Design and Analysis of Algorithms Lab

Name: Arya Bodkhe

Section-A4 Batch: B-1

Roll No.:09

PRACTICAL NO. 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).

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

void OBST(float p[], float q[], int n) {
    int i, j, r, l;
    float e[MAX][MAX], w[MAX][MAX];
    int root[MAX][MAX];
    float t;

    for (i = 1; i <= n + 1; i++) {
        e[i][i - 1] = q[i - 1];
        w[i][i - 1] = q[i - 1];
    }



    for (l = 1; l <= n; l++) {
        for (i = 1; i <= n - l + 1; i++) {
            j = i + l - 1;
            e[i][j] = FLT_MAX;
            w[i][j] = w[i][j - 1] + p[j] + q[j];

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

    printf("\nMinimum expected cost of Optimal BST: %.4f\n", e[1][n]);
```

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

int main() {
    int n, i;
    float p[MAX], q[MAX];
    int keys[MAX];

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

    printf("Enter sorted book IDs:\n");
    for (i = 1; i <= n; i++) {
        scanf("%d", &keys[i]);
    }

    printf("Enter probabilities of successful searches (p[1] to p[%d]):\n", n);
    for (i = 1; i <= n; i++) {
        scanf("%f", &p[i]);
    }

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

    OBST(p, q, n);
    return 0;
}
```

OUTPUT:

```
Enter number of book IDs: 4
Enter sorted book IDs:
10 20 30 40
Enter probabilities of successful searches (p[1] to p[4]):
0.1 0.2 0.4 0.3
Enter probabilities of unsuccessful searches (q[0] to q[4]):
0.05 0.1 0.05 0.05 0.1

Minimum expected cost of Optimal BST: 2.9000

Root Table:
root[1][1] = 1  root[1][2] = 2  root[1][3] = 3  root[1][4] = 3
root[2][2] = 2  root[2][3] = 3  root[2][4] = 3
root[3][3] = 3  root[3][4] = 3
root[4][4] = 4
```

Task 2: https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/1

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
```

CODE:

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

        int[][] e = new int[n + 2][n + 2];
        int[][] w = new int[n + 2][n + 2];
        int[][] root = new int[n + 1][n + 1];



        for (int i = 1; i <= n + 1; i++) {
            e[i][i - 1] = 0;
            w[i][i - 1] = 0;
        }



        for (int l = 1; l <= n; l++) {
            for (int i = 1; i <= n - l + 1; i++) {
                int j = i + l - 1;
                e[i][j] = Integer.MAX_VALUE;
                w[i][j] = w[i][j - 1] + freq[j - 1];

                for (int r = i; r <= j; r++) {
                    int cost = ((r > i) ? e[i][r - 1] : 0) +
                               ((r < j) ? e[r + 1][j] : 0) +
                               w[i][j];
                    if (cost < e[i][j]) {
                        e[i][j] = cost;
```

```
                root[i][j] = r;
              }
            }
          }
        }

        return e[1][n];
    }
}
```

```java
class Solution {
    static int optimalSearchTree(int keys[], int freq[], int n) {

        int[][] e = new int[n + 2][n + 2];
        int[][] w = new int[n + 2][n + 2];
        int[][] root = new int[n + 1][n + 1];


        for (int i = 1; i <= n + 1; i++) {
            e[i][i - 1] = 0;
            w[i][i - 1] = 0;
        }


        for (int l = 1; l <= n; l++) {
            for (int i = 1; i <= n - l + 1; i++) {
                int j = i + l - 1;
                e[i][j] = Integer.MAX_VALUE;
                w[i][j] = w[i][j - 1] + freq[j - 1];

                for (int r = i; r <= j; r++) {
                    int cost = ((r > i) ? e[i][r - 1] : 0) +
                               ((r < j) ? e[r + 1][j] : 0) +
                              w[i][j];
                    if (cost < e[i][j]) {
                        e[i][j] = cost;
                        root[i][j] = r;
                    }
                }
            }
        }

        return e[1][n];
    }

}
```

OUTPUT:

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Compilation Completed

• Case 1

Input:

```
2
10 12
34 50
```

Your Output:

118

Expected Output:

118

Problem Solved Successfully

Suggest Feedback

Test Cases Passed

104 / 104

Attempts : Correct / Total

2 / 2

Accuracy : 100%

Time Taken

0.2