Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2018/19)

Prova scritta 5 giugno 2019

NB: I punteggi sono indicativi.

Esercizio 1 - Ordinamenti (punti 5) Si consideri il seguente array, sul quale è stata già eseguita la prima fase dell'algoritmo heapsort, cioè l'array è già stato trasformato in uno heap a massimo:

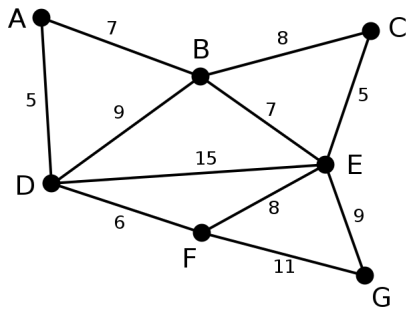
90	80	75	40	20	10	5	25

1. Disegnare la rappresentazione ad albero dello heap.
2. Eseguire la seconda fase dell'algoritmo heapsort, cioè quella che produce l'array ordinato. Ad ogni passo spiegare che cosa è successo, mostrare il nuovo stato di tutto l'array e la rappresentazione ad albero della parte di array che è heap.

Esercizio 2 - Strutture dati (punti 6) Consideriamo 7 elementi con chiavi tutte distinte, per semplicità siano 10, 20, 30, 40, 50, 60, 70 tali chiavi.

1. Consideriamo un albero binario di ricerca (BST). Mostrare un ordine di inserimento che porta all'albero migliore per bilanciamento. Disegnare l'albero BST risultante. Tale sequenza è l'unica?
2. Considerare l'ordine di inserimento crescente, a quale caso corrisponde? Mostrare l'albero BST risultante.
3. Considerare la sequenza crescente ed inserirla invece in un 2-3 albero. Mostrare e spiegare che cosa succede ad ogni elemento inserito.

Esercizio 3 - Grafi (Punti 7) Si esegua, sul seguente grafo:



l'algoritmo di Prim a partire dal nodo A. Inizialmente quindi si avrà `dist`(A)=0, `dist`(B)=∞, `dist`(C)=∞, `dist`(D)=∞, `dist`(E)=∞, `dist(F)`=∞, `dist(G)`=∞. Per ogni iterazione del ciclo while si dia:

- il nodo che viene estratto con la `getMin`
- i nodi per i quali viene modificata `dist` e come
- il minimo albero ricoprente alla fine dell'iterazione, evidenziando chiaramente la parte di albero definitiva.

Non dovete disegnare lo heap.

Esercizio 4 – Design e analisi di algoritmi (Punti 8) Si consideri un array $a[0..n-1]$ contenente valori interi ordinati in senso non decrescente; possono essere presenti valori duplicati.

1. Scrivere un algoritmo ricorsivo di tipo divide-et-impera che, dato in input a e un intero x, restituisce l'indice della *prima* occorrenza di x in a, oppure restituisce n se il valore x non è presente. Per esempio, se $a = [1, 3, 4, 4, 4, 5, 6, 6]$ e $x = 4$, l'algoritmo deve restituire 2.
2. Giustificare la correttezza dell'algoritmo.
3. Valutarne la complessità.
4. Modificare l'algoritmo in modo da ottenere l'*ultima* occorrenza.

Esercizio 5 - NP-completezza (Punti 7) Si consideri la seguente istanza ϕ del problema *3SAT*:

$$(\overline{x_1} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_4)$$

1. Si dia la corrispondente istanza del problema *CLIQUE*, ottenuta attraverso la riduzione vista a lezione.
2. Si dia un'assegnazione di valori di verità che rende vera ϕ e si mostri la corrispondente clique.
3. Come potremmo ottenere in modo semplice una riduzione da *SAT* a *CLIQUE*?