Complementi di Algoritmi e Strutture Dati

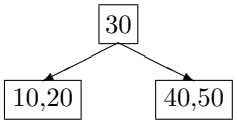
(III anno Laurea Triennale - a.a. 2017/18)

Prova scritta 28 giugno 2018

NB: I punteggi sono indicativi.

Esercizio 1 – Alberi (punti 6)

Dato il seguente 2-3 albero:



1. Inserire un elemento di chiave 25.
2. Sul risultato, inserire un elemento di chiave 5.
3. Sul risultato, inserire un elemento di chiave 32.
4. Sul risultato, inserire un elemento di chiave 38.

Per ogni inserimento, far vedere che cosa succede nell'albero e spiegare perché.

Esercizio 2 – Sorting (Punti 6)

Si consideri il seguente array:

5	12	10	40	30	8	4	2	50

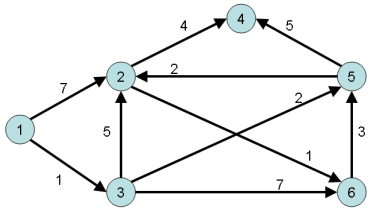
Eseguire la prima fase dell'algoritmo heapsort, cioè quella che trasforma l'array in uno heap a massimo. Si chiede di eseguirla

- PREFERIBILMENTE con la procedura *heapify*.
  Ad ogni passo disegnare tutto l'array come albero ed indicare quali sotto-parti sono già heap.
- IN ALTERNATIVA (ma allora l'esercizio vale un punto in meno) con una serie di chiamate a *insert*. Per ogni chiamata indicare l'elemento che viene inserito e disegnare lo heap in cui viene inserito prima e dopo l'operazione (ovviamente il "dopo" di un inserimento è il "prima" dell'inserimento seguente e non occorre ripetere il disegno).

Ricordare che deve essere uno heap *a massimo*.

Esercizio 3 – Grafi (Punti 7)

Si consideri il seguente grafo orientato e pesato.



Applicando l'algoritmo di Dijkstra, si determinino i pesi dei cammini minimi che collegano il nodo 1 con tutti gli altri nodi. Più precisamente, si completi la seguente tabella:

	1	2	3	4	5	6
0	0	∞	∞	∞	∞	∞
1						
2						
...						

dove ogni riga corrisponde a un'iterazione, e ogni casella contiene: per i nodi per i quali è già stata trovata la distanza definitiva, un simbolo speciale (per esempio -), per gli altri la distanza provvisoria corrente. Si considerino gli adiacenti a un nodo in ordine crescente.

Esercizio 4 – Tecniche algoritmiche (Punti 7)

Si consideri una scacchiera quadrata rappresentata da una matrice $M[1..n, 1..n]$. Scopo del gioco è spostare una pedina dalla casella in alto a sinistra di coordinate $(1, 1)$ alla casella in basso a destra di coordinate (n, n). A ogni mossa la pedina può essere spostata di una posizione verso il basso oppure verso destra (senza uscire dai bordi della scacchiera). Quindi, se si trova in (i, j) potrà essere spostata in $(i + 1, j)$ oppure $(i, j + 1)$, se possibile. Ogni casella $M[i, j]$ contiene un numero; man mano che la pedina si muove, il giocatore accumula il punteggio segnato sulle caselle attraversate, incluse quelle di partenza e di arrivo.
Indichiamo con $P[i, j]$, con $1 \leq i, j \leq n$, il sottoproblema di trovare il massimo punteggio ottenibile spostando la pedina da $(1, 1)$ a (i, j).

1. Si definisca induttivamente $P[i, j]$ giustificando la correttezza della definizione.
2. Si descriva un algoritmo di programmazione dinamica che calcoli $P[n, n]$.
3. Si descriva come ottenere anche la sequenza di mosse corrispondente al massimo punteggio.

Esercizio 5 – NP-completezza (Punti 7)
Ricordiamo i due seguenti problemi NP-completi visti a lezione, entrambi aventi come input un grafo non orientato $G = (V, E)$ e un $k \in \mathbb{N}$:

- INDEPENDENT-SET: G ha un insieme indipendente di dimensione (almeno) k (un *insieme indipendente* è un insieme $V' \subseteq V$ di nodi tale che per ogni coppia di essi non esiste l'arco che li collega)
- VERTEX-COVER: G ha un vertex-cover di dimensione (almeno) k (un *vertex cover* è un insieme $V' \subseteq V$ di nodi tali che ogni arco ha almeno un estremo in V').

1. Potrebbe essere INDEPENDENT-SET $\in$ P e VERTEX-COVER $\notin$ P? (Si giustifichi la risposta.)
2. Si dia una riduzione da VERTEX-COVER a INDEPENDENT-SET.