Ramdeobaba University, Nagpur

Department of Computer Science and Engineering

Session: 2025-26

DAA LAB **III Semester**

Name: Vardhan Ingole
Section: A4
Batch: B3
Roll No: 43

Practical - 6

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).

Code:

```c
#include <stdio.h>
#include <limits.h>
#define MAX 100

void input(int n, float p[], float q[]) {
    printf("Enter probabilities of successful searches p[1..%d]:\n", n);
    for (int i = 1; i <= n; i++)
        scanf("%f", &p[i]);

    printf("Enter probabilities of unsuccessful searches q[0..%d]:\n", n);
    for (int i = 0; i <= n; i++)
        scanf("%f", &q[i]);
}

void optimalBST(int n, float p[], float q[], float e[][MAX], float
w[][MAX], int root[][MAX]) {
    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] = INT_MAX;
            w[i][j] = w[i][j - 1] + p[j] + q[j];

            for (int r = i; r <= j; r++) {
                float cost = e[i][r - 1] + e[r + 1][j] + w[i][j];
                if (cost < e[i][j]) {
                    e[i][j] = cost;
                    root[i][j] = r;
                }
            }
        }
    }
}

void printTables(int n, float e[][MAX], int root[][MAX]) {
```

```c
    printf("\nCost Table e[i][j]:\n");
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++)
            printf("%8.3f ", e[i][j]);
        printf("\n");
    }

    printf("\nRoot Table root[i][j]:\n");
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++)
            printf("%5d ", root[i][j]);
        printf("\n");
    }
}

void printBST(int root[][MAX], int i, int j, int parent, char child) {
    if (i > j)
        return;

    int r = root[i][j];
    if (parent == -1)
        printf("Root: K%d\n", r);
    else
        printf("K%d is %c child of K%d\n", r, child, parent);

    printBST(root, i, r - 1, r, 'L');
    printBST(root, r + 1, j, r, 'R');
}

int main() {
    int n;
    float p[MAX], q[MAX], e[MAX][MAX], w[MAX][MAX];
    int root[MAX][MAX];

    printf("Enter number of keys: ");
    scanf("%d", &n);

    input(n, p, q);
    optimalBST(n, p, q, e, w, root);
    printTables(n, e, root);
```

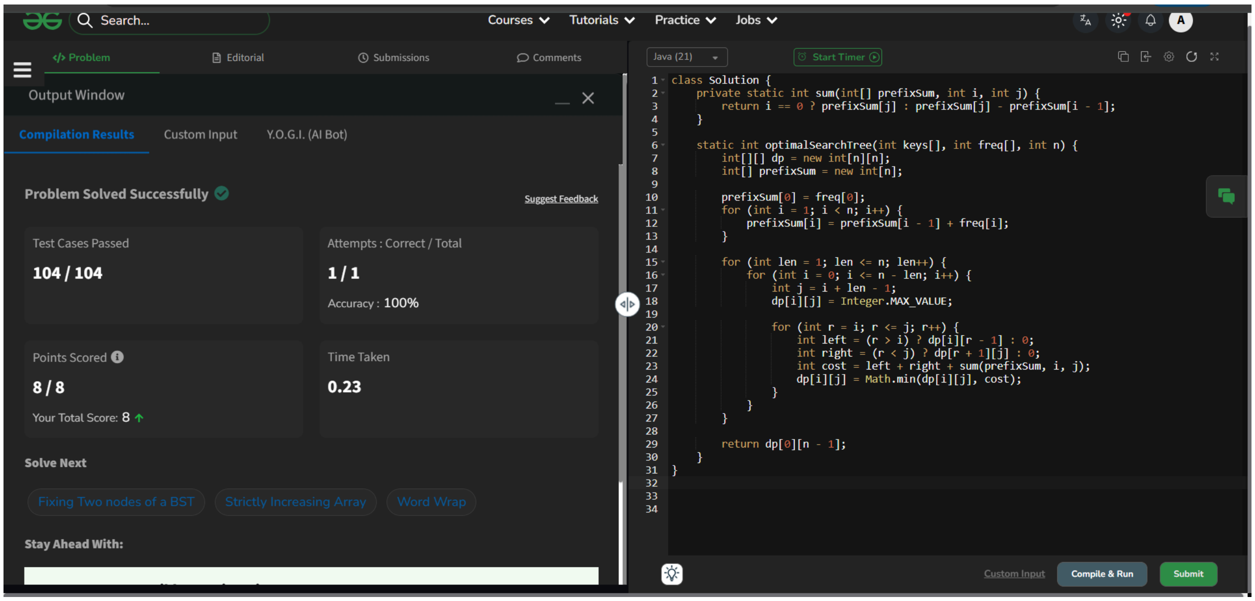
```
    printf("\nMinimum Expected Cost of OBST = %.4f\n", e[1][n]);
    printf("\nStructure of Optimal Binary Search Tree:\n");
    printBST(root, 1, n, -1, ' ');

    return 0;
}
```

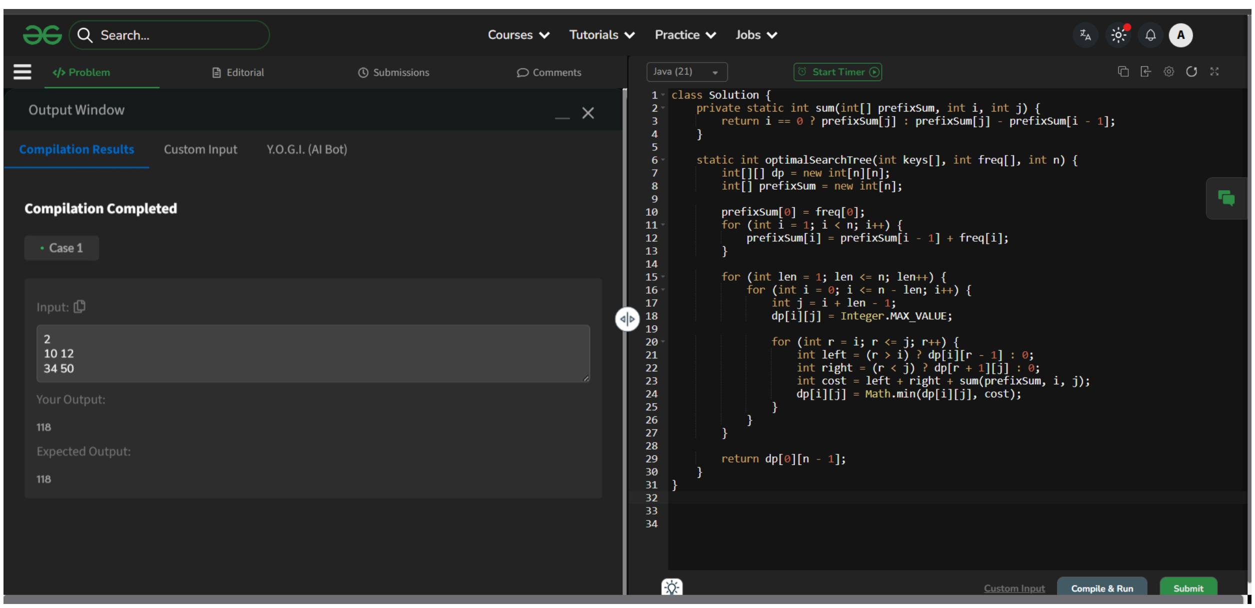
Output:

```
Output                                                        Clear
Enter number of keys: 4
Enter probabilities of successful searches p[1..4]:
0.1
0.2
0.4
0.3
Enter probabilities of unsuccessful searches q[0..4]:
0.05
0.1
0.05
0.05
0.1

Cost Table e[i][j]:
   0.400    0.950    1.950    2.900
   0.500    1.350    2.300
   0.600    1.550
   0.600

Root Table root[i][j]:
    1     2     2     3
    2     3     3
    3     3
    4

Minimum Expected Cost of OBST = 2.9000

Structure of Optimal Binary Search Tree:
Root: K3
K2 is L child of K3
K1 is L child of K2
K4 is R child of K3
```

TASK-2: SUBMISSION ON GEEKS FOR GEEKS

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





My Solution:

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