Data Structures and Algorithms Lab

9. Sorting and Searching

Subject Code: 19ECSP201 **Lab No:** 9 **Semester:** III

Date: 14 Oct 2019 **Batch:** C2

Question: Recommendation Engine Simulation

Objective: Usage of searching, sorting and appropriate data structures in implementing a mock-recommendation engine

---

Note: A recommendation engine is more than what you see here. This is only a mini version to get a mock feel of it.

Had it been not for online shopping, there would be a guy at the counter recommending you the products of interest or at times, a marketing strategy to roll out a new product into customers pool. Now, the recommendation feature comes out as an automated program where the market shifts from offline to online.

Recommendations online take several forms. Without getting into the technical details, here is what happens overall: A user usually gets recommended based on user profile or item profile. In one, the user profile is monitored. His purchases are tracked. Then a similar user is looked for. These other purchases from other users become the recommendations. Then the same is maintained for items. Every item purchased along and together is tracked and monitored (usually called user-matrix / item-matrix). Similar purchases, related purchases, ratings and feedbacks, etc are tracked and processed. Usually, online recommendation engines are hybrid, and they use a combination of several methods.

Item you have purchased:

Probable Recommendations:

E-commerce sites like Amazon, Flipkart, eBay etc all have some form of recommendation engines. No, we don't want to be a competitor yet! But let's implement a simpler one. Let's Implement a 3C Recommendation Engine.

It's an III semester C division recommendation engine (C2, to be specific) specifically built for movies and television shows.

The implementation is detailed to you stepwise. Follow the order. You have also been given with starter code, to begin with.

Step 00: You have already been given a file named – **MovieIndex.txt** which has few data sets. Each data set is of the following example format:

```
avengers ant-man 89
```

which is to be read as "89 people who have seen avengers have also seen the movie ant-man."
Populate the file with more data sets if you want.

Step 01: Load all the data from the file into an appropriate data structure (You are here working from secondary storage to primary storage). Again, this part is already done for you. If you want to change the data structure, you can.

Step 02: Collect a search string from the user.
Example input strings from the user:

```
ant
ant-man
the
the-big-bang-theory
theo
```

The user input is always in lower case letters. The user always gives input with – instead of providing a space between the words.

Step 03: Make a search based on user-entered query string, on the **`movie-name`** data member of the recommendation_data structure. If there is a match, load the result into an appropriate structure (the result will contain recommendation_name and the number of viewers).

Step 04: Sort the results based on the number of viewers. Ties can be broken arbitrarily when there is the same number of viewers for more than one recommended movie.

Step 05: Print all the results to the user as movie recommendations. Print the names without – which is used in place of space. If no match, print an appropriate message.

Example:

```
ant man
the big bang theory
NO RECOMMENDATIONS
```

Step 06: Analyze the efficiency of your recommendation engine. Can you make it better? If so, optimize the necessary operations.

**** Happy Coding ****