CISC 220
Liz Racca
Allan Chandy

Dijkstra's Algorithm: Newark GPS

Summary

For our final Data Structures project, we applied Dijkstra's algorithm to a map of Newark to find the shortest path between locations on campus. We were able to work with a research group on campus to get a data set for our project. This included a list of map segments with lengths, starting and ending points, and road names for the city of Newark. We used this data to build a graph of the data in C++. After the graph was built, we picked a source Node and applied Dijkstra's algorithm. We updated the map so that every Node stored the minimum distance to the source Node. Additionally, each time we updated the tentative distance, we stored the edge that caused the update. This allowed us to trace back a path, leading from the Node to a specified target. For any given source Node, the minimum path to every Node in the Map could be found. This is because Dijkstra's algorithm fills out the distance and path to every Node from the given source.

Data Structures

To represent our map, we created a Node and Edge class. Nodes stored information about the points on the map and Edges stored information about the line segments between Nodes. We decided to use these as our base data types because it allowed us to represent the Map in a way that worked well with the format of our data. Another advantage of these data structures was that we could give each Node a linked list of the Edges leading out of it to run Dijkstra's algorithm without having to look up the Edge and tNode we were interested in at each step. We used a linked list to store all the Nodes in the map and moved them into another linked list once they had been visited. This meant that at each step of Dijkstra's algorithm, the amount of time it took to find the unvisited Node with the minimum tentative distance was shortened. Other implementations of Dijkstra's algorithm utilize a priority queue or a min heap to store the unvisited Nodes. This replaces the findMin function in our code by insuring that the Node with the minimum tentative distance can easily be popped off and used as the next visitNode. For a data set of our size, searching the nodeList at each time step does not pose a problem, but switching to a priority queue or min heap would be a potential improvement for a redesign of our project.

Results:

In order to produce results for our program we first had to specify a source node and a target node. Using this information, our program populated the map of Newark and then used Dijkstra's algorithm to set the shortest path from one node to another. This information was important since it eventually was able to find the shortest path from the source node to the target node by ensuring that program only used the path that had the shortest total length from the source to the target node. Our program finally outputs step by step directions to get from the source node to the target node.Three example paths generated from our program are shown below.

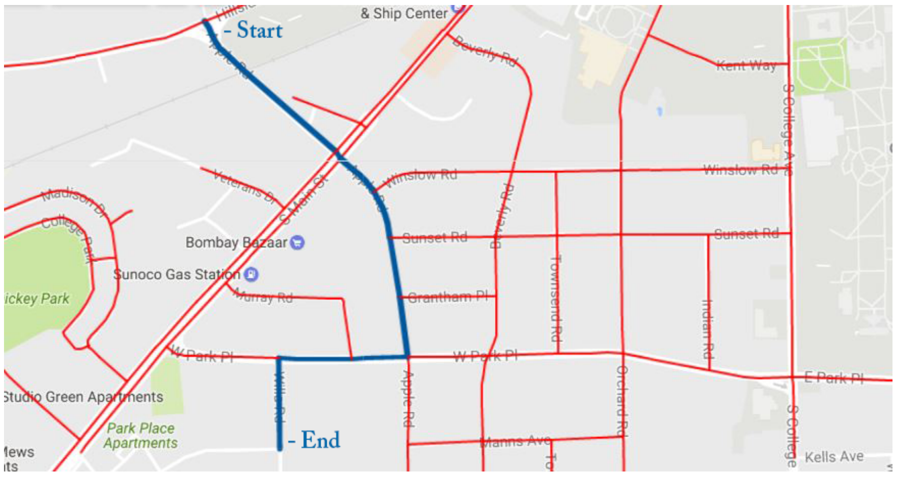

Figure 1. Google maps image of path 1 going from Node 170139186518 to Node 170259185821

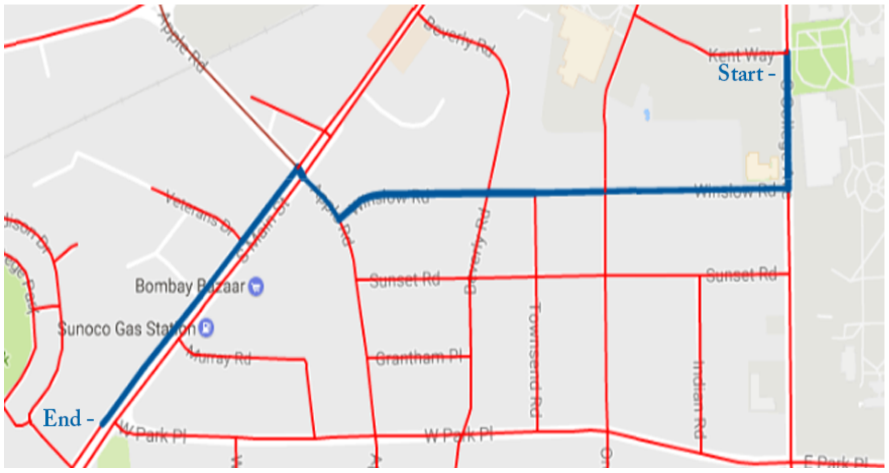

Figure 2. Google maps image of path 2 going from Node 171097186444 to Node 170055185986

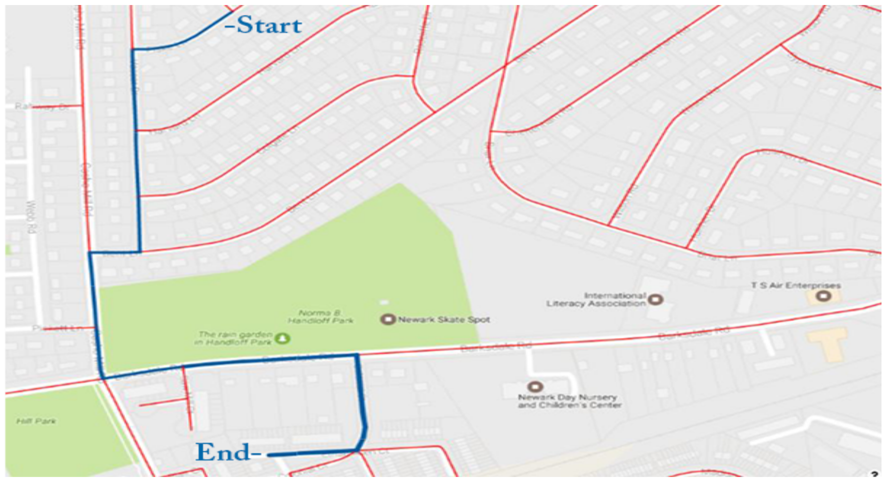
-Start
End-
Norma B Handloff Park
The rain garden in Handloff Park
Newark Skate Spot
International Literacy Association
T S Air Enterprises
Newark Day Nursery and Children's Center

Figure 3. Google maps image of path 3 going from Node 169386186949 to Node 169423186269