Algoritmi e Strutture Dati

a.a. 2016/17

Compito del 16/01/2018

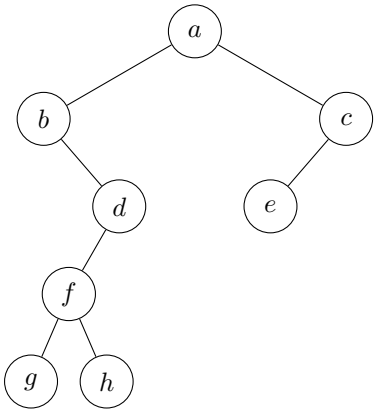
Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

(30 minuti; ogni esercizio vale 2 punti)

1. Dato il seguente albero



Eseguire una visita in preordine, una visita in ordine simmetrico, una visita in postordine e una visita in ampiezza elencando nei quattro casi la sequenza dei nodi incontrati.

2. Per un certo problema sono stati trovati due algoritmi risolutivi (A_1 e A_2) con i seguenti tempi di esecuzione:

A_1: $T(n) = 4n^2 + 7\log n^2$

A_2: $T(n) = 4 \cdot T(n/2) + \log n$

Si dica, giustificando tecnicamente la risposta, quale dei due algoritmi è preferibile per input di dimensione sufficientemente grande.

3. Si completi la tabella sottostante, specificando le complessità degli algoritmi indicati in funzione del numero n dei vertici del grafo:

	Grafo sparso	Grafo denso
Bellman-Ford		
Dijkstra (array)		
Dijkstra (heap)		

Algoritmi e Strutture Dati

a.a. 2016/17

Compito del 16/01/2018

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Scrivere una funzione **efficiente,** di nome simmetrico, che, dato un albero binario, ritorna *1* se l'albero è **speculare**, sia dal punto di vista strutturale che nel contenuto dei nodi, altrimenti ritorna *0*.

   Specificare la chiamata della funzione nel main.
   Analizzare la complessità della funzione.

   Per l'esame da **12 CFU**, deve essere fornita una **funzione C**.
   Per l'esame da **9 CFU**, è sufficiente specificare lo pseudocodice.

2. Sia BST^+ la struttura dati che si ottiene aggiungendo ad ogni nodo x di un albero binario di ricerca un nuovo attributo **diff** che contiene la differenza fra il numero di nodi nel sottoalbero sinistro e quelli nel sottoalbero destro di x.

   Modificando la procedura Tree-insert si definisca una procedura BST^+-insert per l'inserimento di una nuova chiave in un BST^+.
   Il prototipo della procedura è:
   BST^+-insert(TreeConDiff t, NodeConDiff z)

   Si assuma che il nodo z sia così inizializzato:
   $z.p = z.left = z.right = NULL$
   $z.key = k$ (nuova chiave)
   $z.diff = 0$

   Analizzare la complessità della procedura.

3. Si enunci e si dimostri il teorema fondamentale degli alberi di copertura minimi e lo si utilizzi per mostrare la correttezza degli algoritmi di Kruskal e Prim.

4. Sia $G = (V, E)$ un grafo orientato con funzione peso $w : E \rightarrow \mathrm{R}$ e vertici numerati da 1 a n: $V = \{1, 2, \ldots, n\}$. Si scriva un algoritmo che, per ogni coppia di vertici $i, j \in V$, determini la lunghezza del cammino minimo tra i e j i cui vertici intermedi non superino un valore dato k ($1 \leq k \leq n$). Si dimostri la correttezza dell'algoritmo proposto e si determini la sua complessità.