CMU 17-630 Computer Science for Practicing Engineers Fall

ASSIGNMENT 5

Lattanze Six Degrees Due: See Calendar

Introduction:

The Six Degrees of Separation has long been a matter of folklore. This is the principle that we are all linked by short chains of acquaintances/relationships. This concept was ushered in by experimental social scientists in the 1960’s; namely the pioneering work of Stanley Milgram. His work quantified the phenomena and defined the Six Degrees of Separation that exists between any two people in the United States. Since then, a number of more formal models have been proposed as frameworks in which to study the Six Degrees and related problems analytically. One of the most refined of these models was formulated in recent work of Watts and Strogatz; their framework provided evidence that the Six Degrees (or Small World) phenomenon is pervasive in a range of problems that arise natural and technological contexts (networking). This work gave rise to many social networking systems on the Web today. Much of this research today relies on computer models that make use of graph theory, graph ADTs, and associated algorithms that we explored in class.

Assignment Description:

This is an individual assignment. In this assignment you will investigate the degrees of separation for a particular actor. You have been provided with a text table of movie data as follows:

movie title (rating) (year) (actor, actor,…)

Examples from the file are shown below…

```
:
Afterglow (R) (1997) (Nick Nolte, Julie Christie)
Against the Ropes (PG-13) (2004) (Meg Ryan, Omar Epps)
Agent Cody Banks (PG) (2003) (Frankie Munoz, Hilary Duff)
Agent Cody Banks: Destination London (PG) (2004) (Frankie Muniz,
Hannah Spearritt)
Agnes Browne (R) (1999) (Angelica Huston, Marion O'Dwyer)
A.I. - Artificial Intelligence (PG-13) (2001) (Haley Joel Osment, Jude
Law)
Air Bud (PG) (1997) (Kevin Zegers, Michael Jeter)
:
```

Two actors or actresses are directly linked if they appeared in a movie together – this is 0 degrees of separation. For example the degrees of separation between Michael J. Fox and Christopher Lloyd are zero since they appeared together in *Back to the Future*. However the degrees of separation between Michael J. Fox and Jeff Daniels are 1 since Michael J. Fox is linked to Christopher Lloyd via *Back to the Future* and Christopher Lloyd is linked to Jeff Daniels via *My Favorite Martian*. That is to say there is one person, or one degree of separation, between Fox and Daniels (that would be Lloyd) – at least this is one possible link.

Your program will take as input 2 actors and your program will compute the degrees of separation between the first actor and the second actor. You will use the file MovieData.txt as the data upon which your program will compute the degrees of separation.

As output, your program should list the degrees of separation and the linkages. As an example assume that the input for the program is Michael J. Fox and Jeff Daniels. The output should be displayed as follows:

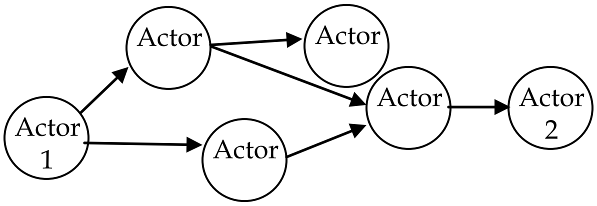
```
Michael J. Fox -> Jeff Daniels: 1 degree of separation

   -  Back to the Future: Michael J. Fox; Christopher Lloyd

   -  My Favorite Martian: Christopher Lloyd; Jeff Daniels

:
```

If the degrees of separation are greater than 6, then abandon the search. If there are no degrees of separation less than 6, then indicate that no links were found (hint: this is a classic graph problem). You could think in terms of the first actor as being the starting vertex of a graph and all of the actors they directly interact with as being paths from the starting vertex:



In this example there are 2 persons between Actor 1 and Actor 2 – or 2 degrees of separation. As soon as we move beyond 6 degrees, we abandon the search. Your program should list all the degrees of separation less than 6 that it finds. Model your ADTs after this graph structure.

You may use Java, C, or C++ for this assignment. Please do not use any prepackaged, open-source, or commercial libraries APIs to do this processing (such as those available from the IMDB). You may use any ADTs you like, but as always, maintain a clean separation of concerns. Model your ADTs after this graph structure. Make sure that you handle all possible errors within the ADTs and the end-user application and that your application is reasonably user-friendly. Your source code MUST BE WELL STRUCTURED AND DOCUMENTED! Ensure that you include headers and comments in your code, properly indent code blocks, and otherwise use good coding practices. If we can't read and understand your code, we will not grade it. You will also be graded on how well structured your application is. Note that a working program alone is not enough to get a good grade on this assignment – structure and code quality matter as much as functionality.

Deliverables:

1. Report that includes:
   - Describe the design of your various ADTs, the data structures you selected, what information and operations you hid and your reasoning for your decisions.
   - Describe how to use your program.
   - Show the analysis of your application in terms of correctness and complexity (performance and memory) as described in class.

2. Demonstration program.

NOTE: Please be thorough, but as concise as possible in your discussion. Note that grammar, good style, and format counts. Any reasonable formatting style/structure is acceptable.

Provide your source code, data files, and executable program.

Put all of these materials (described above) into a zip file and turn it in electronically (blackboard or email). Use Windows file compression or Winzip to compress your files. Include a "readme" file in your archive that explains the contents of your archive. Name your archive as follows: A4+YourFamilyName. So my assignment archive would be named: A4+Lattanze.

Grading:

This assignment is worth 50 points as follows:

- **Report content:** 20 points total for content.
- **Implementation:** Correctness of the application for a total of 20 points. Note the design and structure of your API, application, and how well it matches your design is an important part of the evaluation.
- **Ease of use and write-up quality:** Your application must be easy to use – easy to enter input, easy to read and understand your output (5 points). The write-up will also be evaluated in terms of grammar, format, and readability will also be considered (5 points).