**Practical 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?**

**\*/**

**Fm**

**Cost of tree =**

**Sum of[no of nodes \* no of comparisons(height of tree)] / total no of node present**

#include<iostream>

#define SIZE 10

using namespace std;

class OBST

{

private:

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];

public:

OBST();

void get\_data();

int Min\_Value(int,int);

void OBST1();

void build\_tree();

};

OBST::OBST()

{

front=rear=-1;

}

void OBST::get\_data()

{

int i;

cout<<"\nOptimal Binary Search Tree\n";

cout<<"\nEnter the number of nodes::";

cin>>n;

cout<<"\nEnter the data as....\n";

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

{

cout<<"\na["<<i<<"]:";

cin>>a[i];

}

cout<<"\nEnter the Probabilities for successful searches::";

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

{

cout<<"\np["<<i<<"]:";

cin>>p[i];

}

cout<<"\nEnter the Probabilities for unsuccessful searches::";

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

{

cout<<"\nq["<<i<<"]:";

cin>>q[i];

}

}

int OBST::Min\_Value(int i,int j)

{

int m,k;

int minimum=32000;

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

{

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

{

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

k=m;

}

}

return k;

}

void OBST::OBST1()

{

int i,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=Min\_Value(i,j);

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

r[i][j]=k;

}

}

}

void OBST::build\_tree()

{

int i,j,k;

cout<<"\nThe Optimal Binary Search Tree For The Given Nodes Is.......";

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

cout<<"\nThe Cost of this OBST is::"<<c[0][n];

cout<<"\n\n\n\tNODE\tLEFT CHILD\tRIGHT 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;

}

int main()

{

OBST obj;

obj.get\_data();

obj.OBST1();

obj.build\_tree();

return 0;

}

**Output**

Optimal Binary Search Tree

Enter the number of nodes::4

Enter the data as....

a[1]:1

a[2]:2

a[3]:3

a[4]:4

Enter the Probabilities for successful searches::

p[1]:2

p[2]:6

p[3]:3

p[4]:1

Enter the Probabilities for unsuccessful searches::

q[0]:8

q[1]:7

q[2]:9

q[3]:3

q[4]:5

The Optimal Binary Search Tree For The Given Nodes Is.......

The root of OBST is:: 2

The Cost of this OBST is::91

NODE LEFT CHILD RIGHT CHILD

------------------------------------

2 1 3

1 - -

3 - 4

4 - -