Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: DAA (01CT0512)	**AIM: Implementing Matrix Chain Multiplication using Dynamic Programming Approach**	
Experiment No: 10	**Date:**	**Enrolment No: 92301733046**

Theory:

This method solves the Matrix Chain Multiplication (MCM) problem using Dynamic Programming. Given a sequence of matrices, the goal is to find the optimal order of multiplication that minimizes the total number of scalar multiplications. A 2D DP table m[i][j] stores the minimum cost of multiplying matrices from index i to j. For each possible split k, the cost is computed as the sum of costs of the two subproblems plus the cost of multiplying the resulting matrices.

- Subproblem Overlapping & Optimal Substructure

Code:

```
#find minimum number of multiplication

def matrix_chain_multiply(n):
    m = [[0 for _ in range(n)] for _ in range(n)]
    for d in range(1, n):
        for i in range(1, n - d):
            j = i + d
            m[i][j] = float('inf')
            for k in range(i, j):
                q = m[i][k] + m[k + 1][j] + dn[i - 1] * dn[k] * dn[j]


                if q < m[i][j]:
                    m[i][j] = q
    return m[1][n - 1]
```

```
dn = [3, 5, 2, 3, 4]
n = len(dn)
print("Minimum number of multiplications is:",matrix_chain_multiply(n))
```

output:

```
PS D:\DAA\Lab and Lecture Codes> python -u "d:\DAA\Lab and Lecture Codes\matrix_chain_multiplication
.py"
Minimum number of multiplications is: 78
PS D:\DAA\Lab and Lecture Codes>
```

Space Complexity:

- **$O(n^2)$**
- **Justification:** A DP table m of size n × n is created to store the minimum multiplication costs for different subproblems of the matrix chain.

Best Case Time Complexity:

- **$O(n^3)$**
- **Justification:** Even if the optimal split is straightforward, the algorithm must still evaluate all possible splits k for every subproblem.

Worst Case Time Complexity:

- **$O(n^3)$**
- **Justification:** Regardless of the matrix dimensions, the DP table must be filled for all (i, j) pairs, and each requires checking all possible k.