Project 3 Report Da Cheng id: 0028653515

Abstract

This project is designed to find the shortest path from the given source to the given destination in a graph. My method consists of two parts: building adjacent lists and applying Dijkstra algorithm based on the adjacent list.

The description of my solution:

For the part of building adjacent list, I use a link list to store all the information of every vertices including their value, distances from the source vertex, their coordinates, the path to the destination, the addresses pointing to other vertices, the marks to indicate if they are visited or not, and the addresses pointing to their adjacent neighbors. Also, I create another link list to store all the vertices values of the adjacent neighbors of each vertex in the graph and its head address would be stored in the link list of each vertices. For reading the coordinates of each vertices from the map file, I simply read one vertex at a time including its coordinate and store it in the link list of the vertices I just built. For reading the edges from the map file, I read two values of one line at a time. What's more, I use linear search function to find the vertex address of the first value and store the second value as the adjacent neighbor of the first value. Similarly, I use linear search function to find the vertex address of the second value and store the first value as the adjacent neighbor of the second value. These two steps would make sure all the adjacent neighbors could be found for each vertex. Finally, I use two while loop to go through the link lists of vertices and adjacent neighbors and print all the adjacent neighbors for each vertex.

For the part of Dijkstra algorithm, I firstly set the distance of the source vertex to zero. Then, I try to find the all the neighbors of the source vertex using adjacent lists and update their

distances from the source vertex if the distances are smaller than their original distance values. Besides, I also update the prev components of adjacent neighbors with the source vertex value. What's more, I marked the source vertex as a visited vertex by setting its visited value to 1. Then, I try to find the vertex who has the least distance in the unvisited list and repeat to do the steps above until the destination vertex is marked as visited. The way I print the shortest path is to use a recursive function to print the values inside the prev of the vertices link lists from source to destination.

Result and improvements

All my program results for the given test case are correct. However, the performance of the running time and the memory usage is not very good. The reason is that I use linear search to find the minimum distance inside the vertices list and the correct address of vertices list for every adjacent neighbor founded, which cost O(n). If I use min heap to find the minimum distance, which cost O(logn) , and the binary search instead of linear search, which cost O(logn) , the running time would be extremely improved. Also, if I use an array to store all the vertices' values and each element inside the array is followed by its corresponding adjacent vertex with its information, this could also optimize the memory usage.

gcc command for adjacent.c: gcc -Werror -Wall -O3 adjacent.c -o adjacent

gcc command for proj3.c: gcc -Werror -Wall -lm -O3 proj3.c -o proj3

Reference:

N. (n.d.). Nkorai/ECE368-Projects. Retrieved from https://github.com/nkorai/ECE368-Projects/blob/master/Project 3 - Dijkstras Algorithm/proj3.c