

**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:

```

        e[i][j] = t;

        root[1][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/>

The screenshot shows a successful submission for the "Optimal Binary Search Tree" problem on GeeksforGeeks. The interface includes tabs for Problem, Editor, Submissions, and Connect. The main area displays the following information:

- Output Window:** Shows the status "Compilation Results" and "Y.O.U. W.Bet!".
- Problem Solved Successfully:** Shows "Test Cases Passed: 104 / 104", "Attempts: Correct / Total: 1 / 2", and "Accuracy: 50%".
- Points Scored:** 8 / 8.
- Your Total Score:** 8 +.
- Solve Next:** Options include "Fixing Two nodes of a BST", "Strictly Increasing Array", and "Word Wrap".
- Stay Ahead With:** A link to "GeeksQuiz".
- Code Area:** The code is a Java class named "solution" with a static method "optimalSearchTree". The code implements a dynamic programming solution to find the minimum cost of a BST given frequencies and costs for each node.

```
1. class solution {
2.     int optimalSearchTree(int freq[], int n) {
3.         int dp[][] = new int[n][n];
4.         int sum[] = new int[n];
5.         sum[0] = freq[0];
6.         for (int i = 1; i < n; i++) {
7.             sum[i] = sum[i - 1] + freq[i];
8.         }
9.         for (int l = 0; l < n; l++) {
10.             dp[0][l] = freq[l];
11.         }
12.         for (int l = 1; l < n; l++) {
13.             for (int r = l; r < n; r++) {
14.                 int j = l + r - 1;
15.                 dp[l][r] = Integer.MAX_VALUE;
16.                 int freqSum = sum[r] - (j > 0 ? sum[j - 1] : 0);
17.                 for (int c = l; c < r; c++) {
18.                     int costLeft = (c < l) ? dp[l][c - 1] : 0;
19.                     int costRight = (c < r) ? dp[c + 1][r] : 0;
20.                     int cost = costLeft + costRight + freqSum;
21.                     dp[l][r] = Math.min(dp[l][r], cost);
22.                 }
23.             }
24.         }
25.         return dp[0][n - 1];
26.     }
27. }
28. }
```