

PRIMA PARTE

1. **[3 punti]** Sia A un algoritmo di complessità $\Theta(n^2)$. Può esistere una istanza per cui A esegue $10n$ operazioni? E una per cui ne esegue $\frac{n^3}{20}$? Motivare brevemente le risposte.
2. **[4 punti]** Dimostrare per induzione che in un albero binario proprio T non vuoto, si ha $m = n - m + 1$, dove n è il numero di nodi e m è il numero di foglie di T .
3. **[4 punti]** Si consideri il metodo **insert** di una Priority Queue implementata tramite uno heap P .
 - (a) Descrivere il metodo in pseudocodice.
 - (b) Dimostrare che la complessità è $O(\log n)$, dove n è il numero di entry in P . (Si può usare il valore dell'altezza di P senza dimostrarlo.)
4. **[5 punti]** Sia G un grafo diretto. Rispondere alle seguenti domande
 - (a) Affinché G sia *fortemente connesso* cosa deve valere?
 - (b) Affinché G sia *debolmente connesso* cosa deve valere?
 - (c) Fare un esempio di grafo diretto G con 4 nodi che sia debolmente connesso ma non fortemente connesso.

SECONDA PARTE

1. [6 punti] Si consideri un albero binario proprio T dove ciascun nodo v contiene un intero $v.val \geq 0$. Un nodo $v \in T$ si dice *massimale* se vale la condizione $v.val \geq u.val$ per ogni u antenato di v . Si noti che la radice è sempre massimale.
 - (a) Progettare in pseudocodice un algoritmo ricorsivo `MaximalSet` che, per ogni nodo v , imposta un flag binario $v.maximal$ a 1, se v è massimale, a 0 altrimenti. Alla fine dell'algoritmo il campo `maximal` di tutti i nodi deve risultare impostato.
 - (b) Analizzare la complessità dell'algoritmo progettato per il punto precedente.
2. [5 punti] Progettare un algoritmo che, date due sequenze ordinate S_1, S_2 , ciascuna contenente n interi distinti, determini il numero di elementi $k \in S_1$ per cui S_2 contiene k^2 . L'algoritmo deve avere complessità $\Theta(n)$. (Suggerimento: modificare Merge)
3. [5 punti] Sia $G = (V, E)$ un grafo con $V = \{1, 2, \dots, n\}$ e $E \subseteq \{(i-1, i) : 1 < i \leq n\}$. Ogni vertice $i \in V$ ha un flag binario $f(i)$. Il seguente pseudocodice determina se ogni componente connessa di G ha almeno un vertice con flag 1.

```

prevComponents ← true;
if (f(1) = 1) then currComponent ← true else currComponent ← false;
for i ← 2 to n do
  if ((i-1, i) ∈ E) then
    if (f(i) = 1) then currComponent ← true;
  else
    prevComponents ← prevComponents AND currComponent;
    if (f(i) = 1) then currComponent ← true;
    else currComponent ← false;
return prevComponents AND currComponent

```

Dire che cosa rappresentano `prevComponents` e `currComponent` alla fine di un'arbitraria iterazione i del for. (A titolo d'esempio, per il grafo con $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ e $E = \{(1, 2), (3, 4), (4, 5), (7, 8)\}$ e flag a 1 nei vertici 1,5,6,8, restituisce `true`.)