ECE 368 Project 3 Final Report

Algorithm Description

The program first load the map to memory. Each vertex contains its x, y coordinate, index, temporary distance, predecessor, a linked list that contains all its neighbors and whether it has been visited. Then load the queries and store the index of starting vertex and destination index into an array. Each pair uses two entries. In order to find shortest path for each query, first initialize all vertices. Assign maximum distance to every vertex, delete all predecessors and mark all vertices "unvisited". Copy them to an array which contains all unvisited vertices. Set starting vertex's distance equal to 0.

Next the program starts to look for shortest path. Linearly search and take out the vertex with smallest distance. Mark the vertex as "visited". Then visit every neighbor and calculate corresponding distance and compare to current temporary distance. If a shorter path is found, reassign the distance and predecessor. Keep previous steps until destination vertex is visited or unvisited vertices' set is empty. If destination is still unvisited, that means there's no route from starting point to destination. If destination is visited, follow predecessor to origin and that is the shortest path.

Time and Space Complexity

The program uses additional memory to store vertices as well as edges, so space complexity is O(E + V). Also it takes O(V) to select vertex with shortest distance from unvisit set, so it takes O(V^2 * n) time to find shortest path. (V represents the number of vertices, E represents the number of edges, and n is the number of neighbors for each vertex)

Compilation requirement

This program uses math library so when using GCC please add **–lm** flag.

Limitations and Assumption

When initializing the distance of all vertices, the program uses 0x7FFFFFFF as the maximum distance. This number is the largest number that a signed integer can represent on 64 bit OS. Although some of 32bit machine also use 4 bytes as integer's length, this could not ensure the program runs properly on every machine.

Also when the number of vertices or edges is extremely large, the program may run out of memory. But this program doesn't have mechanism to check if the memory is allocated successfully.

Whenever updating the temporary distance of vertex, the program typecast the result to an integer, which means that there's subtle difference between calculated distance and actual distance. Because program uses previous result to determine next distance, when the route passes by many vertices, the error could be accumulated.

(Note: This program uses part of code from milestone 2 to load vertices and edges and modifies a little bit.)

Shuang Zhai
12/11/2014