Programming Assignment 2

CMPE 250, Data Structures and Algorithms, Fall 2011

Instructor: A. T. Cemgil
TA's: Barış Kurt, Murat Arar, Zeynep Gözen Sarıbatur

Due: 23 Nov 2011, 17:00 Sharp

Problem Definition

Kevin Bacon Game is to link a movie actor to Kevin Bacon via shared movie roles. The minimum number of links is an actor's *Bacon number*. For instance, Tom Hanks has a Bacon number of 1; he was in *Apollo 12* with Kevin Bacon. Sally Fields has a Bacon number of 2, because she was in *Forrest Gump* with Tom Hanks, who was in *Apollo 12* with Kevin Bacon. Almost all well-known actors have a Bacon number of 1 or 2 (course book, exercise 9.53).

In this project you are going to calculate a generalized version of Bacon number. In this version you will work with a bipartite graph G where actors and movies are the nodes.

1. The distance between two actors is defined as the number of movie nodes on the shortest path between them.
2. The distance between an actor and a movie is defined as the number of movie nodes on the shortest path between the actor and the target movie, counting the movie itself.
3. The distance between two movies is defined as the number of actors on the shortest path.

Part 1:

Your program will take three arguments, one database file, and two actors or films, and return the distance between them. For example:

```
./your_program imdb.small.txt 'Kevin Bacon' 'Morgan Freeman'
./your_program imdb.no-tv-v.txt 'Godfather' 'Being John Malkovich'
./your_program imdb.no-tv-v.txt 'John Malkovich' 'Being John Malkovich'
```

The output will be just a number. If the actor or movie does not exist, the output will be -1.

Part 2:

You will find the maximum distance between two actors, two movies, and an actor and a movie. The input format is:

```
./your_program imdb.small.txt -max actors
./your_program imdb.no-tv-v.txt -max movies
./your_program imdb.no-tv-v.txt -max actor-movie
```

The output will be just a number. If the actor or movie does not exist, the output will be -1.

What you will need

- You can find a small data set in `http://cs.oberlin.edu/~gr151/s11/imdb.small.txt`. The format should be self explanatory.
- Here is a larger data set `http://cs.oberlin.edu/~gr151/s11/imdb.no-tv-v.txt` .

Bonus

Calculate all distances between all actor pairs, and draw a weighted graph with *GraphViz*. GraphViz adjusts the graph layout according to the edge weights. The nodes connected with highly weighted edges are going to be drawn closer, therefore you need to invert the distance values, to make two close actors to be shown closer on the graph. Your code for the bonus will be tested with command

```
./your_program filename -bonus
```

Please name your graph file as "actors.dot".

Submission & Grading

- **What, How and When to submit**
  - You should submit your project in electronic form.
  - You should compress your source code (.cpp and .h files) in a zip file, name it as [pr]_[#]_[Student ID] (e.g. `pr_1_200700803.zip`), and email to `bucmpe250@gmail.com`.
  - The subject of your e-mail must also be the same as your zip file (e.g. `pr_1_200700803`)
  - Your zip file should NOT contain any executable file (.exe) or any folder. If you sent multiple e-mails, only the last one will be taken into account.
  - The deadline is 23 Nov 2011, 17:00 Sharp. Emails tagged later won't be considered.
- **Grading**
  - Warning: All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code.
  - Your program will be graded based the correctness of your output and the clarity of the source code. Correctness of your output will be tested automatically so make sure you stick with the format described above.
  - There are several issues that makes a code piece 'quality'. In our case, you are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively.
  - Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.
  - Try to write as efficient (both in terms of space and time) as possible. Informally speaking, try to make sure that your program completes under 20 seconds.