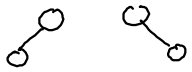
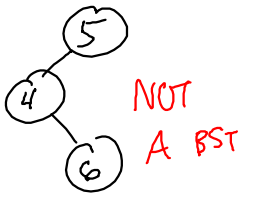
Tree — either empty or a node with some number of trees as children

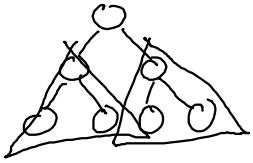
Binary Tree — tree where each node has at most one left child and at most one right child

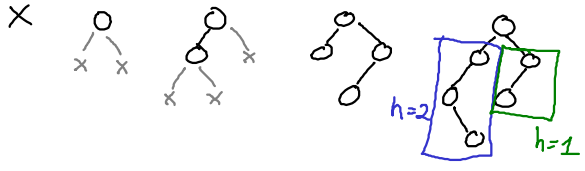


BST — binary tree where, for every node, all left descendants have lesser keys and all right desc. have greater keys



NOT A BST





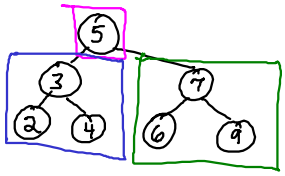
h=-1 h=0 h=1 h=2 h>3

Height — largest # of edges from root to a leaf

$$H(T) = \begin{cases} -1 & \text{when } T \text{ is empty} \\ \max(H(L), H(R)) + 1 & \text{when } T \text{ has children } L \text{ and } R \end{cases}$$

$O(n)$ where n = # nodes

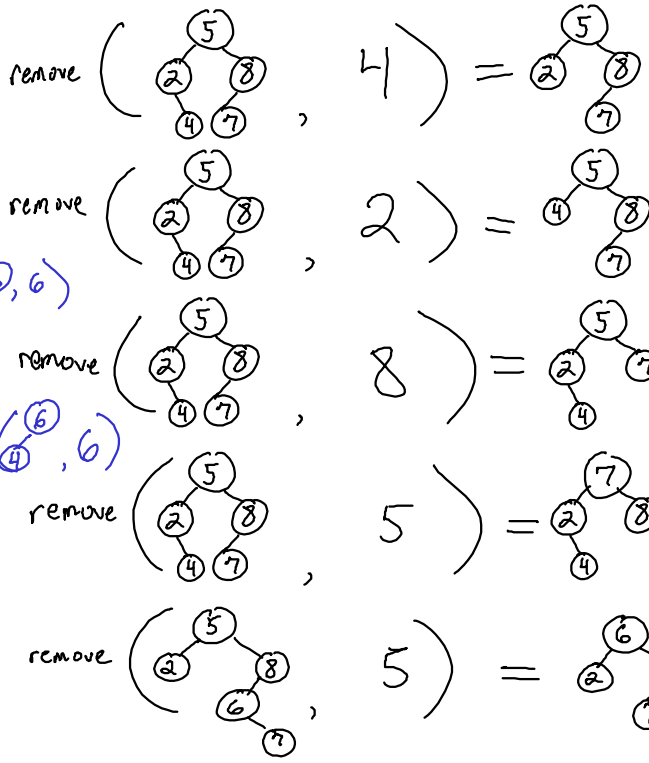
Traversal — some ordering of the nodes in a structure



- pre-order — visit root, visit all left, visit all right
  5, [3, 2, 4], [7, 6, 9]
- in-order — visit left, visit root, visit right
  [2 3 4] 5 [6 7 9]
- post-order — visit left, visit right, visit root
  [2, 4, 3] [6, 9, 7] 5
- level-order — visit all nodes in order of level (BFS)
  5, 3, 7, 2, 4, 6, 9
- level — all nodes w/ same depth (distance from root)

$O(n)$

```
Function removeInSubtree(node, key):        ← return replacement
    If node is null:
        !!
    Else If node.key == key Then
        If node.left == null and node.right == null:
            Return null
        Else If node.left ≠ null and node.right == null:
            Return node.left
        Else If node.left == null and node.right ≠ null:
            Return node.right
        Else:
            newkey ← getMinkey(node.right)
            newval ← getInSubtree(node.right, newkey)
            node.key ← newkey
            node.value ← newval
            node.right ← removeInSubtree(node.right, newkey)
        End If
    Else If key < node.key:
        node.left ← removeInSubtree(node.left, key)
    Else:
        node.right ← removeInSubtree(node.right, key)
    End If
```



remove(6, 6)

remove(6, 6)

newkey ← 6
newval ← ...

O(h)

AVL Tree — BST where, for all nodes, height of left child & height of right child differ by at most 1

AVL algorithms ensure height is $O(\log n)$