Itisha Desai
Sem – 5
Branch – Cyber Security
Batch – CSE54
Enrollment No. – 22162171006

Algorithm Analysis and Design
Practical-6

Question :-

Given a sequence of matrices, we want to find the most efficient way to multiply these matrices together to obtain the minimum number of multiplications. The problem is not actually to perform the multiplication of the matrices but to obtain the minimum number of multiplications. We have many options because matrix multiplication is an associative operation, meaning that the order in which we multiply does not matter. The optimal order depends only on the dimensions of the matrices. The brute-force algorithm is to consider all possible orders and take the minimum. This is a very inefficient method. Implement the minimum multiplication algorithm using dynamic programming and determine where to place parentheses to minimize the number of multiplications. Find an optimal parenthesization of a matrix chain product whose sequence of dimensions are (5, 10, 3, 12, 5, 50, 6).

Code :-

app.py :-

```
from flask import Flask, render_template, request

app = Flask(__name__)

def matrix_chain_order(p):
    n = len(p) - 1
    m = [[0 for _ in range(n)] for _ in range(n)]
    s = [[0 for _ in range(n)] for _ in range(n)]

    for length in range(2, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            m[i][j] = float('inf')
            for k in range(i, j):
                q = m[i][k] + m[k + 1][j] + p[i] * p[k + 1] * p[j + 1]
                if q < m[i][j]:
                    m[i][j] = q
                    s[i][j] = k

    return m, s

def construct_optimal_solution(s, i, j):
    if i == j:
        return f"A{i + 1}"
    else:
        k = s[i][j]
        left = construct_optimal_solution(s, i, k)
        right = construct_optimal_solution(s, k + 1, j)
        return f"({left} x {right})"

@app.route('/', methods=['GET', 'POST'])
def index():
    num_matrices = None
    min_operations = None
    parenthesization = None
    m = None

    if request.method == 'POST':
        dimensions = list(map(int, request.form['dimensions'].split(',')))
        num_matrices = len(dimensions) - 1
        m, s = matrix_chain_order(dimensions)
        parenthesization = construct_optimal_solution(s, 0, len(dimensions) - 2)
        min_operations = m[0][-1]
```

```
    m = [['X' if cell == 0 else cell for cell in row] for row in m]

  return render_template('index.html',
                         min_operations=min_operations,
                         parenthesization=parenthesization,
                         m=m,
                         num_matrices=num_matrices)

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

index.html :-

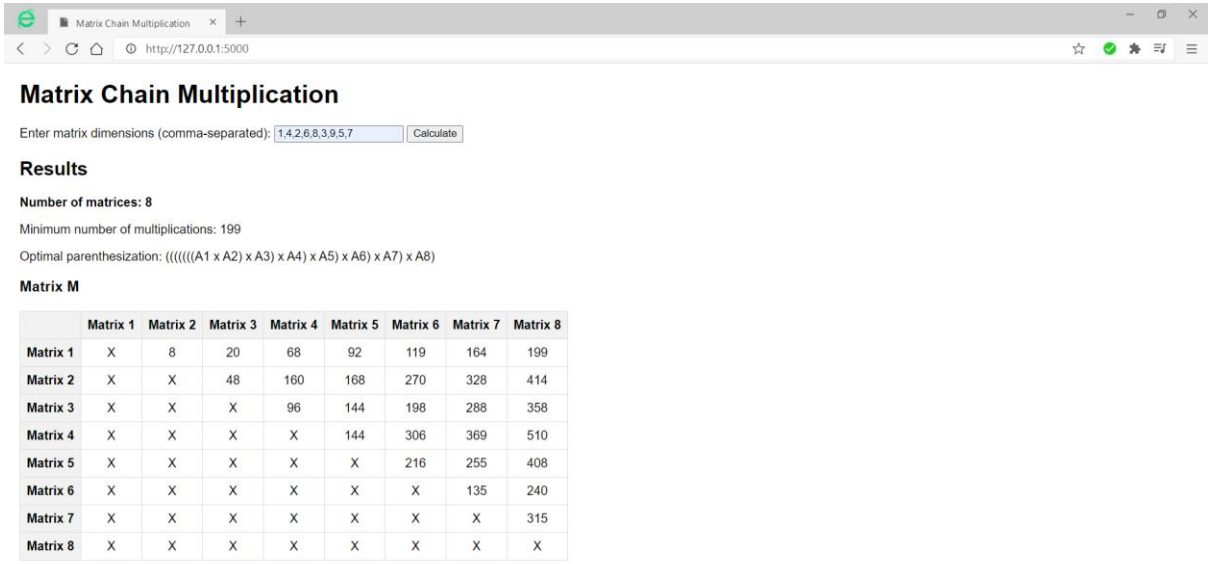
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Matrix Chain Multiplication</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    table {
      border-collapse: collapse;
      margin: 20px 0;
    }
    table, th, td {
      border: 1px solid #ddd;
    }
    th, td {
      padding: 8px;
      text-align: center;
    }
    th {
      background-color: #f2f2f2;
    }
    .matrix-label {
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h1>Matrix Chain Multiplication</h1>
  <form method="POST">
    <label for="dimensions">Enter matrix dimensions (comma-separated):</label>
```

```
    <input type="text" id="dimensions" name="dimensions" required>
    <button type="submit">Calculate</button>
  </form>

  {% if min_operations is not none %}
    <h2>Results</h2>
    {% if num_matrices is not none %}
      <p class="matrix-label">Number of matrices: {{ num_matrices }}</p>
    {% endif %}
    <p>Minimum number of multiplications: {{ min_operations }}</p>
    <p>Optimal parenthesization: {{ parenthesization }}</p>

    {% if m is not none %}
      <h3>Matrix M</h3>
      <table>
        <tr>
          <th></th>
          {% for i in range(num_matrices) %}
            <th>Matrix {{ i + 1 }}</th>
          {% endfor %}
        </tr>
        {% for i in range(num_matrices) %}
          <tr>
            <th>Matrix {{ i + 1 }}</th>
            {% for j in range(num_matrices) %}
              <td>{{ m[i][j] }}</td>
            {% endfor %}
          </tr>
        {% endfor %}
      </table>
    {% endif %}
  {% endif %}
</body>
</html>
```

Output :-



	Matrix 1	Matrix 2	Matrix 3	Matrix 4	Matrix 5	Matrix 6	Matrix 7	Matrix 8
Matrix 1	X	8	20	68	92	119	164	199
Matrix 2	X	X	48	160	168	270	328	414
Matrix 3	X	X	X	96	144	198	288	358
Matrix 4	X	X	X	X	144	306	369	510
Matrix 5	X	X	X	X	X	216	255	408
Matrix 6	X	X	X	X	X	X	135	240
Matrix 7	X	X	X	X	X	X	X	315
Matrix 8	X	X	X	X	X	X	X	X