Progetto di Algoritmi e Strutture Dati - A.A. 2015-16

Esercizio 1

L'associazione “Amici dell’algoritmica“ ha necessità di gestire una collezione di valori di tipo numerico ognuno dei quali identifica in maniera univoca un proprio socio.

In ogni momento nuovi soci si possono iscrivere all’associazione così come possono revocare la propria iscrizione. Ogni volta che una persona diventa socio il suo identificativo viene aggiunto alla collezione; analogamente ogni volta che un socio non rinnova la propria iscrizione l’identificativo viene rimosso dalla collezione.

La struttura dati astratta che gestisce gli identificativi dei soci dovrà essere una lista. Si forniscano due implementazioni di tale struttura dati, una che fa uso di un array la cui dimensione varia dinamicamente (struttura dati 1), l’altra che fa uso di variabili di tipo reference per collegare gli elementi della lista (struttura dati 2). Entrambe le strutture dati dovranno anche implementare l’operazione di ricerca.

Ogni operazione, sia essa di ricerca, inserimento o di cancellazione, dovrà essere eseguita in maniera analoga su entrambe le implementazioni. Per ogni operazione occorre individuare sperimentalmente il numero di istruzioni elementari eseguite per portare a termine l’operazione stessa e confrontarlo con la complessità computazionale teorica calcolata per quel tipo di operazione. Inoltre si confronti anche il numero di istruzioni elementari eseguite per ogni operazione di inserimento, ricerca, cancellazione sulle due strutture dati (ovvero si effettui un confronto tra le istruzioni elementari necessari per eseguire un’operazione sulla struttura dati 1 e quelle necessarie per eseguire la stessa operazione sulla struttura dati 2).

<u>Note implementative</u>

Il file che contiene il main si dovrà chiamare Esercizio1.java.
I dati da inserire dovranno essere letti dal file Input1Es1.txt che conterrà righe secondo il seguente formato:

I *numero*
R *numero*
D numero

Le righe che iniziano con I indicano che il *numero* dovrà essere inserito nella struttura dati, le righe che iniziano con R che il *numero* dovrà essere ricercato e le righe che iniziano con D che il *numero* dovrà essere rimosso.
(ovviamente le operazioni di inserimento, ricerca e cancellazione potranno succedersi in maniera random).

Dopo ogni operazione di inserimento, ricerca o cancellazione dovrà essere

stampata sul file Output1Es1.txt il numero di istruzioni elementari eseguite, il tipo di operazione e l'esito secondo questo formato:

istruzioni1 istruzioni2 Tipo di operazione Esito

Es.

293	324	Inserimento	Riuscito
45	78	Ricerca	Non Trovato

Il file Input1Es1.txt sarà messo a disposizione sul sito del corso.

Si esegua il programma anche sul file Input2Es1.txt. Tale file, preparato dallo studente, dovrà avere lo stesso formato del precedente e dovrà contenere almeno un migliaio di righe con una prevalenza di operazioni di inserimento nelle prime 200 righe.

Analogamente si stampino i risultati ottenuti eseguendo il programma con il file Input2Es1.txt sul file Output2Es1.txt .

Nel main dovranno quindi essere svolte le seguenti operazioni:

- lettura del file contenente le operazioni da effettuare
- esecuzione delle operazione in esse elencate sulle due strutture dati
- stampa del numero di istruzioni elementari eseguite sul file di output

I nomi dei file utilizzati dovranno essere inseriti come parametri a riga di comando. Ovvero il programma dovrà essere eseguito nel seguente modo:

```
java Esercizio1 Input1Es1.txt Output1Es1.txt
```

e

```
java Esercizio1 Input2Es1.txt Output2Es1.txt
```

Una volta ottenuti i dati di output fare un'analisi degli stessi confrontando:
- le istruzioni elementari necessarie per ogni tipo di operazione (cioè per esempio confrontare il costo delle operazioni di inserimento con quelle di ricerca)
- le istruzioni elementari eseguite nella struttura dati 1 e nella struttura dati 2
E' consigliabile rappresentare il confronto anche attraverso l'utilizzo di grafici.

Esercizio 2

Il Dizionario TOTO ha la seguente struttura:

a) le chiavi sono interi;
b) il valore di ogni singola chiave k è nell'intervallo[0 e 2^(21));
b) l'informazione relativa alla chiave k è rappresentata da una lista, L(k), costituita da N(k) elementi;
c) l'elemento i-esimo della generica lista L(k) contiene le seguenti informazioni:

chiave k, un codice (cod(i)), di 3 caratteri, riferimento all'elemento successivo di lista.
Si vuole implementare il Dizionario descritto in precedenza con una tabella Hash avente le seguenti caratteristiche:

1) dimensione della tabella: m=16;
2) funzione hash h(k) basata sul metodo della moltiplicazione con valore di C pari a 0.7;
3) collisioni gestite mediante liste di trabocco.

Una volta definita opportunamente la tabella hash, si chiede di implementare gli algoritmi di ricerca, inserimento e cancellazione degli elementi (si ricorda che l'elemento che corrisponde ad una specifica chiave è una lista)

Note implementative

Il file che contiene il main si dovrà chiamare Esercizio2.java.

Inizializzazione del Dizionario:

I dati relativi al dizionario dovranno essere letti da un file Input1Es2.txt avente il seguente formato.

1) numero di chiavi, K, da inserire nella tabella;
2) informazioni relative ad ogni singola chiave k: <k, N(k), cod(1), cod(2), …..cod(N(k))>;

Il valore K è ottenuto utilizzando da un generatore uniforme nell'intervallo [20,30).
Le chiavi vengono prodotte da un generatore uniforme nell'intervallo [0, 2^(21)).
per ogni singola chiave k, N(k) è prodotto da un generatore uniforme nell'intervallo[3-7).
Utilizzare il proprio numero di matricola come seme dei generatori.
I codici di tre caratteri di ogni elementi di lista vanno anch'essi generati utilizzando un appropriato generatore.
Ovviamente i dati precedentemente generati vanno inseriti nel file Input1Es2.txt.

Gestione del Dizionario:

Le operazioni da eseguire per il programma Esercizio2.java dovranno essere lette dal file Input2Es2.txt avente la seguente struttura:

I - <k, N(k), cod(1), cod(2),…..cod(N(k))>
R - k
E - k

le righe che iniziano per I indicano che l'elemento di chiave k deve essere inserito (se non presente nella tabella);
le righe che iniziano per R indicano che si ricerca l'elemento di chiave k;
le righe che iniziano per E indicano che l'elemento di chiave k deve essere eliminato (se presente);

Il file Input2Es2.txt sarà messo a disposizione sul sito del corso.

Il programma Esercizio2 dovrà produrre:

1) il file Output1Es2.txt che contiene la descrizione sintetica della tabella al termine della fase di inizializzazione: numero di chiavi generate, lista delle chiavi generate e, per ogni elemento della tabella identificato dall'indice j, il numero di chiavi la cui trasformazione hash ha prodotto il valore j e la lista delle chiavi concatenate.

2)il file Output2Es2.txt: ogni riga di questo file dovrà contenere il risultato dell'operazione indicata nella stessa riga del file InputEs2.txt. Per le operazioni di inserimento si scriverà semplicemente "Inserimento riuscito", per le operazioni di cancellazione "cancellazione riuscita", per le operazioni di ricerca il risultato della ricerca (chiave e lista dei codici) oppure "chiave non trovata".

I nomi dei file utilizzati dovranno essere inseriti come parametri a riga di comando. Ovvero i programmi dovranno essere eseguiti nel seguente modo:

```
java Esercizio2 Input1Es2.txt Input2Es2.txt Output1Es2.txt Output2Es2.txt
```

Nota:

Il seguente pseudocodice (Java like) può essere utilizzato per generare valori uniformemente distribuiti nell'intervallo [a,b), a e b interi, b>a.

```
long seme;
seme = <numero di matricola>
int a= <estremo inferiore dell'intervallo>;
int b <estremo superiore dell'intervallo>;
int q= b-a;
int k;
Random casuale = new Random(seme);
k= casuale.nextInt(q) +a;
```

Esercizio 3

Gli interventi di manutenzione ordinaria e straordinaria sulla rete idrica di OZ negli ultimi anni sono avvenuti in modo disordinato: raccordi temporanei non rimossi e nessun attenzione alla struttura originale della rete di cui si è ormai persa traccia. Le uniche informazioni attualmente disponibili sono:

- il numero di sorgenti;
- il numero di aree urbane che utilizzano l'acqua;
- la rete idrica attuale che viene rappresentata mediante connessione (tubature) che:
    a) portano l'acqua direttamente dalle sorgenti alle singole aree urbane;

b) portano acqua da una area urbana ad altre aree urbane.

Notare che per le connessioni tra aree urbane (e tra sorgente e aree urbane) viene rappresenta anche la direzione di scorrimento dell'acqua.

Si teme che la struttura della rete idrica di Oz contenga flussi circolari di acqua con conseguenti problemi di stabilità nella fornitura della stessa.
Nell'esempio di Fig. 1., la rete è caratterizzata da un solo flusso circolare (3,4,5,6,7,8) mentre nell'esempio in Fig 2. la rete evidenzia due flussi circolari: (3,4,5,6,7,8) e (4,5,6,7). Notare che uno stesso ciclo può essere descritto utilizzando differenti aree urbane iniziali.

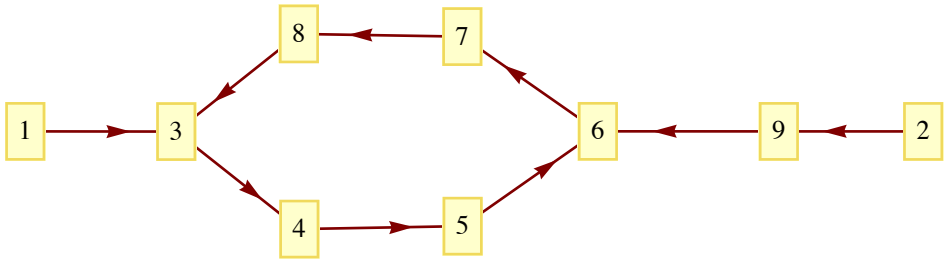


Fig. 1

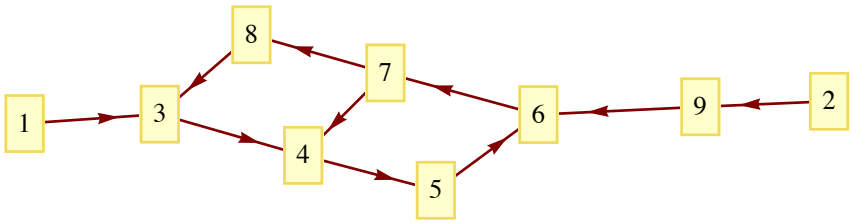


Fig. 2.

Si chiede di:

- Progettare un algoritmo che sia in grado di identificare gli eventuali flussi circolari (anelli) di acqua presenti nell'acquedotto.
Si definisce flusso circolare una sequenza di tubature che partono e terminano nella stessa area urbana senza passare mai due volte dalla stessa area urbana.
Si assume che non esistano tubature che riportano acqua da una area urbana ad una sorgente.

- Implementare l'algoritmo proposto in linguaggio Java rappresentando opportunamente i nodi (sorgenti, aere urbane), le connessioni e la rete idrica.

- Caratterizzare da un punto di vista di costo computazionale l'algoritmo proposto

Dati di Input:

utilizzare il generatore di numeri pseudo-casuali fornito da Java per:

- generare il numero di sorgenti: S = uniforme nell'intervallo [2 ,5); le S sorgenti generate vanno etichettarle con i valori 1,2,..S.
- generare il numero di aree urbane U= uniforme nell'intervallo [10, 20); le U aree urbane vanno etichettate con i valori S+1, S+2,……S+U.

Ovviamente ogni etichetta (numero) sarà preceduta dalle lettera S o dalla lettera U a secondo che si tratti di sorgente o di area urbana.

- generare per ogni singola sorgente s le connessioni dirette verso le aree urbane C(s)= uniforme nell'intervallo [1, max (1, Floor[U/5])); in questo caso anche le aree urbane a cui sono collegate direttamente le sorgenti devono essere scelte casualmente;
- generare, per ogni singola area urbana, u, il numero di connessioni verso altre aree urbane C(u)= uniforme nell'intervallo [1, max(1 e Floor[U/4])); In questo caso anche le aree urbane collegate devono essere scelte casualmente.
Utilizzare come seme dei generatori il proprio numero di matricola.
La rete sarà quindi caratterizzata da una sequenza di coppie del tipo *(in,out)* che rappresentano tutte le connessioni presenti nella rete idrica

Note implementative

Il file che contiene il main si dovrà chiamare Esercizio3.java.

Il programma, potrà essere invocato in due modi. Nel primo modo dovranno essere passati come parametri a riga di comando il numero di matricola e il nome del file di output, mentre nel secondo modo dovranno essere passati come parametri a riga di comando il nome del file di input e il nome del file di output. Il programma potrà cioè essere invocato in entrambi i seguenti modi:

```
java Esercizio3 numero_di_matricola Output1Es3.txt
java Esercizio3 InputEs3.txt Output2Es3.txt
```

Il file di input (InputEs3.txt) avrà il seguente formato:

(in,out)
.
.
.
(in, out)

Le righe indicano tutte le connessioni presenti nella rete idrica. Ogni numero (in o out) nelle connessioni sarà preceduto da S o U a seconda che si tratti di una sorgente o di un'area urbana. Ad esempio, (S1,U8) indica una connessione dalla sorgente 1 all'area urbana 8.
Tale file sarà messo a disposizione sul sito del corso.

L'output del programma, contenuto nel file Output1Es3.txt, sarà costituito da:
a) stampa testuale della rete idrica (lista delle coppie *(in,out)*);
b) numero di cicli presenti nella struttura analizzata;
c) per ogni singolo ciclo (se presente) la lista dei nodi che lo costituiscono con le esplicite connessioni presenti.

Nel caso dell'esempio di Fig. 1. l'output è costituito da:

(S1,U3)
(U3,U4)
(U4,U5)
(U5,U6)
(U6,U7)
(U7,U8)
(U8,U3)
(U9,U6)
(S2,U9)

<1, (3,4,5,6,7,8)>

Nel caso dell'esempio di Fig. 2. l'output è costituito da :

<2, (3,4,5,6,7,8), (4,5,6,7)>

(per brevità si è omessa la stampa della struttura della rete).

Suggerimento per la risoluzione dell'esercizio: Utilizzare algoritmi di visita su grafi, enumerare i cammini e identificare i cicli.

Esercizio 4

Implementare un RB albero (si supponga che le chiavi siano numeri interi) e le operazioni di inserimento e cancellazione.

Il programma dovrà poi leggere in un file di input le chiavi da inserire nell'albero.
Il file di input (InputEs4.txt) conterrà varie righe, una per ogni albero da creare. In ogni riga, separati da uno spazio, saranno presenti i valori delle chiavi da inserire nell'albero. L'ultimo valore di ogni riga dovrà essere prima inserito e poi rimosso dall'albero.

Sapendo che n_i indica il numero di chiavi presenti nella riga i_esima, l'esercizio dovrà stabilire, attraverso risultati sperimentali, se dato un RB albero *a*, ottenuto inserendo nell'albero stesso le prime n_i-1 chiavi, l'albero che si ottiene inserendo l'ultima chiave della riga i_esima e poi rimuovendola è uguale all'albero *a*.

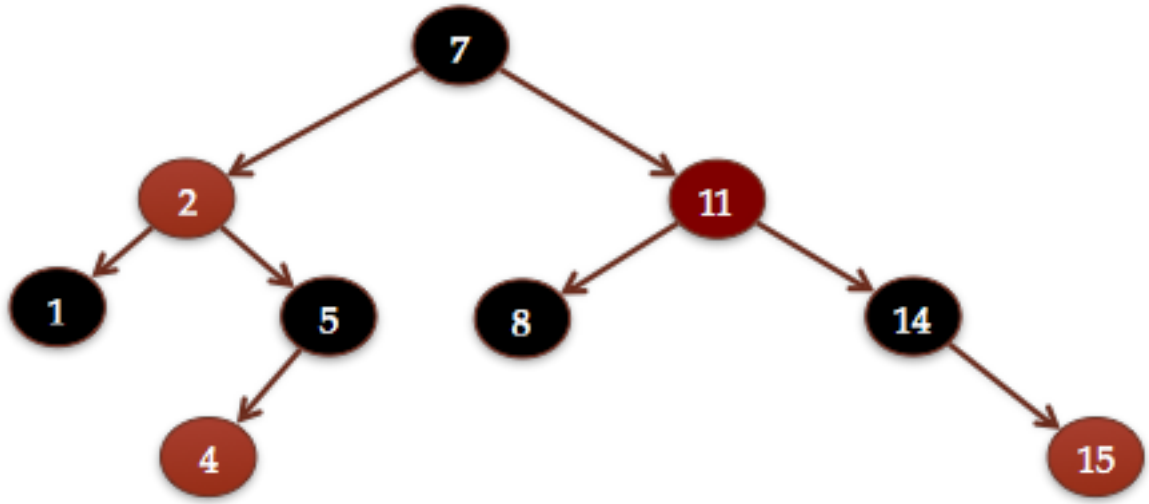
Dato n_1 il numero di chiavi presenti nella prima riga, il programma dovrà inoltre stampare sul file Output2Es4.txt:

- l'albero ottenuto inserendo le prime n_1-1 chiavi della prima riga
- l'albero ottenuto inserendo tutti gli elementi della prima riga
- l'albero ottenuto dal precedente togliendo l'ultimo elemento della prima riga.

Tra la stampa di un albero e la successiva è sufficiente lasciare una riga.
Si noti che si richiede la stampa solo degli alberi ottenuti dai valori della prima riga.

La stampa dell'albero sarà effettuata eseguendo una previsita dello stesso secondo l'esempio che segue. Le chiavi(e il colore del nodo nel quale sono inserite) vanno stampate una per riga; ogni chiave dovrà rientrata di un numero di tab pari al livello del nodo contenente la chiave stessa.

Ad esempio, se l'albero fosse il seguente:



Dovrà essere stampato:

```
B_7
        R_2
                B_1
                        B_nil
                        B_nil
                B_5
                        R_4
                                B_nil
                                B_nil
                        B_nil
        R_11
                B_8
                        B_nil
                        B_nil
                B_14
                        B_nil
                        R_15
                                B_nil
                                B_nil
```

Note implementative

Il file che contiene il main si dovrà chiamare Esercizio4.java.
Il programma Esercizio4 dovrà produrre il file Output1Es4.txt e Output2Es4.txt. Ogni riga del primo file dovrà contenere la stringa “uguale” o “diverso” a seconda che gli alberi ottenuti come descritto in precedenza risultino uguali o diversi.
Il secondo file invece dovrà contenere la stampa di 3 alberi come descritto in precedenza.
Il programma dovrà essere eseguito nel seguente modo:

```
java Esercizio4 InputEs4.txt Output1Es4.txt Output2Es4.txt
```

Tempi e modalità di consegna

I file di input verranno pubblicati al seguente link: http://www.cs.unibo.it/~turrini/DIDATTICA/ALGORITMI/ANNO1516/PROGETTO/
entro fine novembre 2015.

Il progetto dovrà essere svolto singolarmente.

Si precisa che verrà anche valutato l’utilizzo corretto e opportuno del linguaggio Java. I file sorgenti dovranno essere opportunamente commentati.

Inoltre è richiesta la redazione di una relazione che dovrà contenere:

- la descrivere delle scelte progettuali attuate
- la stima della complessità computazionale ove richiesto
- i grafici opportunamente commentati ove richiesto

I sorgenti degli esercizi (file .java) e la documentazione dovrà essere posta in una directory chiamata con il cognome dello studente che ha svolto il progetto. Tale directory andrà poi compressa (formato zip o jar) e spedita a lorenzo.donatiello@unibo.it e a elisa.turrini7@unibo.it entro il 15 gennaio 2016.
Per l’invio del progetto è obbligatorio utilizzare l’indirizzo di email istituzionale.
Entro il 10 febbraio 2016 verrà reso noto chi è stato ammesso alla discussione del progetto, discussione che si terrà a partire dai giorni seguenti.

Si ricorda che è possibile consegnare gli esercizi solo qualora questi siano funzionanti e rispettino le specifiche proposte. Per automatizzare la compilazione, le classi non dovranno essere contenute in pacchetti o suddivise per directory. La compilazione e l'esecuzione avverrà digitando da shell i seguenti comandi:

```
javac *.java

java Esercizio1 Input1Es1.txt Output1Es1.txt

java Esercizio1 Input2Es1.txt Output2Es1.txt

java Esercizio2 Input1Es2.txt Input2Es2.txt Output1Es2.txt
```

Output2Es2.txt

```
java Esercizio3 numero_di_matricola Output1Es3.txt
java Esercizio3 InputEs3.txt Output2Es3.txt

java Esercizio4 InputEs4.txt Output1Es4.txt Output2Es4.txt
```