

DAA Practical 6
Name : Prashant Raghuwanshi
Roll No : A4 B2 34

Aim : Construction of OBST

Problem Statement : Smart Library Search Optimization

Task 1:

Code :

```
def optimal_bst(n, keys, p, q):
    e = [[0.0]*(n+2) for _ in range(n+2)]
    w = [[0.0]*(n+2) for _ in range(n+2)]

    for i in range(1, n+2):
        e[i][i-1] = q[i-1]
        w[i][i-1] = q[i-1]

    for length in range(1, n+1):
        for i in range(1, n-length+2):
            j = i + length - 1
            e[i][j] = float('inf')
            w[i][j] = w[i][j-1] + p[j-1] + q[j]
            for r in range(i, j+1):
                cost = e[i][r-1] + e[r+1][j] + w[i][j]
                if cost < e[i][j]:
                    e[i][j] = cost

    return e[1][n]

n = int(input())
keys = list(map(int, input().split()))
p = list(map(float, input().split()))
q = list(map(float, input().split()))

min_cost = optimal_bst(n, keys, p, q)
print(f"{min_cost:.4f}")
```

DAA Practical 6
Name : Prashant Raghuwanshi
Roll No : A4 B2 34

Output :

```
4
10 20 30 40
0.1 0.2 0.4 0.3
0.05 0.1 0.05 0.05 0.1
2.9000

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

Task 2:

The screenshot displays a web IDE interface for the 'Optimal Binary Search Tree' problem. The left sidebar shows the 'Output Window' with 'Compilation Results' indicating a successful submission. The main area displays the problem details and the user's solution code.

Problem Solved Successfully ✓

Test Cases Passed: **104 / 104**

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

Accuracy: 100%

Points Scored: **8 / 8**

Time Taken: **1.32**

Your Total Score: 8 ↑

Solve Next

- Fixing Two nodes of a BST
- Strictly Increasing Array
- Word Wrap

Code Snippet:

```
1 #User function Template for python3
2
3 class Solution:
4     def optimalSearchTree(self, keys, freq, n):
5         # code here
6         dp = [[0 for _ in range(n)] for _ in range(n)]
7
8         sumFreq = [[0 for _ in range(n)] for _ in range(n)]
9
10        for i in range(n):
11            sumFreq[i][i] = freq[i]
12            for j in range(i+1, n):
13                sumFreq[i][j] = sumFreq[i][j-1] + freq[j]
14
15        for length in range(1, n+1):
16            for i in range(n - length + 1):
17                j = i + length - 1
18                dp[i][j] = float('inf')
19
20            for r in range(i, j+1):
21                left = dp[i][r-1] if r > i else 0
22                right = dp[r+1][j] if r < j else 0
23                cost = left + right + sumFreq[i][j]
24                if cost < dp[i][j]:
25                    dp[i][j] = cost
26
27        return dp[0][n-1]
```