Progetto di Algoritmi e Strutture Dati - A.A. 2016-17

Esercizio 1

L'associazione “Amici della ricerca“ ha necessità di gestire una collezione di attrezzature ognuna caratterizzata da un codice alfanumerico. Ad ogni attrezzatura è associata una lista di luoghi (magazzini, laboratori di ricerca, etc…) nei quali l' attrezzatura è presente. Per ogni luogo deve essere specificato la quantità di attrezzature di quel tipo che son presenti. In uno stesso luogo può essere presente anche più di un'attrezzatura e uno stesso tipo di attrezzatura può essere presente in più luoghi.

Ad esempio, il macchinario X è presente nel magazzino della scuola di scienze, nel museo di biologia e nel dipartimento di farmacia. In quest'ultimo è presente in due esemplari.

La struttura dati astratta che gestisce le attrezzature dovrà essere un albero binario di ricerca. Ad ogni nodo dell'albero dovrà quindi essere associata una lista dei luoghi nei quali l'attrezzatura è presente e il numero di esemplari.

E' possibile in ogni momento aggiungere o rimuovere attrezzature (occorre quindi implementare le operazioni di inserimento e cancellazione). Quando si aggiunge un'attrezzatura si deve specificare il luogo nella quale essa verrà posta, mentre quando si rimuove un'attrezzatura occorre specificare il luogo dal quale si rimuove. Se nel luogo non rimane alcuna attrezzatura di quel tipo si rimuove il luogo dalla lista. Se l'attrezzatura non risulta più essere presente in nessun luogo si rimuove dalla struttura dati.

Si dovrà anche implementare l'operazione di ricerca che presa in input un'attrezzatura e un luogo restituirà il numero di esemplari dell'attrezzatura presenti nel luogo specificato.

Per ogni operazione sia essa di ricerca, inserimento o di cancellazione 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.

Note implementative

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 Id_attrezzatura luogo
C Id_attrezzatura luogo
R Id_attrezzatura luogo
```

dove sta per `I` inserimento, `C` per cancellazione e *`R`* ricerca.

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:

istr Tipo di operazione Esito

Es.

293 Inserimento Riuscito
45 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
java Esercizio1 Input2Es1.txt Output2Es1.txt
```

Una volta ottenuti i dati di output effettuare un'analisi degli stessi confrontando tali valori con il costo computazionale calcolato per ogni tipo di operazione (ricerca, cancellazione, inserimento).
E' consigliabile rappresentare il confronto anche attraverso l'utilizzo di grafici.
Il file `Input2Es1.txt` dovrà essere consegnato assieme ai file sorgenti.

Esercizio 2

Scopo dell'esercizio è verificare sperimentalmente il comportamento della funzione Hash basata sullo metodo della moltiplicazione.

Il test vede essere effettuato considerando la lunghezza della lista di trabocco per ogni singolo elemento della tabella.

A questo scopo si forniscano l'implementazione di una tabella hash nella quale le collisioni sono gestite mediante liste di trabocco che utilizzi una funzione Hash basato sul metodo della moltiplicazione con parametri C=0,15.

Si proceda quindi ad inserire le chiavi nella tabella Hash; al termine dell'inserimento si procederà alla misurazione della lunghezza della lista di trabocco.

Note implementative

Il file che contiene il main si dovrà chiamare Esercizio2.java. I file contenenti le chiavi da inserire saranno messi a disposizione sul sito del corso.

Gli esperimenti dovranno essere di due tipi. Il primo esperimento dovrà confrontare i risultati ottenuti dall'inserimento delle chiavi presenti nel file Input1Es2.txt con quelli ottenuti inserendo le chiavi presenti nel file Input2Es2.txt. Per questi esperimenti la dimensione della tabella hash dovrà essere pari a 16.

Nel secondo esperimento invece si dovrà confrontare i risultati ottenuti dall'inserimento delle chiavi presenti nel file Input3Es2.txt con quelli ottenuti inserendo le chiavi presenti nel file Input4Es2.txt. Per questi esperimenti la dimensione della tabella hash dovrà essere pari a 32.

Il calcolo delle misure descritte in precedenza deve essere effettuato al termine degli inserimenti.

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

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

java Esercizio2 Input3Es2.txt Output3Es2.txt
java Esercizio2 Input4Es2.txt Output4Es2.txt
```

Il formato dei file di output è libero.

Una volta condotti entrambi gli esperimenti si confrontino e si discutano anche i risultati ottenuti nel primo esperimento con quelli ottenuti nel secondo.

Esercizio 3

In questo esercizio si richiede la realizzazione di una struttura ad albero basata su puntatori che implementi un Min Heap.

Nel Min Heap dovranno essere inseriti i valori presenti nel file Input1Es3.txt.

Una volta ultimato l'inserimento si richiede di trasformare l'albero così costruito in un array (che ovviamente deve mantenere le proprietà dello heap) e di stamparlo sul file `Output1Es3.txt` (un elemento per riga)

A questo punto si ordini l'array precedentemente ottenuto e lo si stampi nel file `Output2Es3.txt` (un elemento per riga)

Stimare e discutere la complessità computazionale dell'algoritmo proposto.

Note implementative

Il file che contiene il main si dovrà chiamare Esercizio3.java. Il file Input1Es3.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 Esercizio3 Input1Es3.txt Output1Es3.txt
```

Esercizio 4

Ricerca dei tesori nel mondo sotterraneo di Agarthi.

Il mondo sotterraneo di Agarthi è costituito da grotte collegate da cunicoli. I cunicoli sono unidirezionali: è possibile utilizzare il cunicolo di collegamento solo in una direzione e non è quindi possibile, una volta raggiunta una grotta, utilizzare il medesimo cunicolo in direzione contraria.

Alcune delle grotte sono accessibili solo dall'esterno, e rappresentano i portali di ingresso al mondo sotterraneo. Partendo da uno di questi nodi di ingresso, il cacciatore di tesori accede al mondo sotterraneo.

Alcune delle grotte contengono tesori, non sono collegate mediante archi uscenti con altre grotte ma hanno al loro interno portali di uscita che permettono al cacciatore di tesori di lasciare Agarthi dopo essersi impossessato del tesoro contenuto nella grotta.
Nel suo esplorare Agathi, ad un cacciatore è consentito visitare una caverna al più una volta. Se viene violata questa condizione, al cacciatore è impedito ogni movimento ed è quindi condannato a trascorrere immobile nel mondo sotterraneo il resto della sua esistenza.
Esistono mappe che descrivono il mondo sotterraneo di Agathi. Le mappe sono rappresentate mediante matrici quadrate (NXN) dove N rappresenta il numero totale di caverne. L'elemento (i,j) della matrice è uguale ad 1 se esiste una connessione tra il nodo i e il nodo j, in caso contrario il valore è pari a 0. I nodi che non hanno connessioni in uscita con altri nodi rappresentano caverne che consentono l'uscita dal mondo sotterraneo, mentre l'ingresso al mondo di Agarthi è consentito da nodi che non hanno connessioni in entrata con altri nodi.

Molte mappe sono fasulle, possono, ad esempio, includere percorsi che portano un cacciatore a visitare una stessa caverna più di una volta, oppure non prevedono punti di ingresso e/o uscita da Agarthi.

Quali sono le condizioni da verificare per stabilire se una mappa è autentica?
Scrivere un programma in linguaggio Java che prenda in input una generica mappa nel formato descritto in precedenza e:

a) verifichi che la mappa sia autentica;
b) nel caso di mappa autentica, calcoli il numero totale di percorsi che da un qualunque nodo di ingresso portano ad un generico nodo di uscita. Inoltre, per ogni singolo percorso, sia prodotta la lista dei nodi attraversati;
c) nel caso di mappa fasulla stabilisca se esistono punti di ingresso che permettano al cacciatore di impossessarsi del tesoro contenuto in un eventuale nodo di uscita. Anche in questo caso calcolare il numero totale di cammini e, per ogni singolo cammino, produrre la lista dei nodi attraversati.
d) discutere la complessità computazionale che caratterizza le soluzioni proposte.

Testare il programma sulle tre mappe allegate.

Prima Mappa (`Input1Es4.txt`)

```
0 1 0 0 0
0 0 1 1 0
0 0 0 0 0
0 1 0 0 0
0 1 0 1 0
```

Seconda Mappa (`Input2Es4.txt`)

```
0 1 1 1 0 0
0 0 0 1 0 0
0 0 0 1 0 0
0 1 1 0 0 0
0 1 0 1 0 0
0 0 0 0 0 0
```

Terza Mappa (`Input3Es4.txt`)

```
0 1 0 0 0 0 0 0
```

0 0 1 0 0 0 0 0
0 0 0 1 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0

Note implementative

Il file che contiene il main si dovrà chiamare Esercizio4.java. I file Input1Es4.txt, Input2Es4.txt, Input3Es4.txt saranno messi a disposizione sul sito del corso.

Il file di Output dovrà essere prodotto secondo il seguente formato:

MAPPA AUTENTICA/FASULLA
Elenco dei cammini, uno per riga. Es. 4,0,3,7,2
(si noti come la numerazione dei nodi inizia dal numero 0)

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

```
java Esercizio4 Input1Es4.txt Output1Es4.txt
java Esercizio4 Input2Es4.txt Output2Es4.txt
java Esercizio4 Input3Es4.txt Output3Es4.txt
```

Tempi e modalità di consegna

I file di input verranno pubblicati al seguente link: http://www.cs.unibo.it/~turrini/DIDATTICA/ALGORITMI/ANNO1617/PROGETTO/

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 16 gennaio 2016. Per l'invio del progetto è obbligatorio utilizzare l'indirizzo di email istituzionale.

La discussione del progetto si terrà indicativamente a partire dal giorno 10 febbraio. L'elenco e l'ordine dei progetti risultati sufficienti e ammessi alla discussione verrà spedita via email.

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 Output1Es2.txt
java Esercizio2 Input2Es2.txt Output2Es2.txt
java Esercizio2 Input3Es2.txt Output3Es2.txt
java Esercizio2 Input4Es2.txt Output4Es2.txt

java Esercizio3 Input1Es3.txt Output1Es3.txt

java Esercizio4 Input1Es4.txt Output1Es4.txt
java Esercizio4 Input2Es4.txt Output2Es4.txt
java Esercizio4 Input3Es4.txt Output3Es4.txt
```