CptS 223 Micro Assignment - Dijkstra's Algorithm

For this micro assignment, you must implement the function that executes Dijkstra's shortest path algorithm inside Graph.h. The function is called dijkstra(int startingID). It's found at the very end of the file.

I've left in a timesaver test on dijkstra's that only re-runs the algorithm if it's from a new starting point than you ran it last time. This comes into play when you query for paths in the graph since they first call dijkstra's to make sure they have a valid data path to test.

You'll find that all of the vertices have ID numbers for their names. These are integers and they match the bucket they live in inside of graph._vertices. This is a total time saver and simplifying approach that I used for this implementation. They *should* be kept in an unordered_map<string, Vertex*> instead of a simple vector, but I don't have time to re-do everything this term.

The graph we're testing with is the same one from the book in Chapter 9 (page 394). I've added a single node with ID #0 that's disconnected from the graph to show that you can't reach the node. I'm using a library to provide a path weight of infinity and it shows up in the double type as 'inf'.

When you run 'make bigtest' you'll see it takes a while to run the Mouse Brain test. That's a second graph in MouseBrainGraph.dot with about 216 vertices and 21k edges (give or take). It takes about 15 seconds to run with my implementation and computer. It takes about 30 seconds on sig3.eecs.wsu.edu.

Grading

Your submission will be graded based on the following:

1. [10] Your solution builds, does not cause any runtime issues, and passes all test cases

Due Date

As per normal, check your code into the Git branch for this assignment and upload only the file with the data about your submission: name, EECS username, WSUID, commit hash.