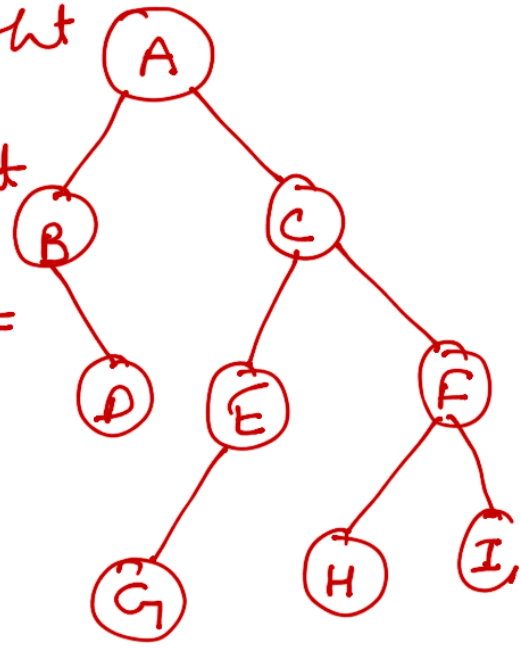
Tree Traversal

Inorder – Left, Root, Right

Preorder – Root, Left, Right

Postorder – Left, Right, Root



Inorder

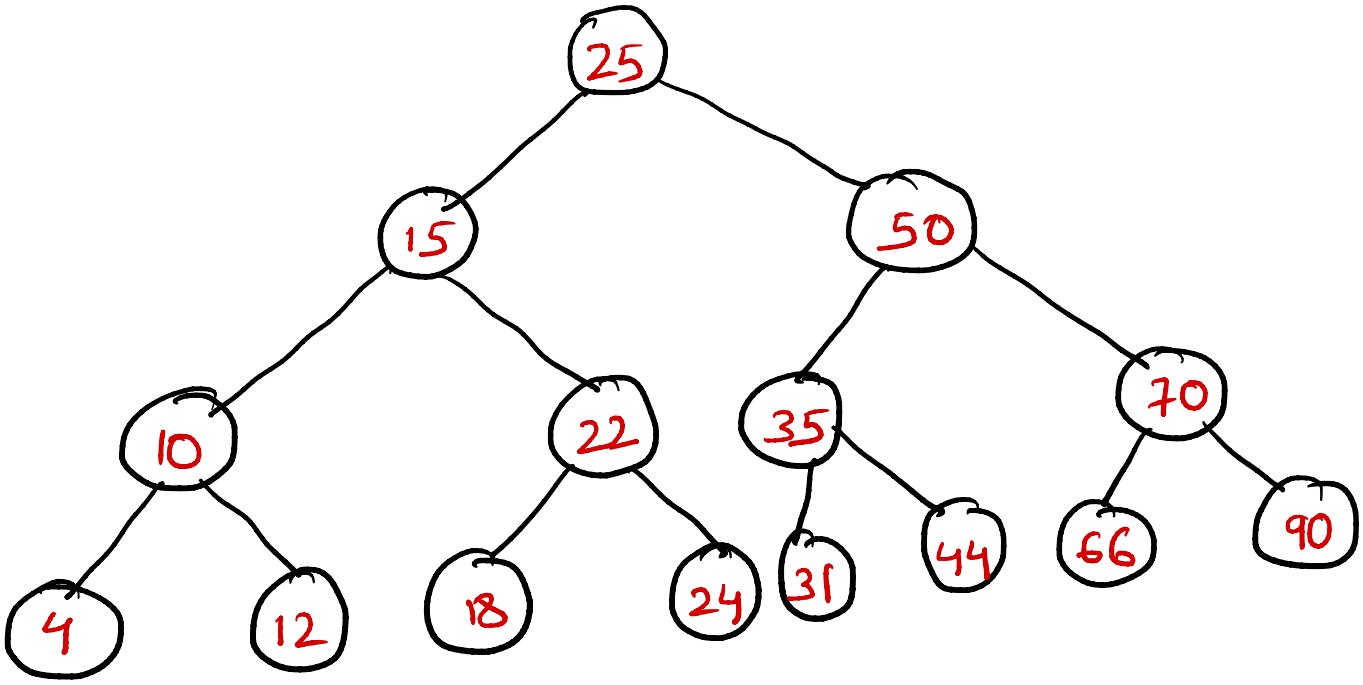
B D A G E C H F I

Preorder

A B D C E G F H I

Postorder

D B G E H I F C A

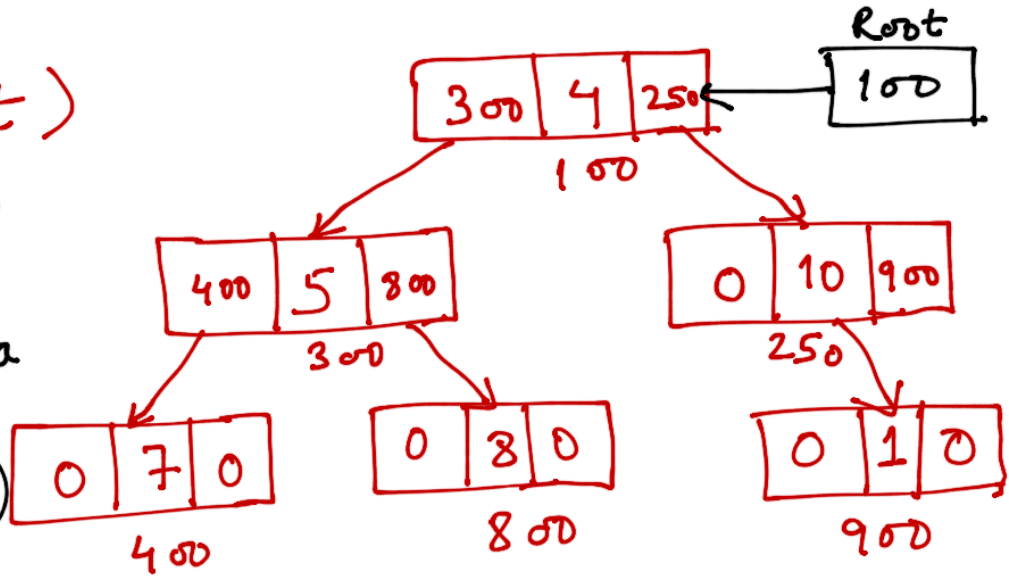


Inorder: 4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

Preorder: 25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

Postorder: 4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25

```
Preorder (Root)
{   if (Root == 0)
      { return }
  PRINT Root -> data
  Preorder (Root->left)
  Preorder (Root-> right)
}
```



Inorder: Left Root Right

7, 5, 8, 4, 10, 1

Pre-order: Root L R

4, 5, 7, 8, 10, 1

Post-order: L R Root

7, 8, 5, 1, 10, 4

```
Inorder (Root)
{   if (Root == 0)
      { Return }
  Inorder (Root -> left)
  PRINT Root -> data
  Inorder (Root -> right)
}
```

```
Postorder (Root)
{   if (Root == 0) { Return }
  Postorder (Root -> left)
  Postorder (Root -> right)
  PRINT Root -> data
}
```

Construct a Binary tree from Preorder & Inorder

Preorder: 1, 2, 4, 8, 9, 10, 11, 5, 3, 6, 7
(Root L R)

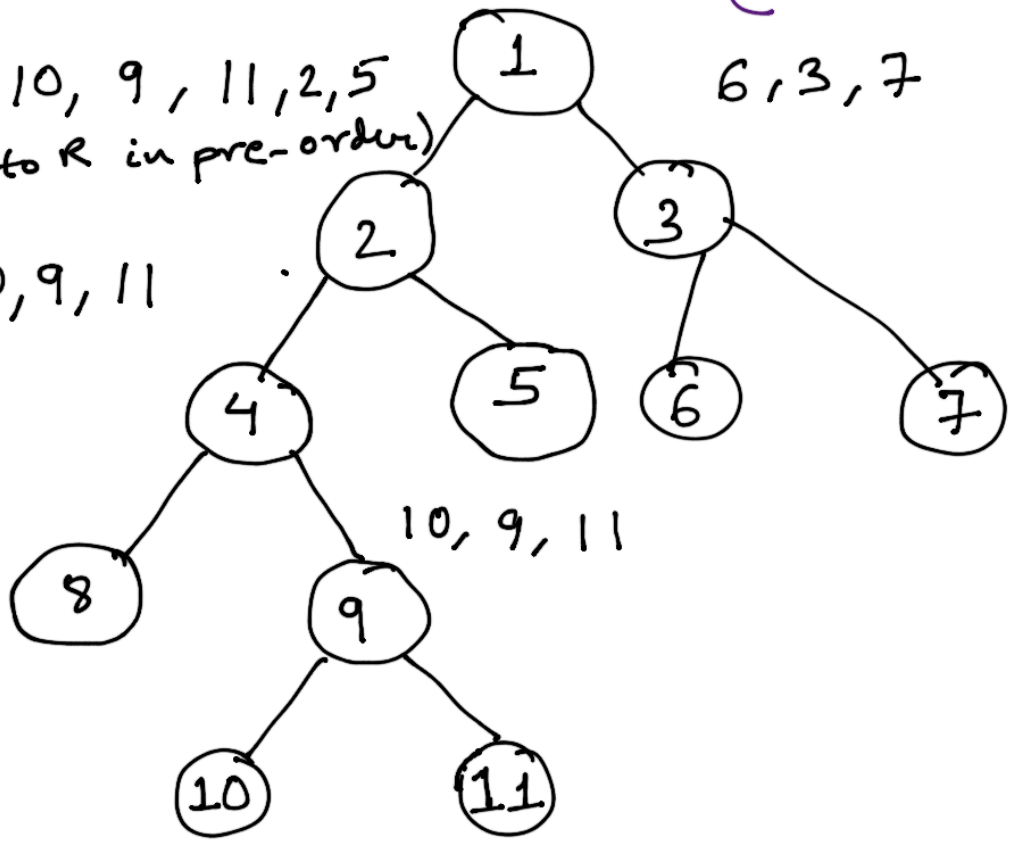
Inorder: 8, 4, 10, 9, 11, 2, 5, 1, 6, 3, 7
(L Root R)

8, 4, 10, 9, 11, 2, 5
(Scan L to R in pre-order)

6, 3, 7

8, 4, 10, 9, 11

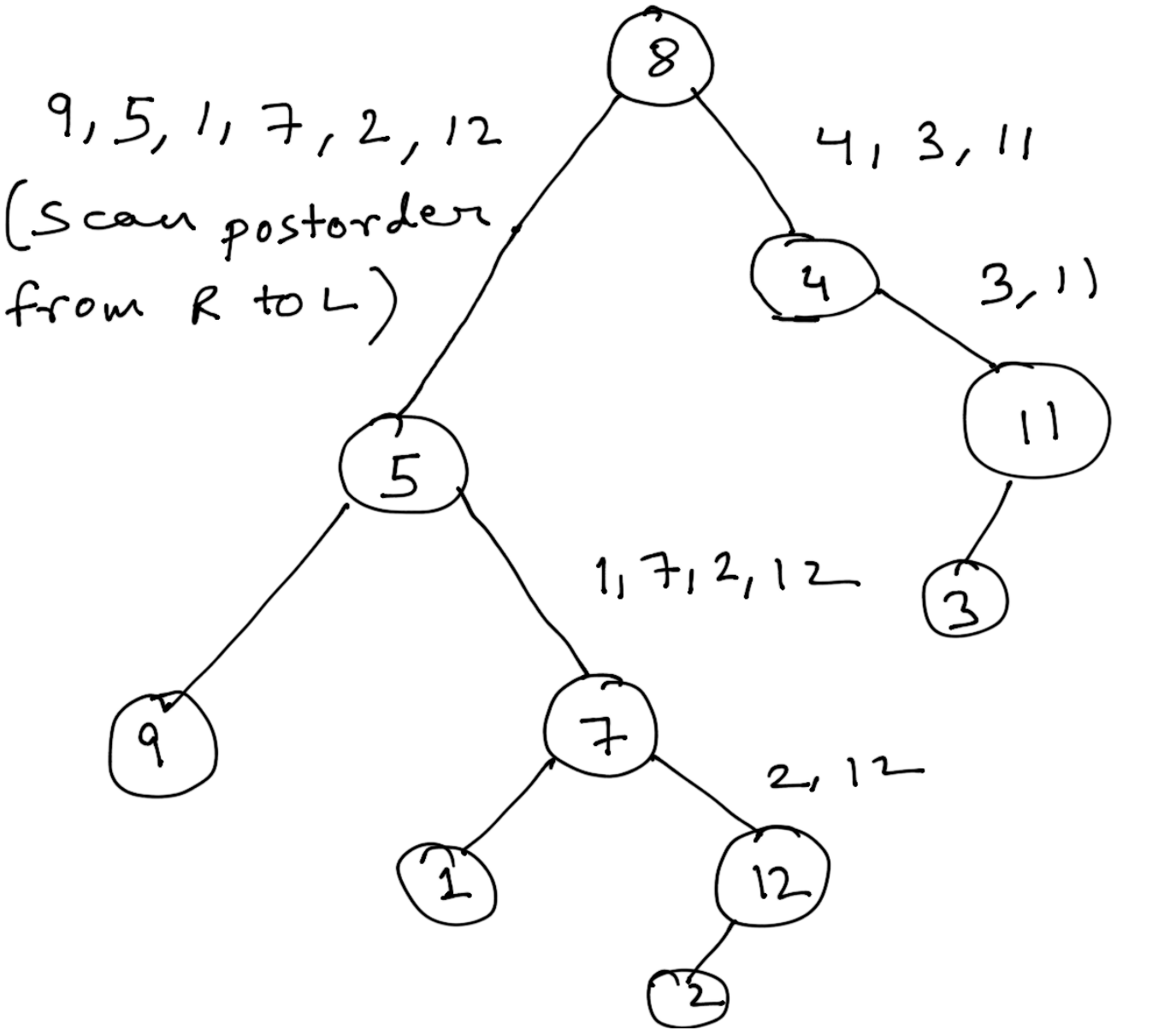
10, 9, 11



Construct a Binary tree from Postorder & Inorder

Postorder: 9, 1, 2, 12, 7, 5, 3, 11, 4, 8
(L R Root)

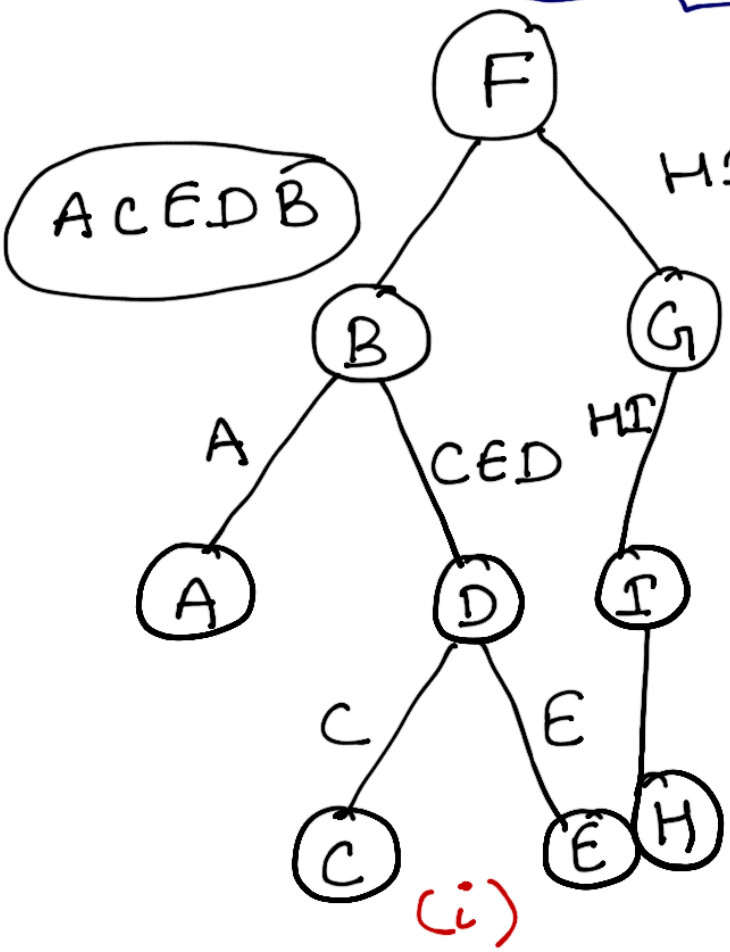
Inorder: 9, 5, 1, 7, 2, 12, 8, 4, 3, 11
(L Root R)



Construct Binary Tree from Preorder & Postorder

Preorder: F B A D C E G I H
(Root L R)

Postorder: A C E D B H I G F
(L R Root)



(i)

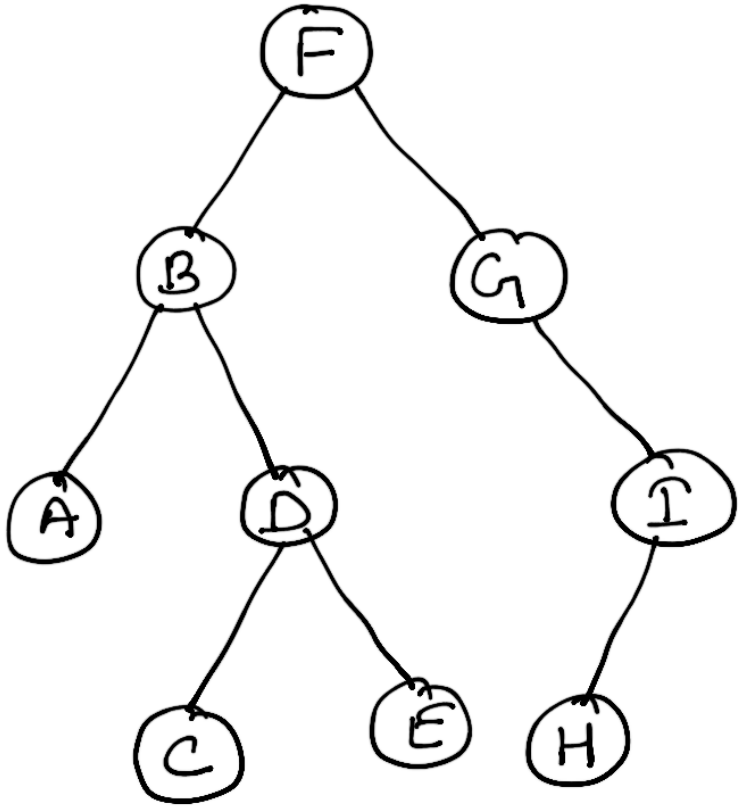
Pre: B A D C E
Post: A C E D B

Pre: D C E
Post: C E D

Pre: G I H
Post: H I G

Pre: I H
Post: H I

*Successor of root in Pre-order & then all elements till successor are part of left sub-tree else right sub-tree

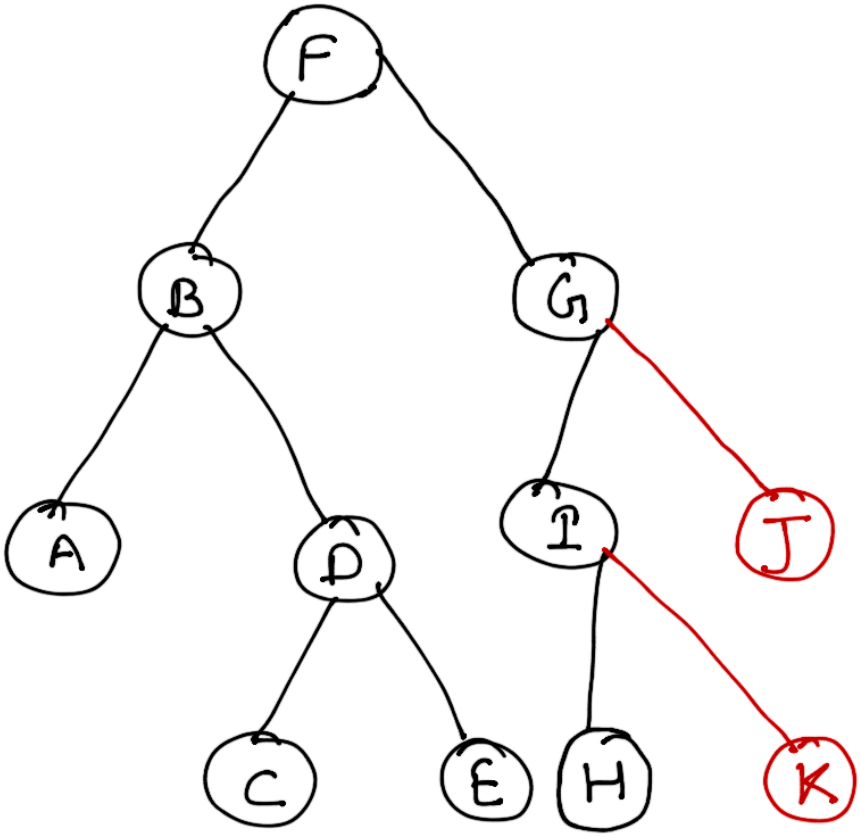


(ii)

find preorder traversal & postorder traversal ??

Preorder:

Postorder:



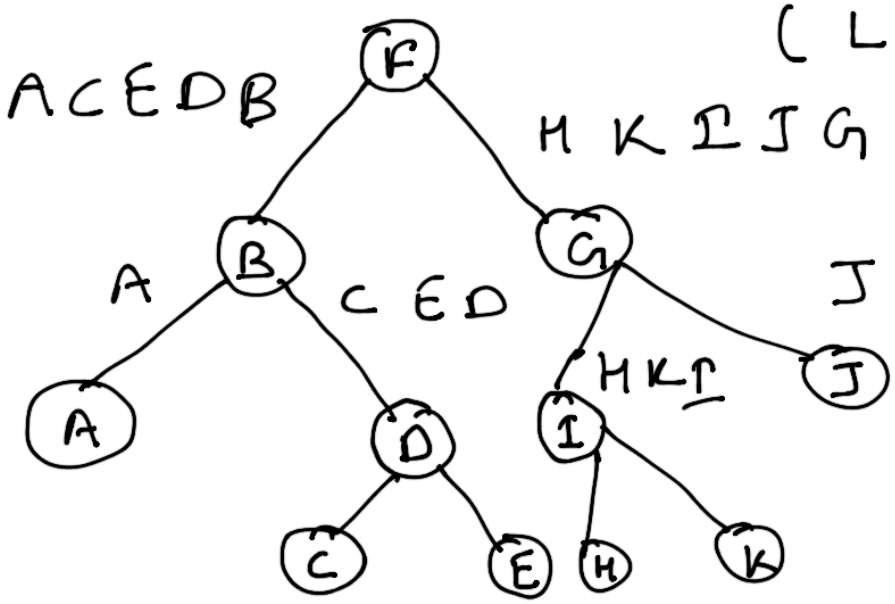
(i) Full Binary Tree

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

(Root L R)

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

(L R Root)



(ii) Full Binary Tree.

Remember:

If preorder & postorder are given, we can construct

- → A unique full Binary Tree
- → NOT a unique Binary Tree