Quarto compito scritto di ASD, 9 settembre 2019

Punteggio massimo: 9 punti
Sufficienza: 5 punti
Tempo a disposizione: 3 ore se si deve svolgere solo lo scritto; 2.5 ore se si deve svolgere anche il quiz

Risposte scritte con grafia illeggibile saranno cestinate senza nessun tentativo di interpretarle.

Lo pseudo-codice e i suoi commenti devono essere auto-contenuti: non devono comparire elementi (ad esempio parametri delle funzioni, variabili ausiliarie, etc) il cui ruolo e significato non sia perfettamente chiaro. Nelle domande che richiedono di indicare e motivare la complessità, la spiegazione deve essere perfettamente comprensibile ed ogni passaggio logico deve essere chiaramente illustrato e motivato. L'assenza anche di un solo passaggio logico nella spiegazione comporterà una forte penalizzazione sul voto assegnato alla domanda e la mancanza di spiegazione di come si ottiene una data complessità, anche se corretta, comporterà l'assegnazione di 0 punti.

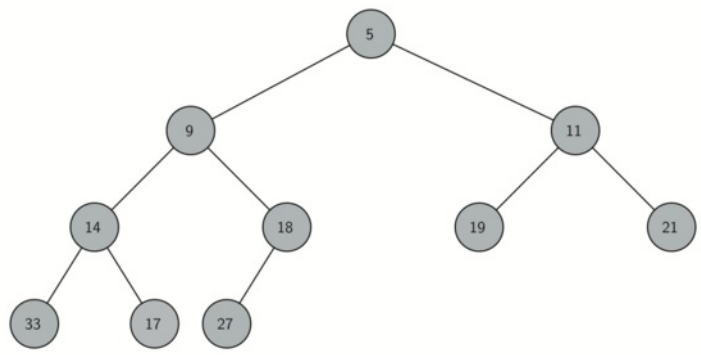
Domanda 1, 2.25 punti: si spieghi la complessità della funzione **quicksort** ricorsiva nel caso **peggiore**, caratterizzando in modo chiaro sotto quali condizioni si verifica il caso peggiore e spiegando - con tutti i passaggi logici necessari a fornire una motivazione chiara e convincente - la ragione per cui si ottiene la complessità indicata.

Domanda 2, 2.25 punti: si spieghi la complessità della funzione **mergesort** ricorsiva nel caso **migliore**, caratterizzando in modo chiaro sotto quali condizioni si verifica il caso migliore e spiegando - con tutti i passaggi logici necessari a fornire una motivazione chiara e convincente - la ragione per cui si ottiene la complessità indicata.

Domanda 3, 1 punto: Si consideri una tabella di hash con liste di collisione che implementa un dizionario D in cui le chiavi sono stringhe di 4 cifre decimali che indichiamo con c1, c2, c3, c4 e i valori sono caratteri.
La tabella di hash ha 10 bucket indicizzati da 0 a 9 e la funzione di hash è **h: (c1+c2+c3+c4) mod 10**.
Ad esempio, se la chiave è `“9031”, h(“9031”) = (9+0+3+1) mod 10 = 3.`
Il dizionario D implementato dalla tabella di hash è inizialmente vuoto. Disegnare la tabella di hash che si ottiene dai seguenti inserimenti effettuati in sequenza come appaiono nel testo, dall'alto verso il basso (prima `insert(D, “0000”, 'a'),` poi `insert(D, “2314”, 'b')`, ecc).

```
insert(D, “0000”, 'a').
insert(D, “2314”, 'b').
insert(D, “3520”, 'c').
insert(D, “4313”, 'd').
insert(D, “4370”, 'e').
insert(D, “0315”, 'f').
insert(D, “9000”, 'g').
insert(D, “1678”, 'h').
insert(D, “2781”, 'm').
insert(D, “2224”, 'n').
```

Domanda 4, 1.5 punti: Si consideri lo heap binario **pq** di tipo min (la radice contiene l'elemento con chiave minima) disegnato a fianco e si spieghino - usando dei disegni che coinvolgano lo heap disegnato come albero, non la sua rappresentazione come array - i passaggi principali dell'algoritmo **deleteMin(pq)**; si spieghi inoltre quanto vale e come si calcola la complessità di deleteMin nel caso peggiore.



Domanda 5, 2 punti: Si scriva lo pseudo-codice con commenti della **visita in ampiezza** (iterativa) e della **visita in profondità ricorsiva** di un grafo non orientato con creazione dell'albero di ricoprimento. Si disegni un grafo non orientato con almeno 8 nodi e 10 archi e si disegnino gli alberi di ricoprimento che si ottengono dalle due visite BFS e DFS, partendo da uno stesso nodo. Si specifichi chiaramente da quale nodo si è partiti.