Algoritmi e Strutture Dati

a.a. 2012/13

Compito del 23/01/2014

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

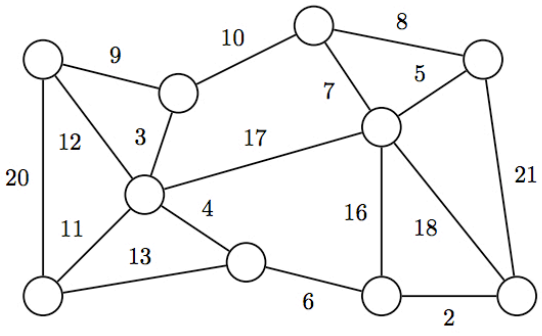
(30 minuti; ogni esercizio vale 2 punti)

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	**Massimo**	**Predecessore**	**Costruzione**
Lista doppia ordinata in senso crescente con sentinella				
Max-Heap				

2. Si definiscano formalmente le relazioni O, Ω, Θ e, **utilizzando le definizioni date e nient'altro**, si dimostri la verità o la falsità di ciascuna delle seguenti affermazioni:
   a) $n \log n = \Theta(n^2)$
   b) $2^n = \Omega(2^{n+k})$, dove k è una costante intera positiva
   c) $2^{n+n} = \mathrm{O}(2^n)$

3. Si determini un albero di copertura minimo nel seguente grafo:



Algoritmi e Strutture Dati

a.a. 2012/13

Compito del 23/01/2014

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Dato un albero binario i cui nodi sono colorati di *bianco* o di *nero*, progettare un algoritmo **efficiente** che calcoli il numero di nodi aventi lo stesso numero di discendenti bianchi e neri. (Un nodo è discendente di se stesso.)
   Inoltre analizzare la complessità di tale algoritmo.

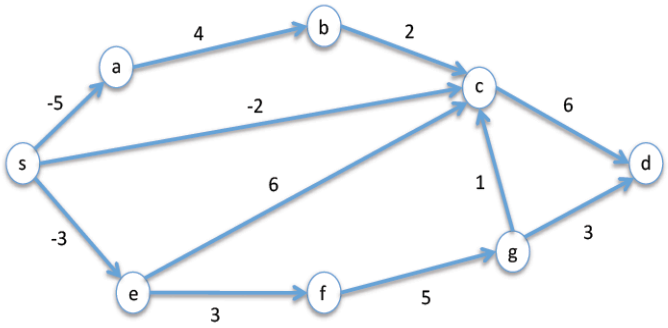
   Il tipo **Node** utilizzato per rappresentare l'albero binario è il seguente:

```
typedef struct node{
        char * colore;
        struct node * left;
        struct node * right;
} * Node;
```

   Per l'esame da **12 CFU**, deve essere fornita **una funzione C**.
   Per l'esame da **9 CFU o 15 CFU**, è sufficiente specificare lo pseudocodice.

2. Si può ordinare un dato insieme di n numeri costruendo un albero binario di ricerca che contiene questi numeri (usando ripetutamente *Tree-Insert* per inserire i numeri uno alla volta) e stampando poi i numeri utilizzando un certo tipo di visita. Scrivere l'algoritmo che realizza questo ordinamento e specificare il tipo di visita effettuata e il relativo algoritmo.
   Quali sono i tempi di esecuzione nel caso peggiore e nel caso migliore per questo algoritmo di ordinamento?

3. Si scriva l'algoritmo di Dijkstra (e le procedure da esso utilizzate) e si simuli accuratamente la sua esecuzione sul seguente grafo:



   La soluzione trovata è corretta?

   In generale: sia $G = (V, E)$ un grafo orientato e pesato, sia $s \in V$ un vertice "sorgente" e si supponga: 1) che gli archi uscenti dal vertice s abbiano peso negativo; 2) che tutti gli altri archi abbiano un peso positivo; 3) che non esistano cicli di peso negativo raggiungibili dalla sorgente. L'algoritmo di Dijkstra funzionerà correttamente su G? In caso affermativo si fornisca una dimostrazione formale, in caso negativo un controesempio.

4. Un cuoco intende creare una nuova ricetta utilizzando un insieme di n ingredienti diversi (numerati da 1 a n). Per ciascuna coppia di ingredienti è possibile associare un coefficiente, compreso tra 0 e 1, che rappresenta la bontà del loro abbinamento (più alto è il numero, più i due ingredienti "stanno bene" insieme). Di seguito è riportato un esempio di possibili abbinamenti tra cinque ingredienti:

	1	**2**	**3**	**4**	**5**
1	1.0	0.6	0.8	0.1	0.0
2	0.6	1.0	0.9	0.0	0.8
3	0.8	0.9	1.0	0.2	0.5
4	0.1	0.0	0.2	1.0	0.8
5	0.0	0.8	0.5	0.8	1.0

Quando questo numero è inferiore a 0.5, i corrispondenti ingredienti producono un sapore sgradevole ed è quindi sconsigliabile combinarli insieme. Il cuoco vuole creare una ricetta che utilizzi il maggior numero di ingredienti, evitando abbinamenti sgradevoli. (Si provi a risolvere il problema utilizzando i valori contenuti nella precedente tabella.)

In generale, si consideri il seguente problema:

RICETTA-CREATIVA
Input: Un insieme $\mathcal{I} = \{ 1, \ldots, n \}$ di ingredienti e una matrice $n \times n$ di "coefficienti di abbinamento".
Output: Il più grande sottoinsieme di $\mathcal{I}$ che non contenga abbinamenti sgradevoli.

Si dimostri che se RICETTA-CREATIVA è risolvibile in tempo polinomiale, allora P = NP.