K. J. Somaiya College of Engineering
(A Constituent College of Somaiya Vidyavihar University)

Batch:A1 **Roll No.:16010120015**

Experiment No.___8___

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.

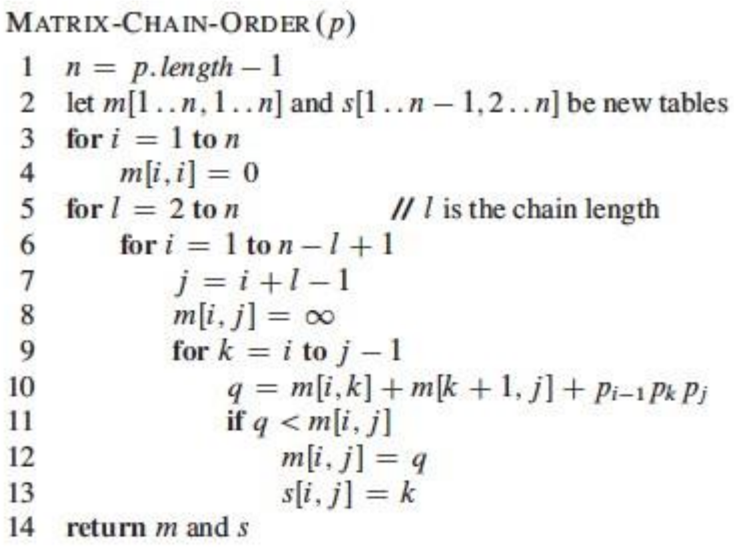
Optimal Substructure: parenthesization/ select the subgroup of matrices that will result in least number of computations.

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:

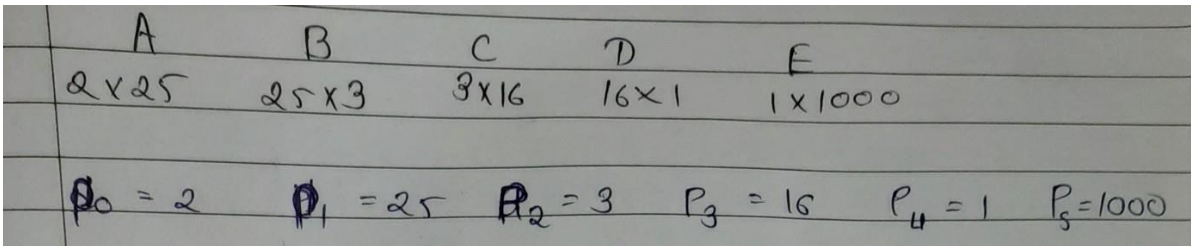
$$m[i,j] = \begin{cases} 0 & i = j, \\ \min_{i \le k < j} (m[i,k] + m[k+1,j] + p_{i-1}p_k p_j) & i < j \end{cases}$$

Algorithm:



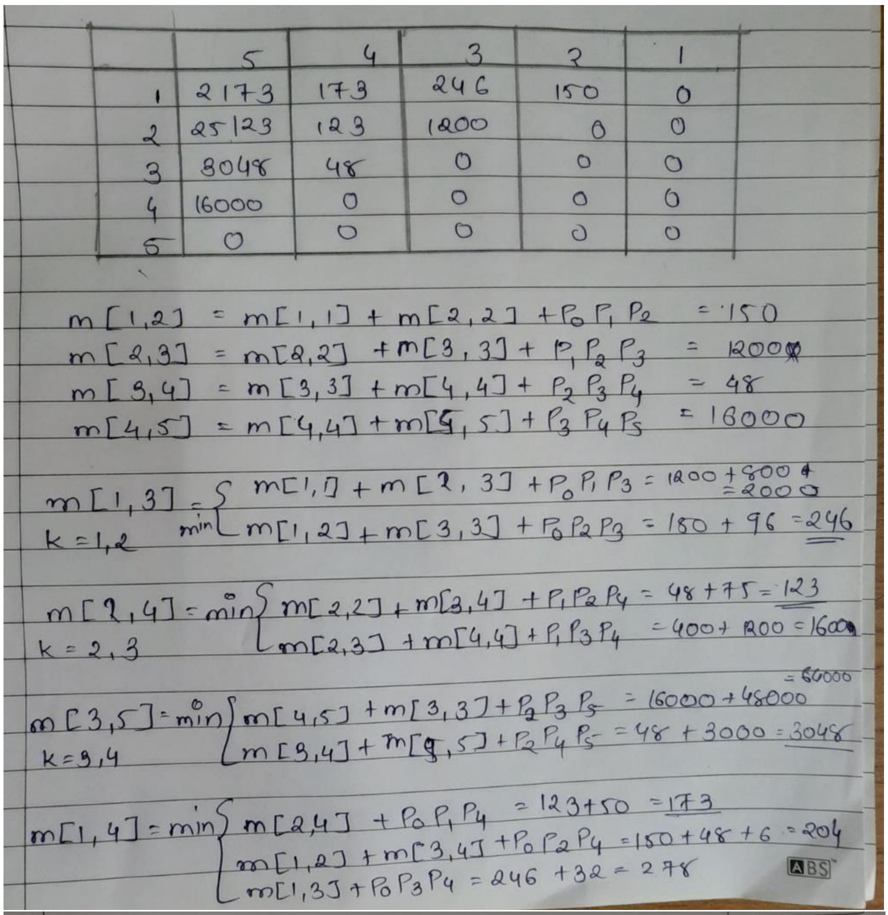
```
MATRIX-CHAIN-ORDER(p)
n = p.length − 1
let m[1..n, 1..n] and s[1..n − 1, 2..n] be new tables
for i = 1 to n
    m[i, i] = 0
for l = 2 to n                // l is the chain length
    for i = 1 to n − l + 1
        j = i + l − 1
        m[i, j] = ∞
        for k = i to j − 1
            q = m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
            if q < m[i, j]
                m[i, j] = q
                s[i, j] = k
return m and s
```

Example:



A	B	C	D	E	
2×25	25×3	3×16	16×1	1×1000	
$P_0 = 2$	$P_1 = 25$	$P_2 = 3$	$P_3 = 16$	$P_4 = 1$	$P_5 = 1000$

Solution for the example:



	5	4	3	2	1
1	2173	173	246	150	0
2	25123	123	1200	0	0
3	3048	48	0	0	0
4	16000	0	0	0	0
5	0	0	0	0	0

$$m[1,2] = m[1,1] + m[2,2] + P_0 P_1 P_2 = 150$$
$$m[2,3] = m[2,2] + m[3,3] + P_1 P_2 P_3 = 1200$$
$$m[3,4] = m[3,3] + m[4,4] + P_2 P_3 P_4 = 48$$
$$m[4,5] = m[4,4] + m[5,5] + P_3 P_4 P_5 = 16000$$

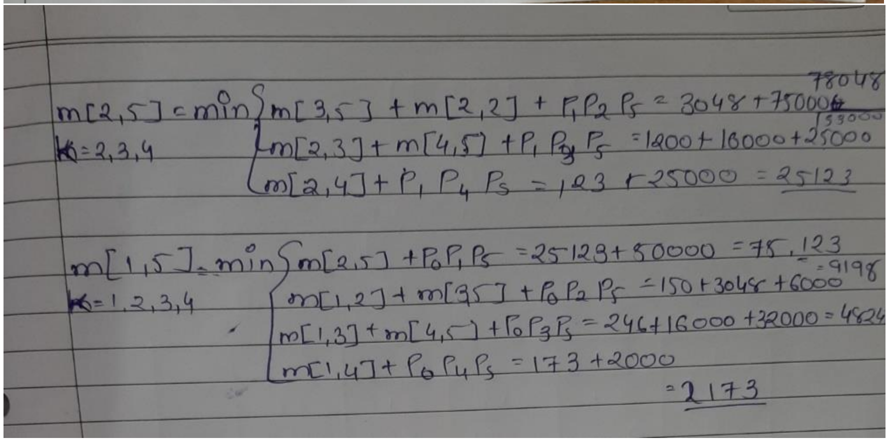
$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + P_0 P_1 P_3 = 1200 + 800 = 2000 \\ m[1,2] + m[3,3] + P_0 P_2 P_3 = 180 + 96 = \underline{\underline{246}} \end{cases}$$
$k = 1, 2$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + P_1 P_2 P_4 = 48 + 75 = \underline{\underline{123}} \\ m[2,3] + m[4,4] + P_1 P_3 P_4 = 400 + 1200 = 1600 \end{cases}$$
$k = 2, 3$

$$m[3,5] = \min \begin{cases} m[4,5] + m[3,3] + P_2 P_3 P_5 = 16000 + 48000 = 64000 \\ m[3,4] + m[5,5] + P_2 P_4 P_5 = 48 + 3000 = \underline{3048} \end{cases}$$
$k = 3, 4$

$$m[1,4] = \min \begin{cases} m[2,4] + P_0 P_1 P_4 = 123 + 50 = \underline{173} \\ m[1,2] + m[3,4] + P_0 P_2 P_4 = 150 + 48 + 6 = 204 \\ m[1,3] + P_0 P_3 P_4 = 246 + 32 = 278 \end{cases}$$

ABS



$$m[2,5] = \min \begin{cases} m[3,5] + m[2,2] + P_1 P_2 P_5 = 3048 + 75000 = 78048 \\ m[2,3] + m[4,5] + P_1 P_3 P_5 = 1200 + 16000 + 25000 = 53000 \\ m[2,4] + P_1 P_4 P_5 = 123 + 25000 = \underline{25123} \end{cases}$$
$k = 2, 3, 4$

$$m[1,5] = \min \begin{cases} m[2,5] + P_0 P_1 P_5 = 25123 + 50000 = 75,123 \\ m[1,2] + m[3,5] + P_0 P_2 P_5 = 150 + 3048 + 6000 = 9198 \\ m[1,3] + m[4,5] + P_0 P_3 P_5 = 246 + 16000 + 32000 = 4824 \\ m[1,4] + P_0 P_4 P_5 = 173 + 2000 = \underline{2173} \end{cases}$$
$k = 1, 2, 3, 4$

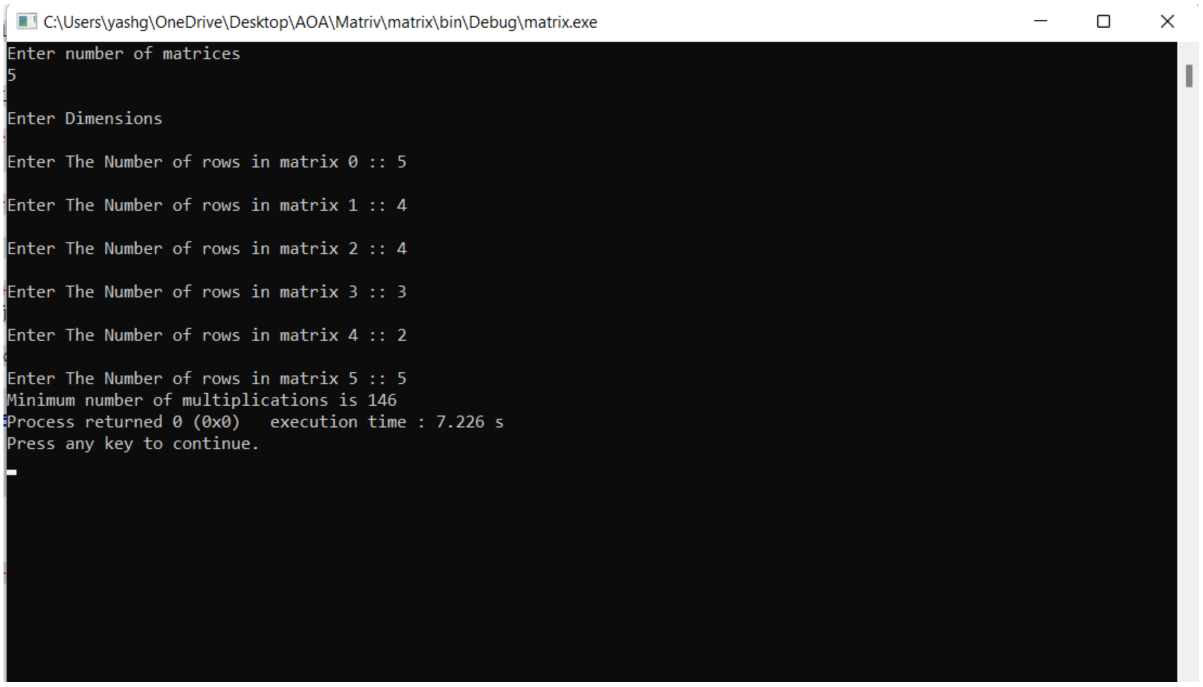
Code:

```
#include<stdio.h>

#include<limits.h>


int MatrixChainMultiplication(int p[], int n)

{

    int m[n][n];

    int i, j, k, L, q;


    for (i=1; i<n; i++)

        m[i][i] = 0;


    for (L=2; L<n; L++)

    {

        for (i=1; i<n-L+1; i++)

        {

            j = i+L-1;

            m[i][j] = INT_MAX;


            for (k=i; k<=j-1; k++)

            {

                q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];

                if (q < m[i][j])

                {
```

```
                m[i][j] = q;
            }
        }
    }
  }
  return m[1][n-1];
}

int main()
{
  int n,i;
  printf("Enter number of matrices\n");
  scanf("%d",&n);
  n++;
  int arr[n];
  printf("\nEnter Dimensions \n");

  for(i=0;i<n;i++)
  {
    printf("\nEnter The Number of rows in matrix %d :: ",i);
    scanf("%d",&arr[i]);
  }

  int size = sizeof(arr)/sizeof(arr[0]);
  printf("Minimum number of multiplications is %d ", MatrixChainMultiplication(arr, size));
```

```
    return 0;

}
```

OUTPUT:


C:\Users\yashg\OneDrive\Desktop\AOA\Matriv\matrix\bin\Debug\matrix.exe

```
Enter number of matrices
5

Enter Dimensions

Enter The Number of rows in matrix 0 :: 5

Enter The Number of rows in matrix 1 :: 4

Enter The Number of rows in matrix 2 :: 4

Enter The Number of rows in matrix 3 :: 3

Enter The Number of rows in matrix 4 :: 2

Enter The Number of rows in matrix 5 :: 5
Minimum number of multiplications is 146
Process returned 0 (0x0)   execution time : 7.226 s
Press any key to continue.
```

Analysis of algorithm:

There are three nested loops. Each loop executes a maximum n times.

1. l, length, O (n) iterations.
2. i, start, O (n) iterations.
3. k, split point, O (n) iterations

Total Complexity is: O (n^3)

CONCLUSION:

Matrix chain multiplication is an optimization problem concerning the most efficient way to multiply a given sequence of matrices.

It compute the minimum cost way to group the objects to apply the operation over the sequence In dynamic programming algorithm It is the fastest way to concatenate a sequence of strings.

we can say that the matrix chain multiplication algorithm using Dynamic Programming in the best case and average case takes O(n^3).

Therefore , we have understood and successfully implemented Matrix Chain Multiplication of Dynamic Programming in the expirement.