Name : Harpalsinh Bhati

Enr. No.: 22162171010

Sub: Algorithm Analysis & Design

Branch: CS

Batch: 54

Sem: 5-B

Practical-6

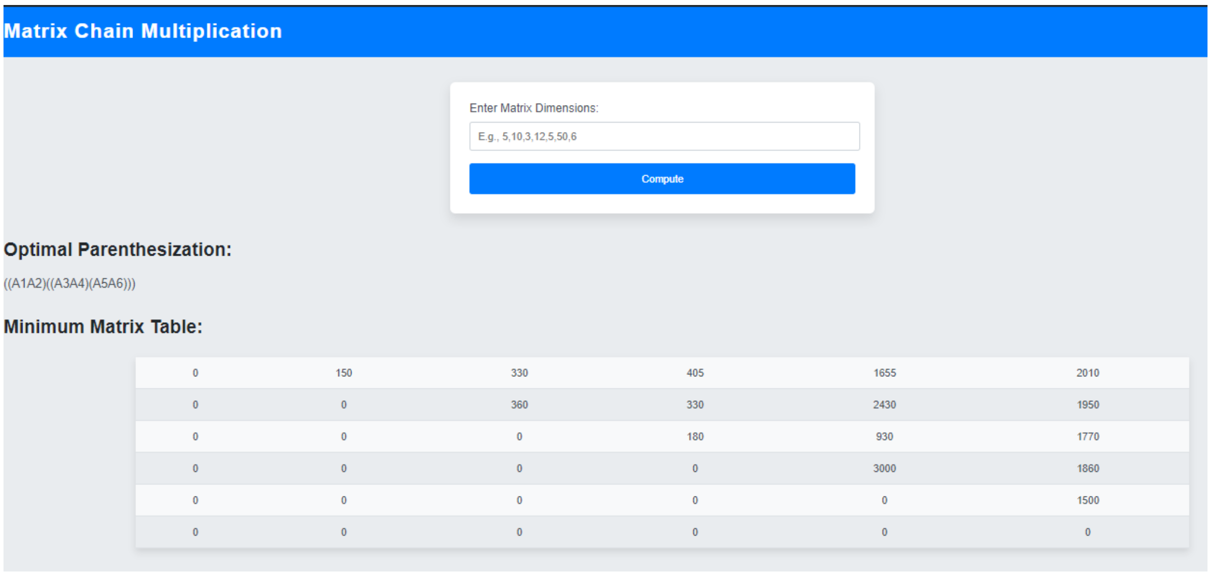
Given a sequence of matrices, we want to find the most efficient way to multiply these matrices together to obtain the minimum number of multiplications. The problem is not actually to perform the multiplication of the matrices but to obtain the minimum number of multiplications. We have many options 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
from flask import Flask, request, render_template
import numpy as np

app = Flask(__name__)

# Matrix chain multiplication algorithm to calculate minimum multiplications
def practical_6(p):
    n = len(p) - 1
    m = [[0] * n for _ in range(n)]
    s = [[0] * n for _ in range(n)]

    # L is the chain length
    for L in range(2, n + 1):
        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

# Function to generate the optimal parenthesization
def optimal(s, i, j):
    if i == j:
        return f"A{i + 1}"
    return f"({optimal(s, i, s[i][j])}{optimal(s, s[i][j] + 1, j)})"

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        dimensions = list(map(int, request.form['dimensions'].split(',')))
        m, s = practical_6(dimensions)
        op = optimal(s, 0, len(dimensions) - 2)

        # Prepare the matrix for display
        matrix = []
        n = len(m)
        for i in range(n):
            row = ["0" if j < i else str(m[i][j]) if m[i][j] != float('inf')
else '∞' for j in range(n)]
            matrix.append(row)

        return render_template('Prac_6.html', op=op, matrix=matrix)

    return render_template('Prac_6.html')

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

Output:



Matrix Chain Multiplication

Optimal Parenthesization:

((A1A2)((A3A4)(A5A6)))

Minimum Matrix Table:

0	150	330	405	1655	2010
0	0	360	330	2430	1950
0	0	0	180	930	1770
0	0	0	0	3000	1860
0	0	0	0	0	1500
0	0	0	0	0	0