Computer Networks
Lab Continuous Assessment

Name: Parth Pawar

Roll no: 19

PRN: 1032221829

Panel: C

LCA Link: https://github.com/Cryptic101104/LinkStateAlgorithm.git

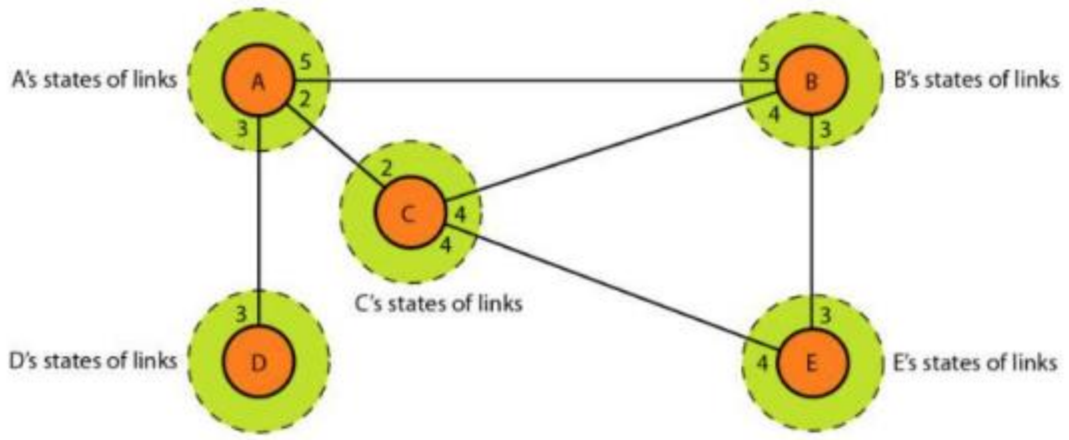
Problem Statement:

Write a program (C/C++/python/java) to implement Link State

Theory:

The Link State Algorithm is a pivotal routing algorithm in computer networking, ensuring efficient packet routing by providing routers with comprehensive knowledge of the network's topology. Through Link State Packets (LSPs), routers exchange information about neighbouring nodes and links. Using Dijkstra's Algorithm, each router calculates the shortest paths to all other routers, forming a shortest path tree. This enables routers to make optimal routing decisions. Continuously updating LSPs in response to network changes ensures resilience and adaptability. Overall, the Link State Algorithm optimizes network performance by enabling routers to make informed and efficient routing decisions based on up-to-date network topology information.

Input:



<u>Code (Without Web-Interface):</u>

```python
def minDistance(dist, present, n):
    min_val = float('inf')
    min_index = -1
    for i in range(n):
        if not present[i] and dist[i] <= min_val:
            min_val = dist[i]
            min_index = i
    return min_index

def displayMinDistance(routers, dist, n):
    print("Router \t Distance from src")
    for i in range(n):
        print(routers[i], "\t\t", dist[i])

def dijkstra(routers, table, src, n):
    dist = [float('inf')] * n
    present = [False] * n
    dist[src] = 0
    for _ in range(n - 1):
        u = minDistance(dist, present, n)
        present[u] = True
        for j in range(n):
            if not present[j] and table[u][j] != 0 and dist[u] != float('inf')
and dist[u] + table[u][j] < dist[j]:
                dist[j] = dist[u] + table[u][j]
    displayMinDistance(routers, dist, n)

def linkStateTable(routers, table, n):
    for i in range(n):
        LST = {}
        for j in range(n):
            if table[i][j] != 0:
                LST[routers[j]] = table[i][j]
        print("\nLink State Table for", routers[i], ":")
        for router, distance in LST.items():
            print(router + ":", distance)

n = int(input("Enter number of routers: "))
routers = []
table = [[0] * n for _ in range(n)]
src = int(input("Enter source router: "))
for i in range(n):
    x = input("Enter name of router {}: ".format(i+1))
    routers.append(x)
ch = 'y'
while ch == 'y':
    r1 = int(input("Enter router 1: "))
    r2 = int(input("Enter router 2: "))
    d = int(input("Enter distance: "))
    table[r1][r2] = d
    table[r2][r1] = d
    ch = input("Enter more links?(y/n): ")
print("\nDistance table:")
for row in table:
    print(" ".join(map(str, row)))
print("\n\nLink State Tables:")
linkStateTable(routers, table, n)
print("\n\nDijkstra's Algorithm:")
dijkstra(routers, table, src, n)
```

Input:

```
Enter number of routers: 5

Enter source router: 0

Enter name of router 1: a

Enter name of router 2: b

Enter name of router 3: c

Enter name of router 4: d

Enter name of router 5: e

Enter router 1: 0

Enter router 2: 1

Enter distance: 5

Enter more links?(y/n): y

Enter router 1: 0

Enter router 2: 2

Enter distance: 2

Enter more links?(y/n): y

Enter router 1: 0

Enter router 2: 3

Enter distance: 3

Enter more links?(y/n): y

Enter router 1: 1

Enter router 2: 2

Enter distance: 4
Enter more links?(y/n): y
Enter router 1: 1
Enter router 2: 4
Enter distance: 3
Enter more links?(y/n): y
Enter router 1: 2
Enter router 2: 4
Enter distance: 4
Enter more links?(y/n): n
```

Output:

```
Distance table:
0 5 2 3 0
5 0 4 0 3
2 4 0 0 4
3 0 0 0 0
0 3 4 0 0


Link State Tables:

Link State Table for a :
b: 5
c: 2
d: 3

Link State Table for b :
a: 5
c: 4
e: 3

Link State Table for c :
```

```
a: 2
b: 4
e: 4

Link State Table for d :
a: 3

Link State Table for e :
b: 3
c: 4



Dijkstra's Algorithm:
Router          Distance from src
a               0
b               5
c               2
d               3
e               6

=== Code Execution Successful ===
```

Code(With Web-Interface):

Main.py:

```python
from flask import Flask, render_template, request
import sys

app = Flask(__name__)

def minDistance(dist, present, n):
    min_val = sys.maxsize
    min_index = -1
    for i in range(n):
        if not present[i] and dist[i] <= min_val:
            min_val = dist[i]
            min_index = i
    return min_index

def displayMinDistance(routers, dist, n):
    result = "Router \t Distance from src<br>"
    for i in range(n):
        result += f"{routers[i]} \t\t {dist[i]}<br>"
    return result

def dijkstra(routers, table, src, n):
    dist = [sys.maxsize] * n
    present = [False] * n
    dist[src] = 0
    for _ in range(n - 1):
        u = minDistance(dist, present, n)
        present[u] = True
        for j in range(n):
            if not present[j] and table[u][j] != 0 and dist[u] != sys.maxsize
and dist[u] + table[u][j] < dist[j]:
                dist[j] = dist[u] + table[u][j]
    return displayMinDistance(routers, dist, n)

def linkStateTable(routers, table, n):
    result = ""
    for i in range(n):
        LST = {}
        for j in range(n):
            if table[i][j] != 0:
                LST[routers[j]] = table[i][j]
        result += f"<br>Link State Table for {routers[i]} :<br>"
        for router, distance in LST.items():
            result += f"{router}: {distance}<br>"
    return result

@app.route('/')
def index():
    return render_template('index.html', n=5)


@app.route('/result', methods=['POST'])
def result():
    n = int(request.form['n'])
    routers = []
    table = [[0] * n for _ in range(n)]
    src = int(request.form['src'])
    for i in range(n):
        x = request.form[f'router_{i}']
        routers.append(x)
    for i in range(n):
        for j in range(n):
            table[i][j] = int(request.form.get(f'table_{i}_{j}', 0))
    link_state_tables = linkStateTable(routers, table, n)
    dijkstra_result = dijkstra(routers, table, src, n)
    return render_template('result.html', link_state_tables=link_state_tables,
dijkstra_result=dijkstra_result)

if __name__ == '__main__':
    app.run(debug=True)
```

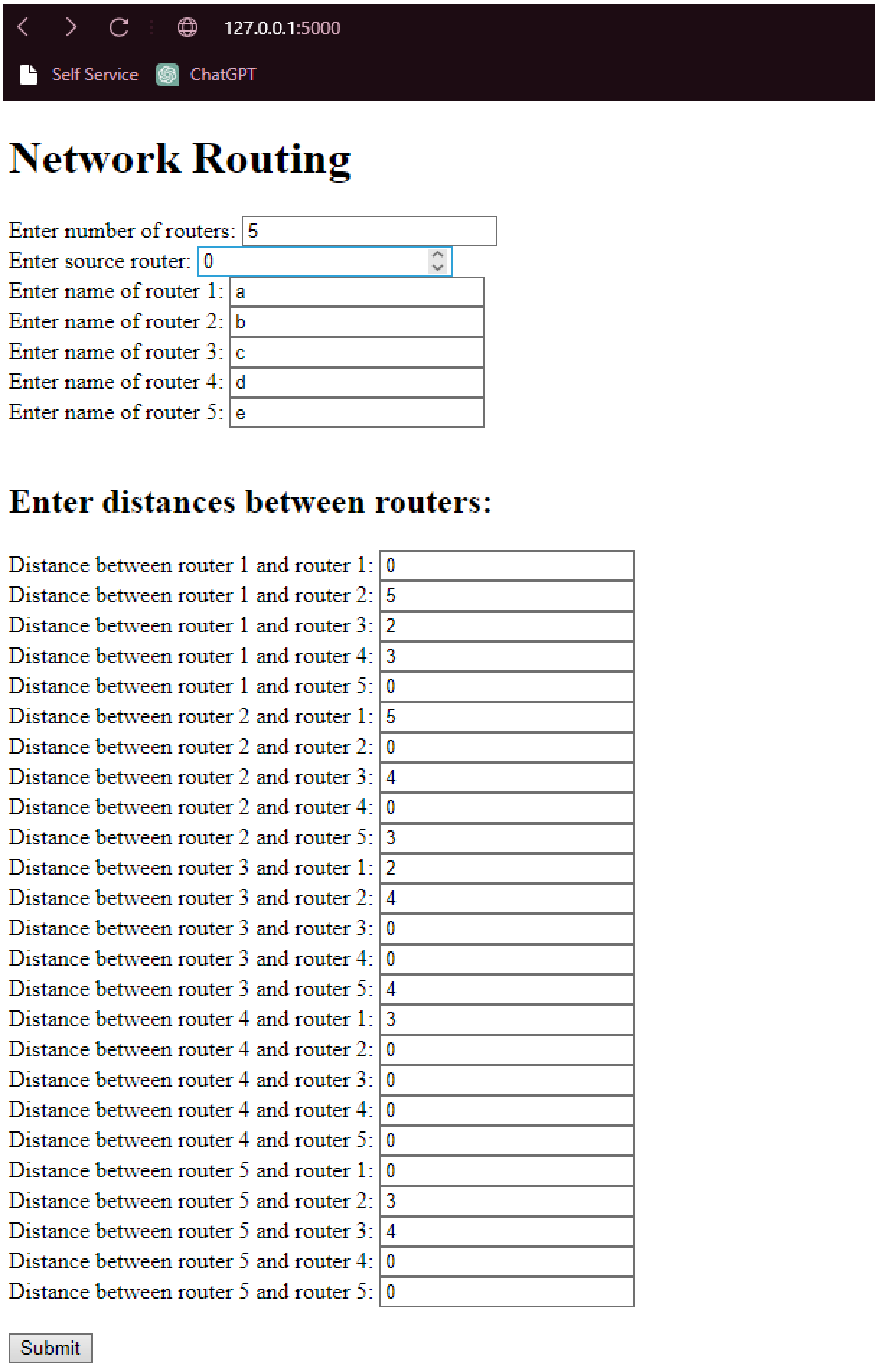
index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Network Routing</title>
</head>
<body>
    <h1>Network Routing</h1>
    <form action="/result" method="post">
        <label for="n">Enter number of routers:</label>
        <input type="number" name="n" id="n"><br>
        <label for="src">Enter source router:</label>
        <input type="number" name="src" id="src"><br>
        {% for i in range(n) %}
            <label for="router_{{ i }}">Enter name of router {{ i+1
}}:</label>
            <input type="text" name="router_{{ i }}" id="router_{{ i }}"><br>
        {% endfor %}
        <br>
        <h2>Enter distances between routers:</h2>
        {% for i in range(n) %}
            {% for j in range(n) %}
                <label for="table_{{ i }}_{{ j }}">Distance between router {{
i+1 }} and router {{ j+1 }}:</label>
                <input type="number" name="table_{{ i }}_{{ j }}" id="table_{{
i }}_{{ j }}"><br>
            {% endfor %}
        {% endfor %}
        <br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Result.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result</title>
</head>
<body>
    <h1>Result</h1>
    <h2>Link State Tables:</h2>
    {{ link_state_tables | safe }}
    <h2>Dijkstra's Algorithm:</h2>
    {{ dijkstra_result | safe }}
</body>
</html>
```

Input:



127.0.0.1:5000

Self Service ChatGPT

Network Routing

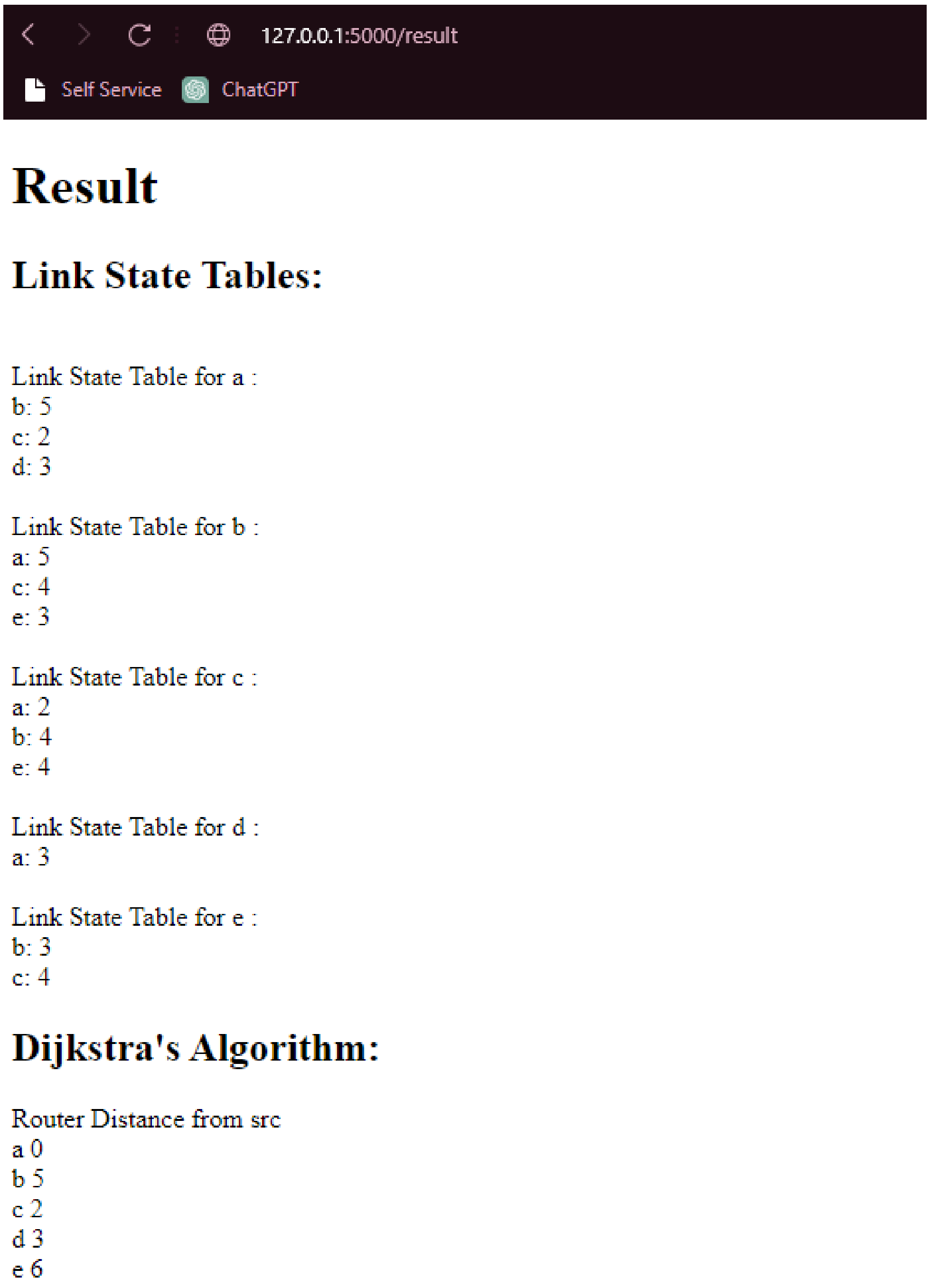
Enter number of routers:	5
Enter source router:	0
Enter name of router 1:	a
Enter name of router 2:	b
Enter name of router 3:	c
Enter name of router 4:	d
Enter name of router 5:	e

Enter distances between routers:

Distance between router 1 and router 1:	0
Distance between router 1 and router 2:	5
Distance between router 1 and router 3:	2
Distance between router 1 and router 4:	3
Distance between router 1 and router 5:	0
Distance between router 2 and router 1:	5
Distance between router 2 and router 2:	0
Distance between router 2 and router 3:	4
Distance between router 2 and router 4:	0
Distance between router 2 and router 5:	3
Distance between router 3 and router 1:	2
Distance between router 3 and router 2:	4
Distance between router 3 and router 3:	0
Distance between router 3 and router 4:	0
Distance between router 3 and router 5:	4
Distance between router 4 and router 1:	3
Distance between router 4 and router 2:	0
Distance between router 4 and router 3:	0
Distance between router 4 and router 4:	0
Distance between router 4 and router 5:	0
Distance between router 5 and router 1:	0
Distance between router 5 and router 2:	3
Distance between router 5 and router 3:	4
Distance between router 5 and router 4:	0
Distance between router 5 and router 5:	0

Submit

Output:



```
Result

Link State Tables:

Link State Table for a :
b: 5
c: 2
d: 3

Link State Table for b :
a: 5
c: 4
e: 3

Link State Table for c :
a: 2
b: 4
e: 4

Link State Table for d :
a: 3

Link State Table for e :
b: 3
c: 4

Dijkstra's Algorithm:

Router Distance from src
a 0
b 5
c 2
d 3
e 6
```