

Practical No : 06

Name: Prarthana Agrawal

Section : A4_B1

Roll no: 15

Task 1:

Code:

```
#include <stdio.h>
#include <limits.h>
#include <float.h>

void OBST_general(int n, double P[], double Q[], double E[n + 1][n + 1], double W[n + 1][n + 1], int R[n + 1][n + 1]) {
    for (int i = 0; i <= n; i++) {
        E[i][i] = Q[i];
        W[i][i] = Q[i];
        R[i][i] = 0;
    }

    for (int d = 1; d <= n; d++) {
        for (int i = 0; i <= n - d; i++) {
            int j = i + d;

            W[i][j] = W[i][j - 1] + P[j] + Q[j];

            double min_cost = DBL_MAX;
            int k_root = -1;

            for (int k = i + 1; k <= j; k++) {
                double current_cost = E[i][k - 1] + E[k][j] + W[i][j];

                if (current_cost < min_cost) {
                    min_cost = current_cost;
                    k_root = k;
                }
            }
            E[i][j] = min_cost;
            R[i][j] = k_root;
        }
    }
}

void OptimalBST_Successful(double P[], int n, double C[n + 2][n + 2], int R[n + 2][n + 2]) {
    for (int i = 1; i <= n; i++) {
        C[i][i - 1] = 0.0;
        C[i][i] = P[i];
    }
}
```

```

    R[i][i] = i;
}
C[n + 1][n] = 0.0;

for (int d = 1; d <= n - 1; d++) {
    for (int i = 1; i <= n - d; i++) {
        int j = i + d;
        double minval = DBL_MAX;
        int kmin = -1;

        for (int k = i; k <= j; k++) {
            double current_cost = C[i][k - 1] + C[k + 1][j];

            if (current_cost < minval) {
                minval = current_cost;
                kmin = k;
            }
        }
        R[i][j] = kmin;

        double sum = 0.0;
        for (int s = i; s <= j; s++) {
            sum += P[s];
        }

        C[i][j] = minval + sum;
    }
}

void print_matrices_general(int n, double E[n + 1][n + 1], double W[n + 1][n + 1], int R[n + 1][n + 1]) {
    printf("\n--- General OBST Results (Probabilities, Indices 0 to %d) ---\n", n);

    printf("\nExpected Cost Matrix E (E[i,j] is for keys k_i+1 to k_j):\n");
    printf(" "); for(int j=0; j<=n; j++) printf("%6d", j); printf("\n");
    for (int i = 0; i <= n; i++) {
        printf("%d |", i);
        for (int j = 0; j <= n; j++) {
            if (i > j) {
                printf(" - ");
            } else {
                printf("%6.2f", E[i][j]);
            }
        }
        printf("\n");
    }
    printf("\nWeight Matrix W:\n");
    printf(" "); for(int j=0; j<=n; j++) printf("%6d", j); printf("\n");
    for (int i = 0; i <= n; i++) {
        printf("%d |", i);

```

```

    for (int j = 0; j <= n; j++) {
        if (i > j) {
            printf(" - ");
        } else {
            printf("%.2f", W[i][j]);
        }
    }
    printf("\n");
}

printf("\nRoot Matrix R (R[i,j] is the index k_k):\n");
printf(" "); for(int j=0; j<=n; j++) printf("%6d", j); printf("\n");
for (int i = 0; i <= n; i++) {
    printf("%d |", i);
    for (int j = 0; j <= n; j++) {
        if (i >= j) {
            printf(" %d ", R[i][j]);
        } else {
            printf("%6d", R[i][j]);
        }
    }
    printf("\n");
}
printf("\nOptimal Expected Cost E[0][%d] = %.2f\n", n, E[0][n]);
}

void print_matrices_successful(int n, double C[n + 2][n + 2], int R[n + 2][n + 2]) {
    printf("\n--- Successful OBST Results (Indices 1 to %d) ---\n", n);

    printf("\nCost Matrix C (C[i,j] is the cost for keys k_i to k_j):\n");
    printf(" "); for(int j=0; j<=n+1; j++) printf(" %7d", j); printf("\n");
    for (int i = 1; i <= n + 1; i++) {
        printf("%d |", i);
        for (int j = 0; j <= n + 1; j++) {
            if (i > j + 1) {
                printf(" - ");
            } else {
                printf(" %7.2f", C[i][j]);
            }
        }
        printf("\n");
    }

    printf("\nRoot Matrix R:\n");
    printf(" "); for(int j=0; j<=n+1; j++) printf(" %7d", j); printf("\n");
    for (int i = 1; i <= n + 1; i++) {
        printf("%d |", i);
        for (int j = 0; j <= n + 1; j++) {

```

```

        if (i > j + 1) {
            printf(" - ");
        } else {
            printf(" %7d", R[i][j]);
        }
    }
    printf("\n");
}
printf("\nOptimal Cost C[1][%d] = %.2f\n", n, C[1][n]);
}

```

```

int main() {
    int n_gen = 4;
    double P_gen[5] = {0.0, 0.1, 0.2, 0.4, 0.3};
    double Q_gen[5] = {0.05, 0.1, 0.05, 0.05, 0.1};

    double E[5][5];
    double W[5][5];
    int R_gen[5][5];

    OBST_general(n_gen, P_gen, Q_gen, E, W, R_gen);
    print_matrices_general(n_gen, E, W, R_gen);

    int n_succ = 4;
    double P_succ[5] = {0.0, 0.1, 0.2, 0.4, 0.3};

    double C[6][6];
    int R_succ[6][6];

    OptimalBST_Successful(P_succ, n_succ, C, R_succ);
    print_matrices_successful(n_succ, C, R_succ);

    return 0;
}

```

Output:

```

--- General OBST Results (Probabilities, Indices 0 to 4) ---

Expected Cost Matrix E (E[i,j] is for keys k_i+1 to k_j):
0 | 0.05 0.40 0.95 1.95 2.90
1 | - 0.10 0.50 1.35 2.30
2 | - - 0.05 0.60 1.55
3 | - - - 0.05 0.60
4 | - - - - 0.10

Weight Matrix W:
0 | 0.05 0.25 0.50 0.95 1.35
1 | - 0.10 0.35 0.80 1.20
2 | - - 0.05 0.50 0.90
3 | - - - 0.05 0.45
4 | - - - - 0.10

Root Matrix R (R[i,j] is the index k_k):
0 | 0 1 2 3 4
1 | 0 1 2 3 3
2 | 2636 0 2 3 3
3 | 4 0 0 3 3
4 | 0 0 0 0 4
5 | 255 255 0 -16777216 0

Optimal Expected Cost E[0][4] = 2.90

--- Successful OBST Results (Indices 1 to 4) ---

Cost Matrix C (C[i,j] is the cost for keys k_i to k_j):
0 | 0.00 0.10 0.40 1.10 1.70 0.00
1 | - 0.00 0.20 0.80 1.40 0.00
2 | - - 0.00 0.40 1.00 0.00
3 | - - - 0.00 0.30 0.00
4 | - - - - 0.00 0.00

Root Matrix R:
0 | 0 1 2 3 4 5
1 | 0 1 2 3 3 31846
2 | - 32766 2 3 3 31846
3 | - - 0 3 3 31846
4 | - - - 31846 4 0
5 | - - - - -496006448 31846

Optimal Cost C[1][4] = 1.70

=== Code Execution Successful ===

```

```

--- Successful OBST Results (Indices 1 to 4) ---

Cost Matrix C (C[i,j] is the cost for keys k_i to k_j):
0 | 0.00 0.10 0.40 1.10 1.70 0.00
1 | - 0.00 0.20 0.80 1.40 0.00
2 | - - 0.00 0.40 1.00 0.00
3 | - - - 0.00 0.30 0.00
4 | - - - - 0.00 0.00


Root Matrix R:
0 | 0 1 2 3 4 5
1 | 0 1 2 3 3 31846
2 | - 32766 2 3 3 31846
3 | - - 0 3 3 31846
4 | - - - 31846 4 0
5 | - - - - -496006448 31846

Optimal Cost C[1][4] = 1.70

=== Code Execution Successful ===

```

Task 2:



Courses

Tutorials

Practice

Jobs

Start Timer

Problem

Editorial

Submissions

Comments

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed

104 / 104

Attempts: Correct / Total

2 / 2

Accuracy: 100%

Time Taken

1.88

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Solve Next

Fixing Two nodes of a BST

Strictly Increasing Array

Word Wrap

Python3

Start Timer

```

1 import sys
2
3 class Solution:
4     def optimalSearchTree(self, keys, freq, n):
5         cost = [[0 for _ in range(n)] for _ in range(n)]
6
7         for i in range(n):
8             cost[i][i] = freq[i]
9
10        for L in range(2, n + 1):
11            for i in range(n - L + 1):
12                j = i + L - 1
13                cost[i][j] = sys.maxsize
14
15                sum_freq = sum(freq[k] for k in range(i, j + 1))
16
17                for k in range(i, j + 1):
18                    left_cost = cost[i][k - 1] if k > i else 0
19                    right_cost = cost[k + 1][j] if k < j else 0
20
21                    current_cost = left_cost + right_cost + sum_freq
22                    cost[i][j] = min(cost[i][j], current_cost)
23
24        return cost[0][n - 1]

```

Custom Input

Compile & Run

Submit