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

Batch: B2	**Roll No.:** 16010121110
Experiment No.___7___	
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.Rajasekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algorithms",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:

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

Algorithm:

```
/****************************************************************************
****

Matrix Multiplication

****************************************************************************
****/

public class Main

{

        public static void main(String[] args) {

            int[][] matrixData = {{4,10},{10,3},{3,12},{12,20},{20,7}};

            int [][] jaggedArray = new int [5][5];
```

```
for (int diff=1;diff<=5-1;diff++){
for(int j=0;j<=5-1;j++){
for (int i=0;i<=5-1;i++){
    if((j-i)!=diff){
        continue; //for diagonal elements order of execution
    }
    if(i>j){
        continue;
    }
    if(i==j){
        jaggedArray[i][j]=0;
        continue;
    }
    int min=10000;
    for(int k=i;k<j;k++){
      // System.out.print(jaggedArray[k+1][j]+",");
    int
temp=jaggedArray[i][k]+jaggedArray[k+1][j]+matrixData[i][0]*matrixData[k][1]*matrixDat
a[j][1];
    if(min>temp){
        min=temp;
    }
    }
     jaggedArray[i][j]=min;
}
}
}
```

```
 for (int i=0;i<=5-1;i++){
    for (int j=0;j<=5-1;j++){
     System.out.print(jaggedArray[i][j]+",");
   }
   System.out.println();
   }
        System.out.println("Hello World");
  }
}
```

Example and Solution for the example:

Matrix chain Multiplication

$M(i,j) = \min\{M(i,k) + M(k+1,j) + P_0P_2P_3\}$ $\forall i,j$ $\forall k>(i,j)$ → column × column × row

Example → Matrix have size 4×10, 10×3, 3×12, 12×20, 20×7

1	2	3	4	5	
0	120	264	1080	1344	1
	0	360	1320	1350	2
		0	720	1140	3
			0	4080	4
				0	5

Time Complexity

$O(n^3)$

Space Complexity $O(n^2)$

$M(1,2) = 4\times10\times3 = 120$
$M(2,3) = 10\times3\times12 = 360$
$M(3,4) = 3\times12\times20 = 720$
$M(4,5) = 12\times20\times7 = 4080$

$M(1,3) = \begin{cases} M(1,2)+M(3,3)+P_0P_1P_3 = 120+0+4\times3\times12 = \underline{264} \\ M(1,1)+M(2,3)+P_0P_1P_3 = 0+360+4\times10\times12 = 840 \end{cases}$

$M(2,4) = \begin{cases} M(2,3)+M(4,4)+C_0C_1C_2 = 360+0+10\times12\times20 = 2760 \\ M(2,2)+M(3,4)+C_0C_1C_2 = 0+720+10\times3\times20 = \underline{1320} \end{cases}$

$M(3,5) = \begin{cases} M(3,3)+M(4,5)+C_0C_1C_2 = 0+4080+3\times12\times7 = 1930 \\ M(3,4)+M(5,5)+C_0C_1C_2 = 720+3\times20\times7 = \underline{1140} \end{cases}$

$M(1,4) = \begin{cases} M(1,2)+M(3,4)+C_0C_1C_2 = 264+0+4\cdot12\cdot20 = 1224 \\ M(1,2)+M(3,4)+C_0C_1C_2 = 120+720+4\times3\times20 = \underline{1080} \\ M(1,3)+M(3,4)+C_0C_1C_2 = 0+1320+4\cdot10\cdot20 = 2120 \end{cases}$

$M(2,5) = \begin{cases} M(2,4)+M(5,5)+C_0C_1C_2 = 1320+0+10\times20\times7 = 2720 \\ M(2,3)+M(4,5)+C_0C_1C_2 = 360+1680+10\times12\times7 = 2880 \\ M(2,2)+M(3,5)+C_0C_1C_2 = 0+1140+10\times3\times7 = \underline{1350} \end{cases}$

$M(1,5) = \text{Min}\begin{cases} M(1,4)+M(5,5)+C_0C_1C_2 = 1554 \\ M(1,3)+M(4,5)+C_0C_1C_2 = 2016 \\ M(1,2)+M(3,5)+C_0C_1C_2 = \underline{1344} \leftarrow \\ M(1,1)+M(2,5)+C_0C_1C_2 = 1670 \end{cases}$

(1 2) (3,5)
(12) ((3,4)(4,5))

Analysis of algorithm:

Matrix Chain → Time Complexity

loop 1 → $(n-1) \times 1$ $= O(n) \times O(n)$

[illegible]

loop 2 → $n-2 \times 2$ $O(n) \times O(n)$

loop 3 → $n-3 \times 3$ $O(n) \times O(n)$

⋮

loop n $n-(n+1) \times n-1$ $O(n) \times O(n)$

n times

$= O(n^3)$

Space Complexity

$\frac{n}{2} \times \frac{n}{2} = O(n^2)$ Matrix is Maintained

Matrix Chain Theory

Matrix Chain Multiplication is a dynamic programing algorithm which is used to find the Most efficiant method for multiplication of Matrix

Cost of Multiplication

$n\,[\quad]_{m} * m\,[\quad]_{p} = n\,[\quad]_{p}$

Cost $= n \times m \times p$

CONCLUSION: Thus we have performed the matrix chain multiplication algorithm using dynamic programming. This algorithm has time complexity O(n^3) and space complexity O(n^2). This algorithm is used for finding the most efficient manner in which two matrices can be multiplied.