

## Assignment #3 - Search

### Overview

The search project generates an undirected graph with N nodes and a 25% chance that there is an edge between any two nodes. It will then randomly select two nodes and display the shortest path between them using Dijkstra's Algorithm – an algorithm used to find the shortest path on a weighted graph. The results are then displayed to the user via a graphical user interface for inspection.

### Implementation

For my implementation I used the Jung Library for Java (<http://jung.sourceforge.net/>). This assisted with displaying and managing the graph and determining the shortest path. In my implementation I created N nodes and established the edges using the scheme mentioned above. When choosing edges, I made sure that there were no loops (when a node connects to itself) and since this is an undirected graph, no parallel edges (when an edge is established between two nodes that already have an edge. For example establishing an edge between nodes 7 and 5 when an edge already exists between 5 and 7). The weight for each node was determined using a random number generator which gave me a number between 0 (inclusive) and 100 (exclusive). After the graph has been initialized we randomly select two different nodes and determine the shortest path between them. The program then displays this information visually so that you can easily see the shortest path. In the console I print out two pieces of information that's used for debugging: the weights of the edges on the shortest path and the total weight of the shortest path.

### Results

For my results you'll find five example screenshots that displays the results of a sample run. Three of these screenshots have 10 nodes, one has 20 nodes and the last one has 30 nodes. In the first example we want to find the shortest path between nodes 1 and 2. There are four possible paths:

1->4->2 - 75  
1->5->2 - 107  
1->8->2 - 70  
1->9->2 - 60

As expected, the program chooses 1->9->2 as the shortest path.

In the case of our fourth example, we are searching for a path between nodes 13 and 10. If we look at vertex 10 we'll notice that it only connects to vertex 17 which only connects to 10. The only way to reach vertex 10 is through 17 which doesn't connect to anything else, thus vertex 10 is unreachable from any other node. My program detects this and gives us a path length of null.

The last example demonstrates that we can run the program for relatively large values of N, but it becomes difficult to read read the various edge weights.