COMPUTER NETWORKS LAB (CSL5403)

Name: Lakhan Kumawat

Roll: 1906055

Program: B.Tech CSE
(5th Sem JUL-DEC 2021)

Assignment - 6

PROGRAM CODE JAVA:

```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.PriorityQueue;

public class LinkStateRouting {

    static class DijkstrasPair implements
Comparable<DijkstrasPair >{
        String vtx;
        String path;
        int cost;

        DijkstrasPair(String vtx, String path, int cost)
        {
            this.vtx=vtx;
            this.path=path;
            this.cost=cost;
        }

        @Override
        public int compareTo(DijkstrasPair o) {
            return this.cost-o.cost;
        }
    }

    static class Graph{
        HashMap<String, HashMap<String,Integer>> graph;

        Graph(){
            graph = new HashMap<>();
        }
```

```
        public void addEdge(String src, String dest, int cost)
        {
            if(!this.graph.containsKey(src))
                this.graph.put(src, new HashMap<>());
            if(!this.graph.containsKey(dest))
                this.graph.put(dest, new HashMap<>());

            this.graph.get(src).put(dest, cost);
            this.graph.get(dest).put(src, cost);
        }

        // This method will implement Dijkstra's algorithm find
shortest path from any vertex
        public void printShortestPath(String source) {

            HashMap<String, DijkstrasPair> map = new
HashMap<>();
            PriorityQueue<DijkstrasPair> pq = new
PriorityQueue<>();

            for(String vtx : graph.keySet())
            {

              DijkstrasPair dp = vtx != source ? new
DijkstrasPair(vtx, "", Integer.MAX_VALUE) :
                                                 new
DijkstrasPair(vtx, "C", 0);
              pq.add(dp);
              map.put(vtx, dp);
            }

            while(!pq.isEmpty())
            {
                DijkstrasPair dp = pq.remove();
                map.remove(dp.vtx);
```

```
                for(String nbr :
this.graph.get(dp.vtx).keySet())
                {
                    if(map.containsKey(nbr))
                    {
                        int oc = map.get(nbr).cost;
                        int nc = dp.cost +
this.graph.get(dp.vtx).get(nbr);

                        if(nc < oc)
                        {
                            pq.remove(map.get(nbr));
                            map.get(nbr).cost = nc;
                            map.get(nbr).path = dp.path + "->"
+ nbr;
                            pq.add(map.get(nbr));

                        }
                    }
                }

                System.out.println("reach router " + dp.vtx + "
via path "+dp.path+" with cost "+dp.cost);
            }

        }
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Graph graph = new Graph();
        graph.addEdge("A", "B", 5);
        graph.addEdge("A", "D", 16);
        graph.addEdge("A", "E", 7);
        graph.addEdge("B", "C", 8);
        graph.addEdge("B", "D", 12);
```

```
        graph.addEdge("B", "E", 6);
        graph.addEdge("B", "F", 2);
        graph.addEdge("C", "D", 6);
        graph.addEdge("C", "E", 3);
        graph.addEdge("D", "E", 9);
        graph.addEdge("D", "F", 10);
        graph.addEdge("E", "F", 4);

        System.out.println("Shortest Paths from Router --
>  C\n");
        graph.printShortestPath("C");


    }

}
```

OUTPUT SCREENSHOT:

```
C:/users/Lakhan Kumawat/Computer Networks Lab/LinkStateRouting.exe
Shortest Paths from Router -->  C

reach router C via path C with cost 0
reach router E via path C->E with cost 3
reach router D via path C->D with cost 6
reach router F via path C->E->F with cost 7
reach router B via path C->B with cost 8
reach router A via path C->E->A with cost 10

[Done] exited with code=0 in 1.094 seconds
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5403

End Of Assignment