COMPUTER NETWORKS
LINK STATE ALGORITHM

Additional Lab Implementation

Jatin Patil
Roll Number **29**
1032221837
Panel B Computer Science and Engineering

Code -

```
#include<iostream>
#include<climits>
#include<iomanip>
#include<cctype>
using namespace std;
class node
{
    char name;
    node *next;
    friend class graph;
};

class graph
{
    node *head[25];
    int cost[25][25];
    public:
    int n;
    graph();
    void create();
    void display_adjacency_list();
    void display_cost();
    void shortest_distance(int src);
};

graph :: graph()
{
    cout << "Please enter the number of nodes : ";
    cin >> n;
    for(int i = 0; i < n; i++)
        {
          for(int j = 0; j < n; j++)
                {
                    if(i == j)
                        cost[i][j] = 0;
                    else
                        cost[i][j] = INT_MAX;
                }
        }
}

void graph :: create()
{
    //head = new node(n);
    for(int i = 0; i < n; i++)
```

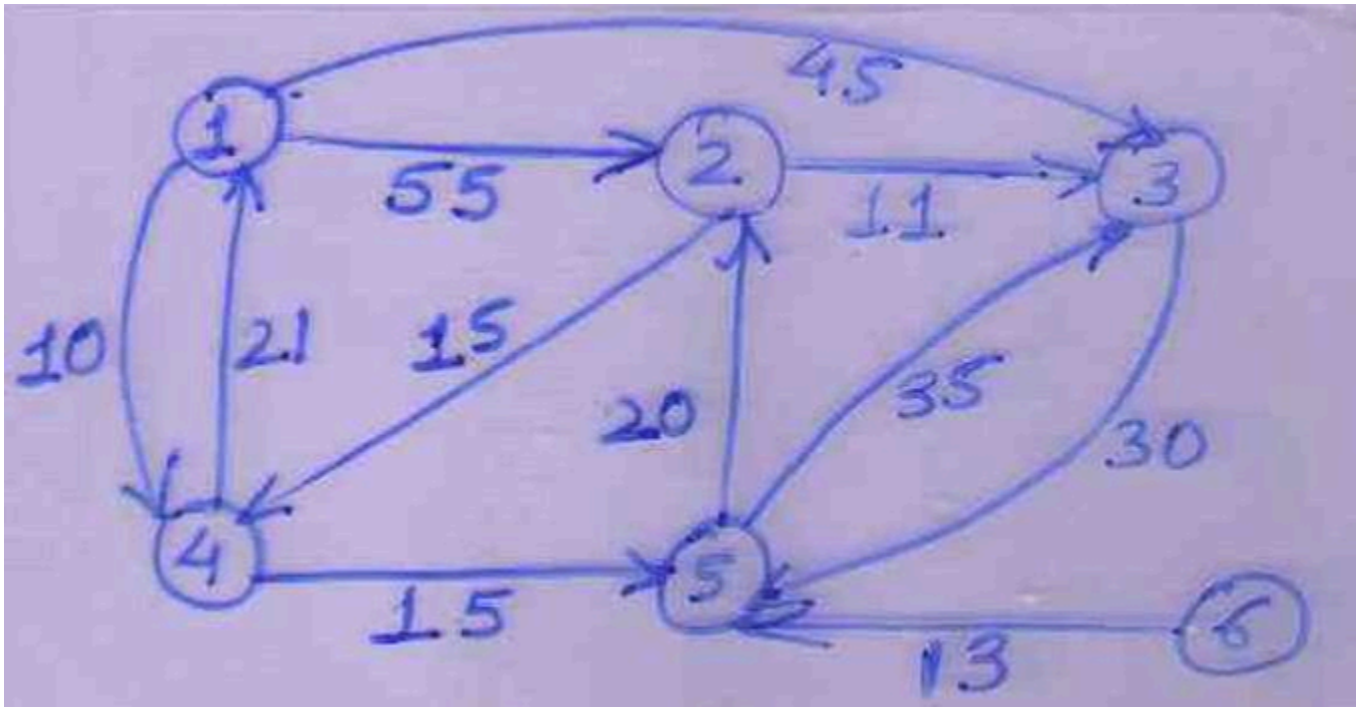
```
    {
        head[i] = new node;
        head[i] -> name = (char)(i + 65);
        head[i] -> next = NULL;
        cout << "\033[1;32mNode " << head[i] -> name << "\033[1;0m\n";

        //adding the adjacent nodes
        char ch;
        cout << "Do you want to add any adjacent nodes ? ";
        cin >> ch;
        node *temp = head[i];
        while(ch == 'y' || ch == 'Y')
        {
            int c;
            node *nnode = new node;
            nnode -> next = NULL;
            cout << "Enter adjacent node name : ";
            cin >> nnode -> name;
            cout << "Enter associated cost : ";
            cin >> c;
            temp -> next = nnode;
            temp = temp -> next;

            //assigning the cost adjacency matrix
            int j = nnode -> name - 65;
            cost[i][j] = c;

            cout << "Do you want to add any adjacent nodes ? ";
            cin >> ch;
        }
        cout << "\n";
    }
}

void graph :: display_adjacency_list()
{
    //displaying the heads and their adjacent nodes
    cout << "HEAD -> adjacent nodes\n";
    for(int i = 0; i < n; i++)
    {
        cout << head[i] -> name;
        node *temp = head[i] -> next;
        if(temp == NULL)
            cout << " -> None";
        else
        {
            while(temp != NULL)
```

```
                {
                        cout << " -> " << temp -> name;
                        temp = temp -> next;
                }
        }
        cout << "\n";
    }
}

void graph :: display_cost()
{
    //displaying the cost adjacency matrix
    cout << "Cost Adjacency Matrix\n   ";
    for(int i = 0; i < n; i++)
        cout << (char)(i + 65) << "  ";
    cout << "\n";
    for(int i = 0; i < n; i++)
    {
        cout << (char)(i + 65) << " ";
        for(int j = 0; j < n; j++)
            {
                if(cost[i][j] != INT_MAX)
                    cout << std::setw(2) << cost[i][j] << " ";
                else
                    cout << " x" << " ";
            }
        cout << "\n";
    }
}

void graph :: shortest_distance(int src)
{
    //applying djikstras algorithm to cal calculate the shortest paths to each router

    int s[25];
    int p[25];
    int dist[25];
    for(int i = 0; i < n; i++)
    {
        s[i] = 0;
        dist[i] = cost[src][i];
        if(cost[src][i] != INT_MAX)
            p[i] = src;
        else
            p[i] = -1;
    }
    //starting from the given vertex
```

```
    s[src] = 1;
    dist[src] = 0;

    for(int j = 1; j < n; j++)
    {
        int min = INT_MAX;
        int u = 0;
        //choose u from among the set of vertices such that d[u] is min and s[u] hasnt been visited
        for(int k = 0; k < n; k++)
        {
            if(s[k] != 1 && dist[k] < min)
            {
                u = k;
                min = dist[k];
            }

        }
        s[u] = 1;
        for(int k = 0; k < n; k++)
        {
            if(cost[u][k] != INT_MAX)
            if(s[k] == 0 && dist[k] > dist[u] + cost[u][k])
            {
                dist[k] = dist[u] + cost[u][k];
                p[k] = u;
            }
        }
    }
        cout << "Routing Table for " << (char)(src + 65) << "\n";
        cout << "+-----------+--------+\n|Destination|Distance|\n+-----------+--------+\n";
    for(int i = 0; i < n; i++)
    {
        if (dist[i] !=  INT_MAX)
            cout << "|"<<setw(11) << (char)(i + 65) << "|" << setw(8) << dist[i] << "|\n";
        else
                cout << "|" << setw(11) << (char)(i + 65) << "|" << setw(8) << " x" << "|\n";
    }
        cout << "+-----------+--------+";
}
int main()
{
    cout << "\n\033[1;32m    LINK STATE ALGORITHM   \033[1;0m"<<endl;
    graph g;
        char src;

    g.create();
    //g.display_adjacency_list();
```

```
    //g.display_cost();

        for(;;)
        {
            cout << "\n\033[1;32m          LINK STATE ALGORITHM   \033[1;0m"<<endl;
            cout << "\033[1;93mWhich router table do you want to see ? \nPress 0 for exit.\033[1;0m\n";
            cin >> src;
            if(src == '0')
            {
                cout << "\033[1;31mEXITING\033[1;0m";
                exit(0);
            }
            else if(src >= 65 && src <= (g.n - 1 +65) || src >= 97 && src <= (g.n - 1 + 97))
                g.shortest_distance((int)(toupper(src) - 65));
            else
                cout << "Please enter a valid input.";
            cout << "\n";
        }
    return 0;
}
```

Graph used for scenario -



Note - 1,2,3,4,5,6 have been used as A,B,C,D,E,F respectively in the program.

Output Screenshots -

```
PS C:\Users\DELL\desktop> g++ linkstate.cpp
PS C:\Users\DELL\desktop> ./a.exe

        LINK STATE ALGORITHM
Please enter the number of nodes : 6
Node A
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : B
Enter associated cost : 55
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : C
Enter associated cost : 45
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : D
Enter associated cost : 10
Do you want to add any adjacent nodes ? N

Node B
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : C
Enter associated cost : 11
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : D
Enter associated cost : 15
Do you want to add any adjacent nodes ? N

Node C
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : E
Enter associated cost : 30
Do you want to add any adjacent nodes ? N
```

```
Node D
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : A
Enter associated cost : 21
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : E
Enter associated cost : 15
Do you want to add any adjacent nodes ? N

Node E
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : B
Enter associated cost : 20
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : C
Enter associated cost : 35
Do you want to add any adjacent nodes ? N

Node F
Do you want to add any adjacent nodes ? Y
Enter adjacent node name : E
Enter associated cost : 13
Do you want to add any adjacent nodes ? N


        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
```

```
        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
A
Routing Table for A
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|       0|
|          B|      45|
|          C|      45|
|          D|      10|
|          E|      25|
|          F|       x|
+-----------+--------+

        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
B
Routing Table for B
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|      36|
|          B|       0|
|          C|      11|
|          D|      15|
|          E|      30|
|          F|       x|
+-----------+--------+
```

```
        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
C
Routing Table for C
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|      86|
|          B|      50|
|          C|       0|
|          D|      65|
|          E|      30|
|          F|       x|
+-----------+--------+

        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
D
Routing Table for D
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|      21|
|          B|      35|
|          C|      46|
|          D|       0|
|          E|      15|
|          F|       x|
+-----------+--------+
```

```
        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
E
Routing Table for E
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|      56|
|          B|      20|
|          C|      31|
|          D|      35|
|          E|       0|
|          F|       x|
+-----------+--------+

        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
F
Routing Table for F
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|      69|
|          B|      33|
|          C|      44|
|          D|      48|
|          E|      13|
|          F|       0|
+-----------+--------+
```

```
Routing Table for F
+-----------+--------+
|Destination|Distance|
+-----------+--------+
|          A|      69|
|          B|      33|
|          C|      44|
|          D|      48|
|          E|      13|
|          F|       0|
+-----------+--------+

        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
G
Please enter a valid input.

        LINK STATE ALGORITHM
Which router table do you want to see ?
Press 0 for exit.
0
EXITING
PS C:\Users\DELL\desktop>
```