Edwin Mak

HW 5 Write-Up CSE 373 HW #5

1) The worst case asymptotic runtime of my adjacentVertices method would be O(V). The method goes into the map and retrieves all the out edges of that particular node. In the worst case, all of the vertices are adjacent to that vertex.

The worst case asymptotic runtime of my edgeCost method is going to be O(V+E). The method retrieves all the adjacent vertices of that vertex, which in the worst case is all of the certices. Then, it iterates through all of the out edges and retrieves the cost of that edge.

The worst case asymptotic runtime of shortestPaths is O(VlogV+ElogV). I used a priority queue in my implementation for Dijkstra's algorithm, which improves its worst case runtime. It would have been O(V^2) if I had used a regular find in the shortestPath method.

2) I did not make extra copies of objects to protect abstractions. This is because I left the fields of my Vertex and Edges immutable. As a result, there is no way for a client to change the fields of my vertices when they're in the priority queue.

3) I tested my code by drawing out the examples from lecture and putting the edges and vertices into a test file. I then chose different paths to test my shortestPath algorithm. I also had to adjust and test my priority queue, which I did by making calling insert, deleteMin, and decreaseKey multiple times.

4) N/A

5) For my above and beyond, I implemented Dijkstra's algorithm using a priority queue. The hardest part was trying to implement descreaseKey because I had to find the index of the heap in constant time.

I also made a real world data set example of a social graph. In my example my vertices represent people. The weights in the graph represent mutual friends, and would be an undirected graph. A call on shortestPath means you are looking for how two people in the social graph connect to one another through the fewest number mutual friends. Though I can't think of a situation where that information would be relevant, it can be fun looking up that kind of information.