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).

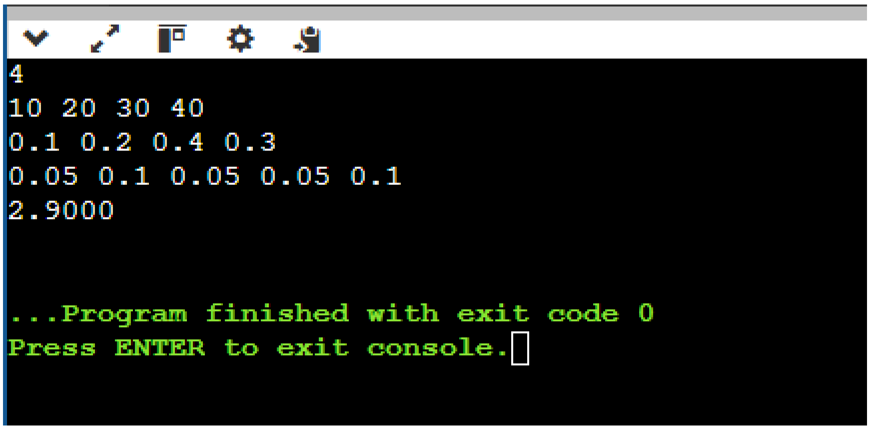
Your task is to determine the minimum expected cost of searching using an Optimal Binary Search Tree (OBST).

Code :

```
#include  <stdio.h>
#include <stdlib.h>
#include <float.h>

#define MAX 100

int main() {
   int n;
   scanf("%d", &n);

   int keys[MAX];
   double p[MAX], q[MAX + 1];
   for (int i = 0; i < n; i++)
      scanf("%d", &keys[i]);
   for (int i = 0; i < n; i++)
```

```
        scanf("%lf", &p[i]);
    for (int i = 0; i <= n; i++)
        scanf("%lf", &q[i]);

    double e[MAX + 1][MAX + 1] = {0};
    double w[MAX + 1][MAX + 1] = {0};
    int root[MAX + 1][MAX + 1] = {0};

    for (int i = 0; i <= n; i++) {
        e[i][i] = q[i];
        w[i][i] = q[i];
    }

    for (int l = 1; l <= n; l++) {
        for (int i = 0; i <= n - l; i++) {
            int j = i + l;
            e[i][j] = DBL_MAX;
            w[i][j] = w[i][j - 1] + p[j - 1] + q[j];
            for (int r = i + 1; r <= j; r++) {
                double t = e[i][r - 1] + e[r][j] + w[i][j];
                if (t < e[i][j]) {
                    e[i][j] = t;
                    root[i][j] = r;
                }
            }
        }
    }

    printf("%.4lf\n", e[0][n]);
    return 0;
}
```

Output :



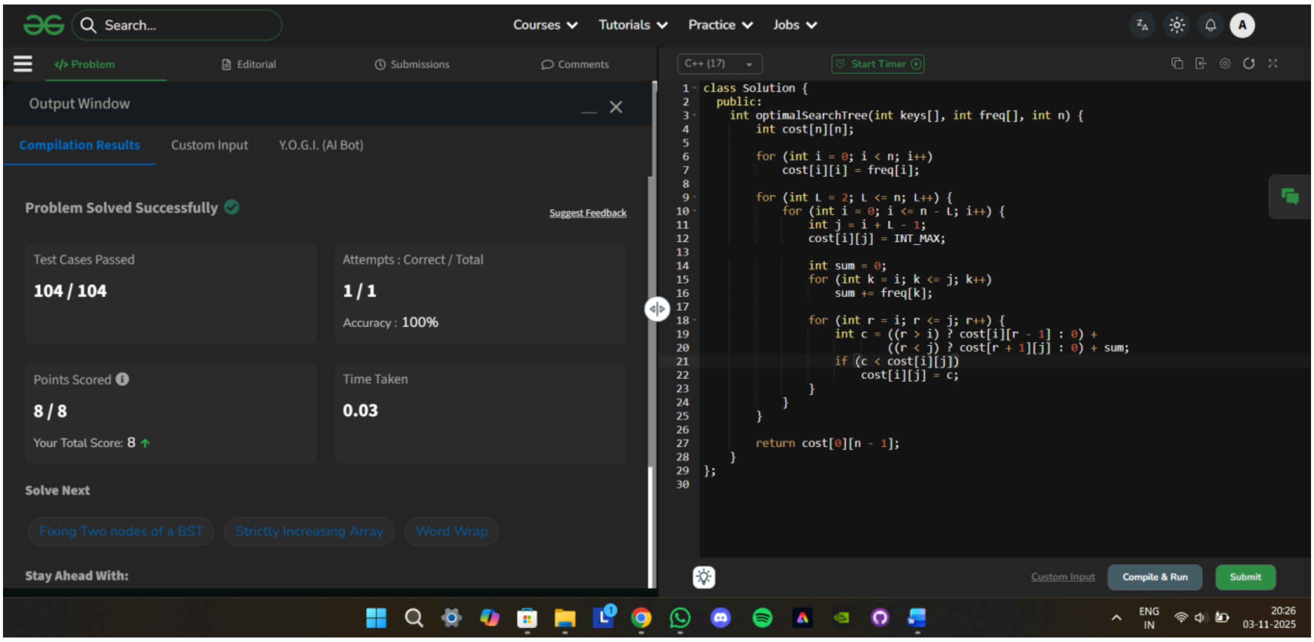
```
4
10 20 30 40
0.1 0.2 0.4 0.3
0.05 0.1 0.05 0.05 0.1
2.9000


...Program finished with exit code 0
Press ENTER to exit console.
```

Task 2:

Code :

```
class Solution {
  public:
    int optimalSearchTree(int keys[], int freq[], int n) {
        int cost[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] = INT_MAX;

                int sum = 0;
                for (int k = i; k <= j; k++)
                    sum += 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) + sum;
                    if (c < cost[i][j])
                        cost[i][j] = c;
                }
            }
        }

        return cost[0][n - 1];
    }
};
```

OUTPUT:



Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
C++ (17)
Start Timer
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
104 / 104
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Points Scored
8 / 8
Your Total Score: 8
Time Taken
0.03
Solve Next
Fixing Two nodes of a BST
Strictly Increasing Array
Word Wrap
Stay Ahead With:
class Solution {
public:
int optimalSearchTree(int keys[], int freq[], int n) {
int cost[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] = INT_MAX;
int sum = 0;
for (int k = i; k <= j; k++)
sum += 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) + sum;
if (c < cost[i][j])
cost[i][j] = c;
}
}
}
return cost[0][n - 1];
}
};
Custom Input
Compile & Run
Submit
ENG
IN
20:26
03-11-2025