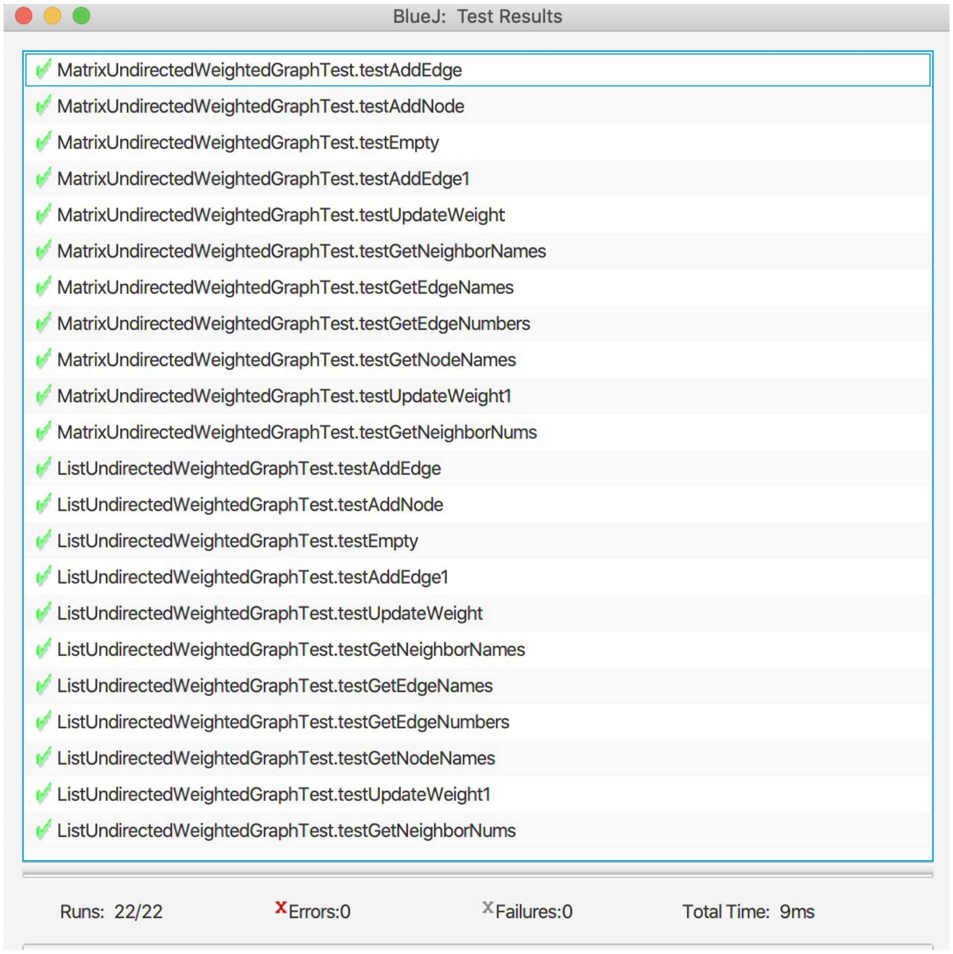
Dylan Maloy
CS150 Lab 11
Lab #11 write-up
11/24/19

Introduction:

The goal of lab 11 was to create both a matrix graph and a list graph. With these graphs, each will be tested for efficiency using Kruskal's minimum spanning tree algorithm, as well as an isConnected() method which checks if the graph is connected. The program contains an abstract class, UndirectedWeightedGraph which contains the kruskal and the isConnected methods. Extending this are the two types of graphs - matrix and list which each contain their own implementations of the abstract graph methods. The ExperimentController class contained the methods which were used to collect data on the speeds of both graphs for both isConnected and Kruskal's algorithm.

Unit Tests:



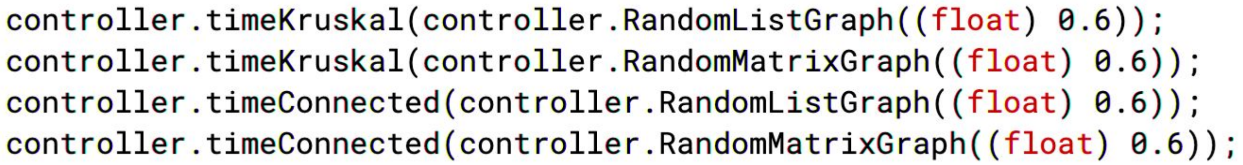
BlueJ: Test Results

✓ MatrixUndirectedWeightedGraphTest.testAddEdge
✓ MatrixUndirectedWeightedGraphTest.testAddNode
✓ MatrixUndirectedWeightedGraphTest.testEmpty
✓ MatrixUndirectedWeightedGraphTest.testAddEdge1
✓ MatrixUndirectedWeightedGraphTest.testUpdateWeight
✓ MatrixUndirectedWeightedGraphTest.testGetNeighborNames
✓ MatrixUndirectedWeightedGraphTest.testGetEdgeNames
✓ MatrixUndirectedWeightedGraphTest.testGetEdgeNumbers
✓ MatrixUndirectedWeightedGraphTest.testGetNodeNames
✓ MatrixUndirectedWeightedGraphTest.testUpdateWeight1
✓ MatrixUndirectedWeightedGraphTest.testGetNeighborNums
✓ ListUndirectedWeightedGraphTest.testAddEdge
✓ ListUndirectedWeightedGraphTest.testAddNode
✓ ListUndirectedWeightedGraphTest.testEmpty
✓ ListUndirectedWeightedGraphTest.testAddEdge1
✓ ListUndirectedWeightedGraphTest.testUpdateWeight
✓ ListUndirectedWeightedGraphTest.testGetNeighborNames
✓ ListUndirectedWeightedGraphTest.testGetEdgeNames
✓ ListUndirectedWeightedGraphTest.testGetEdgeNumbers
✓ ListUndirectedWeightedGraphTest.testGetNodeNames
✓ ListUndirectedWeightedGraphTest.testUpdateWeight1
✓ ListUndirectedWeightedGraphTest.testGetNeighborNums

Runs: 22/22 ✗Errors:0 ✗Failures:0 Total Time: 9ms

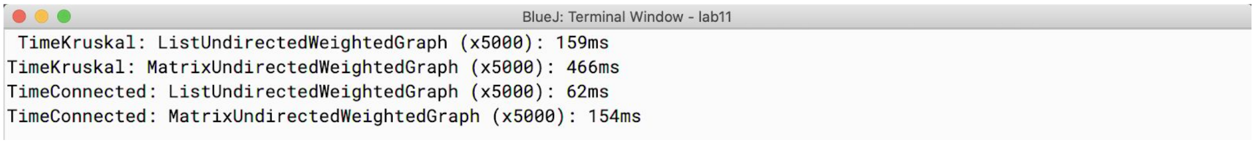
- BlueJ unit testing window

Required Input/Output:



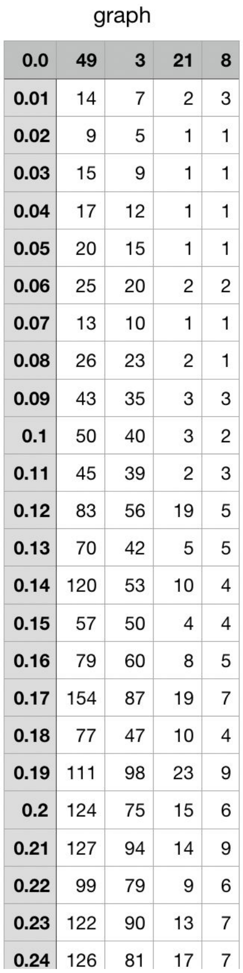
```
controller.timeKruskal(controller.RandomListGraph((float) 0.6));
controller.timeKruskal(controller.RandomMatrixGraph((float) 0.6));
controller.timeConnected(controller.RandomListGraph((float) 0.6));
controller.timeConnected(controller.RandomMatrixGraph((float) 0.6));
```

- Input for a single test



```
 TimeKruskal: ListUndirectedWeightedGraph (x5000): 159ms
TimeKruskal: MatrixUndirectedWeightedGraph (x5000): 466ms
TimeConnected: ListUndirectedWeightedGraph (x5000): 62ms
TimeConnected: MatrixUndirectedWeightedGraph (x5000): 154ms
```

- Output for a single test (reference input for exact code)

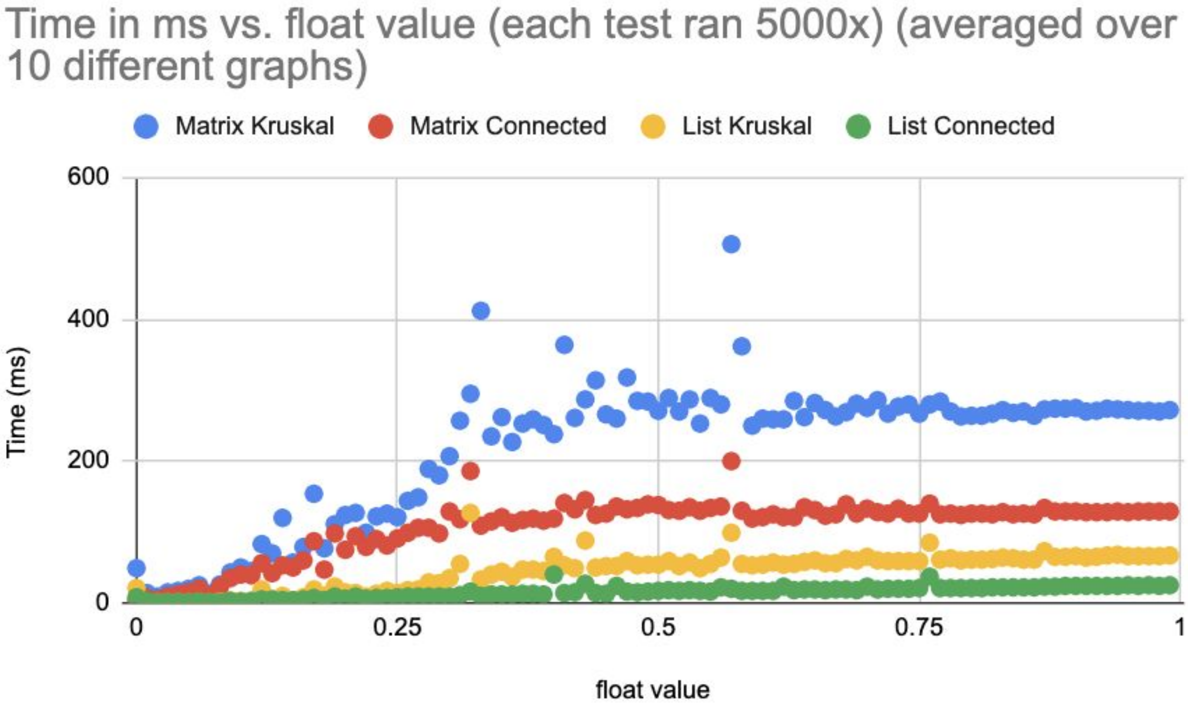


graph

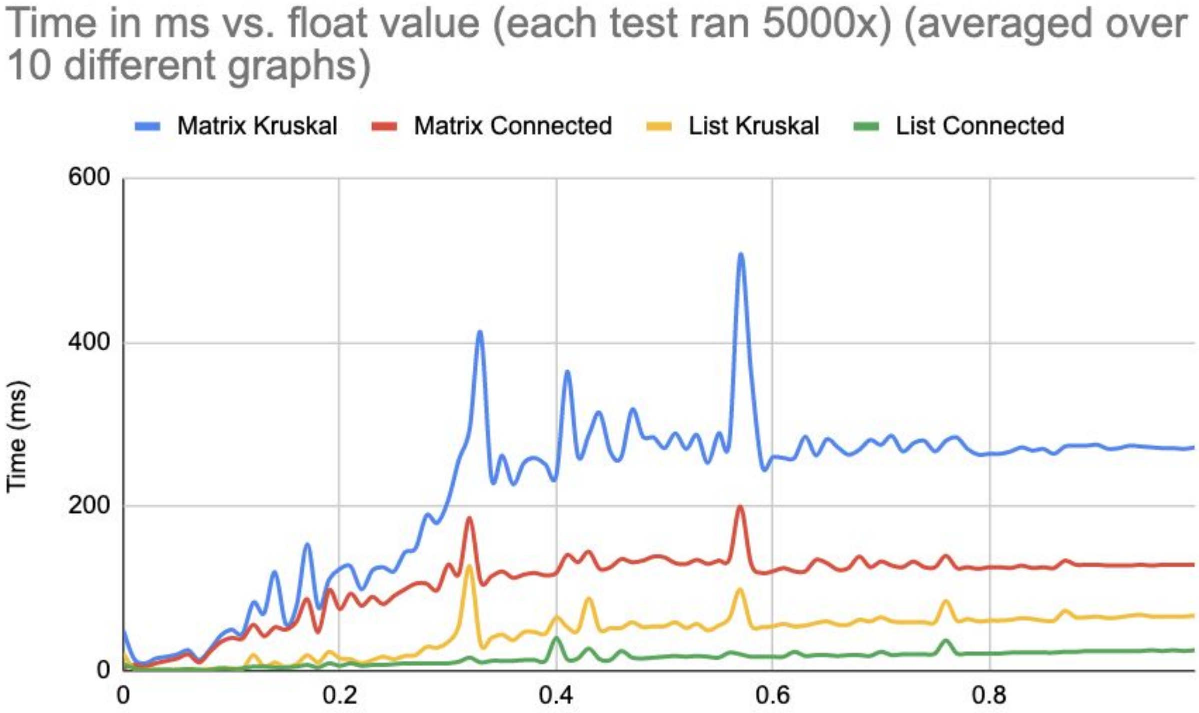
0.0	49	3	21	8
0.01	14	7	2	3
0.02	9	5	1	1
0.03	15	9	1	1
0.04	17	12	1	1
0.05	20	15	1	1
0.06	25	20	2	2
0.07	13	10	1	1
0.08	26	23	2	1
0.09	43	35	3	3
0.1	50	40	3	2
0.11	45	39	2	3
0.12	83	56	19	5
0.13	70	42	5	5
0.14	120	53	10	4
0.15	57	50	4	4
0.16	79	60	8	5
0.17	154	87	19	7
0.18	77	47	10	4
0.19	111	98	23	9
0.2	124	75	15	6
0.21	127	94	14	9
0.22	99	79	9	6
0.23	122	90	13	7
0.24	126	81	17	7

- CSV data output for the test() method (graph.csv)
- (p value, matrix kruskal, matrix connected, list kruskal, list connected)

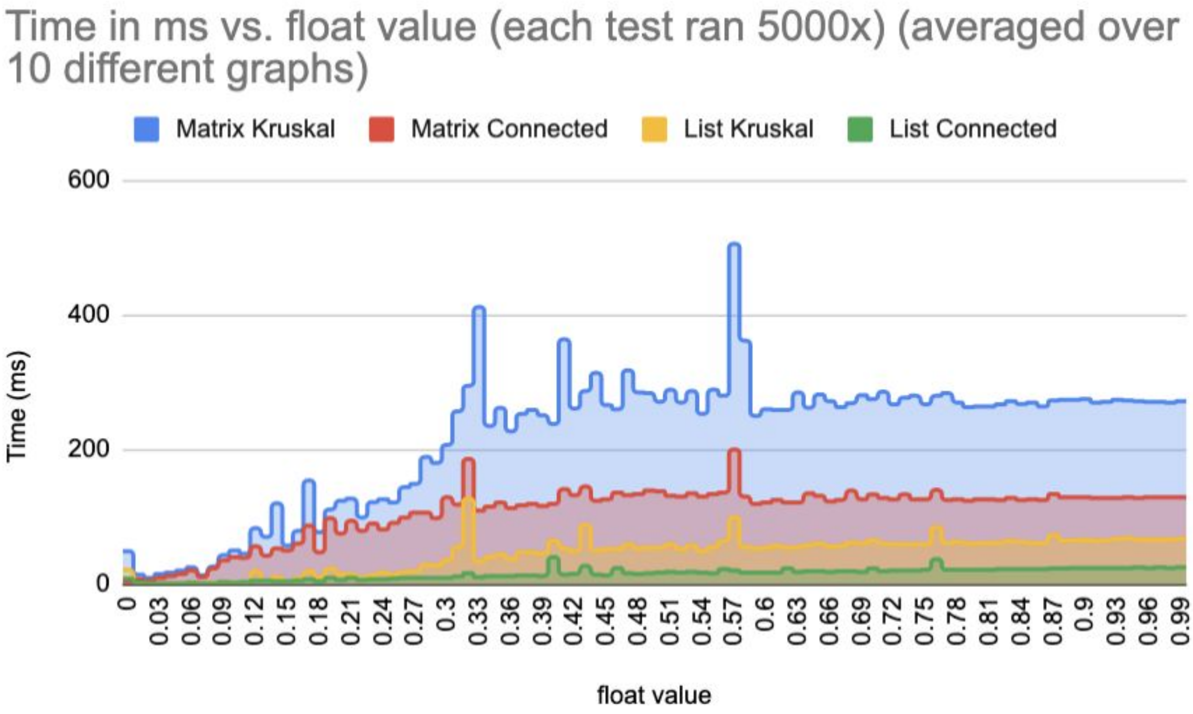
Data:



- Time vs. float value (number of edges) (scatter plot)



- Time vs. float value (number of edges) (line graph)



- Time vs. float value (number of edges) (area graph)

Data Outcome / Explanation:
As the probability p value became closer to 1, all methods took longer. This makes sense because the graphs should become more connected as the p value increases. Interestingly, the list graph was quicker than the matrix graph in both its Kruskal and Connected applications.

Trouble Report:
This section is not applicable because all of my methods work as intended.

References:

Lysecky, R. (2019). Data Structures Essentials. ZyBooks.