritmi e Strutture Dati (04/11/2021)

## Esercizio (Moneta falsa)

insieme di  $n$  monete  
una falsa (pesa meno delle altre)

bilancia



- ① algoritmo per trovare la moneta falsa  $O(\log n)$
- ② limite inferiore
- ③ variante: moneta falsa ha peso diverso (non sappiamo se maggiore o minore)

### ① algoritmo

operazione: suddivido  $S$  insieme di monete in tre insiemi disgiunti

$$S = S_1 \uplus S_2 \uplus S_3 \quad S_i \cap S_j = \emptyset \quad i \neq j$$

confronto il peso di  $S_1$  e  $S_2$

$$\begin{array}{llll} \text{peso } S_1 & < & \text{peso } S_2 \\ \text{"} & \text{"} & = & \text{"} \\ \text{"} & \text{"} & > & \text{"} \end{array}$$

Nota: è conveniente  $|S_1| = |S_2|$

( se  $|S_1| < |S_2|$  )

con questa assunzione

$$\begin{array}{llll} \text{peso } S_1 & < & \text{peso } S_2 & \leadsto \text{falsa } \bar{e} \text{ in } S_1 \\ \text{"} & \text{"} & = & \text{"} \\ \text{"} & \text{"} & > & \text{"} \end{array} \begin{array}{l} \leadsto \text{falsa } \bar{e} \text{ in } S_3 \\ \leadsto \text{falsa } \bar{e} \text{ in } S_2 \end{array}$$

Algoritmo: parto da  $S$   
Considera  $S = S_1 \uplus S_2 \uplus S_3$  con  $|S_1| = |S_2| = \lceil n/3 \rceil$  ( $|S_3| \leq \lceil n/3 \rceil$ )

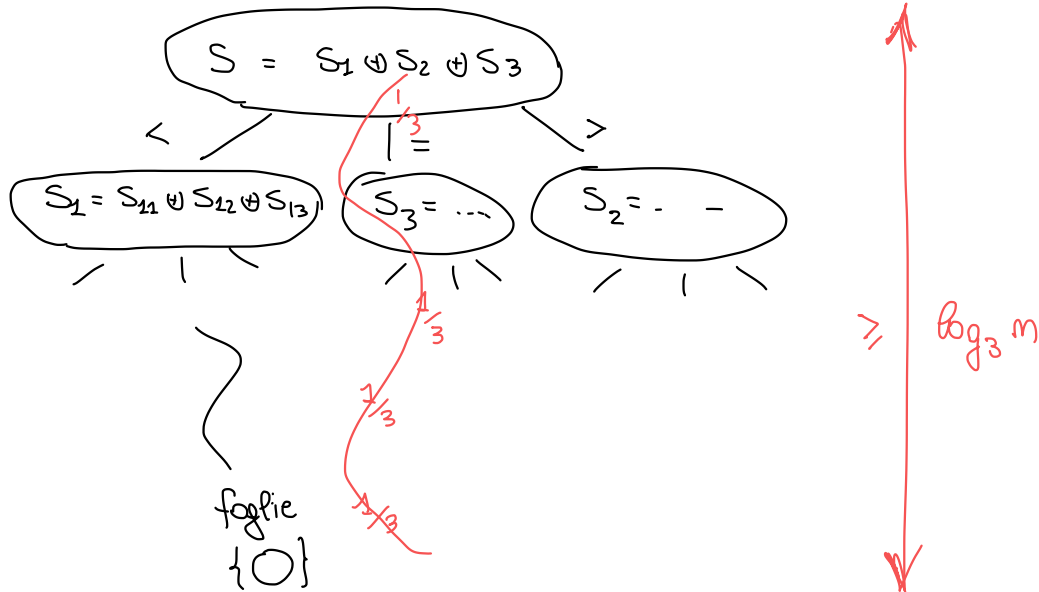
sulla base della pesata  $S = \begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix}$

fino a che  $|S| = 1$

$$\text{costo } O(\log_3 m) \quad (= O(\log_2 m))$$

② limite inferiore?

albero di decisione



OSSERVAZIONE: se  $S = S_1 \oplus S_2 \oplus S_3$  per qualche  $i$   $|S_i| \geq \frac{|S|}{3}$

altezza  $\geq \log_3 m$

$\Rightarrow$  problema  $\Omega(\log m)$

ma anche  $O(\log m)$  per ①

$\Rightarrow \Theta(\log m)$

③ se la moneta falsa ha semplicemente peso "diverso"?

$$S = S_1 \oplus S_2 \oplus S_3$$

$$|S_1| = |S_2| = \lfloor m/3 \rfloor$$

$\rightarrow$  confronto  $S_1$  e  $S_2$

\* stesso peso : falsa in  $S_3$   
 $S_1$  e  $S_2$  monete vere  
 prendo  $|S_3|$  monete da  $S_1 \oplus S_2$

e lo confronto con  $S_3$

ma deduco se la moneta falsa pesa più o meno

\*  $S_1$  pesa meno di  $S_2$

falsa in  $S_1$  oppure  $S_2$

prendo  $|S_1|$  monete da  $S_3$  (+ tutte vere)  $\rightarrow S'_3$

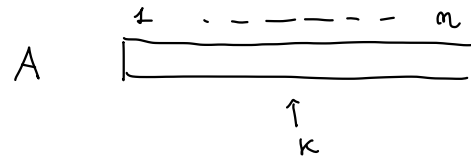
confronto  $S_1$  e queste monete

se = falsa in  $S_2$  e pesa di più

impossibile  $\rightarrow$  ~~se  $S_1$  pesa di più la falsa è in  $S_1$  e pesa di più~~

se  $S_1$  pesa di meno la falsa è in  $S_1$  e pesa di meno

ESERCIZIO :  $\text{Select}(A, k)$

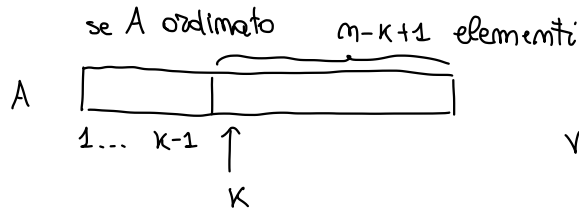


$\rightarrow$  elemento in posizione  $k$  se  $A$  ordinato

① ordino  $A$ , ritorno  $A[k]$   
 $O(m \log m)$        $O(1)$       }  $O(m \log m)$

②  $O(m + k \log m)$

idea :



valgo: il minimo degli  $m-k+1$  elementi più grandi di  $A$

uso min-heap

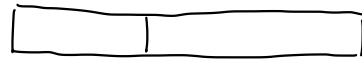
$\rightarrow$  trasformo  $A$  in min-heap  $O(m)$   
 $\left\{ \begin{array}{l} \rightarrow \text{"elimino" i } k-1 \text{ elementi più piccoli (con Extract Min)} \quad O(\log m) \text{ } k-1 \text{ volte} \\ \rightarrow \text{prendo il minimo di ciò che rimane} \quad O(1) \end{array} \right.$

$\leadsto O(m + (k-1) \log m) = O(m + k \log m)$

③  $O(m \log k)$

A ordinato

idea

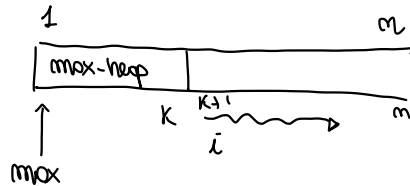


$\uparrow$   
k massimo dei k elementi più piccoli

uso max-heap

①  $\rightarrow$  trasformo  $A[1..k]$  in un max-heap

$O(k)$



inv. :  $A[1..k]$  max heap

$A[1..k] \leq A[1..i-1]$

$\rightarrow$  per ogni elemento  $A[i]$   $i = k+1 \dots m$

$m-k$  volte

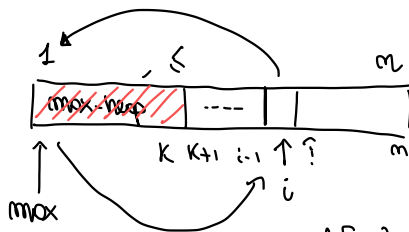
confronto  $A[1]$  con  $A[i]$

se  $A[1] > A[i]$

scambio

max heapify per ristabilire la proprietà di max heap

$O(\log k)$



$A[1] \leq A[i]$

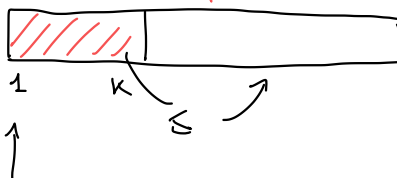
non faccio niente

$\uparrow$

$A[1] > A[i]$

$\rightarrow$  alla fine:

max heap



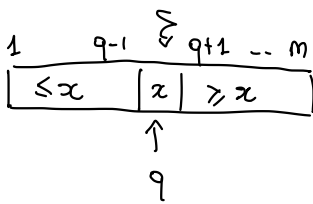
$A[1]$  è il massimo dei k elementi più piccoli di A

costo :

$$\begin{aligned} O\left(\underbrace{k}_m + \underbrace{(m-k)}_m \log k\right) &= O(m + m \log k) \\ &= O(m \log k) \end{aligned}$$

#### ④ Quick Select

partition A 

A 

→ se  $k = q \rightarrow$  ritorno  $A[q]$

→ se  $k < q \rightarrow$  ricorrenza su  $A[1..q-1]$

→ se  $k > q \rightarrow$  " "  $A[q+1..m]$

caso peggiore

$$T(m) = T(m-1) + \Theta(m) \quad \text{ma} \quad T(m) = \Theta(m^2)$$

caso "migliore":

$$T(m) = T\left(\frac{m}{2}\right) + \Theta(m) \quad \text{ma} \quad T(m) = ?$$

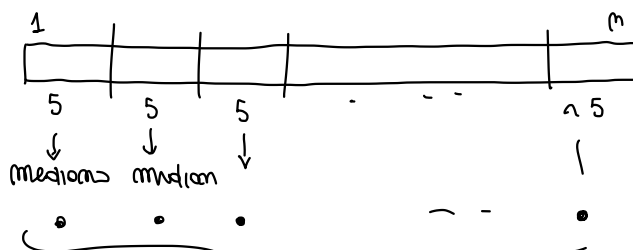
$\uparrow \quad \quad \uparrow \quad \quad \uparrow$   
 $a=1 \quad b=2 \quad f(m)$

$$m^{\log_a b} = m^{\log_2 2} = m^1 \quad \text{con} \quad f(m) = \Theta(m)$$

↳ presunto "caso 3"  
 → verifica rigorosa }  $\rightarrow T(m) = \Theta(m)$

↳ coincide con il caso medio

#### ⑤ select in tempo lineare nel caso peggiore



$\frac{\lceil m \rceil}{5}$  gruppi di 5 elementi

→ mediana di ogni gruppo

→ mediana dei mediani

$\text{select}(\dots, \frac{\lceil m \rceil}{10})$

↓

→ partizione con pivot elemento  
attenuato

$\frac{7}{10}m$

$$T(m) = T\left(\frac{\lceil m \rceil}{5}\right) + T\left(\frac{7}{10}m + \dots\right) + \Theta(m)$$

↑ soluzione  $\Theta(m)$

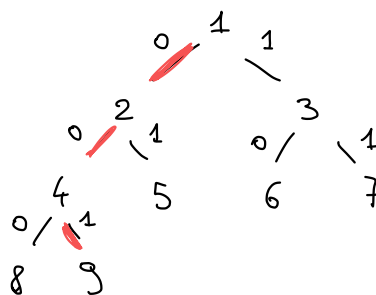
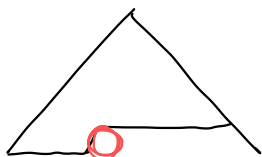
Esercizio: (Linked Max Heap)

modi  $x$   $\left\{ \begin{array}{l} x.\text{left} \\ x.\text{right} \\ x.p \\ x.\text{key} \end{array} \right.$

Heap  $H$   $\left\{ \begin{array}{l} H.\text{root} \\ H.\text{size} \end{array} \right.$

- Max
- MaxHeapify
- MaxHeapifyUp
- ⋮

Insert ( $H, x$ )



modo  $i$

$i = 1 \dots \dots$  bimario  
↑  
cammino  
per arrivare  
ad  $i$

$g = \underline{\underline{1001}}$

Inserimento di un nuovo nodo

$H.\text{size} + 1 \rightarrow$  bimario

$1 \underline{\underline{x \dots x}}$   
cammino  
che porta alla posizione  
della nuova foglia