**Graph Algorithms Write-Up**

For this assignment, instead of taking the slightly easier route of just assuming the type associated with my nodes, I used generics. My previous Edge and Adjacency List classes just assumed integer nodes, so I would’ve need to make String versions at the very least. I took it a step further by including generics. My edge class was copied and pasted, and I then extended that class to a Weighted Edge to store weights as well. I copied over my Adjacency List and implemented support for generic nodes and created a Weighted Adjacency List as I plan to allow the list to retrieve the weights of its nodes. After I had my data structures, the only thing left were the algorithms.

I started out with Dijkstra-Prim as it was the algorithm mentioned first in the textbook. I knew I would need a data structure for the output and fringe, so I mimicked the internal structure of the Adjacency List. I also set up a list to keep track of any node I already visited. The initial setup of the fringe and output was easy; the while loop was a little harder. I decided that I would remove any connection in the fringe once it was added to the output, so I wrote a function to check if there were any connections left in the fringe. Finding the next smallest edge in the fringe wasn’t too difficult, just a simple double for loop. The logic for updating the fringe was slightly more difficult, specifically updating the fringe to always store the lowest weight connection to any node. I had to do incremental printing of the fringe to validate my logic but I eventually got it working.

Doing Dijkstra after Dijkstra-Prim made it a lot easier. I decided to keep track of the current shortest path to every node I had come across, necessitating another data structure in addition to the data structures for the output and fringe. I wrote a function to quickly calculate the total weight of a path which made updating the fringe easy. Finally, for both Dijkstra and Dijkstra-Prim, I wrote a function to handle bidirectionality of edges for the output adjacency list given that the algorithms functionally treat unidirectional and bidirectional edges the same.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence