Gabriella Willis

Summer 2020

Cs 404 – Final Project

Project 2 – Emergency Vehicles

**Project 2: Emergency Vehicle Dispatching System**

**Algorithm:**

In order to implement a way to use the closest emergency vehicles as requested, I used the shortest path algorithm. This was developed after creating and mapping out nodes of the situation first. Using shortest path meant that the edges would need to be weighted out and formed a connected graph.

My initial thought was to create a tree to track zip codes and vehicles. I would then assign values for the points of connection. Ex: 0 is the source and will be picked first. A set containing vertices wouldn’t contain all of them so I would iterate through adjacent “neighborhoods” to add them to the list. Simultaneously I would have to update the weight. If multiple connections were found, I would add up the sums of each path taken and go with the one that was the shortest.

For the data to be looked at, I chose to use a JSON file because I could group together the data easily and iterate through it depending on what I was looking for or how I would group it together in my program. Within the program, I divided up the data between vehicles, requests for type of emergency, and distance that needed to be traveled.

Using information from: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-priority_queue-stl/>, I was able to figure out the kind of algorithm that would best fit shortest path which is called Dijkstras shortest path algorithm. This is how the site implemented and showed what would be the connecting features.

A screenshot of a cell phone

Description automatically generated

SOURCE: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-priority_queue-stl/>

**Complexity:**

Based on the way I implemented the algorithm and again, help from our class sources to identify the type, would classify big O as O(E log V). Through geeks for geeks, I was able to understand that this represents E as edges and V as vertices. If I am correct, going through a sorted array would make this worst case of O(n^2).

After running the code, this is the output that I have obtained and saved to the output.JSON file within the project. \*Showing the first set and the last\*

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

**SOURCES:**

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-priority_queue-stl/>

<https://www.geeksforgeeks.org/shortest-path-unweighted-graph/>

[https://www.geeksforgeeks.org/breadth-first-search-without-using-queue/](https://www.geeksforgeeks.org/breadth-first-search-without-using-queue/?ref=leftbar-rightbar)

<https://www.geeksforgeeks.org/print-nodes-at-k-distance-from-root/>

<https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>

Canvas Resources within our class – CS404 Algorithms and Complexity