Name: Maurya Patel R
Enrollment-No:22162101014
Batch:51
Branch:CBA

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 6

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:

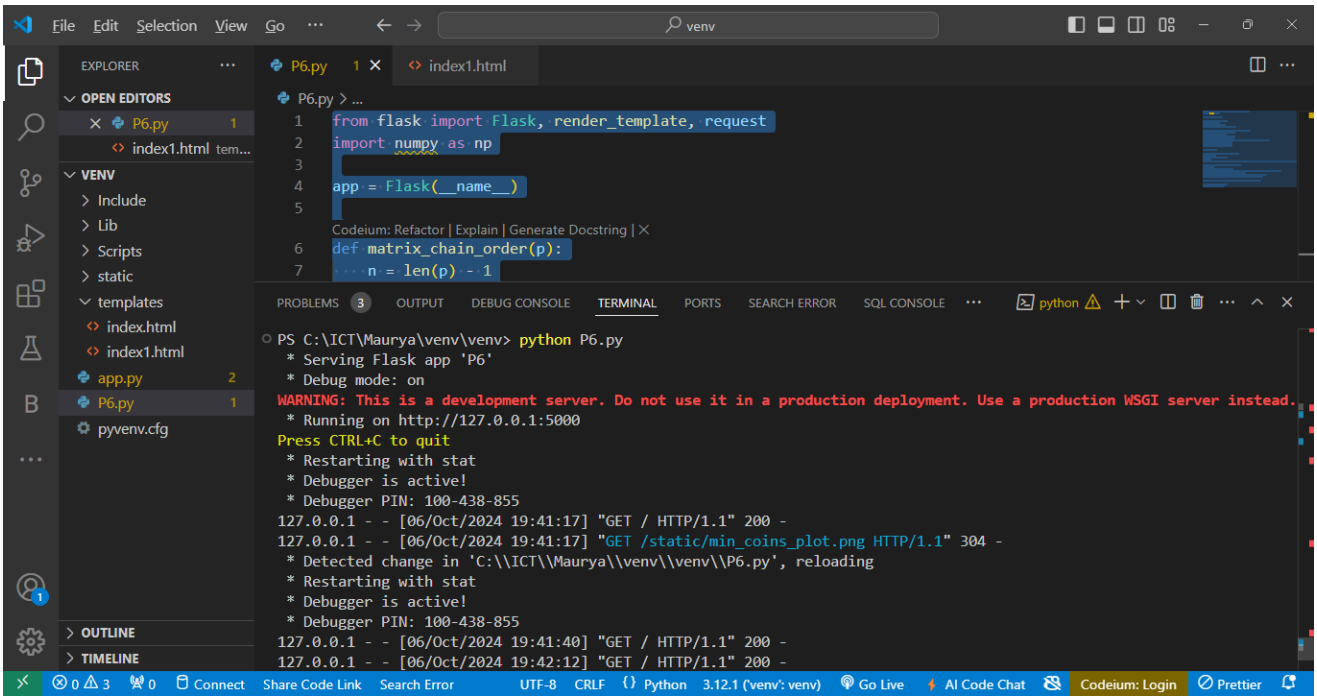
Python code:

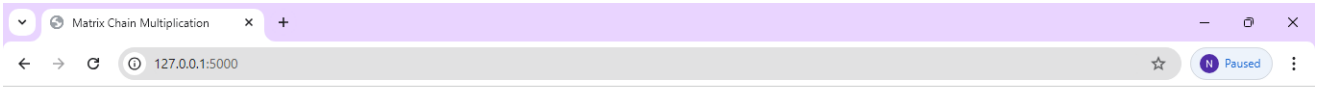
```
from flask import Flask, render_template, request
import numpy as np

app = Flask(__name__)

def matrix_chain_order(p):
    n = len(p) - 1
    m = np.zeros((n, n))
    s = np.zeros((n, n), dtype=int)

    for l in range(2, n + 1):  # l is chain length
        for i in range(n - l + 1):
            j = i + l - 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 optimal_parenthesization(s, i, j):
    if i == j:
        return f"A{i + 1}"
    else:
        return f"({optimal_parenthesization(s, i, s[i][j])} x
{optimal_parenthesization(s, s[i][j] + 1, j)})"

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        dimensions = list(map(int,
request.form['dimensions'].strip().split(',')))
        m, s = matrix_chain_order(dimensions)
        order = optimal_parenthesization(s, 0, len(dimensions) - 2)
        return render_template('index1.html', order=order, m=m)

    return render_template('index1.html', order=None, m=None)

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

Html code:

```
<!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>
</head>
<body>
    <h1>Matrix Chain Multiplication</h1>
    <form method="post">
        <label for="dimensions">Enter matrix dimensions:</label>
        <input type="text" id="dimensions" name="dimensions" required>
        <button type="submit">Calculate</button>
    </form>

    {% if order %}
        <h2>Optimal Parenthesization:</h2>
        <p>{{ order }}</p>
        <h2>Dynamic Programming Table:</h2>
        <table border="1">
            <tr>
                <th></th>
                {% for j in range(m.shape[0]) %}
                    <th>{{ j }}</th>
                {% endfor %}
            </tr>
            {% for i in range(m.shape[0]) %}
                <tr>
                    <th>{{ i }}</th>
                    {% for j in range(m.shape[1]) %}
                        <td>{{ m[i][j] }}</td>
                    {% endfor %}
                </tr>
            {% endfor %}
        </table>
    {% endif %}
</body>
</html>
```

OUTPUT:





Matrix Chain Multiplication

Enter matrix dimensions: 5,10,3,12,5,50,6 Calculate

Optimal Parenthesization:

((A1 x A2) x ((A3 x A4) x (A5 x A6)))

Dynamic Programming Table:

	0	1	2	3	4	5
0	0.0	150.0	330.0	405.0	1655.0	2010.0
1	0.0	0.0	360.0	330.0	2430.0	1950.0
2	0.0	0.0	0.0	180.0	930.0	1770.0
3	0.0	0.0	0.0	0.0	3000.0	1860.0
4	0.0	0.0	0.0	0.0	0.0	1500.0
5	0.0	0.0	0.0	0.0	0.0	0.0

Dynamic Programming Table:

	0	1	2	3	4	5
0	0.0	150.0	330.0	405.0	1655.0	2010.0
1	0.0	0.0	360.0	330.0	2430.0	1950.0
2	0.0	0.0	0.0	180.0	930.0	1770.0
3	0.0	0.0	0.0	0.0	3000.0	1860.0
4	0.0	0.0	0.0	0.0	0.0	1500.0
5	0.0	0.0	0.0	0.0	0.0	0.0

Optimal Parenthesization:

((A1 x A2) x ((A3 x A4) x (A5 x A6)))