Title: *Dijkstra*
Assignment Name: Assignment 6
Student Name: Carson Stevens
Instructor: Mark Baldwin
Date: April 3, 2019
Comments: Compiled with “g++ -std=c++11 main.cpp -o graph” outside of CLion. Executable is can be run from command line with “./graph”. Command input “ACCURACY” runs my implemented tests. The command input “PATH” will print out the cost and path between two vertices. Noted that style guidelines require “large” projects to have classes in separate .h and .cpp files. This was not implemented in this assignment due to the assignment being relatively small (The large bulk of code is in TEST1 and TEST2 and ACCURACY which is all hard coding).

Implementation Idea:

I used several data structures when implementing this. I had a set of visited vertices, a map (called table in the code), that held the current total costs, the vertex name, and the vertex it came from. The last data structure was a minimum priority queue that sorted the edges by total cost. The first edge was place in the table and into the queue. Then the code enters a while loop that repeats until there are no edges left in the queue. If the vertex that is popped from the queue is already in the visited list, it is ignored. If it isn't in the visited set, it is added to the visited set. Then all the neighboring vertices are gathered and iterated through. Then a check to see if the vertex is in the map determines if it should be in the priority queue already. If not in the map, the value is added to table with the sum of the cost to get there and the new cost vertex is added into the priority queue. If the neighbor is in the map, it checks to see if the cost is better, if the cost is better, it updates the table and gets added back into the priority queue. Printing the results is as easy as back tracing the steps through the map's previous vertex attribute. If the looping can't find a vertex in the map or loops back to the start, there isn't a valid path and the algorithm end. Finding the cost is as simple as calling the destination vertex in the map.

Problems

I tried to use a struct instead of a pair for my priority queue and was getting so very strange errors. I have used custom classes and structs in priority queues before, so I'm not sure what the issue was. I change it to a pair<*previous, total_cost*> which solved all the weird problems. Otherwise, I just followed the algorithm using the priority queue for sorting.

Testing

Tests are ran with the command "ACCURACY". This creates test cases that implement the getAdjacent(), getWeight(), and dijkstra() functions from the graph class. It uses cases to test moving from node to node to see if the cost is what it should be. Another test checks the size of the graph to make sure the INIT function works. The driver then tests adding vertices and invalid edges to make sure that it is not able to add edges to vertices that don't work. I then called the dijkstra function on several paths created from the TEST1 command input. I tested to see if the paths were the same as I expected and the cost was the same. I also created a path that wasn't possible to make sure it could handle graphs without a path to possible location. This concluded the testing portion. The accuracy of the tests is then printed (is at 100% with everything working; can't find a test case that breaks it).

TEST2

The creates the game map graph from the slides. No tests were run, but the user can see that all the vertices and edges have been added. It can also be printed to see the results match the slides. *EDIT from last time:* I tested the path with the dijkstra algorithm and it displayed a correct path from the two farthest cities on the map. The path followed the road which made logical sense.