Algoritmi e Strutture Dati

a.a. 2017/18

Compito del 25/01/2019

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

(30 minuti; ogni esercizio vale 2 punti)

1. Completare la seguente tabella indicando la complessità delle operazioni che si riferiscono a un dizionario di n elementi. Per l'operazione **Predecessore** si assuma di essere sull'elemento x a cui si applica l'operazione.

	Ricerca	**Minimo**	**Predecessore**	**Costruzione**
Tabelle Hash con liste di collisione (caso medio)*				
Min-Heap				
Alberi binari di ricerca bilanciati				

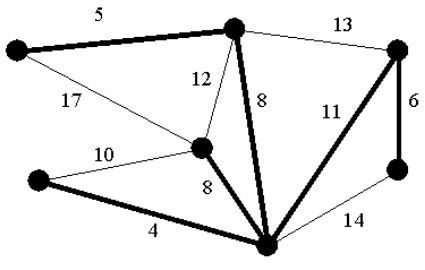
*La Tabella Hash ha dimensione m e il fattore di carico è α.

2. Il Prof. U. N. Known sostiene di aver sviluppato un algoritmo di complessità

$$T(n) = 4T(n/2) + n^2$$

per determinare tutti i cammini minimi in un grafo orientato e pesato, dove n rappresenta il numero di nodi del grafo in ingresso. Si dica se l'algoritmo in questione è preferibile o meno all'algoritmo di Floyd-Warhsall e perché.

3. Si dica, giustificando la risposta, se nel grafo sottostante gli archi indicati in grassetto formano o meno un albero di copertura minimo.



Algoritmi e Strutture Dati

a.a. 2017/18

Compito del 25/01/2019

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

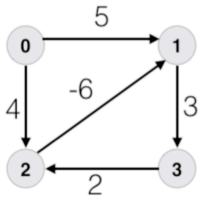
1. (a) Dato un albero binario T, definire l'altezza di un nodo v.
   (b) Sia T un albero binario i cui nodi x hanno campi *left, right* e *key*. Scrivere una **funzione in C efficiente** *altezzaNodi(u, k)* che riceve in input la radice u di un albero binario T e un numero naturale k e restituisce il numero di nodi che hanno altezza k.
   (c) Analizzare la complessità della funzione, indicando eventuali relazioni di ricorrenza.
   Non si possono utilizzare strutture dati ausiliarie.

2. Sia dato un albero binario di ricerca T con n nodi e chiavi naturali. Si scriva tramite pseudocodice una funzione **efficiente** che restituisca un albero binario di ricerca T' contenente tutte e sole le chiavi **pari** di T.
   Qual è la complessità in termini di tempo della funzione proposta? La soluzione proposta è in loco?
   Giustificare le risposte.

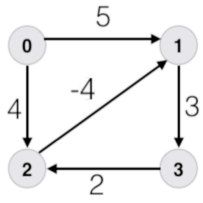
3. Si stabilisca quale problema risolve il seguente algoritmo, che accetta in ingresso un grafo orientato e pesato $G = (V, E)$, la sua funzione peso $w : E \rightarrow \mathrm{R}$, e un vertice $s \in V$:

```
MyAlgorithm(G, w, s)
1.  d[s] = 0
2.  for each u ∈ V[G] \ {s}
3.     d[u] = +∞
4.  for i = 1 to n - 1
5.     for each (u,v) ∈ E[G] do
6.        d[v] = min { d[v], d[u] + w(u,v) }
7.  for each (u,v) ∈ E[G] do
8.     if d[v] != min { d[v], d[u] + w(u,v) } then
9.        print("Yes")
10.    otherwise
11.       print("No")
12. return
```

   Si dimostri la correttezza dell'algoritmo e si determini la sua complessità computazionale. Cosa restituisce l'algoritimo in presenza dei seguente grafi, ponendo $s = 0$?



(a)



(b)

4. Si supponga di voler cambiare una banconota da P euro in monete da a_1, a_2, …, a_k euro. E' possibile? In caso affermativo, qual è il minimo numero di monete da utilizzare per effettuare il cambio? Per esempio, se si volesse cambiare una banconota da 5 euro in monete da 1 e da 2 euro, sarebbero necessarie almeno 3 monete. Se si disponesse soltanto di monete da 2 euro, invece, il cambio non sarebbe possibile.

   Si formuli questo problema come un problema di cammini minimi su grafi. (Suggerimento: si costruisca un grafo con $P + 1$ vertici numerati da 0 a P e si inseriscano archi e pesi in modo tale che i cammini dal vertice 0 al vertice P corrispondano a possibili sequenze di monete da utilizzare per il cambio.)

   A titolo esemplificativo, si consideri il caso di una banconota da $P = 10$ euro e monete da $a_1 = 1$ e $a_2 = 2$ euro. Si costruisca il grafo corrispondente e si utilizzi l'algoritmo di Dijkstra per risolvere il problema