DAA PRACTICAL 6

NAME: DIVYA SONTAKKE

SECTION: A4\_B1 ROLL NO.: 11

**Problem Statement: Smart Library Search Optimization**

**Task 1:** 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 decimalplaces.

**CODE**

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#define MAX 100

double e[MAX][MAX];

double w[MAX][MAX];

int root[MAX][MAX];

double optimalBST(int n, double p[], double q[]) {

for (int i = 0; i <= n; i++) {

e[i][i] = q[i];

w[i][i] = q[i];

}

for (int l = 1; l <= n; l++) {

for (int i = 0; i <= n - l; i++) {

int j = i + l;

e[i][j] = DBL\_MAX;

w[i][j] = w[i][j - 1] + p[j - 1] + q[j];

for (int r = i + 1; r <= j; r++) {

double t = e[i][r - 1] + e[r][j] + w[i][j];

if (t < e[i][j]) {

e[i][j] = t;

root[i][j] = r;

}

}

}

}

return e[0][n];

}

int main() {

int n;

printf("Enter the number of book IDs: ");

scanf("%d", &n);

int keys[n];

printf("Enter %d sorted book IDs:", n);

for (int i = 0; i < n; i++) {

scanf("%d", &keys[i]);

}

double p[n], q[n + 1];

printf("Enter probabilities of successful searches (p[i]):", n);

for (int i = 0; i < n; i++) {

scanf("%lf", &p[i]);

}

printf("Enter probabilities of unsuccessful searches (q[i]):", n + 1);

for (int i = 0; i <= n; i++) {

scanf("%lf", &q[i]);

}

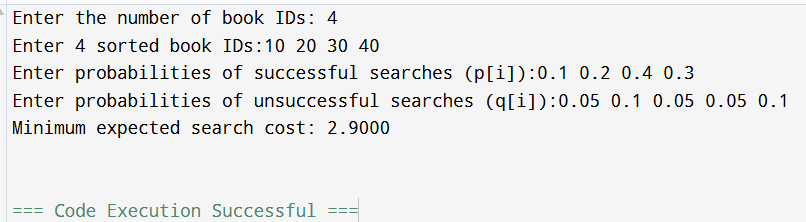
double cost = optimalBST(n, p, q);

printf("Minimum expected search cost: %.4lf\n", cost);

return 0;

}

**OUTPUT**



**Task 2: GeekforGeek**

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

