Marwadi University	**Marwari University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing Matrix Chain Multiplication using Dynamic Programming Approach	
Experiment No: 10	**Date:**	**Enrollment No: 92200133030**

Aim: Implementing Matrix Chain Multiplication using a Dynamic Programming Approach

IDE: Visual Studio Code

Implementing Matrix Chain Multiplication using a Dynamic Programming Approach

Theory: -

- Matrix Chain Multiplication is a classic optimization problem that seeks to find the most efficient way to multiply a sequence of matrices. The goal is to minimize the number of scalar multiplications required. Since matrix multiplication is associative, the order in which the matrices are multiplied can significantly affect the total computational cost.

1. Problem Definition

- Given nnn matrices $A1, A2, \ldots, An$ with dimensions $p0 \times p1, p1 \times p2, \ldots, pn-1 \times pn$, determine the optimal parenthesization to minimize the total number of scalar multiplications.

2. Key Concepts:-

a. Associativity of Matrix Multiplication:

- The multiplication (A1A2)A3 is equivalent to A1(A2A3) but can have different computational costs depending on the dimensions.

b. Cost of Multiplication:

- Multiplying two matrices A of dimensions $p \times q$ and B of dimensions $q \times r$ requires $p \times q \times r$ scalar multiplications.

c. Dynamic Programming Approach:

- The problem is broken into subproblems where the solution to smaller chains is reused to solve larger chains.
- We use a table to store the minimum number of scalar multiplications for each subproblem.

3. Dynamic Programming Approach :-

1. **Define the Problem:**
   - Let m[i][j] represent the minimum number of scalar multiplications required to multiply the subchain $Ai, Ai+1, \ldots, Aj$.
2. **Base Case:**
   - If i=ji = ji=j, a single matrix does not require any multiplication: m[i][i]=0m[i][i] = 0m[i][i]=0
3. **Recursive Relation:**
   - For a subchain Ai to Aj, split it into two parts Ai to Ak and Ak+1 to Aj for $i \leq k < j$: $m[i][j] = \min(i \leq k < j)\{m[i][k] + m[k+1][j] + pi-1 \times pk \times pj\}$

Marwadi University	**Marwari University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing Matrix Chain Multiplication using Dynamic Programming Approach	
Experiment No: 10	**Date:**	**Enrollment No: 92200133030**

- Here, p_{i-1}, p_k, p_j are the dimensions involved in multiplying the two resulting matrices.

4. **Iterative Computation:**
   - The chain length L is varied from 2 to n (length of subchains).
   - For each subchain, compute m[i][j] for all valid i and j.
5. **Result:**
   - The optimal cost for multiplying the entire chain is stored in m[1][n].

Algorithm: -

Programming Language: - C++

Code :-

```
#include <bits/stdc++.h>
using namespace std;

int matrixMultiplication(vector<int>& arr) {
    int n = arr.size();
```

Marwadi University	**Marwari University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing Matrix Chain Multiplication using Dynamic Programming Approach	
Experiment No: 10	**Date:**	**Enrollment No: 92200133030**

```
    vector<vector<int>> dp(n, vector<int>(n, 0));

    for (int len = 2; len < n; len++) {
        for (int i = 0; i < n - len; i++) {
            int j = i + len;
            dp[i][j] = INT_MAX;
            for (int k = i + 1; k < j; k++) {
                int cost = dp[i][k] + dp[k][j] + arr[i] * arr[k] * arr[j];
                dp[i][j] = min(dp[i][j], cost);
            }
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << setw(4) << dp[i][j] << " ";
        }
        cout << endl;
    }
    return dp[0][n - 1];
}

int main() {
    vector<int> arr;
    int n;
    cout << "Enter the Number of Matrix :- ";
    cin >> n;

    for( int i = 0; i < n; i++) {
        int row, column;
        cout << "Enter the Number of Rows and Columns for Matrix " << i + 1 << " :- ";
        cin >> row >> column;
        if (i == 0) {
            arr.push_back(row);
            arr.push_back(column);
        }
        else {
            arr.push_back(column);
        }
    }
    cout << matrixMultiplication(arr);
```

```
    return 0;
}
```

Output :-

```
PS C:\Users\Aaryan> cd "d:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 10\" ; if ($?) { g++ Matrix_Chain_Multi
plication.cpp -o Matrix_Chain_Multiplication } ; if ($?) { .\Matrix_Chain_Multiplication }
Enter the Number of Matrix :- 4
Enter the Number of Rows and Columns for Matrix 1 :- 5 4
Enter the Number of Rows and Columns for Matrix 2 :- 4 6
Enter the Number of Rows and Columns for Matrix 3 :- 6 2
Enter the Number of Rows and Columns for Matrix 4 :- 2 7
   0    0  120   88  158
   0    0    0   48  104
   0    0    0    0   84
   0    0    0    0    0
   0    0    0    0    0
158
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 10>
```

Space Complexity:- _____
Justification: -

Time Complexity:

Best Case Time Complexity: _____
Justification: -

Worst Case Time Complexity:- _____
Justification: -

	**Marwari University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing Matrix Chain Multiplication using Dynamic Programming Approach	
Experiment No: 10	**Date:**	**Enrollment No: 92200133030**

Conclusion:-