Middle East Technical University - Department of Computer Engineering

CENG 443

Introduction to Object Oriented Language and Systems

Fall '2022-2023

Homework 3 - TATEC

Due Date: 10 January 2023, Friday, 23:55
Late Submission Policy will be explained below

Objectives

The objective of this assignment is to familiarize you with the java 8 streams. Your task is to implement the TATEC system that is used for assigning students to the Technical-Electives on CENG curriculum. In this assignment you are obligated to use Java 8 Stream API. Traditional implementations will be penalized. Specifications of the TATEC is given Section 1.
Keywords: Java Streams, TATEC.

1 Specifications

As discussed above you are expected to implement the TATEC System. The formal explanation is as follows. Given a set of course id and capacity pairs C (Equation 1.). Capacities can increase depending on the assignment algorithm.

$$C = \{\{cId_1, cCap_1\}, \{cId_1, cCap_2\}, ..., \{cId_n, cCap_n\}\} \tag{1}$$

And set of students S (Equation 2).

$$S = \{\{sId_1, T_1\}, \{sId_2, T_2\}, ...\{sId_m, T_m\}\} \tag{2}$$

Each student has a set of tokens $T_s = \{t_1, t_2, ..., t_n\}$ each corresponding to a course id, and these are restricted as shown in Equation 3.

$$\sum_{i=1}^{i=n} t_i = 100 \qquad t_i \in \mathbb{Z}^+ \tag{3}$$

Assign students S to the courses. Formally, generate set A_i for each $c_i \in C$ where $A_i = \{s_x, s_y, s_z, ...\}$. After the assignment, calculate the average unhappiness $\overline{U}$ (equation 5). **The algorithm should minimize this value** while assigning students to the courses.

$$\sum_{j=1}^{j=n} cCap_i \geq m \cdot n \tag{4}$$

It is guaranteed that total course capacity is greater than the number of courses multiplied by the number of students (Equation 4).

$$\begin{aligned}\overline{U} &= \frac{1}{m}\sum_{i=1}^{i=m} U(s_i) \\ U(s_i) &= \sum_{j=1}^{j=n} f(t_j, s_i) \\ f(t_j, s_i) &= \chi(s_i, j)\left(\frac{-100}{h}\ln(1-\frac{t_j}{100})\right)\end{aligned} \tag{5}$$

$$\chi(s_i, j) = \begin{cases} 1 & s_i \notin A_j \\ 0 & otherwise \end{cases} \tag{6}$$

Info: We have shown the $f(t_j, s_i)$ function (without the $\chi(s_i, j)$) on a graphing calculator which can bee seen here.

h is a free parameter that will be provided by the user. h will be between 1 and 9 included. Additionally, output of $f(t_j, s_i)$ should not exceed 100, clamp the results to 100 after the calculation.

Additionally, $U(s_i)$ **is squared** if a student did not get admitted to any of the courses that he/she did place a token on.

You are asked to implement TATEC algorithm that tries to minimize the described unhappiness parameter $\overline{U}$. TATEC assigns students greedily. For each $c_i \in C$, students that have highest t_j for the c_i are assigned to the course c_i until $cCap_i$ is reached. Let the token of last student assigned to the course c_i is t_i^{min}. All other students that could not be admitted to that course; where $t_i^{min} = t_i$, are also admitted to the course; thus, increasing the capacity of that course.

You will get a full grade if you implement the TATEC algorithm. You will get a bonus grade if you implement an algorithm **better** than TATEC. Term *better* means that $\overline{U}$ of your algorithm is lower than $\overline{U}$ of the TATEC.

1.1 Random Assignment

Additionally, you are expected to implement completely random assignment disregarding tokens. While assigning randomly, use the original capacities that are provided by the input. Only constraint of the random assignment is that students can be admitted at most x courses, x is the number of non-zero tokens of that student. However, student can be admitted to any of the course regardless of the token amount of that course. We are expecting TATEC should be strictly better than this random assignment.

Info: You can use java Random class for generating random integers.

2 TATEC using Java 8 Streams

You will generate a java program that calculates both TATEC and RANDOM unhappiness values and admitted course student-array pairs. Finally, such program will output to these to files.

Practical details are as follows:

1. You will implement TATEC using Java 8 Streams. You will get **penalized** if you fall back to traditional Java (using for loops etc.). Rare occasions of such code will be tolerated.
2. You will write a command-line style program in which the incoming arguments are as follows:
   (a) **First Argument:** This argument will be a file which will contain C. Each row of the file will be a single course. There will be comma-separated two values; the first one will be the course id (alphanumerical) and the second one will be the course capacity (numerical). There will be a total of n rows.
   (b) **Second Argument:** This argument will only contain the student id of each student (only the id portion of each student in S. Again, this argument will be a path to a file in which each row of the file will contain an alphanumerical student id. There will be a total m number of rows.
   (c) **Third Argument:** This argument will be a file as well. In which there will be a matrix of tokens, each row representing a student. Each row will have comma-separated values of tokens. Every row is T_i of student s_i. There will be a total of m rows and n columns.
   (d) **Fourth Argument:** This will be the value of h.

   Info: Example call; "`java ./Tatec courses.txt students.txt tokens.txt 5`"

3. It is guaranteed that each row of the token list has exactly n amount of columns.
4. Each row of the student list is corresponds to the row of the token list. For example, i^{th} student's id will be on that row of the student list and i^{th} row on the token list will be that student's token.
5. It is **not** guaranteed that the row of tokens sum up to 100 (Equation 3). It is not due to the algorithm but due to a user error. You should check it and if any of the students have wrong set of tokens and return error. This error will be discussed later.

   Hint: Check reduction operation for this.

6. Try to use Java 8 Streams as much as possible. You can collapse the lazily evaluated chain of operations to appropriate data structures at any time. Avoid using classic programming constructs as much as possible.
7. Apply the algorithm to these inputs. You can use any data structure you seem fit to this algorithm.
8. Algorithm should output two set of files. These files are statically named. First set of files should be n row of rows and on each row there should be comma separated values. First value will be the course id, other columns will be the student ids that are admitted to that course. Student ids can be in any order. If no one admitted to that course, that row should only have the course id. Avoid trailing commas at the end of each row. Name of these files are "*admissionOutTATEC.txt*" and "*admissionOutRANDOM.txt*".

   Info: For example, a row on this file:

   ```
   CENG100, e100, e101, e102,
   ```

   Last comma **must not be in the file.**

9. Second set of files will contain $m + 1$ rows of items. Very first row corresponds to $\overline{U}$. Rest of the rows will be the individual $U(s_i$ of each student. This should be in order wrt. to the given student id input file (altough it will be shifted one row due to $\overline{U}$). Name of these files are "*unhappyOutTATEC.txt*" and "*unhappyOutRANDOM.txt*".
10. If an error occurs (item 5), all of the files should not be generated. There should be a print statement informing the user. Exact nature of this error information is up to you and it will not be checked. However; existence of these files will be checked.

3 Class Details

3.1 `public class Tatec`

The only class of the assignment. It has a main function. Some static constants of the output file names. Some basic argument reading is provided for you in this class. Rest is up to you. Entirety of the skeleton code can be seen on Listing 1.

```java
public class Tatec
{
    private static final int CORRECT_TOTAL_TOKEN_PER_STUDENT = 100;
    private static final String OUT_TATEC_UNHAPPY = "unhappyOutTATEC.txt";
    private static final String OUT_TATEC_ADMISSION = "admissionOutTATEC.txt";
    private static final String OUT_RAND_UNHAPPY = "unhappyOutRANDOM.txt";
    private static final String OUT_RAND_ADMISSION = "admissionOutRANDOM.txt";

    public static void main(String args[])
    {
        if(args.length < 4)
        {
            System.err.println("Not enough arguments!");
            return;
        }
        // File Paths
        String courseFilePath = args[0];
        String studentIdFilePath = args[1];
        String tokenFilePath = args[2];
        double h;

        try { h = Double.parseDouble(args[3]);}
        catch (NumberFormatException ex)
        {
            System.err.println("4th argument is not a double!");
            return;
        }
        // TODO: Rest is up to you
    }
}
```

Listing 1: Tatec Class

You can add additional classes if you want. But make sure your code can compile by `java -c *.java` command-line statement. Main function should be in the `Tatec` class. You can use and data structure that suits your needs.

4 FAQ

1. **Q: How do we convert a file directly to a stream?**

   A: Use `Files.lines(...)`. Documentation can be seen here.

2. **Q: What happens if we do not use Streams?**

   A: Algorithm is simple as it is. TATEC technically does n sort operations over the corresponding tokens then selects course capacity amount of students. Then adds remaining students that have the same token as the last admitted student. Unhappiness operations are also simple functions. Because of that you will get **zero** points if you do the assignment without using Streams. Minimal usage of traditional programming paradigms will be tolerated. Moderate usage will be penalized accordingly.

3. **Q: Is there any test input files that we can use ?**

   A: Student pack has a sample input files.

Regulations

- **Programming Language:** You will use Java (version 8 or above).
- **Late Submission Policy:** A penalty of $5 \times Lateday^2$ will be applied for submissions that are late at most 3 days. If unusual circumstances truly beyond your control prevent you from submitting an assignment, you should discuss this with the course staff as soon as possible. If you contact us well in advance of the deadline, we may be able to show more flexibility in some cases.
- **Newsgroup:** You must follow the Forum (`odtuclass.metu.edu.tr`) for discussions and possible updates on a daily basis.
- You should implement your algorithm mostly using Java 8 Streams. Excessive usage of non-functional Java will be penalized.
- Evaluation will be done using both using black-box method and manually. Black-box method will check the "unhappyOut.txt" and "admissionOut.txt" files that is outputted by your implementation.
- Manual checking is done to make sure that you are using Java 8 Streams throughout the algorithm.
- Everything you submit should be your own work. Usage of binary source files and codes found on the internet is strictly forbidden.
- **Submission:** Submission will be made via ODTUClass. Create a zip file named "hw3.zip" that contains all your source code files. Your executable should be named "Tatec" and it should work with the arguments described previously.
- Your code should be able to be compiled using **`"java *.java"`** command line statement.