## Due Nov. 27, 11.59pm

*In the second part of this project, you will implement the Dijkstra's classic shortest path algorithm and visually illustrate these paths on a city-city distance network, that will be provided. Similar to the first part, the project will use BRIDGES to display the transportation network.*

As before you will read in the dataset and build/visualize the graph. The dataset provides locations as X,Y coordinates. The ElementVisualizer class contains a location attribute (setLocation()) that can be used to assign the location of the element.

**Dijkstra's Shortest Path Algorithm.** Generally Dijkstra's algorithm requires a heap to efficiently find the minimum cost edge, but we will dispense with that to reduce the complexity of the algorithm. The attached handout (pdf document along with this assignment) provides a description of this implementation. Follow this algorithm.

**Requirements.**

1. Implement Dijkstra's shortest path algorithm.
2. Test your algorithm on the given dataset and generate a visualization with the locations of the nodes specified by the data file. Use the setLocation() method of the ElementVisualizer class to set node locations. The edges have weights. Come up with a scheme to map the weights of the edges to their thickness (use the setThickness() method of the linkVisualizer class).
3. **Shortest Path.** A user must be able to specify a start node name and end node name and your algorithm should highight the shortest path on the graph. To highlight the path, you should keep track of the parents of each node in the path to the source node. Thus, to highlight the links on the shortest path, once the algorithm finishes, follow the parents of each node (starting from the end node on the path, until the source is reached(source has no parent), highlighting each of the links.

**Implementation Details.** As described in the handout, you will need to maintain a **Mark** array that keeps track of the visited nodes, and a **distance** array to keep the shortest distances. To determine the path, you also need a **parent** array. **Since our node names are strings, use Hashmaps for these 3 arrays, for constant time access**.

**Dataset:**

We will use city-distance dataset of 128 cities and distances between them. This will be provided as a text file.

**Submission Requirements.**

Turn in your source code to Canvas by the deadline; ensure it is well documented. An interactive demo will be required for evaluation.