Algoritmi e Strutture Dati

a.a. 2015/16

Compito del 12/09/2016

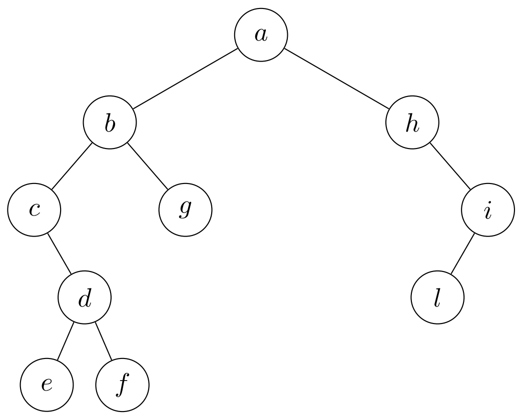
Cognome: ______________________________ Nome: ______________________________

Matricola: ______________________________ E-mail: ______________________________

Parte I

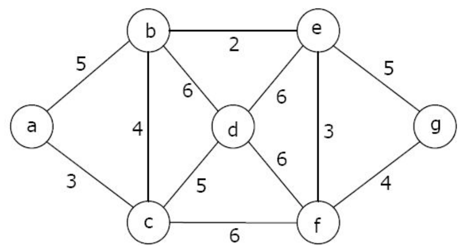
(30 minuti; ogni esercizio vale 2 punti)

1. Dato il seguente albero



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

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



3. Si completi la tabella sottostante, specificando le complessità degli algoritmi indicati in funzione della tipologia di grafo utilizzato:

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

Algoritmi e Strutture Dati

a.a. 2015/16

Compito del 12/09/2016

Cognome: ____________________ Nome: ____________________

Matricola: ____________________ E-mail: ____________________

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Scrivere una funzione **efficiente** *check* che dato un albero binario di ricerca verifica se è soddisfatta la seguente condizione: per ogni intero k, se le chiavi k e k+2 sono nell'albero allora anche la chiave k+1 è nell'albero.

   Analizzare la complessità della funzione. È preferita una soluzione con spazio aggiuntivo O(1).

2. Un collezionista di figurine possiede *K* figurine, non necessariamente distinte. Le figurine vengono stampate con un numero di serie, impresso dal produttore. I numeri di serie sono tutti gli interi compresi tra 1 e *N*. La collezione sarebbe completa se ci fosse almeno un esemplare di tutte le *N* figurine prodotte. Purtroppo la collezione non è completa: sono presenti dei duplicati, mentre alcune figurine mancano del tutto. Il vostro compito è di indicare i numeri di serie delle figurine mancanti.

   Scrivere un algoritmo **efficiente** che dato *N* e l'array $S[1..K]$, ove $S[i]$ è il numero di serie della *i*- esima figurina posseduta, stampa l'elenco dei numeri di serie mancanti. L'array *S* non è ordinato. Ad esempio, se N=10 e S=[1, 4, 2, 7, 10, 2, 1, 4, 3], l'algoritmo deve stampare a video i numeri di serie mancanti 5, 6, 8, 9 (non necessariamente in questo ordine)

   Determinare la complessità dell'algoritmo di cui sopra, in funzione di *N* e *K*. Attenzione: *K* potrebbe essere minore, uguale o maggiore di *N*.

3. Si definisca <u>formalmente</u> la relazione di riducibilità polinomiale tra problemi decisionali ($\leq_P$) e si stabilisca se le seguenti affermazioni sono vere o false:

   1) La relazione $\leq_P$ è transitiva
   2) La relazione $\leq_P$ è riflessiva
   3) Se $\leq_P$ è simmetrica, allora P = NP
   4) Se $\mathcal{P} \leq_P \mathcal{Q}$ e $\mathcal{Q} \in$ P, allora $\mathcal{P} \in$ P
   5) Se $\mathcal{P}, \mathcal{Q} \in$ NPC, allora $\mathcal{P} \leq_P \mathcal{Q}$ se e solo se $\mathcal{Q} \leq_P \mathcal{P}$

   Nel primo caso si fornisca una dimostrazione <u>rigorosa</u>, nel secondo un controesempio.

   Nota: in caso di discussioni poco formali l'esercizio non verrà valutato pienamente.

4. Si stabilisca quale problema risolve il seguente algoritmo, che accetta in ingresso un grafo non orientato $G = (V, E)$:

```
MyAlgorithm(G)
1. A = ∅
2. ordina i vertici di G per grado crescente
3. for each u ∈ V[G] do       /* vertici estratti secondo l'ordine stabilito nel passo 2 */
4.   if MyFunction(G,A,u) then
5.      A = A ∪ {u}
6. return A

MyFunction(G,A,u)
1. for each v ∈ A do
2.    if (u,v) ∈ E[G] then
3.       return FALSE
4. return TRUE
```

Si dimostri la correttezza dell’algoritmo, si determini la sua complessità e si simuli infine la sua esecuzione sul seguente grafo: