Tutorial 07: Tree

CSCI2520 - DATA STRUCTURES AND APPLICATIONS

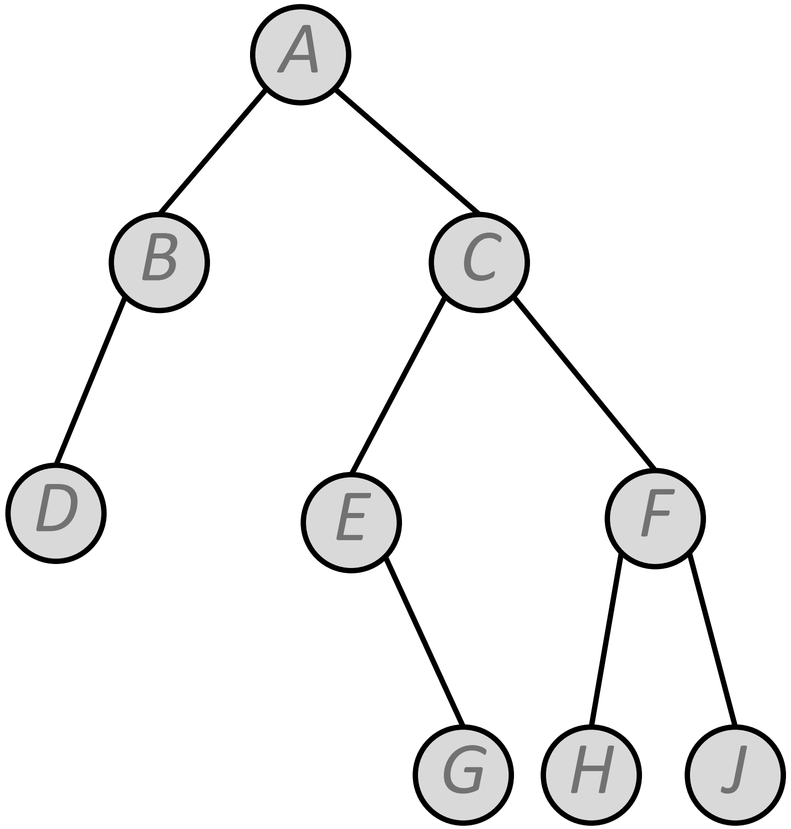
TUTOR: ZHANG KAI

Outline

1. Binary Tree Traversal
2. Find Successor in BST
3. AVL Tree

Binary Tree Traversal

Traversal
- Pre-order
- In-order
- Post-order



Preorder:

A B D C E G F H J

Inorder:

D B A E G C H F J

Postorder:

D B G E H J F C A

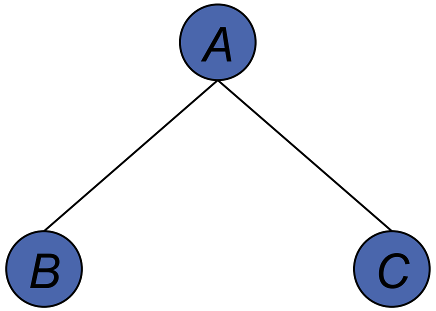
Binary Tree Reconstruction

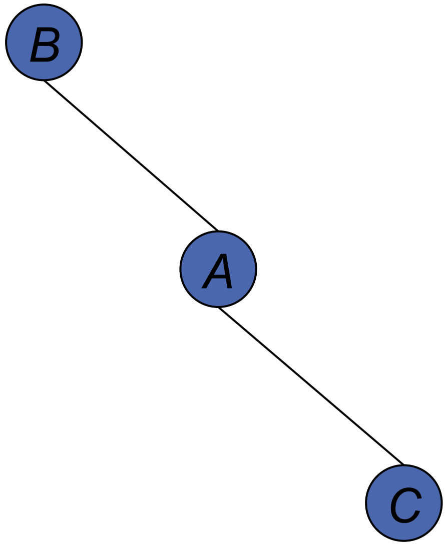
Question: can you reconstruct the binary tree from its *inorder/preorder/postorder* traversal?

Or, if the inorder traversal of a binary tree is (B, A, C), what will it be?

Binary Tree Reconstruction

Both have inorder traversal (B, A, C)!





Exercise 1

Given the following information, try to reconstruct the original binary tree.

In-order:

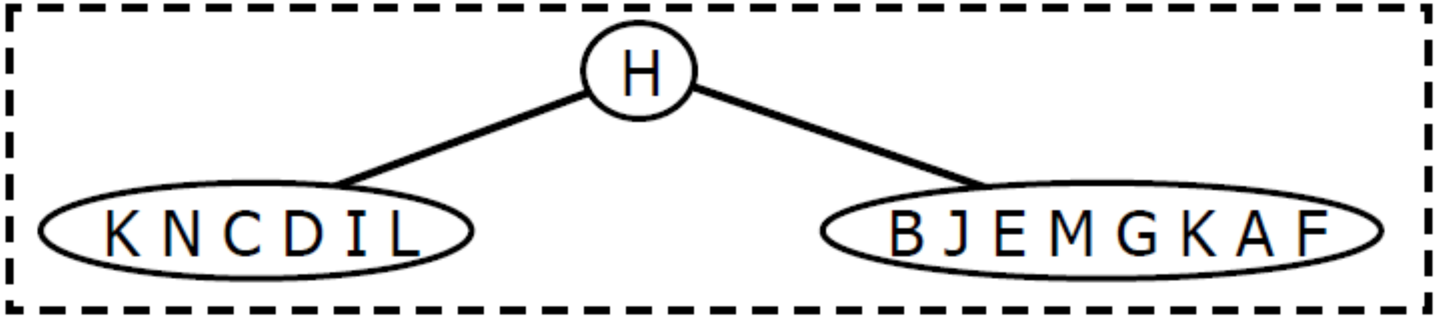
K N C D I L H B J E M G K A F

pre-order:

H D N K C I L G E B J M F K A

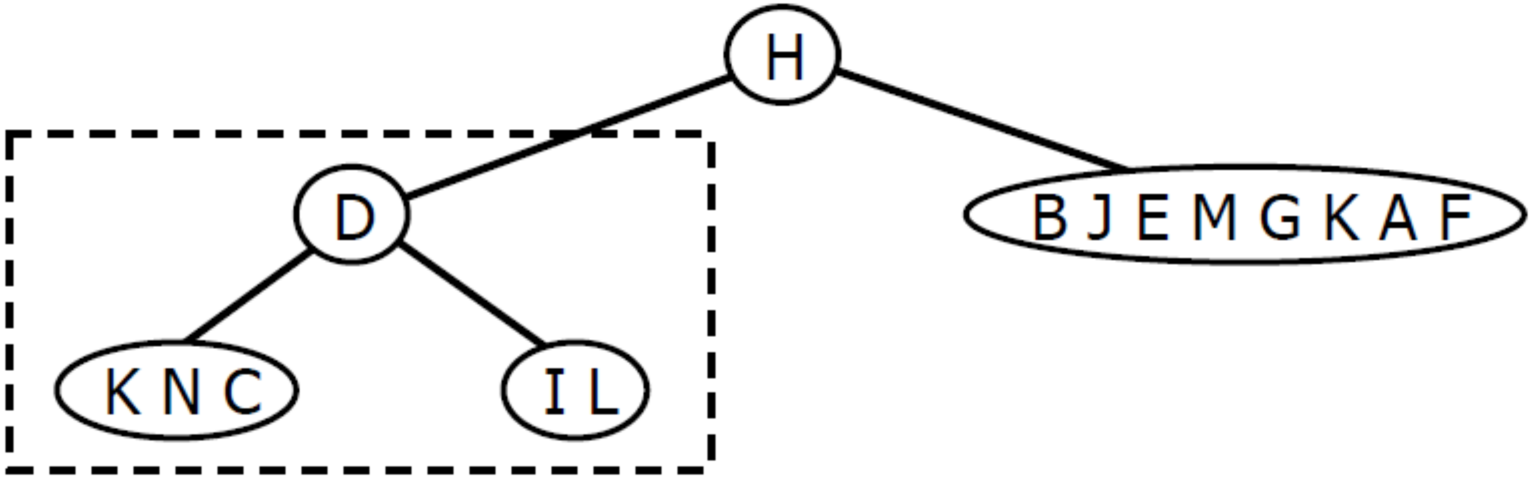
Binary Tree Reconstruction

- Inorder: K N C D I L H B J E M G K A F
- Preorder: H D N K C I L G E B J M F K A



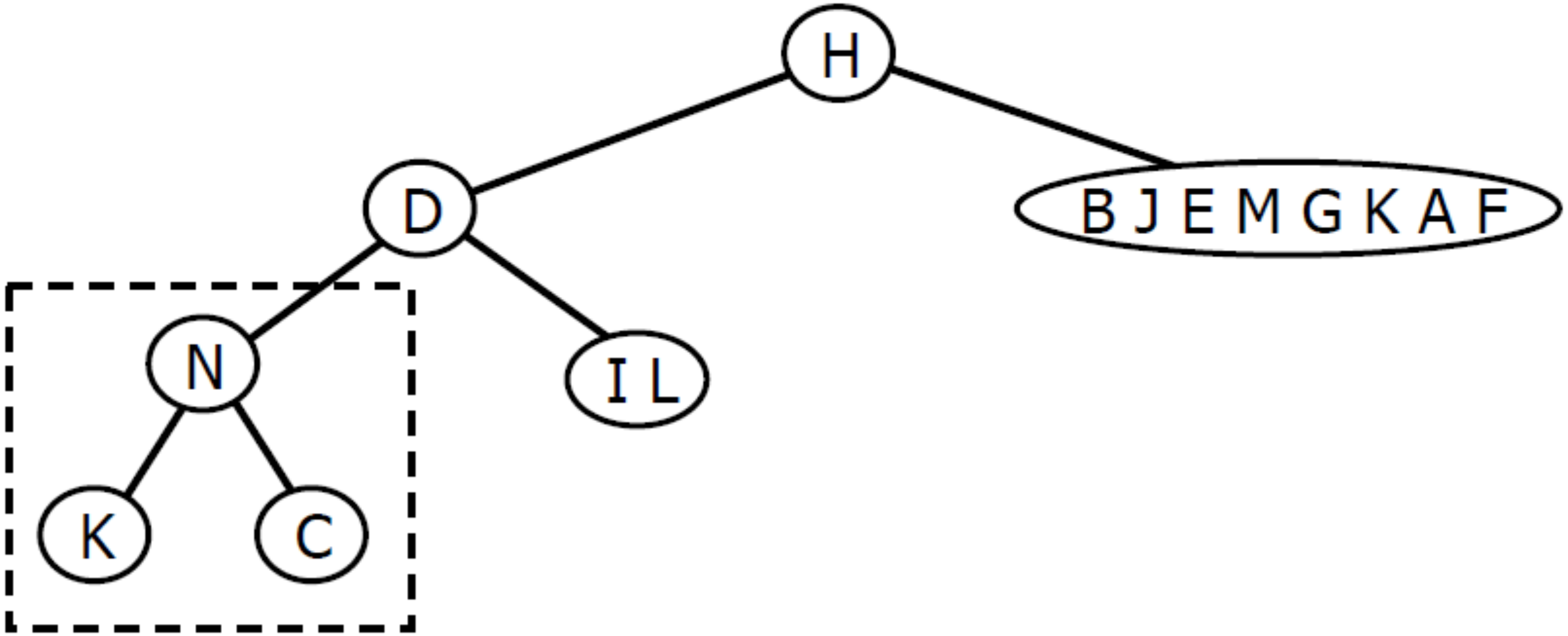
Binary Tree Reconstruction

- Inorder: K N C D I L H B J E M G K A F
- Preorder: H D N K C I L G E B J M F K A



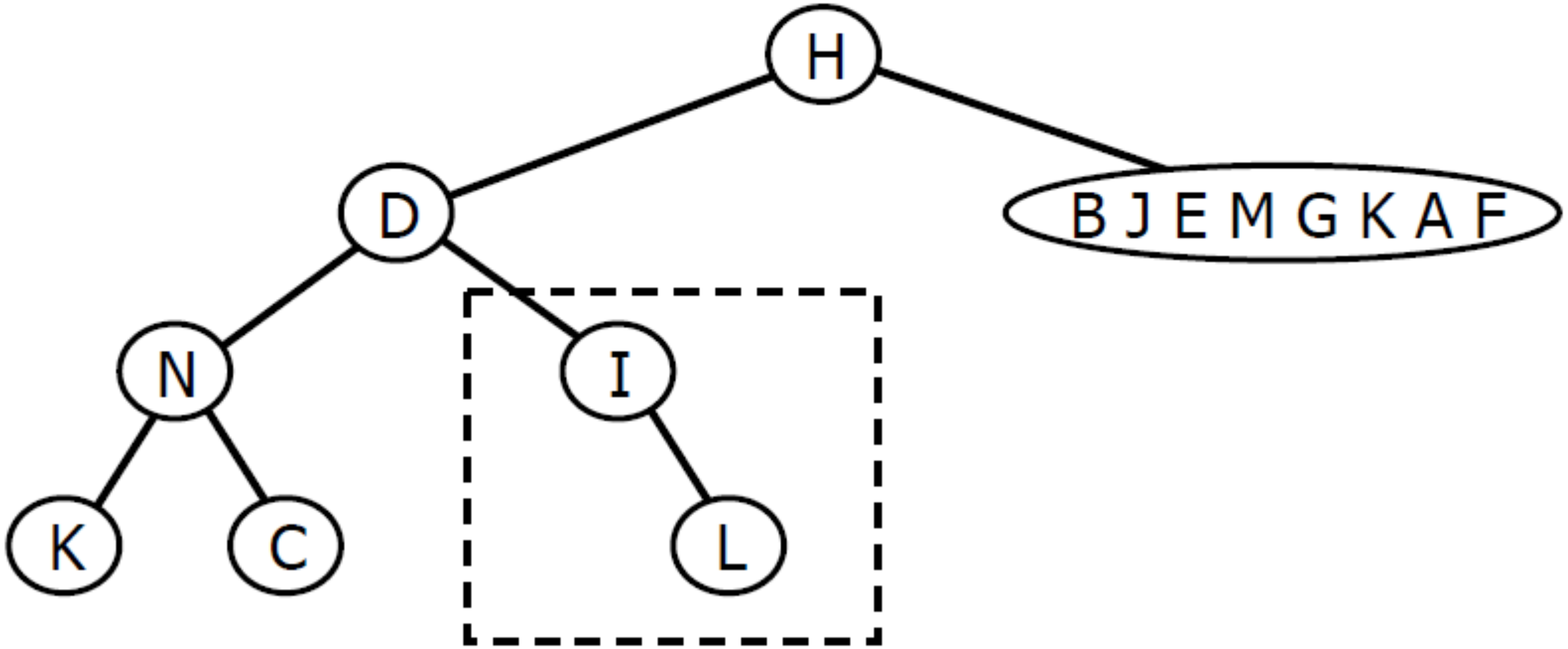
Binary Tree Reconstruction

- Inorder: K N C D I L H B J E M G K A F
- Preorder: H D N K C I L G E B J M F K A



Binary Tree Reconstruction

- Inorder: K N C D I L H B J E M G K A F
- Preorder: H D N K C I L G E B J M F K A



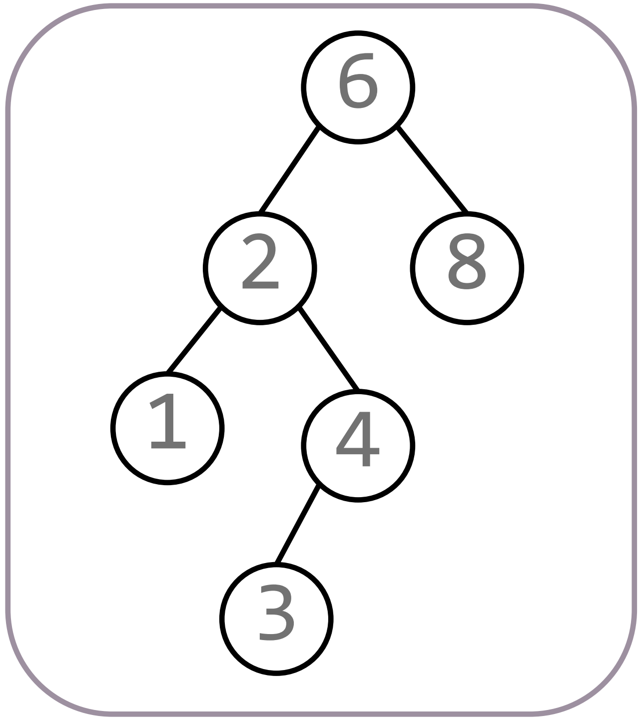
Binary Tree Reconstruction

Key steps:

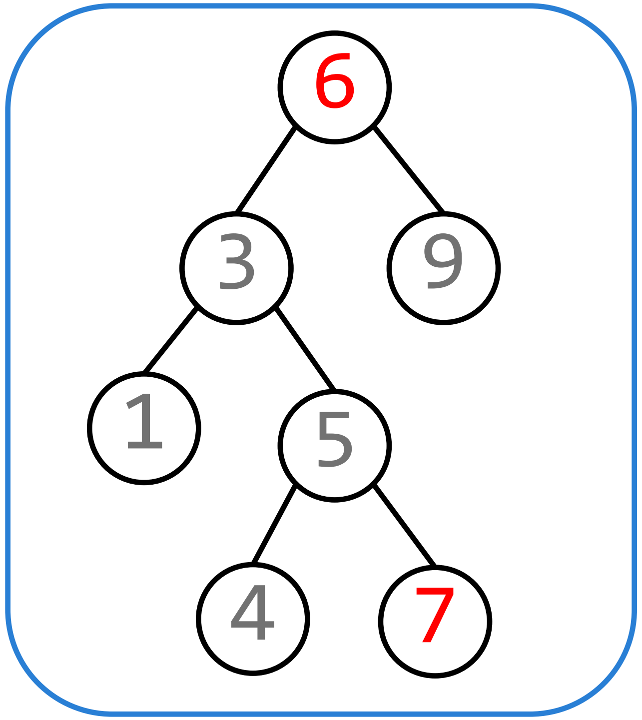
- Find the root
- Find the left and right subtrees and do it recursively

Given both inorder and postorder traversals (or preorder and inorder traversals), the methods are similar.

Binary Search Tree



A binary search Tree



Not a binary search Tree

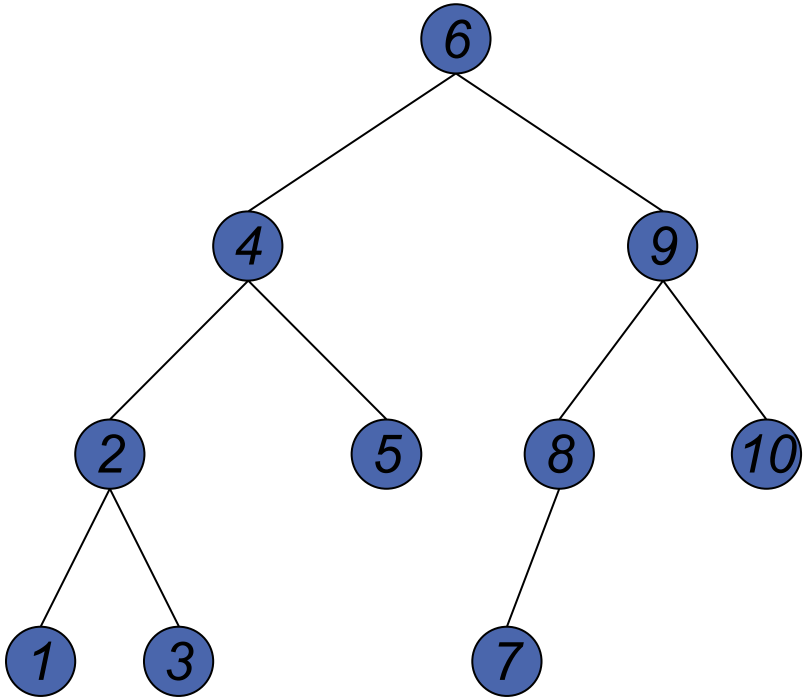
(where is the problem?)

Find Inorder Successor in BST

How to find the inorder successor of a node in BST?

How many cases we have?

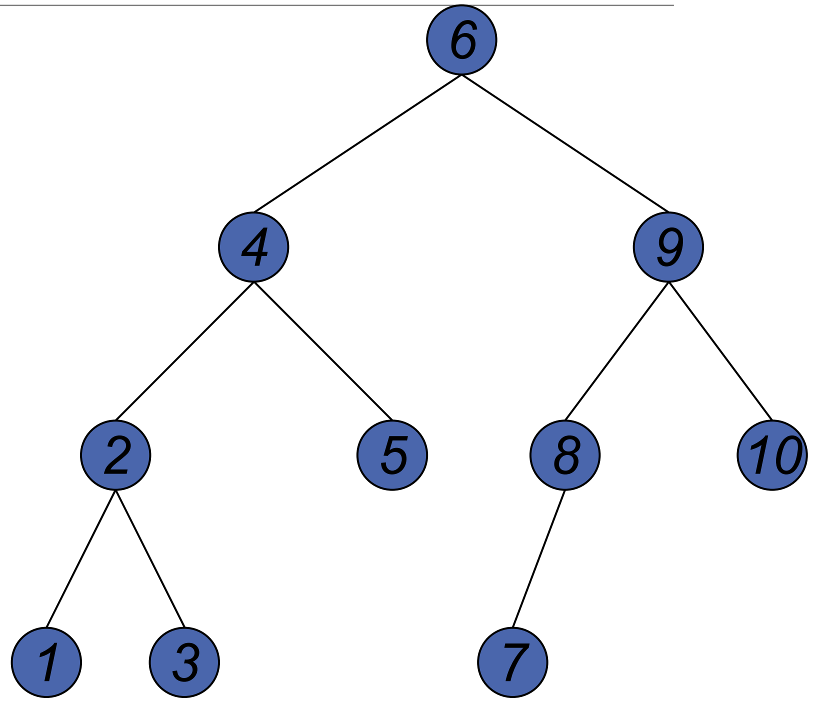
- 2: 3
- 4: 5
- 6: 7
- 3: 4
- 5: 6
- 8: 9



Find Inorder Successor in BST

Two cases!

- **1)** If right subtree of *node* is not *NULL*, then *succ* lies in right subtree. Go to right subtree and return the node with minimum key value in right subtree.
  - 2: 3
  - 4: 5
  - 6: 7
- **2)** If right sbtree of *node* is NULL, then *succ* is one of the ancestors. Travel up using the parent pointer until you see a node which is left child of it's parent. The parent of such a node is the *succ*.
  - 3: 4
  - 5: 6
  - 8: 9



Exercise 2

Implement the following function

```
bstADT FindSuccessor(bstADT n)
```

The definition of bstCDT is as follows

```
struct bstCDT {
    treeNodeADT root;
    bstADT left;
    bstADT right;
    bstADT parent;
};
```

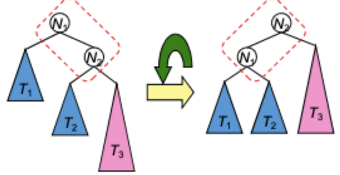
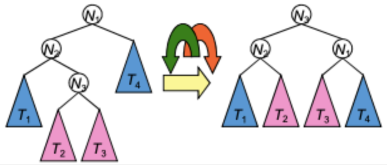
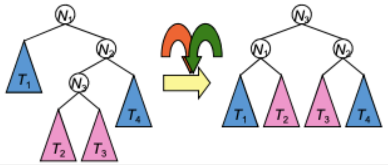
Exercise 2 Solution

```
bstADT FindSuccessor(bstADT n){
    if (n->right != NULL) {
        // go to right subtree
        n = n->right;
        while (n->left != NULL) n = n->left;
        return n;
    } else {
        // trace back to ancestors
        bstADT p = n->parent;
        while (p != NULL && p->left != n){
            n = p;
            p = n->parent;
        }
        return p;
    }
}
```

AVL Tree

A balanced tree maintained by single or double rotations is called an **AVL tree**.

An imbalance caused by insertion can always be fixed by performing one operation, either a single or double rotation.

Case 1: insertion to *right* subtree of *right* child	Case 2: insertion to *left* subtree of *left* child
Solution: *Left* rotation 	Solution: *Right* rotation 
Case 3: insertion to *right* subtree of *left* child	Case 4: insertion to *left* subtree of *right* child
Solution: *Left*-*right* rotation 	Solution: *Right*-*left* rotation 

Exercise 3

What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

- (A) 2
- (B) 3
- (C) 4
- (D) 5

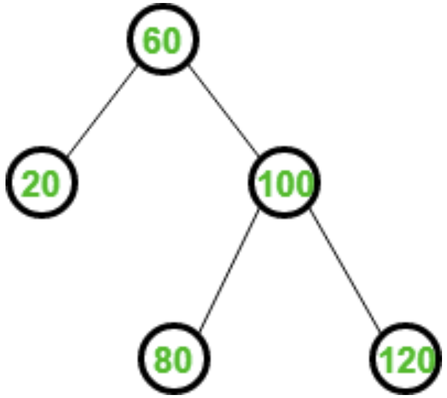
Exercise 3 Solution

For finding maximum height, the nodes should be minimum at each level. Assuming height as 2, minimum number of nodes required:
$N(h) = N(h-1) + N(h-2) + 1$
$N(2) = N(1) + N(0) + 1 = 2 + 1 + 1 = 4.$
It means, height 2 is achieved using minimum 4 nodes.

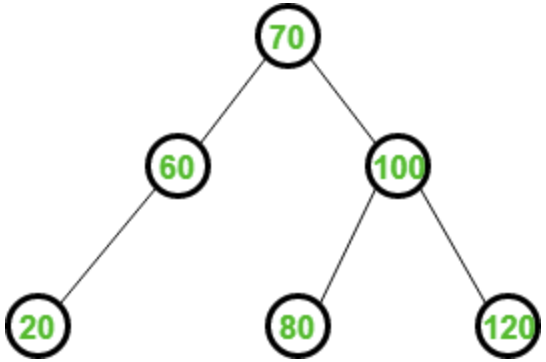
Assuming height as 3, minimum number of nodes required:
$N(h) = N(h-1) + N(h-2) + 1$
$N(3) = N(2) + N(1) + 1 = 4 + 2 + 1 = 7.$
It means, **height 3 is achieved using minimum 7 nodes.**

Exercise 4

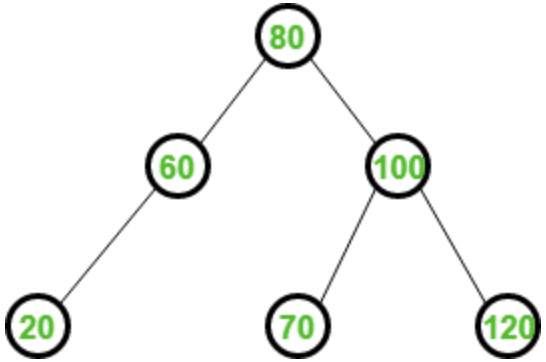
Consider the following AVL tree. Which of the following is updated AVL tree after insertion of 70?



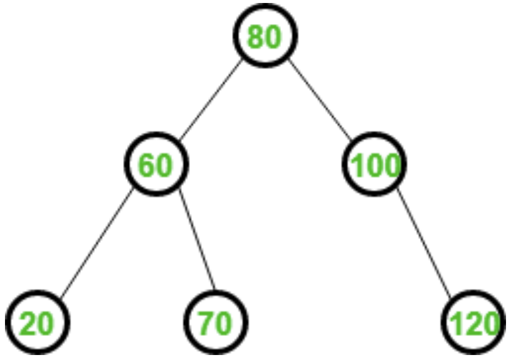
A.



B.

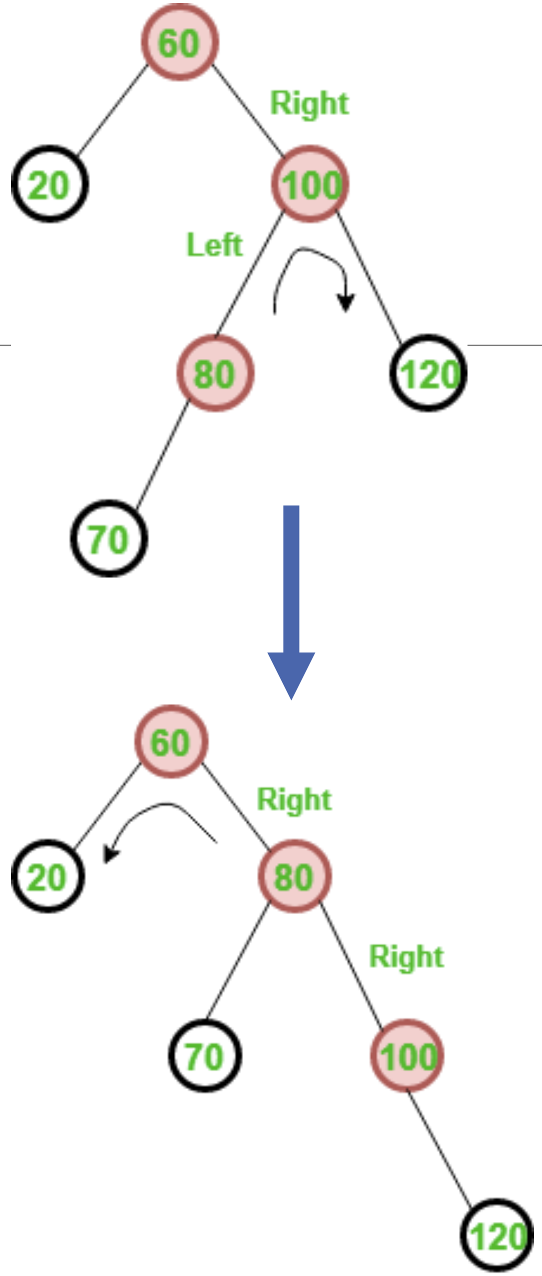


C.



Exercise 4 Solution

The element is first inserted in the same way as BST. Therefore after insertion of 70, BST can be shown as:



However, balance factor is disturbed requiring RL rotation. To remove RL rotation, it is first converted into RR rotation as:

After removal of RR rotation, **AVL tree generated is same as option (C).**

References

1. Slides from Tatiana Jin

2. https://www.geeksforgeeks.org/practice-questions-height-balancedavl-tree/