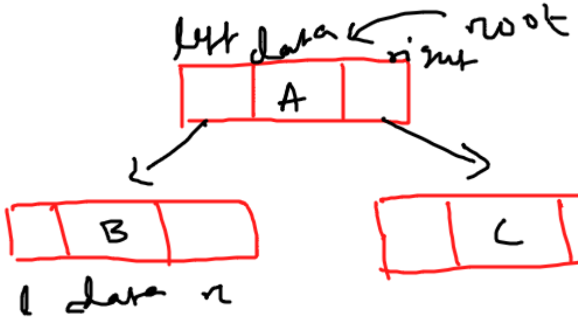
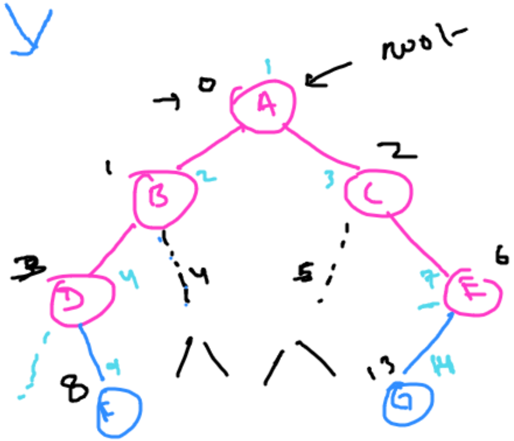


← LL Representation

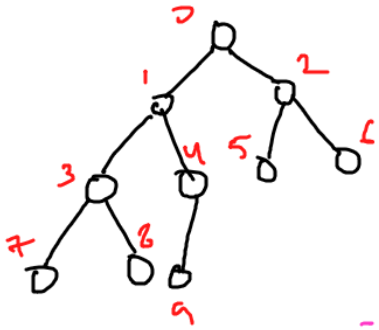


A	B	C	D			E
0	1	2	3	4	5	6

Index of a given node is 'i' :

$L_i = 2*i+1$ $R_i = 2*i+2$

$P_i = (i-1)/2$



Index

0 → 4 → Internal Node Index

5 → 9 → Leaf Node Index

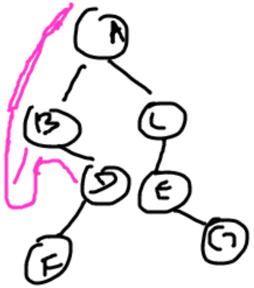
10 = N

Internal Nodes Indices = 0 to $\frac{N}{2} - 1$

Leaf Nodes Indices = $N/2$ to $N-1$

Traversal

1st time → Preorder
2nd time → In order
3rd time → Post order.



A B D

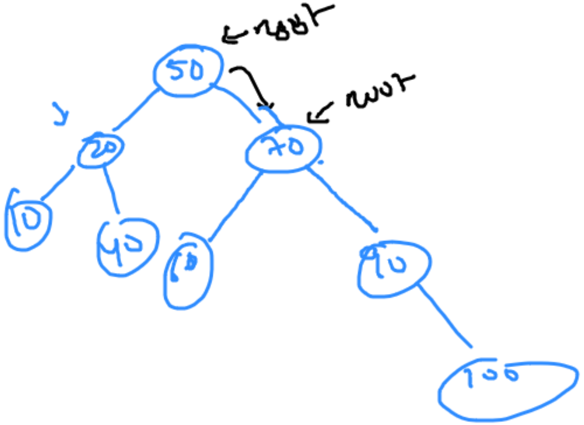
```
void inorder (node root)
{   if(root)
    {
    1. inorder (root → left);
    2. s.o.p (root → data); ✓
    3. inorder (root → right);
    }
}
```

```
void preorder (node root)
{
    if(root)
    {
    1. s.o.p (root → data); ✓
    2. preorder (root → left); ✓
    3. preorder (root → right); ✓
    }
}
```

```
if(root)
{   postorder (root → left)
    postorder (root → right)
    s.o.p (root → data)
}
```

Binary Search Tree - BST

Left < Root < Right ←



BST Construction (Insert operation)

50, 70, 60, 20, 90, 10, 40, 100

To construct a unique binary tree, we need :

(1) Preorder and inorder

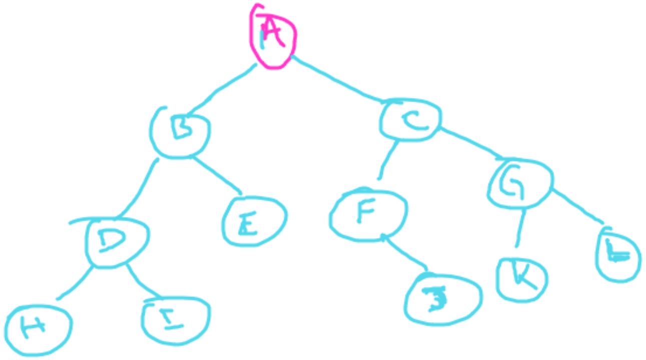
(or)

(2) Postorder and inorder

(1) Preorder : A B D H I E C F J G K L

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

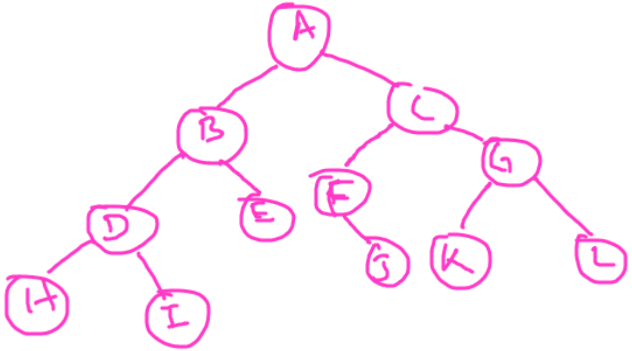
construct Binary tree .



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

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

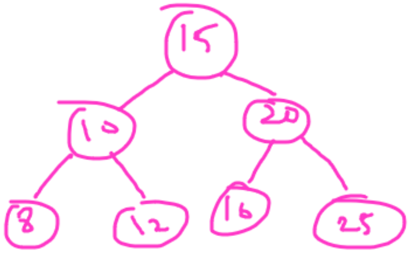
Construct binary tree.



preorder :

: 15, 10, 8, 12, 20, 16, 25 ⇒ Inorder : 8, $\boxed{10}$, 12, $\boxed{15}$, 16, $\boxed{20}$, 25

Given preorder traversal, construct unique BST.



Note :-

Inorder traversal of BST is the ascending order of elements.

left < root < right