Advanced Data Structures Assignment

Name : Shrirang R Mhalgi

Roll number : 222006

Class : SE B

Batch : B1

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

**Problem Statement:**

Convert given binary tree into threaded binary tree. Analyze time and space complexity of the algorithm.

**Code:**

//Convert given binary tree into threaded binary tree. Analyze time and space complexity of the algorithm.

#include<iostream>

using namespace std;

int flag=1,i,arr[10],z=0,k;

int c;

class node

{

int data;

node \*right;

node \*left;

friend class BST;

};

class thread

{

int data;

thread \*left;

thread \*right;

int lbit;

int rbit;

//thread \*lt;

//thread \*rt;

thread(){

left=NULL;

right=NULL;

lbit=0;

rbit=0;

data=0;

// lt=NULL;

// rt=NULL;

}

friend class BST;

};

class BST

{

node \*root,\*temp,\*New;

thread \*dummy,\*f;

public:

int create();

void inodr(node \*);

void convert();

node\* search(node \*, int);

void show\_tbt();

thread \* ret\_d(){

return dummy;

}

node \* ret\_r(){

return root;

}

};

int BST :: create()

{

New= new node;

cout<<"Enter data : ";

cin>>New->data;

New->right=NULL;

New->left=NULL;

if(flag==1)

{

root=New;

temp=root;

flag=0;

}

else

{

temp=root;

while(1)

{

if(temp->data>New->data)

{

if(temp->left==NULL)

{

temp->left=New;

temp=New;

break;

}

else

{

temp=temp->left;

continue;

}

}

if(temp->data<New->data)

{

if(temp->right==NULL)

{

temp->right=New;

temp=New;

break;

}

else

{

temp=temp->right;

continue;

}

}

}

}

}

void BST :: inodr(node \*root){

if(root!=NULL)

{

inodr(root->left);

arr[z]=root->data;

z++;

inodr(root->right);

}

}

void BST :: convert(){

i=0;

node \*temp;

thread \*New1,\*rec;

dummy = new thread;

dummy->data=99999;

New1= new thread;

New1->data=arr[i];

while(i<c){

if(i!=c-1)

{

temp=search(ret\_r(),arr[i+1]);

rec = new thread;

rec->data=temp->data;

rec->left=New1;

New1->right=rec;

if(temp->left!=NULL)

rec->lbit=1;

if(temp->right!=NULL)

rec->rbit=1;

if(i==0)

{

New1->left=dummy;

f=New1;

}

New1=New1->right;

}

else

{

rec->right=dummy;

}

i++;

}

//cout<<f->data<<"A";

}

node\* BST :: search(node \*root, int key){

if(root!=NULL)

{

if(key==root->data)

return root;

else if(key>root->data)

search(root->right,key);

else if(key<root->data)

search(root->left,key);

}

else return NULL;

}

void BST :: show\_tbt(){

cout<<"\nTraversal of threaded binary tree\n";

//cout<<f->data;

while(f->right!=dummy)

{

cout<<f->data<<"\t";

f=f->right;

}

cout<<f->data;

}

int main()

{

BST o;

cout<<"Enter the number of elements : ";

cin>>c;

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

o.create();

o.inodr(o.ret\_r());

o.convert();

o.show\_tbt();

}

**Output:**

Enter the number of elements : 6

Enter data : 10

Enter data : 7

Enter data : 13

Enter data : 8

Enter data : 9

Enter data : 12

Traversal of threaded binary tree

7 8 9 10 12 13