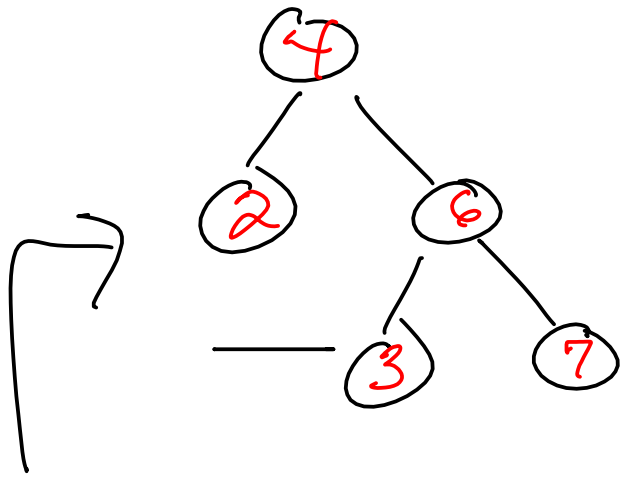
Tree — data structure made of nodes with root and leaves



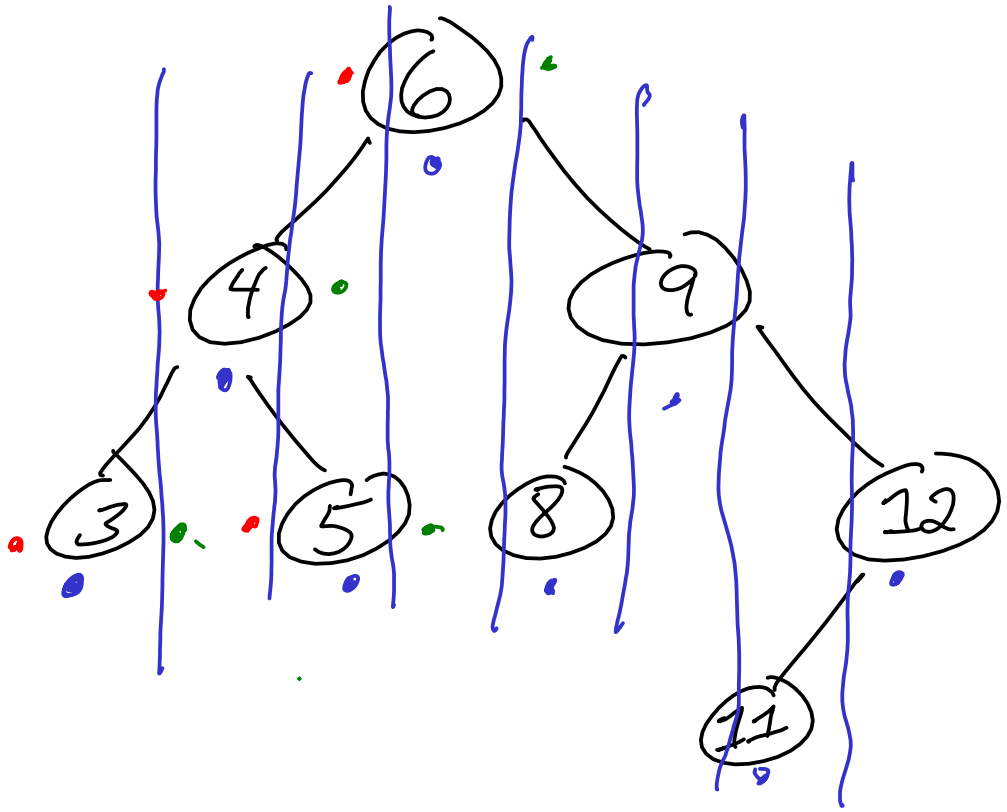
← NOT BST

Binary Tree: each node has ≤2 children

BST: at each node, all data in left descendents is less, all data in right descendents is greater.

Traversal

Traversals



In-order

Traverse left
Visit here
Traverse right

3 4 5 6 8 9 11 12

Pre-Order

Visit Here
Traverse left
Traverse right

6 4 3 5 9 8 12 11

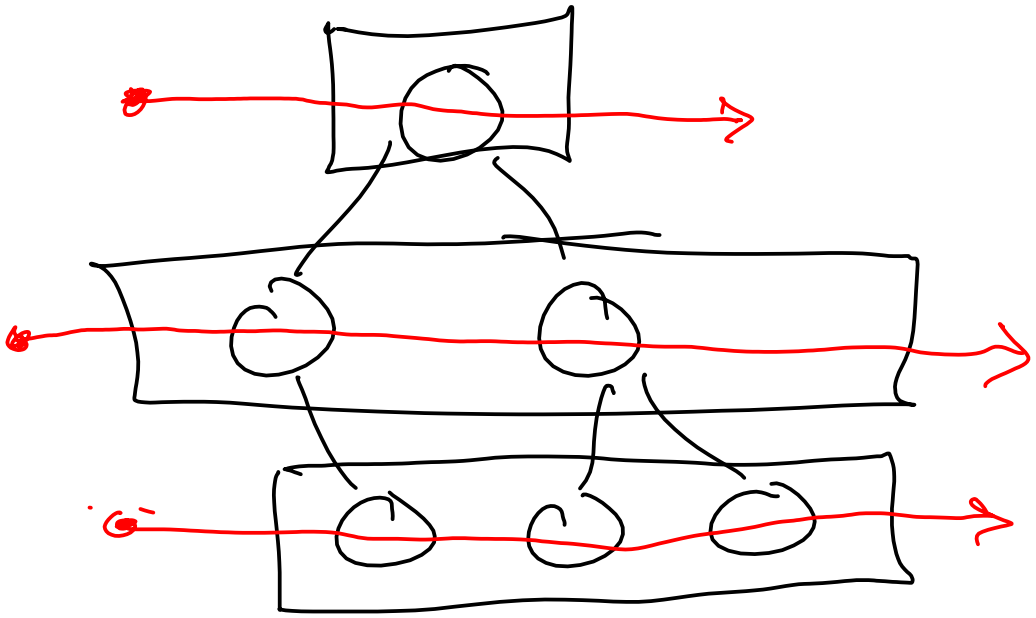
Post-Order

Traverse left
Traverse right
Visit here

3 5 4 8 11 12 9 6

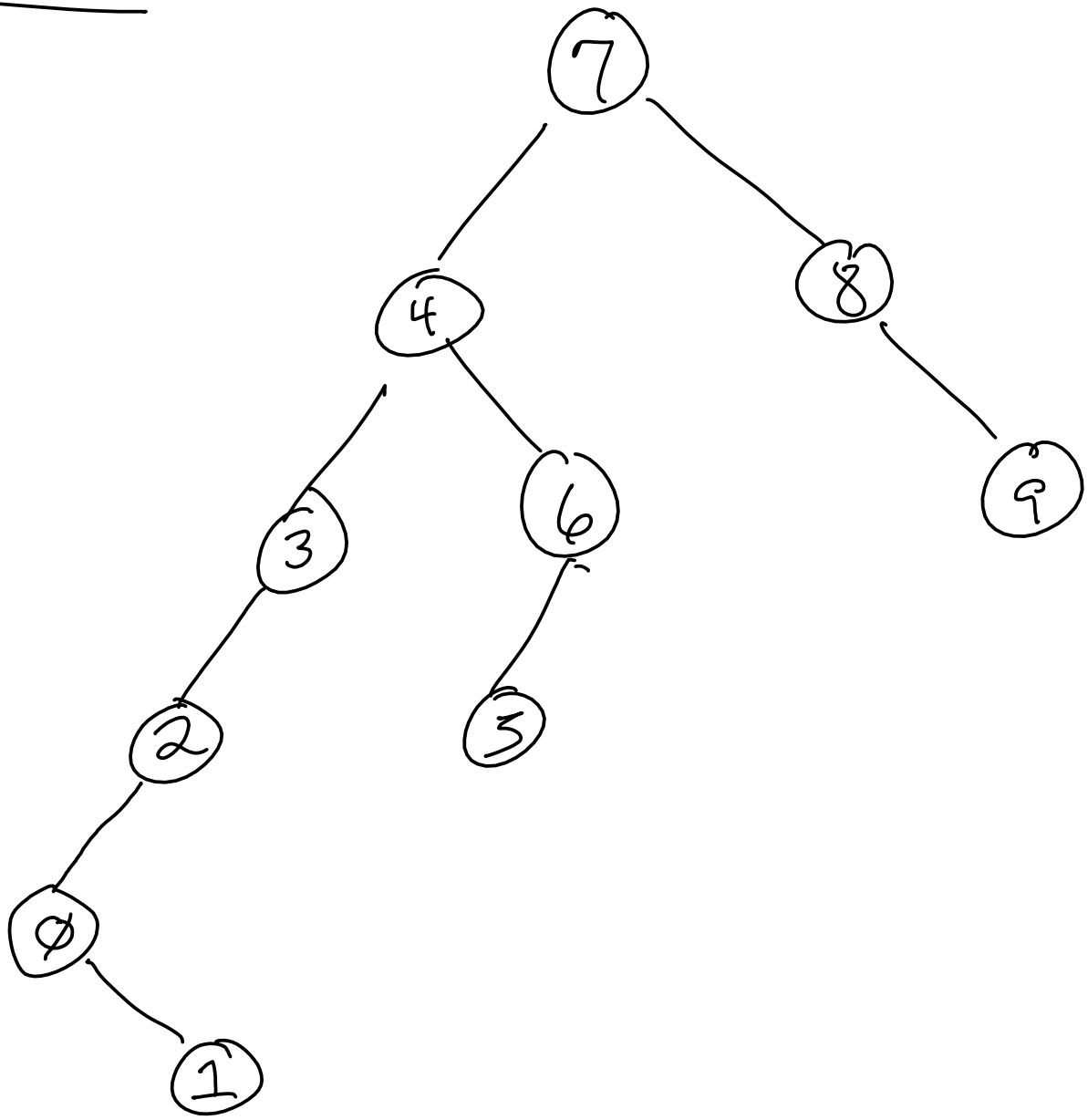
Level-order Traversal

"Level" - nodes same distance from root

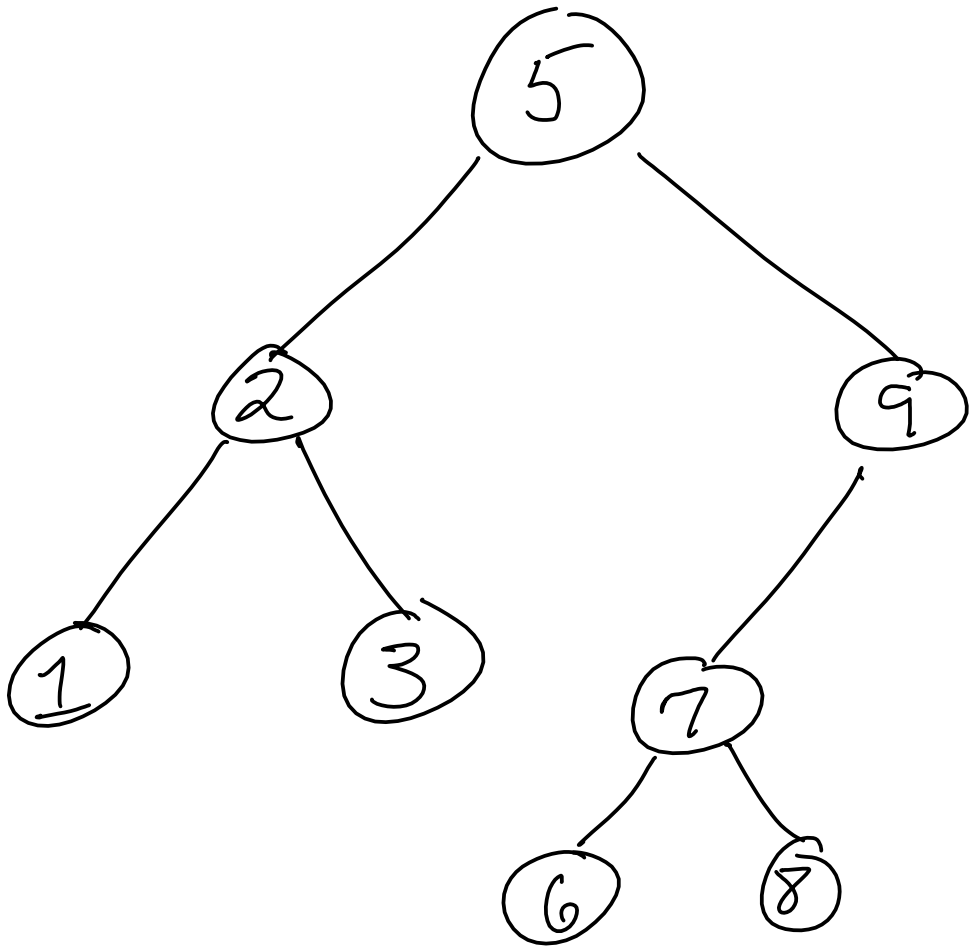


getMin

getMax

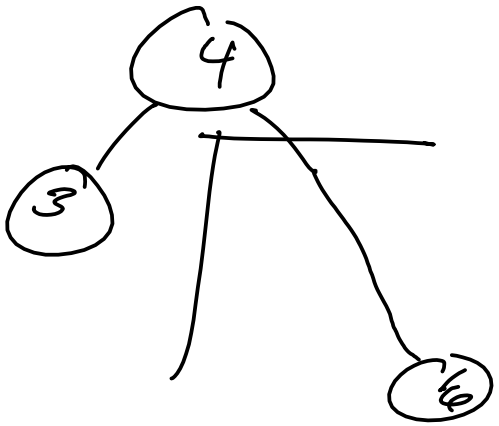


Remove



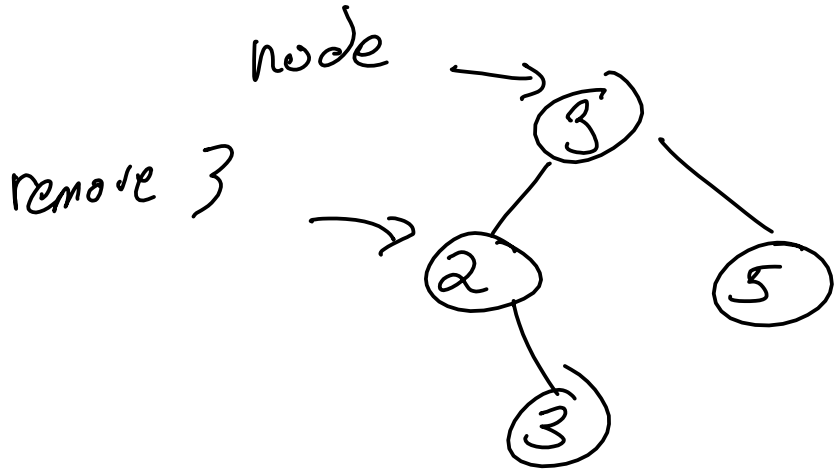
1. No children
2. 1 child.
3. 2 children

```
Function remove(node, key)
    If node->key < key:
        If node->right. is NULL:
            throw exception
        Else:
         node->right ← remove(node->right, key)
    ElseIf node->key > key:
        If node->left is NULL:
            throw
        Else
            node->left ← remove(node←left, key)
    Else:
```



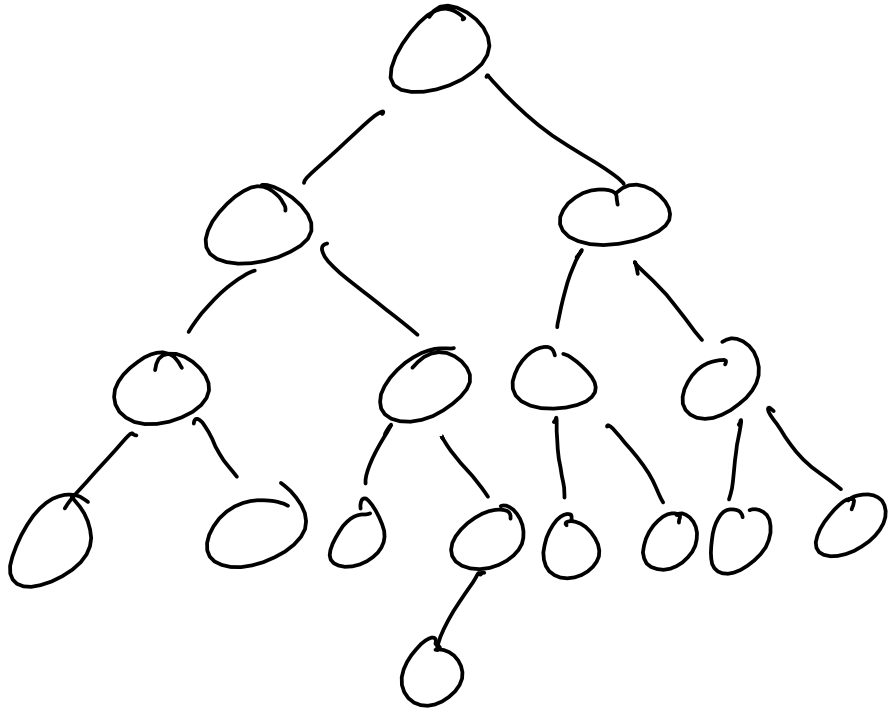
(continued ↓)

If node has no children:
  return NULL
Else If node has one child:
  return node's child
Else: node->key ← max(node->left)
  node contents ← max(node->left) contents
  node->left ← remove(node<-left, node->key)
  return node



Performance

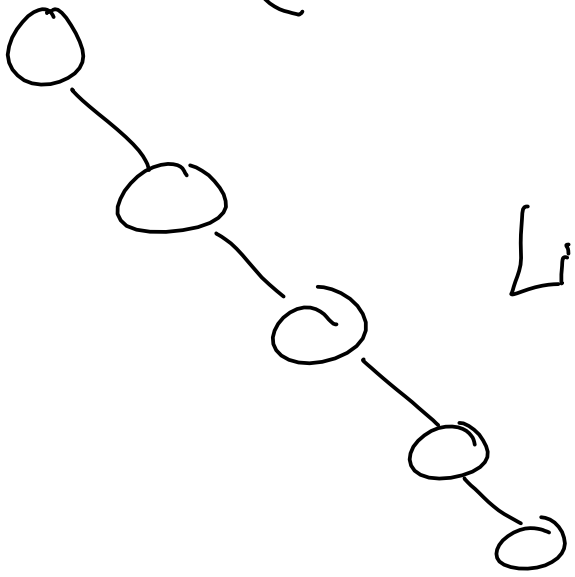
height
O(log n)



15

get
O(height)

all operations
(not traversals)



Linked List !!