Algoritmi e Strutture Dati

a.a. 2009/10

Compito del 18/6/2010

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. Si noti che l'operazione **Predecessore** assume di aver già raggiunto l'elemento *x* a cui si applica l'operazione.

	Ricerca	**Predecessore**	**Costruzione**
Tabelle Hash con liste di collisione (caso medio)*			
Tabelle Hash a indirizzamento aperto (caso pessimo)*			
Tabelle Hash a indirizzamento aperto (caso medio)*			
Alberi binari di ricerca bilanciati			

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

2. Risolvere le seguenti relazioni di ricorrenza (giustificando le risposte):

   (a) $T(n) = 3 \cdot T(n/2) + n^2$ *(b)* $T(n) = 16 \cdot T(n/4) + n$ *(c)* $T(n) = 4 \cdot T(n/2) + n^2$

3. Si definiscano le classi di complessità P, NP, NPC, e si dimostri che $P \cap NPC \neq \varnothing \Rightarrow P = NP$.

Algoritmi e Strutture Dati

a.a. 2009/10

Compito del 18/6/2010

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Dato un albero binario T scrivere una funzione **efficiente** in C che restituisca 1 se **per ogni** nodo *u* di T vale la seguente proprietà: il sottoalbero sinistro di *u* ha una dimensione **almeno doppia** di quella del sottoalbero destro di *u* (la dimensione è il numero di nodi in esso contenuti), 0 altrimenti.
   Inoltre:
   a. discutere la complessità della soluzione trovata.
   b. dimostrare la correttezza della soluzione proposta (facoltativo).

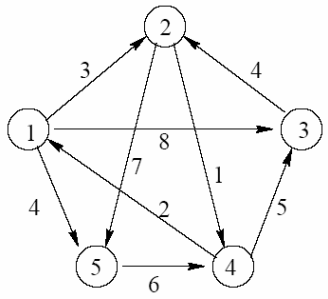
   Si deve utilizzare il seguente tipo per la rappresentazione di un albero binario:

```
typedef struct node{
    int key;
    struct node * left;
    struct node * right;
} * Node;
```

2. Sono dati due alberi binari completamente bilanciati, di radice *r* e *s* rispettivamente, aventi la stessa altezza *h* e dimensione totale (somma dei nodi dei due alberi) *n***.** Le chiavi memorizzate nei nodi di entrambi gli alberi soddisfano la **proprietà di max-heap.**
   Si vogliono fondere i due alberi, ottenendo un unico albero completo a sinistra, di altezza *h+1* e dimensione *n*, che soddisfi la proprietà di max-heap.
   a. Progettare una soluzione di costo in tempo $\Theta(n)$ e in spazio aggiuntivo $\Theta(n)$.
   b. Progettare una soluzione di costo in tempo $O(\log n)$ e spazio aggiuntivo costante.

   La rappresentazione dell'albero binario utilizza esclusivamente i campi **left, right** e **key.**

3. Si scriva l'algoritmo di Dijkstra, si dimostri la sua correttezza, si fornisca la sua complessità computazionale e si simuli accuratamente la sua esecuzione sul seguente grafo (utilizzando il vertice 1 come sorgente):



4. Sia $G=(V,E)$ un grafo non orientato, connesso e pesato avente tutti i pesi distinti. Sia *C* un ciclo di *G* e sia *(u,v)* l'arco in *C* avente peso massimo. Si stabilisca se esiste o meno in *G* un albero di copertura minimo che contiene l'arco *(u,v)*. Potremmo dire lo stesso se i pesi sugli archi non fossero distinti? Giustificare formalmente le risposte.