**Experiment 11- AOA**

Aim: Write a program to implement Matrix chain multiplication using dynamic programming

Theory: 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.

* Given a chain (A1, A2 ,….An) of n matrices, where for i = 1,2, …n, matrix Ai has dimension pi-1 x pi, fully parenthesize the product A1A2A3…An such that the number of scalar multiplications are reduced.
* Note that in the matrix-chain multiplication problem, we are not actually multiplying matrices. Our goal is only to determine an order for multiplying matrices that has the lowest cost.

**Algorithm:**

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] + pi−1pkpj

10 if q < m[i, j]

11 then m[i, j] ← q

12 s[i, j] ← k

13 return m and s

Program : Exp 11: Matrix Chain Multiplication using Dynamic programming

/\* Matrix Ai has dimension p[i-1] x p[i] for i = 1..n \*/

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[] = {1, 2, 3, 4, 3};

    int n = sizeof(arr)/sizeof(arr[0]);

    printf("Minimum number of multiplications is %d ",

                          MatrixChainOrder(arr, 1, n-1));

    getchar();

    return 0;

}

/\*Output

Minimum number of multiplications is 30

\*/