PRACTICAL NO. 6

Name: Sujal Singh

Sec: A4_B4_52

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
Search Tree (OBST).
Input Format
First line: integer n — number of book IDs.
Second line: n integers representing the sorted book IDs (keys).
Third line: n real numbers — probabilities of successful searches (p[i]).
Fourth line: n+1 real numbers — probabilities of unsuccessful searches (q[i]).
Keys: 10 20 30 40
P[i]: 0.1 0.2 0.4 0.3
Q[i]: 0.05 0.1 0.05 0.05 0.1

Output Format
Print the minimum expected cost of the Optimal Binary Search Tree, rounded to 4 decimal places.
Task 2:
https://www.geeksforgeeks.org/problems/optimal-binary-search-tree2214/1

Task1:

Code:

```
import math

def optimal_bst_cost_fixed(keys, p, q):
    n = len(keys)

    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):
        if i - 1 < len(q):
            W[i][i-1] = q[i-1]

    W[0][0] = q[0]

    for l in range(1, n + 1):
        for i in range(1, n - l + 2):
            j = i + l - 1

            W[i][j] = W[i][j-1] + p[j-1] + q[j]

            E[i][j] = float('inf')

            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 round(E[1][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 = optimal_bst_cost_fixed(keys, p, q)

print(min_cost)
```

Output:

```
The minimum cost is 2.55
```

Task2:

Code:
```
class Solution:
    def optimalSearchTree(self, keys, freq, n):
        Cost = [[0] * n for _ in range(n)]
        SumFreq = [[0] * n for _ in range(n)]

        for i in range(n):
            SumFreq[i][i] = freq[i]
            for j in range(i + 1, n):
                SumFreq[i][j] = SumFreq[i][j-1] + freq[j]

        for l in range(1, n + 1):
            for i in range(n - l + 1):
                j = i + l - 1

                if l == 1:
                    Cost[i][i] = freq[i]
                    continue

                Cost[i][j] = float('inf')

                total_freq = SumFreq[i][j]

                for r in range(i, j + 1):
                    left_cost = Cost[i][r-1] if r > i else 0
                    right_cost = Cost[r+1][j] if r < j else 0
```

```
            current_cost = left_cost + right_cost + total_freq

            if current_cost < Cost[i][j]:
                Cost[i][j] = current_cost

    return Cost[0][n-1]
```
Output:

Problem Solved Successfully

Suggest Feedback

Test Cases Passed	Attempts : Correct / Total
104 / 104	**1 / 1**
	Accuracy : 100%

Points Scored	Time Taken
8 / 8	**0.8**
Your Total Score: 15	