Prima Prova Parziale di **Algoritmi e Strutture Dati con Laboratorio**
Giovedì 17 Novembre 2022 – Prof. Guido Proietti (Modulo di Teoria)

[redacted]	Cognome:	Nome:	Matricola:	PUNTI
ESERCIZIO 1	Risposte Esatte:	Risposte Omesse:	Risposte Errate:	

ESERCIZIO 1: Domande a risposta multipla

Premessa: Questa parte 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 × la casella corrispondente alla risposta prescelta. È consentito omettere la risposta. In caso di errore, contornare con un cerchietto la × erroneamente apposta (ovvero, in questo modo ⊗) e rifare la × 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 relativo a questa parte è ottenuto sommando i punti ottenuti e normalizzando su base 30. Se tale somma è negativa, verrà assegnato 0.

1. Detto F_n l'n-esimo numero della sequenza di Fibonacci, e detta $\phi = 1,618\ldots$ la sezione aurea, quale delle seguenti relazioni asintotiche è falsa?
   a) $F_n = \Theta(\phi^n)$ *b) $F_n = \omega(\phi^n)$ c) $F_n = O(2^n)$ d) $F_n = \Omega(\phi^n)$
2. Sia dato in input l'array $A = [1, 2, 3, \ldots, n-1, n]$, e si supponga di applicare su di esso gli algoritmi di ordinamento non decrescente InsertionSort2, Mergesort e HeapSort. Quale dei tre algoritmi esegue il minor numero di confronti asintoticamente?
   *a) InsertionSort2 b) Mergesort c) HeapSort d) Sono tutt'e tre equivalenti
3. 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 $T(n) = O(f(n))$
4. Si consideri l'algoritmo di ricerca di un elemento in un insieme non ordinato di n elementi. Quale delle seguenti opzioni descrive in modo preciso il numero di confronti nel caso migliore, peggiore e medio?
   a) $T_{\text{best}}(n) = 1, T_{\text{worst}}(n) = n, T_{\text{avg}}(n) = n/2$ *b) $T_{\text{best}}(n) = 1, T_{\text{worst}}(n) = n, T_{\text{avg}}(n) = (n+1)/2$
   c)$\mathrm{T}_{\text{best}}(n) = O(1), T_{\text{worst}}(n) = n, T_{\text{avg}}(n) = (n+1)/2$ d) $T_{\text{best}}(n) = 1, T_{\text{worst}}(n) = O(n), T_{\text{avg}}(n) = (n+1)/2$
5. Sotto quali ipotesi la soluzione dell'equazione di ricorrenza $T(n) = a \cdot T(n/b) + f(n)$, con $T(1) = \Theta(1)$, a, b costanti non negative, è pari a $T(n) = \Theta(f(n))$?
   a) Se $f(n) = O\left(n^{\log_b a+\epsilon}\right)$, per qualche $\epsilon > 0$, e se vale la condizione di regolarità: $af(n/b) \leq cf(n)$ per qualche $c < 1$ ed n sufficientemente grande
   *b) Se $f(n) = \Omega\left(n^{\log_b a+\epsilon}\right)$, per qualche $\epsilon > 0$, e se vale la condizione di regolarità: $af(n/b) \leq cf(n)$ per qualche $c < 1$ ed n sufficientemente grande
   c) Se $f(n) = \Theta\left(n^{\log_b a}\right)$ d) Se $f(n) = \Omega\left(n^{\log_b a-\epsilon}\right)$, per qualche $\epsilon > 0$
6. Sia $h(n)$ l'altezza dell'albero di decisione associato all'algoritmo Mergesort. Quale delle seguenti relazioni asintotiche è *falsa*:
   a) $h(n) = o(n^2)$ *b) $h(n) = o(n \log n)$ c) $h(n) = \Theta(\log n!)$ d) $h(n) = \Theta(n \log n)$
7. Sia dato un array A di n elementi in cui l'elemento massimo è pari a k. Trasformando gli elementi da ordinare in base $b = \Theta(n)$, quante passate di Bucket Sort sono necessarie all'algoritmo Radix Sort per ordinare A?
   a) $\Theta(n^k)$ b) $\Theta(n \log_k n)$ *c) $\Theta\left(\frac{\log k}{\log n}\right)$ d) $\Theta(\log n)$
8. Quale tra i seguenti algoritmi non è ottimo se applicato al problema descritto?
   a) Mergesort per ordinare una sequenza di n interi arbitrari
   *b) Heapsort per ordinare una sequenza di n interi con valori compresi tra 1 e n^c
   c) Algoritmo di ricerca binaria per cercare un elemento in una sequenza di n interi ordinati
   d) Integer Sort per ordinare una sequenza di n interi con valori compresi tra $n/2$ e $2n$.
9. Quali sono, rispettivamente, i costi per implementare le operazioni di *IncreaseKey*, *DecreaseKey*, e *Merge* in una coda di priorità di n elementi implementata utilizzando un d-heap?
   a) $O(d \log_d n), O(d \log_d n), \Theta(n)$ b) $O(\log_d n), O(\log_d n), O(n)$ *c) $O(d \log_d n), O(\log_d n), \Theta(n)$ d) $O(n), O(\log_d n), \Theta(d \log_d n)$
10. Dato un heap binomiale H di n elementi, quale delle seguenti affermazioni è <u>vera</u>:
    *a) Il grado della radice di ogni albero in H è $O(\log n)$, e il numero di elementi di qualche albero in H è $\Theta(n)$;
    b) Il grado della radice di ogni albero in H è $\Theta(\log n)$, e il numero di elementi di qualche albero in H è $O(\log n)$;
    c) Il grado della radice di ogni albero in H è $O(\log n)$, e il numero di elementi di ogni albero in H è $\Theta(\log n)$;
    d) Il grado della radice di ogni albero in H è $o(\log n)$, e il numero di elementi di ogni albero in H è $\Theta(\log n)$.

Griglia Risposte

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