Practical No-6

Name-Huzaif Baig

Class -A-4

Roll-40

Subject-DAA

Aim-

Aim: Construction of OBST

Problem Statement: Smart Library Search Optimization

Task 1:

Scenario:

A university digital library system stores frequently accessed books using a binary search

mechanism. The library admin wants to minimize the average search time for book lookups by

arranging the book IDs optimally in a binary search tree.

Each book ID has a probability of being searched successfully and an associated probability for

unsuccessful searches (when a book ID does not exist between two keys).

Your task is to determine the minimum expected cost of searching using an Optimal Binary

Search Tree (OBST).

1)

Code

class Obst {

    public static void main(String[] args) {

        int[] keys = {10, 20, 30, 40};

        int[] freq = {4, 2, 6, 3};

        int n = keys.length;

        int[][] dp = new int[n][n];

        // Base case: one key

        for (int i = 0; i < n; i++) {

            dp[i][i] = freq[i];

        }

        // Chain length from 2 to n

        for (int length = 2; length <= n; length++) {

            for (int i = 0; i <= n - length; i++) {

                int j = i + length - 1;

                dp[i][j] = Integer.MAX\_VALUE;

                // Sum of frequencies from i to j

                int totalFreq = 0;

                for (int k = i; k <= j; k++) {

                    totalFreq += freq[k];

                }

                // Try all keys as root

                for (int r = i; r <= j; r++) {

                    int left = (r > i) ? dp[i][r - 1] : 0;

                    int right = (r < j) ? dp[r + 1][j] : 0;

                    int cost = left + right + totalFreq;

                    if (cost < dp[i][j]) {

                        dp[i][j] = cost;

                    }

                }

            }

        }

        System.out.println("Minimum cost of OBST: " + dp[0][n - 1]);

    }

}

Output



2)



