DESIGN AND ANALYSIS OF ALGORITHMS

EXPERIMENT 5

Name: Janhavi R Deshmukh

Branch: CSE(DS) D1

UID: 2021700017

Aim: – Experiment on dynamic programming- Matrix Chain Multiplication.

Theory:

Matrix Chain Multiplication can be solved using dynamic programming. We can define the minimum number of scalar multiplications needed to iteratively compute the product of a chain of matrices. We start with sub chains of length 1 and then compute the minimum cost for sub chains of increasing length until we have the minimum cost for the entire chain. The time complexity of this algorithm is O(n^3), where n is the number of matrices in the chain.

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] + p_{i-1} p_k p_j
10. If q < m [i,j]
11. then m [i,j] ← q
12. s [i,j] ← k
13. return m and s.
```

PRINT-OPTIMAL-PARENS (s, i, j)

```
1. if i=j
2. then print "A"
3. else print "("
4. PRINT-OPTIMAL-PARENS (s, i, s [i, j])
5. PRINT-OPTIMAL-PARENS (s, s [i, j] + 1, j)
6. print ")
```

Program:

```
#include<stdio.h>
#include <stdlib.h>
#include <math.h>
#include <limits.h>

int
s[20][20],m[20][20],p[20];
int n;
void print(int i,int j){
if (i == j)
printf(" M%d ",i);
else
   {
      printf("(");
      print(i, s[i][j]);
      print(s[i][j] + 1,
j);
      printf(")");
   }
}
void multiply(){
int q,k;
for(int i=n;i>0;i--)
 {
   for(int j=i;j<=n;j++)
    {
     if(i==j)
       m[i][j]=0;
     else
       {
        for(int
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 chain(int p[], int i,
int j)
{
    if(i == j)
        return 0;
    int
k,min=INT_MAX,count=0;
    for (k = i; k < j; k++)
{
        count = chain(p, i,
k) + chain(p, k + 1, j) +
p[i - 1] * p[k] * p[j];
        if (count < min)
            min = count;
    }
    return min;
}
int main(){
printf("\n\t--MATRIX CHAIN
MULTIPLICATION--\n");
printf("\nEnter the no of
matrices: ");
scanf("%d",&n);
printf("\nThe dimensions of
matrices are: \n");
    for (int i = 0; i<=n;
i++) {
        p[i]= (rand()%(46 -
15 + 1)) + 15;
        printf("%d ",
p[i]);
    }
for(int i=1;i<=n;i++)
```

```
for(int j=i+1;j<=n;j++)
{
 m[i][i]=0;
 m[i][j]=INT_MAX;
 s[i][j]=0;
}
multiply();
printf("\n\nMultiplication
Sequence : ");
print(1,n);
printf("\n\nMinimum/Optimal
number of multiplications
is %d\n",chain(p, 1, n));
return 0;
}
```

Output:

```
Enter the no of matrices: 7

The dimensions of matrices are:
22 21 24 34 32 46 25 27

Multiplication Sequence : (( M1 ((( M2  M3 ) M4 )( M5  M6 ))) M7 )

Minimum/Optimal number of multiplications is 119984
```

Conclusion:

I understood how to find optimal parenthesization of a matrix chain. Also understood how dynamic programming approach gives the time complexity as $O(n^3)$ where the recursive approach was giving exponential time complexity.