Anno 2020-2021: Esame Finale
23 Febbraio 2021

Nome:

Cognome:

Matricola:

Regole. Ogni risposta corretta somma 1.2 punti, e ogni **cinque** risposte incorrette o non date, somma -1.2. Punteggio massimo: 24/30.

1) É vero che $lg(n!) = \Omega(n \cdot lg(n))$?
   A) No. Ma è vero che $lg(n!) = O(n \cdot lg(n))$.
   B) Si.
   C) No. Ma è vero che $lg(n!) = \Theta(n \cdot lg(n))$
   D) Si, ma solo sotto l'ipotesi che n sia minore di una certa costante.

2) Si consideri la ricorrenza:

$$T(n) = 4 \cdot T(\frac{n}{2}) + n.$$

   Quale delle seguenti è una soluzione?
   A) $\Theta(n \cdot lg(n))$.
   B) $\Theta(n)$.
   C) $\Theta(n^3)$.
   D) $\Theta(n^2)$.

3) Quando diciamo che un algoritmo di ordinamento è stabile, cosa intendiamo?
   A) Che l'algoritmo ha una prestazione ottima per algoritmi basati sul confronto, quindi $O(n \cdot lg(n))$.
   B) Che l'algoritmo ha una prestazione ottima, quindi non è basato su confronti e costa $O(n)$.
   C) Che l'algoritmo usa, al massimo, una quantitá di spazio costante escludendo l'input e l'output.
   D) Che l'algoritmo, in presenza di elementi uguali, dopo l'ordinamento li mantiene nella stessa posizione relativa.

4) Si consideri un array ordinato circolarmente con k spiazzamenti, cioè un array tale che, se tutti gli elementi fossero spostati di esattamente k posizioni, sarebbe ordinato (per esempio, $A = [5, 7, 10, 12, 30, 3, 4]$ è ordinato circolarmente con 2 spiazzamenti). Qual è il costo del problema trovare k?
   A) $O(lg(n))$.
   B) $\Theta(n)$.
   C) $\Theta(n \cdot lg(n))$.
   D) $\Omega(n^2)$.

5) Si consideri un array ordinato A con n elementi, ed il problema di cercare una chiave k in A. Utilizzando la stessa tecnica che abbiamo utilizzato per trovare la complessità minima per il problema dell' ordinamento basato su confronti, possiamo mostrare che questo problema ha complessità:
   A) $\Omega(n^2)$.
   B) $\Omega(n \cdot lg(n))$.
   C) $\Omega(n)$.
   D) $\Omega(lg(n))$.

6) Quale, tra queste, è una ricorrenza che descrive la complessità di *RandomizedQuickSort* nel caso medio, sotto l'ipotesi che tutte le partizioni siano ugualmente probabili?
   A) $T(n) = \Sigma_{k=1}^{n+1}(T(k) + T(n-k+1)) + \Theta(n)$.
   B) $T(n) = \frac{1}{n} \cdot \Sigma_{k=0}^{n-1}(T(k) + T(n-k-1)) + \Theta(n)$.
   C) $T(n) = n \cdot \Sigma_{k=0}^{2 \cdot n}(T(k-1) + T(n-k-1)) + \Theta(n)$.
   D) $T(n) = \frac{2}{n} \cdot \Sigma_{k=1}^{n+1}(T(k-1) + T(n+k-1)) + \Theta(n)$.

7) Se si modifica il codice di *BuildMinHeap* cambiando il verso del ciclo **for** (da 1 a $\lfloor \frac{A.length}{2} \rfloor$), qual è l'effetto?
   A) Il risultato è una procedura incorretta: si otterrebbe, come risultato, un albero binario non necessariamente quasi completo.
   B) Il risultato è una procedura incorretta: la chiamata a *MinHeapify* non necessariamente rispetta le ipotesi.
   C) La procedura è ancora corretta.
   D) La procedura è ancora corretta, ma la complessitá cambia.

8) Sappiamo che è possibile trovare il minimo di un array non ordinato con n chiavi intere in tempo $O(n)$. Il problema di trovare, peró, il k-esimo minimo, ha complessità?
   A) $\Omega(n^k)$.
   B) $\Omega(n \cdot lg(n))$, visto che il k-esimo massimo è l'elemento in posizione $n-k$ nell'array ordinato, e non abbiamo ipotesi aggiuntive per l'ordinamento.
   C) $\Omega(n^2)$.
   D) Nessuna è corretta, perchè esistono algoritmi che risolvono questo problema con complessità, nel caso peggiore, asintoticamente inferiore a tutte le opzioni indicate.

9) Si consideri l'algoritmo *HeapSort* visto in classe. Si puó dimostrare per induzione che una invariante del ciclo è:
   A) Alla fine dell'iterazione i-esima, $A[1, \ldots, n]$ contiene, ordinati, gli n elementi più grandi di A, e $A[1]$ è il massimo di quelli restanti.

B) Alla fine dell'iterazione i-esima, $A[1,\ldots,n-i]$ contiene, ordinati, gli i elementi più grandi di A, e $A[1]$ è il massimo di quelli restanti.
C) Alla fine dell'iterazione i-esima, $A[n-i,\ldots,n]$ contiene, ordinati, gli i elementi più grandi di A, e $A[1]$ è il minimo di quelli restanti.
D) Alla fine dell'iterazione i-esima, $A[n-i,\ldots,n]$ contiene, ordinati, gli i elementi più grandi di A, e $A[1]$ è il massimo di quelli restanti.

10) In una foresta $\mathcal{S}$ di alberi k-ari che rappresentano insiemi disgiunti, supponiamo che $S_1 = \{a, b, f\}$, dove $a.p = f, b.p = f$, e $f.p = f$, e che $S_2 = \{c, d\}$, dove $c.p = d$ e $d.p = d$, e supponiamo che $f.rank = d.rank = 1$. Cosa accade dopo aver eseguito *Union*(c, b), assumendo di utilizzare l'unione per rango, la compressione del percorso, e l'implementazione vista in classe?
A) Si ottiene un solo insieme con tutti gli elementi elencati, tale che $f.rank = 2$, $d.p = b$.
B) Si ottiene un solo insieme con tutti gli elementi elencati, tale che $d.rank = 2$, $f.p = d$.
C) Si ottiene un solo insieme con tutti gli elementi elencati, tale che $f.rank = 2$, $d.p = f$.
D) Nessuna delle altre risposte è corretta.

11) Si consideri l'implementazione basata su liste con unione pesata per insiemi disgiunti, e si consideri il risultato dell'analisi ammortizzata per m operazioni di cui n *MakeSet*. Qual è, nel caso peggiore, il costo medio di ogni operazione?
A) $\Theta(1)$.
B) $\Theta(lg(n))$.
C) $\Theta(lg^2(n))$.
D) $\Theta(n)$.

12) Date le proprietá, senza considerare la proprietá BST, di un albero red-black T:
1. Ogni nodo è rosso o nero;
2. La radice è nera;
3. Ogni foglia (esterna, $T.Nil$) è nera;
4. Se un nodo è rosso, entrambi i suoi figli sono neri;
5. Per ogni nodo, tutti i percorsi semplici da lui alle sue foglie, contengono lo stesso numero di nodi neri.

Quali, tra queste, possono essere violate applicando la normale procedura di inserimento in un BST assumendo che sia garantito che il nodo z inserito sia colorato di rosso?
A) La 2 e la 5.
B) La 1 e la 3.
C) La 1 e la 4.
D) La 2 e la 4.

13) Partendo da un RBT T inizialmente vuoto, si inseriscono usando *RBTreeInsert* le seguenti chiavi, nell'ordine: 14,10,7,5,1. Dopo l'ultimo inserimento, che chiave ha il figlio sinistro della radice e di che colore è?
A) 14, nero.
B) 10, rosso.
C) 5, nero.
D) 7, nero.

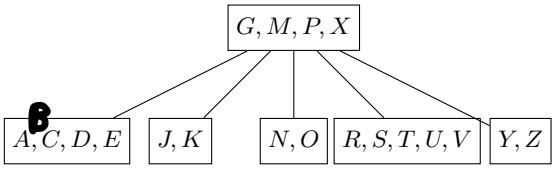

Figure 1. Albero per l'esercizio 15.

14) Si vuole costruire una struttura dati astratta per mantenere chiavi intere e che offra le operazioni *Insert(k,i)*, che inserisce una chiave k nella struttura se, al momento dell'inserimento, non è presente nessuna chiave nell'intervallo $[k-i, k+i]$, e *Delete(k)*, che cancella, se presente, la chiave k. Quale, tra le seguenti, è la struttura dati più adatta, in termini di efficienza asintotica delle operazioni, a supportarla?
A) Una lista concatenata.
B) Un array di interi.
C) Un RBT.
D) Una tabella hash con conflitti risolti via chaining.

15) Si consideri l'albero B con $t = 3$ in Fig. 1, dove si assume l'ordinamento alfabetico. Si consideri, adesso, le procedure *BTreeInsert* e *BTreeSplitChild*, e *BTreeInsertNonFull*, come viste in classe. Dopo l'inserimento, nell'ordine, delle chiavi B, Q e L, quante chiavi ha la radice? Quanto è alto l'albero?
A) La radice ha una sola chiave, P, e l'albero ha altezza 2.
B) La radice ha una sola chiave, P, e l'albero ha altezza 1.
C) La radice ha due chiavi, P e T, e l'albero ha altezza 2.
D) La radice ha due chiavi, P e Q, e l'albero ha altezza 2.

16) In una tabella hash ad indirizzamento aperto, che struttura dati usiamo per il chaining?
A) Liste concatenate, oppure BST, o RBT se le chiavi sono ordinabili.
B) Liste concatenate.
C) Nessuna. L'indirizzamento aperto sostituisce il chaining.
D) Array.

17) Consideriamo la seguente affermazione: *In ogni grafo diretto, aggiungere un nuovo arco diminuisce il numero delle componenti fortemente connesse.* Questa affermazione:
A) E' corretta.
B) E' corretta per tutti i grafi non pesati.
C) E' incorretta: infatti, in ogni grafo diretto, dopo aver aggiunto un nuovo arco il numero delle componenti fortemente connesse rimane lo stesso.

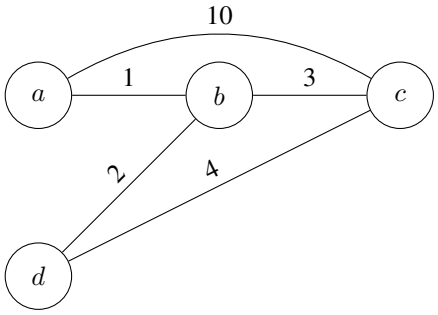

Figure 2. Grafo per l'esercizio 19.

D) E' incorretta: infatti, esiste almeno un grafo diretto tale che, dopo aver aggiunto un nuovo arco, il numero delle componenti fortemente connesse rimane lo stesso.

18) Diciamo che v è un vertice *madre* di un grafo diretto G se e solo se da v è possibile raggiuungere ogni altro vertice in G. Attraverso l'uso dell'algoritmo per trovare le componenti fortemente connesse, possiamo mostrare che il problema di trovare, se esiste, un vertice madre di un grafo diretto $G = (V, E)$ ha complessità:
   A) $O(|V| + |E|)$.
   B) $\Theta(|V|^2)$.
   C) $\Omega(|V|^2)$.
   D) Questo problema, in realtà, non è risolvibile.

19) Considera il problema di trovare il *secondo* albero di copertura minimo di un grafo indiretto pesato G, il grafo in Fig. 2, ed il seguente algoritmo:
   1. Si calcola l'albero di copertura minimo T;
   2. Si elimina dal grafo l'arco di peso minimo contenuto in T, ottenendo un grafo G'
   3. Si calcola l'albero di copertura minimo T' su G', e si restituisce come secondo albero di copertura minimo di G.

   É vero che:
   A) Questo algoritmo è corretto.
   B) Questo algoritmo non è corretto, e il grafo in Fig. 2 costituisce un controesempio.
   C) Questo algoritmo è corretto, ma non efficiente; possiamo ricavare una idea per un algoritmo efficiente dal grafo in Fig. 2.
   D) Questo algoritmo non è corretto, nonostante funzioni sul grafo in Fig. 2.

20) Per una migliore implementazione *MST-Prim* su un grafo indiretto pesato sparso, possiamo usare:
   A) Una lista ordinata per mantere la coda di prioritá Q.
   B) Un array per mantere la coda di prioritá Q.
   C) Una struttura per insiemi disgiunti con unione per rango e compressione del percorso per mantere la coda di prioritá Q.
   D) Una heap binaria per mantenere la coda di priorità Q.

Risposte Corrette Appello 23 Febbraio 2021

Nome: **SOLUZIONI**
Cognome: ***
Data: ***

Domanda 1
A B C D
☐ ■ ☐ ☐

Domanda 2
A B C D
☐ ☐ ☐ ■

Domanda 3
A B C D
☐ ☐ ☐ ■

Domanda 4
A B C D
■ ■ ☐ ☐

Domanda 5
A B C D
☐ ☐ ☐ ■

Domanda 6
A B C D
☐ ■ ☐ ☐

Domanda 7
A B C D
☐ ■ ☐ ☐

Domanda 8
A B C D
☐ ☐ ☐ ■

Domanda 9
A B C D
☐ ☐ ☐ ■

Domanda 10
A B C D
☐ ☐ ■ ☐

Domanda 11
A B C D
☐ ■ ☐ ☐

Domanda 12
A B C D
☐ ☐ ☐ ■

Domanda 13
A B C D
☐ ☐ ■ ☐

Domanda 14
A B C D
☐ ☐ ■ ☐

Domanda 15
A B C D
■ ☐ ☐ ☐

Domanda 16
A B C D
☐ ☐ ■ ■

Domanda 17
A B C D
☐ ☐ ☐ ■

Domanda 18
A B C D
■ ☐ ☐ ■

Domanda 19
A B C D
☐ ■ ☐ ☐

Domanda 20
A B C D
■ ☐ ☐ ■