

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

```
1  #include <stdio.h>
2  #include <limits.h>
3  #include <float.h>
4
5  #define MAX 100
6
7  int main() {
8      int n;
9      printf("Enter number of book IDs: ");
10     scanf("%d", &n);
11
12     int keys[MAX];
13     double p[MAX], q[MAX + 1];
14     double E[MAX + 1][MAX + 1], W[MAX + 1][MAX + 1];
15     int R[MAX + 1][MAX + 1];
16
17     printf("Enter %d sorted book IDs: ", n);
18     for (int i = 1; i <= n; i++)
19         scanf("%d", &keys[i]);
20
21     printf("Enter %d probabilities of successful searches (p[i]): ", n);
22     for (int i = 1; i <= n; i++)
23         scanf("%lf", &p[i]);
24
25     printf("Enter %d probabilities of unsuccessful searches (q[i]): ", n + 1);
26     for (int i = 0; i <= n; i++)
27         scanf("%lf", &q[i]);
28
29     for (int i = 0; i <= n; i++) {
30         E[i][i] = q[i];
31         W[i][i] = q[i];
32         R[i][i] = 0;
33     }
34 }
```

```

35 ▾   for (int d = 1; d <= n; d++) {
36 ▾       for (int i = 0; i <= n - d; i++) {
37           int j = i + d;
38           E[i][j] = DBL_MAX;
39           W[i][j] = W[i][j - 1] + p[j] + q[j];
40
41 ▾       for (int k = i + 1; k <= j; k++) {
42           double cost = E[i][k - 1] + E[k][j] + W[i][j];
43 ▾           if (cost < E[i][j]) {
44               E[i][j] = cost;
45               R[i][j] = k;
46           }
47       }
48   }
49 }
50
51 printf("\nMinimum expected cost of OBST = %.4lf\n", E[0][n]);
52 return 0;
53 }
54

```

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:

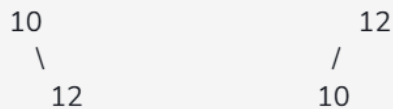
n = 2

keys = {10, 12}

freq = {34, 50}

Output: 118**Explanation:**

There can be following two possible BSTs



*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



```
1 class Solution{
2     private:
3     int solve(int i, int j, int *keys, int *freq, vector<vector<int>> &dp)
4     {
5
6         if (i == j)
7         {
8             return freq[i];
9         }
10
11        if (i > j)
12        {
13            return 0;
14        }
15
16        if (dp[i][j] != -1)
17        {
18            return dp[i][j];
19        }
20
21
22        int cur = 0;
23        for (int k = i; k <= j; k++)
24        {
25            cur += freq[k];
26        }
27
28        int ans = INT_MAX;
29        for (int k = i; k <= j; k++)
30        {
31            int left = solve(i, k - 1, keys, freq, dp);
32            int right = solve(k + 1, j, keys, freq, dp);
33            ans = min(ans, left + right + cur);
34        }
35        return dp[i][j] = ans;
36    }
37}
```

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%