Itisha Desai

Sem – 5

Branch – Cyber Security

Batch – CSE54

Enrollment No. – 22162171006

Algorithm Analysis and Design
Practical-11

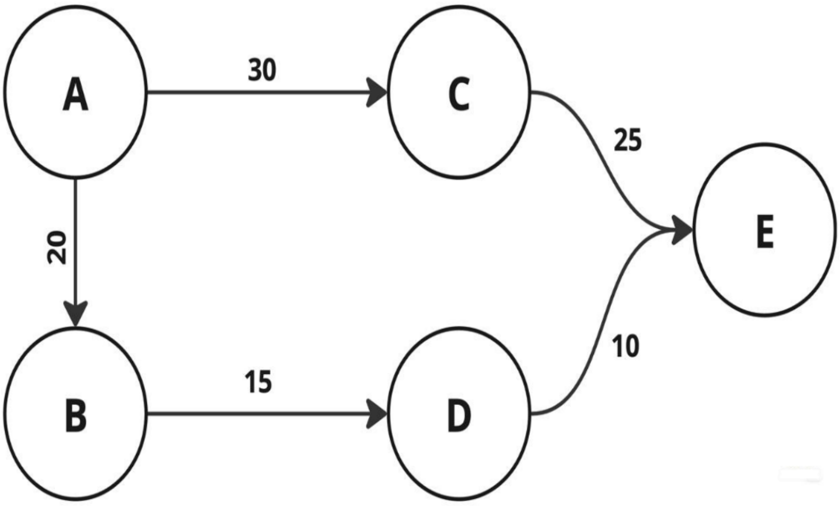
Question :-

AIM:

A government official needs to visit several cities within a state. To minimize travel costs, they want to find the shortest path between their starting city and each destination city.

Task:

Given a graph representing the cities and their connecting roads, determine the minimum cost path from a given starting city to all other cities.



Input:

Enter total number of nodes: 5

Enter the node from where you want to calculate the distance: A

Enter Data (Weight):

	A	*B*	*C*	*D*	*E*
A	0	20	30	∞	∞
B	∞	0	∞	15	∞
C	∞	∞	0	∞	25
D	∞	∞	∞	0	10
E	∞	∞	∞	∞	0

Output:

	A	*B*	*C*	*D*	*E*
A	0	20	30	35	45
B	∞	0	∞	15	25
C	∞	∞	0	∞	25
D	∞	∞	∞	0	10
E	∞	∞	∞	∞	0

OR

Source	Destination	Cost
A	A	0
	B	20
	C	30
	D	35
	E	45

Code :-

App.py:

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

app = Flask(__name__)

def dijkstra(graph, start_node):
    n = len(graph)
    visited = [False] * n
    distance = [sys.maxsize] * n
    previous = [-1] * n
    distance[start_node] = 0

    for _ in range(n):
        min_distance = sys.maxsize
        min_index = -1


        for i in range(n):
            if not visited[i] and distance[i] < min_distance:
                min_distance = distance[i]
                min_index = i

        visited[min_index] = True

        for j in range(n):
            if graph[min_index][j] != float('inf') and not visited[j]:
                new_dist = distance[min_index] + graph[min_index][j]
                if new_dist < distance[j]:
                    distance[j] = new_dist
                    previous[j] = min_index
    return distance, previous

def construct_path(previous, node, cities):
    path = []
    while node != -1:
        path.insert(0, cities[node])
        node = previous[node]
    return ' -> '.join(path)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/calculate', methods=['POST'])
def calculate():
    cities = ['A', 'B', 'C', 'D', 'E']
    graph = [
        [0, 20, 30, float('inf'), float('inf')],
        [float('inf'), 0, float('inf'), 15, float('inf')],
        [float('inf'), float('inf'), 0, float('inf'), 25],
        [float('inf'), float('inf'), float('inf'), 0, 10],
        [float('inf'), float('inf'), float('inf'), float('inf'), 0]
    ]
    start_city = request.form['start_city']
    start_node = cities.index(start_city)
    distances, previous = dijkstra(graph, start_node)

    result = []
    for i in range(len(cities)):
        if distances[i] == sys.maxsize:
            result.append((cities[i], '∞', 'No path'))
        else:
            path = construct_path(previous, i, cities)
            result.append((cities[i], distances[i], path))

    return render_template('result.html', result=result, start_city=start_city)

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

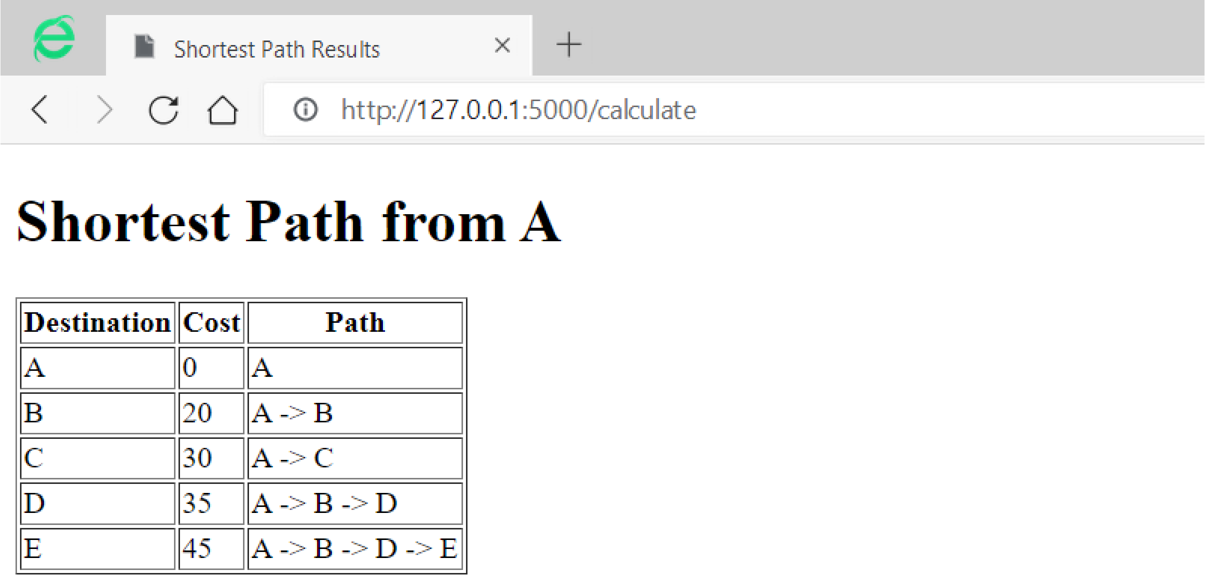
Intex.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shortest Path Finder</title>
</head>
<body>
    <h1>Find Shortest Path Between Cities</h1>
    <form action="/calculate" method="POST">
        <label for="start_city">Enter the Starting City:</label>
        <select name="start_city" id="start_city">
            <option value="A">A</option>
            <option value="B">B</option>
            <option value="C">C</option>
            <option value="D">D</option>
            <option value="E">E</option>
        </select>
        <button type="submit">Calculate</button>
    </form>
</body>
</html>
```

Result.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shortest Path Results</title>
</head>
<body>
    <h1>Shortest Path from {{ start_city }}</h1>
    <table border="1">
        <tr>
            <th>Destination</th>
            <th>Cost</th>
            <th>Path</th>
        </tr>
        {% for city, cost, path in result %}
        <tr>
            <td>{{ city }}</td>
            <td>{{ cost }}</td>
            <td>{{ path }}</td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```

Output :-



Shortest Path from A

Destination	Cost	Path
A	0	A
B	20	A -> B
C	30	A -> C
D	35	A -> B -> D
E	45	A -> B -> D -> E