Lab 11: Implementation of Dijkstra’s Algorithm

**Shortest Path - Dijkstra’s Algorithm**

Dijkstra algorithm is also called single source shortest path algorithm. It is based on greedy technique. The algorithm maintains a list visited[ ] of vertices, whose shortest distance from the source is already known.

If visited[1], equals 1, then the shortest distance of vertex i is already known. Initially, visited[i] is marked as, for source vertex.

At each step, we mark visited[v] as 1. Vertex v is a vertex at shortest distance from the source vertex. At each step of the algorithm, shortest distance of each vertex is stored in an array distance[ ].

**Dijkstra’s Algorithm**

1. Create cost matrix C[ ][ ] from adjacency matrix adj[ ][ ]. C[i][j] is the cost of going from vertex i to vertex j. If there is no edge between vertices i and j then C[i][j] is infinity.

2. Array visited[ ] is initialized to zero.

               for(i=0;i<n;i++)

                              visited[i]=0;

3. If the vertex 0 is the source vertex then visited[0] is marked as 1.

4. Create the distance matrix, by storing the cost of vertices from vertex no. 0 to n-1 from the source vertex 0.

               for(i=1;i<n;i++)

                              distance[i]=cost[0][i];

Initially, distance of source vertex is taken as 0. i.e. distance[0]=0;

5. for(i=1;i<n;i++)

– Choose a vertex w, such that distance[w] is minimum and visited[w] is 0. Mark visited[w] as 1.

– Recalculate the shortest distance of remaining vertices from the source.

– Only, the vertices not marked as 1 in array visited[ ] should be considered for recalculation of distance. i.e. for each vertex v

               if(visited[v]==0)

                              distance[v]=min(distance[v],

                              distance[w]+cost[w][v])