Università degli Studi di Camerino
Scuola di Scienze e Tecnologie
Corso di Laurea in Informatica
Corso di Algoritmi e Strutture Dati 2019/2020
Parte di Laboratorio (6 CFU)
Docente: Luca Tesei

Assegnazione del Miniprogetto 2

Descrizione

Il miniprogetto consiste nell':

1. implementare la classe RBTree<E> e la sua classe interna RBTreeNode. La struttura di entrambe le classi è già impostata nel codice fornito come traccia e vanno implementate le parti segnalate dal `// TODO`;
2. eseguire il framework di valutazione degli algoritmi di ordinamento (presente nel codice fornito come traccia) confrontando le prestazioni degli algoritmi forniti. In particolare l'algoritmo RBTreeSort già fornito si basa sulla corretta ed efficiente implementazione della classe RBTree<E> e della classe interna RBTreeNode.

Alberi Rosso-Neri

Gli alberi Rosso-Neri, in inglese Red-Black (da cui RBTree), sono particolari alberi binari di ricerca in cui i nodi sono colorati di rosso o di nero. Le operazioni di inserimento, ricerca e cancellazione in un RBTree possono essere implementate con complessità $O(log_2 n)$, dove n è il numero dei nodi dell'albero.
Per tutti i dettagli e l'implementazione delle funzionalità in pseudocodice si consiglia di consultare il Capitolo 13 del libro di testo T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, "Introduzione agli Algoritmi e Strutture Dati (terza edizione)", McGraw-Hill, 2010 - https://www.mheducation.it/9788838665158-italy-introduzione-agli-algoritmi-e-strutture-dati-3ed
Lo studio individuale di questo capitolo (o di un capitolo equivalente in un altro testo/documento online) costituisce parte integrante del miniprogetto. Nel libro di testo viene suggerito di usare un nodo *sentinella* per facilitare le condizioni al contorno del codice. Non è obbligatorio usare lo stesso approccio nell'implementazione del miniprogetto, ma l'uso ne è consigliato.

Framework di Valutazione

I dati prodotti dal framework di valutazione (file csv, comma-separated values https://it.wikipedia.org/wiki/Comma-separated_values) dovranno essere elaborati con un foglio elettronico (ad esempio Fogli Google, Excel, OpenOffice o LibreOffice) o con un

framework che permette l'uso di tabelle di dati e calcoli statistici (ad esempio R, MatLab o Mathematica) per calcolare, per valori di lunghezza crescenti delle sequenze numeriche generate casualmente e per ogni algoritmo analizzato:

- il minimo (caso ottimo),
- il massimo (caso pessimo),
- la media aritmetica (caso medio) e
- la deviazione standard (caso medio)

dei seguenti valori:

- numero di confronti effettuati dall'algoritmo,
- tempo di esecuzione dell'algoritmo in nanosecondi.

I valori elaborati dovranno poi essere usati per creare dei grafici che permettano di valutare e confrontare le prestazioni dei vari algoritmi. In particolare devono essere generati i seguenti grafici:

- Per ogni algoritmo **a** eseguito nel framework: un grafico a linee sovrapposte che riporta le curve (spezzate) del valore minimo, massimo, medio, medio + deviazione standard, medio - deviazione standard del numero di confronti eseguiti da **a** sulle varie sequenze. Sulle ascisse vanno riportati i valori *n* crescenti della lunghezza delle sequenze analizzate. Nello stesso grafico vanno inoltre riportate le due curve (spezzate) $c_1 * n^2$ e $c_2 * n * log_2 n$ dove i valori delle costanti di scala c_1 e c_2 devono essere scelti opportunamente per permettere la visualizzazione efficace delle varie curve mostrate e il loro andamento asintotico.
- Per ogni algoritmo **a** eseguito nel framework: un grafico a linee sovrapposte che riporta le curve (spezzate) del valore minimo, massimo, medio, medio + deviazione standard, medio - deviazione standard del tempo di esecuzione in nanosecondi di **a** per le varie sequenze. Sulle ascisse vanno riportati i valori *n* crescenti della lunghezza delle sequenze analizzate. Nello stesso grafico vanno inoltre riportate le due curve (spezzate) $c_1 * n^2$ e $c_2 * n * log_2 n$ dove i valori delle costanti di scala c_1 e c_2 devono essere scelti opportunamente per permettere la visualizzazione efficace delle varie curve mostrate e il loro andamento asintotico. **Opzionalmente**: se la linea spezzata del massimo (caso pessimo) presenta molti picchi fuori scala (dovuti, si spera, ad attività della Java Virtual Machine o del sistema operativo che non si possono spegnere) si può ovviare non considerando i valori outliers nel calcolo del massimo. Per la definizione dei valori outliers si può seguire uno degli approcci illustrati a lezione.
- Per ogni coppia non ordinata ($\mathbf{a_1}$,$\mathbf{a_2}$) di algoritmi eseguiti nel framework: un grafico a linee sovrapposte che riporta le curve (spezzate) del valore massimo e medio del numero di confronti eseguiti da $\mathbf{a_1}$ e da $\mathbf{a_2}$ sulle varie sequenze. Sulle ascisse vanno riportati i valori *n* crescenti della lunghezza delle sequenze analizzate. Nello stesso grafico vanno inoltre riportate le due curve (spezzate) $c_1 * n^2$ e $c_2 * n * log_2 n$ dove i valori delle costanti di scala c_1 e c_2 devono essere scelti opportunamente per permettere la visualizzazione efficace delle varie curve mostrate e il loro andamento asintotico.
- Per ogni coppia non ordinata ($\mathbf{a_1}$,$\mathbf{a_2}$) di algoritmi eseguiti nel framework: un grafico a linee sovrapposte che riporta le curve (spezzate) del valore massimo e medio del tempo di esecuzione in nanosecondi di $\mathbf{a_1}$ e $\mathbf{a_2}$ per le varie sequenze. Sulle ascisse vanno riportati i valori *n* crescenti della lunghezza delle sequenze analizzate. Nello

stesso grafico vanno inoltre riportate le due curve (spezzate) $c_1 * n^2$ e $c_2 * n * log_2 n$ dove i valori delle costanti di scala c_1 e c_2 devono essere scelti opportunamente per permettere la visualizzazione efficace delle varie curve mostrate e il loro andamento asintotico. Anche in questo caso, **opzionalmente**, i valori del massimo possono essere smussati eliminando gli outliers.

Tutti i grafici prodotti, corredati da una breve descrizione testuale, devono essere riportati in un documento in formato PDF.

Traccia e Implementazione

La traccia del codice è fornita come progetto Maven (Apache Maven: https://maven.apache.org/) in Eclipse. La versione del compilatore Java definita nel progetto è la 1.8 (Java 8).

Nell'implementazione:

- **non si possono usare versioni del compilatore superiori a 1.8**;
- **è vietato l'uso di lambda-espressioni** e di funzionalità avanzate di Java 8 o superiore (es. stream, ecc…);
- **è obbligatorio usare il set di caratteri utf8** per la codifica dei file del codice sorgente.

Sono definiti per il progetto molti metodi di test realizzati con JUnit 5 (https://junit.org/junit5/docs/current/user-guide/). I metodi sono definiti per la classe da implementare nella cartella `/src/test/java/it/unicam/asdl1920/mp2` all'interno del progetto. I test definiti per l'inserzione e la rimozione di elementi sono visualizzabili nelle note in PDF fornite nella cartella `/src/main/resources/TestCases`.
L'implementazione dovrebbe passare tutti i test forniti.

Esecuzione del Framework di Valutazione

Per una corretta esecuzione del framework di valutazione si devono evitare tutte le possibili interferenze di altri processi in esecuzione nel computer che si sta utilizzando. Quindi si consiglia di:

- chiudere tutte le applicazioni e tutti i servizi del sistema operativo non necessari;
- staccare/disattivare la rete;
- assicurarsi di aver implementato correttamente la classe RBTree<E> e la sua classe interna RBTreeNode. Se ci sono dei problemi con l'implementazione di queste classi e quindi l'esecuzione del framework lancia eccezioni, è possibile comunque generare dei dati per i grafici escludendo però RBTreeSort dagli algoritmi analizzati. Ciò comporta un abbassamento del voto, ma è meglio di non presentare nessun grafico. Per escludere RBTreeSort si deve commentare la `linea 53` del file `SortingAlgorithmEvaluationFramework.java`;
- eseguire la classe main del framework da riga di comando invece che dall'interno di Eclipse. Per far questo, salvare tutto e chiudere Eclipse. Poi aprire una finestra

terminale (dipende dal sistema operativo, ad esempio su Windows bisogna eseguire cmd.exe oppure Prompt dei comandi). Eseguire:

```
> cd <path del workspace>/asdl1920mp2-traccia/target/classes + INVIO
```

sostituendo a <path del workspace> il percorso della cartella workspace di Eclipse (oppure il percorso della cartella dove si trova il progetto se non è nel workspace di Eclipse) e poi eseguire:

```
> java -cp . it.unicam.cs.asdl1920.mp2.SortingAlgorithmEvaluationFramework CSVData
```

in questo modo verranno generati i due file con i dati `evalfram.csv` e `sequences.csv` all'interno della cartella

```
<path del workspace>/asdl1920mp2-traccia/target/classes/CSVData/
```

si può indicare un'altra cartella mettendone il percorso nella linea di comando al posto di `CSVData`. Se non si indica niente i due file vengono generati nella cartella corrente. Il file `sequences.csv` è solo per la verifica delle sequenze generate e non deve essere utilizzato per generare i grafici.

Modalità di Download e Consegna

Per il download (si assume l'uso di Eclipse, se si usano altri IDE i passi sono più o meno analoghi):

- scaricare la traccia in formato .zip del progetto maven/eclipse fornito in allegato al post di assegnazione del miniprogetto in Google Classroom;
- scompattare lo .zip in una cartella di propria scelta (la scelta migliore è dentro la cartella del workspace della propria installazione di eclipse);
- da eclipse scegliere File -> Import… poi Selezionare Maven->Existing Maven Projects poi selezionare (tasto Browse…) la cartella scompattata poi dare Finish;
- a questo punto si ha il project in Eclipse e si possono aprire i sorgenti e modificarli e si possono lanciare i test JUnit di ogni classe.

Vanno implementati tutti i metodi richiesti (segnalati con commenti della forma // TODO testo). La specifica precisa delle API è data con commenti javadoc del codice. Non è consentito:

- aggiungere classi pubbliche;
- modificare la firma dei metodi già specificati nella traccia;
- modificare le variabili istanza già specificate nella traccia.

E' consentito:

- aggiungere classi interne private per fini di implementazione;
- aggiungere metodi privati per fini di implementazione;
- aggiungere variabili istanza private per fini di implementazione.

I test forniti possono essere lanciati per controllare che l'implementazione sia corretta.

Nel file sorgente di ogni classe implementata, nel commento javadoc della classe, modificare il campo @author come indicato: “INSERIRE NOME E COGNOME DELLO STUDENTE - INSERIRE ANCHE L'EMAIL xxxx@studenti.unicam.it“.

Per la consegna:
Creare una cartella con il seguente nome (lettere maiuscole):

ASDL1920-NOME-COGNOME-MP2

ad esempio **ASDL1920-MARIO-ROSSI-MP2**. Nel caso di più nomi/cognomi usare solo il primo nome e il primo cognome. Nel caso di lettere accentate nel nome/cognome usare le corrispondenti lettere non accentate (maiuscole). Nel caso di apostrofi o altri segni nel nome/cognome ometterli. Nel caso di particelle nel nome o nel cognome, ad esempio De Rossi o De’ Rossi, attaccarle (DEROSSI).

All’interno di questa cartella copiare il file sorgente java modificato con la propria implementazione:

- `RBTree.java`

che si trova nella cartella

`src/main/java/it/unicam/asdl1920/mp2`

all’interno della cartella del progetto.
Aggiungere, inoltre, alla cartella i seguenti file:

- `evalfram.csv`, prodotto utilizzando il framework di valutazione e utilizzato come dati grezzi per generare i grafici
- `sequences.csv`, prodotto utilizzando il framework di valutazione
- il file utilizzato per rielaborare i dati e generare i grafici: se si è usato Excel mettere il file `.xlsx`, se si è usato Fogli Google, OpenOffice o LibreOffice mettere il file `.ods`, se si è utlizzato R, Mathlab o Mathematica mettere i relativi file di dati (`csv`) e i file contenenti gli script utilizzati per la rielaborazione dei dati e la generazione dei grafici
- un file dal nome **ASDL1920-NOME-COGNOME-MP2-GRAFICI.pdf** contenente i grafici generati, ognuno corredato da una breve descrizione testuale.

Comprimere la cartella in formato .zip (non rar o altro, solo zip) e chiamare l’archivio

ASDL1920-NOME-COGNOME-MP2.zip

ATTENZIONE: se i passaggi descritti per la consegna non vengono seguiti ****precisamente**** (ad esempio nome della cartella sbagliato, contenuto dello zip diverso da quello indicato, formato non zip ecc.) lo studente **perderà automaticamente 3 punti nel voto del miniprogetto**.
Consegnare il file **ASDL1920-NOME-COGNOME-MP2.zip** tramite Google Classroom (usare la funzione consegna associata al post di assegnazione del miniprogetto) entro la data di **scadenza**, cioè **Venerdì 20 Dicembre 2019 ore 23.59**.

Valutazione

La valutazione si baserà sui seguenti criteri, in ordine decrescente di importanza:

0. Codice scritto individualmente. Nel caso di conclamato “plagio” il miniproject di tutti i “plagi” verrà decurtato di 7 punti.

1. Numero di test JUnit superati. Nel caso in cui il codice vada in ciclo durante l’esecuzione di un test, il test non riesce a fallire, ma questa situazione viene valutata in maniera peggiore del fallimento del test. Evitare in ogni modo di consegnare se si presenta questa eventualità, piuttosto non implementare il metodo corrispondente (provocando quindi il fallimento del test).
2. Correttezza e qualità dei grafici generati a partire dai dati del framework di valutazione.
3. Codice chiaro e ben commentato.
4. Scelta di strutture dati e implementazione efficienti sia dal punto di vista del tempo di esecuzione che dello spazio richiesto.

Il voto assegnato al miniprogetto verrà comunicato tramite Google Classroom. Il voto sarà espresso in 30esimi e peserà per il 33% del voto finale ottenuto con i miniprogetti. Il miniprogetto 1 peserà per il 34% e il miniprogetto 3 peserà per il restante 33%.