NAME:	Shubham Vishwakarma
UID No.	2021700071
BRANCH:	S.Y CSE-DS
BATCH:	D
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT No.	5
Date of Performance	13/03/2023
Date of Submission	19/03/2023

AIM:	**Dynamic Programming -Matrix Chain Multiplication**
Program 1	
PROBLEM STATEMENT :	Implement the matrix chain multiplication algorithm for the given problem statement.
ALGORITHM/ THEORY:	Matrix chain multiplication (or Matrix Chain Ordering Problem, MCOP) is an optimization problem that to find the most efficient way to multiply a given sequence of matrices. The problem is not actually to perform the multiplications but merely to decide the sequence of the matrix multiplications involved. The matrix multiplication is associative as no matter how the product is parenthesized, the result obtained will remain the same. For example, for four matrices A, B, C, and D, we would have: ((AB)C)D = ((A(BC))D) = (AB)(CD) = A((BC)D) = A(B(CD)) However, the order in which the product is parenthesized affects the number of simple arithmetic operations needed to compute the product. For example, if A is a 10×30 matrix, B is a 30×5 matrix, and C is a 5×60 matrix, then computing (AB)C needs $(10\times30\times5) + (10\times5\times60) = 1500 + 3000 = 4500$ operations while computing A(BC) needs $(30\times5\times60) + (10\times30\times60) = 9000 + 18000 = 27000$ operations. Clearly, the first method is more efficient.

MATRIX-CHAIN-ORDER (p)

1. n   length[p]-1
2. for i ← 1 to n
3. do m [i, i] ← 0
4. for l ← 2 to n   // l is the chain length
5. do for i ← 1 to n-l + 1
6. do j ← i+ l -1
7. m[i,j] ← ∞
8. for k ← i to j-1
9. do q ← m [i, k] + m [k + 1, j] + p_{i-1} p_k p_j
10. If q < m [i,j]
11. then m [i,j] ← q
12. s [i,j] ← k
13. return m and s.

PROGRAM:

Recursive Approach:

```
#include <stdio.h>
#include <limits.h>
#include <stdio.h>


int MatrixChainOrder(int p[], int i, int j)
{
    if (i == j)
        return 0;
    int k;
    int min = INT_MAX;
    int count;

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

        if (count < min)
            min = count;
    }

    return min;
```

```
}

int main()
{
    int arr[] = { 2, 4, 6, 8, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("\nUsing recursion and optimal subtree solution");
    printf("\nMinimum number of multiplications is %d\n
",MatrixChainOrder(arr, 1, n - 1));
    return 0;
}
```

Dynamic Programming Approach:

```
#include <stdio.h>
#include <limits.h>
#include <stdio.h>
int m[10][10];
int MatrixChainOrder(int p[], int n)
{

    //int m[n][n];
    int i, j, k, L, q;
    // cost is zero when multiplying one matrix.
    for (i = 1; i < n; i++)
        m[i][i] = 0;

    // L is chain length.
    for (L = 2; L < n; L++) {
        for (i = 1; i < n - L + 1; i++) {
            j = i + L - 1;
            m[i][j] = INT_MAX;
            for (k = i; k <= j - 1; k++) {
                // q = cost/scalar multiplications
                q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                if (q < m[i][j])
                    m[i][j] = q;
            }
        }
    }
    printf("\n");
    for (int i = 1; i < n; i++)
    {
        for (int j = 1; j < n; j++)
```

```c
        {
            if (i<=j)
            {
               printf("\t\t%d",m[i][j]);
            }
            else
                printf("\t\t0");
        }
        printf("\n");
    }
    return m[1][n - 1];
}

int main()
{
    int arr[] = { 2 ,4 ,6 ,8 ,6 };
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("\nUsing dynamic programming approach");
    printf("\nMinimum number of multiplications is %d\n
",MatrixChainOrder(arr, size));
    return 0;
}
```

RESULT:

```
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp5\output> & .\'mcm.exe'

Using recursion and optimal subtree solution
Minimum number of multiplications is 240

PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp5\output>
```

```
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp5\output> & .\'mcm1.exe'

Using dynamic programming approach
                0               48              144             240
                0               0               192             384
                0               0               0               288
                0               0               0               0

Minimum number of multiplications is 240

PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp5\output>
```

Self-analysis:

Let matrix be 2×4 4×6 6×8 8×6

solving all
{0,0}, {1,1} and {2,2}
will be zero

i \ j	0	1	2	3
0	0	48	144	240
1		0	192	384
2			0	288
3				0

→ {0,1} = 2×4×6 = 48
→ {1,2} = 4×6×8 = 192
{2,3} = 6×8×6 = 288

→ {0,2} will result of minimum of

→ dp[0][0] + dp[1][2] + (2×4×8) = 0 + 192 + 64 = 256
→ dp[0][1] + dp[2][2] + (2×6×8) = 48 + 0 + 96 = 144

Hence will make dp[0][2] as 144

→ {1,3} will result of minimum of

→ dp[1][1] + dp[2][3] + (4×6×6) = 0 + 288 + 144 = 432
→ dp[1][2] + dp[2][2] + (4×8×6) = 192 + 0 + 192 = 384

Hence will make dp[1][3] as 384

→ {0,3} will result of minimum of

→ dp[0][0] + dp[1][3] + (2×4×6) = 0 + 384 + 48 = 432
→ dp[0][1] + dp[2][3] + (2×6×6) = 48 + 288 + 72 = 408
→ dp[0][2] + dp[3][3] + (2×8×6) = 144 + 0 + 96 = 240

Hence will make dp[0][3] as 240.

CONCLUSION :	In the matrix chain multiplication problem, the minimum number of multiplication steps required to multiply a chain of matrices has been calculated. Determining the minimum number of steps required can highly speed up the multiplication process. It takes O(n3) time and O(n2) auxiliary space to calculate the minimum number of steps required to multiply a chain of matrices using the dynamic programming method.