Design Document

Mohammad Immam

COP 3530

Dr. Cheryl Resch

For Assignment 2a, we implemented a weighted graph with consecutive integers as data types. I have used adjacency list implementation of graph using C++ vector, where each vertex is aware of its adjacency. We have already implemented Kruskal's algorithm on the graph which is getting all the vertices connected at minimum cost.

For part b of the assignment we need to implement Dijkstra's algorithm on the graph to calculate the shortest path to each vertex path.

Already implemented:

isEdge(int, int) returns if the given vertices are adjacent to each other.

getWeight(int, int) returns the weight of the edge from and to the given vertices

Plan: I will use a vector of map to hold my final table to work around a 3d table. The indices of the vector will contain a map for the respective vertex number. The first value is going to be the distance to that vertex and the second value will contain the previous vertex number. The steps are

1) create a vector of map <int, int>, first int as distance and second int as previous vertex.
2) Initialize all indices to have distance infinity and previous null.
3) Set the distance to source 0 and previous to source as null.
4) Run a loop as many times as many vertex present:
    a. Get the lowest distance vertex to be u.
    b. Run a loop for all adjacent vertices, v of u.
    c. Update the distance of v as needed. [To update the distance: if sum of distance to u and weight of edge u-v, is less than the distance of v, then update the distance value of v.]

PsuedoCode:

Dijstra(int source)

Initialize a vector of map<int, int> table // first int as distance and second int as previous vertex

(for i from 0 to number of vertices)

    Initialize all values of table with table[nth vertex, inf]

table[source]=[0, null] //setting the first value in vector table to have dist 0 and prev null

for (i from 0 to number of vertices)

    u= getLowest(table) //a method implemented to return lowest unvisited vertex from table map

    for (each neighbor v of u)

        if (table[u].first+getWeight(u,v)<table[v].first)

            table[v]=[[u]+getWeight(u,v), u] //change the value saying u's distance+weight is new distance and u is the previous

Now vector table shall contain our result to Dijkstra's algorithm. Each index is the vertex we are talking about, the map obtained at the index has first value as the distance and the second value as previous. We must perform sort on the vector of map based on the second value of the vector and how the question asks. We then print the edges from the vector like how the assignment warrants.

The already implemented isEdge, the vector of adjacency list and getWeight will help us in performing Dijkstra as mentioned above. The adjacency list implementation shall make this routine a bit easier since getting adjacent vertexes from a vertex is just a matter of traversing a map that is part of the vertex. This is my sketch on tackling assignment 2_b. If implemented correctly, together this shall produce the results desired.