Proposal

Leading question:

In order to increase the efficiency of travelling, it is important to understand the best route of subway lines in a city. To achieve this goal, a comprehensive calculation of each route is needed. So, we want to produce a map of the subway lines in Beijing, China and use a search method to find the nearest path between two stations.

Dataset Acquisition and Processing:

We download the information of various Beijing subway routes from a website[1]. Since the longitude and latitude are hard to recognize and compare, we simplify the longitude and latitude of each station to obtain the subway coordinates, and calculate the distance between two stations. This distance is the weight of the corresponding edge. In this way we can get an expected time of passing from one station to another. After simplifying the data, we will store it into a csv file to use later, which can help us to find the shortest path. We can also get where two subways lines intersect from data, which can help us to draw the graph of the whole subway system.

Graph Algorithms:

We will use BFS, Dijkstra's Algorithm, and graphic output of graphs. To complete our goal to identify the best subway route from one location to another, Dijkstra's Algorithm is used to identify this shortest path. This method will take in two vertices, which represent two different stations, and returns a vector/list of vertices that represents the route we need to take to get from the first station to the second. The expected time complexity of this algorithm is O(V^2), where V is the total number of vertices in the graph.

Also, we want to visualize the shortest path we identified, which means we need to create a graphic output of our data structure and highlight the path on that picture.

Graphic output of graphs will take the data of every station node to draw a graph of the whole subway system. If we can go from one station to another, we connect them with an edge. Finally, we want to get a connected graph of the subway network in Beijing. Since we need to go through every station node, we expect the running time to be O(n). This graphical output function takes no argument and is void, but it will write the created graph as a PNG in the specified folder. If

time permits, we can take a step further and project our graph onto a real map. In this way we can have an understanding of the distance between the stations and what the shortest path looks like.

Timeline:

Nov. 8-Nov. 15 obtain and process data

Nov. 16-Nov. 23 Complete graph Traversal (BFS)

Nov.24-Dec.1 complete Dijkstra's Algorithm

Dec.2-Dec. 8 complete Graphic Output of Graph

Dec.8-Dec.13 debug!!!!! While writing final report and presentation

Reference:[1] https://www.sohu.com/a/147064957_466863