

Final Project: Graph Algorithms

The Model

Graph

Your program needs to represent a graph with nodes that represent capitals of the 48 US states. Each node has a name — the name of the state. For each node, record the information about the capital of that state. Each edge represents a bi-directional connection between two adjacent states. You may consider the *four corner states: Colorado Utah, Arizona and New Mexico* as connected to each other. Each edge has a value that represents the distance between the capitals of the two states. The distances between two cities are based on the geographic distance. (See a separate announcement for a shortcut you can use to compute this distance.)

Algorithms

Your model should implement three graph traversal algorithms:

- Depth-First Search: uses a *Stack* to record the *ToDo* information
- Breadth-First Search: uses a *Queue* to record the *ToDo* information
- Shortest Path Search: uses a *Priority Queue* to record the *ToDo* information

To implement the shortest path you need to represent a priority queue.

The detailed description of the algorithm appears in a separate document. You will encounter a significant penalty for repeating the code - one algorithm implementation should run all three variants, distinguishing between them by selecting the appropriate implementation of a common interface for dealing with the *ToDo* information.

Using Libraries

Furthermore, throughout the project you are encouraged to leverage as much as possible from the existing Java libraries (both the Java Collections Framework, and the JPT libraries). The designer should focus on the design of interfaces between tasks, between components, wrapper and adopter class that allow you to use an existing library class in a customized setting.

The View

The requirements

The view at the minimum should have the following functionality:

- User should be able to see a graphical representation of the graph with labels for the state capitals.
- User should be able to select which of the three algorithms is to be used for the subsequent task.
- User should be able to specify the origin and the destination of the desired path.
- The user should be able to see the resulting path.

Animation and Effects

Of course, the view should be much more elaborate. Here is a list of possible enhancements (you should implement at least a couple of them):

- Highlight the path in a different color in the graphics display.
- Display the steps in the search by highlighting in a different color the *visited* nodes, the *fringe* nodes (those currently in the queue or the stack), the *origin*, the *target*, and the *unseen* nodes. Animate the process using either the timer, or a user advance triggered by a key press.
- Animate the reconstruction of the path by traversing from the found target back to the previous node, all the way up to the origin.
- After the path has been found, traverse the path one step at a time (on timer) and play a short tune for each state.
- Select the origin and the target in a graphics display using a mouse.
- Display in a GUI the path length and possibly the nodes along the path. Or display the sequence of directions from one state to another (e.g. *Go West from MA to NY 130 miles*).
- Allow the user to select a map of Europe or other continent instead of just the continental connected USA.