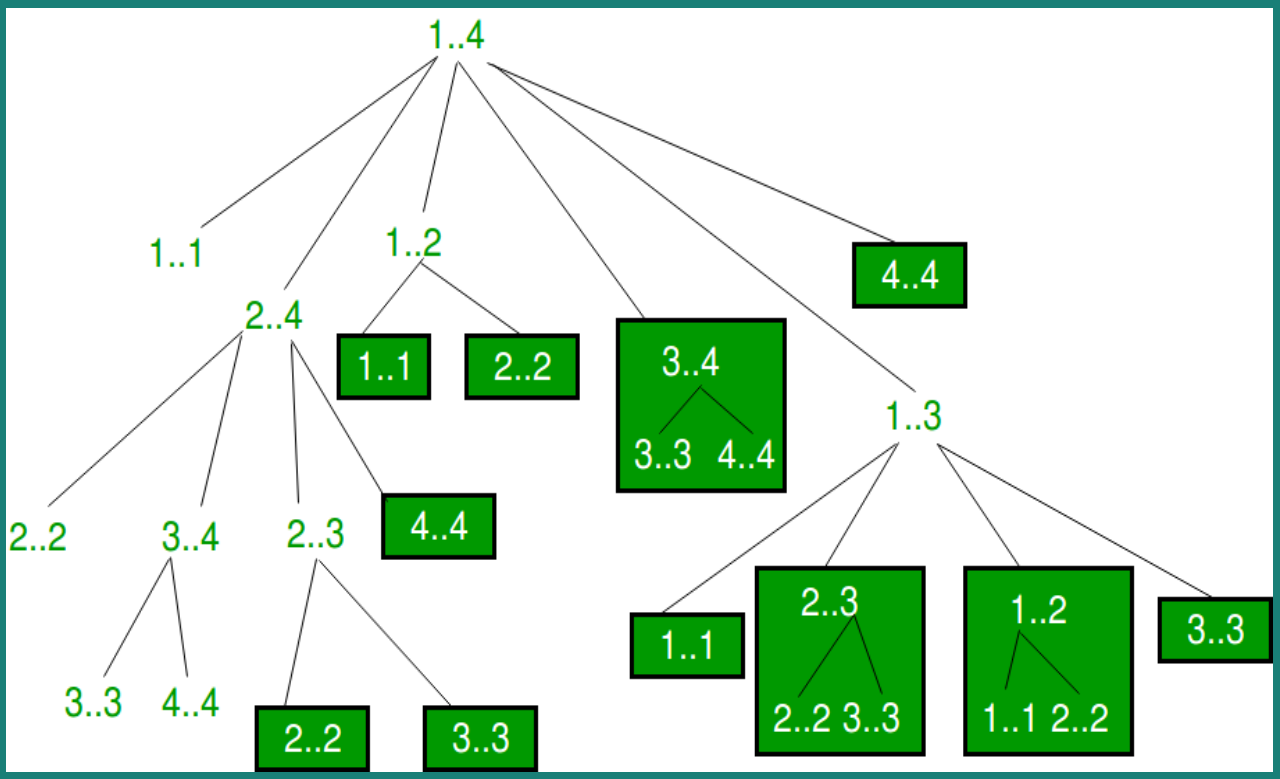


CTA PRACTICAL 8

20MCED08

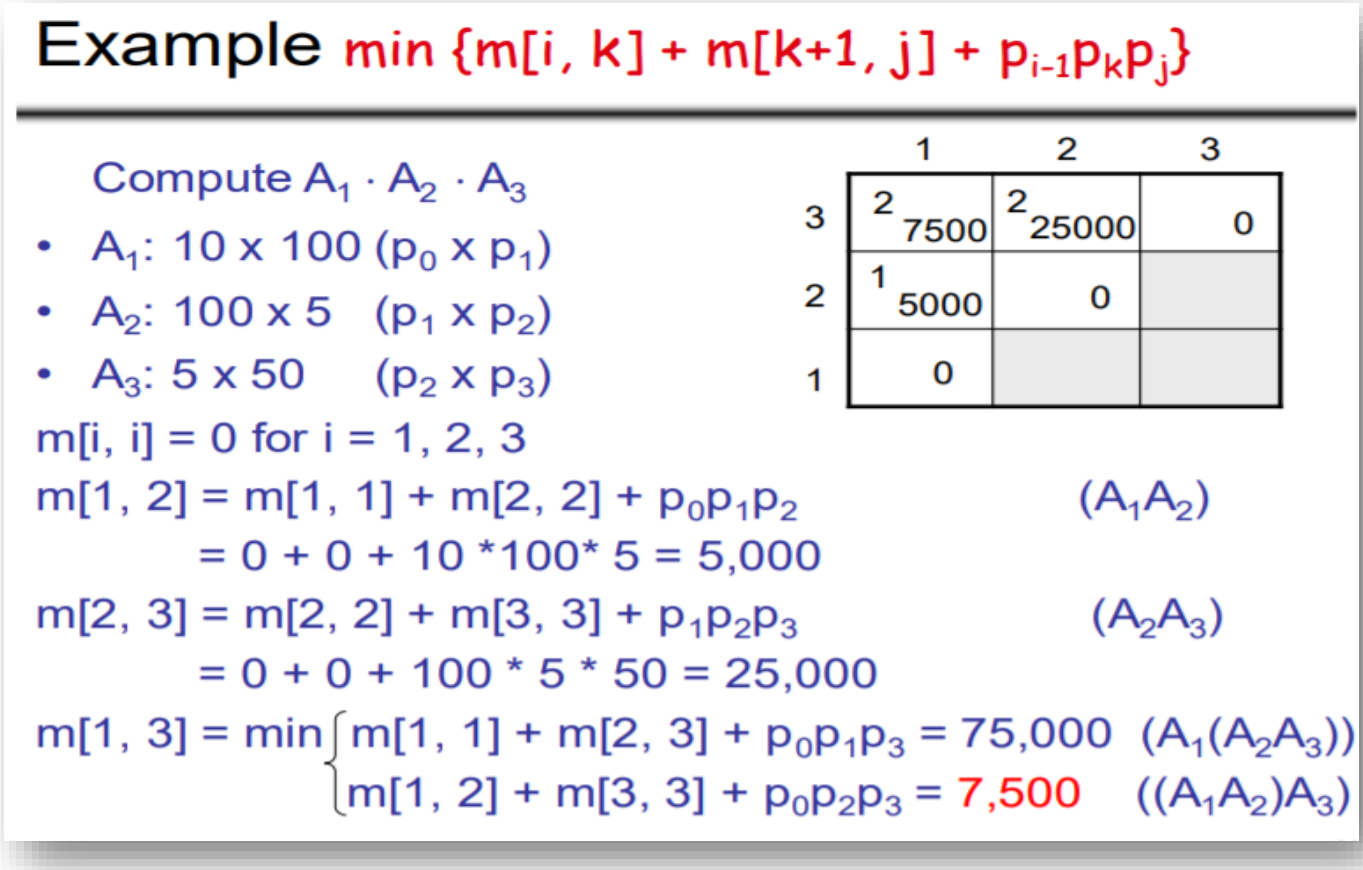
Practical 8

Implement **Matrix Chain Multiplication** using dynamic programming concepts.

Introduction

- Given a sequence of matrices, find the most efficient way to multiply these matrices together.
- The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications.
- We have many options to multiply a chain of matrices because matrix multiplication is associative.
- In other words, no matter how we parenthesize the product, the result will be the same.
- For example, if we had four matrices A, B, C, and D, we would have:

- (ABC)D = (AB)(CD) = A(BCD) = ...



Python Program for Matrix Chain Multiplication

```
import sys
def printParenthesis(m, j, i ):

    # Displaying the parenthesis.
    if j == i:

        # The first matrix is printed as
        # 'A', next as 'B', and so on
        print(chr(65 + j), end = "")
        return;
    else:
        print("(", end = "")

        # Passing (m, k, i) instead of (s, i, k)
        printParenthesis(m, m[j][i] - 1, i)

        # (m, j, k+1) instead of (s, k+1, j)
        printParenthesis(m, j, m[j][i])
        print (")", end = "" )

def MatrixChainOrder(arr, n):
    p = [None for i in range (0,n+1)]
    p[0] = arr[0][0]
    p[n] = arr[n-1][1]

    for i in range(1,len(arr)):
        p[i] = arr[i][0]

    m = [[0 for x in range(n)] for x in range(n)]
    s = [[None for x in range(n)] for x in range(n)]

    for i in range(0, n):
        m[i][i] = 0


    for i in range(0,n-1):
        m[i][i+1] = (p[i]*p[i+1]*p[i+2])


    for L in range(2, n+1):
        for i in range(0, n-L + 1):
            j = i + L-1
#             print(i,j)
            m[i][j] = sys.maxsize
            for k in range(i, j):
#                 print("ijk = ",i , j, k)
                q = m[i][k] + m[k + 1][j] + p[i]*p[k+1]*p[j+1]
#                 print("q = ",q)
                if q < m[i][j]:
                    m[i][j] = q
                    m[j][i] = k + 1
    return m


arr = [[10,100],[100,5],[5,50]]
# arr = [[4,10],[10,3],[3,12],[12,20],[20,7]]
```

```
size = len(arr)
m = MatrixChainOrder(arr, size)
print("Minimum number of multiplications is " +
       str(m[0][size-1]) )
printParenthesis(m, size - 1, 0)

# By Jaynil Patel (20MCED08)
```

Output :

```
Minimum number of multiplications is 7500
```

((AB)C)

Observation:

The time complexity of this tabular method is o(n2).