DIGITAL ANALYSIS AND ALGORITHM

EXPERIMENT – 05

NAME :- MANTHAN AYALWAR

UID :- 2021700003

BATCH :- D1

Aim : Experiment using dynamic programming approach: finding optimal parenthesization for matrix chain multiplication

Algorithm :

```
MATRIX-CHAIN-ORDER(p)
 1  n = p.length − 1
 2  let m[1..n, 1..n] and s[1..n − 1, 2..n] be new tables
 3  for i = 1 to n
 4      m[i, i] = 0
 5  for l = 2 to n                    // l is the chain length
 6      for i = 1 to n − l + 1
 7          j = i + l − 1
 8          m[i, j] = ∞                                          *
 9          for k = i to j − 1
10              q = m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
11              if q < m[i, j]
12                  m[i, j] = q
13                  s[i, j] = k
14  return m and s
```

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

Code :

```
#include <stdio.h>
```

```
#include <limits.h>
#include <stdlib.h>
void printParanthesis(int **s, int i, int j)
{
if (i == j)
printf("A%d", i + 1);
else
{
printf("(");
printParanthesis(s, i, s[i][j]);
printf("*");
printParanthesis(s, s[i][j] + 1, j);
printf(")");
}
}
void parenthesizeMatrixChain(int *p, int n)
{
int m[n][n];
// int s[n][n];
int **s = malloc(sizeof(int *) * n);
for (int i = 0; i < n; i++)
s[i] = malloc(sizeof(int) * n);
for (int i = 0; i < n; i++)
m[i][i] = 0;
int j, cost;
for (int chain_length = 2; chain_length <= n;
chain_length++)
```

```
{
for (int i = 0; i <= n - chain_length; i++) // i
is the starting index of a matrix subchain
{
j = i + chain_length - 1; // j is the ending
index of the matrix subchain
m[i][j] = INT_MAX;
for (int k = i; k <= j - 1; k++)
{
cost = m[i][k] + m[k + 1][j] + p[i] * p[k
+ 1] * p[j + 1];
if (cost < m[i][j])
{
m[i][j] = cost;
s[i][j] = k;
}
}
}
}
printParanthesis(s, 0, n - 1);
for (int i = 0; i < n; i++)
free(s[i]);
free(s);
}
int main()
{
printf("Enter the number of matrices: ");
```

```
int n;
scanf("%d", &n);
int p[n + 1];
printf("Enter the array of matrix dimensions: ");
for (int i = 0; i < n + 1; i++)
scanf("%d", p + i);
parenthesizeMatrixChain(p, n);
}
```

Output :

```
Enter the number of matrices: 5
Enter the array of matrix dimensions: 6
2
7
9
3
5
(A1*(((A2*A3)*A4)*A5))
```

Conclusion : From this experiment, I understand the concept of optimal parenthesization for matrix chain multiplication .