

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
AAD Practical 10

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 10

Problem : Given a sequence of matrices, we want to find the most efficient way to multiply these matrices together to obtain the minimum number of multiplications. The problem is not actually to perform the multiplication of the matrices but to obtain the minimum number of multiplications. We have many options because matrix multiplication is an associative operation, meaning that the order in which we multiply do not matter. The optimal order depends only on the dimensions of the matrices. The brute-force algorithm is to consider all possible orders and take the minimum. This is a very inefficient method.

Implement the minimum multiplication algorithm using dynamic programming and determine where to place parentheses to minimize the number of multiplications.

Find an optimal parenthesization of a matrix chain product whose sequence of dimensions are (5, 10, 3, 12, 5, 50, 6).

Input:

Enter total matrices: 4
Enter no. of rows in matrix 1: 5
Enter no. of rows in matrix 2: 4
Enter no. of rows in matrix 3: 6
Enter no. of rows in matrix 4: 2
Enter no. of columns in matrix 4: 7

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
AAD Practical 10

Output:

The number of scalar multiplications needed: 158
Optimal parenthesization: ((A[1](A[2]A[3]))A[4])

Code:

```
import YSL_io

def mcm(dmsns):
    n = len(dmsns) - 1
    dp = [[0] * n for _ in range(n)]
    parenthesization = [[0] * n for _ in range(n)]

    for chain_length in range(2, n + 1):
        for i in range(n - chain_length + 1):
            j = i + chain_length - 1
            dp[i][j] = float('inf')
            for k in range(i, j):
                cost = dp[i][k] + dp[k + 1][j] + dmsns[i] * dmsns[k + 1] * dmsns[j + 1]
                if cost < dp[i][j]:
                    dp[i][j] = cost
                    parenthesization[i][j] = k

    return dp[0][n - 1], optimal_parenthesization(parenthesization, 0, n - 1)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
AAD Practical 10

```
def optimal_parenthesization(parenthesization, i, j):
    if i == j:
        return f'A[{i + 1}]'
    else:
        k = parenthesization[i][j]
        left = optimal_parenthesization(parenthesization, i, k)
        right = optimal_parenthesization(parenthesization, k + 1, j)
        return f'({left}{right})'

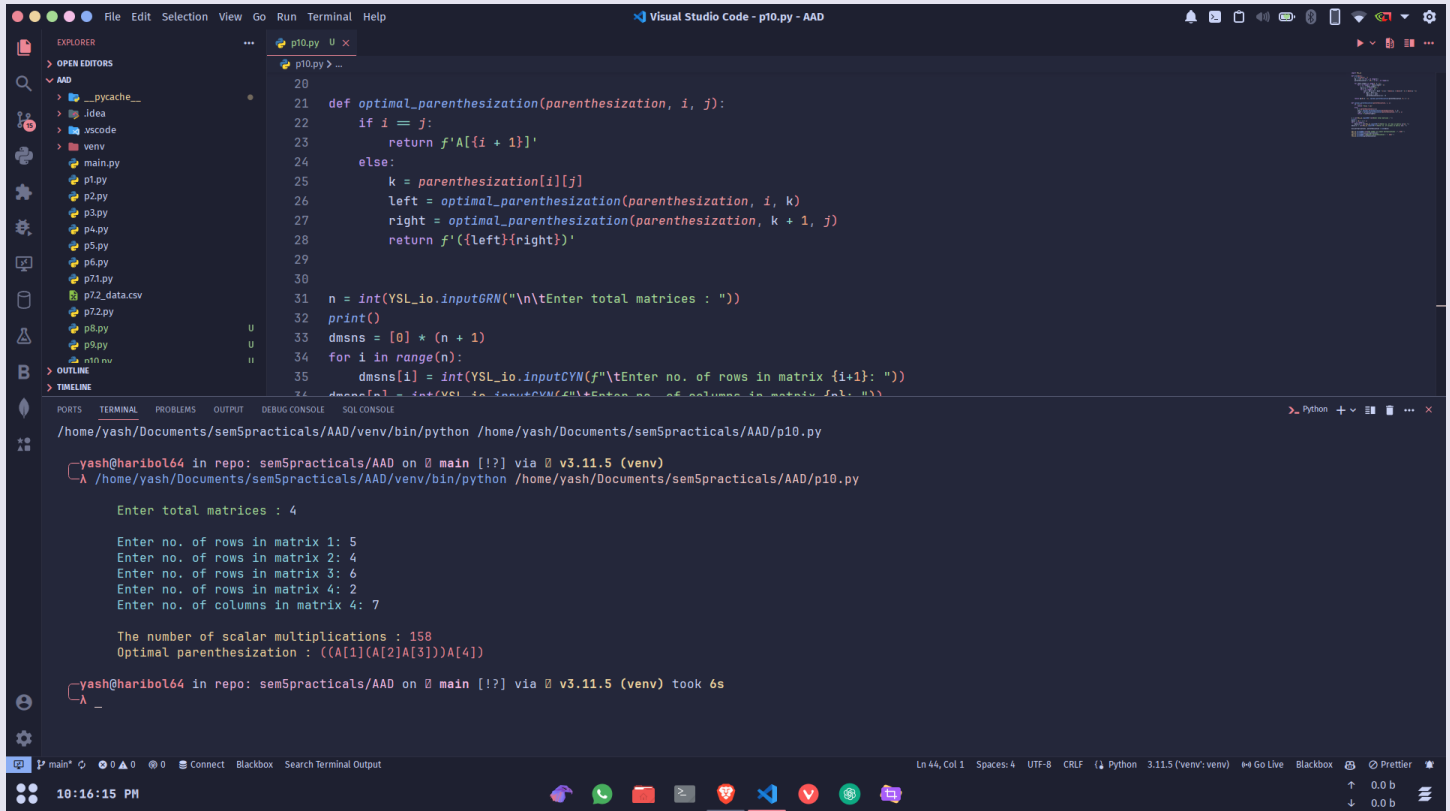
n = int(YSL_io.inputGRN("\n\tEnter total matrices : "))
print()
dmsns = [0] * (n + 1)
for i in range(n):
    dmsns[i] = int(YSL_io.inputCYN(f"\tEnter no. of rows in matrix {i+1}: "))
    dmsns[n] = int(YSL_io.inputCYN(f"\tEnter no. of columns in matrix {n}: "))

min_multiplications, parenthesization = mcm(dmsns)

YSL_io.printORNG("\n\tThe number of scalar multiplications : ", end='')
YSL_io.printRED(min_multiplications)
YSL_io.printORNG("\tOptimal parenthesization : ", end='')
YSL_io.printRED(parenthesization)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
AAD Practical 10

Screenshot :



```
Visual Studio Code - p10.py - AAD

EXPLORER
  AAD
    __pycache__
    .idea
    .vscode
    venv
    main.py
    p1.py
    p2.py
    p3.py
    p4.py
    p5.py
    p6.py
    p71.py
    p72_data.csv
    p72.py
    p8.py
    p9.py
    n10.nv
  OUTLINE
  TIMELINE

p10.py
20
21 def optimal_parenthesization(parenthesization, i, j):
22     if i == j:
23         return f'A[{i + 1}]'
24     else:
25         k = parenthesization[i][j]
26         left = optimal_parenthesization(parenthesization, i, k)
27         right = optimal_parenthesization(parenthesization, k + 1, j)
28         return f'({left}{right})'
29
30
31 n = int(YSL_io.inputGRN("\nEnter total matrices : "))
32 print()
33 dmsns = [0] * (n + 1)
34 for i in range(n):
35     dmsns[i] = int(YSL_io.inputCYW(f"\nEnter no. of rows in matrix {i+1}: "))
36     dmsns[i] = int(YSL_io.inputCYW(f"\nEnter no. of columns in matrix {i+1}: "))

/home/yash/Documents/sem5practicals/AAD/venv/bin/python /home/yash/Documents/sem5practicals/AAD/p10.py

yash@haribol64 in repo: sem5practicals/AAD on  main [!?] via  v3.11.5 (venv)
λ /home/yash/Documents/sem5practicals/AAD/venv/bin/python /home/yash/Documents/sem5practicals/AAD/p10.py

Enter total matrices : 4

Enter no. of rows in matrix 1: 5
Enter no. of rows in matrix 2: 4
Enter no. of rows in matrix 3: 6
Enter no. of rows in matrix 4: 2
Enter no. of columns in matrix 4: 7

The number of scalar multiplications : 158
Optimal parenthesization : ((A[1](A[2]A[3]))A[4])

yash@haribol64 in repo: sem5practicals/AAD on  main [!?] via  v3.11.5 (venv) took 6s
λ _
```