K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch:A3	**Roll No.:16010122083**

Experiment No.______

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation Matrix Chain Multiplication of Dynamic Programming

---

Objective: To learn Matrix chain multiplication using Dynamic Programming Approach

---

CO to be achieved:

CO 2 Describe various algorithm design strategies to solve different problems and analyse Complexity.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://www.lsi.upc.edu/~mjserna/docencia/algofib/P07/dynprog.pdf**
4. **http://www.geeksforgeeks.org/travelling-salesman-problem-set-1/**
5. **http://www.mafy.lut.fi/study/DiscreteOpt/tspdp.pdf**
6. **https://class.coursera.org/algo2-2012-001/lecture/181**
7. **http://www.quora.com/Algorithms/How-do-I-solve-the-travelling-salesman-problem-using-Dynamic-programming**
8. **www.cse.hcmut.edu.vn/~dtanh/download/Appendix_B_2.ppt**
9. **www.ms.unimelb.edu.au/~s620261/powerpoint/chapter9_4.ppt**

Pre Lab/ Prior Concepts:

Data structures, Concepts of algorithm analysis

Historical Profile:

Dynamic Programming (DP) is used heavily in optimization problems (finding the maximum and the minimum of something). Applications range from financial models and operation research to biology and basic algorithm research. So the good news is that understanding DP is profitable. However, the bad news is that DP is not an algorithm or a data structure that you can memorize. It is a powerful algorithmic design technique.

New Concepts to be learned:

Application of algorithmic design strategy to any problem, dynamic Programming method of problem solving Vs other methods of problem solving, optimality of the solution, Optimal Binary Search Tree Problems and their applications

Theory:

Problem definition:

Given a sequence of N matrices, the matrix chain multiplication problem is to find the most efficient way to multiply these matrices by minimizing the number of computations involved during multiplications.

Optimal Substructure: parameterization/ select the subgroup of matrices that will result in least number of computations.

For multiplication of matrix series Ai to Aj, choose Ak such that multiplication of matrices through Ai..k and Ak+1…j will incur least number of computations for any k such that i<=k<j.

Recursive Formula:

$$n[i,j] = \begin{cases} 0 & i = j, \\ \min_{i \le k < j} \left(m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \right) & i < j \end{cases}$$

Algorithm:

Algo: Matrix Chain Multiplication

1) Iterate from $l=2$ to $N-1$ which denotes the length of the range
→ Iterate from $i=0$ to $N-1$
→ Find the right end of the range j having l matrix
→ Iterate from $k=i+1$ to j which denotes the point of partition
→ Multiply the matrices in range (i,k), and (k,j)
→ This will create two matrices with dimensions arr[i-1]*arr[k] and arr[k]*arr[j]
→ The no. of multi to be perform are arr[i-1]*arr[k]*arr[j]
→ The total no of multi is dp[i][k] + dp[k+1][j] + X
→ The value stored at dp[1][N-1] is the answer

Example:

A	B	C	D
2x1	1X3	3x4	4x5

	1	2	3	4
1		1	1	1
2			2	3
3				3
4				

$$c[i,j] = \min_{i \le k < j} \{ c[i,k] + c[k+1,j] + d_{i-1} d_k d_j \}$$

Solution for the example:

Cost matrix

	1	2	3	4
1	0	6	20	42
2		0	12	32
3			0	60
4				0

$d_0 = 2 \quad d_1 = 1 \quad d_2 = 3 \quad d_3 = 4 \quad d_4 = 5$

$c[2,3] = k = 2 \{ c[2,2] + c[3,3] + d_1 d_2 d_3 \}$

$0 + 0 + 12$

$c[3,4] = k = 3 \{ c[3,3] + c[4,4] + d_2 d_3 d_4 \}$

$c[1,3] = \min \begin{cases} k=1 \quad c[1,1] + c[2,3] + d_0 d_1 d_3 \\ = 20 \\ k=2 \quad c[1,2] + c[3,3] + d_0 d_2 d_3 \\ = 30 \end{cases}$

$\therefore$ k=1 is min i.e 20

$c[2,4] \ \min \begin{cases} k=2 \quad c[2,2] + c[3,4] + d_1 d_2 d_4 \\ 6 + 0 + 24 = 30 \\ k=3 \quad c[2,3] + c[4,4] + d_1 d_3 d_4 \\ 12 + 0 + 20 \\ = 32 \end{cases}$

$c[1,4] = \min \begin{cases} k=1 \rightarrow 42 \\ k=2 \\ k=3 \end{cases}$ min value

$\therefore$ The final answer is 42

Analysis of algorithm:

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

int matrixChainMulti(vector<int> p)
{
    int n = p.size();
    vector<vector<int>> dp(n, vector<int>(n, 0));
    for (int i = 1; i < n - 1; i++)
    {
        dp[i][i + 1] = p[i - 1] * p[i] * p[i + 1];
    }
    for (int l = 2; l < n - 1; l++)
    {
        for (int i = 1; i < n - l; i++)
        {
            int j = i + l;
            dp[i][j] = INT_MAX;
            for (int k = i; k < j; k++)
            {
                dp[i][j] = min(dp[i][j], dp[i][k] + dp[k + 1][j] + p[i - 1] * p[k] * p[j]);
            }
        }
    }
    return dp[1][n - 1];
}
int main()
{
    vector<int> p = {2,1,3,4,5};

    cout << "Matrix dimensions: ";
    for (int i = 0; i < p.size() - 1; i++)
    {
        cout << p[i] << "x" << p[i + 1] << " ";
    }
    cout<<endl;
    cout << "Minimum number of multiplications: " << matrixChainMulti(p) << endl;
    return 0;
}
```

```
mkkar@Minav MINGW64 /c/Desktop/DSA/AOA
$ ./a
Matrix dimensions: 2x1 1x3 3x4 4x5
Minimum number of multiplications: 42
```

CONCLUSION:

We learned the Matrix Chain Multiplication of Dynamic Programming and its implementation