Università degli Studi dell'Aquila

Prova Scritta di **Algoritmi e Strutture Dati con Laboratorio**

Mercoledì 18 Febbraio 2015 – Prof. Guido Proietti (Modulo di Teoria)

Scrivi i tuoi dati ⟹	Cognome:	Nome:	Matricola:	PUNTI
ESERCIZIO 1	Risposte Esatte:	Risposte Omesse:	Risposte Errate:	

ESERCIZIO 1: Domande a risposta multipla

Premessa: La prova è costituita da 10 domande a risposta multipla. Per ciascuna domanda vengono fornite 4 risposte, di cui soltanto una è corretta. Per rispondere utilizzare la griglia annessa, barrando con una $\times$ la casella corrispondente alla risposta prescelta. È consentito omettere la risposta. In caso di errore, contornare con un cerchietto la $\times$ erroneamente apposta (ovvero, in questo modo $\otimes$) e rifare la $\times$ sulla nuova risposta prescelta. Se una domanda presenta più di una risposta, verrà considerata omessa. Per tutti i quesiti verrà attribuito un identico punteggio, e cioè: risposta esatta 3 punti, risposta omessa 0 punti, risposta sbagliata -1 punto. Il voto finale è dato dalla somma dei punti ottenuti. Se tale somma è negativa, verrà assegnato 0.

1. La dimensione dell'input degli algoritmi studiati per calcolare l'n-esimo numero della sequenza di Fibonacci è $k = \Theta(\log n)$. Qual è la complessità temporale in funzione della dimensione dell'input dell'algoritmo ricorsivo basato sul prodotto tra matrici?
   a) $\omega(k)$ b) $\Theta(n)$ *c) $\Theta(k)$ d) $\Theta(k \log k)$
2. Un algoritmo ha una complessità temporale $O(f(n))$ se:
   a) Il tempo di esecuzione $T(n)$ dell'algoritmo su uno specifico input di dimensione n verifica $T(n) = O(f(n))$
   *b) Il tempo di esecuzione $T(n)$ dell'algoritmo su ogni input di dimensione n verifica $T(n) = O(f(n))$
   c) Il tempo di esecuzione medio $T(n)$ dell'algoritmo su un input di dimensione n verifica $T(n) = O(f(n))$
   d) Nel caso migliore, il tempo di esecuzione $T(n)$ dell'algoritmo su un input di dimensione n verifica $T(n) = O(f(n))$
3. Un algoritmo con complessità temporale $O(1)$ su un input di dimensione n esegue un numero di operazioni:
   *a) costante, indipendente da n b) pari ad 1 c) pari ad n, a meno di costanti additive d) pari a 0
4. A quale algoritmo di ordinamento corrisponde il seguente pseudocodice (il primo elemento dell'array è indicizzato come $A[1]$):

```
Sort(A)
  for (k = 1) to n − 2 do
    m = k + 1
    for (j = k + 2) to n do
      if (A[j] < A[m]) then m = j
    scambia A[m] con A[k + 1]
```

   a) Selection Sort b) Insertion Sort 1 c) Quicksort *d) L'algoritmo non ordina correttamente.
5. L'altezza dell'albero di decisione associato al problema della fusione ordinata di due sequenze ordinate è (suggerimento: lo si ricavi dalla complessità del relativo algoritmo ottimo studiato a lezione):
   *a) $\Omega(n)$ b) $\omega(n \log n)$ c) $O(1)$ d) $\Theta(n!)$
6. Nell'algoritmo QuickSort, scegliendo il perno a caso, il numero di confronti $C(n)$ tra elementi nel caso atteso è:
   a) $\sum_{a=0}^{n-1}[n-1+C(a)+C(n-a-1)]$ b) $\frac{1}{n}[n-1+C(a)+C(n-a-1)]$
   c) $\sum_{a=0}^{n-1}\frac{1}{n}[C(a)+C(n-a-1)]$ * d) $\sum_{a=0}^{n-1}\frac{1}{n}[n-1+C(a)+C(n-a-1)]$
7. Come viene eseguita in un min-heap binomiale l'operazione *increaseKey*(e, Δ) di un elemento e con chiave k?
   a) *decreaseKey*(e, ∞) seguita da *deleteMin*() *b) *delete*(e) seguita da *insert*$(e, k + \Delta)$
   c) *decreaseKey*(e, ∞) seguita da *insert*$(e, k + \Delta)$
   d) Decrementando la chiave e ripristinando la proprietà dell'heap spingendo l'elemento verso il basso tramite ripetuti scambi.
8. Supponiamo di inserire $\sqrt{n}$ elementi in un albero AVL contenente $\log n$ elementi. A quanto sarà pari l'altezza dell'AVL risultante?
   a) $\Theta(\sqrt{n})$ b) $\Theta(\log \log n)$ c) $\Theta(\log(n \cdot \sqrt{n}))$ *d) $\Theta(\log n)$
9. Sia $G = (V, E)$ un grafo *completo* di 6 vertici, in cui i vertici sono numerati da 1 a 6, ed il peso dell'arco non orientato (i, j) è pari al minimo tra i e j. Qual è la distanza tra il vertice 4 e il vertice 5 in G?
   a) 1 *b) 2 c) 3 d) 6
10. Nel problema della gestione di insiemi disgiunti, quale tra le diverse implementazioni proposte garantisce di poter eseguire nel caso peggiore la *Union* in $O(\log n)$ e la *Find* in $O(1)$?
    *a) nessuna b) QuickFind con *union by size* c) QuickUnion con *union by size* d) QuickUnion

Griglia Risposte

	Domanda									
Risposta	1	2	3	4	5	6	7	8	9	10
a										
b										
c										
d										