Progetti Algoritmi e Strutture Dati A.A. 2017-2018

Esercizio 1

Si intende realizzare una coda di priorità mediante un max-heap.

Ogni singolo elemento della struttura è rappresentato da una coppia < categoria, status>; categoria è di tipo carattere e può assumere uno dei seguenti valori: a, b, c o d, mentre status è di tipo intero e può assume uno dei seguenti valori:1,2,3,4,5.

Il max-heap può contenere elementi uguali.

L'ordine di priorità è stabilito nel seguente modo: <x,y> ha priorità più elevata di <z,t> se:

1) x precede z nell'ordine lessicografico;
2) x=z e y<t.

Ovviamente due elementi hanno la stessa priorità se x=z e y=t.

Implementare le seguenti funzioni:

MaxHeapBuild: realizza un albero MaxHeap partendo da una lista di coppie del tipo <categoria, status> con l'ordine di priorità definito in precedenza;

InsertHeap: inserisce, nella posizione corretta, un nodo nell'albero MaxHeap;

RemoveMax: elimina l'elemento con priorità più elevata.

Note implementative

Il file che contiene il main si dovrà chiamare Es1.java
I dati da inserire per creare il max-heap dovranno essere letti dal file InputEs1.txt che conterrà righe secondo il seguente formato:

<categoria, status>

Il file Input2Es1.txt. conterrà invece righe del seguente tipo:

- **R**
- **I** <categoria, status>

Nel caso **–R** si deve eliminare il valore con priorità più elevata (nodo radice) e ricostruire il max-heap.

Nel caso **- I** <categoria, status> si deve modificare MaxHeap inserendo la coppia <categoria, status>.

Dopo ogni operazione di cancellazione del massimo, dovrà essere stampata sul file Output1Es1.txt una riga con le seguenti informazioni:

RemoveMax, valore rimosso ovvero coppia <categoria, status> eliminata, nuovo valore assunto dalla radice dell'albero, numero di swap effettuati per inserire il nuovo massimo.

Dopo ogni operazione di inserimento, dovrà essere stampata sul file Output1Es1.txt una riga con le seguenti informazioni:

InsertHeap, valore inserito ovvero coppia <categoria, status>, valore contenuto nella radice dell'albero, numero di swap effettuati per inserire il nodo nella posizione corretta.

I file Input1Es1.txt e Input2Es1.txt. saranno messi a disposizione sul sito del corso.

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
java Esercizio1 Input2Es1.txt Output1Es1.txt

Una volta ottenuti i dati di output realizzare un grafico che riporti, per ogni operazione (RemoveMax, InsertHeap), il numero di swap (confronti) osservati.

Esercizio 2

Il Dizionario AMICI 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)), rappresentato da un valore intero;
- riferimento all'elemento successivo di lista.

Si vuole implementare il dizionario descritto in precedenza con un **albero binario di ricerca**. Il singolo nodo dell'albero contiene una chiave k e il riferimento alla lista associata alla chiave.

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).

Si richiede inoltre di implementare un algoritmo per la ricerca di un **elemento all'interno di 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 nell'albero;
2) informazioni relative ad ogni singola chiave k: <k, N(k), cod(1), cod(2), ...cod(N(k))>; ovviamente N(k) indica il numero di elementi contenuti nella lista relativa alla chiave k.

Il valore K è ottenuto utilizzando da un generatore uniforme nell'intervallo [20,40).

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 relativi ai singoli elementi di lista vanno anch'essi generati utilizzando un appropriato generatore.

I dati precedentemente generati, insieme al proprio numero di matricola, vanno inseriti nel file Input1Es2.txt. Tale file sarà utilizzato per inizializzare il dizionario.

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
RR-k x

le righe che iniziano per I indicano che l'elemento di chiave k deve essere inserito (se non presente nell'albero);

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);
le righe che iniziano per RR indicano che si deve ricercare all'interno della lista relativa alla chiave k l'elemento x.

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

ll programma Esercizio2 dovrà produrre:

1) il file Output1Es2.txt che contiene la descrizione sintetica del dizionario al termine della fase di inizializzazione: numero di chiavi generate, lista delle chiavi generate e, per ogni elemento dell'albero, la chiave e la lista degli elementi contenuti nella lista che fa riferimento a quella chiave.

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".

Infine per la ricerca di un elemento all'interno di una lista il risultato potrebbe essere 'elemento trovato', 'elemento non trovato', '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 Output2Es3.txt

Una volta ottenuti i dati di output realizzare un grafico che riporti, per ogni operazione (I, R, E, RR), il numero di 'operazioni elementari' necessarie.

Esercizio 3

Utilizzare la classe TreeMap di java per implementare il dizionario descritto nell'esercizio precedente. Stampare i file di output come descritto nell'esercizio precedente utilizzando i medesimi file di input.

Esercizio 4

Il generico nodo del grafo **non orientato** SOCIAL è caratterizzato da una struttura che prevede la presenza di chiavi e liste, in particolare per ogni singolo nodo:

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

- chiave k;
- un codice (cod(i)), rappresentato da un valore intero;
- riferimento all'elemento successivo di lista.

e) la sua rappresentazione è realizzata mediante una matrice di adiacenza.

Inizializzazione del Grafo

I dati relativi al dizionario dovranno essere letti da un file Input1Es4.txt avente il seguente formato

1) il numero di nodi, K;

2) Per ogni singolo nodo del grafo, le informazioni relative ad ogni singola chiave k: <k, N(k), cod(1), cod(2), ...cod(N(k))>; ovviamente N(k) indica il numero di elementi contenuti nella lista relativa alla chiave k.

3) la matrice di adiacenza M, memorizzata per righe.

Il valore K è ottenuto utilizzando un generatore uniforme nell'intervallo [10,20);
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).

I codici relativi ai singoli elementi di lista vanno anch'essi generati utilizzando un appropriato generatore.

La matrice di adiacenza M (KxK), è ottenuta utilizzando un opportuno generatore; l'arco (i,j) esiste con probabilità p.
Quindi il nodo i è collegato al nodo j con probabilità p.

Il valore di p è prodotto da un generatore uniforme nell'intervallo (0,1).

Utilizzare il proprio numero di matricola come seme dei generatori.

I dati precedentemente generati, insieme al proprio numero di matricola, vanno inseriti nel file Input1Es4.txt. Tale file sarà utilizzato per inizializzare il grafo.

Gestione del Grafo

1) Si vuole implementare un algoritmo che verifichi se il grafo SOCIAL è connesso, ovvero trovare le componenti connesse di cui il grafo è composto.

2) data una componente connessa del grafo (o l'intero grafo) verificare in quali liste è presente uno specifico elemento E.

ll programma Esercizio4 dovrà produrre:

a) il file Output1Es4.txt che contiene la descrizione sintetica del grafo al termine della fase di inizializzazione: numero di chiavi generate, lista delle chiavi generate e, per ogni elemento dell'albero, la chiave e la lista degli elementi contenuti nella lista che fa riferimento a quella chiave e la matrice di adiacenza rappresentata per righe.

b) il file Output2Es4.txt contenente il numero di componenti connesse e per ogni componente connessa l'elenco delle chiavi e l'insieme degli elementi distinti (ricavati analizzando opportunamente le liste di riferimento della componente) che fanno riferimento alla stessa.

Tempi e modalità di consegna

I file di input verranno pubblicati entro il 7 dicembre 2017 nel sito

http://www.cs.unibo.it/~donat/progetti-2017

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 descrizione delle scelte progettuali attuate;
- i grafici opportunamente commentati ove richiesto.

I codici sorgenti degli esercizi (file .java), insieme alla documentazione, dovranno essere inseriti in una directory etichettata con il cognome-nome 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 luca.sciullo@unibo.it entro il **28 gennaio 2017.**

Per l’invio del progetto è obbligatorio utilizzare l’indirizzo di email istituzionale.

Entro **l’11 febbraio 2017** sarà reso noto il risultato della valutazione dei progetti e le date previste per la discussione degli stessi.

Si ricorda che è possibile consegnare il progetto solo se il codice prodotto per tutti gli esercizi è corretto e rispetta le specifiche indicate nel testo.

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
java Esercizio1 Input2Es1.txt Output2Es1.txt

java Esercizio2 Input1Es2.txt Output1Es2.txt
java Esercizio2 InputEs2.txt Output2Es2.txt

java Esercizio3 Input1Es3.txt Output1Es3.txt
java Esercizio3 Input2Es3.txt Output2Es3.txt

java Esercizio4 Input1Es4.txt Output1Es4.txt
java Esercizio4 Input2Es4.txt Output2Es4.txt
```