

DIJKSTRA'S ALGORITHM

```
import java.util.*;

public class DijkstraAlgorithm {

    static class Node {

        int id;

        int distance;

        Node(int id, int distance) {

            this.id = id;

            this.distance = distance;

        }

    }

    static int[][] graph = {

        {0, 4, 0, 0, 0, 0, 0, 8, 0},

        {4, 0, 8, 0, 0, 0, 0, 11, 0},

        {0, 8, 0, 7, 0, 4, 0, 0, 2},

        {0, 0, 7, 0, 9, 14, 0, 0, 0},

        {0, 0, 0, 9, 0, 10, 0, 0, 0},

        {0, 0, 4, 14, 10, 0, 2, 0, 0},

        {0, 0, 0, 0, 0, 2, 0, 1, 6},

        {8, 11, 0, 0, 0, 0, 1, 0, 7},

        {0, 0, 2, 0, 0, 0, 6, 7, 0}

    };

    static int V = graph.length;

    static void dijkstra(int source) {
```

```

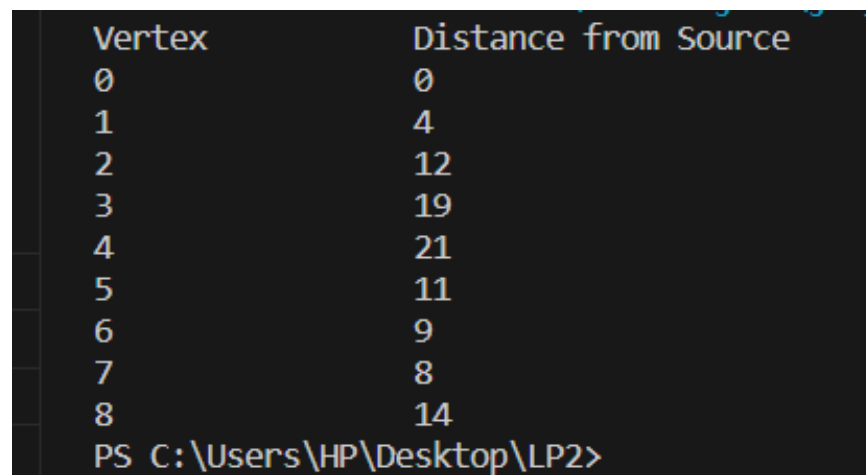
int[] dist = new int[V];
boolean[] visited = new boolean[V];
Arrays.fill(dist, Integer.MAX_VALUE);
dist[source] = 0;
for (int count = 0; count < V - 1; count++) {
    int u = minDistance(dist, visited);
    visited[u] = true;
    for (int v = 0; v < V; v++) {
        if (!visited[v] && graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE
&& dist[u] + graph[u][v] < dist[v]) {
            dist[v] = dist[u] + graph[u][v];
        }
    }
}
printSolution(dist);
}

static int minDistance(int[] dist, boolean[] visited) {
    int min = Integer.MAX_VALUE;
    int minIndex = -1;
    for (int v = 0; v < V; v++) {
        if (!visited[v] && dist[v] <= min) {
            min = dist[v];
            minIndex = v;
        }
    }
    return minIndex;
}

```

```
static void printSolution(int[] dist) {  
    System.out.println("Vertex \t\t Distance from Source");  
    for (int i = 0; i < V; i++) {  
        System.out.println(i + " \t\t " + dist[i]);  
    }  
}  
  
public static void main(String[] args) {  
    dijkstra(0);  
}  
}
```

OUTPUT:



Vertex	Distance from Source
0	0
1	4
2	12
3	19
4	21
5	11
6	9
7	8
8	14

PS C:\Users\HP\Desktop\LP2>