Name:-Devansh Koyani
Er no:-22162171007
Batch:-54

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

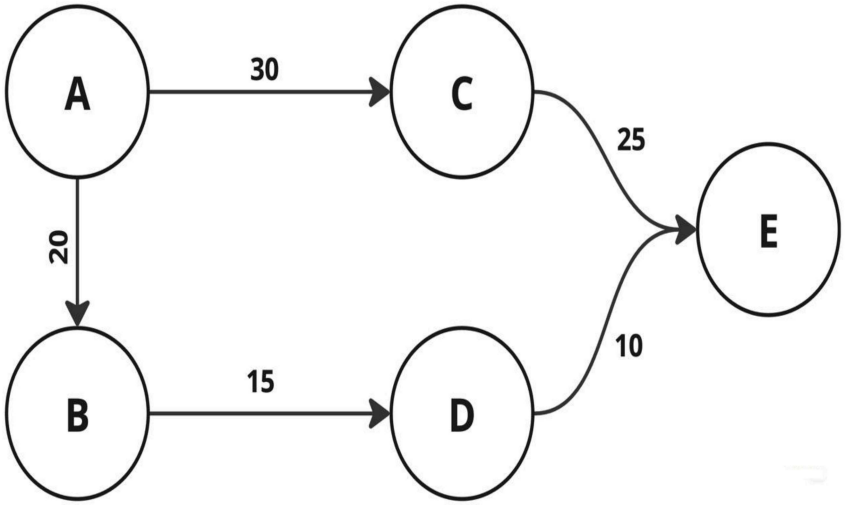
Practical 11

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:-

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


app = Flask(__name__)


def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            if weight == float('inf'):
                continue  # Skip unconnected nodes
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
```

```
                heapq.heappush(priority_queue, (distance, neighbor))


    return distances


@app.route("/", methods=["GET", "POST"])
def index():
    result = None
    nodes = 0
    start_city = None
    graph = {}

    if request.method == "POST":
        nodes = int(request.form.get("nodes"))
        start_city = request.form.get("start_city").upper()
            cities = [request.form.get(f"city_{i}").upper() for i in
range(nodes)]
        graph = {city: {} for city in cities}


        for i in range(nodes):
            for j in range(nodes):
                neighbor = cities[j]
                weight = request.form.get(f"weight_{i}_{j}")
                    weight_value = float('inf') if weight == '∞' else
int(weight)
```

```
                graph[cities[i]][neighbor] = weight_value


        result = dijkstra(graph, start_city)


        for city in result:
            if result[city] == float('inf'):
                result[city] = '∞'
            else:
                result[city] = str(result[city])


        renderable_graph = {
                    city: {neighbor: ('∞' if weight == float('inf') else
str(weight))
                    for neighbor, weight in neighbors.items()}
            for city, neighbors in graph.items()
        }


         return render_template("prac11.html", result=result, nodes=nodes,
start_city=start_city, graph=renderable_graph)


       return render_template("prac11.html", result=result, nodes=nodes,
start_city=start_city, graph=graph)


if __name__ == "__main__":
```

```
    app.run(debug=True)
```

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>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            color: #333;

            margin: 0;

            padding: 20px;

        }



        h2,
```

```
        h3 {
            color: #2c3e50;
        }

        .container {
            display: flex;
            flex-wrap: wrap;
            justify-content: space-between;
            gap: 20px;
        }

        .form-container,
        .result-container {
            background: #ffffff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
            flex: 1;
            min-width: 300px;
            max-width: 48%;
        }

        label {
```

```
        display: block;
        margin-bottom: 8px;
        font-weight: bold;
    }

    input[type="number"],
    input[type="text"] {
        width: 60px;
        padding: 5px;
        margin-bottom: 15px;
        border: 1px solid #ccc;
        border-radius: 4px;
        transition: border-color 0.3s;
    }

    input[type="number"]:focus,
    input[type="text"]:focus {
        border-color: #3498db;
        outline: none;
    }

    button {
        background-color: #3498db;
```

```
        color: white;
        padding: 10px 15px;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        transition: background-color 0.3s;
    }

    button:hover {
        background-color: #2980b9;
    }

    table {
        border-collapse: collapse;
        width: 100%;
        margin: 20px 0;
    }

    th,
    td {
        padding: 12px;
        text-align: center;
        border: 1px solid #ddd;
```

```
}


th {
    background-color: #3498db;
    color: white;
}


tr:nth-child(even) {
    background-color: #f2f2f2;
}


tr:hover {
    background-color: #d1e7fd;
}


.input-row {
    display: flex;
    align-items: center;
    margin-bottom: 15px;
}


.city-names,
.distance-inputs {
```

```
            display: flex;
            flex-direction: column;
            gap: 10px;
            margin-bottom: 15px;
        }

        .distance-input-row {
            display: flex;
            align-items: center;
            gap: 5px;
        }

        .distance-input-row label {
            margin-right: 10px;
        }
    </style>
</head>

<body>
    <h2>Shortest Path Finder</h2>
    <div class="container">
        <div class="form-container">
            <form method="POST">
```

```
                <label for="nodes">Enter Total Number of Cities:</label>
                <input type="number" id="nodes" name="nodes" required><br>

                <label for="start_city">Enter the Starting City:</label>
                    <input type="text" id="start_city" name="start_city"
required><br>

                <h3>Enter City Names</h3>
                <div id="city_names" class="city-names"></div>

                <h3>Enter Distances (Use '∞' for no connection)</h3>
                <div id="graph_inputs" class="distance-inputs"></div>

                <button type="submit">Submit</button>
            </form>
        </div>

        <div class="result-container">
            {% if result %}
                <h3>Input Distance Table</h3>
                <table>
                    <thead>
                        <tr>
                            <th></th>
```

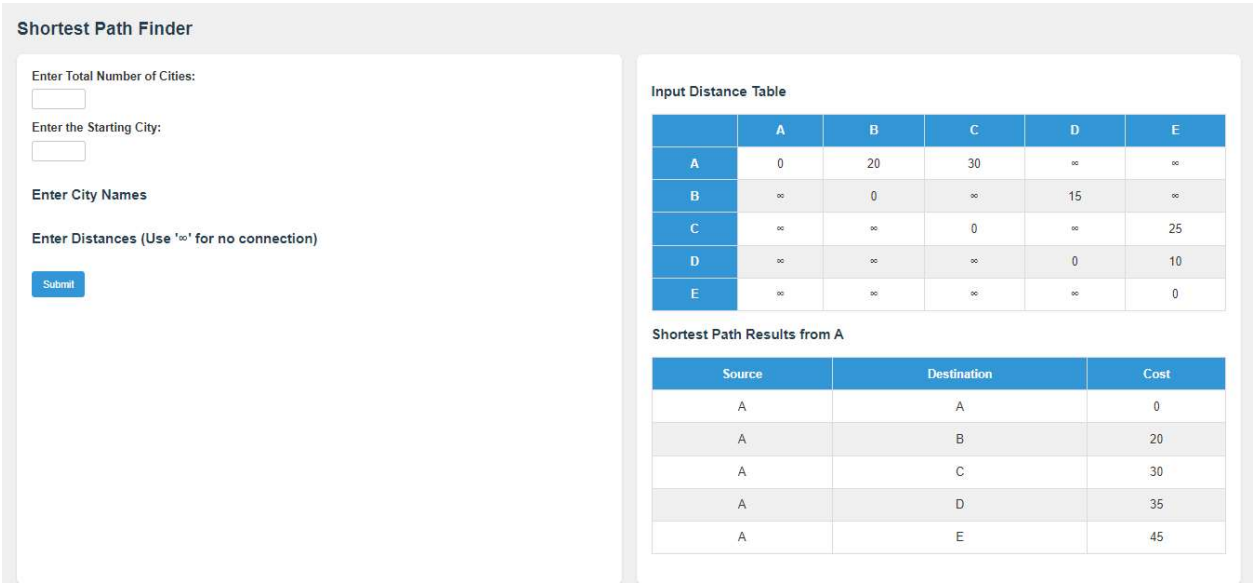
```
                            {% for city in graph.keys() %}
                                <th>{{ city }}</th>
                            {% endfor %}
                        </tr>
                    </thead>
                    <tbody>
                        {% for city in graph.keys() %}
                            <tr>
                                <th>{{ city }}</th>
                                {% for neighbor in graph[city].keys() %}
                                    <td>{{ graph[city][neighbor] }}</td>
                                {% endfor %}
                                {% for missing_neighbor in graph.keys() %}
                                        {% if missing_neighbor not in
graph[city] %}
                                            <td>∞</td>
                                        {% endif %}
                                {% endfor %}
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>

                <h3>Shortest Path Results from {{ start_city }}</h3>
```

```
            <table>
                <thead>
                    <tr>
                        <th>Source</th>
                        <th>Destination</th>
                        <th>Cost</th>
                    </tr>
                </thead>
                <tbody>
                    {% for city in result.keys() %}
                        <tr>
                            <td>{{ start_city }}</td>
                            <td>{{ city }}</td>
                            <td>{{ result[city] }}</td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        {% endif %}
    </div>
</div>

<script>
```

```
            document.getElementById('nodes').addEventListener('change',
function () {
            const nodes = parseInt(this.value);
            const cityContainer = document.getElementById('city_names');
                                                const    graphContainer    =
document.getElementById('graph_inputs');
            cityContainer.innerHTML = '';
            graphContainer.innerHTML = '';

            for (let i = 0; i < nodes; i++) {
                const input = document.createElement('input');
                input.type = 'text';
                input.id = `city_${i}`;
                input.name = `city_${i}`;
                input.placeholder = `City ${i + 1}`;
                cityContainer.appendChild(input);
            }

            for (let i = 0; i < nodes; i++) {
                const rowDiv = document.createElement('div');
                rowDiv.classList.add('distance-input-row');
                       rowDiv.innerHTML = `<h4>Distances from City ${i +
1}</h4>`;
```

```
                for (let j = 0; j < nodes; j++) {
                    const label = document.createElement('label');
                    label.setAttribute('for', `weight_${i}_${j}`);
                    label.textContent = `To City ${j + 1}:`;

                    const input = document.createElement('input');
                    input.type = 'text';
                    input.id = `weight_${i}_${j}`;
                    input.name = `weight_${i}_${j}`;
                    input.value = i === j ? '' : '∞';

                    rowDiv.appendChild(label);
                    rowDiv.appendChild(input);
                }

                graphContainer.appendChild(rowDiv);
            }
        });
    </script>
</body>

</html>
```

Output:-


Shortest Path Finder
Enter Total Number of Cities:
Enter the Starting City:
Enter City Names
Enter Distances (Use '∞' for no connection)
Submit
Input Distance Table
A B C D E
A 0 20 30 ∞ ∞
B ∞ 0 ∞ 15 ∞
C ∞ ∞ 0 ∞ 25
D ∞ ∞ ∞ 0 10
E ∞ ∞ ∞ ∞ 0
Shortest Path Results from A
Source Destination Cost
A A 0
A B 20
A C 30
A D 35
A E 45