

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

Institute of Computer Technology

B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

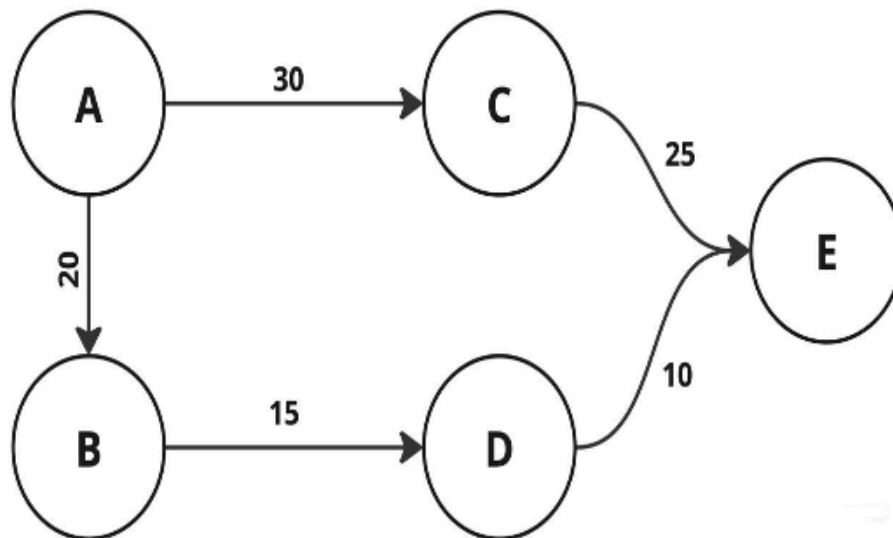
### Practical 11

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):

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

### PRACTICAL-11

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>A</i>	0	20	30	$\infty$	$\infty$
<i>B</i>	$\infty$	0	$\infty$	15	$\infty$
<i>C</i>	$\infty$	$\infty$	0	$\infty$	25
<i>D</i>	$\infty$	$\infty$	$\infty$	0	10
<i>E</i>	$\infty$	$\infty$	$\infty$	$\infty$	0

Output:

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>A</i>	0	20	30	35	45
<i>B</i>	$\infty$	0	$\infty$	15	25
<i>C</i>	$\infty$	$\infty$	0	$\infty$	25
<i>D</i>	$\infty$	$\infty$	$\infty$	0	10
<i>E</i>	$\infty$	$\infty$	$\infty$	$\infty$	0

OR

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

Python Code:-

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

app = Flask(__name__)

# Dijkstra's algorithm to find shortest paths
def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

#### PRACTICAL-11

```
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()
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

#### PRACTICAL-11

```
# Gather city names
cities = [request.form.get(f"city_{i}").upper() for i in
range(nodes)]

graph = {city: {} for city in cities}

# Build graph with weights
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

# Run Dijkstra's algorithm
result = dijkstra(graph, start_city)

# Replace 'inf' with '∞' in results for clarity
for city in result:
    result[city] = '∞' if result[city] == float('inf')
else str(result[city])

# Renderable graph for HTML visualization
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("index.html", result=result,
nodes=nodes, start_city=start_city, graph=renderable_graph)

if __name__ == "__main__":
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

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

Index.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>
    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
</head>

<body>
    <div class="main-container">
        <h1 class="title">Shortest Path Finder</h1>

        <div class="container">
            <!-- Input Form -->
            <div class="form-container">
                <form method="POST">
                    <label for="nodes">Enter Total Number of
Cities:</label>
                    <input type="number" id="nodes" name="nodes"
required>

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

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

### PRACTICAL-11

```
<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"
class="btn">Submit</button>
</form>
</div>

<!-- Results -->
<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() %}
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

#### PRACTICAL-11

```
<td>{{ graph[city][neighbor] }}</td>
    {% 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, cost in result.items() %}
        <tr>
            <td>{{ start_city }}</td>
            <td>{{ city }}</td>
            <td>{{ cost }}</td>
        </tr>
        {% endfor %}
    </tbody>
</table>
{% endif %}
</div>
</div>
</div>

<script>
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

### PRACTICAL-11

```
// Dynamically generate input fields for city names and
distances

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++) {
        cityContainer.innerHTML += `<input type="text"
name="city_${i}" placeholder="City ${i + 1}" required>`;
    }

    for (let i = 0; i < nodes; i++) {
        graphContainer.innerHTML += `<h4>Distances from
City ${i + 1}</h4>`;
        for (let j = 0; j < nodes; j++) {
            graphContainer.innerHTML += `<label>To City
${j + 1}</label>
            <input type="text"
name="weight_${i}_${j}" value="${i === j ? ' ' : '∞'}"
required>`;
        }
    }
});
</script>
</body>
```



NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

```
</html>
```

Style.css:-

```
body {
    font-family: 'Arial', sans-serif;
    background-color: #f0f8ff;
    margin: 0;
    padding: 20px;
}

h1 {
    text-align: center;
    color: #2c3e50;
    animation: fadeInDown 1s;
}

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

.form-container,
.result-container {
    background: #ffffff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s, box-shadow 0.3s;
}

.form-container:hover,
.result-container:hover {
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

#### PRACTICAL-11

```
transform: translateY(-5px);
box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
}

input[type="text"],
input[type="number"] {
  width: calc(100% - 20px);
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
  transition: border-color 0.3s;
}

input:focus {
  border-color: #3498db;
  outline: none;
}

button {
  width: 100%;
  padding: 10px;
  background: linear-gradient(90deg, #3498db, #2ecc71);
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 16px;
  transition: background 0.3s, transform 0.2s;
}

button:hover {
  background: linear-gradient(90deg, #2ecc71, #3498db);
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

```
        transform: scale(1.05);
    }

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

    th, td {
        border: 1px solid #ddd;
        padding: 10px;
        text-align: center;
    }

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

    tr:nth-child(even) {
        background-color: #f9f9f9;
    }

    tr:hover {
        background-color: #f1c40f;
        color: white;
    }

    @keyframes fadeInDown {
        from {
            opacity: 0;
            transform: translateY(-20px);
        }
    }
```

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

```
}  
to {  
  opacity: 1;  
  transform: translateY(0);  
}  
}
```

Output:-

Shortest Path Finder

Enter Total Number of Cities:  
5

Enter Starting City:  
A

Enter City Names

a  
v  
b  
s  
e

Enter Distances (Use '∞' for no connection)

Distances from City 1

To City 1:  
0

To City 2:

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

The screenshot shows a web browser window titled "Shortest Path Finder" at the URL "127.0.0.1:5000". The page has a teal sidebar with a home icon and a plus sign. The main content area contains the following elements:

- Input fields for "To City 2:", "To City 3:", "To City 4:", and "To City 5:", each with a value of "∞".
- A section titled "Distances from City 2" with input fields for "To City 1:" (∞), "To City 2:" (0), "To City 3:" (∞), "To City 4:" (∞), and "To City 5:" (∞).

The Windows taskbar at the bottom shows the system clock as 15:13 on 16-11-2024.

The screenshot shows the same web browser window after further calculations. The page content is as follows:

- The "Distances from City 2" section remains unchanged with all values at "∞".
- A new section titled "Distances from City 3" has been added, with input fields for "To City 1:" (∞), "To City 2:" (∞), "To City 3:" (25), "To City 4:" (∞), and "To City 5:" (∞).
- A section titled "Distances from City 4" has been added, with input fields for "To City 1:" (∞), "To City 2:" (∞), and "To City 3:" (∞).

The Windows taskbar at the bottom shows the system clock as 15:14 on 16-11-2024.

NAME:- MARVIN PATEL

ER NO :- 22162101013

SUB :- AAD

## PRACTICAL-11

The screenshot shows a web browser window titled "Shortest Path Finder" with the address bar displaying "127.0.0.1:5000". The page contains several input fields for city names and distances. The "To City 3:" field contains "∞", "To City 4:" contains "22", and "To City 5:" contains "∞". Below these, under the heading "Distances from City 5", there are five more input fields: "To City 1:" (∞), "To City 2:" (∞), "To City 3:" (∞), "To City 4:" (∞), and "To City 5:" (10). A green "Submit" button is at the bottom.

The screenshot shows the "Shortest Path Finder" web application with the title "Shortest Path Finder" centered at the top. On the left, there is an input form with fields for "Enter Total Number of Cities:", "Enter Starting City:", "Enter City Names", and "Enter Distances (Use '∞' for no connection)". A green "Submit" button is at the bottom of this form. On the right, there is a section titled "Input Distance Table" containing a 5x5 table with cities A, V, B, S, and E as both sources and destinations. The values in the table are: A to A (0), A to V (∞), A to B (∞), A to S (∞), A to E (∞); V to A (∞), V to V (0), V to B (∞), V to S (∞), V to E (∞); B to A (∞), B to V (∞), B to B (25), B to S (∞), B to E (∞); S to A (∞), S to V (∞), S to B (∞), S to S (22), S to E (∞); E to A (∞), E to V (∞), E to B (∞), E to S (∞), E to E (10). Below this table is a section titled "Shortest Path Results from A" containing a table with 5 rows showing the shortest path from source A to destinations A, V, B, S, and E, with costs 0, ∞, ∞, ∞, and ∞ respectively.

Source	Destination	Cost
A	A	0
A	V	∞
A	B	∞
A	S	∞
A	E	∞