ADS

JSPM

Page No.: 1

Assignment No-1

*Title :- program for constructing the binary tree and print the node e.g. book.

* Objective :-

To study and implement the binary tree.

* problem statement :-

A book consist of chapter, chapters consist of section and sections consist of subsection construct a tree and print the nodes find the time and space requirements of your Method.

*Outcomes :-

Input

1) Enter chapter name.
2) Enter chapter containing section.
3) Enter section containing subsections.

output :- Display the output of binary tree.

* Hardware Requirement :-

8GB ram, 500GB dual processor.

* Software Requirement :-

gedit software, terminal on fedora

* Theory :-

* Binary tree :-

Binary tree is define as parent node having at least two node then it will called Binary tree

A tree is Binary tree each node of the tree can have maximum of two childern. moreover childern of a node of Binary tree are ordered. One child is called the "left" child and the other is called the "Right" child An Example of binary tree as shown below.

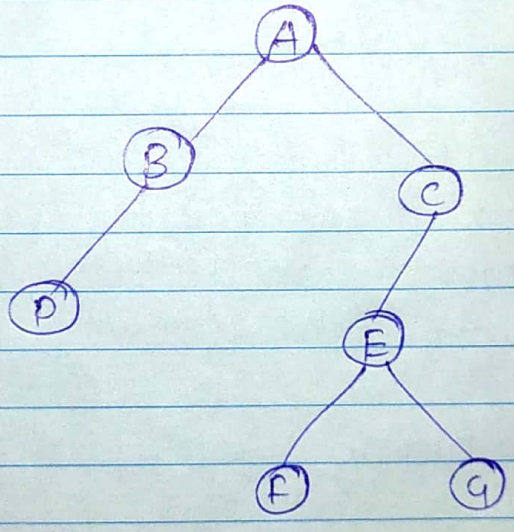


fig. Binary tree.

In above fig. Node A has two childern B and C. with have one child name G and D respectively

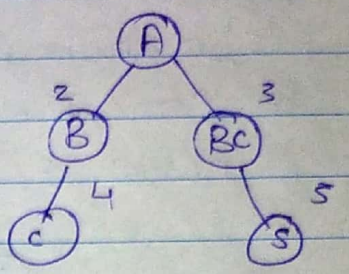
* Properties of Binary tree :-

1) Height of binary tree :-

° Maximum height of Binary tree :-

$H_{max} = N$

e.g.:- N = 5 then, $H_{max} = 5$



° Minimum height of the tree :-

$H_{min} = [\log_2 8] + 1$

$= 3 + 1$

$= 4$

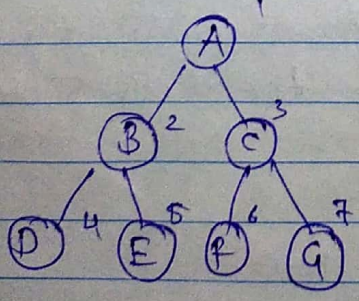
2) No. of nodes = N_{min} = N(height of tree)

Max no. of nodes = $N_{MAX} = 2^H - 1$

* Representation of a Binary tree using an array :-

In order to represent a tree in a single one-dimensional array, the nodes are number sequentially level by level left to right. Even empty nodes are numbered.

When the data of the tree is stored in an array then the number appointing against the node will work as indicates of the nodes from an array.



0	1	2	3	4	5	6	7
7	A	B	C	D	E	F	G

Fig. Numbering of Nodes

Location number zero of the array can be using to store the size of the tree in terms of total number of nodes (Existing or not existing).

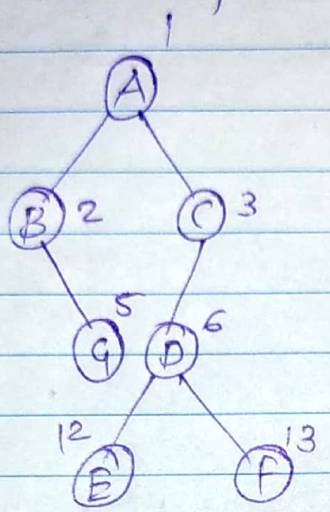


fig. Numbering of nodes

0	1	2	3	4	5	6	7	8	9	10	11	12	13
13	A	B	C	\0	G	D	\0	\0	\0	\0	\0	E	F

fig. Array representation of the tree.

All non-Existing childern are show by "\0" in array

Index of the left child of a node $= i = 2*i$
Index of right child of node $i = 2*i+1$
Index of parent of a node $i = i/2$.

Sibling a node i will be found at the location i+1, if i is a left child of its parent similarly, if i is a right child of its parent then its siblings will be found at i-1:

*. Linked representation of Binary tree:-

Linked representation of a binary tree is more efficient than array representation. A node binary

hree consists of three fields as shown below:

* Data
* Address of Left child
* Address of Right child.

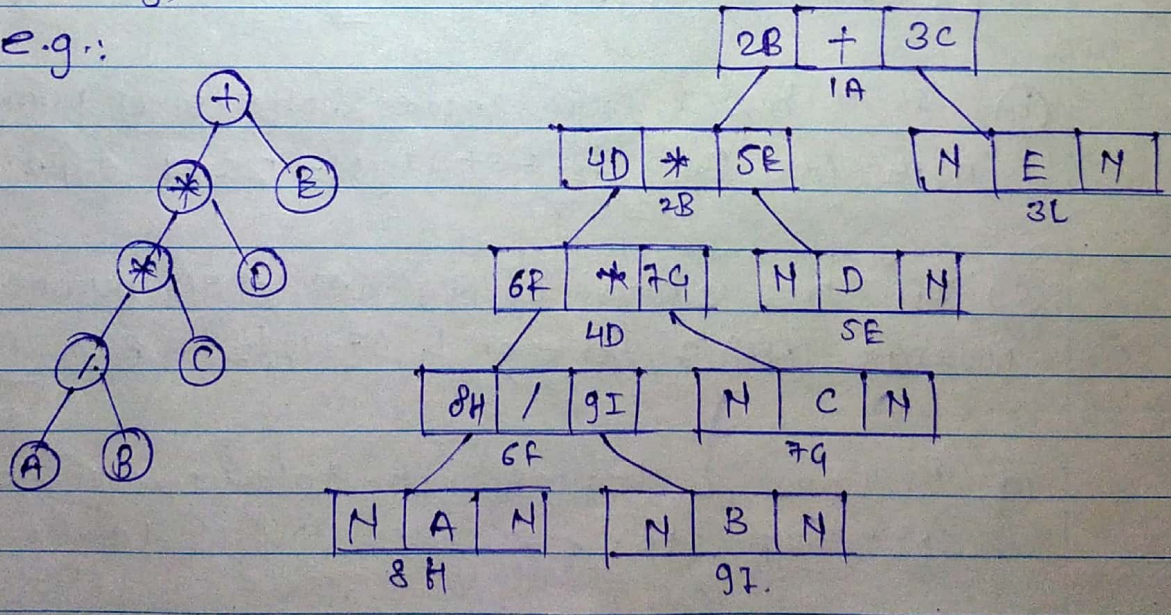
Left	Data	Right.

Left and Right are pointer type fields. Left hands holds the address of Left child and right side holds the address of right child.

In C++ language, the following structure can be It is assumed that a node contains an integer data.

```
struct node
{
    int data;
    node *left;
    node *Right;
};
```

e.g.:



* Types of Binary tree :-

① full Binary tree :-

A full Binary tree is a binary tree in which each node has exactly zero or two childern.

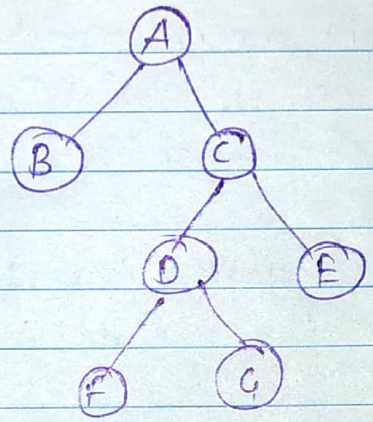


fig. full Binary tree.

- Therom :- full Binary tree.

Let T be a non-empty, full binary tree then,

(a) If T has I internal nodes, then the number of leaves is L = I + 1 for eg. L = 3 + 1 = 4 (leaf node).

(b) If T has I internal nodes, the total number of nodes in a BST ~~in fact~~ is N = 2I + 1 e.g. 2(3) + 1 = 7

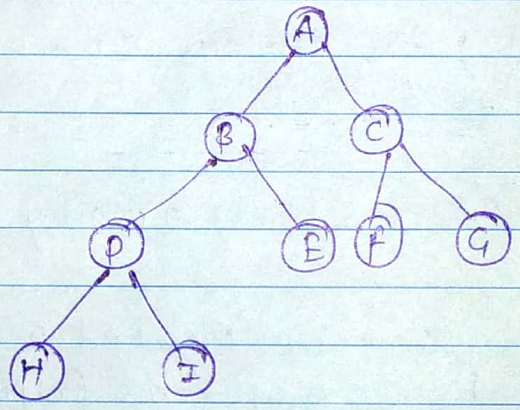
(c) If T has a total of N nodes. then number of internal nodes are (N-1)/2 for e.g L = (7+1)/2 = 8/2 = 4

(d) If T has L leaves, the total number of node is N = 2L - 1 for e.g. N = 2 (No. of leaf node i.e 4

2) Complete Binary tree:-

A complete binary tree is a binary tree in which each node is completely filled with the possible exception of bottom level, which is filled from left to right.

All leaf nodes of complete binary tree is present at or n-1 levels.



Therom-:

1) level i has 2 nodes for e.g level 2 has $2^2 = 4$ nodes

2) In a tree of height h leaves are at level h; h-1

3) No. of leaves is $L = 2^n$. This formula indicates. How many leaves is $L = 2^n$ this formula indicates that present at that particular h.

4) No. of internal nodes $= 1 + 2 + 2^2 + \ldots 2^{n-1} + 2^n = 2^n - 1$

for e.g $n = 3$. so, $2^n - 1 = 2^3 - 1 = 8 - 1 = 7$.

5) How to calculate no. of internal nodes in complete tree is no. of internal node (I) = no. of leaf - 1

for. e.g In above tree I = 3 - 1 = 4 internal nodes.

⑥ Total no. of maximum node in CBT if height is given $N = 2^{h+1} - 1$ i.e $N = 2^{3+1} = 1\,2^4 - 1 = 16 - 1 \neq 15$

⑦ Suppose, In CST total nodes N is given and. give have of find out no. of leaf node $L = (n+1)/2$ for. e.g find in above CST : $L = (9+1)/2 = 5$

⑧ In CST total nodes N is given you have to findout the height of channing tree: (Height (H) = floor $(\log_2(h))$ or. $H = \log_2$ (no. of levels) for e.g $H = \log_2(9) = 3$

* C++ function for creation of a binary tree:-

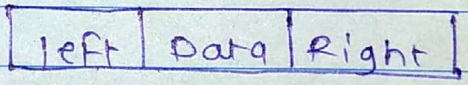
A binary tree can be created recursively. The function given below works as follows.

1) Read a new data in x.
2) Acquire memory for new nodes. for with its address in pointer P.
3) store the data x in the node p.
4) Recursively create the right subtree of P and make it the right child.

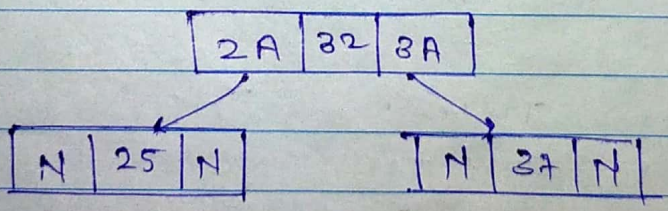
```
Struct node
{   int data;
    node *leftchild; node *rightchild;
};
```

```
node create()
{
    node *p;
    int x;
    cout<<"\n Enter a data (-1 for no data)";
    cin>>x;
    if(x==-1)
        return NULL;
    p=new node;
    p->data=x;
    cout<<"\n Enter left child of "<<x;
    p->left=create();
    cout<<"\n Enter right child of "<<x;
    p->right=create();
    return p;
}
```

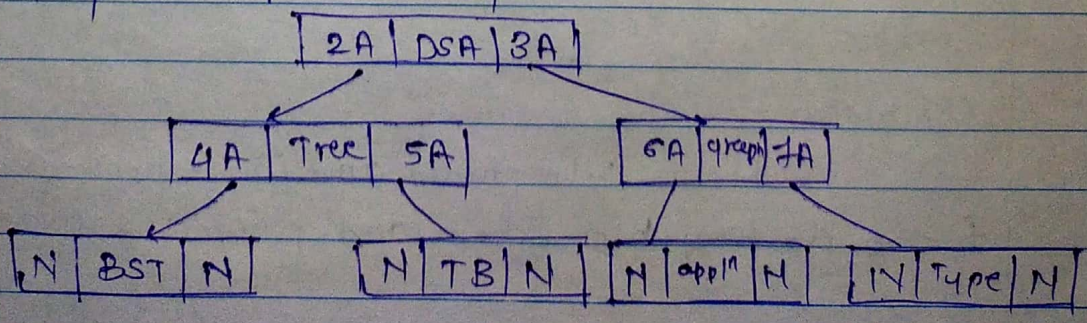
for. e.g



① 32 → 37.



• Display the output of binary tree:-



DSA → Tree → BST → TB → graph → application → type.

• Algorithm -:

i) start.

ii) Take declare structure of class.

iii) Take data a member and Memberfunction. as per need in definition of (reate fun^n display measure. Enter book name -1 if book is not available

iv) If book name is enter, then allocate memory to store this book and its NULL RC-NULL

v) Repeat step 4 until the LC and RC=1

vi) Call appropriate fun^n.

vii) Stop.

Test case -:

Test case NO.	Test case	Expected Result	Actual result	Status fail/Pass.
1.	check tree is NULL	Show the tree is NULL	Tree is NULL	Pass.
2.	cheek whether the tree has 3 field like structure.	The tree structure is showed successfully	Tree structure show successfully	pass.

• Conclusion -:

Hence, we should and implemented the program for ~~a~~ creating binary tree and print its node.

Group A1
Title:- Construct The Tree And Print The Nodes

```
#include<iostream>
using namespace std;

struct TreeNode
{
    int data;
    TreeNode *left;
    TreeNode *right;
};
class BinTree
{
    TreeNode *root,Temp;
    public:
        TreeNode* create();
        TreeNode* insert(TreeNode *);
        void Inorder(TreeNode *);
        void PostOrder(TreeNode *);
        void PreOrder(TreeNode *);
        void Display(TreeNode *);
};
TreeNode* BinTree::create()
{
    TreeNode *p;
    cout<<"\nEnter the Data For Root Node";
    p = new TreeNode;
    cin>>p->data;
    p->left=NULL;
    p->right=NULL;
    root=p;
    cout<<"Node Has Been Inserted "<<root->data;
    return root;
}
TreeNode* BinTree::insert(TreeNode *root)
{
    int db;
    char ans,ans2;
    TreeNode *node;
    node = new TreeNode;
do{
    cout<<"\nEnter the Data";
    cin>>db;
    if(db==-1)
```

```
    {
        return NULL;
    }
    else
    {
        node->data=db;
        node->left=NULL;
        node->right=NULL;
            cout<<"Do You wana Insert This Node at Left";
            cin>>ans;
            if(ans=='y')
            {
                root->left=node;
                cout<<"\nInserted at Left";
            }
            else if(root->right==NULL)
            {
                cout<<"\nDo You Wana Insert This Node At Right";
                cin>>ans;
                if(ans=='y')
                    root->right=node; cout<<"\nInserted at Right";

            }
    }
    cout<<"\nCountinue??";
    cin>>ans2;
 }while(ans2=='y');
    return root;
}
void BinTree::PreOrder(TreeNode *root)
{
    if(root)
    {
        cout<<"\t"<<root->data;
        PreOrder(root->left);
        PreOrder(root->right);
    }
}
void BinTree::Inorder(TreeNode *root)
{
    if(root)
    {
        Inorder(root->left);
        cout<<"\t"<<root->data;
        Inorder(root->right);
```

```
    }
}
void BinTree::PostOrder(TreeNode *root)
{
    if(root)
    {
        PostOrder(root->left);
        PostOrder(root->right);
        cout<<"\t"<<root->data;
    }
}
void BinTree::Display(TreeNode *root)
{
    TreeNode *temp=NULL;
    temp=root;
    while(temp!=NULL)
    {
        if(temp->left!=NULL)
        {
            cout<<temp->data;
            temp=temp->left;
        }
        else if(temp->right!=NULL){
            cout<<temp->data;
            temp=temp->right;
        }
    }
}
int main()
{
    BinTree B1;
    TreeNode *root;
    int in; char b;
    do
    {
        cout<<"\n1.Create\n2.Insert\n3.Preorder\n4.Inorder\n5.Postorder\n6.Displa
y\nEnter Tour choice";
        cin>>in;
        switch(in)
        {
            case 1: root=B1.create();
                    break;
            case 2: B1.insert(root);
                    break;
            case 3: B1.PreOrder(root);
```

```
                    break;
            case 4: B1.Inorder(root);
                    break;
            case 5: B1.PostOrder(root);
                    break;
            case 6: B1.Display(root);
                    break;

        }
        cout<<"\nDo You Want to Perform any other Operations ??";
        cin>>b;
    } while (b=='y');
    return 0;
}
```

Output:-

1.Create

2.Insert

3.Preorder

4.Inorder

5.Postorder

6.Display

Enter Tour choice1

Enter the Data For Root Node50

Node Has Been Inserted 50

Do You Want to Perform any other Operations ??y

1.Create

2.Insert

3.Preorder

4.Inorder

5.Postorder

6.Display

Enter Tour choice2

Enter the Data60

Do You wana Insert This Node at Lefty

Inserted at Left

Countinue??y

Enter the Data60

Do You wana Insert This Node at Leftn

Do You Wana Insert This Node At Righty

Inserted at Right

Countinue??y

Enter the Data-1

Do You Want to Perform any other Operations ??y

1.Create

2.Insert

3.Preorder

4.Inorder

5.Postorder

6.Display

Enter Tour choice3

50 60 60

Do You Want to Perform any other Operations ??y

1.Create

2.Insert

3.Preorder

4.Inorder

5.Postorder

6.Display

Enter Tour choice4

60 50 60

Do You Want to Perform any other Operations ??y

1.Create

2.Insert

3.Preorder

4.Inorder

5.Postorder

6.Display

Enter Tour choice5

60 60 50

Do You Want to Perform any other Operations ??y

1.Create

2.Insert

3.Preorder

4.Inorder

5.Postorder

6.Display

Enter Tour choice