Advanced Data Structures Assignment

Name : Shrirang R Mhalgi

Roll number : 222006

Class : SE B

Batch : B1

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

Problem Statement:

Beginning with an empty binary search tree, Construct binary search tree by inserting the values in the order given. After constructing a binary tree - i. Insert new node ii. Find number of nodes in longest path iii. Minimum data value found in the tree iv. Change a tree so that the roles of the left and right pointers are swapped at every node v. Search a value

Code:

#include<iostream>

using namespace std;

int flag=1,key;

class node

{

int data;

node \*next;

node \*prev;

friend class BST;

};

class BST

{

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

public:

int create();

void display(node \*);

void insert();

void ins(node \*);

int search(node \*);

void min();

node\* swap(node \*);

int longp(node \*);

node \* ret\_r(){

return root;

}

};

int BST :: create(){

char ans;

New= new node;

cout<<"Enter data : ";

cin>>New->data;

New->next=NULL;

New->prev=NULL;

if(flag==1)

{

root=New;

temp=root;

flag=0;

}

else

{

temp=root;

while(1)

{

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

{

if(temp->prev==NULL)

{

temp->prev=New;

temp=New;

break;

}

else

{

temp=temp->prev;

continue;

}

}

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

{

if(temp->next==NULL)

{

temp->next=New;

temp=New;

break;

}

else

{

temp=temp->next;

continue;

}

}

}

}

}

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

if(root!=NULL)

{

display(root->prev);

cout<<root->data<<endl;

display(root->next);

}

}

void BST :: insert(){

New= new node;

cout<<"\nEnter data of new node to be inserted: ";

cin>>New->data;

New->next=NULL;

New->prev=NULL;

ins(ret\_r());

}

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

if(New->data<root->data && root->prev==NULL)

{

root->prev=New;

}

else if(New->data>root->data && root->next==NULL)

{

root->next=New;

}

else if(New->data<root->data && root->prev!=NULL)

{

ins(root->prev);

}

else if(New->data>root->data && root->next!=NULL)

{

ins(root->next);

}

}

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

if(root!=NULL)

{

if(key==root->data)

return 1;

else if(key>root->data)

search(root->next);

else if(key<root->data)

search(root->prev);

}

else return 0;

}

void BST :: min(){

temp=root;

while(temp->prev!=NULL)

{

temp=temp->prev;

}

cout<<"\nMinimum data value found in the tree : "<<temp->data<<endl;

}

node \* BST :: swap(node \*temp){

node \*t;

if(temp!=NULL)

{

t=temp->prev;

temp->prev=swap(temp->next);

temp->next=swap(t);

}

return temp;

}

int BST :: longp(node \*temp){

int l,r;

if(temp==NULL)

return 0;

if(temp->prev==NULL && temp->next==NULL)

return 0;

l=longp(temp->prev);

r=longp(temp->next);

if(l>r)

return (l+1);

return (r+1);

}

int main(){

BST o;

node \*root,\*New;

int c,d,v,e;

do{

cout<<"\nEnter your choice\n";

cout<<"1: Creation of tree\n";

cout<<"2: Display tree\n";

cout<<"3: Insert Node\n";

cout<<"4: Search\n";

cout<<"5: Find minimum data value in tree\n";

cout<<"6: Swap\n";

cout<<"7: Longest path in tree\n";

cout<<"0: Exit\n";

cin>>d;

switch(d)

{

case 1: cout<<"\nEnter number of nodes : ";

cin>>c;

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

o.create();

break;

case 2: cout<<"\nInorder display\n";

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

break;

case 3: o.insert();

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

break;

case 4: cout<<"\nEnter value to be searched : ";

cin>>key;

v = o.search(o.ret\_r());

if(v==1)

cout<<"\n Found\n";

else

cout<<"\n Not Found\n";

break;

case 5: o.min();

break;

case 6: o.swap(o.ret\_r());

cout<<"\nIn-order traversal of nodes after swapping\n";

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

break;

case 7: e=o.longp(o.ret\_r());

cout<<e;

}

}while(d!=0);

}

Output:

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

1

Enter number of nodes : 4

Enter data : 10

Enter data : 5

Enter data : 15

Enter data : 20

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

2

Inorder display

5

10

15

20

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

3

Enter data of new node to be inserted: 7

5

7

10

15

20

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

2

Inorder display

5

7

10

15

20

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

4

Enter value to be searched : 10

Found

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

5

Minimum data value found in the tree : 5

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

7

2

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

6

In-order traversal of nodes after swapping

20

15

10

7

5

Enter your choice

1: Creation of tree

2: Display tree

3: Insert Node

4: Search

5: Find minimum data value in tree

6: Swap

7: Longest path in tree

0: Exit

0