Name : Abhishek Jagam

Class : A4-B2

Roll no. : 21

Practical 6

**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 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.

**Code:-**

import math

def Optimal\_Binary\_Search\_Tree(keys, prob, n):

e = []

w = []

for i in range(n+2):

e.append([0]\*(n+2))

w.append([0]\*(n+2))

for i in range(1, n+2):

e[i][i-1] = prob[i-1]

w[i][i-1] = prob[i-1]

for l in range(1, n+1):

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

j = i + l - 1

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

w[i][j] = w[i][j-1] + keys[j-1] + prob[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 round(e[1][n])

n = int(input('Enter the size of the successful array keys:-'))

m = int(input('Enter the size of the unsuccessful array keys:-'))

p = []

q = []

print('Enter the elements for successful keys:-')

for i in range(n):

x = float(input())

p.append(x)

print('Enter the elements for unsuccessful search between keys:-')

for i in range(m):

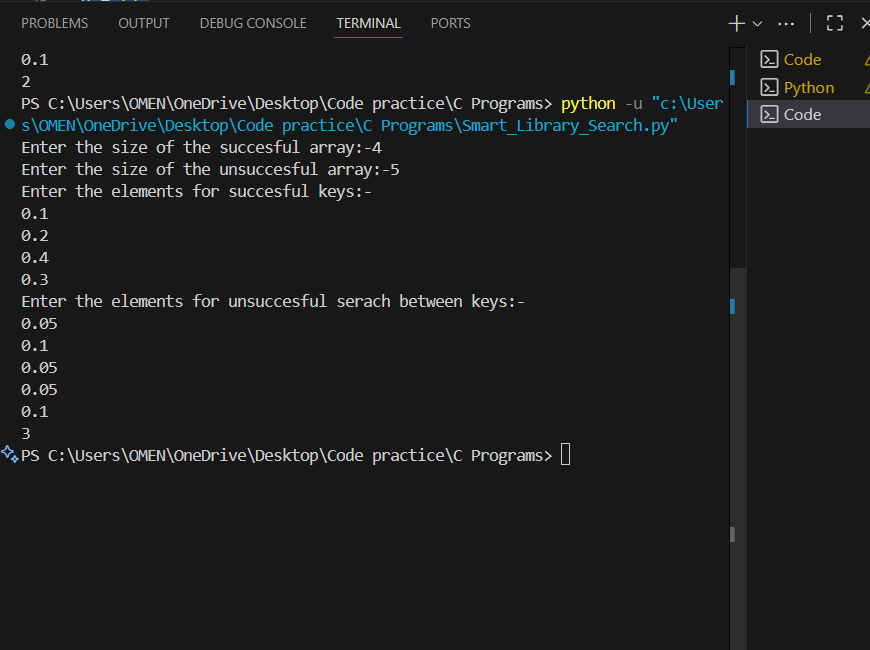
y = float(input())

q.append(y)

length = len(p)

print(Optimal\_Binary\_Search\_Tree(p, q, length)**)**

**Output:-**

****

**Task2:**

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

**Output:**

