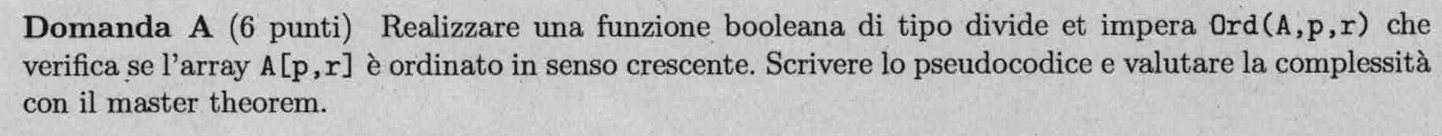
Immagine che contiene testo, ricevuta, Carattere, bianco

Descrizione generata automaticamente

Input:

* A con lunghezza “n”
* “p”, “r”, indici non vuoti

Induzione:

* Caso “base” 🡪 abbiamo gli indici uguali
  + Il minimo è la metà
* Caso induttivo (n+1)
  + Minimum (A, p+1, q), A(q+1, r)
  + Divido in due metà e, dato che salvo il minimo ad ogni iterazione
  + Allora la metà che ottengo è quella minima

Ord\_routine(A, p, r)

MergeSort(A)

Ord(A, p, r)

Ord(A, p, r) 🡪 1 2 3 4

if(p == r) return True

else

return (Ord(A, p, q), Ord(A, q+1, r))

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Esempio pratico: la somma di tutti gli elementi è al più 1 (voglio minimizzare le iterazioni per trovare “subito” la somma che si ferma ad 1 (o si avvicina il più possibile)

0.1 0.3 0.5 0.1 0.9

0.1 0.3 | 0.5 0.1 0.9

0.1 0.3 0.5 | 0.1 0.9

Due partizioni da due elementi, ma il minimo è 1 (0.1 e 0.9)

2-1\_Boxing(A)

n = length(A)

sol = {A\_n}

for(A from 1 to n-1)

if(sol – A[i] <=1)

sol = sol {A\_i}

else

break

return sol

2)

Scelta greedy: Dato un array ordinato, salvo e procedo induttivamente all’indietro aggiungendo finché la somma è .

La soluzione è ottima perché:

* Data , aggiungo , tutte le tali che
* Se sol è maggiore di 1 esci, altrimenti continua

Domanda (B)

toTree 🡪 Max heap trasformato ad albero

toTree(A)

x = A.root

x = A/2

for(int i = x-1 down to 1)

A[i] = node()

if(A[i] < x)

x = x.left

else

x = x.right

for(int i = x+1 to A.length)

A[i] = node()

if(A[i] < x)

x = x.left

else

x = x.right