Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

(Autonomous College Affiliated to University of Mumbai)

PRACTICAL NO. 5	DESIGN AND ANALYSIS OF ALGORITHMS
NAME	VEDANTI ANIL WADATKAR
UID	2021700072
BATCH	D4
PROBLEM STATEMENT	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.
ALGORITHM	• Iterate from l = 2 to N-1 which denotes the length of the range • Iterate from i = 0 to N-1: • Find the right end of the range (j) having l matrices. • Iterate from k = i+1 to j which denotes the point of partition. • Multiply the matrices in range (i, k) and (k, j). • This will create two matrices with dimensions arr[i-1]*arr[k] and arr[k]*arr[j]. • The number of multiplications to be performed to multiply these two matrices (say X) are arr[i-1]*arr[k]*arr[j]. • The total number of multiplications is dp[i][k]+ dp[k+1][j] + X. The value stored at dp[1][N-1] is the required answer.
CODE	#include <stdio.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;
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 \n",
                        MatrixChainOrder(p, 1, n));


}
```

OUTPUT

```
itlab@itlab-OptiPlex-3010:~$ gcc v.c
itlab@itlab-OptiPlex-3010:~$ ./a.out
Enter the no. of elements: 3

Enter the dimensions:
P0: 1
P1: 2
P2: 3
P3: 4

Cost Matrix M:
m[1][1]:0
m[1][2]:6
m[1][3]:18
m[2][2]:0
m[2][3]:24
m[3][3]:0

Multiplication Sequence : ( (  A1  A2  ) A3  )
Minimum number of multiplications is : 18
itlab@itlab-OptiPlex-3010:~$
```