**Batch: B-1              Roll No.: 16010122104**

**Experiment No. 6**

**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**](http://www.lsi.upc.edu/~mjserna/docencia/algofib/P07/dynprog.pdf)
4. [**http://www.geeksforgeeks.org/travelling-salesman-problem-set-1/**](http://www.geeksforgeeks.org/travelling-salesman-problem-set-1/)
5. [**http://www.mafy.lut.fi/study/DiscreteOpt/tspdp.pdf**](http://www.mafy.lut.fi/study/DiscreteOpt/tspdp.pdf)
6. [**https://class.coursera.org/algo2-2012-001/lecture/181**](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**](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**](http://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](https://en.wikipedia.org/wiki/Matrix_multiplication) by minimizing the number of computations involved during multiplications.

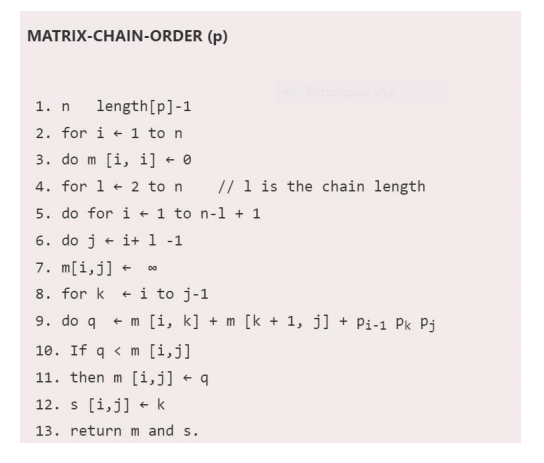
**Optimal Substructure:**  parameterization/ select the subgroup of matrices that will result in least number of computations.

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.

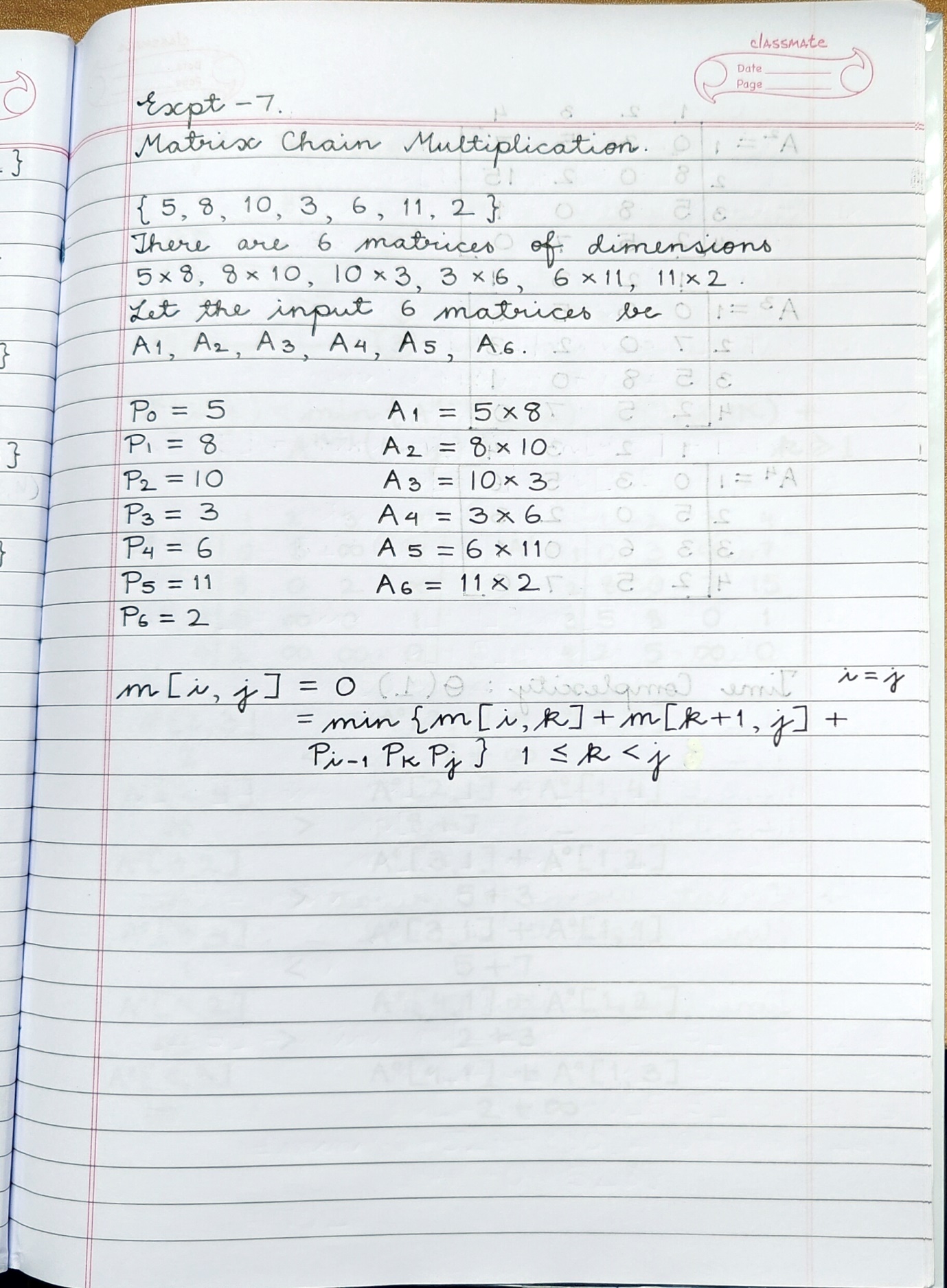
**Recursive Formula:**

https://lh7-us.googleusercontent.com/ugbHVSLxwx6ml2fCpuOweRxggEEdlMFEsZMAKV18SRAepBxlIwqtW7vkjgSjtNrsLtO3Gj3Mji08hAH9tt6wmh00rjajUY7aH-0dxDWolGUXuRf_oruMhF0H6v9GCzu9TsXb3gEJ_qDMhvzAM1jZ1g

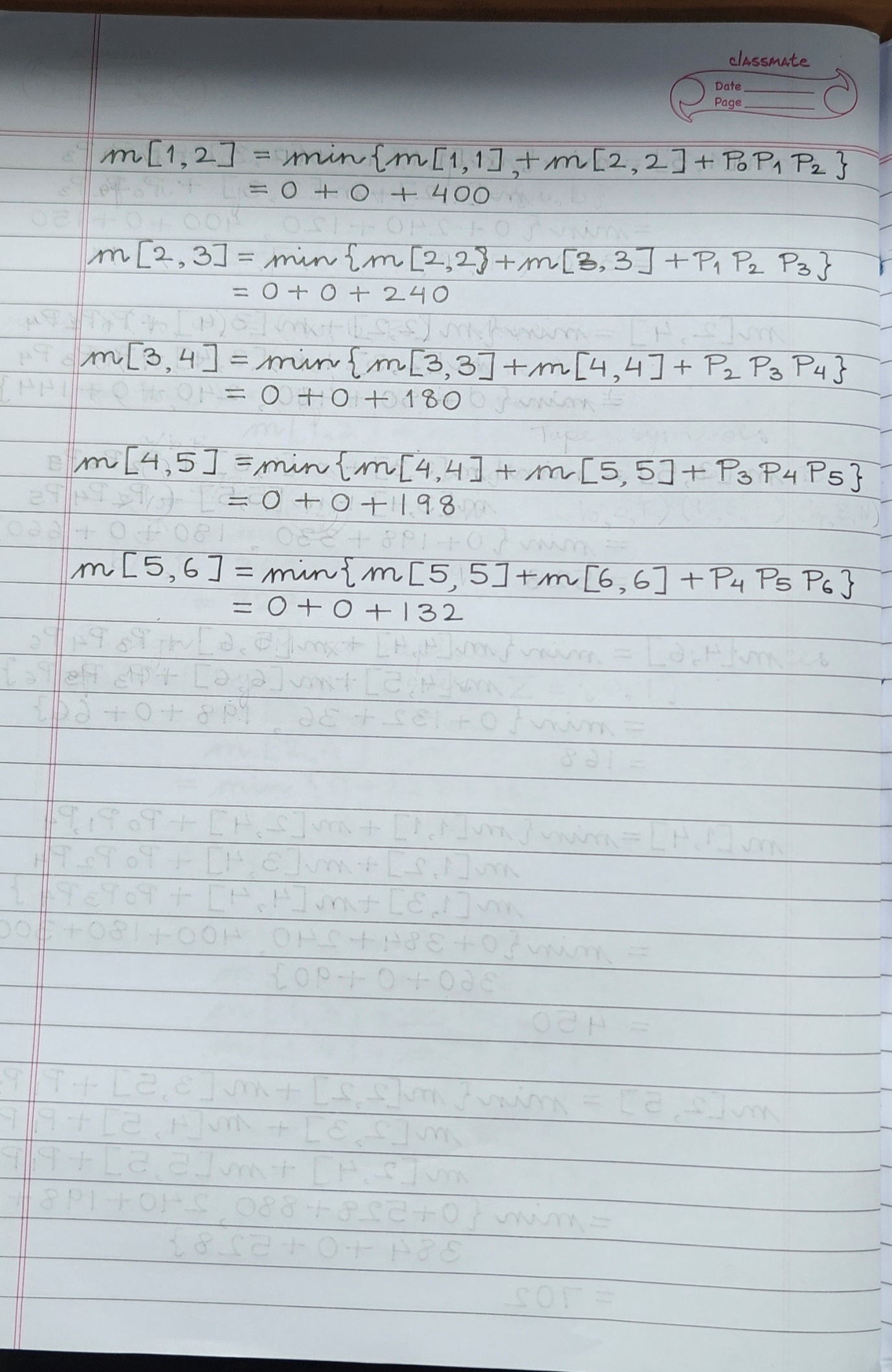
**Algorithm:**

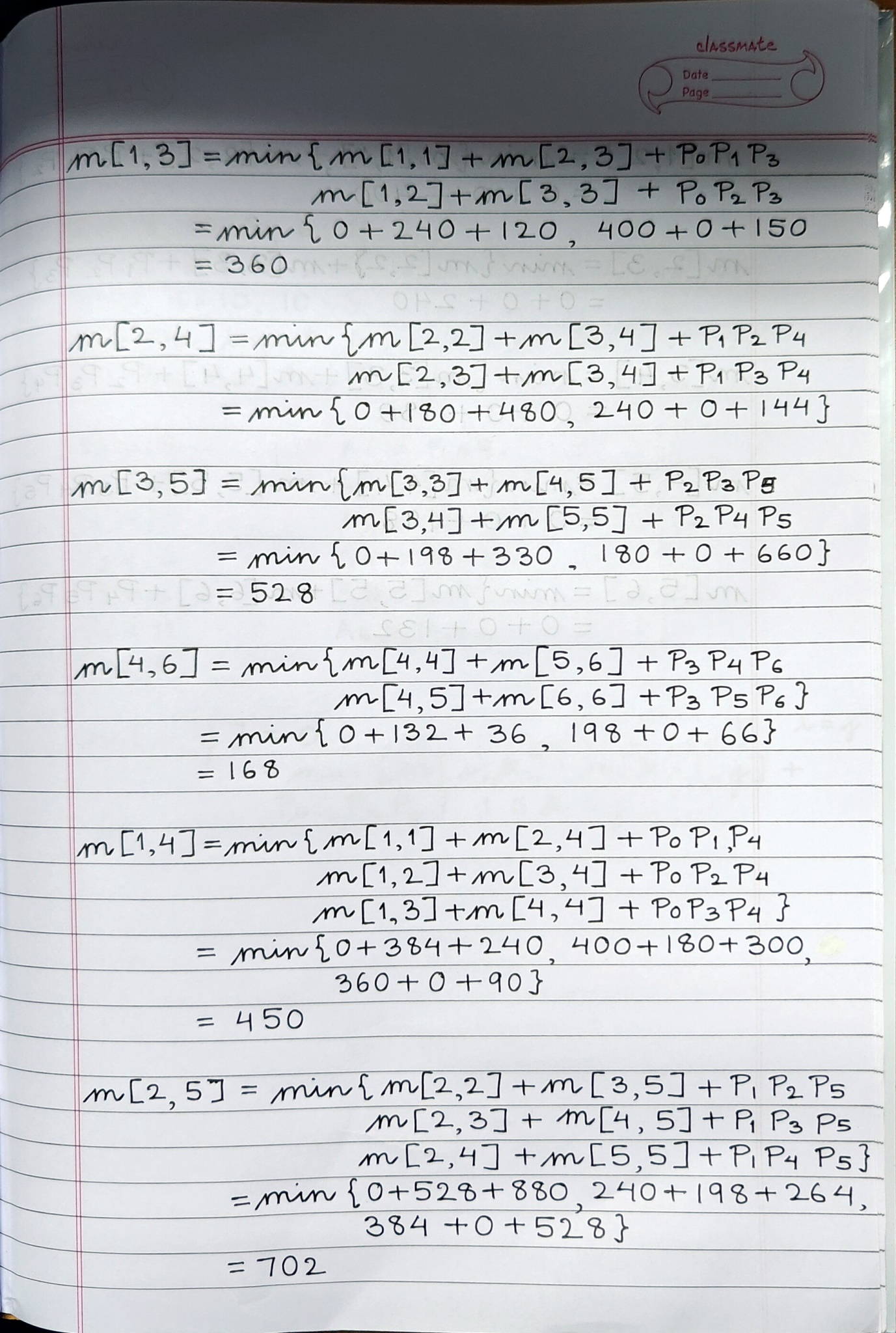


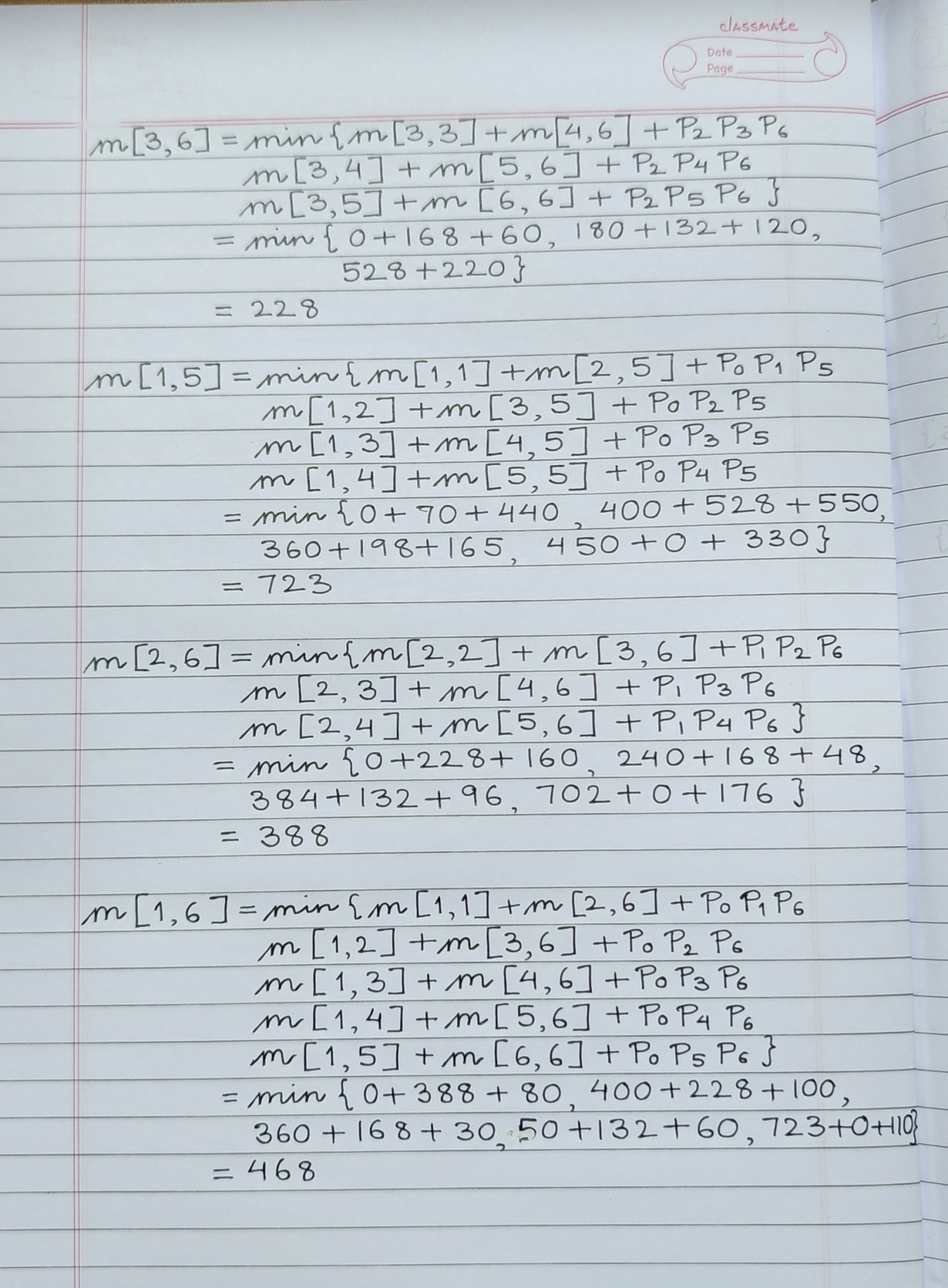
**Example:**



**Solution for the example:**



****

****

**Analysis of algorithm:**

**Total Complexity is: O (n3)**

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

Body of loop constant complexity

**Space Complexity:**

O(n\*n) where n is the number present in the chain of the matrices. We create a DP matrix that

stores the results after each operation.

**Code:**

import java.io.\*;

import java.util.\*;

class MatrixChainMultiplication {

// Matrix Ai has dimension p[i-1] x p[i]

// for i = 1 . . . n

static int MatrixChainOrder(int p[], int i, int j)

{

if (i == j)

return 0;

int min = Integer.MAX\_VALUE;

// Place parenthesis at different places

// between first and last matrix,

// recursively calculate count of multiplications

// for each parenthesis placement

// and return the minimum count

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 minimum count

return min;

}

// Driver code

public static void main(String args[])

{

int arr[] = new int[] { 1, 2, 3, 4, 3 };

int N = arr.length;

// Function call

System.out.println(

"Minimum number of multiplications is "

+ MatrixChainOrder(arr, 1, N - 1));

}

}

**CONCLUSION:**

In this experiment, we have learnt Implementation of Matrix Chain Multiplication by dynamic programming

approach. We have also understood the algorithm and implemented the same on java. Additionally, we have

compared the time complexity of the program.