

DAA EXP 5

NAME: Rahul Chalwadi

UID: 2021700012

CSE-DS

AIM:

Implement matrix chain multiplication using dynamic programming.

THEORY:

matrices of size $m \times n$ and $n \times p$ when multiplied, they generate a matrix of size $m \times p$ and the number of multiplications performed are $m \times n \times p$. Now, for a given chain of N matrices, the first partition can be done in $N-1$ ways. For example, sequence of matrices A, B, C and D can be grouped as $(A)(BCD)$, $(AB)(CD)$ or $(ABC)(D)$ in these 3 ways. So a range $[i, j]$ can be broken into two groups like $\{[i, i+1], [i+1, j]\}$, $\{[i, i+2], [i+2, j]\}$, \dots , $\{[i, j-1], [j-1, j]\}$. Each of the groups can be further partitioned into smaller groups and we can find the total required multiplications by solving for each of the groups. The minimum number of multiplications among all the first partitions is the required answer.

ALGORITHM:

MatrixChainMultiplication(mat[], low, high):

// 0 steps are required if low equals high

// case of only 1 sized sub-problem.

If(low=high): return 0

// Initialize minCost to a very big number.

minCost = Infinity

// Iterate from k = low to k = high-1

For(k=low to high-1):

/* Cost = Cost of Multiplying chain on left side + Cost of Multiplying chain on right side + Cost of Multiplying matrix obtained from left and right side. */

```
cost=MatrixChainMultiplication(mat, low, k)+ MatrixChainMultiplication(mat,
low+1, high)+ mat[low-1]*mat[k]*mat[high]
```

```
// Update the minCost if cost<minCost. If(cost<minCost): minCost=cost return
minCost
```

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;

```

```

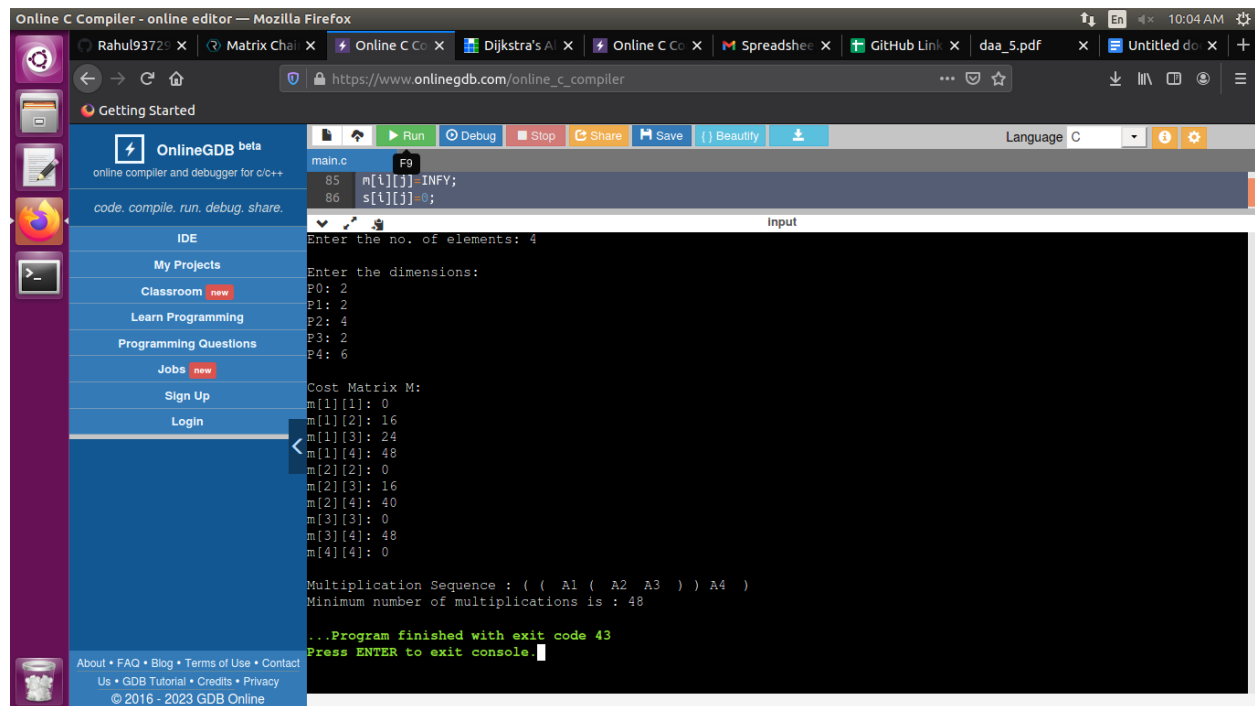
printf("Enter the no. of elements: ");
scanf("%d",&n);
for(i=1;i<=n;i++)
for(j=i+1;j<=n;j++)
{
    m[i][i]=0;
    m[i][j]=INFY;
    s[i][j]=0;
}
printf("\nEnter the dimensions: \n");
for(k=0;k<=n;k++)
{
    printf("P%d: ",k);
    scanf("%d",&p[k]);
}
matmultiply();
printf("\nCost Matrix M:\n");
for(i=1;i<=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 ",
        MatrixChainOrder(p, 1, n));

}

```

OUTPUT:



```
main.c
85 m[i][j] = INFY;
86 s[i][j] = 0;

Enter the no. of elements: 4

Enter the dimensions:
P0: 2
P1: 2
P2: 4
P3: 2
P4: 6

Cost Matrix M:
m[1][1]: 0
m[1][2]: 16
m[1][3]: 24
m[1][4]: 48
m[2][2]: 0
m[2][3]: 16
m[2][4]: 40
m[3][3]: 0
m[3][4]: 48
m[4][4]: 0

Multiplication Sequence : ( ( A1 ( A2 A3 ) ) A4 )
Minimum number of multiplications is : 48

...Program finished with exit code 43
Press ENTER to exit console.
```

CONCLUSION:

IN this experiment i HAVE UNDERSTOOD matrix chain multiplication.