

Algoritmi e Strutture Dati - 17/06/13

Esercizio 1 – Punti ≥ 6

Trovare un limite superiore e un limite inferiore alla complessità della seguente procedura:

```

fun(integer[]  $V$ , integer  $i$ , integer  $j$ )
    if  $i = j$  then
        | return 1
    integer  $T \leftarrow 0$ 
    for  $k \leftarrow i$  to  $j$  do
        |  $T \leftarrow T + V[k]$ 
    return  $T + \text{fun}(V, i, j - 1) + \text{fun}(V, i, i + \lfloor \sqrt{j - i + 1} \rfloor)$ 

```

inizialmente chiamata in questo modo

$\text{fun}(V, 1, n)$

Esercizio 2 – Punti ≥ 6

Sia $A[1 \dots 4n]$ un vettore di interi distinti. Si scriva un algoritmo efficiente per suddividere il vettore A nelle quattro righe di una matrice $B[1 \dots 4, 1 \dots n]$; ovvero nelle righe $B[1], B[2], B[3], B[4]$, ciascuna composta di n elementi. La suddivisione deve essere tale che se $i < j$ allora ogni elemento nella riga $B[i]$ è minore di ogni elemento nella riga $B[j]$. Gli elementi all'interno di ogni sottovettore non devono essere necessariamente essere ordinati. Per efficiente si intende un algoritmo che abbia complessità strettamente inferiore a $\Theta(n \log n)$.

Illustrare il funzionamento e discutere la complessità dell'algoritmo proposto.

Esercizio 3 – Punti $\geq 9 + 3$

Hateville è un villaggio particolare, composto da n case, numerate da 1 a n lungo una singola strada. Ad Hateville ognuno odia i propri vicini della porta accanto, da entrambi i lati; quindi il vicino i odia i vicini $i - 1$ e $i + 1$ (se esistenti). Hateville vuole organizzare una sagra e ha lanciato una raccolta fondi che è vostro compito organizzare. Ogni abitante i è in grado di donare una quantità $D[i]$, ma non intende partecipare ad una raccolta fondi a cui partecipano uno o entrambi i propri vicini.

Il vostro compito è il seguente:

- calcolare la quantità massima di fondi che può essere raccolta
- stampare gli indici delle case che dovranno donare

Illustrare il funzionamento e discutere la complessità degli algoritmi proposti.

Esercizio 4 – Punti ≥ 9

Scrivere un algoritmo che dato un grafo orientato G e due nodi u, v , ritorna vero se esistono almeno k cammini edge-disjoint (ovvero che non abbiano alcun arco in comune) da u a v .

Illustrare il funzionamento e discutere la complessità dell'algoritmo proposto.