Patrick Raley

Professor Elizes

CMSC 350

**Final**

**Analysis:** a  
The Dijkstra’s shortest path algorithm costs, at worst-case, O(n2) to perform. The algorithm tries to find the shortest path from its starting node to the target destination node. It will try to traverse the path with the least amount of nodes and edges with least weight. Sometimes the shortest path might involve traversing multiple nodes with edge weight of 1 than traverse directly to its neighbor (and target node) who’s edge is weight 10.

Something to be noted about this algorithm is that more often than not, this algorithm does not actually cost O(n2). The worst-case cost is a somewhat misleading representation of the algorithm as a whole. It usually takes somewhere between O(n) and O(n2), usually on the lower end of that spectrum. The only time it takes O(n2) is if it absolutely must traverse every other node in the graph to get to its destination. Which is a very unlikely circumstance considering its general context.

**Test Plan:**   
  
**Step-by-step instructions:**

1. Add some extra code to what I copied so that I can read and implement my own graph representations from text files
2. Create test graph where all nodes are connected
3. Create test graph where all nodes are disconnected
4. Create test graph where path is linear
5. Create test graph where edges are weighted
6. Run and verify these work properly before trying to carry out assigned graphs

**Limitations:**

**Expectation of results for user:** Test Case 1 should produce 4 vertices, 6 edges: [Node b; a b], [Node c; a c], [Node d; a d].

Test Case 2 should produce 4 vertices, 0 edges: [Node b; unreachable], [Node c; unreachable], [Node d; unreachable].

Test Case 3 should produce 4 vertices, 3 edges: [Node b; a b], [Node c; a b c], [Node d; a b c d].

Test Case 4 should produce 4 vertices, 4 edges: [Node b; a b], [Node c; a c], [Node b; a c d].

**Test Case 1:**

* Create a test graph where all nodes are connected
* Make a driver that reads and executes it
* Verify results

**Test Case 2:**

* Create a test graph where none of the nodes are connected
* Make a driver that reads and executes it
* Verify results

**Test Case 3:**

* Create a test graph that is linear, all nodes are in a single path
* Read file and execute
* Verify results

**Test Case 4:**

* Create a test graph that has a weighted path
* Read file and execute
* Verify results

**Comprehensive Documentation:   
  
Approach:**

* Found code for Dijkstra’s Algorithm, which includes a Vertex, Edge, Graph, and Algorithm Class. (From http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html).
* Created a method to read text files to create the graphs and create graph representations in notepad.
* Create test graphs to ensure that algorithm and program runs smoothly
* Recreated the graphs from the assignments

**Lessons Learned:**

* t

**Possible Improvements:**

* H