

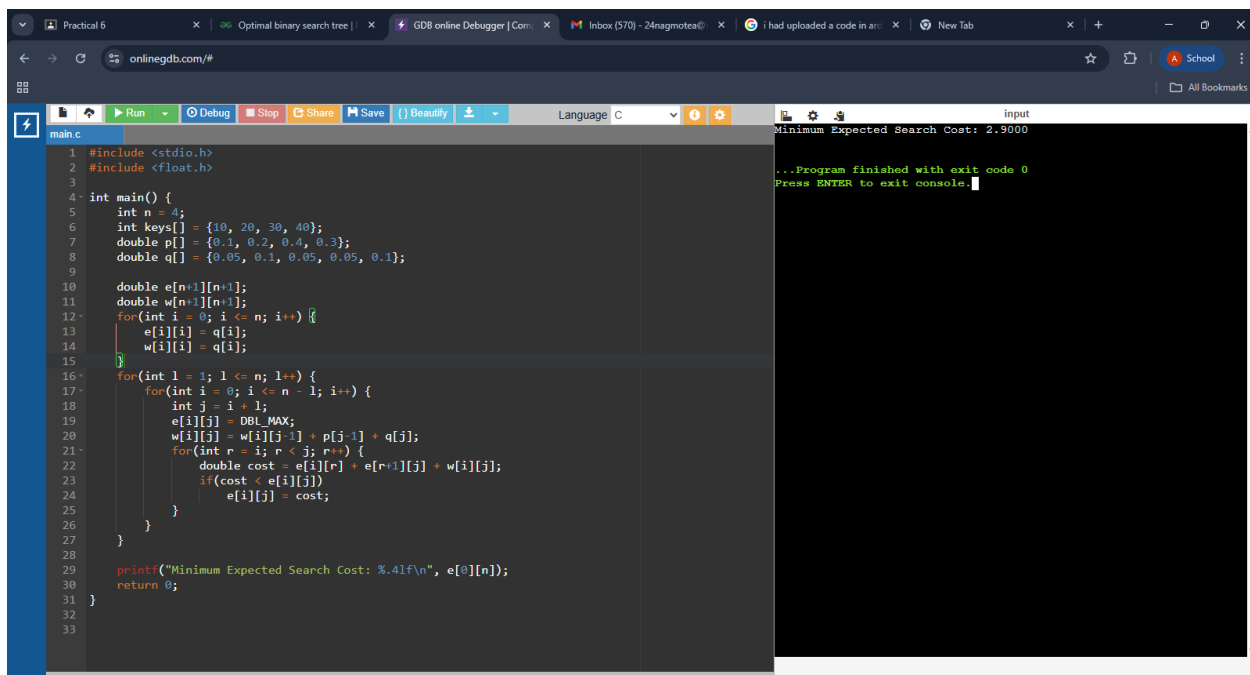
DAA Lab Pract6

Name: Anshika Tiwari

Roll No : A7_B1_14

Aim: Construction of OBST

Code & Output:



The screenshot shows a web browser window with the onlinegdb.com website. The code editor displays a C program for constructing an Optimal Binary Search Tree (OBST). The code includes headers for stdio.h and float.h, defines an array of keys and probabilities, and uses dynamic programming to calculate the minimum expected search cost. The output window shows the result: 'Minimum Expected Search Cost: 2.9000' and a message indicating the program finished with exit code 0.

```
1 #include <stdio.h>
2 #include <float.h>
3
4 int main() {
5     int n = 4;
6     int keys[] = {10, 20, 30, 40};
7     double p[] = {0.1, 0.2, 0.4, 0.3};
8     double q[] = {0.05, 0.1, 0.05, 0.05, 0.1};
9
10    double e[n+1][n+1];
11    double w[n+1][n+1];
12    for(int i = 0; i <= n; i++) {
13        e[i][i] = q[i];
14        w[i][i] = q[i];
15    }
16    for(int l = 1; l <= n; l++) {
17        for(int i = 0; i <= n - l; i++) {
18            int j = i + l;
19            e[i][j] = DBL_MAX;
20            w[i][j] = w[i][j-1] + p[j-1] + q[j];
21            for(int r = i; r < j; r++) {
22                double cost = e[i][r] + e[r+1][j] + w[i][j];
23                if(cost < e[i][j])
24                    e[i][j] = cost;
25            }
26        }
27    }
28
29    printf("Minimum Expected Search Cost: %.4f\n", e[0][n]);
30    return 0;
31 }
32
33
```

Minimum Expected Search Cost: 2.9000

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

geeks of geeks submission:

The screenshot shows a web browser with the URL `geeksforgeeks.org/problems/optimal-binary-search-tree2214/1`. The page displays the solution for the 'Optimal binary search tree' problem. The left sidebar shows the 'Output Window' with 'Compilation Results' and 'Custom Input' tabs. The 'Problem Solved Successfully' message is prominent, along with statistics: Test Cases Passed (104 / 104), Points Scored (8 / 8), Attempts: Correct / Total (1 / 4), Accuracy (25%), and Time Taken (0.03). The 'Solve Next' section lists 'Fixing Two nodes of a BST', 'Strictly Increasing Array', and 'Word Wrap'. The 'Stay Ahead With:' section is also visible. The main content area shows the C++ code for the solution, which includes a `sumFreq` function and an `optimalSearchTree` function.

```
1 class Solution {
2 public:
3
4 int sumFreq(int freq[], int i, int j) {
5     int sum = 0;
6     for (int k = i; k <= j; k++)
7         sum += freq[k];
8     return sum;
9 }
10
11 int optimalSearchTree(int keys[], int freq[], int n) {
12     int cost[n][n];
13
14     for (int i = 0; i < n; i++)
15         cost[i][i] = freq[i];
16
17     for (int L = 2; L <= n; L++) {
18         for (int i = 0; i <= n - L; i++) {
19             int j = i + L - 1;
20             cost[i][j] = INT_MAX;
21             int fsum = sumFreq(freq, i, j);
22
23             for (int r = i; r <= j; r++) {
24                 int c = ((r > i) ? cost[i][r - 1] : 0) +
25                     ((r < j) ? cost[r + 1][j] : 0) + fsum;
26                 if (c < cost[i][j])
27                     cost[i][j] = c;
28             }
29         }
30     }
31
32     return cost[0][n - 1];
33 }
34 }
35 }
```

Link for my git hub repo: <https://github.com/24tiwaria2-code/DAA->