COMP1921 Data Structures and Algorithms

Assignment 3: Music Recommender

Due Date: Week 11. Friday, 17th October, 5:00 p.m.

Value: 15%

In this assignment, you will work in pairs, with a partner from your own tutorial group. Details of how to register your “team” and submit your assignment will be given in tutorials.

By doing this assignment, you will learn:
• how to divide and discuss work in a small team (basic project management),
• how to decompose a problem into simpler subproblems (problem solving),
• how to use multiple data structures in a single program (program design and abstraction).

You will also gain more experience programming with linked lists and pointers in C and how to analyse complexity of programs.

Scenario: You have just had the idea to write a music recommender plug-in for facebook. The idea is that your program should make recommendations of interesting artists to a user by combining the ratings of their friends (each facebook user has a number of explicitly identified friends).

To make this work, users must provide a set of ratings; each rating is an artist with an integer score from -3 to 3 (where -3 means “terrible”, 0 means “so-so” and 3 means “great”); a user need only rate some, not all, of the artists. An artist is recommended to a user if the artist is new to the user (i.e. the user has not previously rated that artist) and their friends’ combined overall rating of that artist is positive. The recommendations should be made in order of overall rating (highly rated artists before lower rated). To determine the overall rating of an artist for a user, rather than simply adding up the ratings of that artist by the user’s friends, each friend’s ratings are weighted by the similarity of that person’s tastes to those of the user. Friends’ tastes can be negatively correlated, so one person’s dislikes can be another’s likes.

To make this more precise, suppose \( A \) is the set of all artists and that users \( u \) and \( v \) have defined ratings \( r_u(a) \) and \( r_v(a) \) for each artist \( a \) in possibly different subsets \( A_u \) and \( A_v \) of \( A \). Assume that if \( r_u(a) \) is not defined by the user \( u \) then it is 0. The similarity of users \( u \) and \( v \) is given by:

\[
sim(u, v) = \sum_{a \in A} (r_u(a) \cdot r_v(a))/(9\cdot\text{size}(A_u \cap A_v))
\]

In other words, the similarity of two users is a number between -1 and 1 that is a rough “correlation” of the users’ ratings based on artists they both have rated; the division by 9 is because individual ratings go from -3 to 3. Of course, if two users have rated no artists in common, their similarity is 0.

Let \( f(u,a) \) be the set of friends of user \( u \) who have rated an artist \( a \). The overall rating of the artist \( a \) for user \( u \) is given by (again assuming this is non-zero):

\[
rating_u(a) = \sum_{v \in f(u,a)} \sim(u, v) \cdot r_v(a) / \text{size}(f(u,a))
\]
C Programming Assignment

Before writing the actual plug-in, you write a C program to test the accuracy and usefulness of the recommendation algorithm. This program has a minimal interface. Your program should accept as input a series of command lines (similar to Assignment 2). Commands are for defining users and their ratings, and for requesting recommendations for a user.

Assume there are at most 50 users, that user and artist names can be at most 20 characters long and do not contain spaces, and that ratings are given as integers from -3 to 3. Assume that all ratings are entered before any friends. The ‘friends’ relation is assumed symmetric, i.e. if \( u \) is a friend of \( v \) then \( v \) is also a friend of \( u \). Suggested recommendations should be given in order along with their overall rating (rounded to two decimal places).

Use the following input format (where name stands for a string and int for an integer):

<table>
<thead>
<tr>
<th>Input</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>R name1 name2 int</td>
<td>user name1 gives rating int to artist name2</td>
</tr>
<tr>
<td>F name1 name2</td>
<td>name1 and name2 are friends of each other</td>
</tr>
<tr>
<td>S name</td>
<td>List current suggestions for user name</td>
</tr>
</tbody>
</table>

Example

```
R Alice Abba -2   Alice rates Abba -2
R Alice Britney 1 Alice rates Britney 1
R Bob Abba 1      Bob rates Abba 1
R Bob Britney 1   Bob rates Britney 1
R Bob Dido -1     Bob rates Dido -1
R Cate Abba -2    Cate rates Abba -2
R Cate Coldplay 2 Cate rates Coldplay 2
R Cate Dido 2     Cate rates Dido 2
R Cate Ella 1     Cate rates Ella 1
F Alice Bob       Alice and Bob are friends
F Alice Cate      Alice and Cate are friends
F Cate Bob        Cate and Bob are friends
S Alice           List suggestions for Alice
Coldplay 0.89     Program output
Dido 0.47         
Ella 0.44         
S Bob             List suggestions for Bob (none given, so no output)
```

Note that in this example, the user and artist names are entered alphabetically (for ease of understanding the algorithm) but this cannot be assumed in general.

Sample Calculation

\[
sim(Alice, Bob) = (-2*1 + 1*1)/18 = -0.055; \quad sim(Alice, Cate) =(-2*-2)/9 = 0.444
\]
\[
\text{rating}_{Alice}(Dido) = (-0.055*-1 + 0.444*2)/2 = 0.47
\]
Assignment Diary

As in Assignment 2, complete an assignment diary (the diary will be even more important in this assignment for ensuring good teamwork). Your diary should include brief notes on your team meetings, work plan, main design decisions and test cases. Make sure you include some test cases for the boundary conditions. When you have finished, it is a good idea to look over the diary and reflect on what you have learned from the process of doing the assignment.

Submission

You will need to submit the following files:
- C program (assn3.c)
- Assignment diary (assn3-diary.txt)

Submit your files using the command:

```
give cs1921 assn3 assn3.c assn3-diary.txt
```

You can check that your submission has been received using the command:

```
1921 classrun -check assn3
```

Assessment

Marks for this assignment are allocated as follows:
- Correctness (automarked): 60%
- Solution design, programming style, comments: 20%
- Diary notes including complexity analysis: 20%

There is a 10% late penalty applying to the maximum mark per calendar day the submission is late, with an absolute deadline of Wednesday, 22\textsuperscript{nd} October, 5:00 p.m.

Assessment Criteria

- For design, choose suitable data structures for implementing your solution (ensure the code is easy to understand, and data structure manipulation is efficient given the characteristics of the problem).
- For programming style, follow the [C programming style guide](https://www.example.com) and ensure variable names are meaningful, comments are adequate, etc. If in doubt, make sure you can understand the code written by your partner!
- For complexity, provide the worst case runtime performance and space requirements for each major function in terms of parameters such as: \( N \) (the total number of users), \( F \) (the maximum number of friends of any user), and \( R \) (the maximum number of ratings given by any user). What is the worst case time cost to provide suggestions for \( M \) users as a function of these parameters?

Challenge Question

Find out how many users Facebook actually has. Comment on whether your program would scale up to this number of users, and if not, what you would need to do to make it scale up. What values would you estimate for the other parameters?
**Plagiarism**

Remember that **ALL** work submitted for this assignment must be your own work (that of your team) and no code sharing or copying is allowed. You may use code from the textbook, from the course notes or from the Internet with **suitable attribution** of the source in your program. See the [School of Computer Science and Engineering Plagiarism Policy](#) for specific details.