SYNOPSIS FOR CG MINI PROJECT

DIJKSTRA'S ALGORITHM

ARVIND MAHTO (1bo15cs011)
IPSITA CHAKRABORTY (1bo15cs032)
VI SEM (SEC-A)

ABSTRACT

Dijkstra's shortest path algorithm is commonly used to solve the single source shortest path problem. For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path algorithm is widely used in network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and Open Shortest Path First (OSPF).

In this project we would create a program using OPENGL library to visualize the algorithm and thus understand it.

OUTLINE

Dijkstra's algorithm has many variants but the most common one is to find the shortest paths from the source vertex to all other vertices in the graph.

Algorithm Steps:

1. Set all vertices distances = infinity except for the source vertex, set the source distance = 0.
2. Push the source vertex in a min-priority queue in the form (distance , vertex), as the comparison in the min-priority queue will be according to vertices distances.
3. Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).

4. Update the distances of the connected vertices to the popped vertex in case of “current vertex distance + edge weight < next vertex distance”, then push the vertex with the new distance to the priority queue.
5. If the popped vertex is visited before, just continue without using it.
6. Apply the same algorithm again until the priority queue is empty.

FUNCTIONALITY

- Various action that can be taken by the user:
    - (a) Add nodes or vertices.
    - (b) Add edges.
    - (c) Delete nodes or vertices.
    - (d) Delete edges.
    - (e) Specify directed or undirected graph.
    - (f) Specify start and destination nodes.
    - (g) Get list of vertices, as shortest path.

A SAMPLE GRAPH

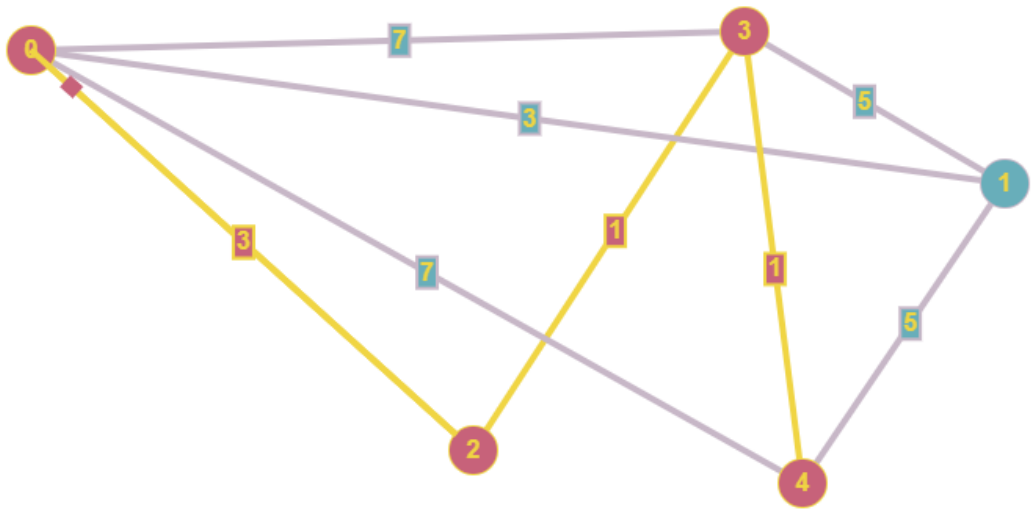

Figure 1: Dijkstra's algorithm being used on a sample graph

IMPLEMENTATION:

The project will be implemented in C++ using OPENGL library. Data structure to be used for the storage of vertices may be decided as project progresses.

REQUIREMENTS:

Recommended Hardware Requirements:

- Processor: 2.142 GHz
- RAM: 4GB
- Hard Disk:20GB
- Monitor

Recommended Software Requirements:

- Operating System: Linux / Windows 7
- IDE: Visual Studio / Eclipse
- C++ Compiler