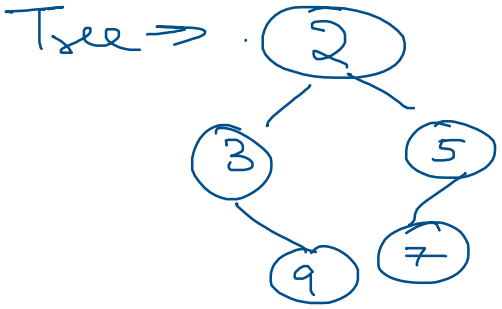
Binary Trees, July 29

29 July 2020 19:07

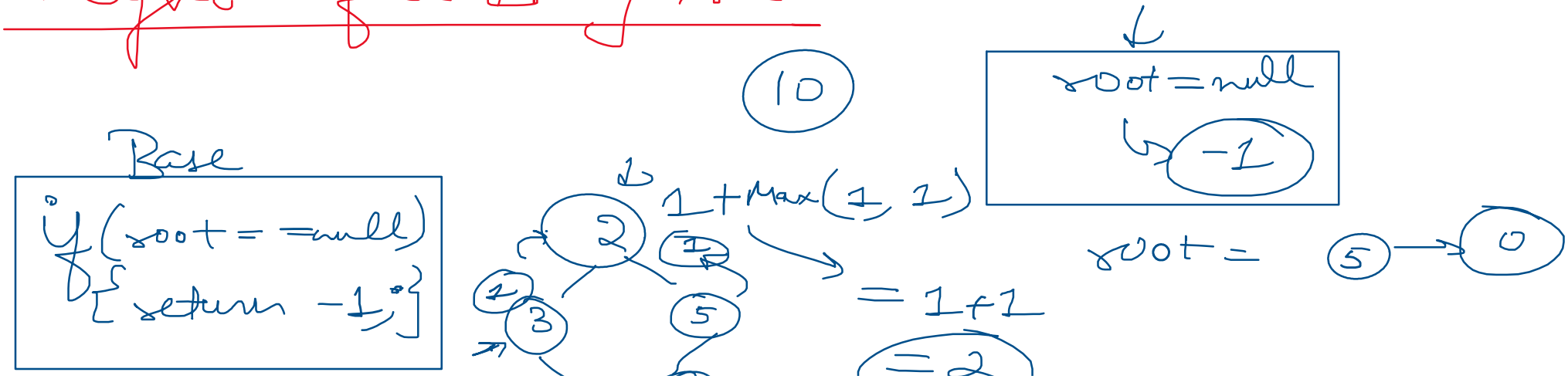
1. Height of a Binary Tree //Done
2. Print elements at a particular level //Done
3. Traversal of Tree - DFS(PreOrder, InOrder, PostOrder) //Done
4. Check if two trees are identical //Done
5. Check if two trees are mirror of each other //Done
6. Convert a tree into its mirror //Done
7. Check if two trees have the same structure //Done
8. Check if two trees are IsoMorphic
9. BFS -> Breadth First Search

Tasks:

1. Check if two trees are IsoMorphic(Khud se code krna)
2. BFS -> Breadth First Search(Khud se sochna)



Height of a Binary Tree



1 + Max(-1, 0)

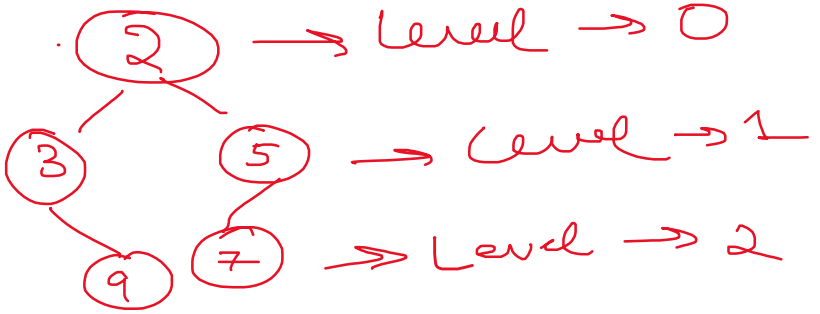
9 7

$$1 + \text{Max}(\text{Height}(LST), \text{Height}(RST))$$

$$1 + \text{Max}(\text{Height}(\text{null}), \text{Height}(\text{null}))$$

$$1 + \text{Max}(-1, -1) \rightarrow 1 + (-1) = 0$$

Print elements at a particular level

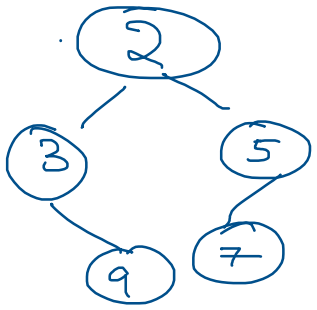


printAtLevel (Node root, int level)

```
{   if(root == null) return;
    if (level == 0)
    { S.O.P(root.data);
      return;
    }
    printAtLevel(root.left, level-1);
    printAtLevel(root.right, level-1);
```

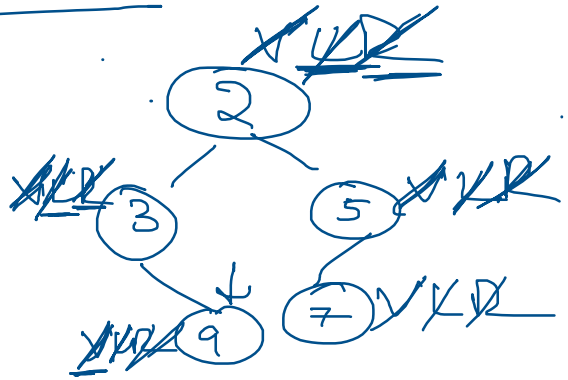
Traversals

DFS → Depth First Search



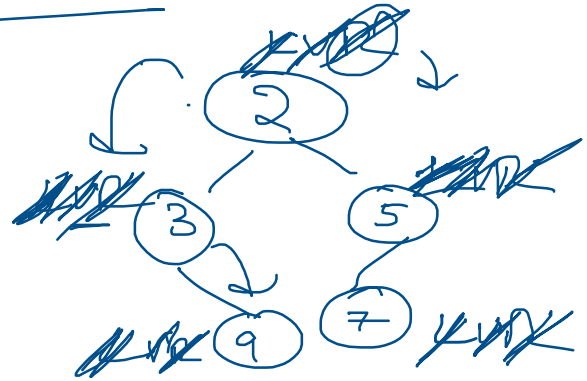
3 ways

Pre-Order → Value Left Right (VLR)



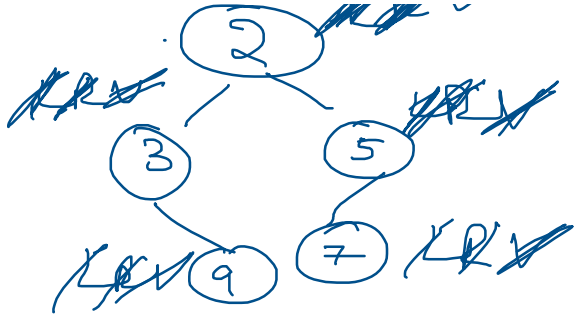
Print →
2 3 9 5 7

In-Order → Left Value Right



Print
3 9 2 7 5

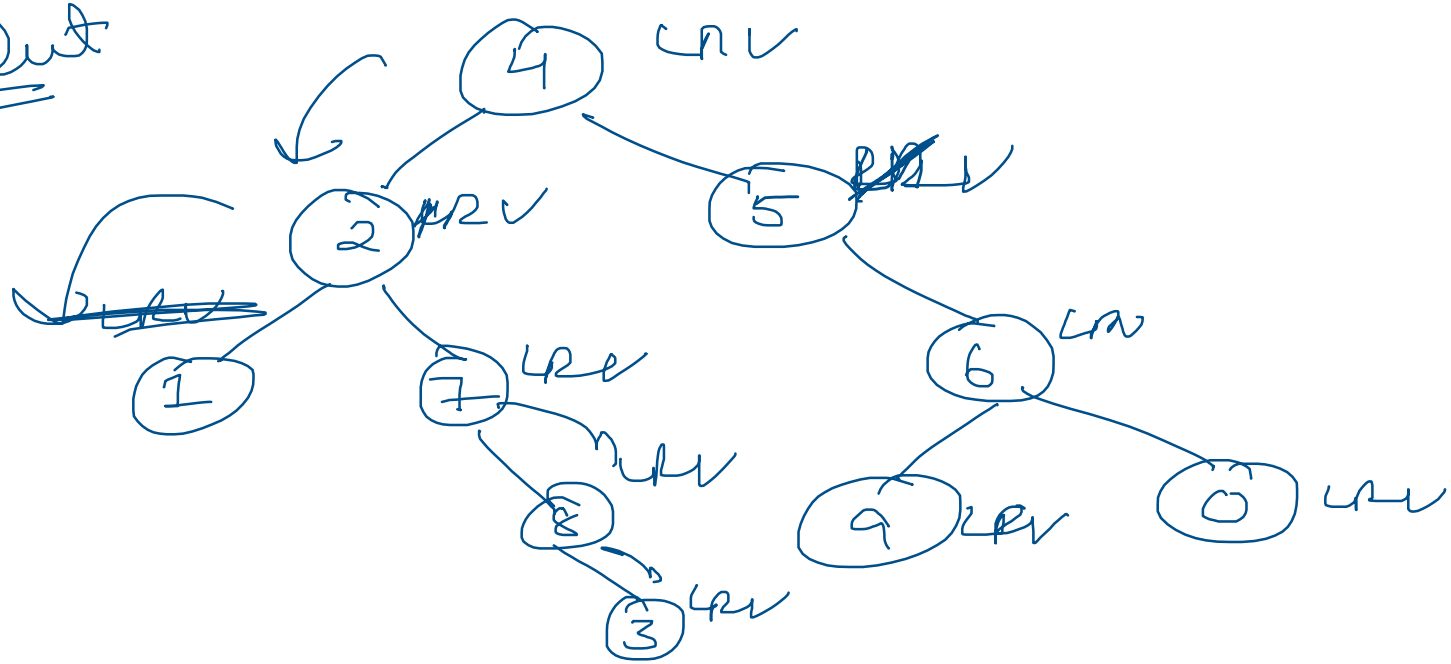
Post-Order → Left Right Value



Print

9 3 7 5 2

Try it Out

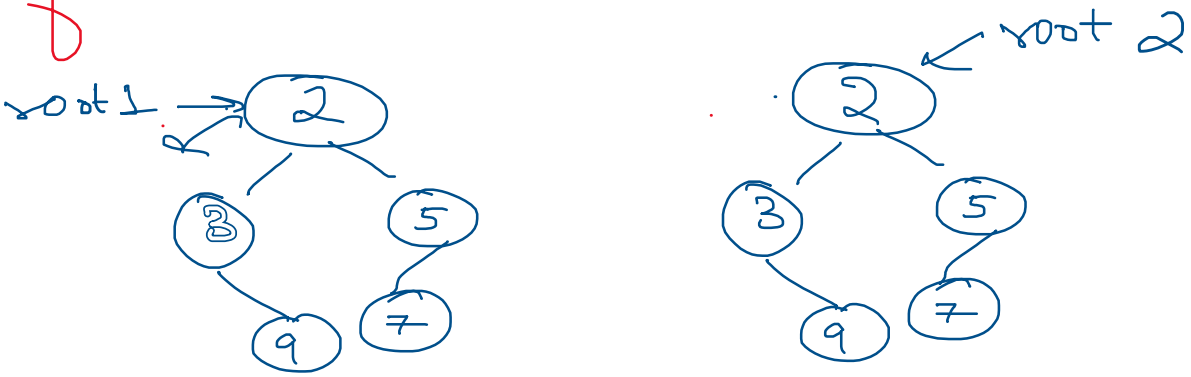


PreOrder → 4 2 1 7 8 3 5 6 9 0

Inorder → 1 2 7 8 3 4 5 9 6 0

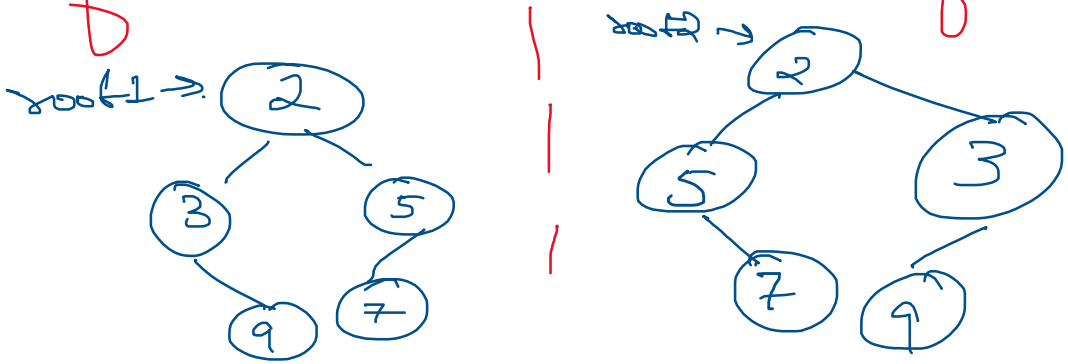
PostOrder → 1 3 8 7 2 9 0 6 5 4

Check if 2 trees are identical



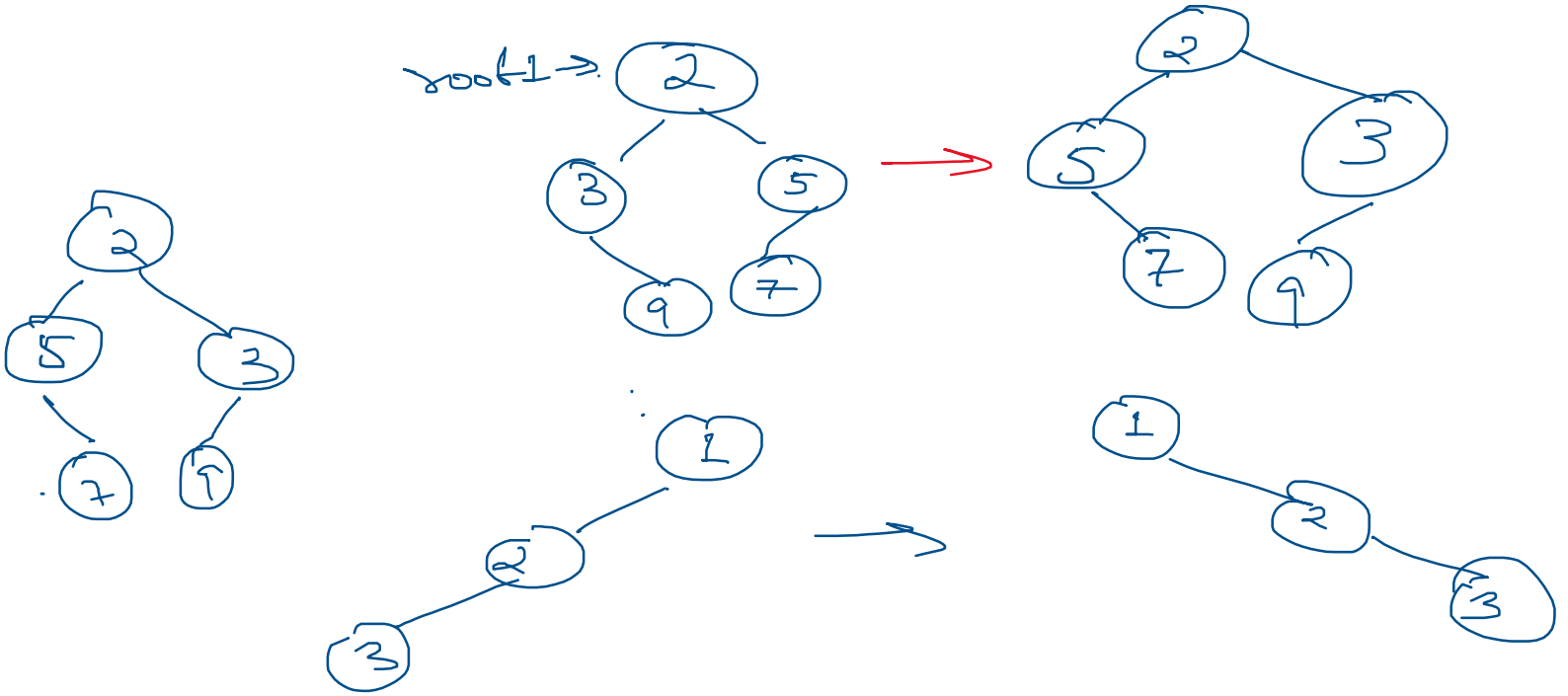
If Both roots are null then identical.
If any 1 of them is null then not identical.
Check data of both the roots
Check LST's
Check RST's

Check if 2 trees are Mirror of each other



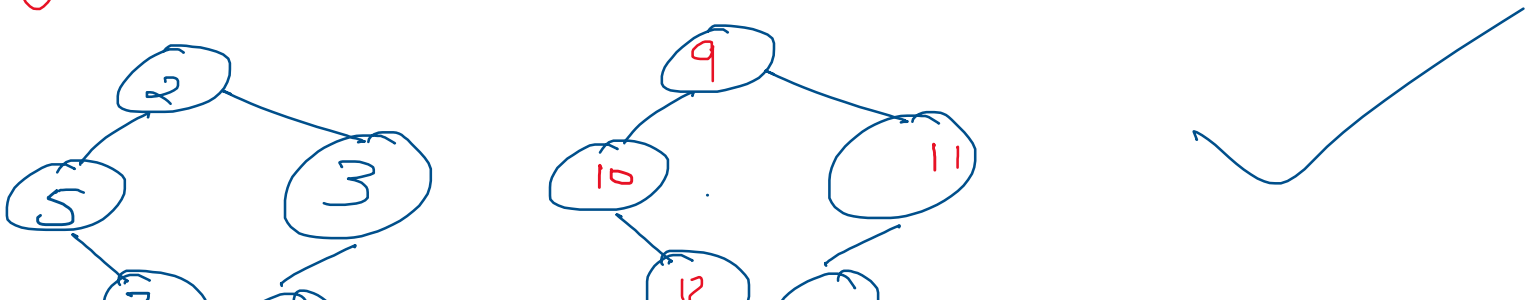
If Both roots are null then Mirror.
If any 1 of them is null then not Mirroe.
Check data of both the roots
Check LST of 1st tree && RST of 2nd tree, they must be mirror
Check LST of 2nd tree && RST of 1nd tree, they must be mirror

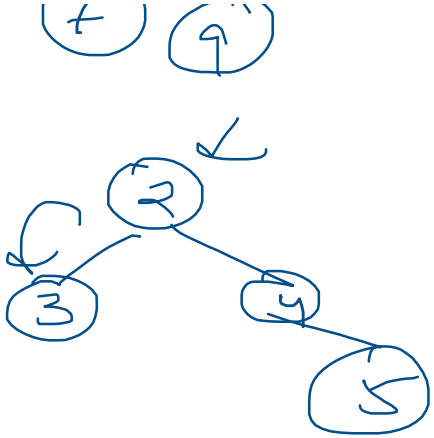
Convert a tree into its mirror.

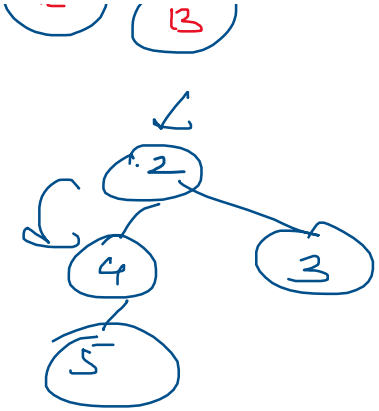


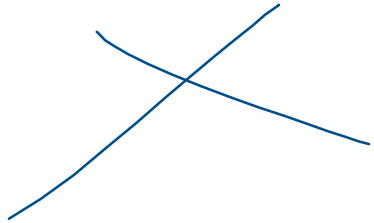
For each Node Swap LST && RST

Check if 2 trees have same structure.



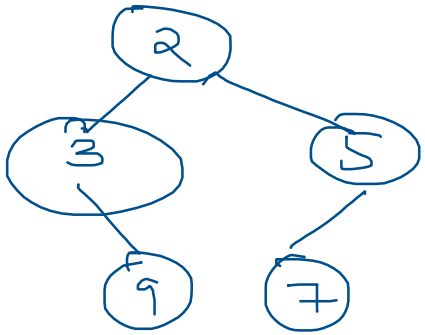


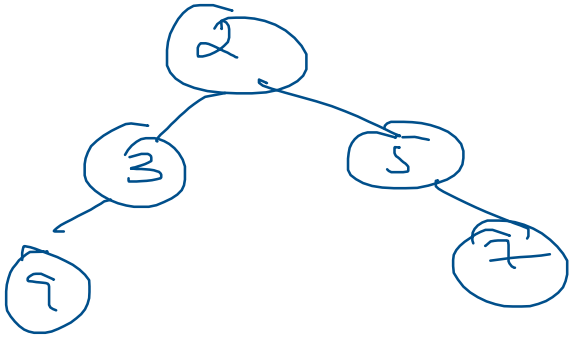


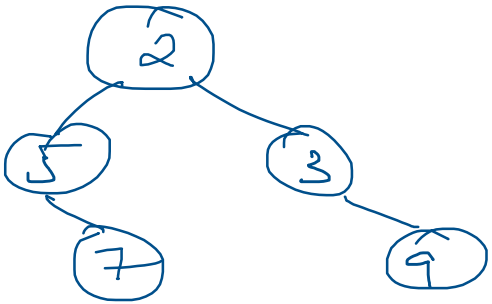


If Both roots are null then Symmetric.
If any 1 of them is null then not Symmetric.
Check LST's
Check RST's

IsoMorphic







IsoMorphic ->For every Node, either the children are identical or they can be mirror to each other.

(Data being the same)