Algoritmi e Strutture Dati

a.a. 2014/15

Compito del 1/09/2015

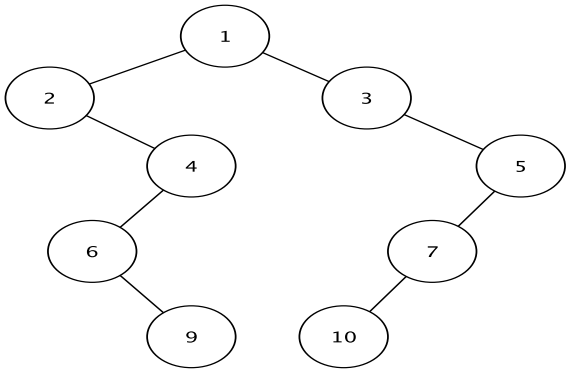
Cognome: _____ Nome: _____

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.

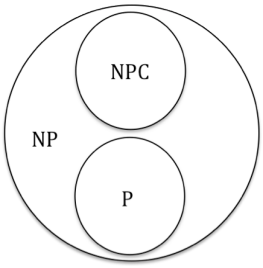
2. Utilizzando la definizione di O, e nient'altro, si stabilisca se le seguente affermazioni sono vere o false:

   a) $f(n) = O(n)$

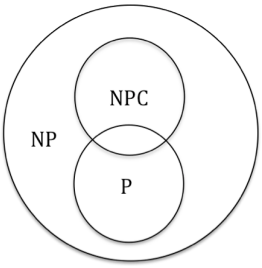
   b) $f(n) = O(n^2)$

   dove $f(n) = (2n - 1) / 2$.

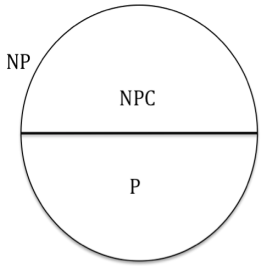
3. Si enunci il teorema fondamentale della NP-completezza e lo si utilizzi per stabilire quale delle seguenti rappresentazioni è ritenuta sicuramente falsa:



(a)



(b)



(c)

Algoritmi e Strutture Dati

a.a. 2014/15

Compito del 01/09/2015

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Supponete di avere un *min-heap* di n elementi, e di cercare il valore **massimo**. In quali posizioni del vettore cercate? Giustificare la risposta
   Scrivere un algoritmo che dato un min-heap non vuoto restituisca il massimo. Calcolare la complessità.

2. Progettare un algoritmo di ordinamento che si comporti come MergeSort, ma che divida ricorsivamente l'array in tre pari anziché due.
   a. Scrivere lo pseucodice della nuova procedura MergeSort3.
   b. Scrivere lo pseudocodice della nuova procedura Fusione3.
   c. Scrivere e risolvere l'equazione di ricorrenza associata.

3. Il seguente algoritmo accetta in ingresso la matrice di adiacenza A di un grafo orientato ed un intero k (con $k \geq 2$), e restituisce un valore Booleano (TRUE / FALSE):

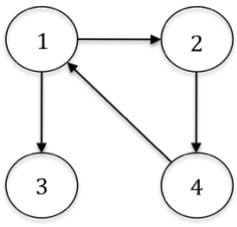
```
MyAlgorithm(A,k)
 1. n = rows(A)              /* determina il numero di vertici del grafo */
 2. let B be the n x n identity matrix
 3. for i = 1 to k - 1
 4.    B = MyFunction(A,B)
 5. for i = 1 to n
 6.    for j = i + 1 to n
 7.       if B[i,j]*A[j,i] ≠ 0 then
 8.          return FALSE
 9. return TRUE
```

   L'algoritmo utilizza la seguente funzione che accetta in ingresso due matrici quadrate (di dimensione identica) e restituisce un'altra matrice quadrata della stessa dimensione:

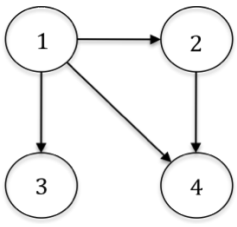
```
MyFunction(X,Y)
 1. n = rows(X)
 2. for i = 1 to n
 3.    for j = 1 to n
 4.       Z[i,j] = 0
 5.       for k = 1 to n
 6.          Z[i,j] = Z[i,j] + X[i,k]*Y[k,j]
 7. return Z
```

   Si calcoli la complessità computazionale complessiva di `MyAlgorithm` e si determini qual è il problema che risolve (ovvero: in quali casi restituisce TRUE? in quali casi FALSE?). (Nota: Nel determinare la complessità si ignori per comodità la complessità delle istruzioni 1 e 2 di MyAlgorithm e dell'istruzione 1 di MyFunction).

   Si simuli inoltre il suo comportamento sui seguenti due grafi, verificando che restituisca il risultato atteso:



G_1



G_2

4. Sia $G = (V, E)$ un grafo non orientato, connesso e pesato. Dato un taglio $(S, V \setminus S)$ di G, sia (u,v) un arco che lo attraversa tale che per tutti gli altri archi (x,y) che attraversano il taglio risulta $w(u,v) \leq w(x,y)$ (arco leggero). Si stabilisca, giustificando formalmente la risposta, se la seguente affermazione è vera o falsa: «Se T è un albero di copertura di G che non contiene (u,v), allora T non è un albero di copertura minimo.»

   Si può affermare la stessa cosa se si assume che tutti i pesi di G siano distinti? Perché?

   Nota: nel fornire le giustificazioni *non* si faccia ricorso al teorema fondamentale degli MST.