Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of Master of Computer Application

Experiment	3
Aim	To understand and implement Dynamic Programming Approach
Objective	1) Write Pseudocode for given problems and understanding the implementation of Dynamic Programming 2) Solve Matrix Multiplication Problem using Dynamic Programming 3) Calculating time complexity of the given problems
Name	Vaibhav Agarwal
UCID	2023510001
Class	FYMCA
Batch	D

Algorithm and Explanation of the technique used	**<u>Pseudocode for Matrix Chaining Multiplication</u>**
	MatrixChainOrder(p, i, j)
	    if i == j return 0
	
	    min = INFINITY
	    for k from i to j-1:
	        count = MatrixChainOrder(p, i, k) + MatrixChainOrder(p, k + 1, j) + p[i - 1] * p[k] * p[j]
	        if count < min:
	            min = count

return min

3, 4, 5, 6

P_0	P_1	P_2	P_3
3	4	5	6

The Matrix Size = 3x4, 4x5, 5x6.

	1	2	3
1	0	60	150
2		0	120
3			0

$3 \times 4 \times 5 = 60$
$4 \times 5 \times 6 = 120$

$M[1,3] = M[1,2] + M[3,3] + P_0 . P_2 . P_3 = 60 + 0 + 90 = 150$
$= M[1,1] + M[2,3] + P_0 . P_1 . P_3 = 0 + 120 + 72 = 192$

$\therefore M[1,3] = 150$

$\therefore (M_1 \times M_2) M_3$

Time Complexity
The time complexity for Matrix Chaining Multiplication is O(n^3).

Program(Code)

```
import java.util.*;
class MatrixChainMultiplication {
  static int MatrixChainOrder(int p[], int i, int j)
  {
    if (i == j)
      return 0;

    int min = Integer.MAX_VALUE;

    for (int k = i; k < j; k++) {
      int count = MatrixChainOrder(p, i, k)
            + MatrixChainOrder(p, k + 1, j)
            + p[i - 1] * p[k] * p[j];
      if (count < min)min = count;
    }

    return min;
  }
```

	<pre> public static void main(String args[])
	  {
	    int arr[] = new int[] {3, 4, 5,6};
	    int N = arr.length;
	
	    System.out.println("Minimum number of multiplications is "+ MatrixChainOrder(arr, 1, N - 1));
	  }
	}</pre>

Output	`Minimum number of multiplications is 150`
Justification of the complexity calculated	Subproblem Time complexity - O(n^2) Within each subproblem, the algorithm involves a loop that iterates over the possible positions to split the subchain. This loop runs in O(n) time. For each subproblem, the algorithm calculates the cost of multiplying matrices at each possible split point. Multiplying these factors together, we get the overall time complexity: O(n^2) * O(n) * O(1) = O(n^3) Therefore, the time complexity for Matrix Chaining Multiplication is O(n^3)
Conclusion	Advantages: Using dynamic programming we can store the solutions to these sub problems so that it does not use extra space for the same sub problem. Applications: Optimization - Utilized in compilers to optimize code generation and execution, particularly for mathematical operations, resulting in faster and more efficient programs. Data Compression - Used in various compression algorithms to efficiently encode and decode data, reducing storage requirements and transmission times. Computer Graphics - Used for transformations like scaling, rotation, and translation of images and objects in computer graphics rendering engines.