Sardar Patel Institute of Technology

SEM IV: DESIGN AND ANALYSIS OF ALGORITHMS.

Name	SMRUTI SONEKAR
UID no.	2021700064
BRANCH:	SE CSE (DATA SCIENCE)
Experiment No.	5

TOPIC:	MATRIX CHAIN MULTIPLICATION.
QUERY:	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)) There will be given a sequence of matrices; your task will determine the most efficient technique to multiply them together.

Structure of an optimal parameterization: Our first step in the dynamic paradigm is to find the optimal substructure and then use it to construct an optimal solution to the problem from an optimal solution to subproblems.

Let $A_{i....j}$ where i≤ j denotes the matrix that results from evaluating the product

$A_i A_{i+1}....A_j$.

If i < j then any parameterization of the product $A_i A_{i+1}$ .....A_i must split that the product between A_k and A_{k+1} for some integer k in the range i ≤ k ≤ j. That is for some value of k, we first compute the matrices $A_{i.....k}$ & $A_{k+1....j}$ and then multiply them together to produce the final product $A_{i....j}$. The cost of computing $A_{i....k}$ plus the cost of computing $A_{k+1....j}$ plus the cost of multiplying them together is the cost of parameterization.

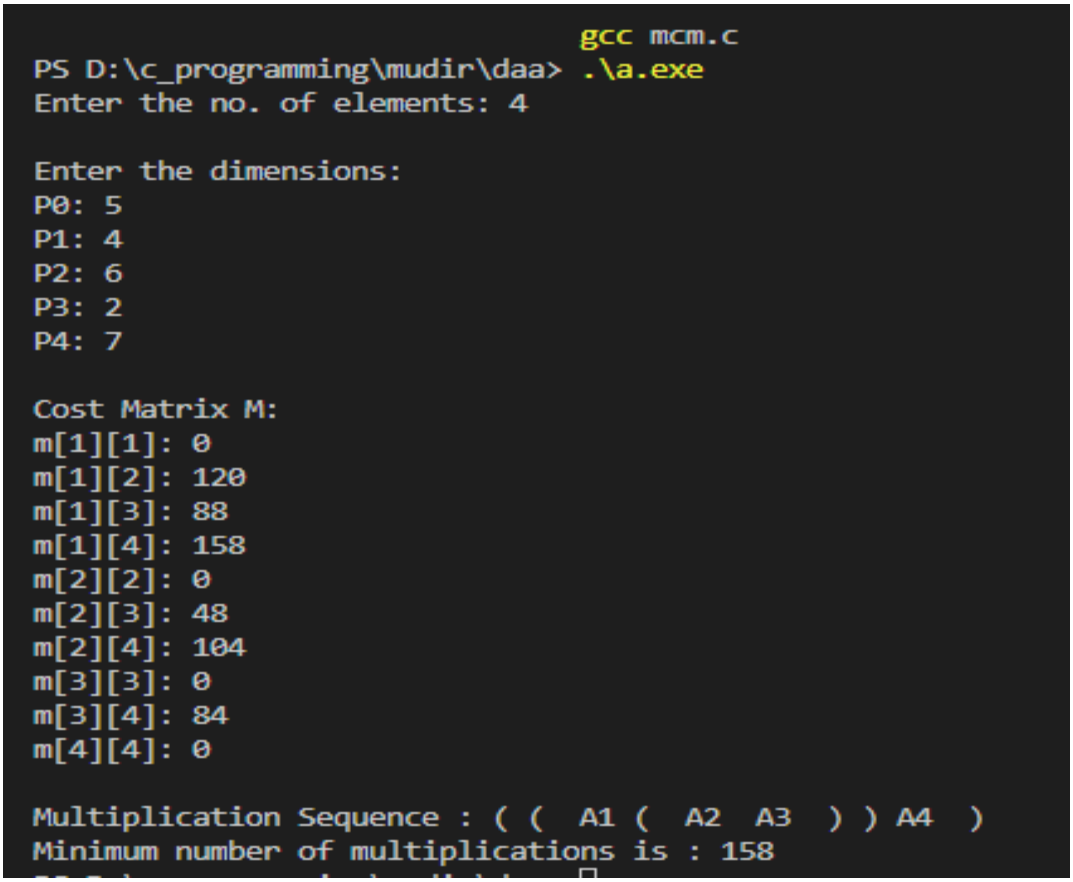
- m [i,j] = m [i, k] + m [k + 1, j] + pi–1 * pk *pj.

PROGRAM:

```c
#include <stdio.h>
#include<stdlib.h>
#include<limits.h>
#define INFY 999999999
long int m[20][20];
int s[20][20];
int p[20],i,j,n;

void print_optimal(int i,int j)
{
if (i == j)
printf(" A%d ",i);
else
   {
      printf("( ");
      print_optimal(i, s[i][j]);
      print_optimal(s[i][j] + 1, j);
      printf(" )");
   }
}

void matmultiply(void)
{
long int q;
int k;
```

```
for(i=n;i>0;i--)
 {
   for(j=i;j<=n;j++)
    {
     if(i==j)
       m[i][j]=0;
     else
       {
        for(k=i;k<j;k++)
        {
         q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
         if(q<m[i][j])
          {
            m[i][j]=q;
            s[i][j]=k;
          }
         }
        }
      }
 }
}

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 minimum count
    return min;
}
```

```
void main()
{
int k,upper,lower;
printf("Enter the no. of elements: ");
scanf("%d",&n);

for(i=1;i<=n;i++)
for(j=i+1;j<=n;j++)
{
 m[i][i]=0;
 m[i][j]=INFY;
 s[i][j]=0;
}
printf("\nEnter the dimensions: \n");
for(k=0;k<=n;k++)
{

 printf("P%d: ",k);
 scanf("%d",&p[k]);
}
matmultiply();
printf("\nCost Matrix M:\n");
for(i=1;i<=n;i++)
 for(j=i;j<=n;j++)
  printf("m[%d][%d]: %ld\n",i,j,m[i][j]);


i=1,j=n;
printf("\nMultiplication Sequence : ");
print_optimal(i,j);
printf("\nMinimum number of multiplications is : %d ",
                          MatrixChainOrder(p, 1, n));
}
```

RESULT:	
CONCLUSION:	Dynamic programming concept is applied on matrix chain multiplication algorithm for optimal solution .