Suggerimenti per risolvere gli esercizi II esonero ASD 8 gennaio 2015

Esercizio 1. Albero AVL.

Soluzione. L'esercizio è simile a uno svolto in aula. Gli alberi AVL sono trattati nella sezione 6.2 del libro di Demetrescu et al.

Esercizio 2. Fornire lo pseudo-codice degli algoritmi di pre-vista, post-visita e visita simmetrica di un albero binario.

Soluzione. La pre-visita di un albero è anche detta visita DFS (depth-first search o visita in profondità).
Lo pseudo-codice richiesto dall'esercizio è fornito nella Fig. 3.14 del libro di Demetrescu et al.
Lo pseudo-codice delle altre due visite è simile, cambia solo la posizione dell'istruzione che visita il nodo r.

Esercizio 3. Scrivere il codice dell'operatore di cancellazione di un elemento da una coda con priorità implementata con heap.

Soluzione. Per la coda con priorità si suggerisce di far riferimento al Capitolo 7 del libro di Bertossi. L'operatore di cancellazione richiesto dall'esercizio è descritto a Pag. 114. La soluzione richiedeva il codice, sottintendendo il codice C++ visto che questo è il linguaggio di programmazione utilizzato nel corso.

Esercizio 4. Si considerino i collegamenti ferroviari di alcuni capoluoghi di provincia dell'Italia Meridionale. Ad ogni collegamento sono associate le informazioni relative a

- tempo di durata del viaggio
- costo del biglietto.

Progettare un algoritmo che, data una citta, stampi le informazioni di tutti i collegamenti diretti con le altre città, cioè dei collegamenti che non hanno fermate intermedie.
Risolvere l'esercizio considerando i seguenti punti:

- definendo il problema in termini di output attesi, input, ipotesi/requisiti di progetto, struttura/e dati che si intende utilizzare;
- delineando l'algoritmo, descritto in pseudo-linguaggio;
- indicando, motivandone la scelta, una possibile realizzazione della/e struttura/e dati .

Nota bene: L'analisi del problema per definire output attesi, input, ipotesi/requisiti di progetto, struttura/e dati che si intende utilizzare deve essere eseguita secondo lo schema fornito dal docente e incluso nel materiale didattico.

Soluzione

1. Analisi

Output: Data una città in input, determinare tutte le città collegate ad essa direttamente, cioè che possono essere raggiunte senza fermate intermedie, e stampare le informazioni di tali collegamenti, cioè tempo di durata del viaggio e costo del biglietto.

Input: i collegamenti ferroviari di alcuni capoluoghi di provincia (città). Ad ogni collegamento sono associate le informazioni relative a

- tempo di durata del viaggio: valore numerico intero che indica i minuti di durata del viaggio

• costo del biglietto: valore numerico che indica il costo in euro e centesimi del viaggio tra le due città.

Struttura dati che si intende utilizzare: Grafo orientato che rappresenta i collegamenti tra due città. Ogni nodo è una città e ogni arco è un collegamento ferroviario e quindi ha associate le informazioni indicate sopra.

Ipotesi/requisiti del progetto:
nessuna di rilievo

2. Pseudo-codice
(Osservazione: data una città in input, l'esercizio si risolve identificando il nodo **n** del grafo **g** corrispondente alla città data in input e identificando la lista dei nodi adiacenti a n attraverso l'operatore adiacenti (n). Per ognuno dei nodi adiacenti, si stamperanno le informazioni relative all'arco che collega n con un nodo adiacente. Per la soluzione dell'esercizio è sufficiente fornire il seguente pseudo-codice).

```
Algoritmo cittaCollegate (Citta c)
Begin
1.    n = nodo (c)
2.    lista_adiacenti = g.adiacenti(n) {identifica la lista dei nodi adiacenti al nodo n nel grafo g}
3.    if (lista_adiacenti è vuota)
4.        print "non ci sono città collegate"
5.    else
6.        p = primo_lista(lista_adiacenti)
7.        while (not fine_lista (p, lista_adiacenti)) do
8.            m = legginodo (p, lista_adiacenti)
9.            a = leggiarco (n, m)
10.           print a.durata, a.costo
11.           p = succlista (p, lista_adiacenti)
12.       end while
13.   end
```