Date _____ Page _____

Day - 157

Trees - 7

* Construct a tree from inorder and preorder:

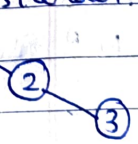
In	4	2	8	5	9	1	6	3	7	10

Pre	1	2	4	5	8	9	3	6	7	10

- Our first question can be —
- Can we create a tree by using pre-order only?
- So, we can't create a unique tree only by using preorder.
- In the same way, we can't create a unique tree by using inorder or postorder alone.

- So, by using —
  Preorder & inorder &
  Postorder & inorder
- we can create a unique tree.
- But we can't — Preorder & Postorder.

Ex: Pre: 1 2 3
In: 1 2 3



Date _____ Page _____

⇒ We know that, in preorder — N L R.

⇒ So, we ~~kn~~ don't know that, after selecting node, we can't decide that the second element is left or right child.

⇒ For that, we will require, inorder.

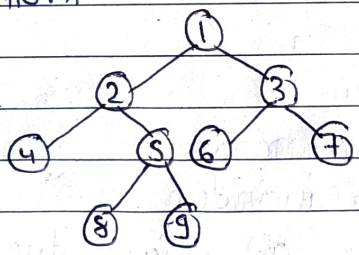
⇒ Inorder → L N R

⇒ So, after identifying 'Node' from preorder, we can easily identify left & right from inorder.

Preorder → N L R
Inorder → L N R

⇒ So, preorder helps in knowing element & inorder ~~tells~~ helps in knowing their position.

Ex:

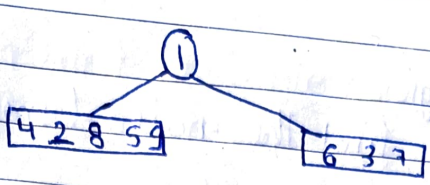


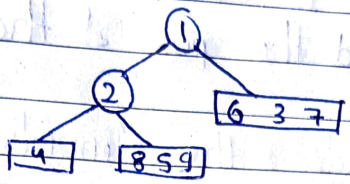
N L R Pre: 1 2 4 5 8 9 3 6 7

L N R In: 4 2 8 5 9 1 6 3 7

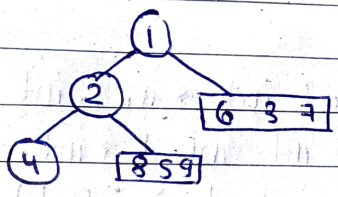
L N R

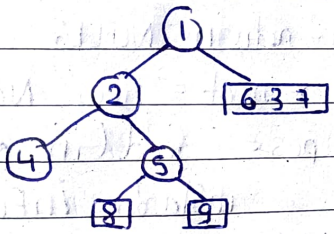
Date _____ Page _____

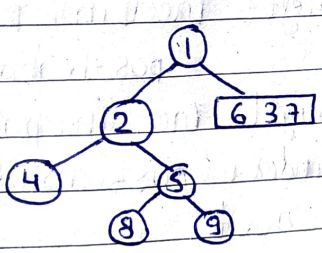


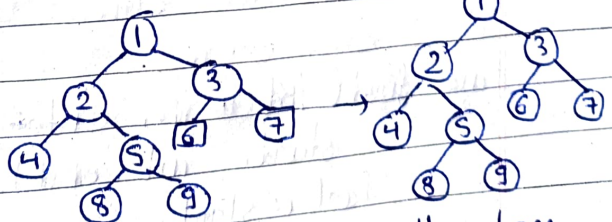


~~And so on ...~~









Elements should be unique in the tree.

Date _____ Page _____

Code Logic

⇉ First, find the node element from preorder.

⇉ Then, find the pos of that element in order.

⇉ And divide the elements in left & right according to inorder.

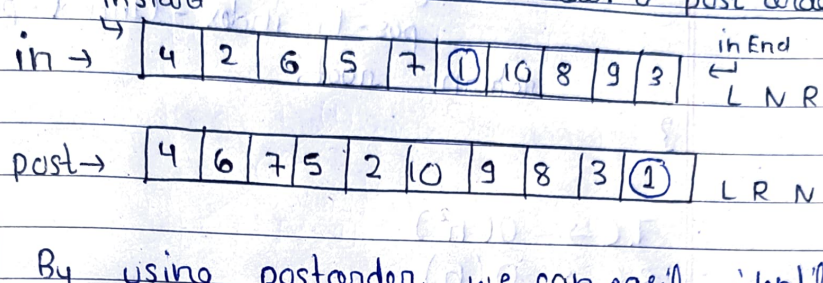
Code

```
Node * Tree( int *in , int *pre, int
  inStart, int inEnd , int index){
      if( inStart > inEnd)
          return NULL;
      Node * root = new Node( Pre[index]);
      int pos = find(in, pre[index],
              inStart, inEnd);
      root->left = Tree( in, pre, inStart,
                  pos-1, index+1);
      root->right = Tree(in, pre, pos+1,
          index+(pos-inStart)+1);
      return root;
}

int find( int * in, int * target, int
      start, int end){
    for( i=start; i<= end; i++){
        if( in[i] == target)
            return i;
    }
    return -1;
}
```

Date _____ Page _____

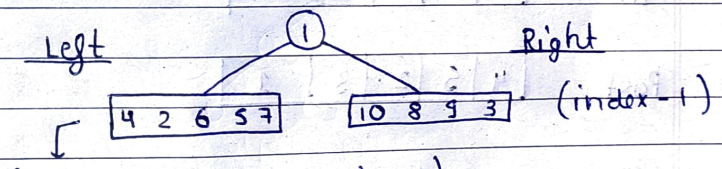
* Construct a Tree from inorder & post order.



⇉ By using postorder, we can easily identify root node i.e. 1.

⇉ We will first see the right side then left side.

⇉ Also, we will see from the end of the inorder.



(index - (InEnd - pos) - 1)

Code

```
Node * Tree( int * in, int *~~pre~~ post, int
  inStart, int inEnd, int index){
      if( inStart > inEnd)
          return NULL;
      Node * root = new Node( post[index]);
      int pos = find ( in, post[index],
          inStart, inEnd);
      root→ right = Tree( in, post, pos+1,
      InEnd, index - (~~-~~1));
```

Date _____ Page _____

```
    root->left = Tree(in, post, inStart,
                      pos-1, index-(InEnd-pos)-1);
    return root;
}
```

T.C. → $O(n^2)$
S.C. → $O(n)$

* Check Tree Traversal:

Pre	1	2	4	5	3
In	4	2	5	1	3
Post	4	5	2	3	1

⇒ We have to check that all the given tree traversal are of one tree.

⇒ For this, we can create two trees by using in & pre and in & post.

⇒ Then, we can check both the trees are same or not.

⇒ If same return 1 otherwise 0.