**DAA LAB PRACT 6**

**HARSHITA CHAUDHARI**

**A5-B2-23**

<https://github.com/24chaudharih-cmd/Daa-practical-6>

**TASK 1**

**CODE -**

import java.util.\*;

public class Main {

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

double[][] E = new double[n + 1][n + 1];

double[][] W = new double[n + 1][n + 1];

int[][] R = new int[n + 1][n + 1];

// Step 1: Initialization

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

E[i][i] = q[i];

W[i][i] = q[i];

R[i][i] = 0;

}

// Step 2: Compute optimal costs

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

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

int j = i + d;

E[i][j] = Double.MAX\_VALUE;

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

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

double cost = E[i][k - 1] + E[k][j] + W[i][j];

if (cost < E[i][j]) {

E[i][j] = cost;

R[i][j] = k;

}

}

}

}

// Step 3: Print results

System.out.println("E Matrix:");

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

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

System.out.printf("%.2f ", E[i][j]);

}

System.out.println();

}

System.out.println("\nW Matrix:");

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

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

System.out.printf("%.2f ", W[i][j]);

}

System.out.println();

}

System.out.println("\nR Matrix:");

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

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

System.out.print(R[i][j] + " ");

}

System.out.println();

}

}

public static void main(String[] args) {

int n = 3;

double[] p = {0.15, 0.10, 0.05};

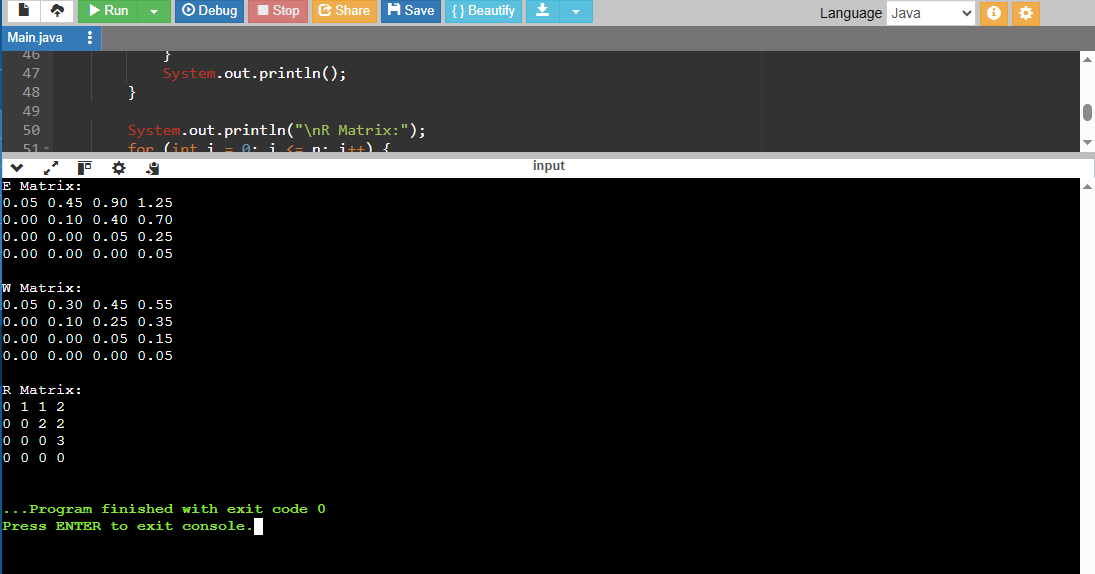
double[] q = {0.05, 0.10, 0.05, 0.05};

optimalBST(p, q, n);

}

}

**Output** -



**TASK 2**

**Code -**

class Solution:

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

dp = [[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 length in range(1, n+1):

for i in range(n - length + 1):

j = i + length - 1

if i == j:

dp[i][j] = freq[i]

else:

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

for r in range(i, j+1):

cost\_left = dp[i][r-1] if r > i else 0

cost\_right = dp[r+1][j] if r < j else 0

cost = cost\_left + cost\_right + sumFreq[i][j]

if cost < dp[i][j]:

dp[i][j] = cost

return dp[0][n-1]

