| | |
|---|---|
| **Name** | Ronak Thakar |
| **UID no.** | 2021700066 |
| **Experiment No.** | 5 |

| | |
|---|---|
| **AIM:** | Dynamic Programming - Matrix Chain Multiplication. |
| **Program** | |
| **PROBLEM STATEMENT:** | Use Dynamic Programming method to find the optimal way to multiply(parenthesize) the matrices to find the minimum number of multiplications required to solve the matrix. |
| **ALGORITHM/ THEORY:** | Dynamic Programming is a technique in computer programming that helps to efficiently solve a class of problems that have overlapping sub-problems and optimal substructure property. If any problem can be divided into sub-problems, which in turn are divided into smaller sub-problems, and if there are overlapping among these subproblems, then the solutions to these sub-problems can be saved for future reference. The approach of solving problems using dynamic programming algorithm has following steps:<br>1. Characterize the structure of an optimal solution.<br>2. Recursively define the value of an optimal solution.<br>3. Compute the value of an optimal solution, typically in a bottom-up fashion.<br>4. Construct an optimal solution from computed information.<br>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.<br>Note: Here we just find the way to multiply them but we don’t multiply the content of matrices as such.<br>1] Optimal Substructure: Here we break the number of matrices into smaller groups and solve them to find the minimum number of multiplications.<br>2] Recursive method: We use recursive call to find the possible ways to multiply them and solve them. The recursive formula is:<br>c[i, j] = 0 if i=j<br>= min(c[i, j],c[i, k] + c[k+1, j] + p[i-1]*p[k]*p[j]) if i<=k<=j<br>3] Computing the optimal cost<br>4] Constructing an optimal solution. |

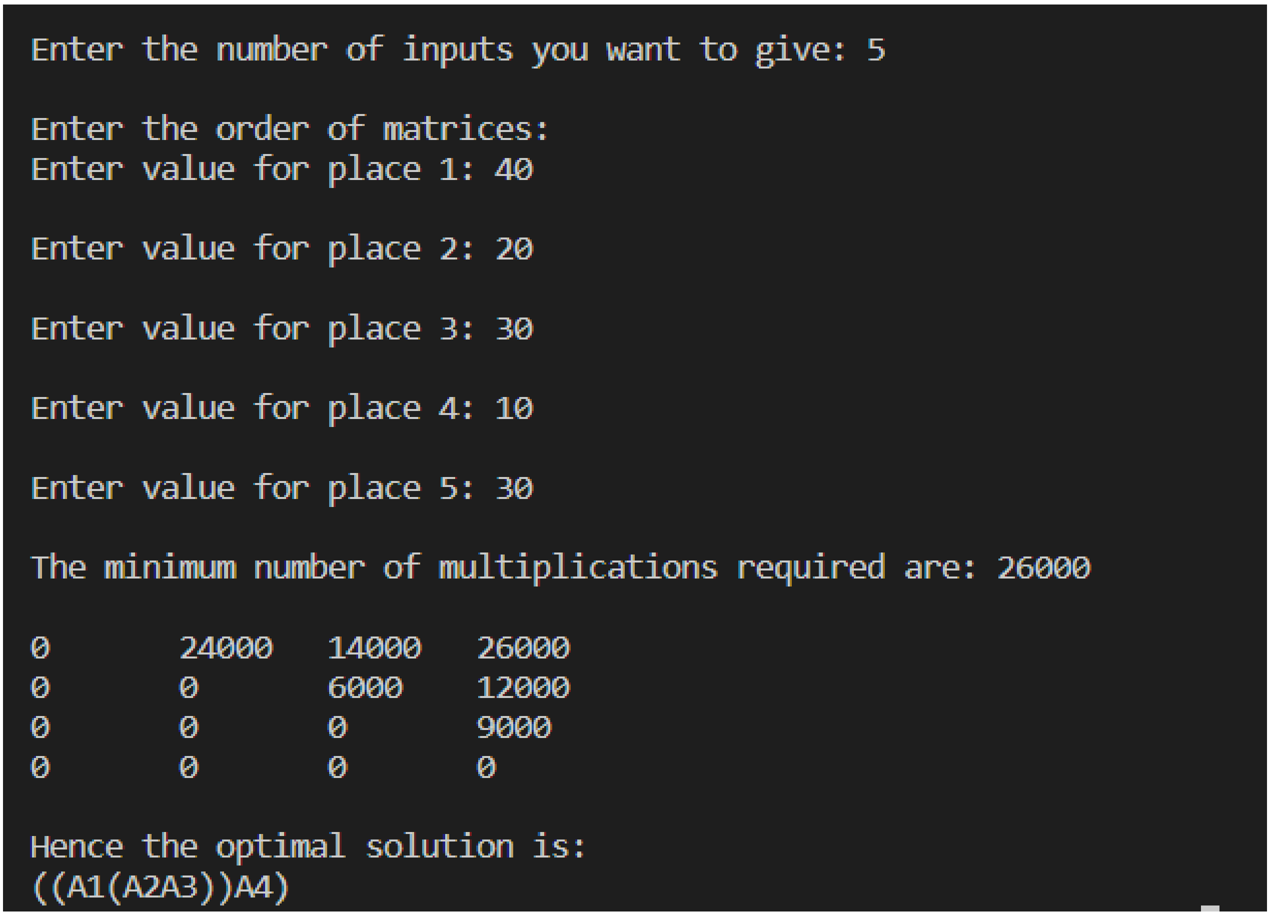
```
if(i==j)
      printf("A%d",i);
else{
      printf("(");
      POP(i,s[i][j]);
      POP(s[i][j]+1,j);
      printf(")");
}
```

**PROGRAM:**

```
#include<stdio.h>

int mat[100][100],s[100][100],count=0;

int MCM(int p[], int i, int j){
    if(i==j){
      mat[i][j] = 0;
      return 0;
    }
    mat[i][j] = 30000;
    for(int k=i; k<j; k++){
       count = MCM(p,i,k) + MCM(p,k+1,j) + p[i-1]*p[k]*p[j];
       if(count<mat[i][j]){
         mat[i][j] = count;
         s[i][j] = k;
       }
    }
    return mat[i][j];
}

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

void main(){
    int num;
    printf("\nEnter the number of inputs you want to give: ");
    scanf("%d",&num);
    int p[num];
    printf("\nEnter the order of matrices: ");
    for(int i=0;i<num;i++){
```

```c
        printf("\nEnter value for place %d: ",i+1);
        scanf("%d",&p[i]);
    }
    printf("\nThe minimum number of multiplications required are: %d\n\n",MCM(p,1,num-1));
    for(int i=1;i<num;i++){
        for(int j=1;j<num;j++){
            printf("%d\t",mat[i][j]);
        }
        printf("\n");
    }
    printf("\nHence the optimal solution is: \n");
    POP(1,num-1);
}
```

**RESULT:**



```
Enter the number of inputs you want to give: 5

Enter the order of matrices:
Enter value for place 1: 40

Enter value for place 2: 20

Enter value for place 3: 30

Enter value for place 4: 10

Enter value for place 5: 30

The minimum number of multiplications required are: 26000

0       24000   14000   26000
0       0       6000    12000
0       0       0       9000
0       0       0       0

Hence the optimal solution is:
((A1(A2A3))A4)
```

**CONCLUSION:** We used Dynamic Programming steps to solve Matrix Chain Multiplication problem.