CS310: Advanced Data Structures and Algorithms

Fall 2018 Programming Assignment 2

Due: Monday, November 6 2018 before midnight

Goals

This assignment aims to help you:

- Practice graphs
- Try greedy algorithms

In this assignment we begin our systematic study of algorithms and algorithm patterns.

Reading

- Graphs (K&T, chapter 3, S&W Chapter 4.1-4.2)
- Greedy algorithms (K&T chapter 4, S&W Chapter 4.4)
- Recursion (from CS210)

Advice

Before writing the code try to run a small example on paper. Think what you would do if you were given the set of instructions or hints and had to do it without a computer. Then start programming. It is always advisable, but especially important in this programming assignment. This assignment is about 80% reading, 20% coding...

Questions

1. **Graphs** (based on the "small world phenomenon" exercise from S&W, see here:

   http://www.cs.princeton.edu/courses/archive/spring03/cs226/assignments/bacon.html.

   Briefly, the assignment asks you to read in a file containing information about films and actors, and produce a histogram of the Kevin Bacon numbers (or anyone else... The center of the universe, Bacon in this case, is given as a parameter).

   The part that writes the histogram is already implemented by S&W:

   https://algs4.cs.princeton.edu/41graph/BaconHistogram.java.html

   Your task is to write a class that, when given the graph with the shortest paths from Bacon (or any other actor), takes an actor's name as a command line and produces this actor's Bacon number and the shortest path to Kevin Bacon. The name format is "Last, First" . So for an input of:

   ```
   Dane, Cynthia
   ```

   The output is:

   ```
   Dane, Cynthia has a Bacon number of 3
   Dane, Cynthia was in "Solstice (1994)" with Scott, Dennis
   ```

```
Scott, Dennis was in "What About Bob?  (1991)" with Murray, Bill
Murray, Bill was in "Wild Things (1998)" with Bacon, Kevin
```

You may use the DegreesOfSeparation class for ideas (see S&W code), but the output has to be different. Name your class DegreesOfSeparationBFS.java .

Make sure that you understand the SymbolGraph class very well before you start coding. An illustration appears in the Undirected Graphs class notes.

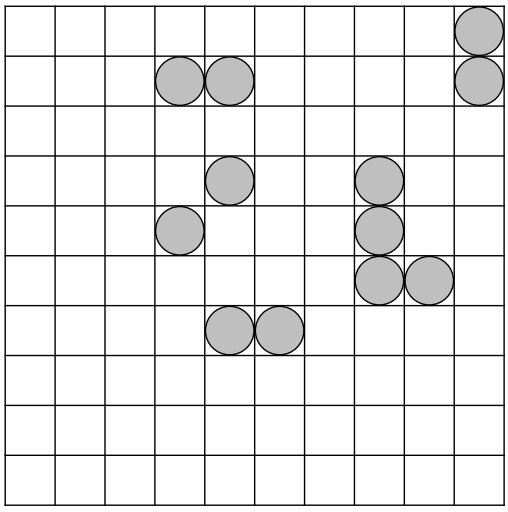
Clarification: The command line parameters should be as in DegreesOfSeparation (it's probably easiest this way). That is – three command line parameters: The input movie file name, the separator and the "center" (Kevin Bacon in this case). Then when an actor's name is written into the standard input (w/o quotes), the path is printed per the instructions. Due to the short notice, if you already implemented it in a different way please state it in the memo.txt and you'll get the full marks if your program is otherwise correct. That is – prints the correct path as stated above.

2. **Greedy recursive search:** Consider an $N \times N$ grid in which some squares are occupied (see figure below). The squares belong to the same group if they share a common edge. In the figure there is one group of four occupied squares, three groups of two squares, and two individual occupied squares. Assume the grid is represented by a 2 dimensional array. Write a program that:

   (a) Compute the size of a group when a square in the group is given.

   (b) Computes the number of given groups.

   (c) Lists all groups.



For the class setup, use the attached Grid.java. The attached file contains a main, the Grid, and the Spot class. You will have to write only the groupSize method and a helper functions.

Here is one idea: set up a local Set of Spots to hold spots you've found so far in the cluster. From the current spot-position, greedily add as many direct neighbor spots as possible. For each neighbor spot that is actually newly-found, call recursively from that position. See the full version of the MakeChange.java code in the book for an example of a recursion helper method – you need a recursion helper here to fill the set. Once the set is filled with spots in the group, the top-level method just returns its size. Notice that you may use the S&W code but it's probably best if you just use standard java.

Usage: "java cs310.Grid 3 7" for example, to search from (3,7), the top occupied square of the L-shaped group. This case should print out "4".

Clarification: The command line parameters should be as above, two numbers. The output should be the group size. In addition, you should be able to list all groups and count the number of groups. You should write a separate function that does it (I did it in one function) and maybe add a variable or two to the class. To print all the groups you can take advantage of the Spot class toString function.

3. Dijkstra's algorithm modification: Modify Dijkstra's algorithm (code is available at the algs4 library, DijkstraSP.java), to implement a "tie breaker". When relaxing the edge, i.e., comparing distTo[w] to distTo[v] + e.weight(), if $distTo[w] > distTo[v] + e.weight()$ then proceed as usual. However, if there are two paths with equal weights leading to a vertex, the one with the *smaller number of edges* will be selected. The big-O runtime should stay the same! So you should also keep track of the number of edges along a path. Before you code, make sure you understand Dijkstra's algorithm or at least its outline. If we don't cover it in class, I'll give an outline on Wednesday. Think about all you have to change in the algorithm to make it work in the same big-O. Name your class DijkstraTieSP.java .

   Test it on the attached file, tinyEDW2.txt, which is a modified version of S&W's tinyEDW.txt , containing a case where a tie breaker is needed.

Delivery

Your source files should be copied to your cs310 account, under the pa2 directory. The code should be arranged into src, classes and lib exactly as in pa1, and all your sources should be a part of the cs310 package – so that there must be a cs310 subdirectory under src and classes. For compilation and running instructions, see pa1. The memo.txt file should be directly under pa2. In particular, the classes should be named as follows:

- DegreesOfSeparationBFS.java: usage java cs310.DegreesOfSeparationBFS movies.txt "/" "Bacon, Kevin" . After the graph is constructed, enter a name to the command line, as in "Kidman, Nicole" (including the quotes). On my home laptop the full movies.txt file caused a stack overflow. You can use a reduced file size for testing, but please indicate in your memo.txt if you did so and what file you used.
- Grid.java, usage: java cs310.Grid 3 7
- java cs310.DijkstraTieSP tinyEDW2.txt
- In classes that use algs4, don't forget the -cp .:../lib/algs4.jar during compilation and runtime.
- In your memo.txt file state the following:
  - Whether you used late days and if so – how many
  - In question 3 – what is the difference in the paths returned by the original implementation of Dijkstra's algorithm and the one you implemented? Specifically, run the algs4 DijktsraSP implementation and save the output to the memo.txt file. Then run your version, save it to the memo.txt file. Explain the difference briefly.