C-Programming and DS

Today Class Topics

Trees :

-N-ary tree

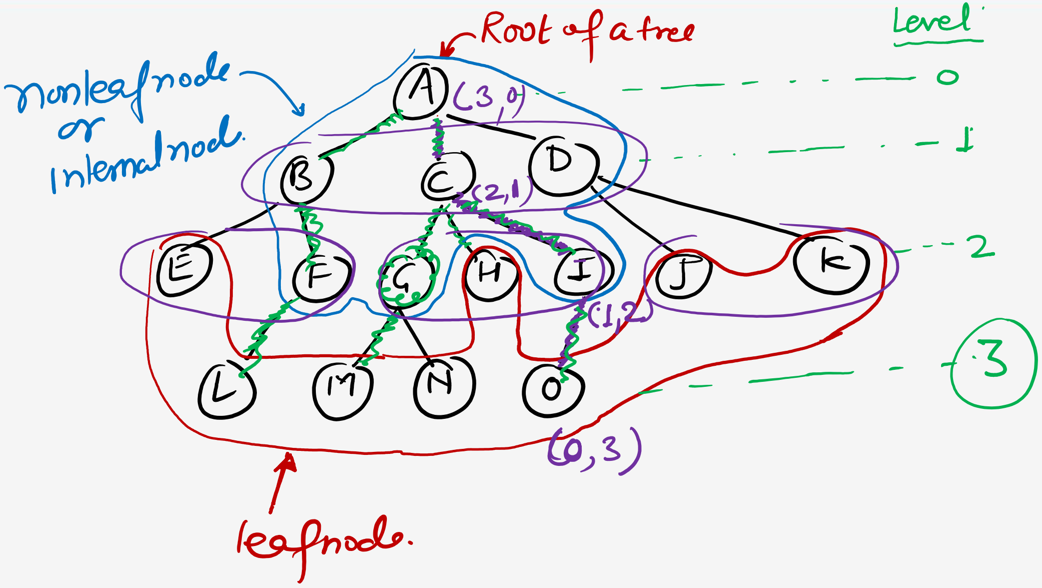
-Binary tree

-Tree traversal

-Binary search tree(BST)

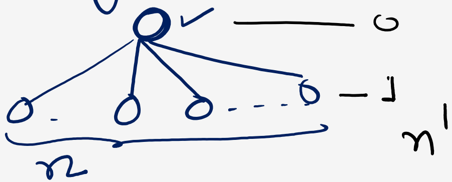
Trees:-

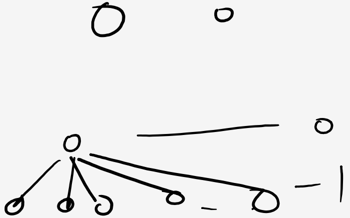
1) n-ary Tree ✓
2) Binary Tree ✓



n-ary tree: A rooted tree is called as n-ary tree if every internal vertex has no more than n children. The tree is called a full n-ary tree if every internal vertex has exactly n-children.

An n-ary tree with $n = 2$ is called a Binary Tree.





1-internal node $\rightarrow$ n leafs

2-internal node $\rightarrow$ $n + n-1 = 2n-1$ leaf

3-internal node $\rightarrow$ $n + n-1 + n-1 = 3n-2$ leaf

⋮

i-internal node $\rightarrow$ $i * n - (i-1) = i * n - i + 1 \Rightarrow i*(n-1)+1$ leafs

** No of leaf $\rightarrow$ $\boxed{L = i*(n-1)+1}$ ✓ , Total nodes = leaf node + Internal node

$$= i*(n-1)+1+i$$
$$= i*n - \cancel{i} + 1 + \cancel{i}$$

** Total nodes $\rightarrow$ $\boxed{N = i*n+1}$

$\boxed{\text{No of Leaf nodes on height } h = n^h}$

Q-1) A 3-ary tree is a tree in which every internal node has exactly 3-children. The no. of leaf nodes in such a tree with 6 internal node will be ?

a) 10

b) 23

c) 17

d) 13

3-ary

$n = 3$

$L = e * (n-1) + 1$

$L = 6 * (3-1) + 1 = 13$

Q-2) A Complete n-ary tree is a tree in which each node has n-children or no children. Let I be the no. of internal node and L be the no. of Leaves in a Complete n-ary tree. If L=41 and I=10, what is the value of n ___5___ ?

$$L = I * (n-1) + 1$$

$$41 = 10 * (n-1) + 1$$

$$(n-1) = \frac{41-1}{10} = 4$$

$$n = 4+1 = 5$$

Q.3) The no. of Leaf nodes in a rooted tree of n-nodes with each node having 0 or 3 Children is

a) $n/2$

b) $(n-1)/3$

c) $(n-1)/2$

d) $(2n+1)/3$ ✓

$$\cancel{L = e*(n-1)+1}$$

$$L = e*(3-1)+1$$

$$L = e*2+1$$

$$L = \frac{(n-1)}{3}*2+1$$

$$= \frac{2n-2+3}{3}$$

$$= \frac{2n+1}{3}$$

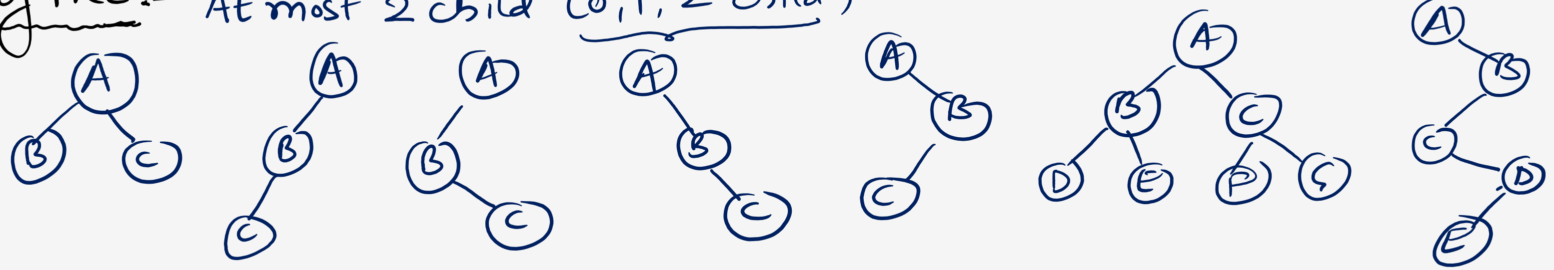
$$\boxed{N = n*e+1}$$

$$n = 3*e+1$$

$$e = \frac{n-1}{3}$$

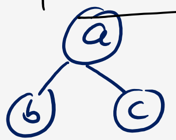
Binary Tree :- At most 2 child (0, 1, 2 child)

Ex:

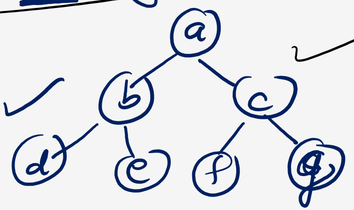


Nested Representation: (Root, Left, Right)

(a, NULL, NULL)



(a, b, c)



=> (a (bde) (cfg))

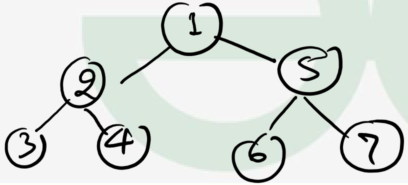
Ex: Which one of the following is valid Binary Tree.

a) (1 2 (4 5 6 7))

b) (1 ((2 3 4) 5 6) 7)

c) (1 (2 3 4) (5 6 7))

d) (1 (2 3 NULL) (4 5))



Memory Representation of Binary Tree:

- Array
- Linked list

A	B	–	–	C	–	–
1	2	3	4	5	6	7

Array Representation:-

Algo:

Parent(i)
- return $\lfloor i/2 \rfloor$

left(i)
- return $2i$

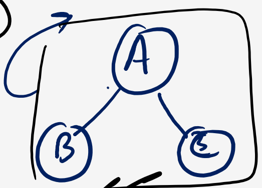
Right(i)
- return $2i+1$

BT

A	B	C
1	2	3

$i \rightarrow$ 1 2 3

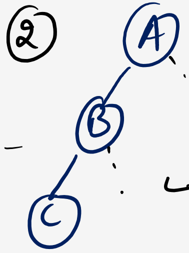
(1)



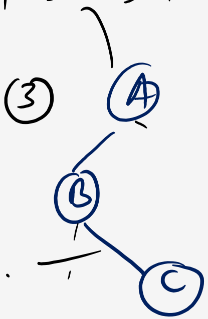
BT

A	B	-	C	-	-	-
1	2	3	4	5	6	7

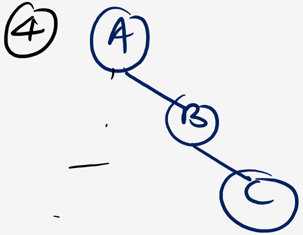
(2)



(3)

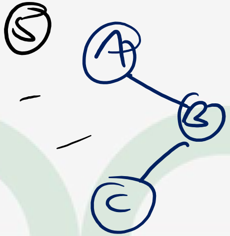


(4)



A	–	B	–	–	–	C
1	2	3	4	5	6	7

(5)

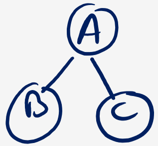


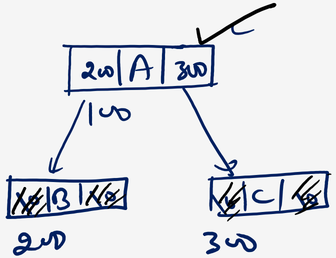
A	–	B	–	–	C	–
1	2	3	4	5	6	7

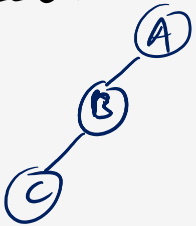
NOTE: if n search keys then

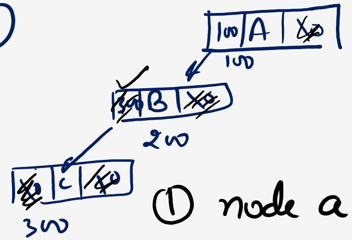
|Array| <
- $\boxed{max = 2^n - 1}$
- $\boxed{min = n}$

Linked list Representation:-









```
Struct BTnode
{ Struct BTnode *left;
  int data;
  Struct BTnode *right;
};

Typedef Struct BTnode node;
```

① node a = {NULL, 10, NULL};

② node b = {NULL, 20, NULL};

③ node C = {NULL, 30, NULL};

⑤ node *t = NULL

t = &a;

⑥ a.left = &b;

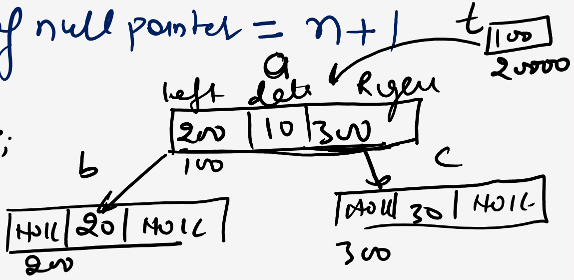
⑦ a.right = &c;

⑧ -

NOTE:

1) If no. of keys n Then maximum and minimum no. of nodes $= n$

2) No. of null pointer $= n+1$



1) Pf(t) = ? 100

2) Pf(t->data) = 10

3) Pf(t->left) // 200

4) if (t==NULL) (or) if (!t) // empty tree

5) if ((t->left==NULL)&&(t->right==NULL))
(or)
if(!(t->left) && !(t->right))
} leafnode

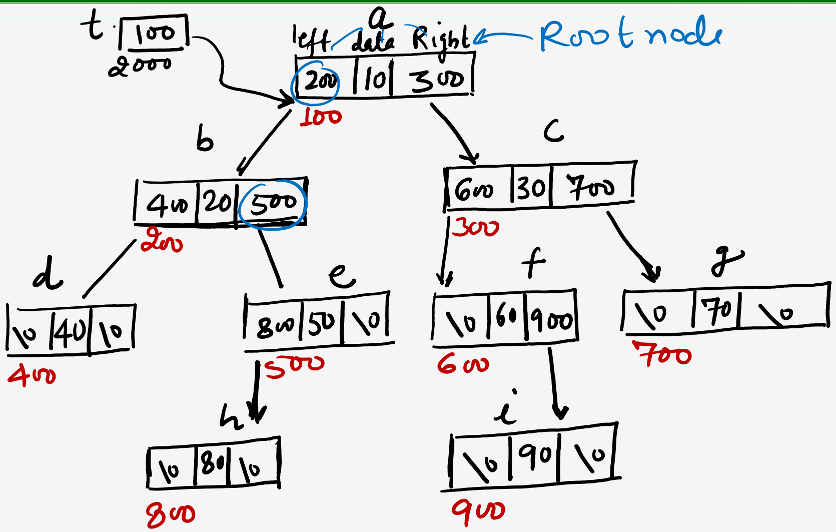
6) (t->left) is given ? it means Left Subtree is given

7) (t->right) is given ? Right Subtree is given

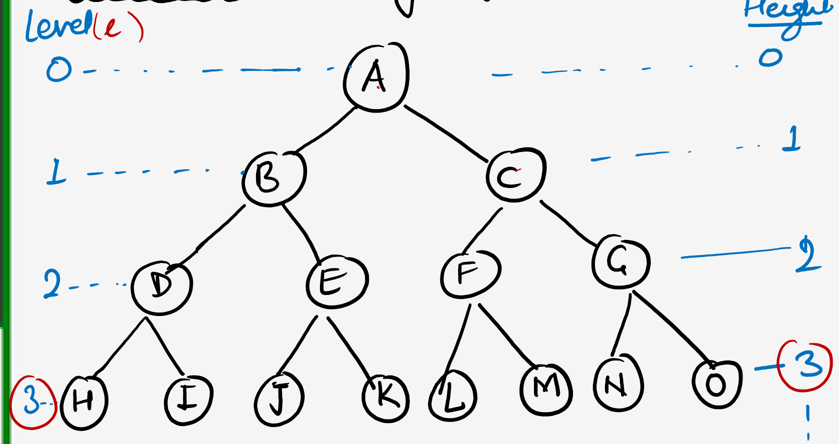
8) Pf(t->Left->right->data) = ? 50 Ans
200 -> 500 ->

9) Pf(t->right->left->right->data) = ?
300 -> 600 -> 900 -> ⇒ 90

10) Pf(t->Left->left->Left->data) = ? Error
200 -> 400 -> 10 ->



Relationship among height, total nodes, leaf nodes and Internal nodes :-



Height	Leaf node		Internal nodes		Total nodes
0	2^0	+	0	=	1
1	2^1	+	1	=	3
2	2^2	+	3	=	7
3	8	+	7	=	15
h	$\boxed{L = 2^h}$		$\boxed{I = 2^h - 1}$		$\boxed{N = 2^{h+1} - 1}$

Height $\longrightarrow$ h

$$\boxed{N_{max} = 2^{h+1} - 1}$$

$$h+1 = \log_2(N+1)$$

$$\boxed{h = \log_2(N+1) - 1}$$

$h=0$ A

$min = 0+1 = 1$

0 – A

1 – B

$min = 1+1 = 2$

Properties of Binary Tree :-

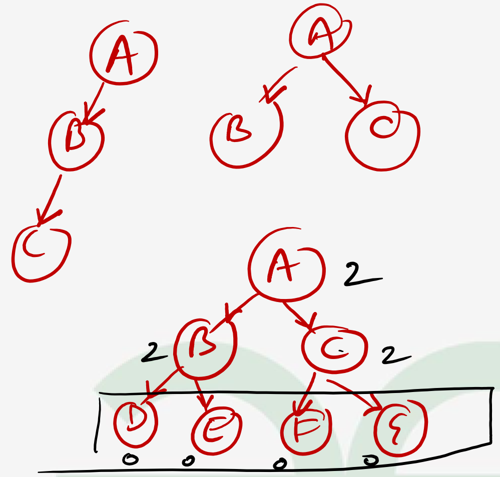
- At each Level of i the maximum No. of nodes = 2^i
- Maximum No. of nodes at height h = $\boxed{2^{h+1}-1}$ ⇒ N_{max}
- Minimum No. of nodes Possible at height h = $\boxed{h+1 = N_{min}}$

$$\boxed{\underbrace{h+1}_{\text{Min Nodes}} \le N \le \underbrace{2^{h+1}-1}_{\text{Max Nodes}}}$$

$$\boxed{\underbrace{\log_2(N+1)-1}_{\text{Min height}} \le h \le \underbrace{N-1}_{\text{max height}}}$$

**

- $\boxed{n_0 = n_2+1}$



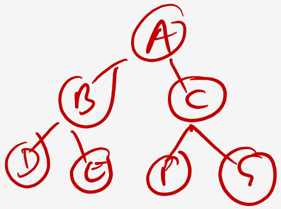
Question: In a Binary tree, the number of internal nodes of degree 1 is 5 and no. of internal node of degree 2 is 10 The number of Leaf nodes in the Binary tree?

$$n_0 = n_2 + 1$$

$$n_0 = 10 + 1 = 11$$

Types of Binary Tree:

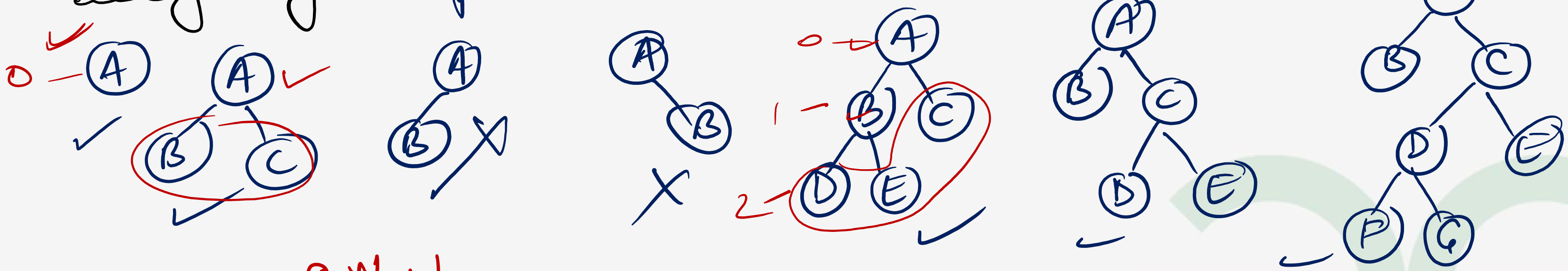
- Full / Proper / Strictly Binary Tree
- Complete Binary Tree
- Perfect Binary Tree



$2^{h+1}-1$

$2n-1$

Strictly Binary Tree:- each node must either <u>0 or 2</u> children



$2*h+1$

$2^{h+1}-1$

Properties of Strictly Binary tree:-

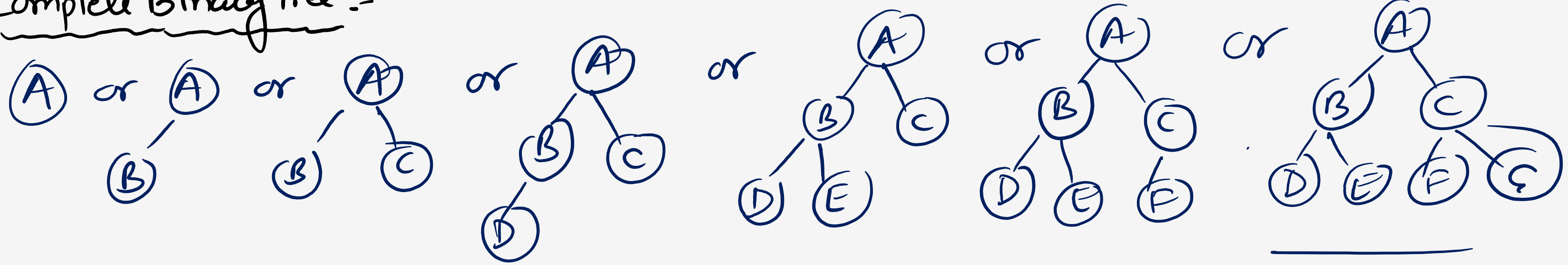
1) The no. of leaf nodes is equal to the number of internal nodes plus 1
2) If leaf nodes n then total no. of nodes = $2n-1$ (odd)
3) Strictly Binary tree have odd No. of nodes
4) Min No. of nodes of height h = $\boxed{2*h+1}$
5) Max No. of nodes = $\boxed{2^{h+1}-1}$

$$\underbrace{2h+1}_{\text{min Nodes}} \leq N \leq \underbrace{2^{h+1}-1}_{\text{max Nodes}}$$

6) $\boxed{\text{Min height} = \log_2(N+1)-1}$

$\boxed{\text{Max height} = \frac{N-1}{2}}$

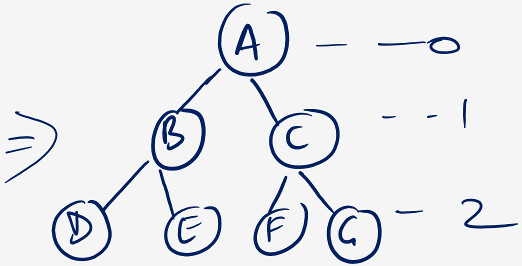
Complete Binary Tree :-



Properties of Complete Binary Tree:-

- max No of nodes in Complete Binary Tree $= 2^{h+1}-1$
- min No of nodes " $= 2^{h}$
- min height $= \log_2(N+1)-1$
- max height $= \log_2 N$

Perfect Binary Tree:-



NOTE:

Number of Binary Tree :- if n number of Keys or nodes then

No. of Unlabeled Binary Tree $= \dfrac{{}^{2n}C_n}{n+1}$ $= 5$

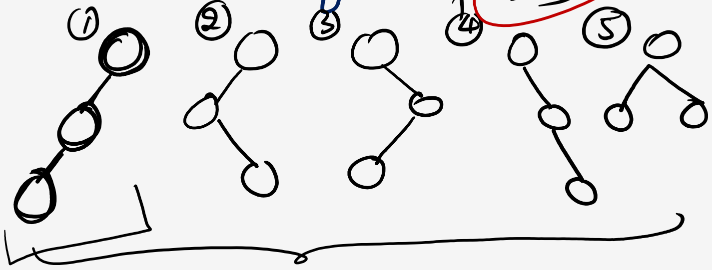
No of Labeled Binary Tree $= \dfrac{{}^{2n}C_n}{n+1} * n!$ $= 5 * 3! = 30$

$n = 4$

$= 14$

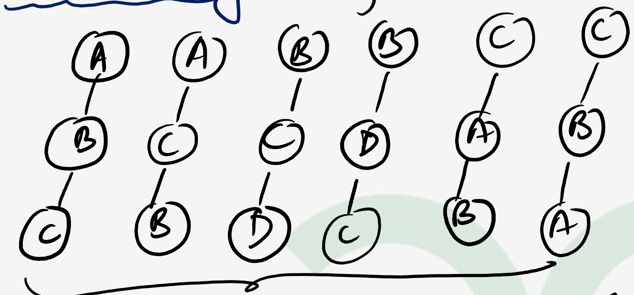
$= 14 * 4!$

Unlabeled Binary Tree of $n = 3$



5 Unlabeled Binary Tree.

Labeled Binary Tree of $n = 3$



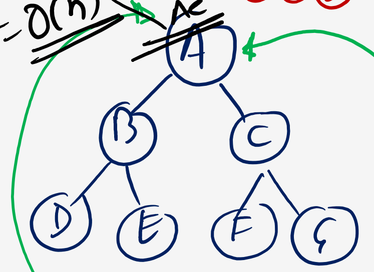
Single structure filled by $3!$ ways, Therefore

Total no. of labeled Binary Tree $= n! *$ Unlabeled Binary Tree

Traversal of Binary Tree:-

- Inorder (L N R) ① ② ③
- Preorder (N L R) ① ② ③
- Postorder (L R N) ① ② ③

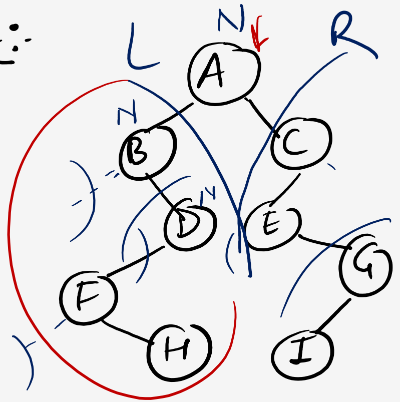
T = O(n) BC, WC, AC



IN: D B E A F C G

Pre: A B D E C F G

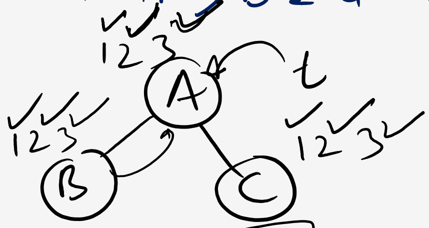
Post: D E B F G C A



IN: B F H D A E I G C

Pre: A B D F H C E G I

Post: H F D B I G E C A



B A C

T(n) — Algo: void Inorder(Struct node *t)

{ if(t)

{

T(n/2) — ① Inorder(t→left); L

② printf("%c", t→data); N

T(n/2) — ③ Inorder(t→right); R

}

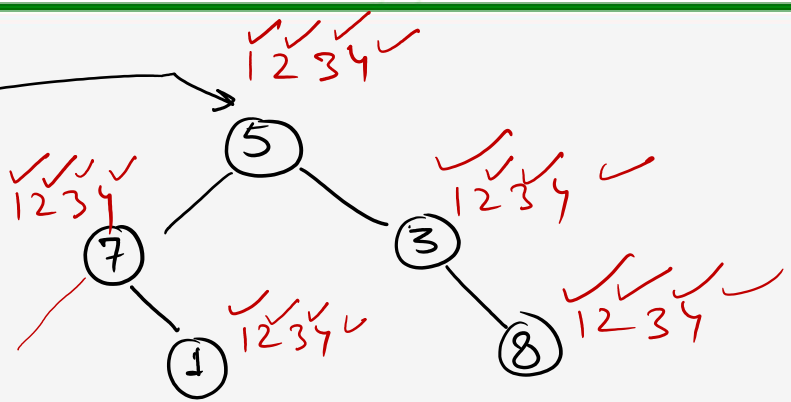
}

$T(n) = 2T(n/2) + 1$

$= O(n)$

Double order traversal:-

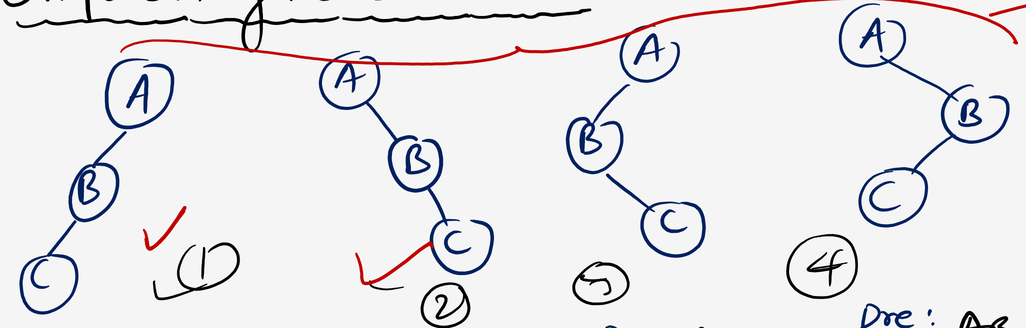
```
DO(node *t)
{ if(t)
  { ① pf(t->data);
    ② DO(t->left);
    ③ pf(t->data);
    ④ DO(t->right);
  }
}
```



O/p = 5 7 7 1 1 5 3 3 8 8

Unique Binary Tree Construction:-

Not Unique Binary Tree if
Pre: ABC
Post: CBA



	(1)	(2)	(3)	(4)
Pre	ABC	ABC	ABC	ABC
Post	CBA	CBA	CBA	CBA
IN	CBA	ABC	BCA	ACB

$n = 3$

NOTE: if preorder & Postorder given and n search keys Then No. of Binary trees $= 2^{n-1}$

NOTE: 1) if Preorder & Postorder given then Unique Binary tree not Possible

2) if Preorder & Inorder (or) Postorder & Inorder given then Unique Binary

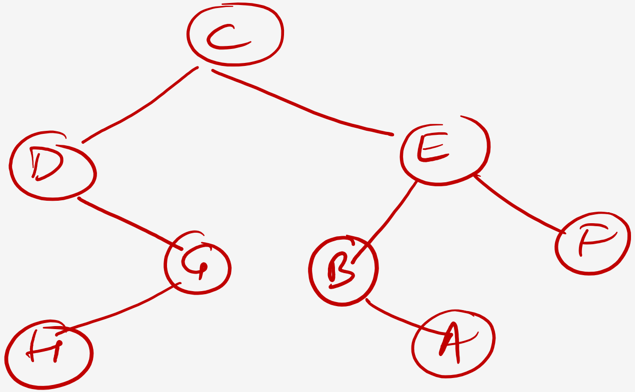
Pre : C D G H E B A F

IN : D H G C B A E F

Post: ?

Pre: NLR

IN: LNR



Postorder = H G D A B F E C

Question:

IN: H F G I J K B A E C D

Post: G F H I K E A C B D J

Pre ?

TC: $\boxed{O(n^2)}$

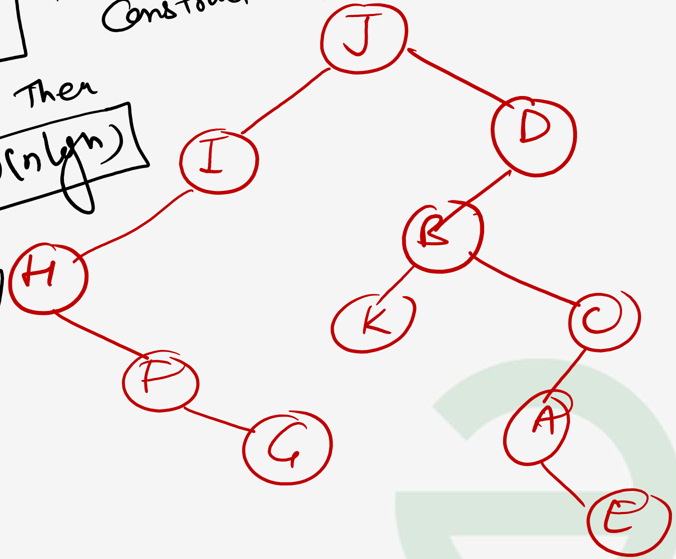
for Unique Binary Tree Construction

if Inorder sorted Then

$\boxed{TC = O(n \log n)}$

Post: L R N

IN: L N R



Preorder:- JIHFGDBKCAE

TC: Pre: A B C D E

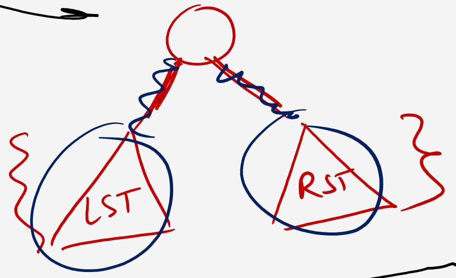
IN: E D C B A

n
$n-1$
$n-2$
$\vdots$
1

$\Rightarrow \frac{n(n+1)}{2} \Rightarrow O(n^2)$

Height of tree:-

$$\Rightarrow H(T) = \begin{cases} 0, & \text{if } T \text{ is empty} \\ 0, & \text{if } T \text{ is leaf} \\ 1 + \max(H(LST), H(RST)), & \text{otherwise} \end{cases}$$




```
int H(Struct node *t)
{ if(!t)          // if(t==NULL)
      return 0;
   if(!(t->left) && !(t->right))   // ((t->left==NULL) && (t->right==NULL))
        return 0;
   int l,r;
   l= H(t->left);
   r= H(t->Right);
   return (1+ (l>r?l:r));
}
```

O

$\boxed{1 + \max(H(LST), H(RST))}$

$\Rightarrow O(\log n)$

Count No. of Nodes :-

$$NN(T) = \begin{cases} 0, & \text{if } T \text{ is NULL} \\ 1 + NN(LST) + NN(RST), & \text{otherwise} \end{cases}$$

```
int NN(struct node *t)
{ if (t)
   { int l, r;
     l = NN(t->left);
     r = NN(t->right);
     return (1+l+r);
   }
  return 0;
}
```

Count No. of leaf nodes:

```
int LN(struct node *t)
{
  if(t == NULL)
      return 0;
  if((t->left == NULL) && (t->right == NULL))
      return 1;
  else
      return (LN(t->left) + LN(t->right));
}
```

$$LN(T) = \begin{cases} 1, & \text{if } T \text{ is leaf} \\ LN(LST) + LN(RST), & \text{otherwise} \end{cases}$$

Count No of nonLeaf (internal) nodes:

$$NL(T) = \begin{cases} 0, & \text{if } T \text{ is NULL} \\ 0, & \text{if } T \text{ is leaf} \\ 1 + NL(LST) + NL(RST), & \text{otherwise} \end{cases}$$

```
int NL(struct node *t)
{ if (t==NULL) return 0;
    if ((t->left==NULL) && (t->right==NULL))
        return 0;
  else return (1+ NL(t->left) + NL(t->right));
}
```

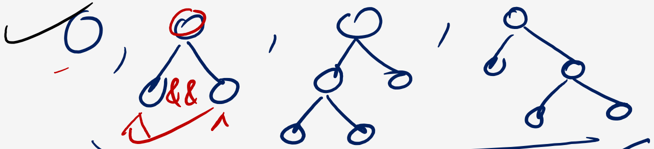
Count full nodes (2 child nodes):-

$$FN(T) = \begin{cases} 0, & \text{if } T = NULL \\ 0, & \text{if } T \text{ is leaf} \\ 1 + FN(LST) + FN(RST), & \text{if } T \text{ is full node} \\ FN(LST) + FN(RST), & \text{otherwise} \end{cases}$$

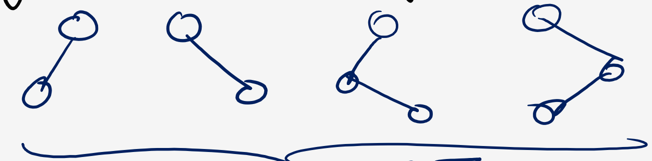
```
int FN(struct node *t)
{ if (!t) return 0;
    if (!(t->left) && !(t->right))
        return 0;
    return (FN(t->left) + FN(t->right) + ((t->left) && (t->right) ? 1 : 0));
}
```

Count single child nodes:-

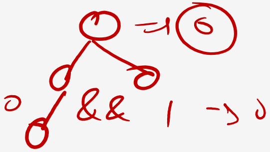
WAP to check whether the given Binary tree is Strict Binary Tree or not :-



Strictly BT



× not strictly BT



Soln.

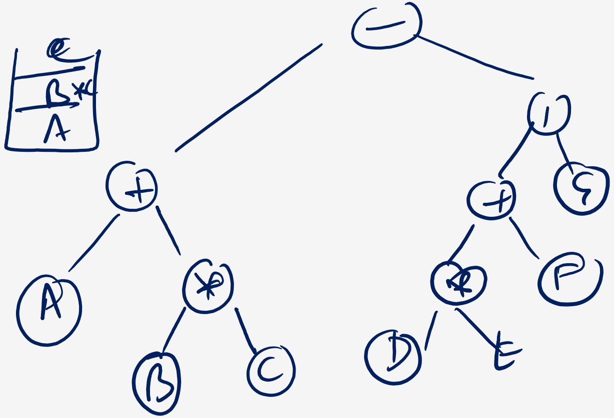
```
int SBT(struct node *t)
{ if(t==NULL)
      return 1;
   if((t->left==NULL)&&(t->right==NULL))
      return 1;
   if((t->left!=NULL)&&(t->right!=NULL))
      return((SBT(t->left))&&(SBT(t->right)));
   else
      return 0;
}
```

Expression Tree:

Internal nodes → operators

leaf node → operands

ABC*+DE*F+G/−



Question: Consider new order Strategy for traversing a Binary Tree

- Visit the root N
- Visit the right Subtree R
- Visit the Left Subtree N

$\Rightarrow$ $\boxed{NRL}$

The new order traversal for the expression $\boxed{34*5-2\wedge 67*1+-}$

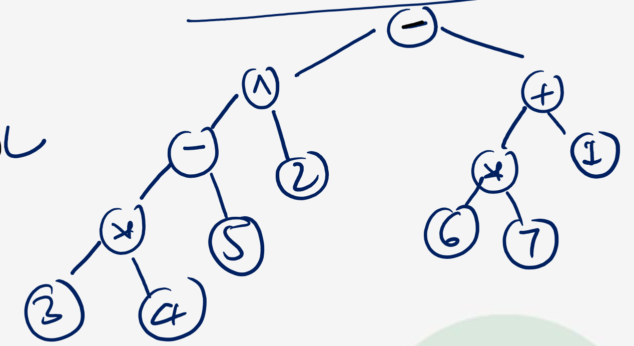
a) $+-167*2\wedge 5-34*$

b) $-+1*67\wedge 2-5*34$

c) $-+1*76\wedge 2-5*43$

d) $176*+2543*-\wedge-$

NRL



$-+1*76\wedge 2-$

Binary Search tree (BST) :-

No. of Binary search trees :-

Binary Search Tree Construction :-

Question: A binary search tree is generated by inserting in order of the following integers 50, 15, 62, 5, 20, 58, 91, 3, 8, 37, 60, 24, The no. of nodes in the Left Subtree and right Subtree of the root respectively is ___

a) (4, 7)

b) (7, 4)

c) (8, 3)

d) (3, 8)

Question:- How many distinct Binary search trees can be constructed out of 4-distinct keys

a) 5

b) 14

c) 24

d) 35

Question:- The numbers 1, 2, ... n are inserted in a binary search tree in some order. In the resulting tree, the right subtree of the root contain P nodes. The first number to be inserted in the tree must be

a) P

b) P+1

c) n-P

d) n-P+1

Question:- We are given a set of n-distinct elements and an unlabeled binary tree with n-nodes. In how many ways can we populate the tree with given set so that it becomes a binary search tree?

a) 0

b) 1

c) $n!$

d) $\dfrac{{}^{2n}C_n}{n+1}$

Binary search tree operation:

- Search
- Insert
- Delete

Question: Preorder of BST = 30, 20, 10, 15, 25, 23, 39, 35, 42

Postorder Traversal = ?

a) 10, 20, 15, 23, 25, 35, 42, 39, 30

b) 15, 10, 25, 23, 20, 42, 35, 39, 30

c) 15, 20, 10, 23, 25, 42, 35, 39, 30

d) 15, 10, 23, 25, 20, 35, 42, 39, 30

Question:- (1-100) in BST search for 55 which of the following sequence can not be the sequence of nodes examined?

a) {10, 75, 64, 43, 60, 57, 55}

b) {90, 12, 68, 34, 62, 45, 55}

c) {9, 85, 47, 68, 43, 57, 55}

d) {79, 14, 72, 56, 16, 53, 55}

Thank You !