PRACTICAL 6

Name: Chinta Bhumika Reddy

Class: A4

Roll No: 54

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.

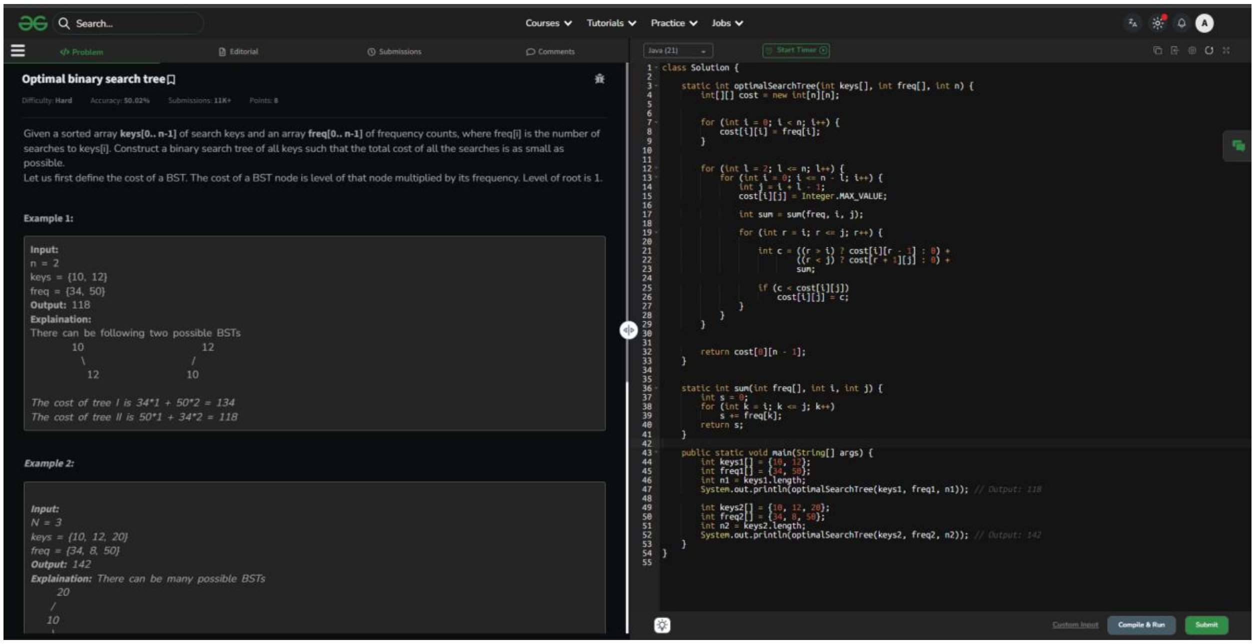
CODE:

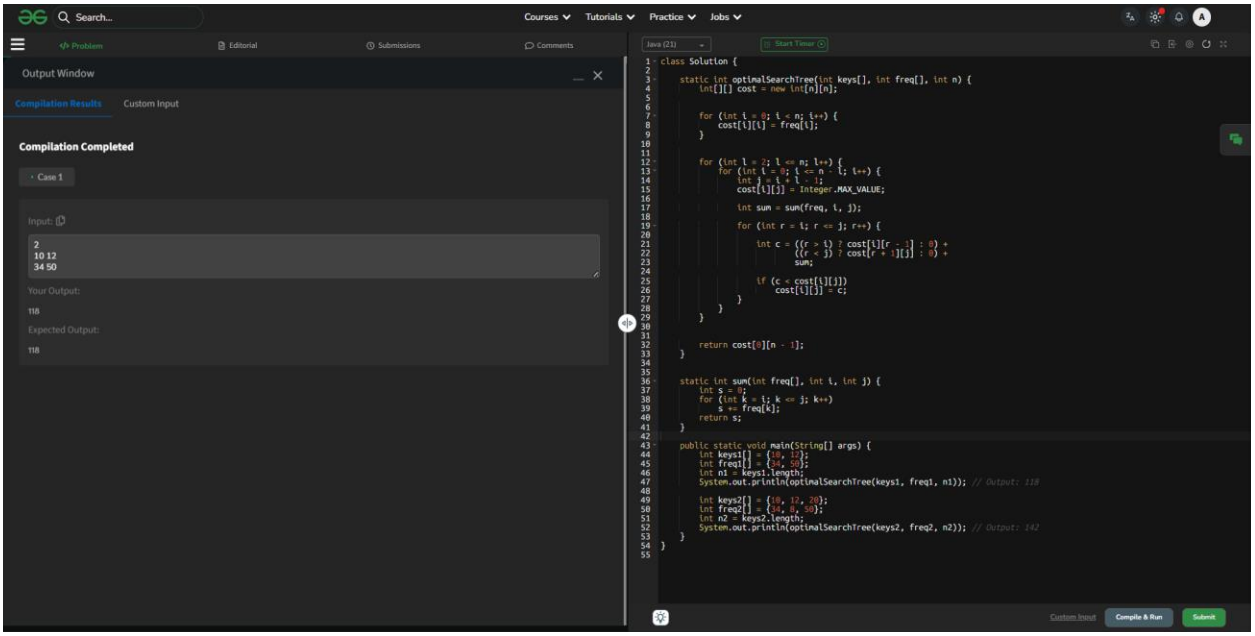
```
#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#include <math.h>


double OptimalBST(int n, double p[], double q[]) {

  double E[n + 1][n + 1];

  double W[n + 1][n + 1];

  int R[n + 1][n + 1];


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

    E[i][i] = q[i];

    W[i][i] = q[i];

    R[i][i] = 0;

  }

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

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

      int j = i + d;

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

      E[i][j] = DBL_MAX

      for (int 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;
                }
            }
        }
    }

    return E[0][n];
}
int main() {
    int n = 4;
     double P[] = {0.0, 0.1, 0.2, 0.4, 0.3}
    double Q[] = {0.05, 0.1, 0.05, 0.05, 0.1};
    double min_cost = OptimalBST(n, P, Q);
    printf("%.4f\n", min_cost);

    return 0;
}
```

```
2.9000
```

Task2:


Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Optimal binary search tree
Difficulty: Hard
Accuracy: 50.02%
Submissions: 11K+
Points: 8
Given a sorted array keys[0.. n-1] of search keys and an array freq[0.. n-1] of frequency counts, where freq[i] is the number of searches to keys[i]. Construct a binary search tree of all keys such that the total cost of all the searches is as small as possible.
Let us first define the cost of a BST. The cost of a BST node is level of that node multiplied by its frequency. Level of root is 1.
Example 1:
Input:
n = 2
keys = {10, 12}
freq = {34, 50}
Output: 118
Explaination:
There can be following two possible BSTs
10 12
\ /
12 10
The cost of tree I is 34*1 + 50*2 = 134
The cost of tree II is 50*1 + 34*2 = 118
Example 2:
Input:
N = 3
keys = {10, 12, 20}
freq = {34, 8, 50}
Output: 142
Explaination: There can be many possible BSTs
20
/
10
Java (21)
Start Timer
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 sum = sum(freq, i, j);
for (int r = i; r <= j; r++) {
int c = ((r > i) ? cost[i][r - 1] : 0) +
((r < j) ? cost[r + 1][j] : 0) +
sum;
if (c < cost[i][j])
cost[i][j] = c;
}
}
}
return cost[0][n - 1];
}
static int sum(int freq[], int i, int j) {
int s = 0;
for (int k = i; k <= j; k++)
s += freq[k];
return s;
}
public static void main(String[] args) {
int keys1[] = {10, 12};
int freq1[] = {34, 50};
int n1 = keys1.length;
System.out.println(optimalSearchTree(keys1, freq1, n1)); // Output: 118
int keys2[] = {10, 12, 20};
int freq2[] = {34, 8, 50};
int n2 = keys2.length;
System.out.println(optimalSearchTree(keys2, freq2, n2)); // Output: 142
}
}
Custom Input
Compile & Run
Submit


Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Compilation Completed
Case 1
Input:
2
10 12
34 50
Your Output:
118
Expected Output:
118
Java (21)
Start Timer
Custom Input
Compile & Run
Submit