BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre

EXPERIMENT NO: 05

EXPERIMENT TITLE: To implement dynamic algorithms
5.1 To implement Matrix chain multiplication

Objective:
1.To Implement Matrix chain multiplication

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Program code: -

```
import java.util.Scanner;

public class MatrixChainMultiplication {
    static void matrixChainOrder(int[] p, int n) {
        int[][] m = new int[n][n];
        int[][] s = new int[n][n];

        for (int L = 2; L < n; L++) {
            for (int i = 1; i < n - L + 1; i++) {
                int j = i + L - 1;
                m[i][j] = Integer.MAX_VALUE;
                for (int k = i; k < j; k++) {
                    int 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;
                    }
                }
            }
        }

        System.out.print("Optimal Parenthesis is: ");
        printOptimalParens(s, 1, n - 1);
        System.out.println();

        System.out.println("\nDP Table (Minimum Cost of Multiplications):");
        printDPTable(m, n);
    }

    static void printOptimalParens(int[][] s, int i, int j) {
        if (i == j) {
            System.out.print("A" + i);
        } else {
            System.out.print("(");
            printOptimalParens(s, i, s[i][j]);
            printOptimalParens(s, s[i][j] + 1, j);
            System.out.print(")");
        }
    }

    static void printDPTable(int[][] m, int n) {
        for (int i = 1; i < n; i++) {
            for (int j = 1; j < n; j++) {
                if (i > j) System.out.print("\t");
                else System.out.print(m[i][j] + "\t");
```

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

```
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of matrices: ");
        int numMatrices = sc.nextInt();
        int[] dimensions = new int[numMatrices + 1];

        System.out.print("Enter dimensions of matrix A1 (format: rows
columns): ");
        int rows = sc.nextInt();
        int cols = sc.nextInt();
        dimensions[0] = rows;
        dimensions[1] = cols;

        for (int i = 2; i <= numMatrices; i++) {
            System.out.print("Enter dimensions of matrix A" + i + " (format:
" + cols + " columns): ");
            rows = sc.nextInt();
            if (rows != cols) {
                System.out.println("Invalid dimensions! Number of rows in
matrix A" + i + " must be " + cols + ".");
                i--;
                continue;
            }
            cols = sc.nextInt();
            dimensions[i] = cols;
        }

        matrixChainOrder(dimensions, numMatrices + 1);
        sc.close();
    }
}
```

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Output:

```
Enter the number of matrices: 4
Enter dimensions of matrix A1 (format: rows columns): 5 4
Enter dimensions of matrix A2 (format: 4 columns): 4 6
Enter dimensions of matrix A3 (format: 6 columns): 6 2
Enter dimensions of matrix A4 (format: 2 columns): 2 7
Optimal Parenthesis is: (A1((A2A3)A4))

DP Table (Minimum Cost of Multiplications):
0   120 88  104
    0   48  64
        0   24
            0

Process finished with exit code 0
```

Conclusion:

In this practical session, I explored how dynamic programming optimizes matrix chain multiplication. The approach involves storing intermediate results in a table and evaluating different split points to reduce the total number of scalar multiplications. By taking user input and computing the optimal solution, I saw how careful planning and efficient computation can tackle complex problems effectively.

UCID: 2024510001