

PRACTICAL 6

Name: Isha Raut

Section: A4

Batch: B2

Roll no: 23

Aim: Construction of OBST

Problem Statement: Smart Library Search Optimization.

Task 1:

Code:

```
#include <stdio.h>
#define MAX 20

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

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

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

    printf("Enter successful search probabilities p[i]: ");
```

```

for (int i = 1; i <= n; i++)
{
    scanf("%lf", &p[i]);
}

printf("Enter unsuccessful search probabilities q[i]: ");
for (int i = 0; i <= n; i++)
{
    scanf("%lf", &q[i]);
}

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

for (int length = 1; length <= n; length++)
{
    for (int i = 1; i <= n - length + 1; i++)
    {
        int j = i + length - 1;
        e[i][j] = 9999999;
        w[i][j] = w[i][j - 1] + p[j] + q[j];

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

```

```

    }

    printf("\nMinimum expected cost of OBST: %.4lf\n", e[1][n]);

    return 0;
}

```

Output:

```

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

Minimum expected cost of OBST: 2.9000

```

Task 2:

The screenshot shows the GeeksforGeeks website interface for the problem 'Optimal Binary Search Tree'. The left sidebar displays the 'Output Window' with 'Compilation Results' and a 'Problem Solved Successfully' message. It also shows statistics: 104/104 Test Cases Passed, 1/1 Attempts: Correct / Total, 100% Accuracy, 8/8 Points Scored, and 0.25 Time Taken. The main area shows the Java code for the solution, which implements a dynamic programming approach to find the minimum expected cost of an Optimal Binary Search Tree (OBST) given sorted keys and their search probabilities.

Problem Solved Successfully ✓

Test Cases Passed: **104 / 104**

Attempts: Correct / Total: **1 / 1**

Accuracy: 100%

Points Scored: **8 / 8**

Your Total Score: **8** ↑

Time Taken: **0.25**

Solve Next

Five Two nodes of a BST | Strictly Increasing Array | Word Wrap

Java (21) [Start Timer]

```

1 // User function Template for Java
2 class Solution {
3     static int optimalSearchTree(int keys[], int freq[], int n) {
4         int[][] cost = new int[n][n];
5
6         // Base case: cost when there is only one key
7         for (int i = 0; i < n; i++) {
8             cost[i][i] = freq[i];
9         }
10
11         // Chain length from 2 to n
12         for (int L = 2; L <= n; L++) {
13             for (int i = 0; i <= n - L; i++) {
14                 int j = i + L - 1;
15                 cost[i][j] = Integer.MAX_VALUE;
16
17                 // sum of frequencies from i to j
18                 int sumFreq = 0;
19                 for (int k = i; k <= j; k++) {
20                     sumFreq += freq[k];
21                 }
22
23                 // Try making each key in interval [i..j] as root
24                 for (int r = i; r <= j; r++) {
25                     int leftCost = (r > i) ? cost[i][r - 1] : 0;
26                     int rightCost = (r < j) ? cost[r + 1][j] : 0;
27                     int totalCost = leftCost + rightCost + sumFreq;
28
29                     if (totalCost < cost[i][j]) {
30                         cost[i][j] = totalCost;
31                     }
32                 }
33             }
34         }
35         return cost[0][n - 1];
36     }
37 }

```

Custom Input [Compile & Run] [Submit]

Link:

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