Lab 10 – Graph

# Problem

Assigned the task of creating a graph of strings. The bulk Graph class has already been created, along with a driver program that should run the graph. We were tasked with coding: addVertex, vertexIsContained, addEdge, getVertex, printDFS, printBFS and printLazyDFS methods.

# Proposed Solution

Create graph of strings, that each have a weight (distance) from one node to the other. The add vertex should add a new node with the provided string name, and it should only do this if the vertexisContained method does not detect that this name already exists. The add edge method should have a pointed direction from one vertex to another, along with a weight that denotes the distance from point a to b. We utilize a get vertex method to help us implement the add edge method. PrintDFS method should print out the names of all the vertices in depth first order. DFS does not have verifiability, and thus we also need to demonstrate BFS. Print BFS should print out all the vertices name in breadth first order. PrintLazyDFS should print out all vertices in depth first order, but only if the distance to travel there doesn’t exceed a value given in the driver program.

# Tests and Results

Was able to code all methods to work properly.

# Problems Encountered

Didn’t have any issues with this lab, the only real problem I had was that when I was coding the printLazyDFS method, I forgot to change my method calls inside the DFS method to printLazyDFS, and it was instead calling back to DFS original method. Fixing this issue fixed my printing disparity with the example.

# Conclusions and Discussions

I think this was a graph. It demonstrates the complex nature of the connections between nodes(can be more complex than trees for sure). Unsure what kind of problem you would specifically need to use a graph to solve, but it seems to be a very niche data structure.

# Additional Questions

**Lab Report Questions:**

1. Would this instance of a graph be considered at tree as well?  Why?

No, this graph is not a tree. There is no clear root, and you could get stuck in a loop traversing from v4 to v7 to v6 to v4. Trees have no loops.

1. Hypothetically if there were a graph that had an infinite amount of nodes branching to one side, would the DFS algorithm visit every node?  Would the BFS algorithm visit every node?

The DFS algorithim could get stuck travelling down an infinite path of nodes. Lack of verifiability.

BFS algorithm fixes this by first looking at the neighbors (starting at the origin or some arbitrary node) and then traversing to the next node in the neighbor list.