Assignment no .8

Problem Statement:

Given sequence k = k1 <k2 < ... <kn of n sorted keys, with a search probability pi for each key ki . Build the Binary search tree that has the least search cost given the access probability for each key?.

INPUT:

```
#include <iostream>

#include <limits.h> using

namespace std;

#define SIZE 15


class OBST {     int prob[SIZE] = {};          //Probabilities with which we search for an

element     int keys[SIZE] = {};          //Elements from which OBST is to be built     int

weight[SIZE][SIZE] = {};    //Weight weight[i][j]’ of keys tree having root ’root[i][j]’

int cost[SIZE][SIZE] = {};      //Cost ‘cost[i][j] of keys tree having root ‘root[i][j]     int

root[SIZE][SIZE] = {};      //represents root     int n;                        // number of nodes

public:

   void get_data();     int

Min_Value(int, int);     void

build_OBST();     void

build_tree();     void print(int

[][SIZE], int);

};


/* This function accepts the input data */ void

OBST::get_data() {

   int i;

   cout << "\nOptimal Binary Search Tree \n\nEnter the number of nodes: ";

   cin >> n;     cout << "\nEnter " << n

<< " nodes: ";

   for (i = 1; i <= n; i++)

cin >> keys[i];


   cout << "\nEnter " << n << " probabilities: ";
```

```
    for (i = 1; i <= n; i++)
cin >> prob[i];

}

/* This function returns keys value in the range ‘r[i][j-1]’ to ‘r[i+1][j]’so
that the cost ‘cost[i][k-1]+cost[k][j]’is minimum */ int
OBST::Min_Value(int i, int j) {
    int l, k;
    int minimum = INT_MAX;
    for (l = root[i][j - 1]; l <= root[i + 1][j]; l++)
{        if ((cost[i][l - 1] + cost[l][j]) < minimum)
{            minimum = cost[i][l - 1] + cost[l][j];
            k = l;
        }
    }
    return k;
}

/* This function builds the table from all the given probabilities It
basically computes cost,root,weight values */ void
OBST::build_OBST() {
    int i, j, k, l; for (i = 0;
    i    <    n;    i++)    {
    //initialize
    weight[i][i]            =
    root[i][i]  =  cost[i][i]
    =  0;          //Optimal
    trees with one node
        weight[i][i + 1] = cost[i][i + 1] = prob[i + 1];
root[i][i + 1] = i + 1;
    }
```

```
    weight[n][n] = root[n][n] = cost[n][n] = 0;
//Find optimal trees with 'm' nodes
    for (l = 2; l <= n; l++) {
for (i = 0; i <= n - l; i++) {
            j = i + l;
            weight[i][j] = weight[i][j - 1] + prob[j];              k
= Min_Value(i, j);             cost[i][j] = weight[i][j] +
cost[i][k - 1] + cost[k][j];            root[i][j] = k;
        }
    }

    cout << "\nCost are: \n";
    print(cost, n);

    cout << "\nRoot are: \n";
    print(root, n);
}

/* This function builds the tree from the tables made by the OBST function */
void OBST::build_tree() {
    int i, j, k;
    int queue[20], front = -1, rear = -1;
    cout << "\nThe Optimal Binary Search Tree For the Given Nodes Is…\n"; cout
    << "\nThe Root of this OBST is:: " << keys[root[0][n]];

    cout << "\nThe Cost of this OBST is:: " << cost[0][n];
    cout << "\n\n\tNODE\tLEFT CHILD\tRIGHT
CHILD";     cout << "\n";     queue[++rear] = 0;
queue[++rear] = n;     while (front != rear) {         i =
queue[++front];         j = queue[++front];
        k = root[i][j];         cout
<< "\n\t" << keys[k];
```

```
        if (root[i][k - 1] != 0) {              cout <<
"\t\t" << keys[root[i][k - 1]];
queue[++rear] = i;             queue[++rear] =
k - 1;
        }
        else              cout << "\t\t";         if
(root[k][j] != 0) {              cout << "\t"
<< keys[root[k][j]];
queue[++rear] = k;
queue[++rear] = j;
        }
        else
cout << "\t";
    }
    cout << "\n";
}

void OBST::print(int arr[][SIZE], int n) {
    int i, j; for(i = 0; i <=
    n; i++) {          for(j =
    0; j <= n; j++)
    cout << arr[i][j] <<
    '\t';          cout << '\n';
    }
}

int main() {
OBST obj;
obj.get_data();
obj.build_OBST();
obj.build_tree();
return 0;
```

```
}

/*
Optimal Binary Search Tree

Enter the number of nodes: 9

Enter 9 nodes: 8 3 10 1 6 14 4 13 7

Enter 9 probabilities: 4 8 2 1 4 2 6 4 7

Cost are:
0    4    16    20    23    34    40    60    72    93
0    0    8    12    15    26    32    48    60    81
0    0    0    2    4    11    15    29    38    56
0    0    0    0    1    6    10    23    31    49
0    0    0    0    0    4    8    20    28    46
0    0    0    0    0    0    2    10    18    36
0    0    0    0    0    0    0    6    14    30
0    0    0    0    0    0    0    0    4    15
0    0    0    0    0    0    0    0    0
7 0    0    0    0    0    0    0    0    0
0

Root are:
0    1    2    2    2    2    2    2    5    7
0    0    2    2    2    2    2    5    5    7
0    0    0    3    3    5    5    5    7    7
0    0    0    0    4    5    5    7    7    7
0    0    0    0    0    5    5    7    7    7
0    0    0    0    0    0    6    7    7    7
0    0    0    0    0    0    0    7    7    8
```

```
0	0	0	0	0	0	0	0	8	9
0	0	0	0	0	0	0	0	0
9 0	0	0	0	0	0	0	0	0
0

The Optimal Binary Search Tree For the Given Nodes Is…

The Root of this OBST is:: 4
The Cost of this OBST is:: 93

	NODE	LEFT CHILD	RIGHT CHILD

	4	3	7
	3	8	6
	7	13
	8
	6	10	 14
	13
	10		1
	14
	1
*/
```