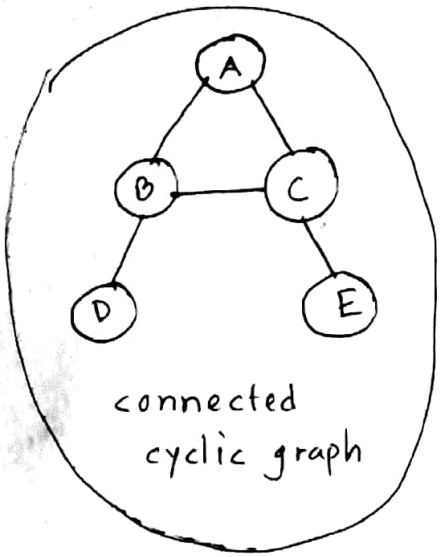
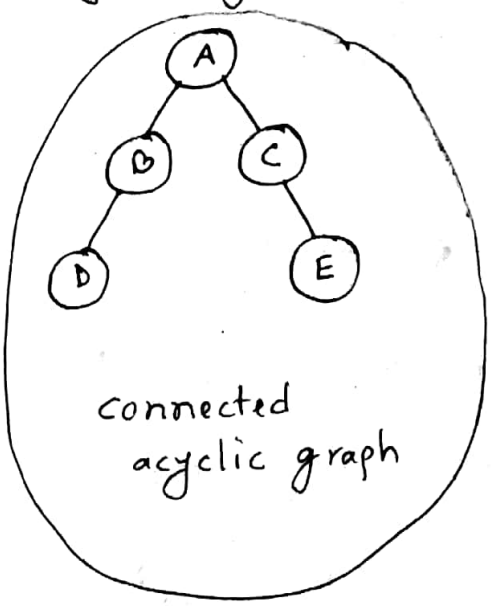
Binary Tree

Tree:

A Tree is a connected acyclic graph.

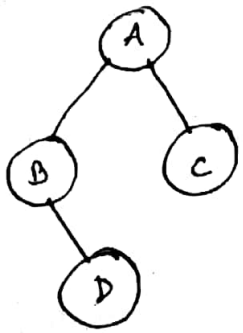


Not tree



Tree

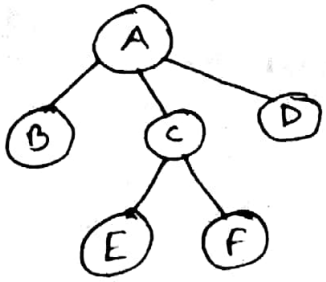
Different Trees:



⇐ Binary Tree

A → B, C → 2 children
B → D → 1 child
D, C → No children

} At most 2 children so, binary Tree

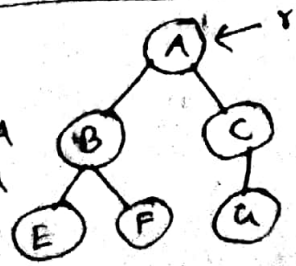


A → B, C, D → 3 children
C → E, F → 2 children
B, D, E, F → No child

At most 3 children
so, Ternery Tree

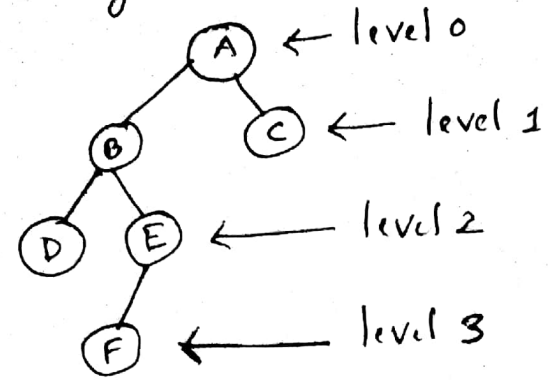
Family structure of Tree:

Father of B is A
" of C is A
Grand father of E is A
" " " F is A
" " " G is A



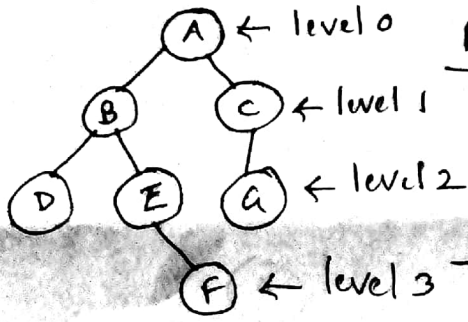
A has two children B, C
B " " " E, F
C " one child G
E, F, G No child → Leaf node

B, C are siblings
E, F are siblings



$$height(Tree) = max(level_i) = 3$$

Binary Tree Traversal Technique:



Preorder (root – left – right)
A B D E F C G

Inorder (left – root – right)
D B E F A G C

Postorder (left – right – root)
D F E B G C A

Level order (level by level)
A B C D E G F

```
void Preorder(node *root){
    if (root != NULL){
root →  printf("%c", root→data);
Left →  if (root→left != NULL)
            Preorder(root→left);
right → if (root→right != NULL)
            Preorder(root→right);
    }
}
```

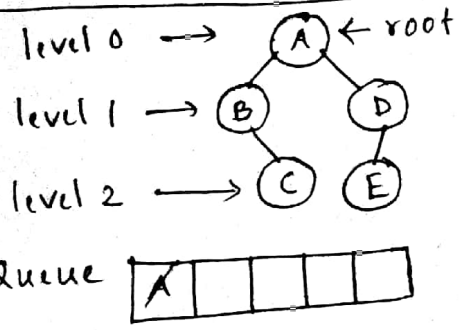
```
void Inorder(node *root){
    if (root != NULL){
left →  if (root→left != NULL)
            Inorder(root→left);
root →  printf("%c", root→data);
right → if (root→right != NULL)
            Inorder(root→right);
    }
}
```

Postorder algorithm try yourself

Level order Algorithm

```
Queue ⇐ root
while( Queue != Empty){
        v ⇐ Queue
        print*, v
        if ( left-child(v) exists)
              Queue ⇐ left-child(v)
        if (right- child(v) exists)
              Queue ⇐ right-child(v)
}
```

Mechanism of level order algorithm



v ⇐ Queue	Print*, v	Queue ⇐ left-child(v)	Queue ⇐ right-child(v)
v = 'A'	A	Queue B	Queue D
v = 'B'	B	~~Queue~~ X	Queue C
v = 'D'	D	Queue E	X
v = 'C'	C	X	X
v = 'E'	E	X	X

Visited Sequence = ABDCE ⇐ level order sequence

Find a binary tree from the following tree traversal sequences

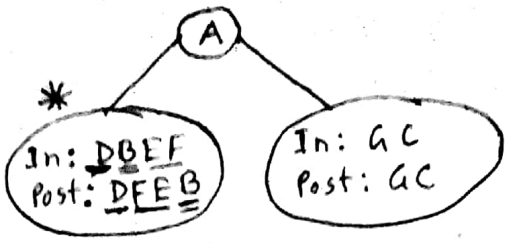
Inorder: DBEFAGC (left-root-right)

Postorder: DFEBGCA (left-right-root)

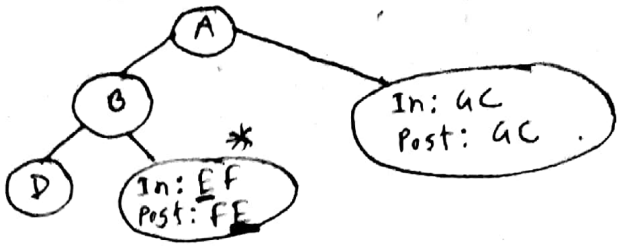
Ans:

root

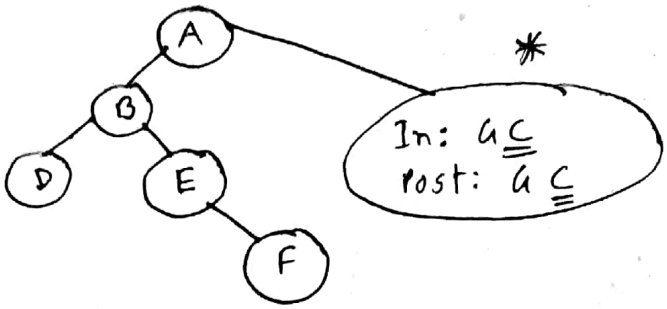
A



B



E



C