**PROGRAM TO SIMULATE LINK STATE ROUTING ALGORITHM**

**Aim:**

To write a java program to simulate Link State Routing Algorithm.

**Algorithm**:

**1)** Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.
**2)** Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
**3)** While sptSet doesn’t include all vertices
….**a)** Pick a vertex u which is not there in sptSet and has minimum distance value.
….**b)** Include u to sptSet.
….**c)** Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

**Program:**

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class ShortestPath

{

 static final int V = 4;

 int minDistance(int dist[], Boolean sptSet[])

 {

 int min = Integer.MAX\_VALUE, min\_index = -1;

 for (int v = 0; v < V; v++)

 if (sptSet[v] == false && dist[v] <= min)

{

 min = dist[v];

 min\_index = v;

 }

 return min\_index;

 }

 void printSolution(int dist[], int n)

 {

 System.out.println("Vertex Distance from Source");

 for (int i = 0; i < V; i++)

 System.out.println(i + " tt " + dist[i]);

 }

 void dijkstra(int graph[][], int src)

 {

 int dist[] = new int[V];

 Boolean sptSet[] = new Boolean[V];

 for (int i = 0; i < V; i++)

{

 dist[i] = Integer.MAX\_VALUE;

 sptSet[i] = false;

 }

 dist[src] = 0;

 for (int count = 0; count < V - 1; count++)

 {

 int u = minDistance(dist, sptSet);

 sptSet[u] = true;

 for (int v = 0; v < V; v++)

 if (!sptSet[v] && graph[u][v] != 0 &&

 dist[u] != Integer.MAX\_VALUE &&

dist[u] + graph[u][v] < dist[v])

 dist[v] = dist[u] + graph[u][v];

 }

 printSolution(dist, V);

 }

 // Driver method

 public static void main(String[] args)

 {

 /\* Let us create the example graph discussed above \*/

 int graph[][] = new int[][] {

 {0,5,10,0},

 {5,0,3,11},

 {10,3,0,2},

 {0,11,2,0},

 };

 ShortestPath t = new ShortestPath();

 t.dijkstra(graph, 0);

 }

}

Output:

Vertex Distance from Source

0 tt 0

1 tt 5

2 tt 8

3 tt 10