DAA PRACTICAL 6

Name: Ishita Premchandani

Roll no. : A3\_B1\_08

**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

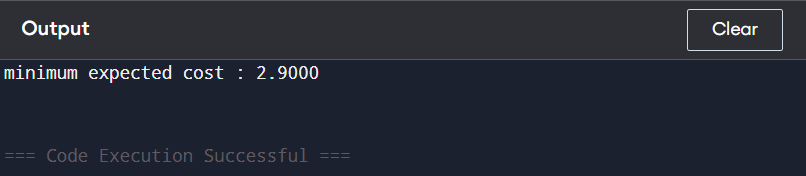
Print the minimum expected cost of the Optimal Binary Search Tree, rounded to 4 decimal places.

Code:

#include <stdio.h>  
#include <float.h>  
  
#define MAX 100  
  
int main() {  
    int n = 4;  
    int i, j, k, d;  
    double p[] = {0.1, 0.2, 0.4, 0.3};  
    double q[] = {0.05, 0.1, 0.05, 0.05, 0.1};  
    double E[MAX][MAX], W[MAX][MAX];  
    int R[MAX][MAX];  
  
    for (i = 0; i <= n; i++) {  
        E[i][i] = q[i];  
        W[i][i] = q[i];  
        R[i][i] = 0;  
    }  
  
    for (d = 1; d <= n; d++) {  
        for (i = 0; i <= n - d; i++) {  
            j = i + d;  
            E[i][j] = DBL\_MAX;  
            W[i][j] = W[i][j - 1] + p[j - 1] + q[j];  
            for (k = i + 1; k <= j; k++) {  
                double cost = E[i][k - 1] + E[k][j] + W[i][j];  
                if (cost < E[i][j]) {  
                    E[i][j] = cost;  
                    R[i][j] = k;  
                }  
            }  
        }  
    }  
  
 printf("minimum expected cost : %.4lf\n", E[0][n]);

return 0;  
}

Output:



Task 2:

<https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/1>

Code:

class Solution {

static int optimalSearchTree(int keys[], int freq[], int n) {

int[][] cost = new int[n][n];

for (int i = 0; i < n; i++)

cost[i][i] = freq[i];

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

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

int j = i + L - 1;

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

int fsum = 0;

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

fsum += freq[k];

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

int c = ((r > i) ? cost[i][r - 1] : 0) +

((r < j) ? cost[r + 1][j] : 0) + fsum;

if (c < cost[i][j])

cost[i][j] = c;

}

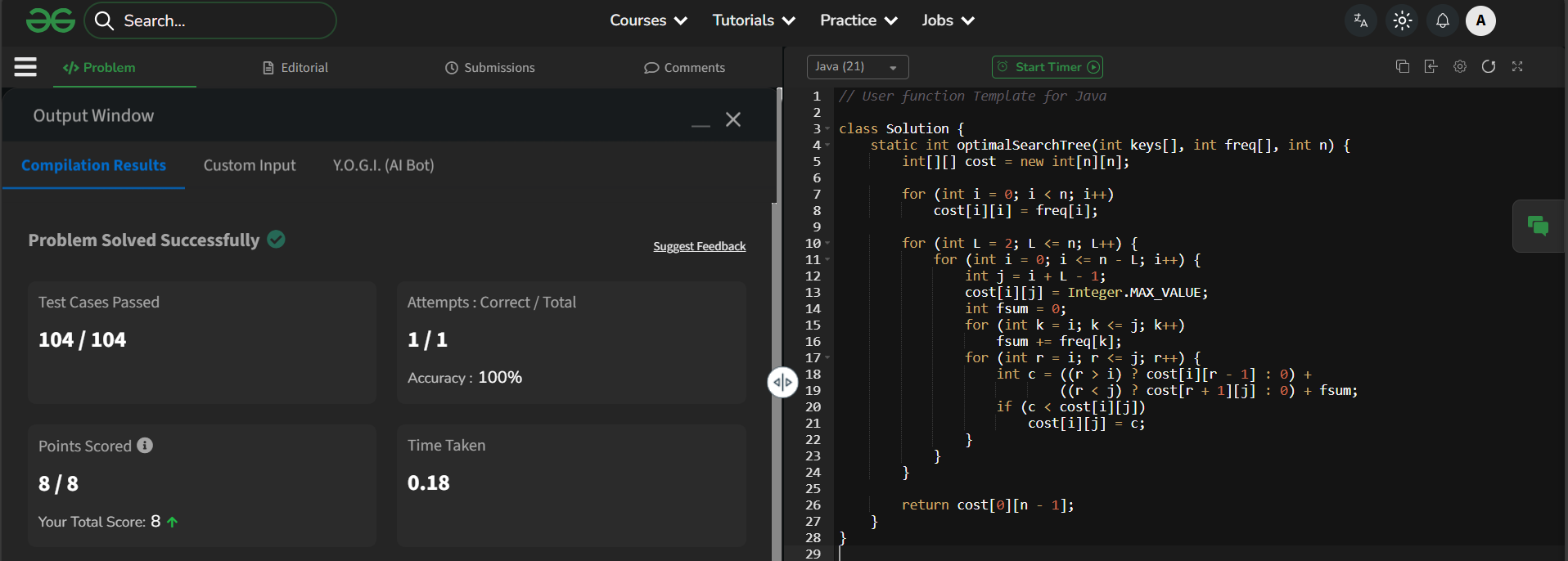
}

}

return cost[0][n - 1];

}

}

Output: