Ramdeobaba University, Nagpur
Department of Computer Science and Engineering
Session: 2024-2025

Design and Analysis of Algorithms Lab III Semester PRACTICAL NO. 6

Name: Ananya Biyani
Roll number: 02
Section: A4-B1

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).

Input Format

First line: integer n — number of book IDs.

Second line: n integers representing the sorted book IDs (keys).

Third line: n real numbers — probabilities of successful searches (p[i]).

Fourth line: n+1 real numbers — probabilities of unsuccessful searches

(q[i]). Keys: 10 20 30 40

P[i]: 0.1 0.2 0.4 0.3

Q[i]: 0.05 0.1 0.05 0.05 0.1

Output Format

Print the minimum expected cost of the Optimal Binary Search Tree, rounded to 4 decimal places.

CODE:

```
public class OptimalBST {

    public static double optimalBST(double[] p, double[] q, int n)

        { double[][] e = new double[n + 2][n + 2];

        double[][] w = new double[n + 2][n + 2];

        int[][] root = new int[n + 2][n + 2];



        for (int i = 1; i <= n + 1; i++) {

            e[i][i - 1] = q[i - 1];

            w[i][i - 1] = q[i - 1];

        }



        for (int l = 1; l <= n; l++) {

            for (int i = 1; i <= n - l + 1; i++) {

                int j = i + l - 1;

                e[i][j] = Double.MAX_VALUE;

                w[i][j] = w[i][j - 1] + p[j - 1] + q[j];

                for (int r = i; r <= j; r++) {
                    double t = e[i][r - 1] + e[r + 1][j] + w[i][j];

                    if (t < e[i][j]) {
```

```java
                    e[i][j] = t;

                    root[i][j] = r;

                }

            }

        }

    }



    return e[1][n];

}



public static void main(String[] args) {

    int n = 4;

    double[] p = {0.1, 0.2, 0.4, 0.3};

    double[] q = {0.05, 0.1, 0.05, 0.05, 0.1};



    double minCost = optimalBST(p, q, n);

            System.out.printf("Minimum expected search cost: %.4f\n",
minCost);

}

}
```
OUTPUT:

```
Minimum expected search cost: 2.9000
```

Task 2:
https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/