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

LAB ESE Examination

EXPERIMENT TITLE 1: To implement Travelling salesman problem.
EXPERIMENT TITLE 2: To implement Matrix chain Multiplication.

Explanation:

1) Travelling salesman problem.
   The Travelling Salesman Problem is a classic combinatorial optimization problem. Given a set of cities and the cost of travel between every pair of cities, the goal is to find the shortest possible route that visits each city exactly once and returns to the starting city.
   TSP is NP-hard, meaning there is no known polynomial-time solution for large numbers of cities. In this code, the branch and bound technique is used to efficiently prune paths that cannot yield a better solution than the best found so far

2) Matrix chain Multiplication.
   Matrix Chain Multiplication is an optimization problem that determines the most efficient way to multiply a given sequence of matrices. The order of multiplication affects the total number of scalar multiplications required, but the result is the same regardless of the order.
   The problem is solved using dynamic programming by breaking it into subproblems, storing the minimum cost for multiplying each sub-chain of matrices, and using this information to build up the solution for the entire chain.

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

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

CODE:

1) Travelling salesman problem

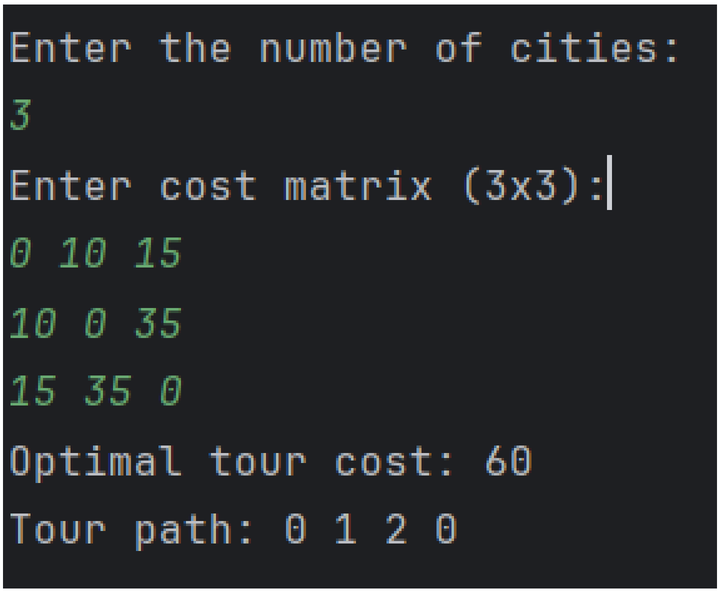
```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class travellingSalesman {
    public static int n;
    public static int[][] cost;
    public static boolean[] visited;
    public static int[] bestTour;
    public static int bestCost = Integer.MAX_VALUE;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of cities: ");
        n = sc.nextInt();
        cost = new int[n][n];
        System.out.println("Enter cost matrix (" + n + "x" + n + "):");
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                cost[i][j] = sc.nextInt();
        visited = new boolean[n];
        bestTour = new int[n + 1];
        solve();
        printResult();
    }

    public static void solve() {
        visited[0] = true;
        ArrayList<Integer> path = new ArrayList<>();
        path.add(0);
        branchAndBound(0, 1, 0, path);
    }
    private static void branchAndBound(int currentCity, int level, int currentCost,
List<Integer> path) {
        if (level == n) {
            int totalCost = currentCost + cost[currentCity][0];
            if (totalCost < bestCost) {
                bestCost = totalCost;
                for (int i = 0; i < n; i++)
                    bestTour[i] = path.get(i);
                bestTour[n] = 0;
            }
            return;
        }
        for (int next = 0; next < n; next++) {
            if (!visited[next]) {
                int tempCost = currentCost + cost[currentCity][next];
                if (tempCost < bestCost) {
                    visited[next] = true;
                    path.add(next);
                    branchAndBound(next, level + 1, tempCost, path);
                    path.remove(path.size() - 1);
                    visited[next] = false;
                }
            }
        }
    }
    public static void printResult() {
        System.out.println("Optimal tour cost: " + bestCost);
        System.out.print("Tour path: ");
        for (int city : bestTour) System.out.print(city + " ");
        System.out.println();
    }

}
```

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 cities:
3
Enter cost matrix (3x3):
0 10 15
10 0 35
15 35 0
Optimal tour cost: 60
Tour path: 0 1 2 0
```

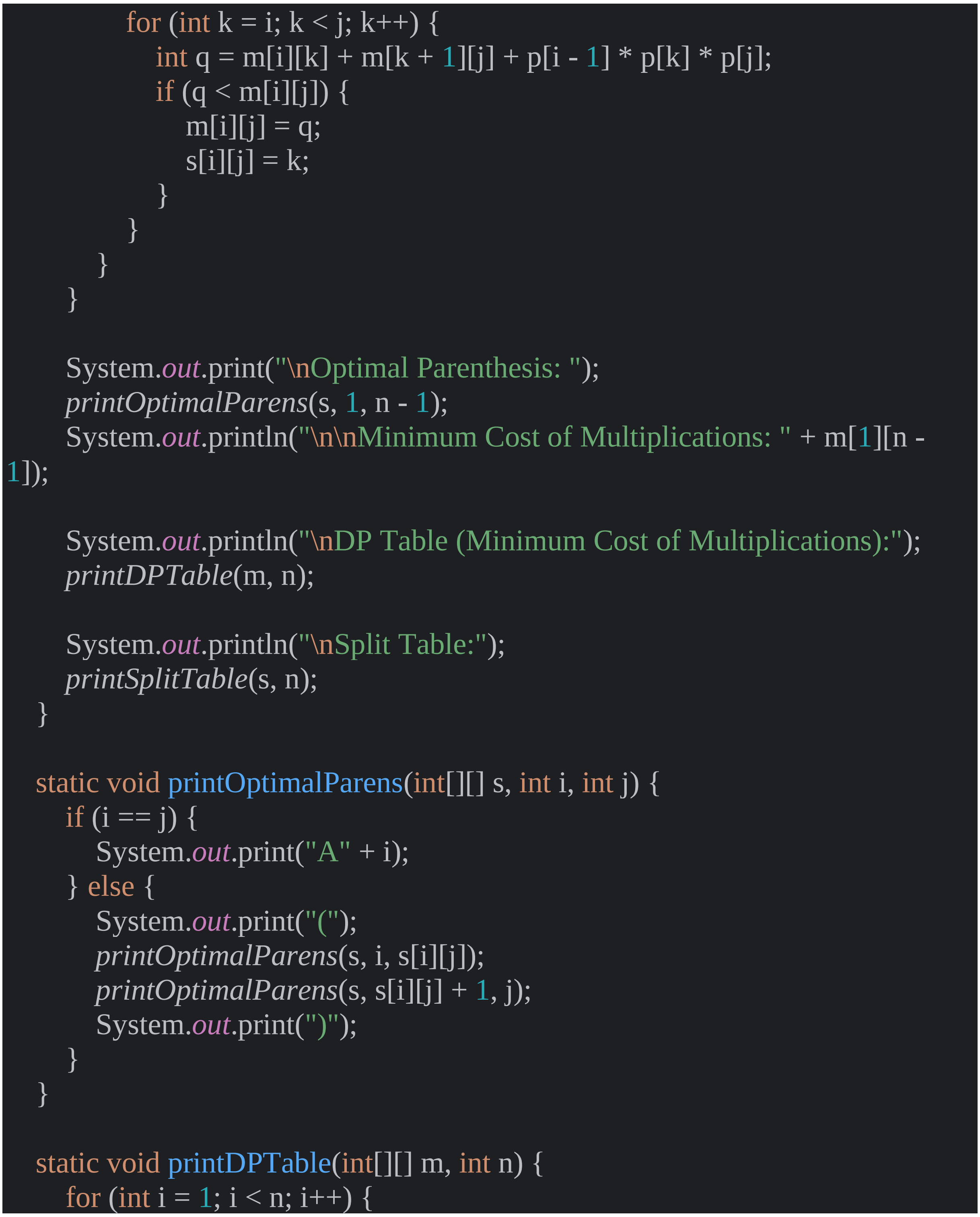
Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

2) Matrix chain Multiplication

```
`
import java.util.Scanner;

public class matrixChainMultiplication {

    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.println("Enter the array of dimensions (space-separated): ");
        for (int i = 0; i <= numMatrices; i++) {
            dimensions[i] = sc.nextInt();
        }

        System.out.println("\nMatrix dimensions:");
        for (int i = 1; i <= numMatrices; i++) {
            System.out.println("A" + i + " : " + dimensions[i - 1] + " x " + dimensions[i]);
        }

        matrixChainOrder(dimensions, numMatrices + 1);
    }

    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;
```



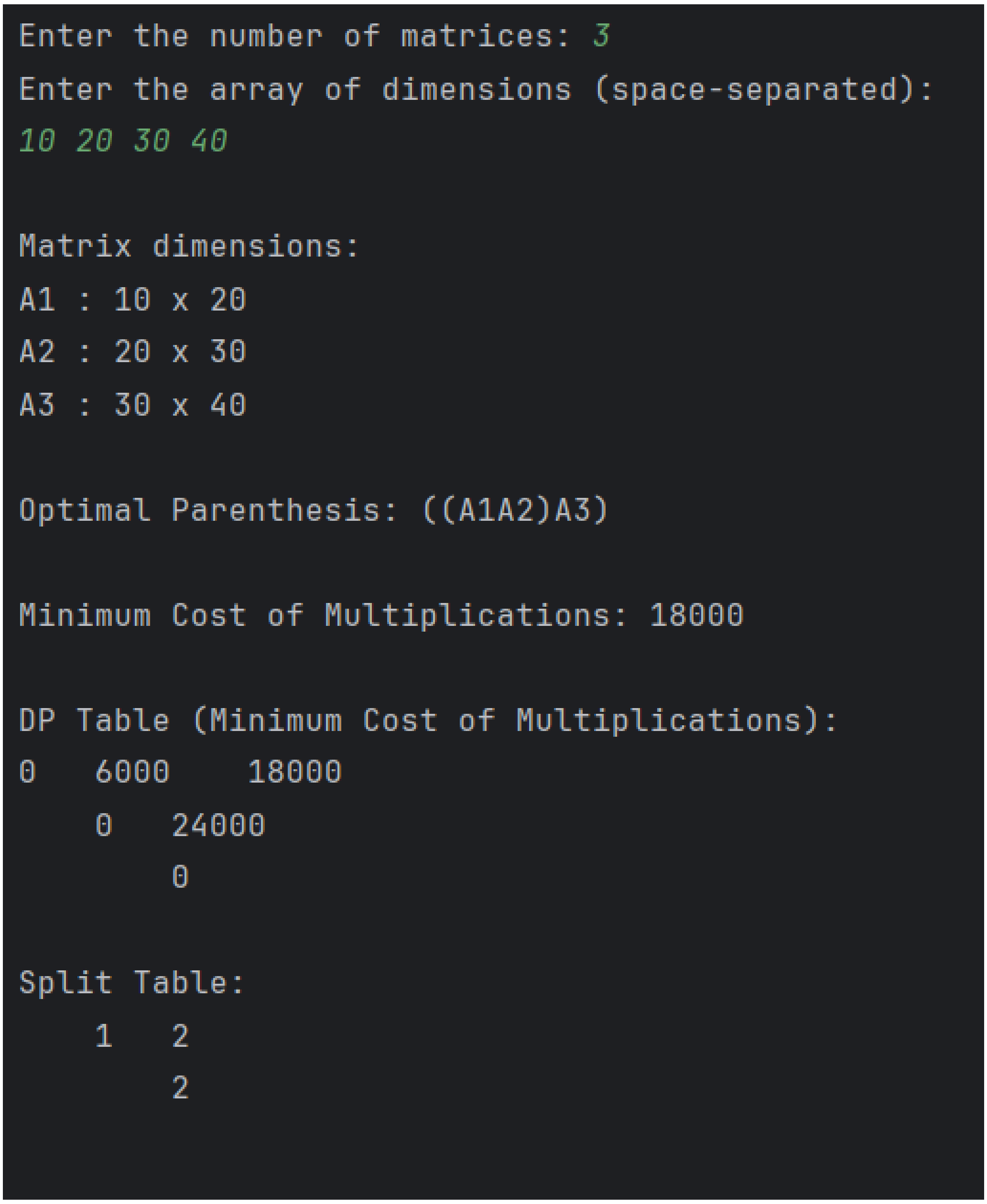
```
                for (int k = i; k < j; k++) {
                    int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                    if (q < m[i][j]) {
                        m[i][j] = q;
                        s[i][j] = k;
                    }
                }
            }
        }

        System.out.print("\nOptimal Parenthesis: ");
        printOptimalParens(s, 1, n - 1);
        System.out.println("\n\nMinimum Cost of Multiplications: " + m[1][n - 1]);

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

        System.out.println("\nSplit Table:");
        printSplitTable(s, 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");
            }
            System.out.println();
        }
    }

    static void printSplitTable(int[][] s, 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(s[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

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: 3
Enter the array of dimensions (space-separated):
10 20 30 40

Matrix dimensions:
A1 : 10 x 20
A2 : 20 x 30
A3 : 30 x 40

Optimal Parenthesis: ((A1A2)A3)

Minimum Cost of Multiplications: 18000

DP Table (Minimum Cost of Multiplications):
0   6000    18000
    0   24000
        0

Split Table:
    1   2
        2
```