

## PRIMA PARTE

1. **[4 punti]** Siano  $A$  e  $B$  due algoritmi che risolvono lo stesso problema  $\Pi$ . Si definisca  $t_X(n)$  la complessità al caso pessimo di  $X$  e  $t_{X,i}$  il numero di operazioni eseguite da  $X$  per risolvere l'istanza  $i$ , per  $X \in \{A, B\}$ . Sapendo che  $t_{B,i} \geq (1/10)t_{A,i}$  per ogni  $i$ , e che  $t_A(n) \in \Theta(n^2)$ , cosa si può dire di  $t_B(n)$ ? Motivare la risposta.
2. **[4 punti]** Si consideri l'array di entry:  $P = [(11, *), (9, *), (7, *), (13, *), (3, *), (4, *)]$ , e l'approccio *bottom-up* per la costruzione di uno *heap* a partire da  $P$ . Mostrare l'albero associato all'array alla fine di ogni iterazione dell'approccio *bottom-up*. (È sufficiente mostrare le chiavi delle entry.)
3. **[4 punti]** Si faccia riferimento alle implementazioni della Mappa basate su Alberi Binari di Ricerca (ABR) e (2,4)-Tree.
  - (a) Descrivere brevemente ma in modo preciso come il (2,4)-Tree generalizza l'ABR.
  - (b) Quali sono i vantaggi dell'implementazione della Mappa con (2,4)-Tree rispetto a quella con ABR?
4. **[4 punti]** Sia  $S$  una sequenza di  $n$  chiavi intere da ordinare. Dire quale tra gli algoritmi di ordinamento che conoscete usereste nei seguenti casi, specificando la complessità risultante: (i) le chiavi da ordinare appartengono all'intervallo  $[1, n^2]$ ; (ii) esistono 3 chiavi, tolte le quali  $S$  risulta già ordinata. Motivare le risposte.

## SECONDA PARTE

1. **[5 punti]** Siano  $[a_0, b_0], [a_1, b_1], \dots, [a_{n-1}, b_{n-1}]$   $n$  intervalli sulla retta reale, con  $a_i < b_i$ . Gli intervalli sono ordinati in base all'estremo sinistro ( $a_0 \leq a_1 \leq \dots \leq a_{n-1}$ ) ma possono sovrapporsi. Il seguente ciclo determina la lunghezza massima di un segmento continuo della retta reale coperto interamente dagli intervalli.

```

Acurr  $\leftarrow$   $a_0$ ; Bcurr  $\leftarrow$   $b_0$ ; max_length  $\leftarrow$  ( $b_0 - a_0$ );
for  $i \leftarrow 1$  to  $n - 1$  do
    if ( $a_i > Bcurr$ ) then
        Acurr  $\leftarrow$   $a_i$ ;
        Bcurr  $\leftarrow$   $b_i$ ;
    else Bcurr  $\leftarrow$   $\max\{Bcurr, b_i\}$ ;
    max_length  $\leftarrow$   $\max\{\text{max\_length}, Bcurr - Acurr\}$ ;
return max_length

```

Trovare un invariante per il ciclo **for**, che specifichi che cosa rappresentano le variabili **Acurr**, **Bcurr**, **max\_length** alla fine di ciascuna iterazione.

2. **[6 punti]** Sia  $T$  un albero binario proprio dove ogni nodo  $v \in T$  contiene un valore binario. Un nodo  $v \in T$  si dice *3-balanced*, se  $|n_0(v) - n_1(v)| \leq 3$ , dove  $n_0(v)$  e  $n_1(v)$  denotano il numero di 0 ( $n_0(v)$ ) e 1 ( $n_1(v)$ ) nel sottoalbero  $T_v$ . Si progetti in pseudocodice un algoritmo ricorsivo **All3Balanced** per determinare se *tutti i nodi di  $T$  sono 3-balanced*, con complessità lineare nel numero di nodi di  $T$ .
3. **[5 punti]** Sia  $G = (V, E)$  un grafo non diretto, non pesato e connesso, con  $n = |V|$  e  $m = |E|$ , dove ogni vertice  $v$  ha un bit  $v.\mathbf{C}$  che vale 1 se  $v$  è un *centro*, e 0 altrimenti. Sia  $C \subset V$  l'insieme dei centri e  $k = |C|$ . Progettare in pseudocodice un algoritmo che dato un tale grafo  $G$  calcoli, per ogni vertice  $v \in V$ , la distanza dal centro più vicino a lui, ovvero  $\min_{s \in C} d(v, s)$ , dove  $d(v, s)$  è il numero di archi nel cammino minimo da  $v$  a  $s$ , memorizzandola in un campo  $v.\mathbf{distC}$ . Analizzare la complessità dell'algoritmo (per avere punteggio massimo deve essere  $O(km)$ ).