Algoritmi e Strutture Dati

a.a. 2011/12

Compito del 28/01/2013

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

(30 minuti; ogni esercizio vale 2 punti)

1. Dato l'insieme delle chiavi {1,4,5,10,16,17,21}, quale è l'altezza minima *hmin* di un albero binario di ricerca che contenga esattamente queste chiavi? E l'altezza massima *hmax*?
   Disegnare 3 alberi binari di ricerca con le chiavi dell'insieme specificato rispettivamente di altezza *hmin*, *hmax* e di un'altezza *h* tale che $hmin < h < hmax$.

2. Si enunci la proprietà fondamentale degli alberi di copertura minimi (definendo accuratamente tutti i termini impiegati) e la si utilizzi per mostrare che se (u,v) è un arco di peso minimo in un grafo non orientato G, allora (u,v) appartiene a un albero di copertura minimo di G.

3. Si stabilisca se la seguente affermazione è vera o falsa, fornendo nel primo caso una dimostrazione, nel secondo un controesempio:

   "Se $P \neq NP$, allora $P \cap NPC = \varnothing$".

Algoritmi e Strutture Dati

a.a. 2011/12

Compito del 28/01/2013

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

1. Dare la definizione di albero binario **completo**. Progettare un algoritmo **efficiente** per stabilire se un albero binario è **completo** e calcolarne la complessità al caso pessimo indicando, e risolvendo, la corrispondente relazione di ricorrenza.

2. Insertion sort può essere espresso come una procedura ricorsiva nel modo seguente: per ordinare A[1..n], si ordina in modo ricorsivo A[1..n-1] e poi si inserisce A[n] nell'array ordinato A[1..n-1]. Scrivere la versione **ricorsiva** dell'insertion sort. Infine scrivere una ricorrenza per il tempo di esecuzione di questa versione ricorsiva e risolverla.

3. Determinare il costo computazionale T(n) del seguente algoritmo, in funzione del parametro $n \geq 0$:

```
MyAlgorithm( int n ) → int
int
  a, i, j;
if ( n > 1 ) then
   a := 0;
   for i := 1 to n-1
      for j := 1 to n-1
         a := a + (i+1)*(j+1);
      endfor
   endfor
   for i := 1 to 16
      a := a + MyAlgorithm(n/4);
   endfor
   return a;
else
   return n-1;
endif
```

4. La rete ferroviaria italiana può essere descritta mediante un grafo orientato pesato $G=(V, E, w)$, dove i vertici rappresentano le stazioni, la presenza di un arco tra due vertici indica l'esistenza di una tratta ferroviaria diretta tra le corrispondenti stazioni e, per ogni arco $(u,v)\in E$, il peso $w(u,v)$ rappresenta la quantità di carburante necessaria per raggiungere la stazione v partendo da u. Si scriva un algoritmo che, dati in ingresso il grafo G, la quantità C di carburante inizialmente presente nel serbatoio della locomotiva di un treno, e due nodi s e d, restituisca TRUE se esiste un cammino che consente al treno di raggiungere la stazione d partendo dalla stazione s, e FALSE in caso contrario. Si discuta della correttezza e della complessità computazionale dell'algoritmo proposto e si simuli accuratamente la sua esecuzione sul seguente grafo, con $C = 15$.