Name: Jose Iturria

Section: COP4534, U01 - Algorithm Techniques

Panther ID: 5464894

Project 2

Conclusions

This project proved to be very difficult- In a way, this is a good, since I was able to see the project to completion and overcame the challenges I faced. For one, I had to learn about weighted graphs, and how figuring out a shortest path is implemented using Dijkstra’s Algorithm. The final output of this program included a GUI (shown later), and a small console output, mostly to help me debug the process of obtaining the shortest path.

Text

Description automatically generated

Aside from a couple Java nuances (I’m way more familiar with Python and C#), the biggest challenge by far was creating a GUI that would display all my nodes, their connecting edges and weights, and finally the shortest path within all the weighted edges. I had to bring out my old Trigonometry textbook to figure how to work with pixels.

Here is the code that will dynamically add a new node to a graph:

Text

Description automatically generated

This method will dynamically set nodes in a circular fashion, evenly distributing nodes around the frame of the display, with its boundaries being the leftX and topY of the window.

Figuring all of this out required a lot of research and trial and error. Luckily, the class example we worked on in class sped up the development process. I think the hardest part of creating this GUI, was to figure out the placement of objects. Not only because I had to work with a set screen-size and pixels, but more-so having these be created dynamically, since our input text file had to support varying sizes of graphs with their included vertices. Figuring out how to dynamically set an edge within a for loop is something I never thought I’d have to do, but I’m proud I was able to figure it out. To analyze my output, I created a screenshot of my empty nodes, and manually filled out the graph using Paint. Diagram

Description automatically generated

Once I had a rough idea of what my program was supposed to look like, figuring out how to add edges and weights became simpler. Obviously, the resulting GUI looks much better than my art skills:

Chart

Description automatically generated

You can see a bit of the magic code that results in the picture above. By going through every single element in the matrix, we can determine what nodes are connected to each other, as nodes that are not connected, will have ‘0 weight’ to them.

And of course, this code is dynamic. Adding more nodes will update the graph, and changing weights will be reflected in the shortest path.

A picture containing chart

Description automatically generated