CSCA48

Brian Chen, Yufei Cui

byxc.me/ta/A48_TT2.pdf

Linked Lists

- An object called a node contains its information and a pointer to the next node in the list

- Singly linked vs doubly linked list

```
class SingleNode:
  def __init__(self, data, next=None):
    self.data = data
    self.next = next
```

```
class DoubleNode:
  def __init__(self, data, next=None, prev=None):
    self.data = data
    self.next =next
    self.prev = prev
```

Binary Trees

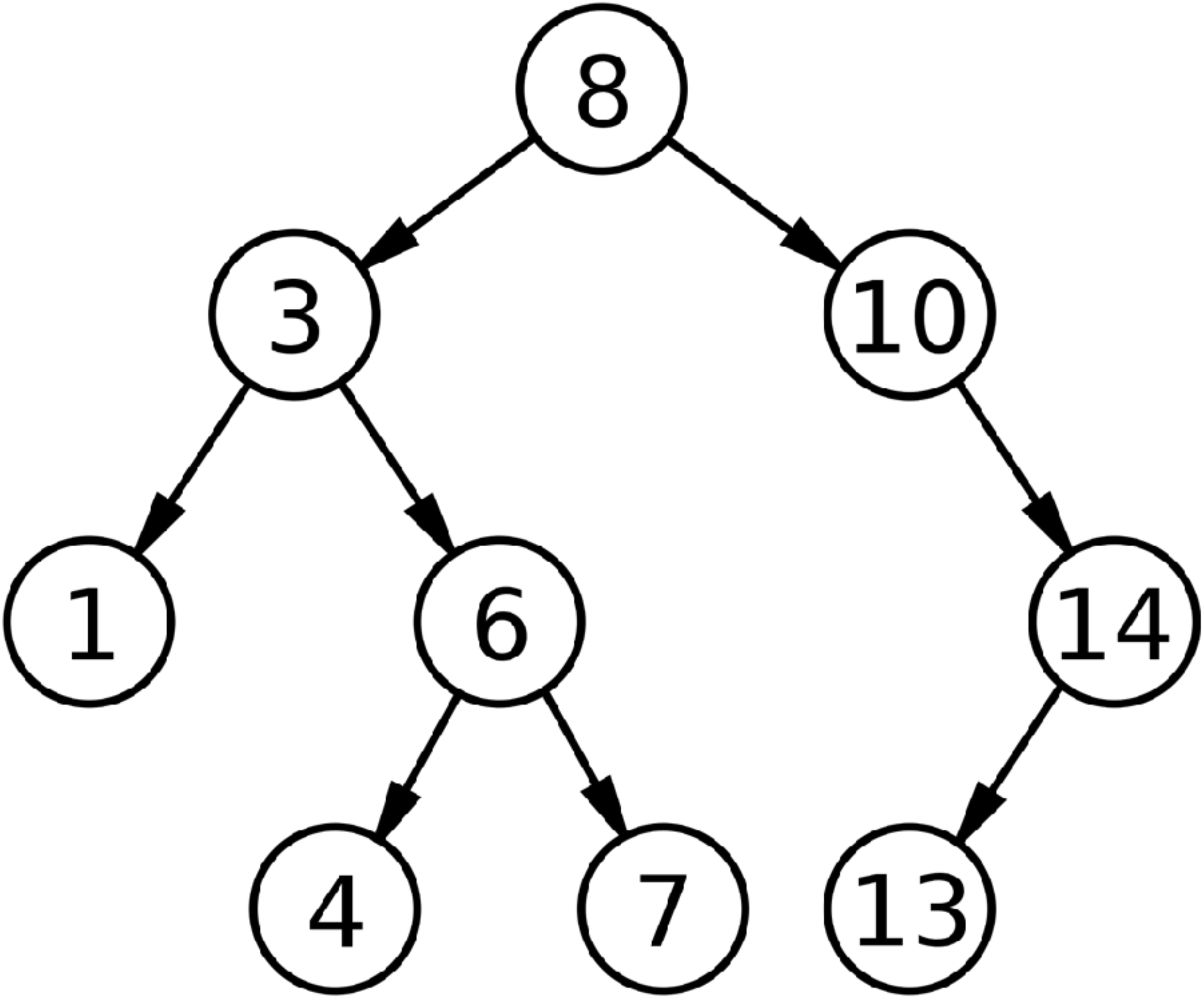
- Linked list, but each node has 2 children (left and right)
- Does not link back up (ie, more akin to singly linked list)

```
class BinaryTreeNode:

  def __init__(self, data, left=None, right=None):
    self.data = data

    self.left = left

    self.right = right
```


8
3
10
1
6
14
4
7
13

Binary Search Tree (BST)

- Binary tree except values on the left are always <= root, values on right are always >= root

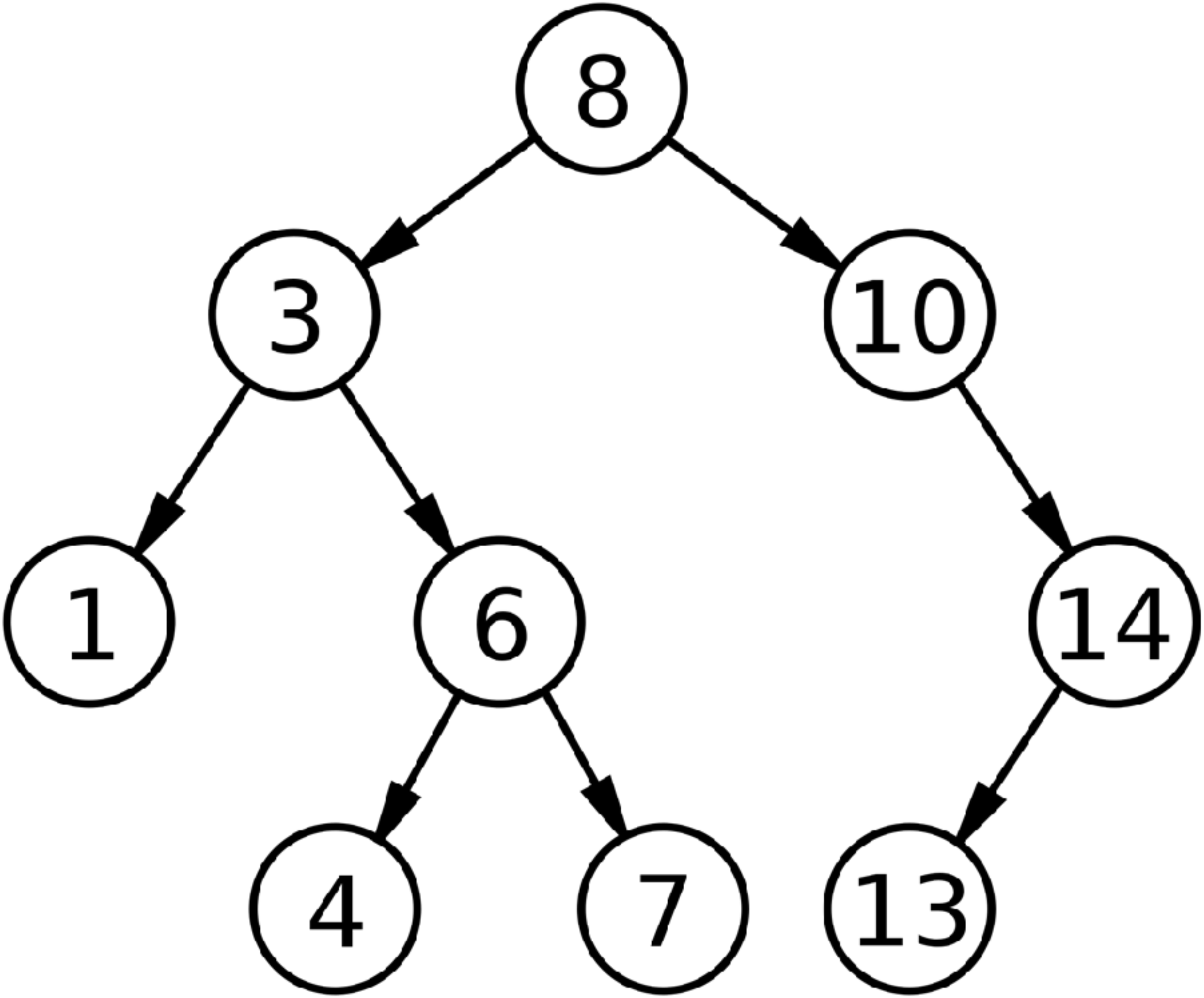
- same properties otherwise

Inserting to BST

```
>>> INSERT
The INSERT operation is easy enough. The basic
algorithm is:
Compare with node (start at root).
If larger - try to compare it with right node
If smaller - try to compare it with left node
If equal - place it in any subtree (shouldn’t
exist, replicates)
```

Deleting from BST

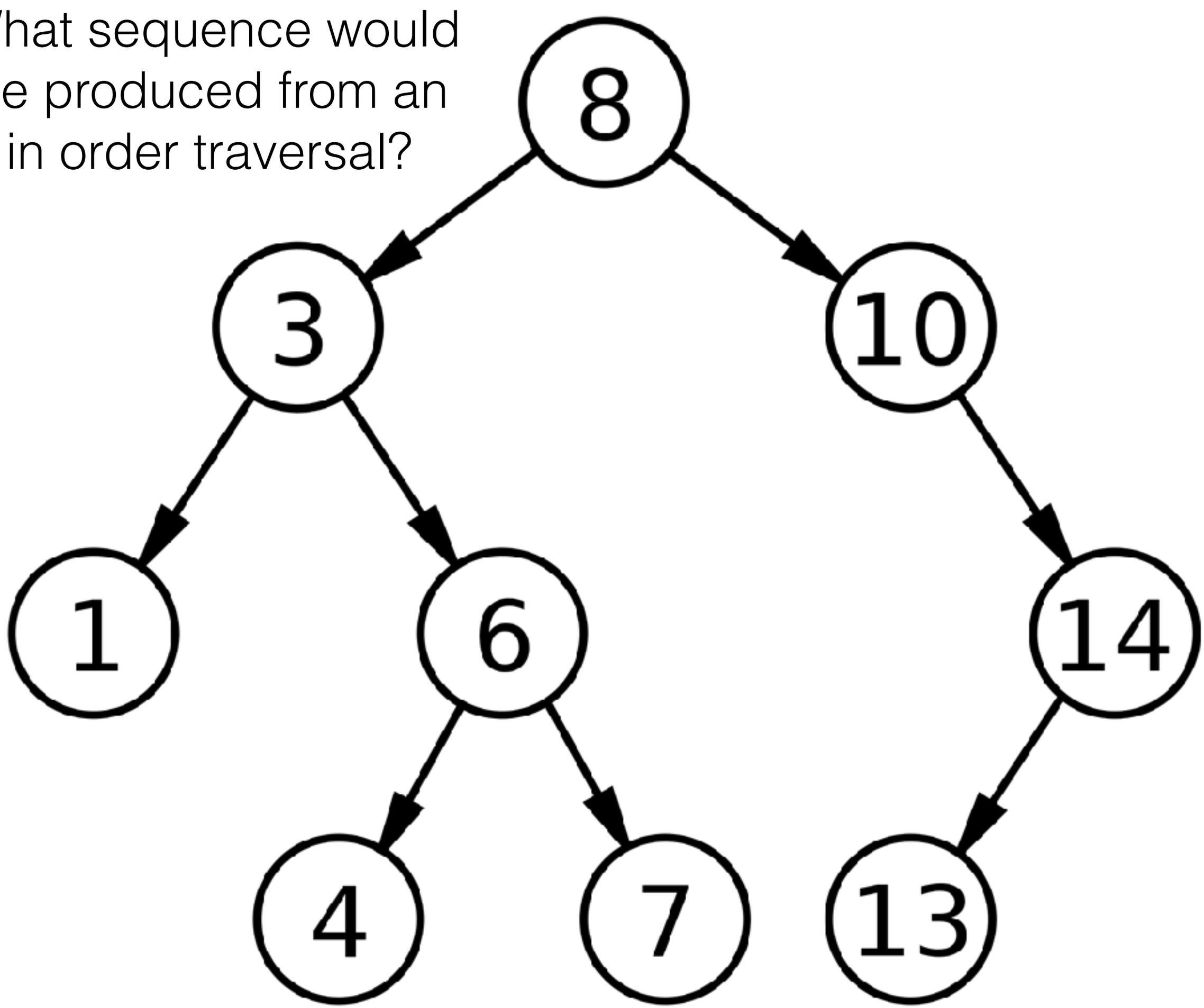
```
>>> DELETE
The DELETE operation is a bit trickier. We
delete the node and then have one of three
cases:
1) If no children - Done
2) If one child - just point parent to child
node
3) If two children - Replace it with the max
element in left subtree, or min element in
right subtree.
```


8
3
10
1
6
14
4
7
13

Tree Traversal

- 3 types:
  - LDR (In order)
  - DLR (Pre order)
  - LRD (Post order)
- Notice L is always read before R
- Name of order refers to where D is in the sequence

What sequence would be produced from an in order traversal?



1, 3, 4, 6, 7,

8,10,13,14

Heap

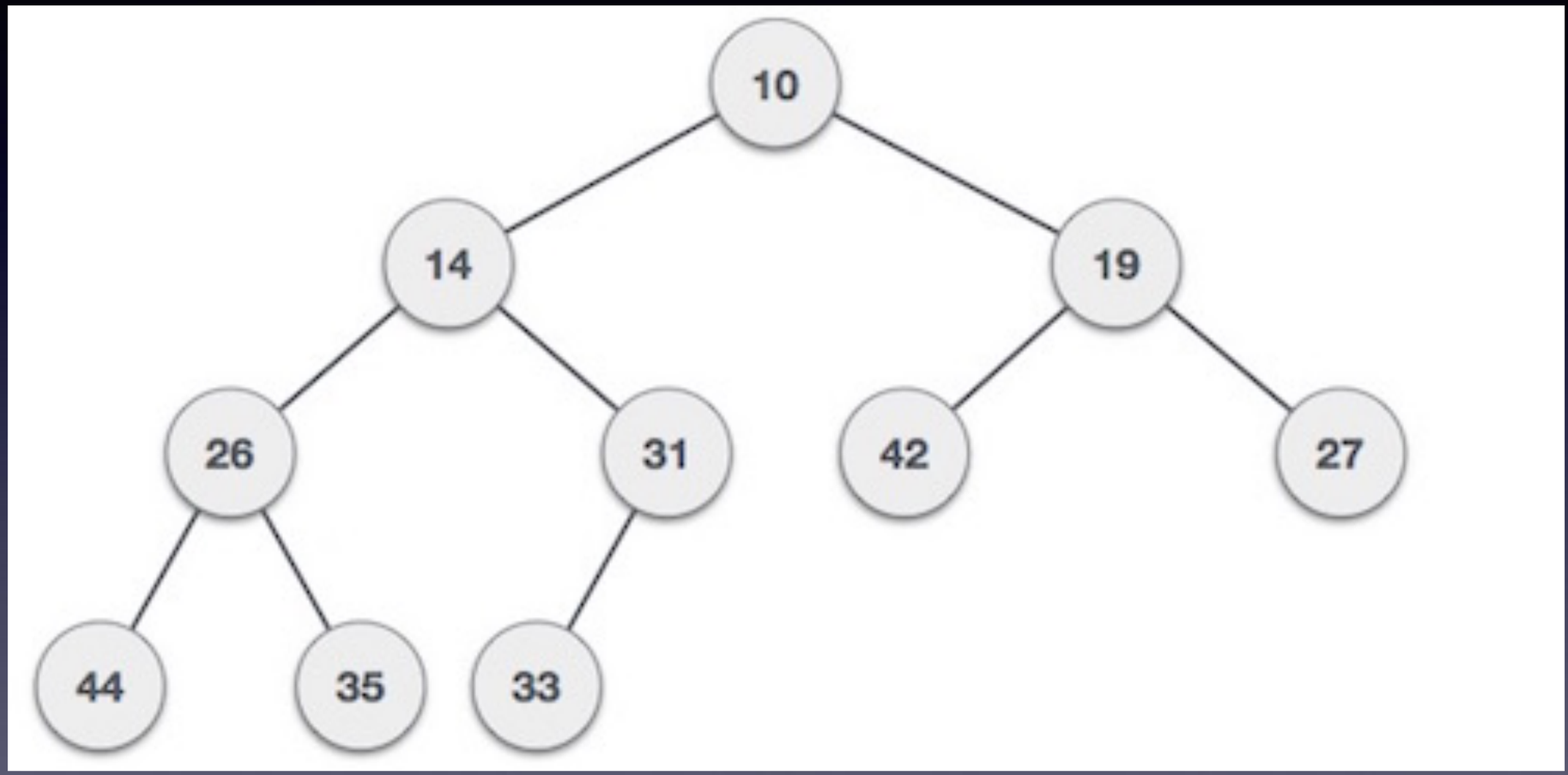
- Binary tree, not BST
- Min heap: all children is >= root
- Max heap: all children is <= root

Heap Properties

We always try to fill each depth level, however this way may invalidate our heap property.

Thus, we either bubble-up or bubble-down in our heaps.

So for example, if we insert 4 under 9 in a min-heap, we would swap 4 and 9. We keep correcting mistakes



Min or max heap?

Recursion

- Whenever a function calls / is dependent upon itself
- Fibonacci sequence: the ith number in the sequence is the sum of the i - 1 st and i - 2 nd number in the sequence

Print Reverse Stack

Given a Stack, s, recursively print the contents of the stack in reverse

```
def revstack(s):
    if(s.is_empty()):
        return
    else:
        ele = s.pop()
        revstack(s)
        print(ele)
        return
```

Tree traversal using recursion

- Linked list type data structures are great for recursion
- Each node is the same except for data

LDR with recursion

```
def sumldr(root):

    if root is None: return 0

    return root.data + sum(root.left) + sum(root.right)
```

Finding depth with recursion

```
def maxdepth(root):
    if root is None: return 0 # base case

    return 1 + max(maxdepth(root.left), maxdepth(root.right))
```

Average of BST

Given a BSTNode, n, recursively find the average of the values found in the tree rooted at n.

avgBST(n —> BSTNode):

Average of BST

```
def avgBSTHelper(n, total=0, counter=0):
    if(not n.left and not n.right):
        total = n.value
        counter = 1
    else:
        total += n.value
        counter += 1
    if(n.left):
        valuePair = avgBSTHelper(n.left)
        total += valuePair[0]
        counter += valuePair[1]
    if(n.right):
        valuePair = avgBSTHelper(n.right)
        total += valuePair[0]
        counter += valuePair[1]
    return total, counter

def avgBST(n):
    valuePair = avgBSTHelper(n)
    total = valuePair[0]
    counter = valuePair[1]
    return total/counter
```