

## PRIMA PARTE

1. **[4 punti]** Sia  $A$  un algoritmo ricorsivo. Descrivere brevemente come, usando l'albero della ricorsione, si può dimostrare che la sua complessità al caso pessimo  $t_A(n)$  è  $O(f(n))$ , per una data funzione  $f(n)$ .
2. **[4 punti]** Sia  $T$  un albero binario proprio non vuoto di altezza  $h$  con  $n$  nodi di cui  $m$  sono foglie. Sapendo che  $m \geq h + 1$ :
  - (a) Disegnare un albero  $T$  in cui il bound vale con uguaglianza ( $h = 4, m = 5$ ).
  - (b) Dimostrare che  $n \geq 2h + 1$ , usando, senza dimostrarla, la relazione nota tra numero di nodi interni e foglie.
3. **[4 punti]** Con riferimento alla Mappa:
  - (a) Dare la specifica precisa del metodo `put`.
  - (b) Supponendo di implementare la Mappa tramite lista position-based, indicare la complessità di `put` giustificando la risposta.
4. **[4 punti]** Il seguente algoritmo ordina una sequenza  $S$  di  $n$  interi.

```
S1 ← interi pari di S; S2 ← interi dispari di S;  
Ordina S1 con MergeSort;  
Ordina S2 con QuickSort deterministico;  
Merge(S1, S2; S);
```

Osservando che la separazione di pari e dispari può essere fatta in tempo  $\Theta(n)$ , determinare la complessità al caso pessimo dell'algoritmo esprimendola con  $\Theta(\cdot)$ .

## SECONDA PARTE

1. **[5 punti]** Sia  $G(n)$  una funzione definita come segue:
  - $G(0) = 1$  e  $G(1) = 2$ ;
  - $G(n - 1) + 2G(n - 2)$ , per ogni  $n > 1$ .
  - (a) Trovare i valori di  $G(n)$ , per  $n = 2, 3, 4, 5$ .
  - (b) Dal punto precedente, intuire una formula chiusa per  $G(n)$ , per  $n \geq 0$ , e provarla per induzione.
2. **[5 punti]** Progettare un algoritmo che calcola il  $k$ -esimo elemento più piccolo di un insieme di  $n$  interi distinti, in tempo  $O(n + k \log n)$  e senza ricorrere all'ordinamento.
3. **[6 punti]** Sia  $G = (V, E)$  un grafo che rappresenta una rete sociale con  $n$  vertici ed  $m$  archi. Ogni vertice  $u \in V$  ha un campo  $L_V[u].\text{influencer}$  che vale 1 se  $u$  è un influencer e 0 altrimenti. Un vertice  $x$  non influencer si dice *influenzabile* se esiste un influencer  $y$  e un cammino tra  $x$  e  $y$ . Progettare in pseudocodice un algoritmo **Influenzabili** che conti il numero di vertici influenzabili in  $G$ , e analizzarne la complessità. Per avere punteggio pieno la complessità deve essere  $O(n + m)$ .