CSE 373 Homework 5 Write up

Who is in your group (Give name, UW NetID & student number of each person)?

Geoff Gray, Austin Meyers

gegray, arm38

1463717, 1228316

gegray@uw.edu, arm38@uw.edu

• Describe the worst-case asymptotic running times of your methods adjacentVertices**,** edgeCost**, and** shortestPath**. In your answers, use |E| for the number of edges and |V| for the number of vertices.** ***Explain and justify your answers.***

Adjacent Vertices – O(|E|)
In the worst-case scenario, every edge in the directed graph originates from the passed vertex so it must iterate over all edges.

Edge Cost – O(|E|)
In the worst-case scenario, every edge in the directed graph originates from the first passed vertex.

Shortest Path – O(|E| + |V|*log|V|)
First, we know each edge must be checked exactly once, which is linear (|E|). Next, we must visit each vertex exactly once (|V|), and from there, visit each vertex's adjacent vertexes (log|V|).

For sufficiently large graphs, we speculate that the problem could be broken down into sub-graphs. For example, a graph with a large number of vertices could first find shortest path among a set of, say, 10 or so vertices. These paths would then form new "vertices," yielding a smaller graph. This method could be continued until the computation is possible. This would not produce a true shortest path, but could be an acceptable approximation.

It is also clear now why negative values could be problematic, especially for cyclic graphs. There could exist a loop with a negative net sum, leading to an infinite loop and [never] ending with a negative infinity cost.

• Describe how you tested your code.
First, we used the debugger… a lot. We found that visualizing the data structure helped us to understand the algorithm. The FindPaths class provided reusable code for testing.
For testing vertices, we input a static vertex array and then transferred those to a Hash Set, monitoring the progress at each add. Then, we used the MyGraph equals() function to compare the two Sets. The debugger helped us double-verify. We took the same approach for testing edges, though using an ArrayList<Edge>. For testing shortest paths, we drew out a diagram on a map (US, not Map<>). From here, it was fairly easy to spot paths that could return multiple results, which were ideal for testing. Additionally, we found an external shortest path finder online to double-check our results.

• If you worked with a partner:

a) Describe the process you used for developing and testing your code. If you divided it, describe that. If you did everything together, describe the actual process used (eg. how long you talked about what, what order you wrote and tested, and how long it took).

We used a version control system to collaborate on the project. We divided tasks as described below. We did work initially in the same room, but continued to work on our own during the development project, using version control to keep our projects up to date. We discussed the tasks and responsibilities for less than an hour, and implemented the code in the order described in the spec (MyGraph + vertex, FindPaths, Test).

b) Describe each group member's contributions/responsibilities in the project.

For development we both took on different areas, Geoff focused on testing and the write up, as well as the implementation of FindPaths.java. Austin was responsible for the implementation of MyGraph.java which required changing some of the code on Vertex.java. We then both ran some tests on our own to make sure the project was working as expected. Then we wrote the tests to analyze the program.

c) Describe at least one good thing and one bad thing about the process of working together.

Working together gave us a more wide view of the task at hand. When developing the solution we realized it could be done a few ways, and having multiple views helped us land on the solution we created.

Working together also introduced an issue of differentiation in writing style. Understanding exactly what the other persons code was doing took an amount of time that could have been used elsewhere if there had only been 1 developer.

• If you did any above-and-beyond, describe what you did.

We used a Priority Queue to implement djikstras algorithm, otherwise the solution follows the specification. We also did pen and paper testing as well as verifying shortest paths with external tools to ensure that our solution works correctly (images of both included in submission).

Appendix

Place anything that you want to add here.
How to code for quantum computer? Plz tell now.