Lab Experiment 6

Write a program to implement Dijkstra Shortest path routing protocol.

Program:

```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
void dijkstras(vector<vector<int>> graph, int start, int noOfRouters)
{
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,
int>>> pq;
    vector<int> distances(noOfRouters, INT_MAX);
    vector<int> isVisited(noOfRouters, false);
    distances[start] = 0;
    pq.push({0, start});
    pair<int, int> present;
    while (!pq.empty())
    {
        present = pq.top();
        pq.pop();
        if (isVisited[present.second])
        {
            continue;
        }
        isVisited[present.second] = true;
        for (int i = 0; i < noOfRouters; i++)
        {
            if (!isVisited[i] && graph[present.second][i] &&
distances[present.second] != INT_MAX && distances[present.second] +
graph[present.second][i] < distances[i])
            {
                distances[i] = distances[present.second] +
graph[present.second][i];
                pq.push({distances[i], i});
            }
        }
    }
    cout << "Distances of the routers from the starting router " << start <<
endl;
    for (int i = 0; i < noOfRouters; i++)
    {
        cout << i << "->" << distances[i] << endl;
    }
}
int main()
{
```

```
    int noOfRouters;
    cout << "Enter the no of Routers: ";
    cin >> noOfRouters;
    cout << "Enter the values: " << endl;
    vector<vector<int>> graph(noOfRouters, vector<int>(noOfRouters, 0));
    for (int i = 0; i < noOfRouters; i++)
    {
        for (int j = 0; j < noOfRouters; j++)
        {
            cin >> graph[i][j];
        }
    }
    int startNode;
    cout << "Enter the starting Router number: ";
    cin >> startNode;
    dijkstras(graph, startNode, noOfRouters);
}
```

Output:

```
Enter the no of Routers: 5
Enter the values:
0 2 0 0 3
2 0 4 0 0
0 4 0 5 0
0 0 5 0 1
3 0 0 1 0
Enter the starting Router number: 0
Distances of the routers from the starting router 0
0->0
1->2
2->6
3->4
4->3
```

Write a program to implement Distance Vector Routing.

Program:

```
#include <iostream>
#include <climits>
#include <vector>

using namespace std;

class Graph
{
public:
    int n;
    vector<vector<int>> adj;
    vector<vector<int>> routing_table;

    Graph(int x) : n(x), adj(n, vector<int>(n)), routing_table(n,
vector<int>(n, INT_MAX))
    {
        for (int i = 0; i < n; i++)
        {
            routing_table[i][i] = 0;
        }
    }

    void create()
    {
        cout << "Enter adjacency matrix:\n";
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                int x;
                cin >> x;
                adj[i][j] = x;
            }
        }
    }

    void display()
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                cout << adj[i][j] << " ";
            }
            cout << endl;
```

```
        }
    }

    void create_routing_table()
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (adj[i][j] != 0)
                {
                    routing_table[i][j] = adj[i][j];
                }
            }
        }

        for (int i = 0; i < n; i++)
        {
            cout << "\nDistance vector of router " << i << endl;
            for (int j = 0; j < n; j++)
            {
                cout << routing_table[i][j] << " ";
            }
            cout << "\n";
        }

        cout << "\n\n\n";
    }

    void distance_vector()
    {
        while (true)
        {
            int flag = 1;
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    if (adj[i][j] != 0)
                    {
                        for (int k = 0; k < n; k++)
                        {
                            if (routing_table[i][j] == INT_MAX ||
routing_table[j][k] == INT_MAX)
                            {
                                continue;
                            }
```

```
                            if (routing_table[i][k] > routing_table[j][k] +
routing_table[j][i])
                            {
                                routing_table[i][k] = routing_table[j][k] +
routing_table[j][i];
                                flag = 0;
                            }
                        }
                    }
                }

                cout << "\n\n\n";
                for (int z = 0; z < n; z++)
                {
                    cout << "\nDistance vector of router " << i << endl;
                    for (int d = 0; d < n; d++)
                    {
                        cout << routing_table[z][d] << " ";
                    }
                    cout << "\n";
                }
            }

            if (flag == 1)
            {
                break;
            }
        }

        cout << "\n\n\n";
        for (int i = 0; i < n; i++)
        {
            cout << "\nDistance vector of router " << i << endl;
            for (int j = 0; j < n; j++)
            {
                cout << routing_table[i][j] << " ";
            }
            cout << "\n";
        }
    }
};

int main()
{
    cout << "Enter the number of routers: ";
    int n;
    cin >> n;
    Graph g(n);
```

```
    g.create();
    g.display();
    g.create_routing_table();
    g.distance_vector();
    return 0;
}
```

Output:

```
Distance vector of router 0
0 2 6 4 3

Distance vector of router 1
2 0 4 6 5

Distance vector of router 2
6 4 0 5 6

Distance vector of router 3
4 6 5 0 1

Distance vector of router 4
3 5 6 1 0
```