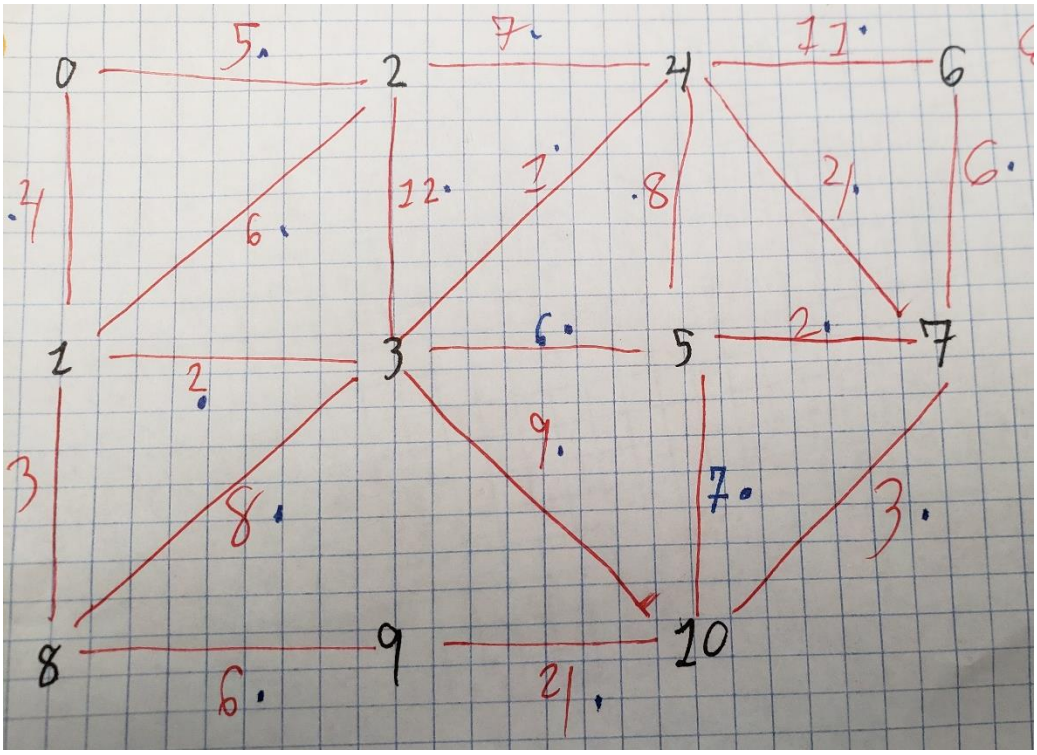
Authors: Tom Rudolph, Connor Van Meter

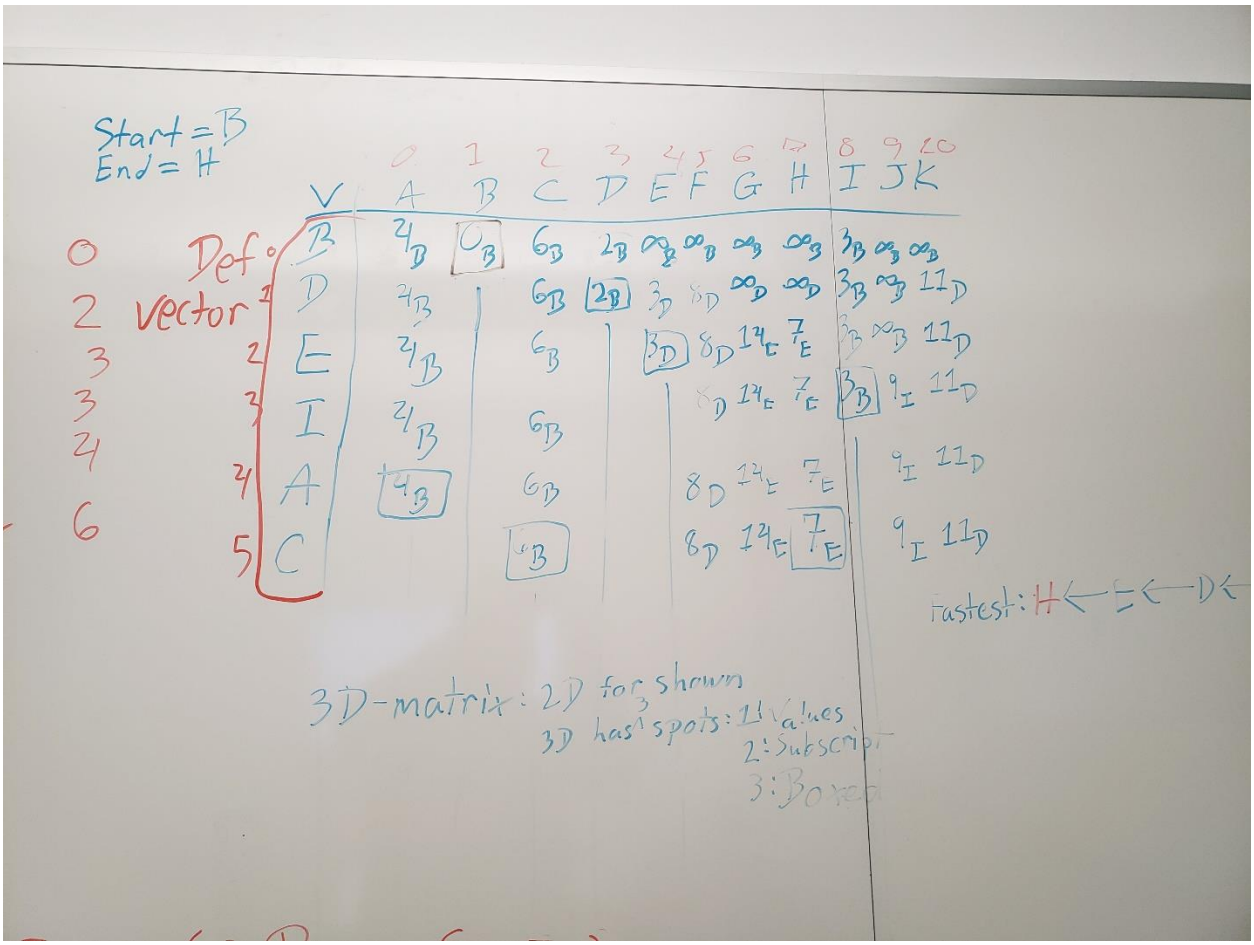
Date: 3/29/19

Lab: 3

The purpose of lab 3 was to use Dijkstra's algorithm to find the shortest path algorithm. To do so a test map was created as shown below.



For the purpose of the next picture shown, the node numbers were replaced by letters in alphabetical order and the direction of the graph was away from node 1 (B in next picture).

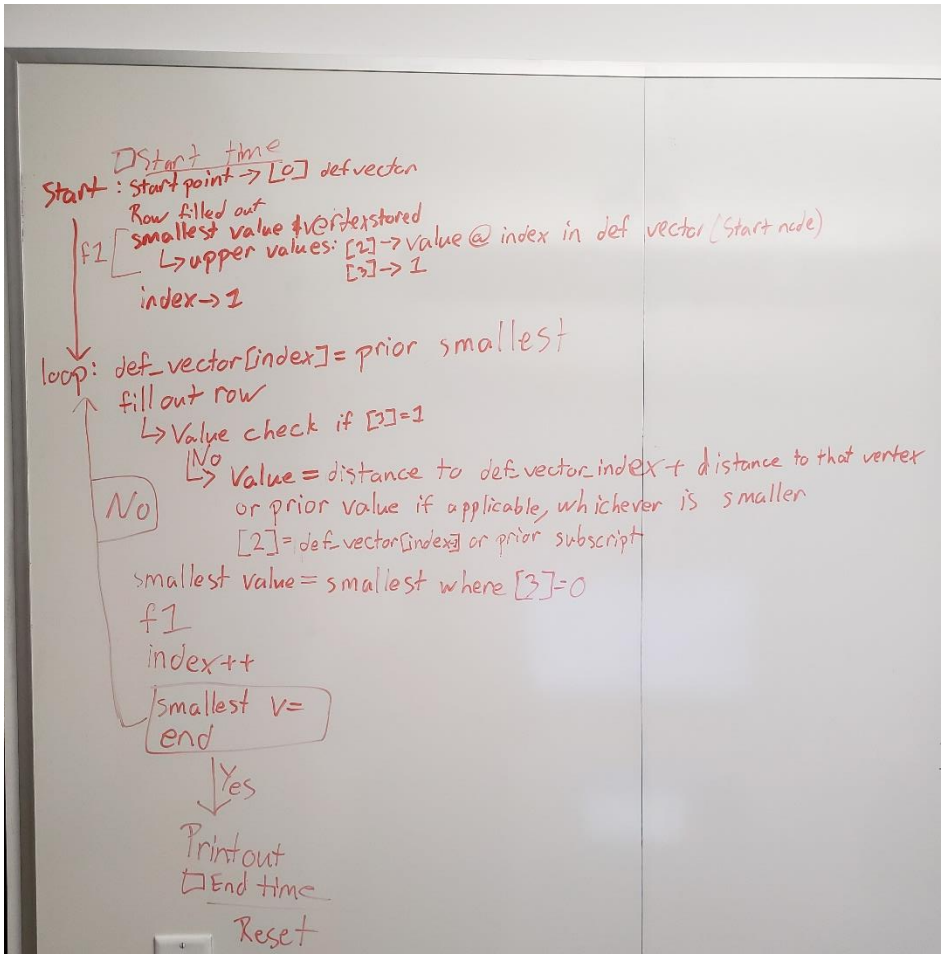


The path we tried to find was from nodes B to H, or 1 to 7. For the first row in the above chart, the distance from the origin node to nodes that it connected with were represented under the destination node. All nodes that do not connect to the origin were set to infinity, indicating that they were not reachable in one step. The origin was put down as a zero and boxed indicating that it is the shortest path to that node.

Next, the smallest distance was boxed on the row beneath and a line was drawn down, indicating that the shortest distance had been reached. Since that distance corresponded to node D, D was the next node to be placed on the side of the chart. This indicated that distance was to be found in relationship to that point. The sum of distance from D, and the distance from the starting point to D, was found to the nodes that it connected to. If that sum was less than the value above it, it would be written down with a subscript D to indicate that the path travelled through D. Otherwise, it would be written down with the subscript B to indicate that the shortest path to it was from the origin.

The shortest path was boxed, and a line drawn down to indicate it was the shortest distance. Then, the above process was repeated. When the shortest path available was the end node, we knew that the end was reached.

Next, pseudocode was written.



The code was broken into three functions: the start, loop and print functions. The start function handled the first row. The start time was set. Then our definition vectors were set. There were two definition vectors, one to keep track of the nodes used on the side of our chart and the second kept track of the distance to that node. With that being set, the first row was filled out. Then the smallest edge was found. The node that that edge went to, and its corresponding distance were stored into the definition vectors.

Next our matrix was filled out. This matrix had a square base with edge lengths equal to the number of vertices. The height of this was three and was used to hold all the data in the chart. Index 0 of the height corresponded to the distance value. Index 1 to the subscript of the value and index 2 to tell if the value had been boxed. Then an index variable was set to one to point to the second row of the chart.

For the loop, the definition vector was set, this was mentioned above. Then the row was filled out. For each destination node, the distance was found to be the sum of the distance to this node and the node found in the prior step. If this was shorter, then the above number would be replaced, otherwise the above value would be propagated down. Both options had the subscript being set to the appropriate values. The smallest non-boxed value was boxed and placed in the definition arrays. The index was then increased.

After this, a check occurred. If the node found was not the end node, then the loop would run again. Otherwise, the run time calculation would be stopped, and the appropriate print out displayed.

The actual implementation of this code did not go as well as hoped. To test, the entire chart was printed out in between each run of the loop, providing the following output:

```
run:
The current graph has vertices from 1 to 11.
Would you like to:
   1. Find a new route
   2. Exit
1
Enter source:
1
Enter destination:
7
4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:0 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:0 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
```

0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0

```
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 0:0:1 0:0:0 0:0:0 0:0:0 0:0:0 0:0:1 0:0:0 0:0:0
1

4:1:0 0:0:1 3:1:0 2:1:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 3:1:1 2:1:1 2147483647:0:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:0 2147483647:0:0 2147483647:0:0
4:1:0 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 3:1:1 2147483647:0:0 2147483647:0:0
4:1:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
0:0:1 0:0:1 0:0:1 2:1:1 7:2:0 2147483647:0:0 2147483647:0:0 2147483647:0:0 0:0:1 6:8:0 2147483647:0:0
1
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 11
        at mygps.MyGPS.dLoop(MyGPS.java:187)
        at mygps.MyGPS.dijkstra(MyGPS.java:78)
        at mygps.MyGPS.main(MyGPS.java:53)
C:\Users\Tom\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 10 seconds)
```

The out of bounds exception is due to the code trying to run too many times. It was assumed that the longest path possible was equal to the number of vertices so that was the maximum number of steps allowed. Since ints were used, infinity was approximated with Integer.MAX_VALUE. In between the first and second lines, there were no shorter routes, so a lack of update was appropriate. After the second line the output updated and again after this. Beyond that, there was no update. The code thought that node I was the shortest distance, failing to recognize that it had been boxed. Because of this, it was not able to find when the end node should have been the shortest, causing it to exit the loop.

After a lot of testing and rewriting the code, we have no clear explanation for the issues we had with the code even though we feel confident with the theory behind the algorithm.