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 3

Descrizione

Il miniprogetto consiste nell':

1. implementare le due classi `LCSSolver` e `AllLCSsSolver` utilizzando la tecnica della programmazione dinamica. La struttura delle classi è già impostata nel codice fornito come traccia e vanno implementate le parti segnalate dal `// TODO`;
2. eseguire il framework di valutazione `LCSEvaluationFramework` (presente nel codice fornito come traccia) per valutare le prestazioni dell'algoritmo implementato nella classe `LCSSolver`.

Il problema LCS

Date due sequenze, il problema della ricerca della più lunga sottosequenza comune (Longest Common Subsequence, acronimo LCS) è definito come la ricerca di una sequenza che sia sotto-sequenza di entrambe le sequenze di input e la cui lunghezza sia massima. Utilizzando un approccio "forza bruta" o un approccio ricorsivo classico (top-down) il problema si risolve in tempo esponenziale. Tuttavia, tramite la tecnica della programmazione dinamica, è possibile ottenere una soluzione in tempo e spazio $\Theta(mn)$, dove m ed n sono le lunghezze delle due sequenze di input.
Per i dettagli della definizione e l'implementazione delle funzionalità in pseudocodice si consiglia di consultare il Capitolo 15 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.
Per implementare la classe `LCSSolver` si può usare la soluzione proposta nel libro di testo, la quale permette di calcolare una sola soluzione, cioè **una certa sottosequenza comune di lunghezza massima**. Questa soluzione viene individuata a partire dall'ordine in cui si effettua la ricostruzione della sequenza a partire dalla matrice che è stata calcolata. Ciò può essere fatto tramite la memorizzazione di informazioni utili allo scopo in un'altra matrice "gemella" (come proposto nel libro) o tramite la chiamata di un metodo ricorsivo detto di

“traceback” che ricostruisce la soluzione navigando la matrice calcolata in maniera opportuna (nel libro viene proposto un esercizio che fa questo). **Attenzione**: i test JUnit che controllano il codice della classe `LCSSolver` si basano sull’ordine di ricostruzione proposto nel libro di testo, **quindi per passare i test bisogna assicurarsi che il proprio metodo di ricostruzione della soluzione segua lo stesso ordine**. E’ infatti possibile che seguendo un’altro ordine si ottenga sempre una soluzione corretta, ma diversa da quella controllata dal test.

Nel miniprogetto, **oltre al calcolo di una sola soluzione**, viene richiesto anche di trovare **tutte le sottosequenze comuni di lunghezza massima**. La soluzione di questo problema, da proporre nella classe `AllLCSsSolver`, si può ricavare dalla soluzione che calcola una sola sottosequenza ragionando sempre in maniera tabellare ma considerando la costruzione di un insieme di sottosequenze invece che di una sola sequenza.

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 semplificare** questa analisi il framework applica l’algoritmo implementato nella classe `LCSSolver` a coppie di stringhe che hanno la stessa lunghezza n, per cui ci si aspetta che il tempo di esecuzione sia $\Theta(n^2)$, corrispondente a $\Theta(mn)$ quando $m = n$. I dati devono essere usati per calcolare, per valori di lunghezza crescenti di coppie di sequenze (stringhe) generate casualmente:

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

del seguente valore:

- tempo di esecuzione dell’algoritmo implementato nella classe `LCSSolver` in nanosecondi.

I dati elaborati dovranno poi essere usati per creare un grafico che permetta di valutare le prestazioni dell’algoritmo implementato. In particolare deve essere generato 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 dell’algoritmo implementato nella classe `LCSSolver` per le varie varie coppie di sequenze. Sulle ascisse vanno riportati i valori n crescenti della lunghezza delle due sequenze analizzate. Nello stesso grafico va inoltre riportata la curva (spezzata) $c_1 * n^2$ dove il valore della costante di scala c_1 deve essere scelto 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.

Il grafico prodotto, corredato da una breve descrizione testuale, deve essere riportato 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 diversi 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/cs/asdl1920/mp3` all'interno del progetto. 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 (usando la programmazione dinamica) la classe `LCSSolver`
- 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>/asdl1920mp3-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.mp3.LCSEvaluationFramework CSVData
  ```

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

`<path del workspace>/asdl1920mp3-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-MP3

ad esempio **ASDL1920-MARIO-ROSSI-MP3**. 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:

- `LCSSolver.java`
- `AllLCSsSolver.java`

che si trova nella cartella

`src/main/java/it/unicam/cs/asdl1920/mp3`

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-MP3-GRAFICO.pdf** contenente il grafico generato 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-MP3.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-MP3.zip** tramite Google Classroom (usare la funzione consegna associata al post di assegnazione del miniprogetto) entro la data di **scadenza**, cioè **Venerdì 17 Gennaio 2020 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 si ha che il test non riesce a fallire, ma questa situazione verrà 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 del grafico generato 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 2 peserà per il restante 33%.