PRACTICAL NO. – 06

NAME : RANVEER M. BHORTEKAR

SECTION/BATCH : A4/B1

ROLL NO. : 06

SUBJECT : DAA

GFG PROFILE ID : Ranveer Mahesh Bhortekar - Ramdeobaba University | GeeksforGeeks Profile

AIM : Construction of OBST

Problem Statement: Smart Library Search Optimization

TASK : 01

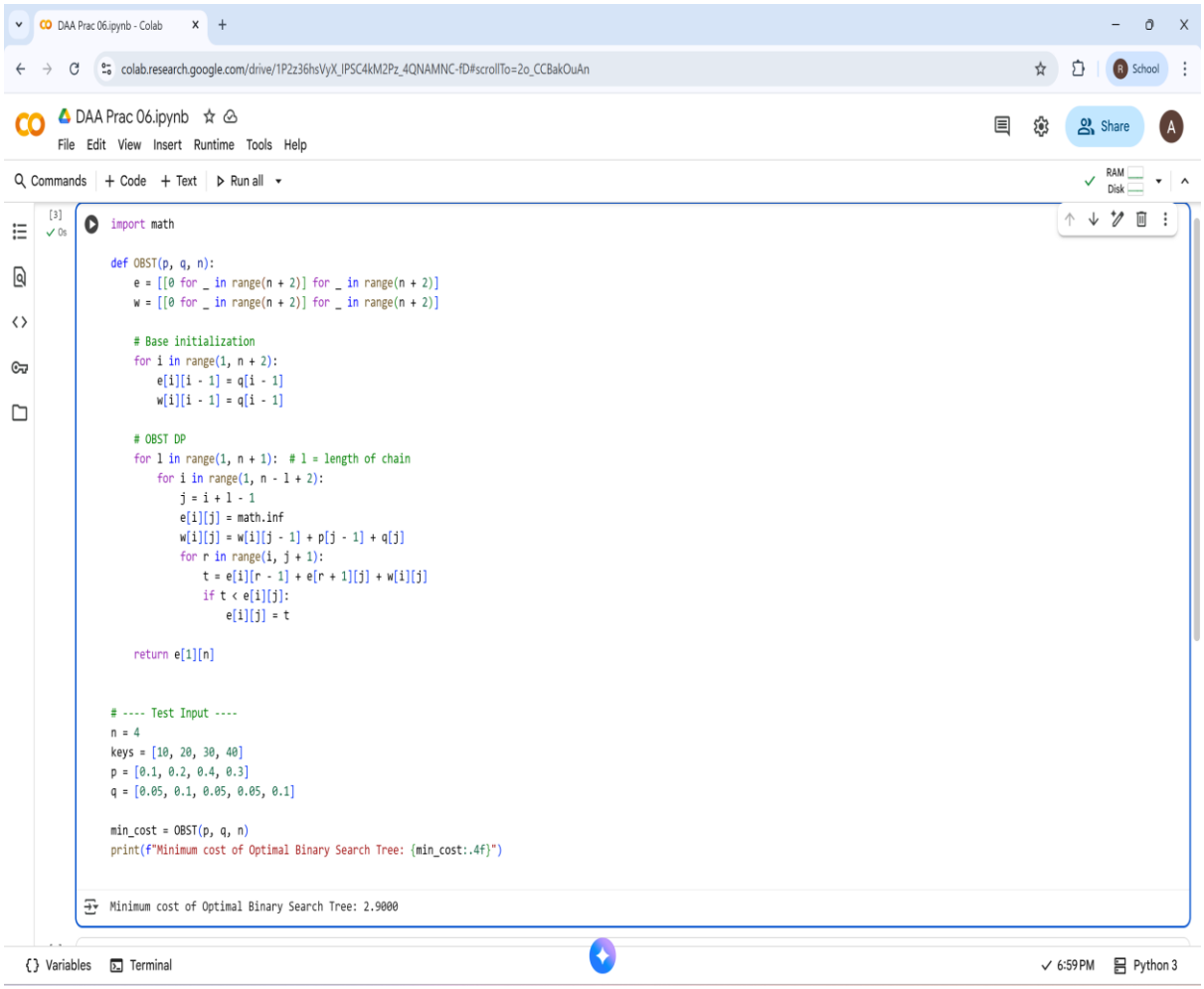
Scenerio : A university digital library system stores frequently accessed books using a binary search mechanism. The library admin wants to minimize the average search time for book lookups by arranging the book IDs optimally in a binary search tree. Each book ID has a probability of being searched successfully 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 : IN TEXT FORMAT

```
import math


def OBST(p, q, n):
    e = [[0 for _ in range(n + 2)] for _ in range(n + 2)]
    w = [[0 for _ in range(n + 2)] for _ in range(n + 2)]

    # Base initialization
    for i in range(1, n + 2):
```

```
        e[i][i - 1] = q[i - 1]
        w[i][i - 1] = q[i - 1]

    # OBST DP
    for l in range(1, n + 1):  # l = length of chain
        for i in range(1, n - l + 2):
            j = i + l - 1
            e[i][j] = math.inf
            w[i][j] = w[i][j - 1] + p[j - 1] + q[j]
            for r in range(i, j + 1):
                t = e[i][r - 1] + e[r + 1][j] + w[i][j]
                if t < e[i][j]:
                    e[i][j] = t

    return e[1][n]

# ---- Test Input ----
n = 4
keys = [10, 20, 30, 40]
p = [0.1, 0.2, 0.4, 0.3]
q = [0.05, 0.1, 0.05, 0.05, 0.1]

min_cost = OBST(p, q, n)
print(f"Minimum cost of Optimal Binary Search Tree: {min_cost:.4f}")
```

CODE AND OUTPUT SCREENSHOT :



TASK : 02

GFG QUESTION ID : https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/1

QUESTION SCREENSHOT

Optimal binary search tree | Practi

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

Search...

Courses Tutorials

Problem Editorial Submissions Comments

Optimal binary search tree

Difficulty: **Hard** Accuracy: **50.02%** Submissions: **11K+** Points: **8**

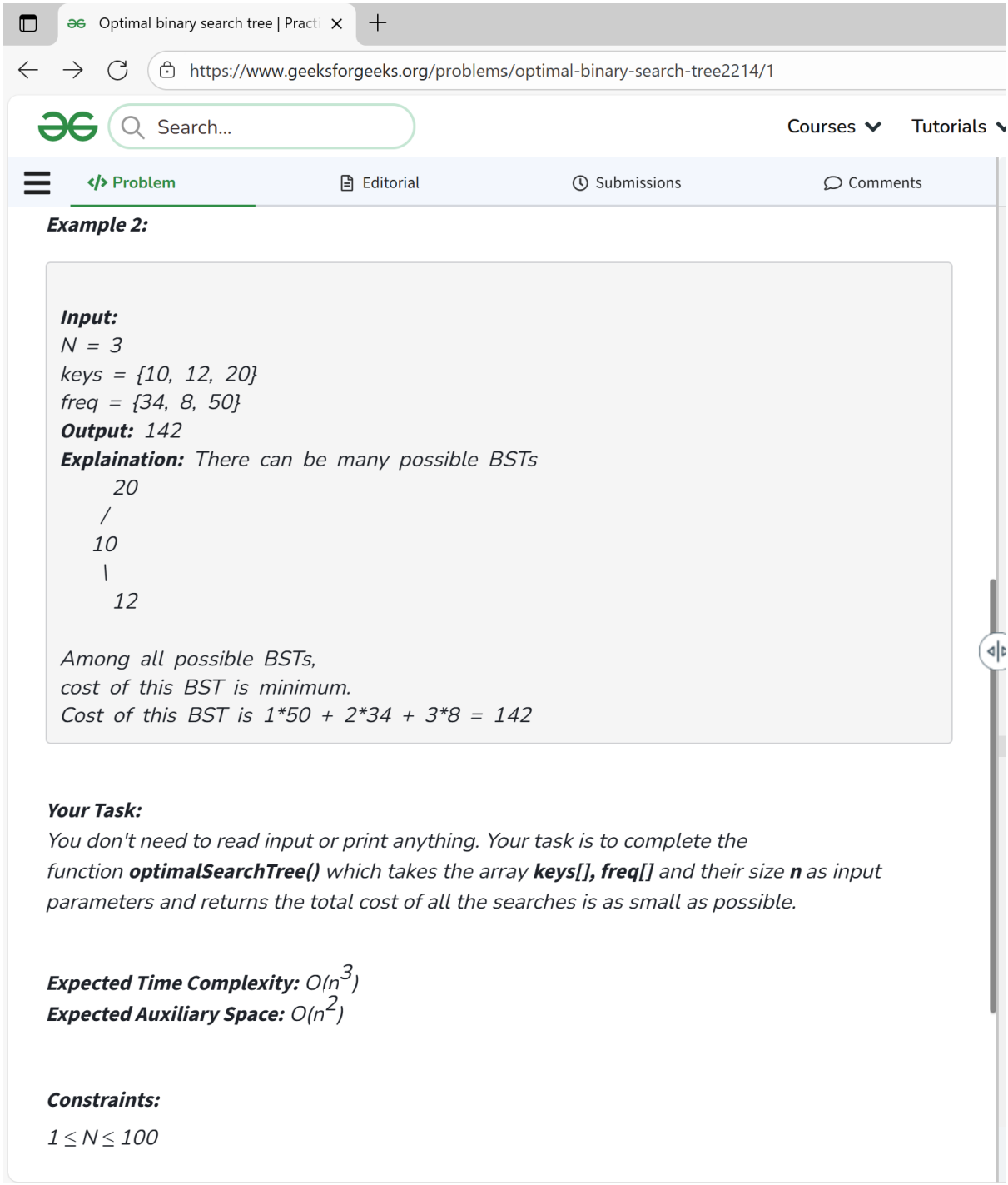
Given a sorted array **keys[0.. n-1]** of search keys and an array **freq[0.. n-1]** of frequency counts, where freq[i] is the number of searches to keys[i]. Construct a binary search tree of all keys such that the total cost of all the searches is as small as possible.
Let us first define the cost of 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*

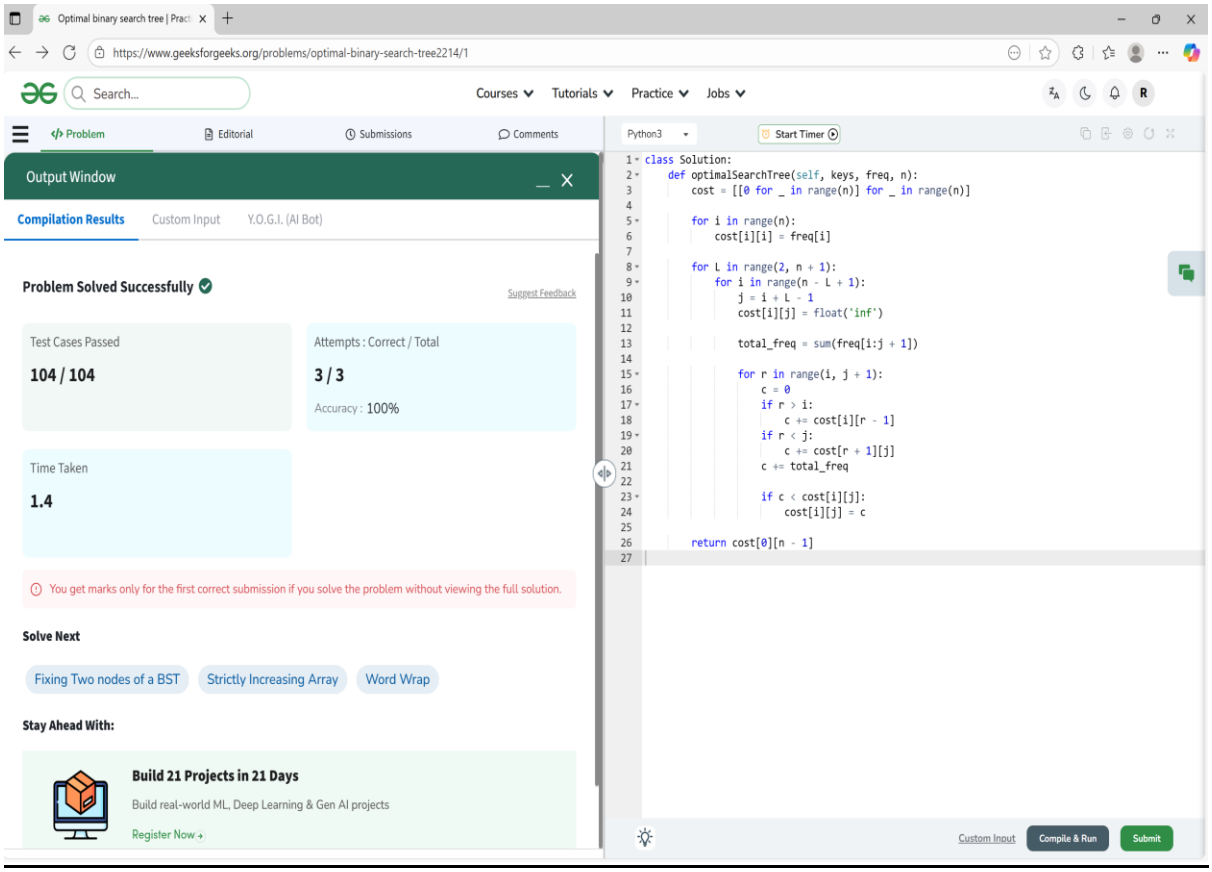


CODE : IN TEXT FORMAT

```
class Solution:

    def optimalSearchTree(self, keys, freq, n):

        cost = [[0 for _ in range(n)] for _ in range(n)]


        for i in range(n):
```

```
        cost[i][i] = freq[i]

    for L in range(2, n + 1):
        for i in range(n - L + 1):
            j = i + L - 1
            cost[i][j] = float('inf')

            total_freq = sum(freq[i:j + 1])

            for r in range(i, j + 1):
                c = 0
                if r > i:
                    c += cost[i][r - 1]
                if r < j:
                    c += cost[r + 1][j]
                c += total_freq

                if c < cost[i][j]:
                    cost[i][j] = c

    return cost[0][n - 1]
```

SUBMISSION SCREENSHOT


Optimal binary search tree | Prac
https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/1
Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
104 / 104
Attempts : Correct / Total
3 / 3
Accuracy : 100%
Time Taken
1.4
You get marks only for the first correct submission if you solve the problem without viewing the full solution.
Solve Next
Fixing Two nodes of a BST
Strictly Increasing Array
Word Wrap
Stay Ahead With:
Build 21 Projects in 21 Days
Build real-world ML, Deep Learning & Gen AI projects
Register Now
Python3
Start Timer
class Solution:
def optimalSearchTree(self, keys, freq, n):
cost = [[0 for _ in range(n)] for _ in range(n)]
for i in range(n):
cost[i][i] = freq[i]
for L in range(2, n + 1):
for i in range(n - L + 1):
j = i + L - 1
cost[i][j] = float('inf')
total_freq = sum(freq[i:j + 1])
for r in range(i, j + 1):
c = 0
if r > i:
c += cost[i][r - 1]
if r < j:
c += cost[r + 1][j]
c += total_freq
if c < cost[i][j]:
cost[i][j] = c
return cost[0][n - 1]
Custom Input
Compile & Run
Submit