Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Experiment 5 (Matrix Chain Multiplication)

NAME:DIVYESH KHUNT **SAPID:60009210116**

Aim: Implementation of Matrix Chain Multiplication using Dynamic approach.

Theory:

Given the dimension of a sequence of matrices in an array **arr[]**, where the dimension of the i^{th} matrix is **(arr[i-1] * arr[i])**, the task is to find the most efficient way to multiply these matrices together such that the total number of element multiplications is minimum.

1) Optimal Substructure: In the above case, we are breaking the bigger groups into smaller subgroups and solving them to finally find the minimum number of multiplications. Therefore, it can be said that the problem has optimal substructure property.
2) Overlapping Sub problems: We can see in the recursion tree that the same sub problems are called again and again and this problem has the Overlapping Sub problems property.
So Matrix Chain Multiplication problem has both properties of a dynamic programming problem. So recomputations of same sub problems can be avoided by constructing a temporary array **dp[][]** in a bottom up manner.

Algorithm - Follow the below steps to solve the problem:

- Build a matrix **dp[][]** of size **N*N** for memoization purposes.
- Use the same recursive call as done in the above approach:
    - When we find a range **(i, j)** for which the value is already calculated, return the minimum value for that range (i.e., **dp[i][j]**).
    - Otherwise, perform the recursive calls as mentioned earlier.
- The value stored at **dp[0][N-1]** is the required answer.

Time Complexity: O(n^3)

Example

$$A_{1}{}_{5x4} \cdot A_{2}{}_{4x6} \cdot A_{3}{}_{6x2} = \min((A_{1}{}_{5x4} \cdot A_{2}{}_{4x6}) A_{3}{}_{6x2} , A_{1}{}_{5x4} (A_{2}{}_{4x6} \cdot A_{3}{}_{6x2}))$$

$$= \min(120 + 5x6x2 , 5x4x2 + 48)$$

$$= \min(180 , 88) = 88$$

$$A_{2}{}_{4x6} \cdot A_{3}{}_{6x2} \cdot A_{4}{}_{2x7} = \min((A_{2}{}_{4x6} \cdot A_{3}{}_{6x2}) A_{4}{}_{2x7} , A_{2}{}_{4x6} (A_{3}{}_{6x2} \cdot A_{4}{}_{2x7}))$$

$$= \min(48 + 4x2x7 , 4x6x7 + 84)$$

$$= \min(104 , 252) = 104$$

	0	1	2	3
0	0	120		
1		0	48	
2			0	84
3				0

=>

	0	1	2	3
0	0	120	88	
1		0	48	104
2			0	84
3				0

Code:-

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define MAX_SIZE 100
int m[MAX_SIZE][MAX_SIZE];
int mat(int p[], int i, int j)
{
    if (i == j)
        return 0;
    if (m[i][j] != -1)
        return m[i][j];
    m[i][j] = INT_MAX;
    for (int k = i; k < j; k++)
    {
        int q = mat(p, i, k) + mat(p, k+1, j) + p[i-1]*p[k]*p[j];
        if (q < m[i][j])
            m[i][j] = q;
    }
    return m[i][j];
}
int main()
{
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    for (int i = 0; i < MAX_SIZE; i++)
    {
        for (int j = 0; j < MAX_SIZE; j++)
        {
            m[i][j] = -1;
        }
    }
    printf("Minimum number of multiplications is %d ", mat(arr, 1, size-1));
    return 0;
}
```

OUTPUT:

```
Minimum number of multiplications is 18

...Program finished with exit code 0
Press ENTER to exit console.
```