**Name: Affan Shaikh**

**Roll no: COB227**

**Code :**

#include <iostream>

#define SIZE 10

using namespace std;

class optimal {

public:

    int p[SIZE];

    int q[SIZE];

    int a[SIZE];

    int w[SIZE][SIZE];

    int c[SIZE][SIZE];

    int r[SIZE][SIZE];

    int n;

    int front, rear, queue[20];

    optimal() {

        front = rear = -1;

    }

    void getdata();

    intminvalue(int, int);

    void OBST();

    void buildtree();

};

void optimal::getdata() {

    inti;

    cout<< "\n Optimal Binary search tree";

    cout<< "\n Enter the number of nodes :";

    cin>> n;

    cout<< "\n Enter the data : \n";

    for (i = 1; i<= n; i++) {

        cout<< "\n a[" <<i<< "]: ";

        cin>> a[i];

    }

    cout<< "\n Enter probabilities for successful search \n";

    for (i = 1; i<= n; i++) {

        cout<< "p[" <<i<< "]: ";

        cin>> p[i];

    }

    cout<< "\n Enter probabilities for unsuccessful search \n";

    for (i = 1; i<= n; i++) {

        cout<< "q[" <<i<< "]: ";

        cin>> q[i];

    }

}

intoptimal::minvalue(inti, int j) {

    int m, k;

    int min = 32000;

    for (m = r[i][j - 1]; m <= r[i + 1][j]; m++) {

        if ((c[i][m - 1] + c[m][j]) < min) {

            min = c[i][m - 1] + c[m][j];

            k = m;

        }

    }

    return k;

}

void optimal::OBST() {

    inti, j, k, m;

    for (i = 0; i< n; i++) {

        w[i][i] = q[i];

        r[i][i] = c[i][i] = 0;

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

        r[i][i + 1] = i + 1;

        c[i][i + 1] = q[i] + q[i + 1] + p[i + 1];

    }

    w[n][n] = q[n];

    r[n][n] = c[n][n] = 0;

    for (m = 2; m <= n; m++) {

        for (i = 0; i<= n - m; i++) {

            j = i + m;

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

            k = minvalue(i, j);

            c[i][j] = w[i][j] + c[i][k - 1] + c[k][j];

            r[i][j] = k;

        }

    }

}

void optimal::buildtree() {

    inti, j, k;

    cout<< "\n The optimal Binary search tree for given nodes is : \n";

    cout<< "\n The root of this OBST is :" << r[0][n];

    cout<< "\n The cost of this OBST is: " << c[0][n];

    cout<< "\n\n Node \t Left child \t Right child";

    cout<< "\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" <<endl;

    queue[++rear] = 0;

    queue[++rear] = n;

    while (front != rear) {

        i = queue[++front];

        j = queue[++front];

        k = r[i][j];

        cout<< "\n\t" << k;

        if (r[i][k - 1] != 0) {

            cout<< "       " << r[i][k - 1];

            queue[++rear] = i;

            queue[++rear] = k - 1;

        } else {

            cout<< "      ";

        }

        if (r[k][j] != 0) {

            cout<< "         " << r[k][j];

            queue[++rear] = k;

            queue[++rear] = j;

        } else {

            cout<< "          ";

        }

    }

    cout<<endl;

}

intmain() {

    optimal obj;

    obj.getdata();

    obj.OBST();

    obj.buildtree();

    return 0;

}

**Output :**

Optimal Binary search tree

Enter the number of nodes :5

Enter the data :

a[1]: 1

a[2]: 2

a[3]: 3

a[4]: 4

a[5]: 5

Enter probabilities for successful search

p[1]: 3

p[2]: 2

p[3]: 7

p[4]: 4

p[5]: 2

Enter probabilities for unsuccessful search

q[1]: 2

q[2]: 1

q[3]: 4

q[4]: 6

q[5]: 3

The optimal Binary search tree for given nodes is :

The root of this OBST is :3

The cost of this OBST is: 89

Node Left child Right child

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3 1 4

1 2

4 5

2

5