PRACTICAL NO. 6

Name : Anjali Lanjewar
Section : A4_B2
Roll no. : 25

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

main.c

Share

Run

```c
#include <stdio.h>
#include <limits.h>
#include <float.h>

#define MAX 100

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

    int keys[MAX];
    double p[MAX], q[MAX + 1];
    double E[MAX + 1][MAX + 1], W[MAX + 1][MAX + 1];
    int R[MAX + 1][MAX + 1];

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

    printf("Enter %d probabilities of successful searches (p[i]): ", n);
    for (int i = 1; i <= n; i++)
        scanf("%lf", &p[i]);

    printf("Enter %d probabilities of unsuccessful searches (q[i]): ", n + 1);
    for (int i = 0; i <= n; i++)
        scanf("%lf", &q[i]);

    for (int i = 0; i <= n; i++) {
        E[i][i] = q[i];
        W[i][i] = q[i];
        R[i][i] = 0;
    }

```

```c
    for (int d = 1; d <= n; d++) {
        for (int i = 0; i <= n - d; i++) {
            int j = i + d;
            E[i][j] = DBL_MAX;
            W[i][j] = W[i][j - 1] + p[j] + q[j];

            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;
                }
            }
        }
    }

    printf("\nMinimum expected cost of OBST = %.4lf\n", E[0][n]);
    return 0;
}
```

OUTPUT :

Output

Clear

```
Enter number of book IDs: 4
Enter 4 sorted book IDs: 10
20
30
40
Enter 4 probabilities of successful searches (p[i]): 0.1
0.2
0.3
0.4
Enter 5 probabilities of unsuccessful searches (q[i]): 0.05
0.1
0.05
0.05
0.1

Minimum expected cost of OBST = 3.0000


=== Code Execution Successful ===
```

Task 2:

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
```

C++ (17)

Start Timer

```cpp
class Solution{
    private:
    int solve(int i, int j, int *keys, int *freq, vector<vector<int>> &dp)
    {

        if (i == j)
        {
            return freq[i];
        }

        if (i > j)
        {
            return 0;
        }

        if (dp[i][j] != -1)
        {
            return dp[i][j];
        }


        int cur = 0;
        for (int k = i; k <= j; k++)
        {
            cur += freq[k];
        }

        int ans = INT_MAX;
        for (int k = i; k <= j; k++)
        {
                      int left = solve(i, k - 1, keys, freq, dp);
            int right = solve(k + 1, j, keys, freq, dp);
            ans = min(ans, left + right + cur);
        }
        return dp[i][j] = ans;
    }

```

Output Window

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%