CS151 Intro to Data Structures

Tree Traversals

Announcements

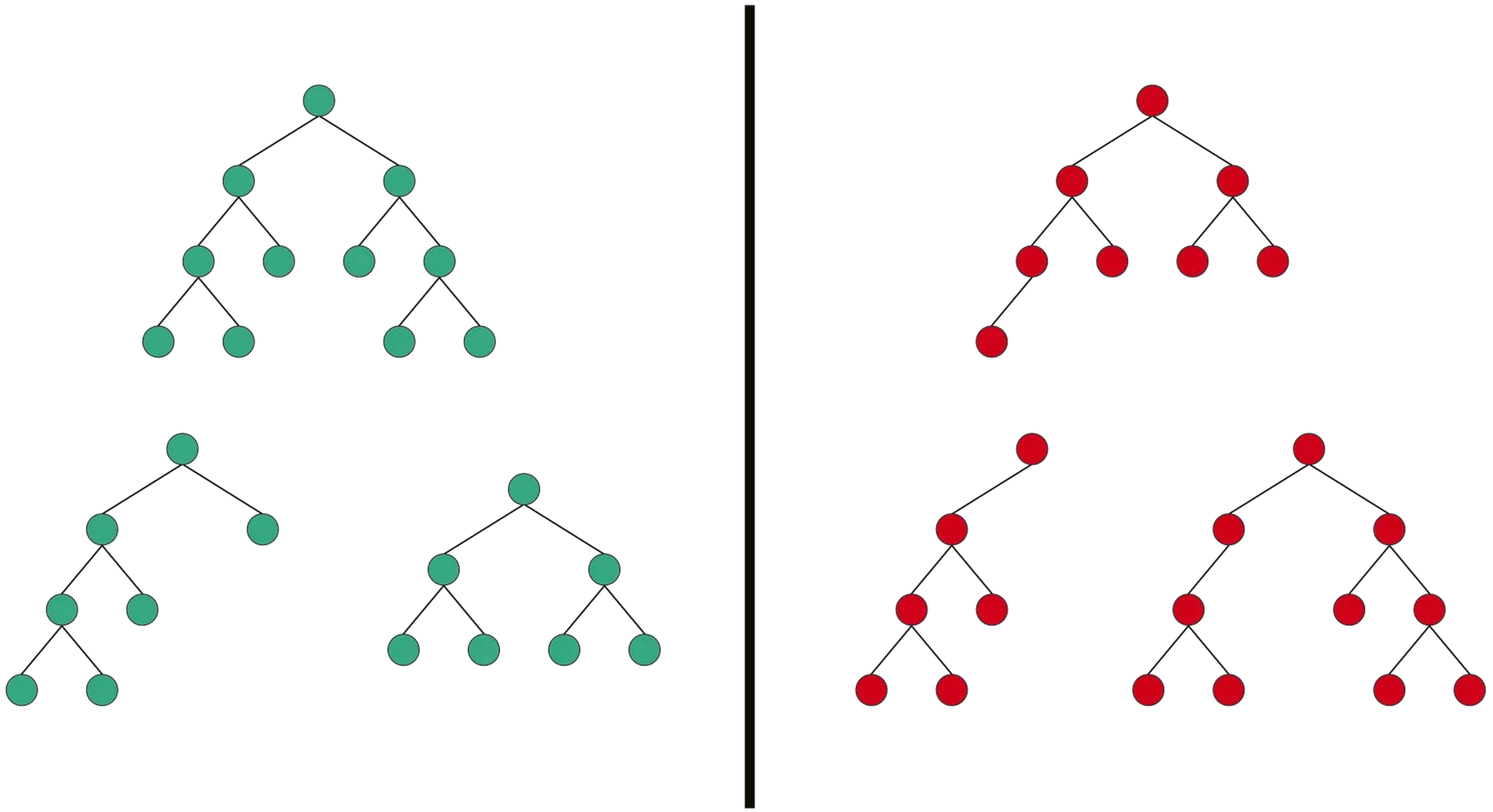
- HW04 due tomorrow
  - Upload ExapndableArray to gradescope
  - **make ArrayList.MyListIterator public**

Outline

- Trees:
  - BST review
  - Binary Search Tree Traversals
    - In order
    - Pre order
    - Post order

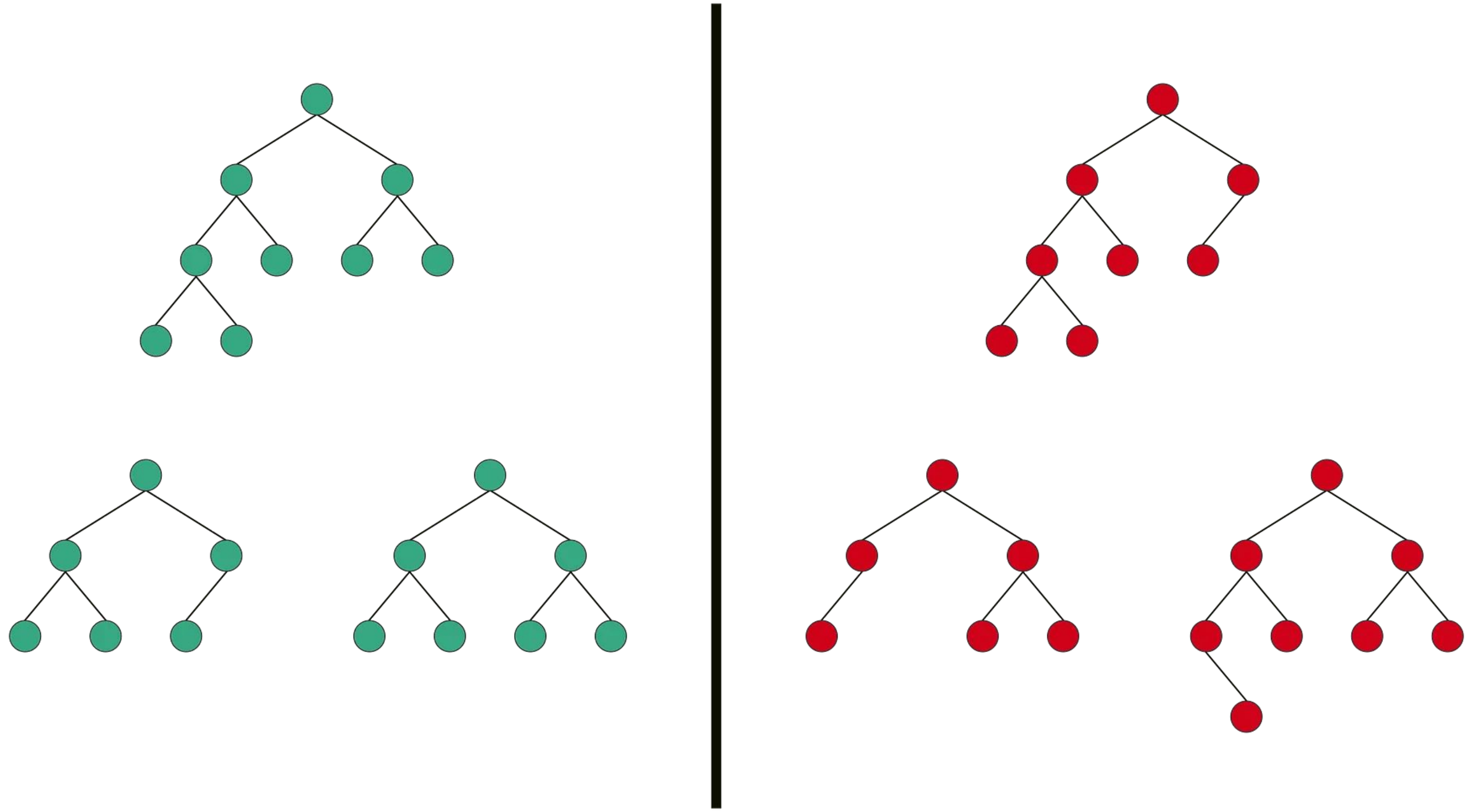
Types of Binary Trees

A binary tree is **full** (proper) if every node has 0 or 2 children



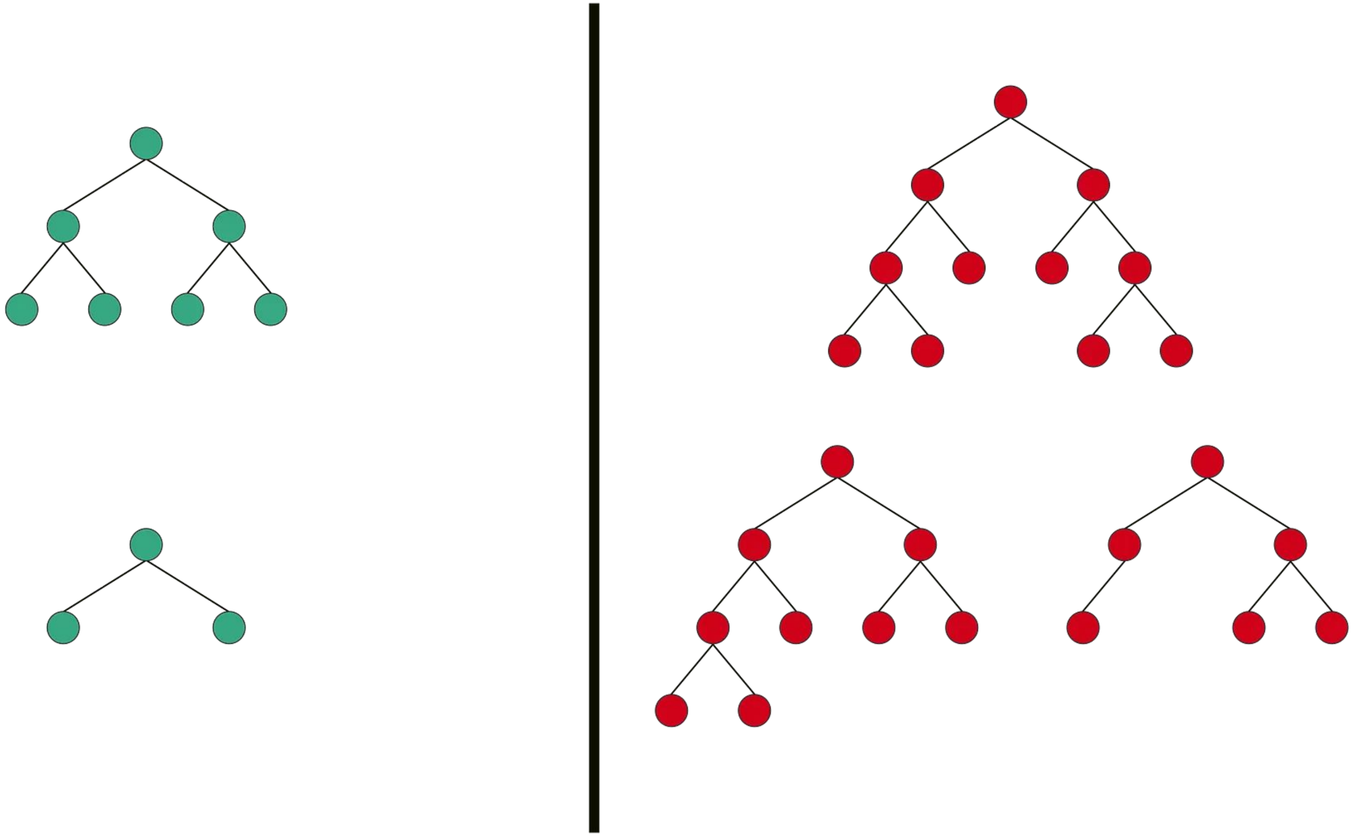
Types of Binary Trees

A binary tree is **complete** if all levels are completely filled (zero or two children) *except possibly the last level* and all the nodes are as left side as possible



Types of Binary Trees

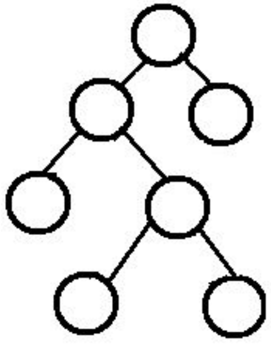
A binary tree is **perfect** if all internal (non-leaf) nodes have 2 children and all the leaf nodes are at the same depth or same level.



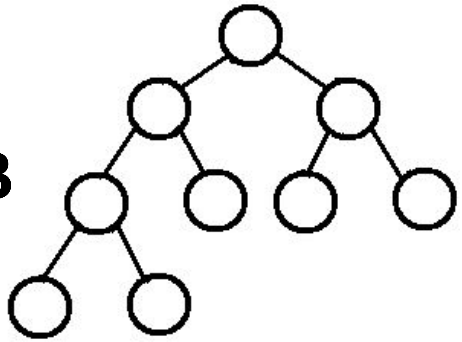
Types of Binary Trees

Full, complete, proper?

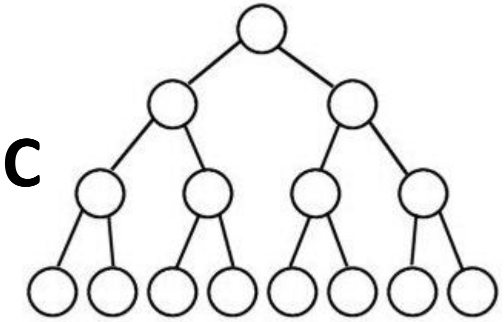
A



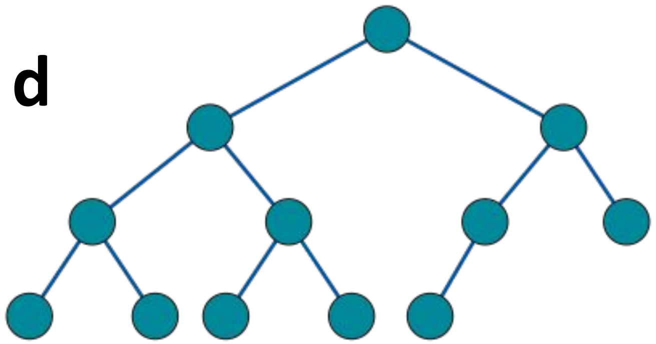
B



C



d



Types of Binary Trees

A binary tree is **full** (proper) if every node has 0 or 2 children

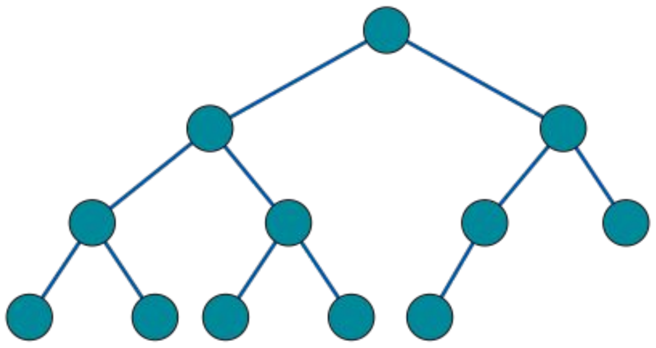
A binary tree is **complete** if all levels are completely filled (zero or two children) *except possibly the last level* and all the nodes are as left side as possible

A binary tree is **perfect** if all internal (non-leaf) nodes have 2 children and all the leaf nodes are at the same depth or same level.

Q1: Is every full binary tree a complete binary tree?

Q2: Is every complete binary tree a full binary tree?

Q3: Is every perfect binary tree a full binary tree?



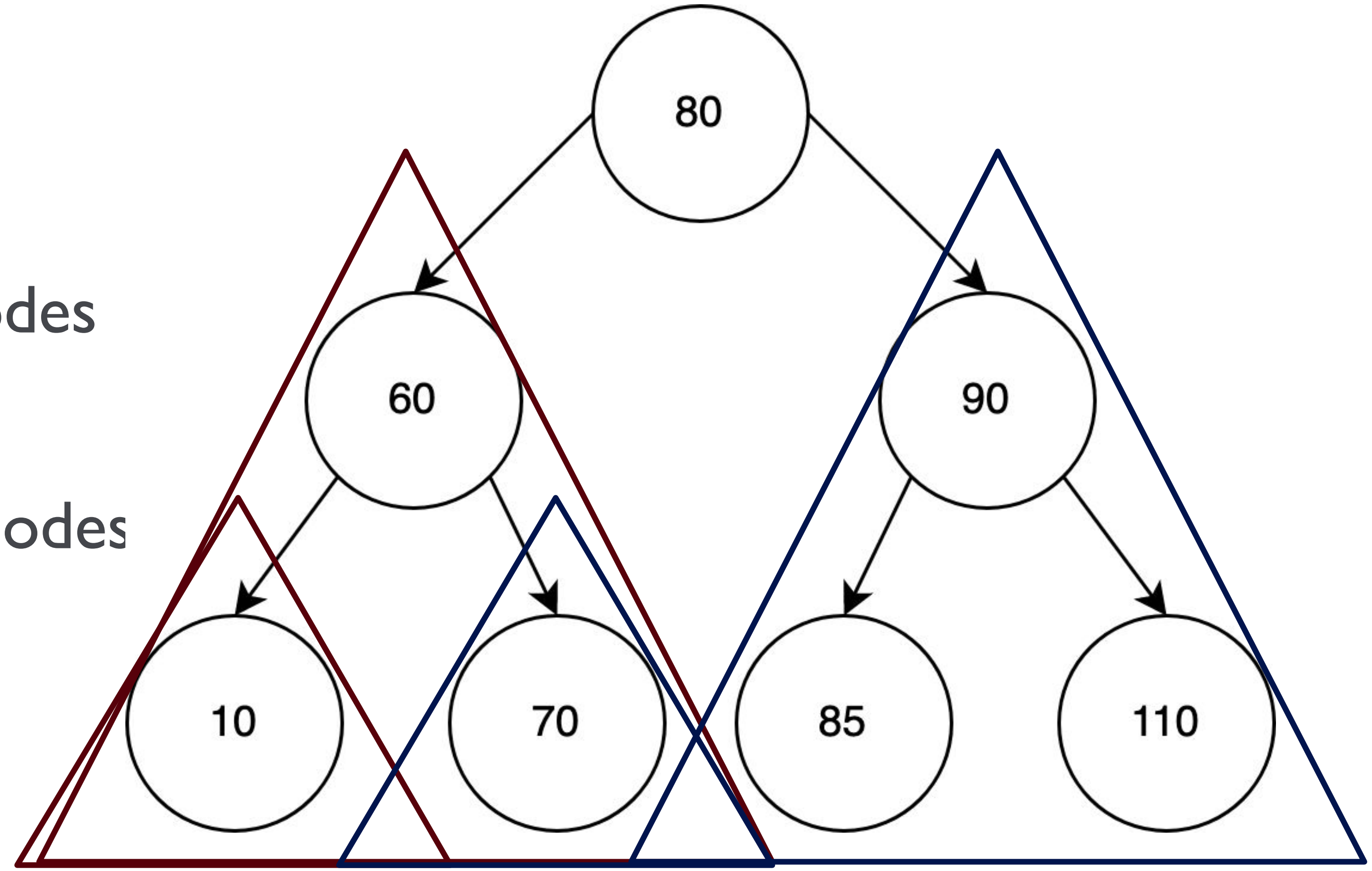
complete but not full

Binary Search Trees

Definition:

At **each node** with value **k**

- Left subtree contains only nodes with value **lesser** than **k**
- Right subtree contains only nodes with value **greater** than **k**
- Both subtrees are a **binary search tree**

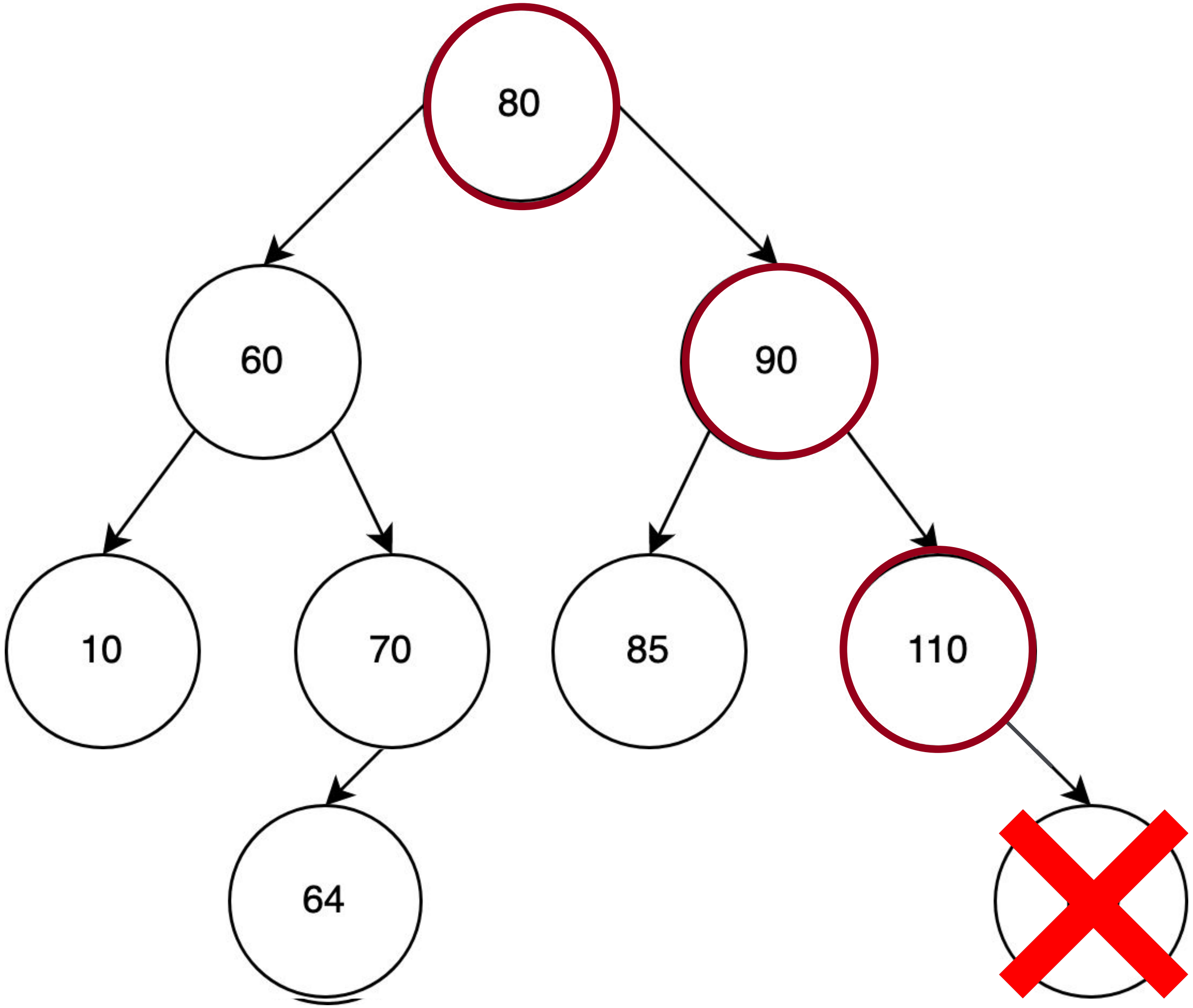


Remove review

Deleting a leaf node

Deletion must maintain the properties of a BST!

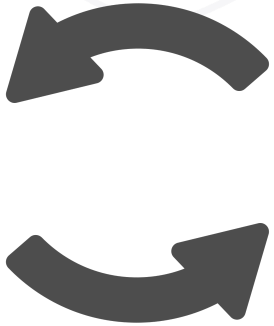
Delete: 150

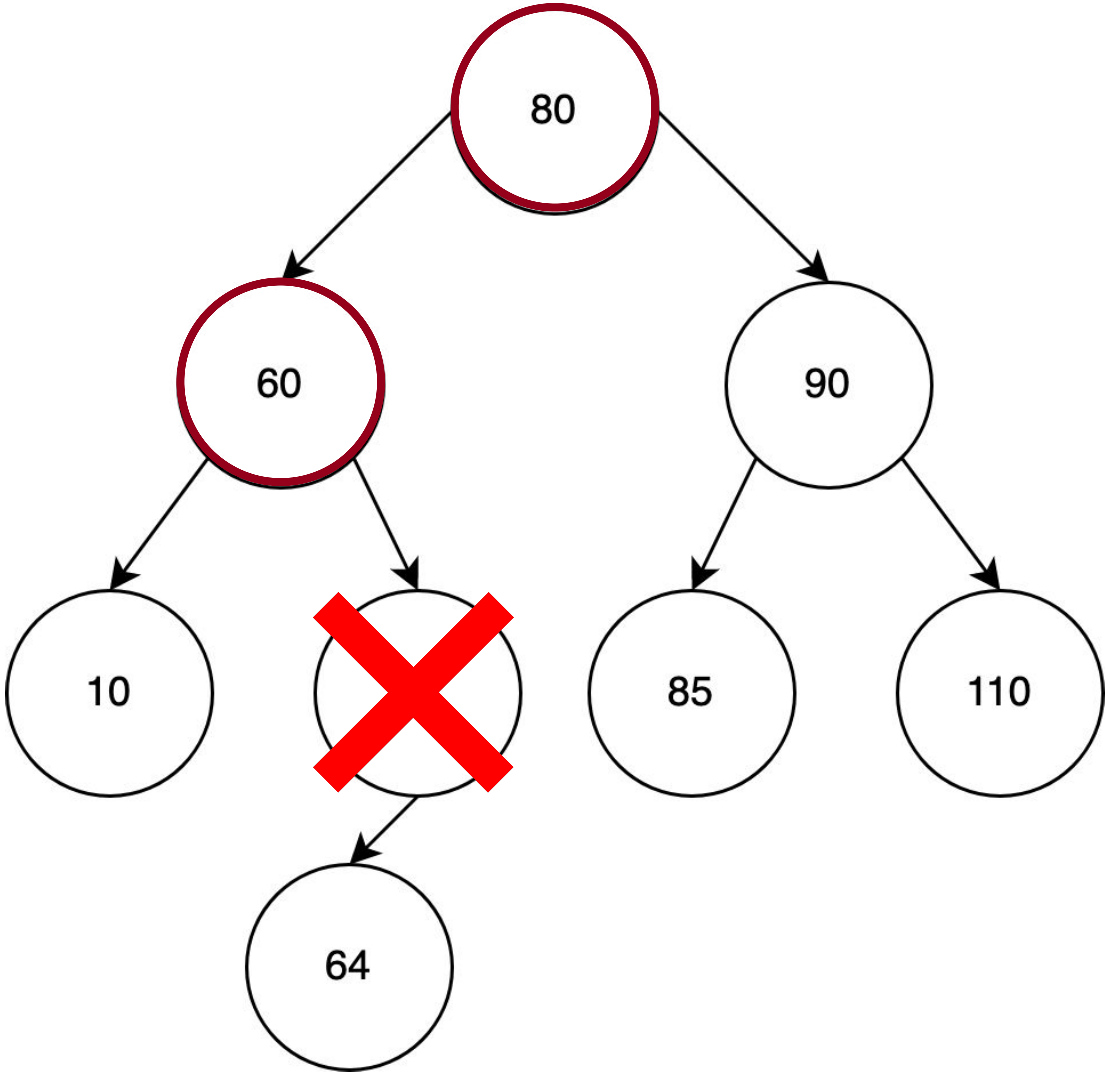


Deleting a node with one child

Deletion must maintain the properties of a BST!

Delete: <u>70</u>





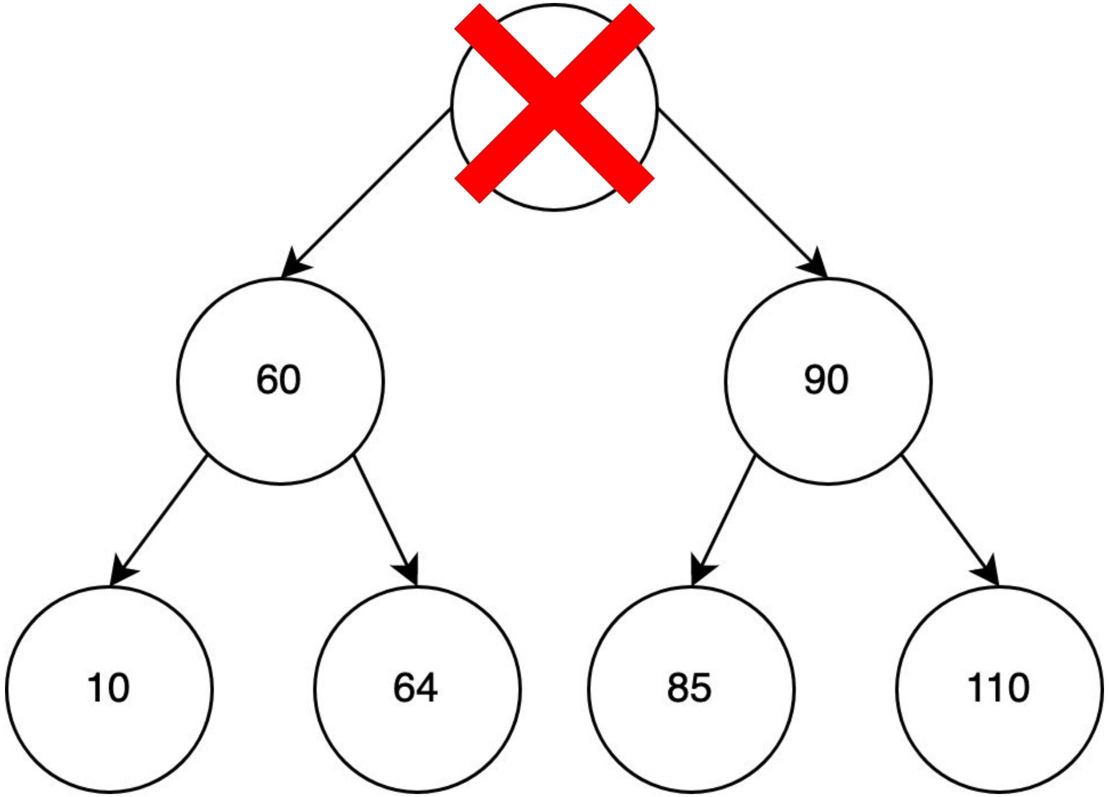
Deleting a node with 2 children

Deletion must maintain the properties of a BST!

Delete: <u>80</u>

At **each node** with value **k**

- Left subtree contains only nodes with value **lesser** than **k**
- Right subtree contains only nodes with value **greater** than **k**
- Both subtrees are a **binary search tree**



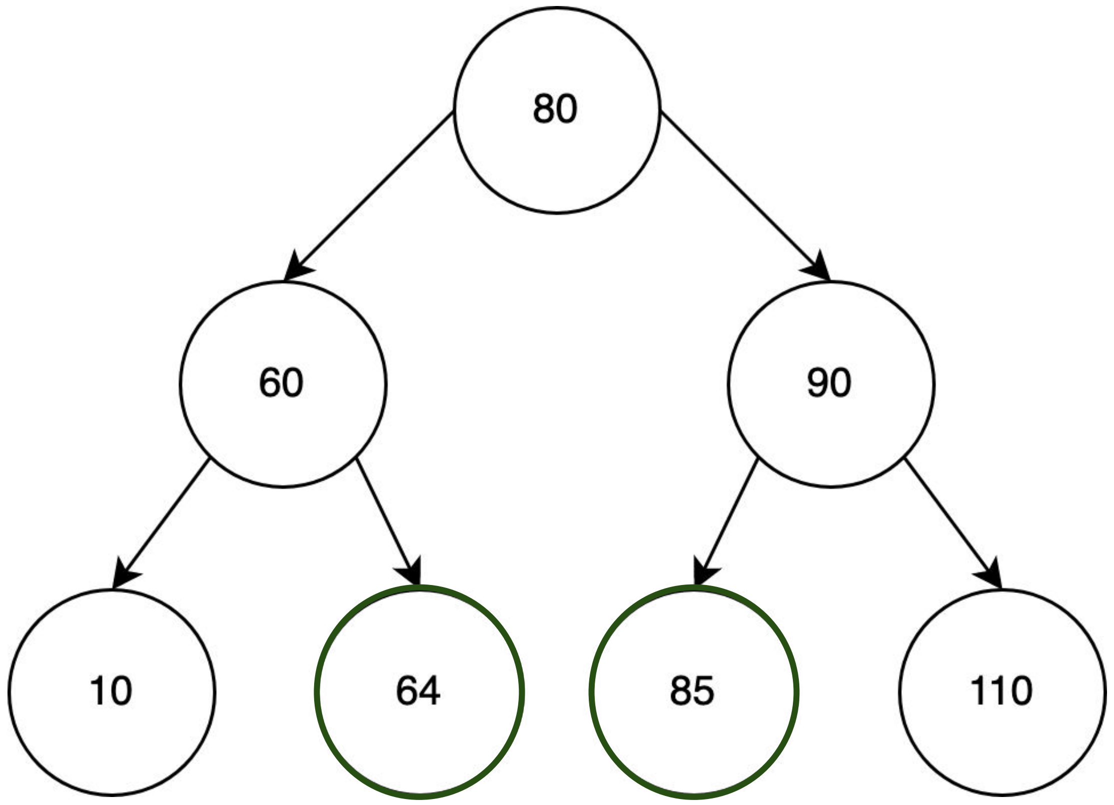
Deleting a node with 2 children

Deletion must maintain the properties of a BST!

Delete: 80

Replace deleted node with either:

1. Smallest value in right subtree
2. Largest value in left subtree



Binary Search Trees: Deletion

Complexity?

Case 1: Removing a **leaf node**

O(log n)

Case 2: Removing a **node with one child**

O(log n)

Case 3: Removing a **node with two children**

O(log n)

Summary

Takeaways:

Binary search trees are an efficient data structure for search

For a *balanced* binary search tree:

- Search: O(log n)
- Insertion: O(log n)
- Removal: O(log n)

Tree Traversals

Binary Tree Traversals

Traversal visits all nodes in a tree in some order

Inorder:

left subtree, current, right subtree

will print "in order" (increasing values)

Preorder:

current, left subtree, right subtree

Postorder:

left subtree, right subtree, current

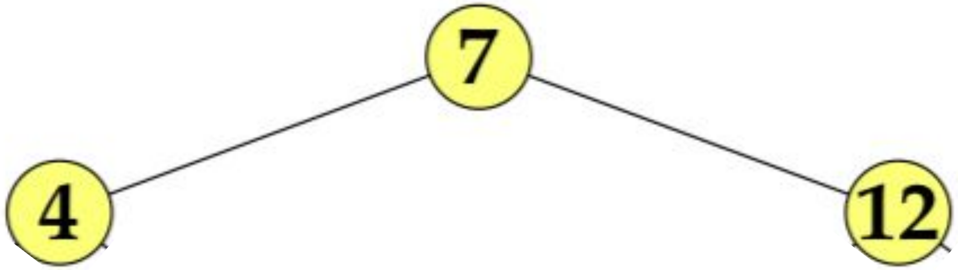
In Order Traversal

1. Move left until you reach a node without a left child
2. Print the current node
3. Move right

Inorder Example 1

What would the in-order traversal be here?

left subtree, current, right subtree



Inorder Example 2

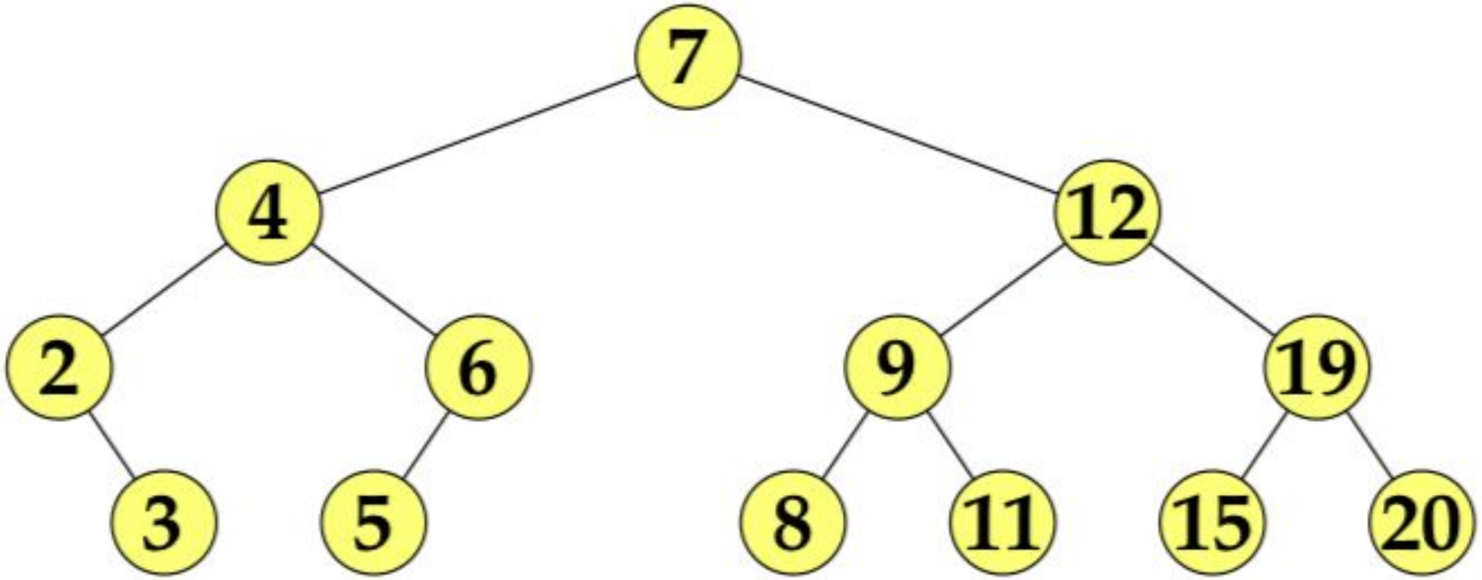
- Larger tree (Height > 1)

- **Process entire left subtree first**
  - **bottom most left node**
  - **current**
  - **bottom most right node**

Inorder Example 2

What would the in-order traversal be here?

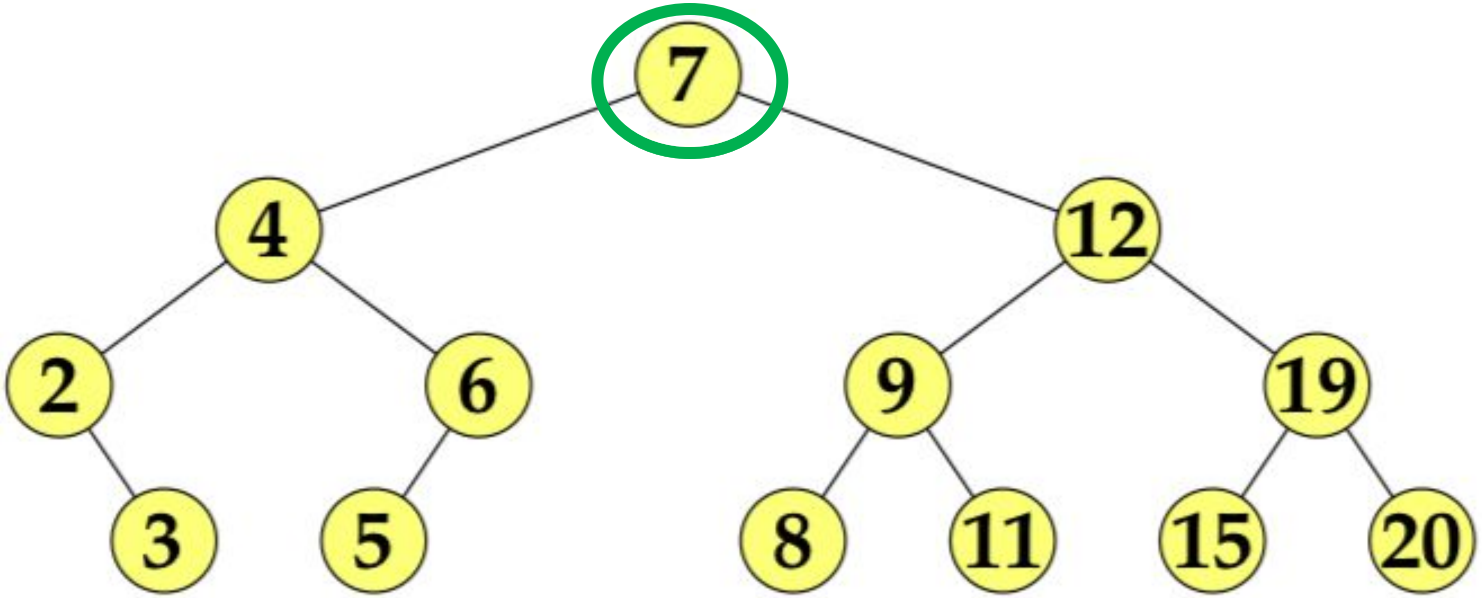
left subtree, current, right subtree



Inorder

What would the in-order traversal be here?

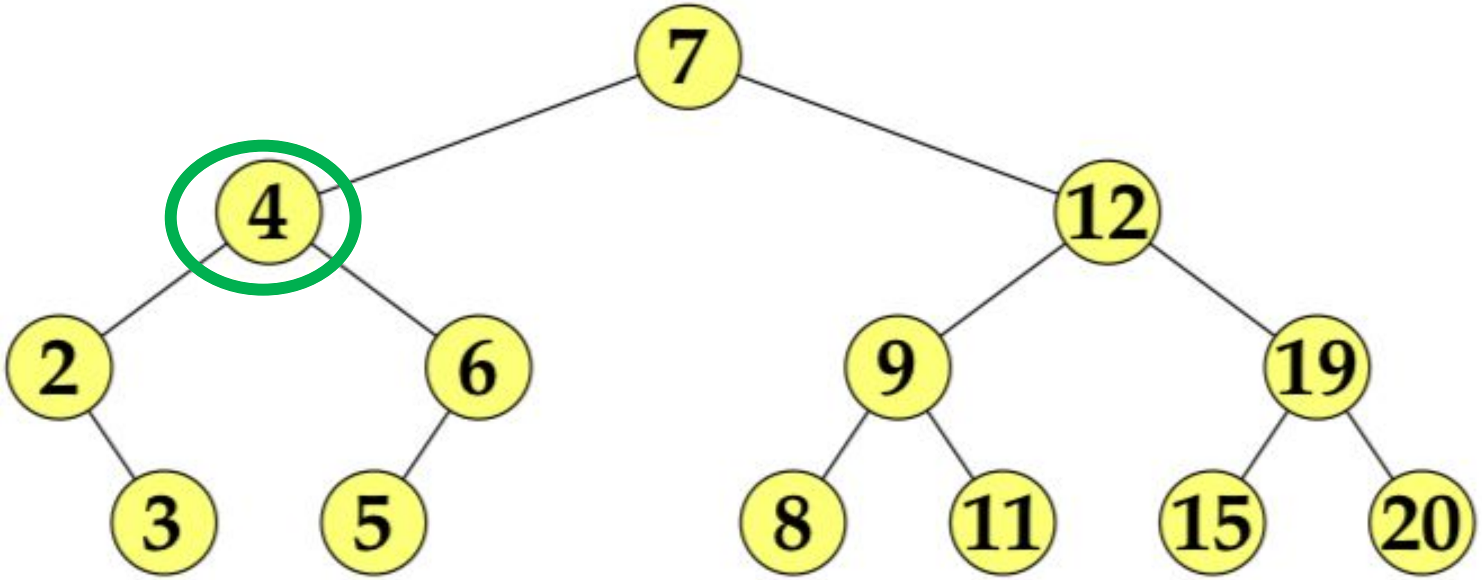
left subtree, current, right subtree



Inorder

What would the in-order traversal be here?

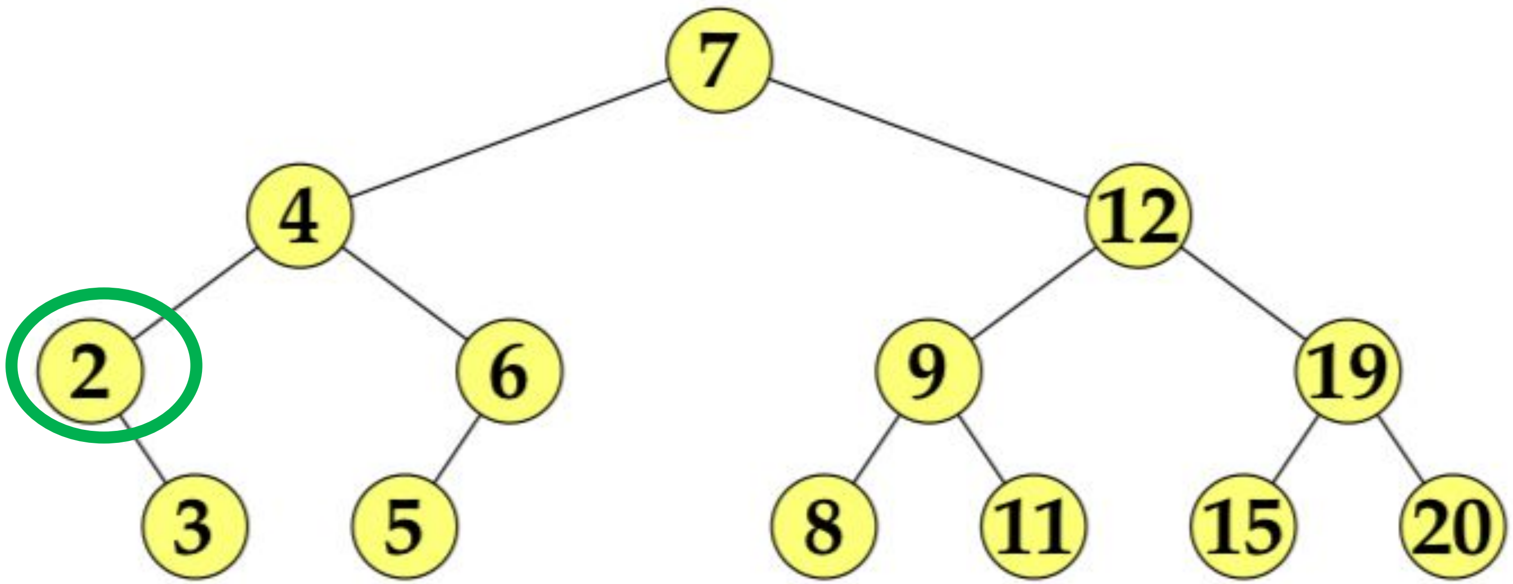
left subtree, current, right subtree



Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

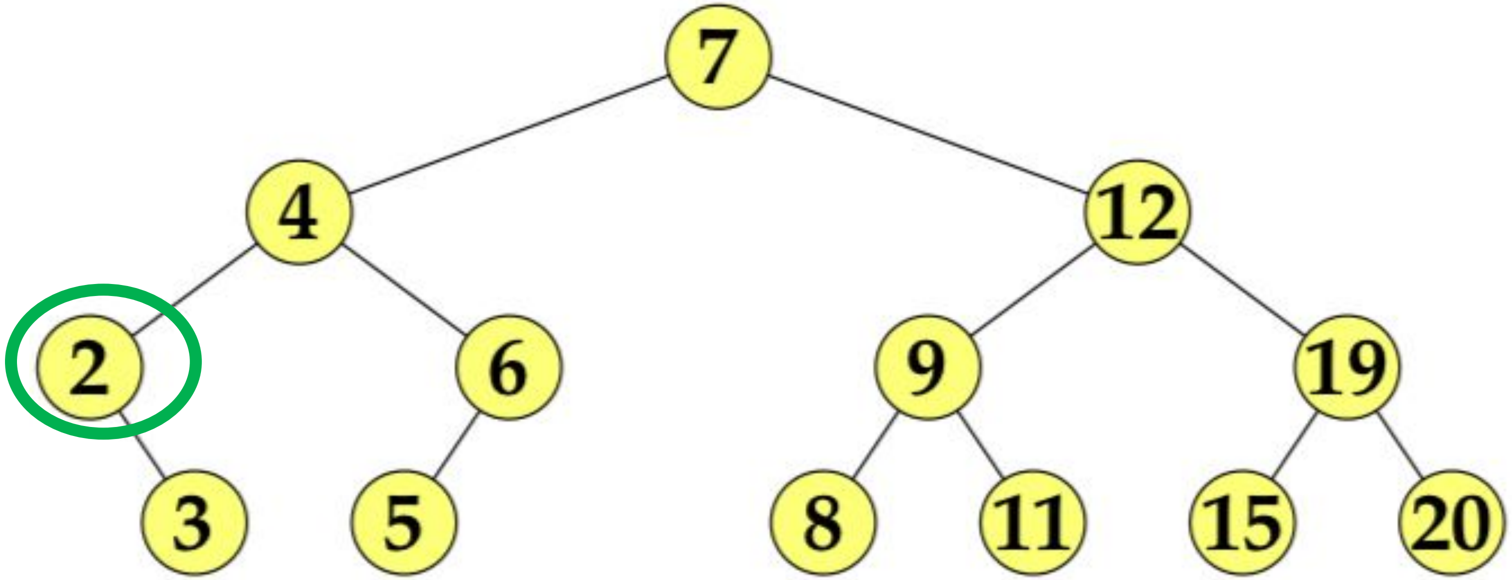


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2,

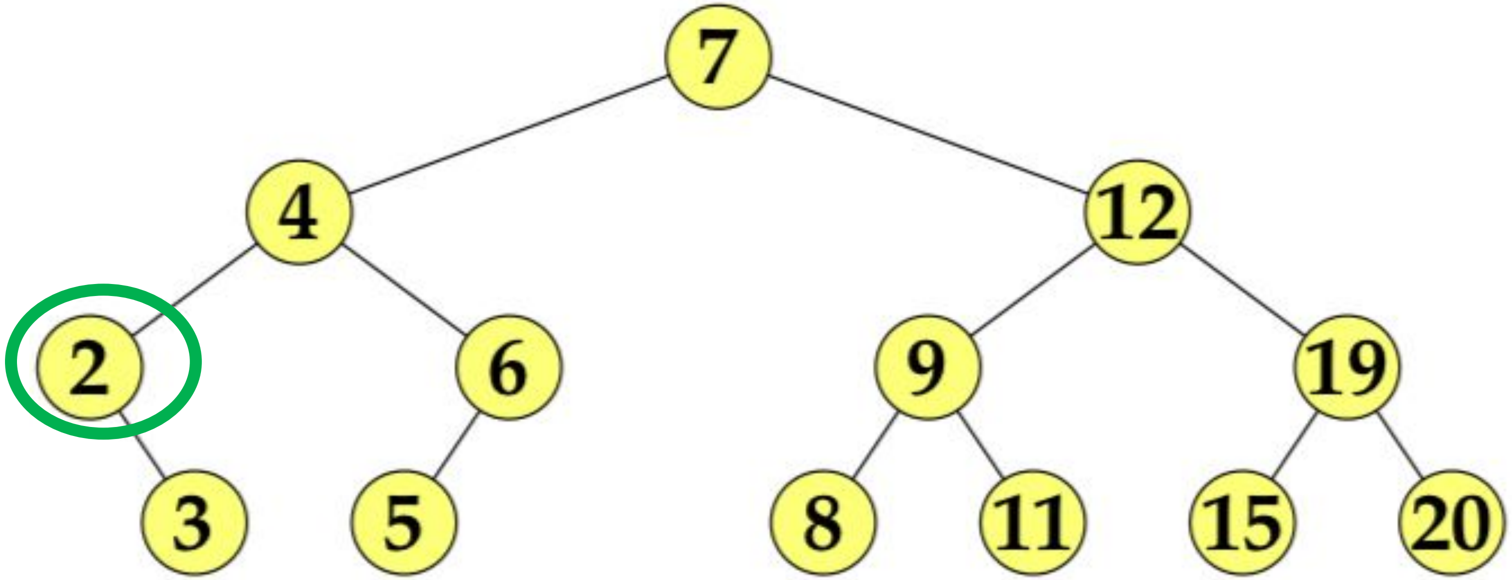


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2,

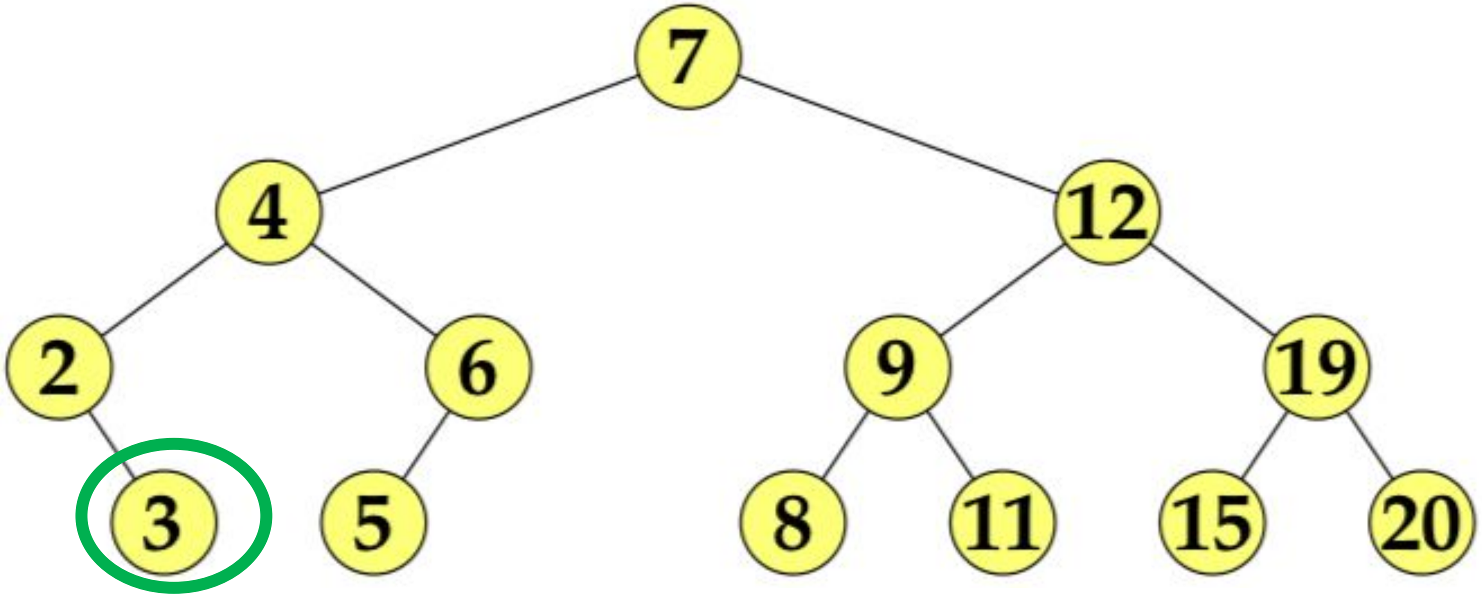


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2,

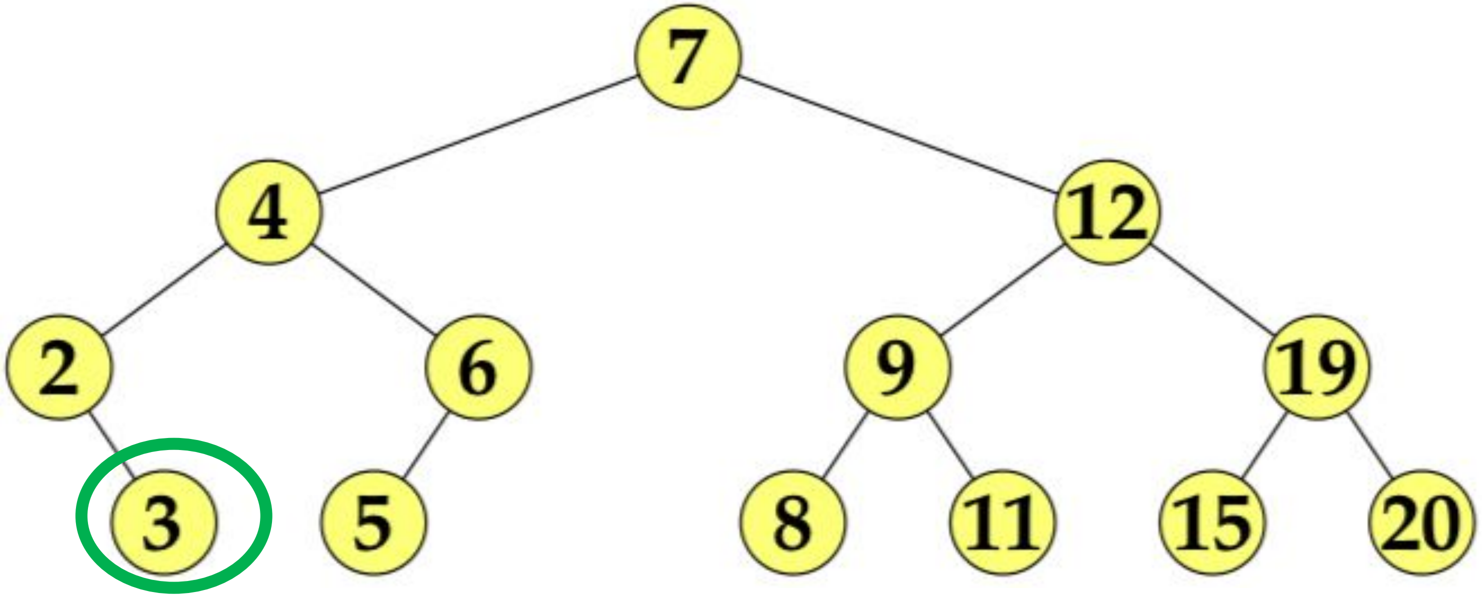


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3

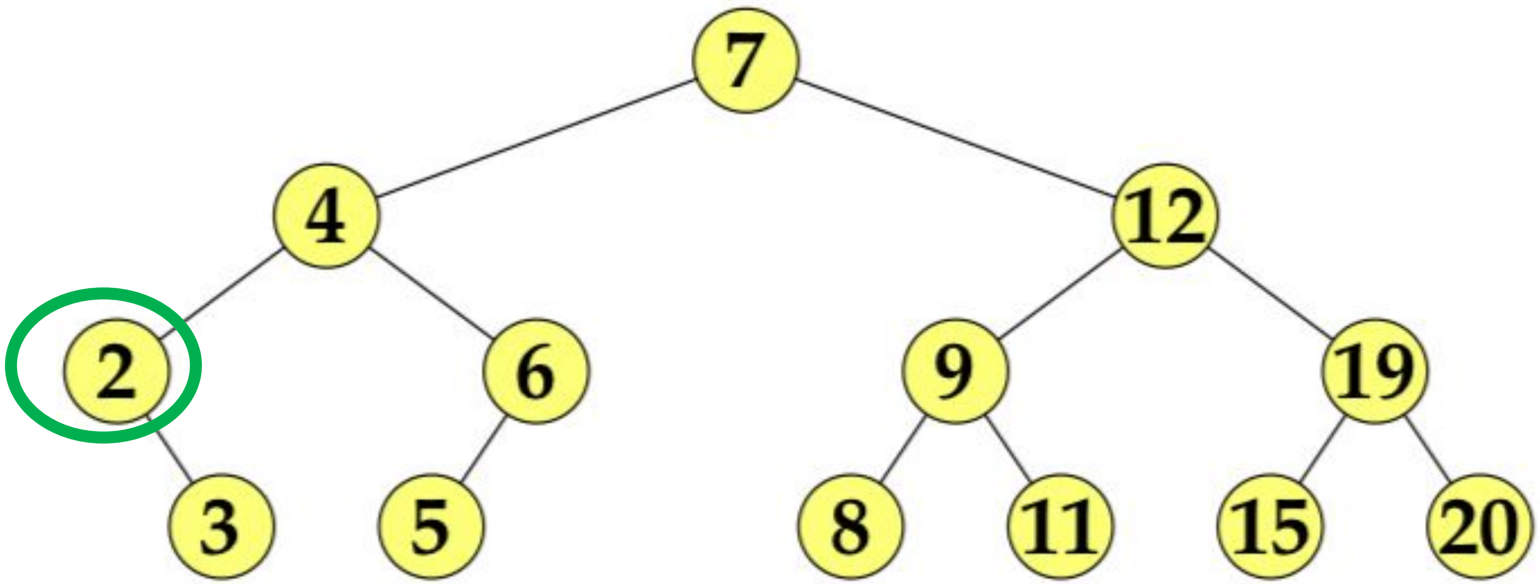


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3

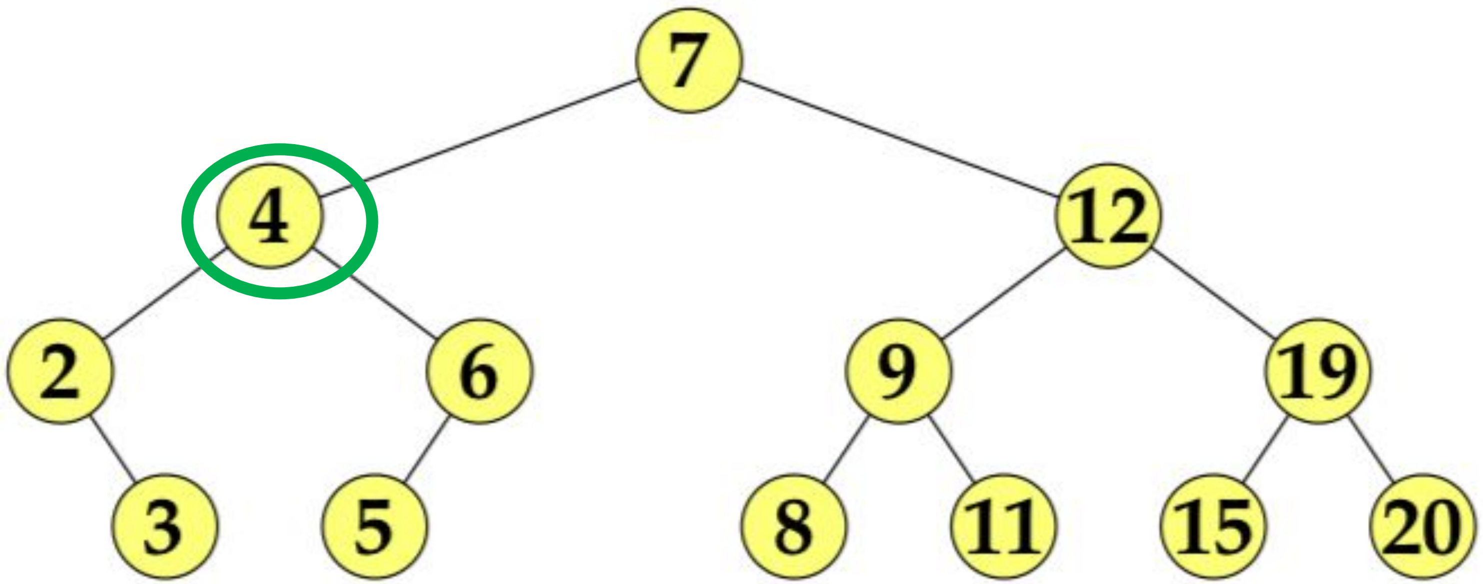


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3

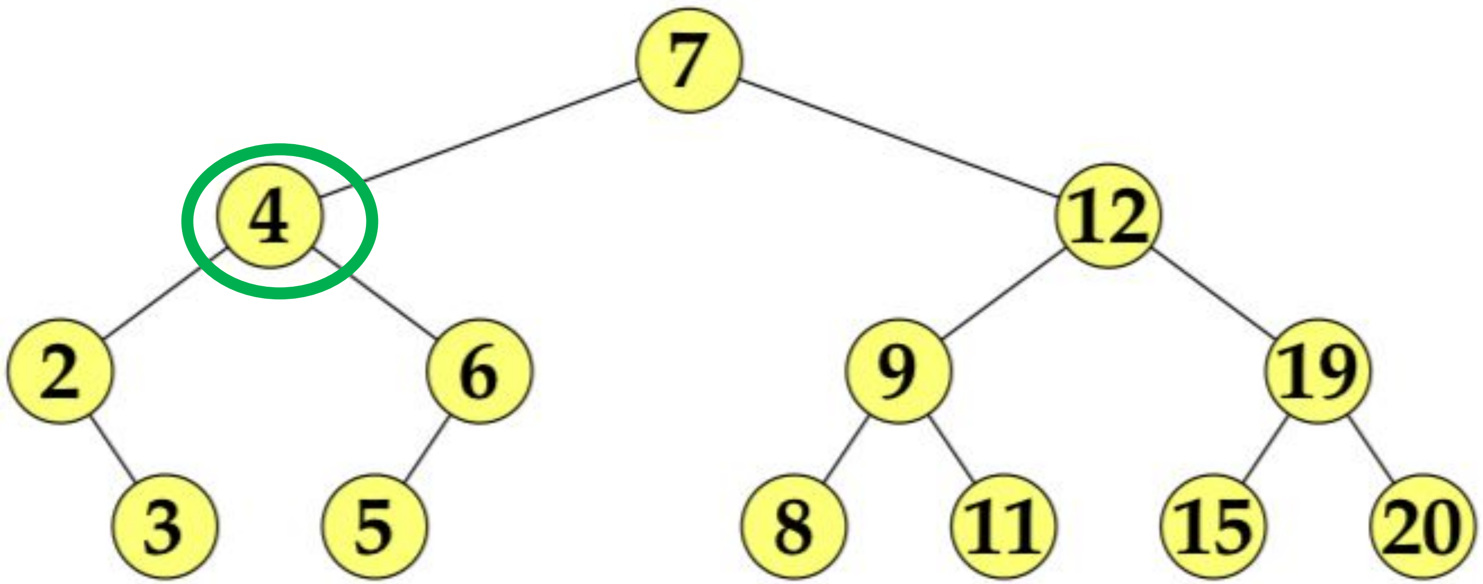


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4

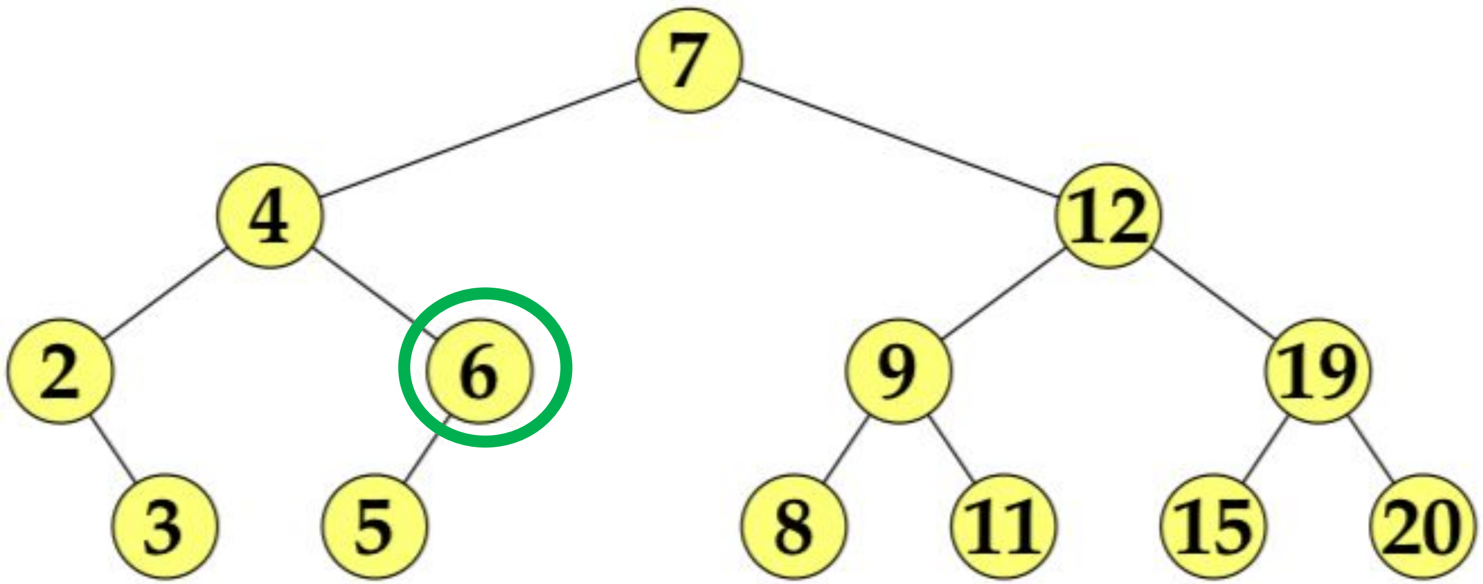


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4

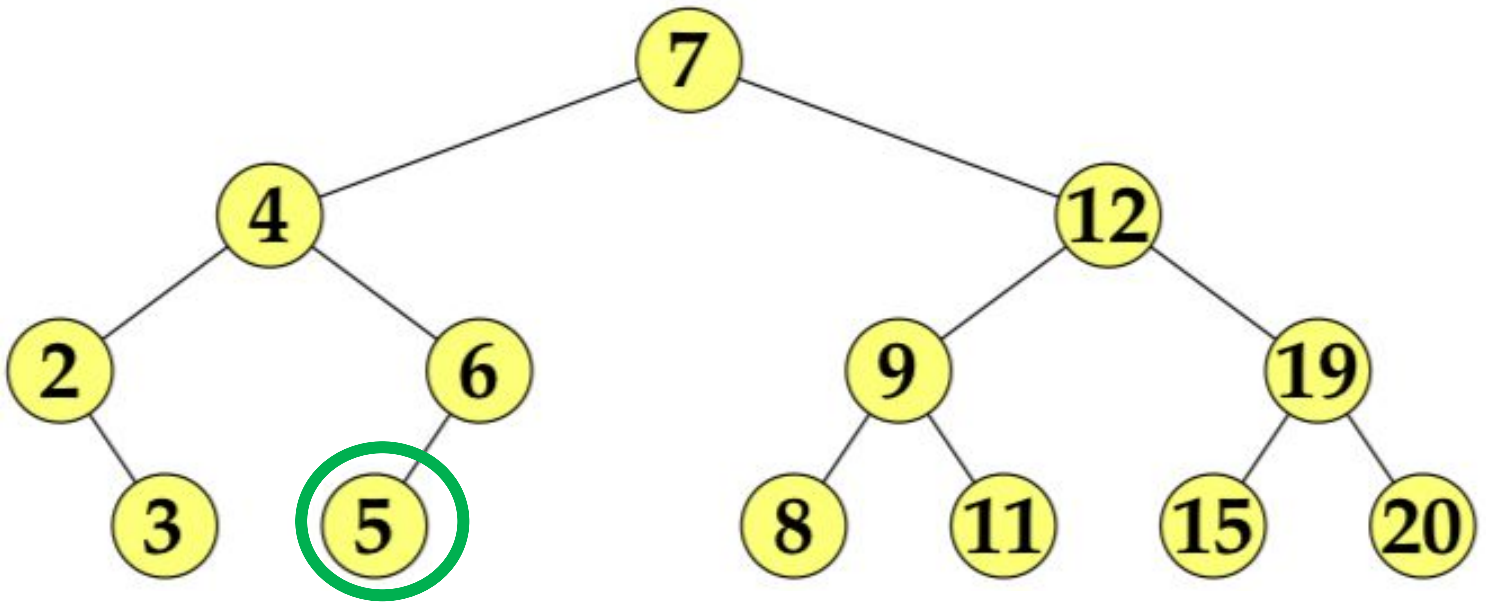


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4

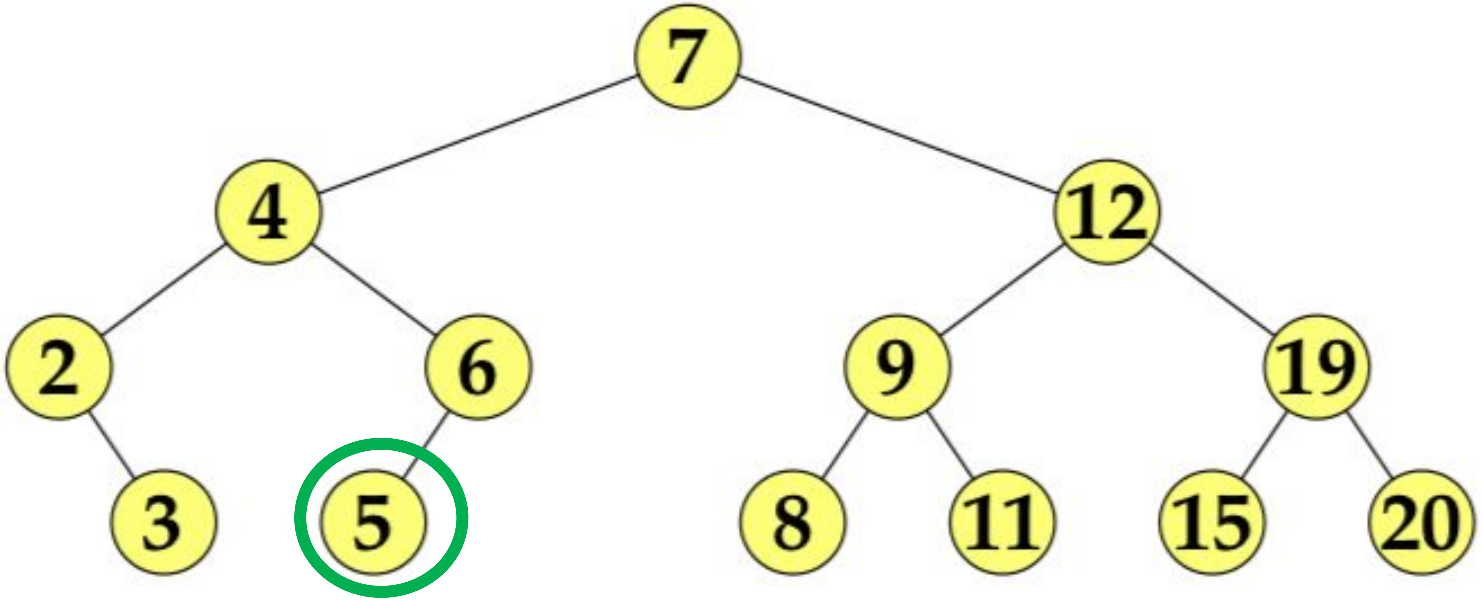


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5

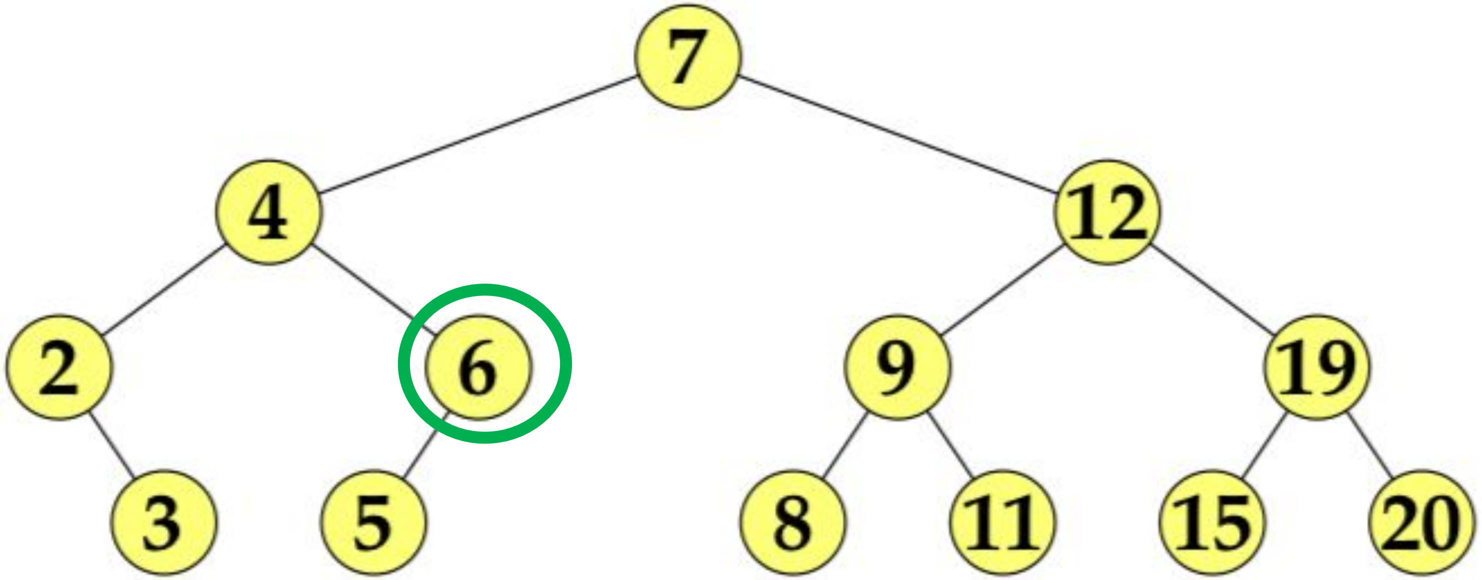


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5

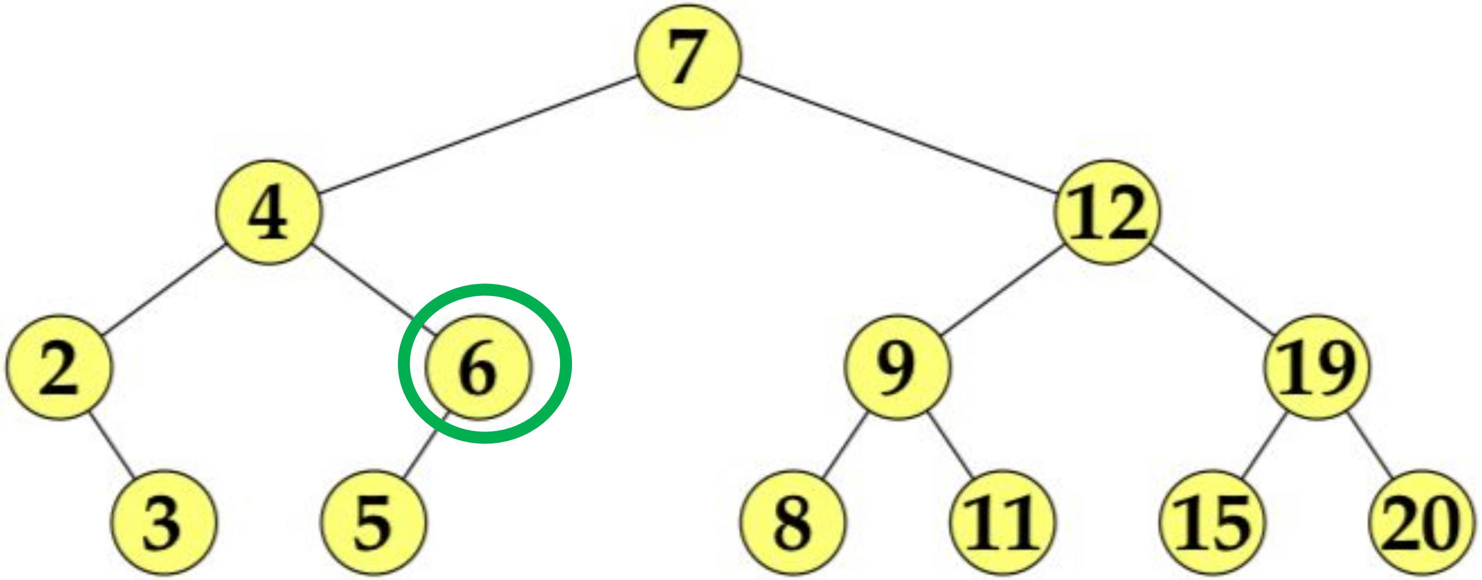


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6

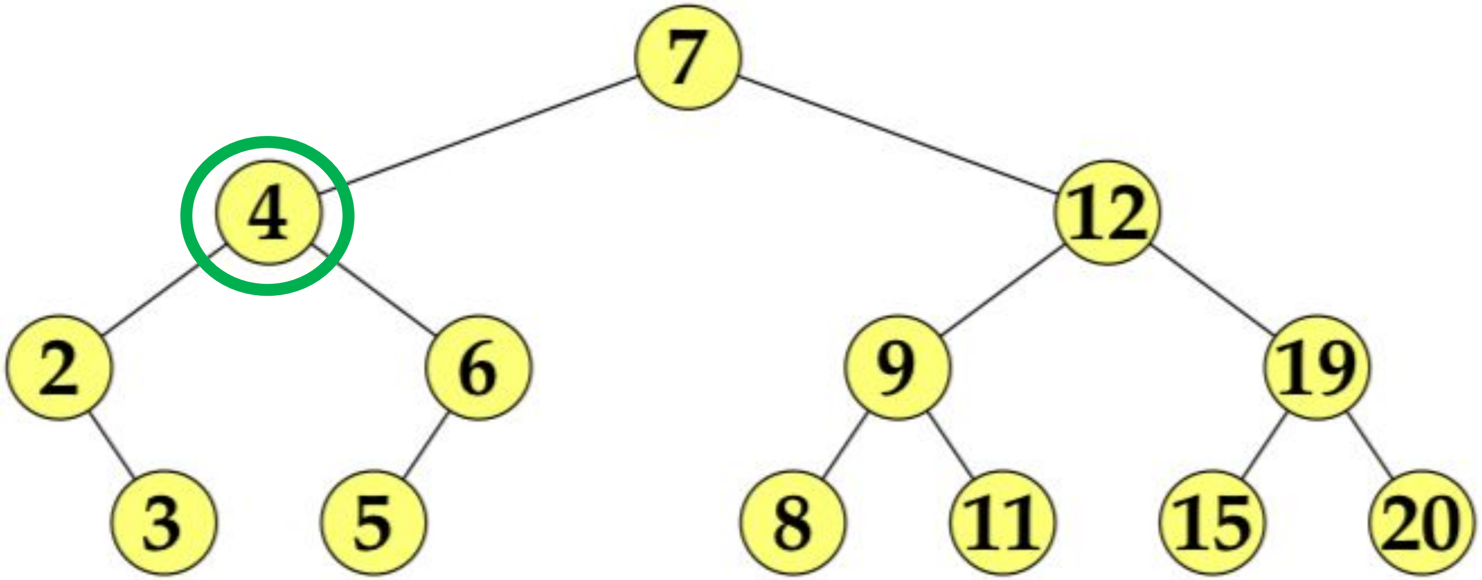


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6

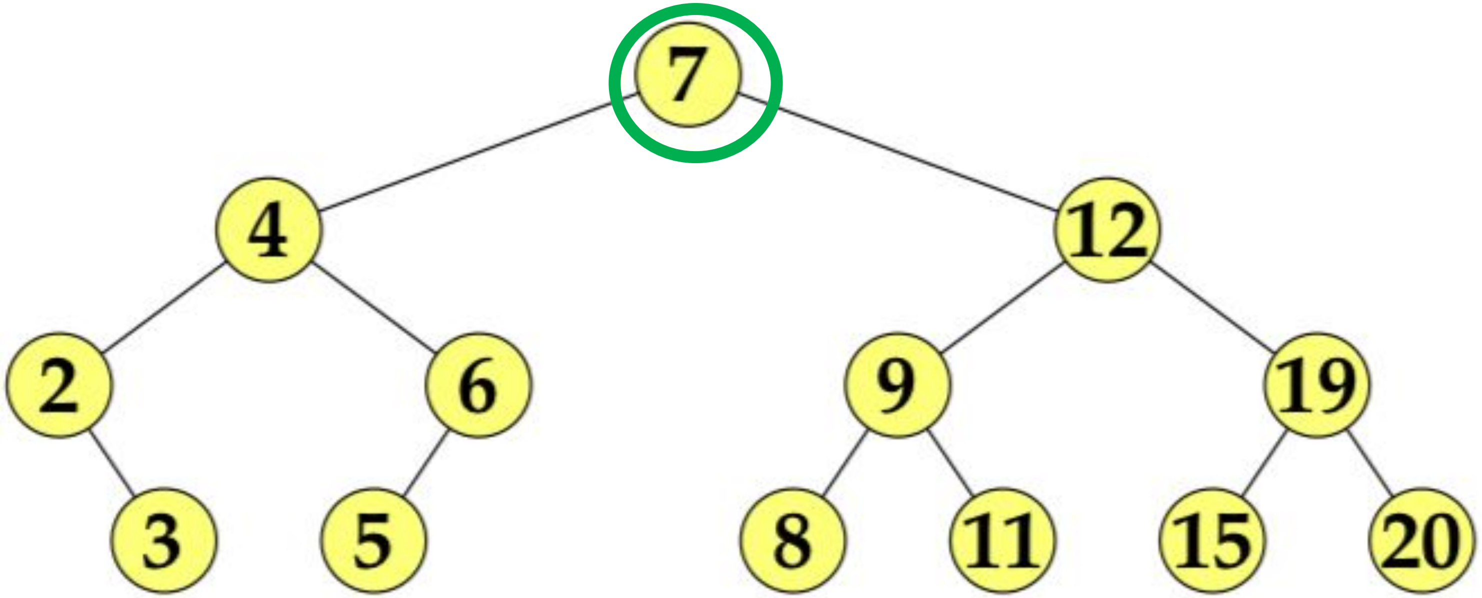


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6

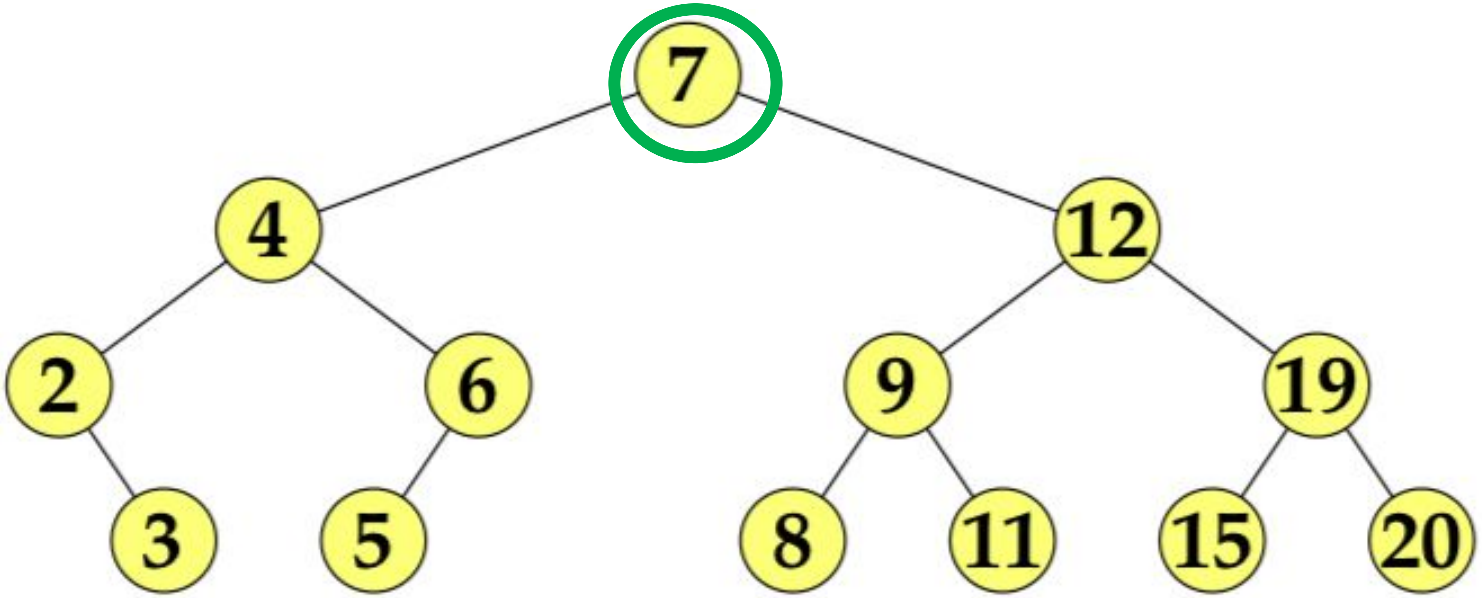


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7

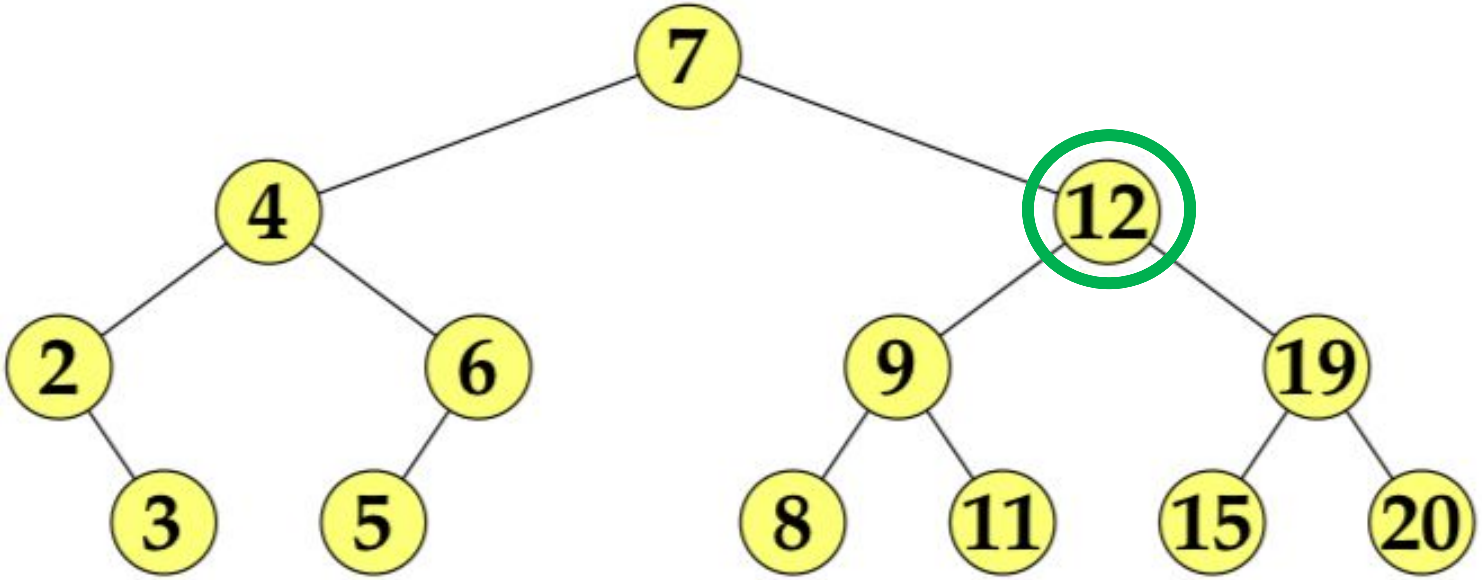


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7

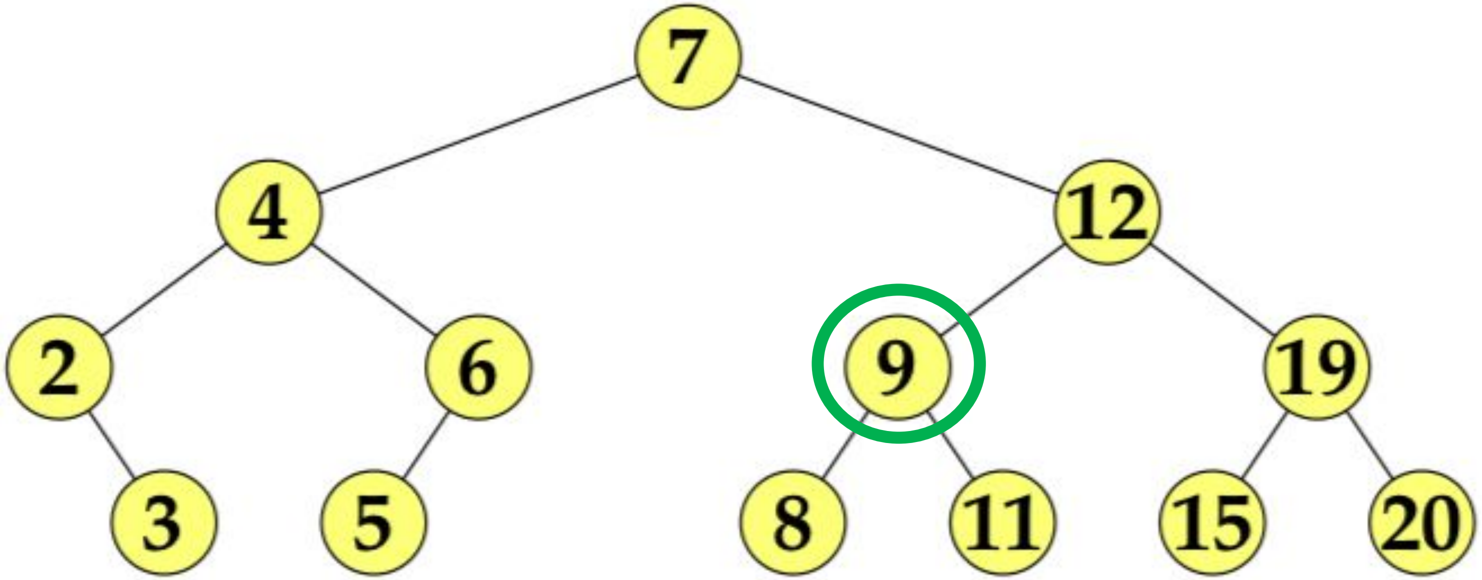


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7

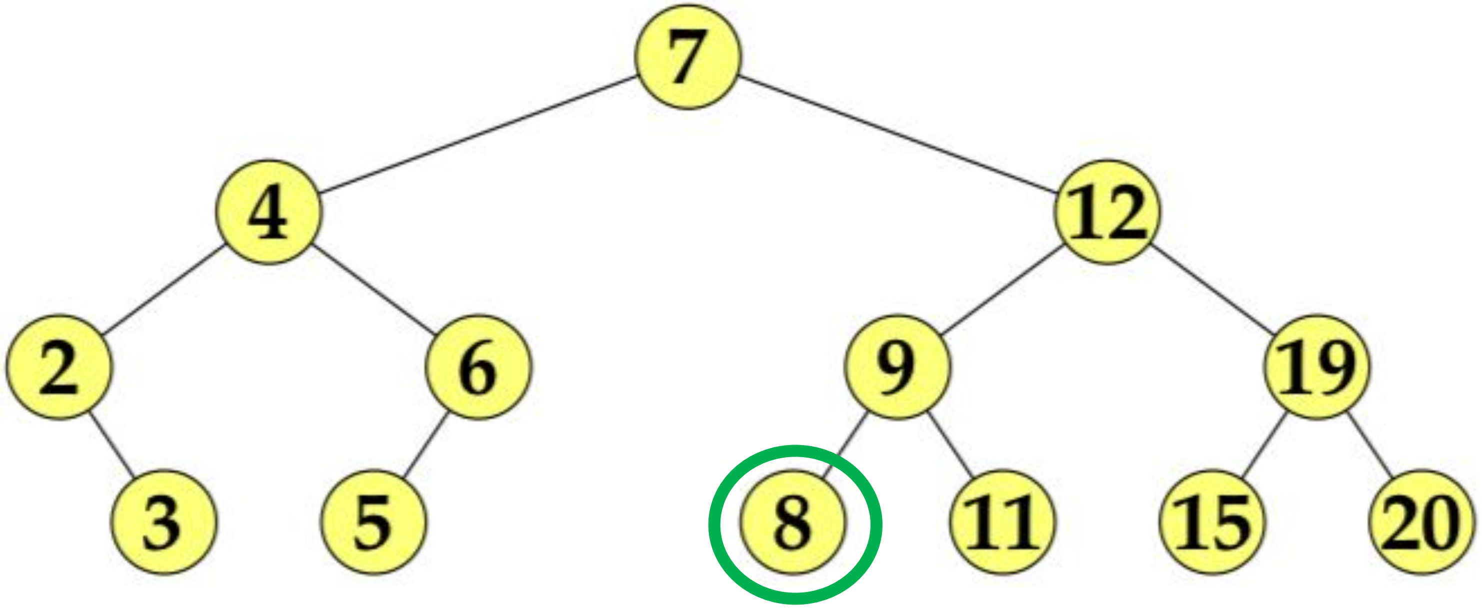


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7

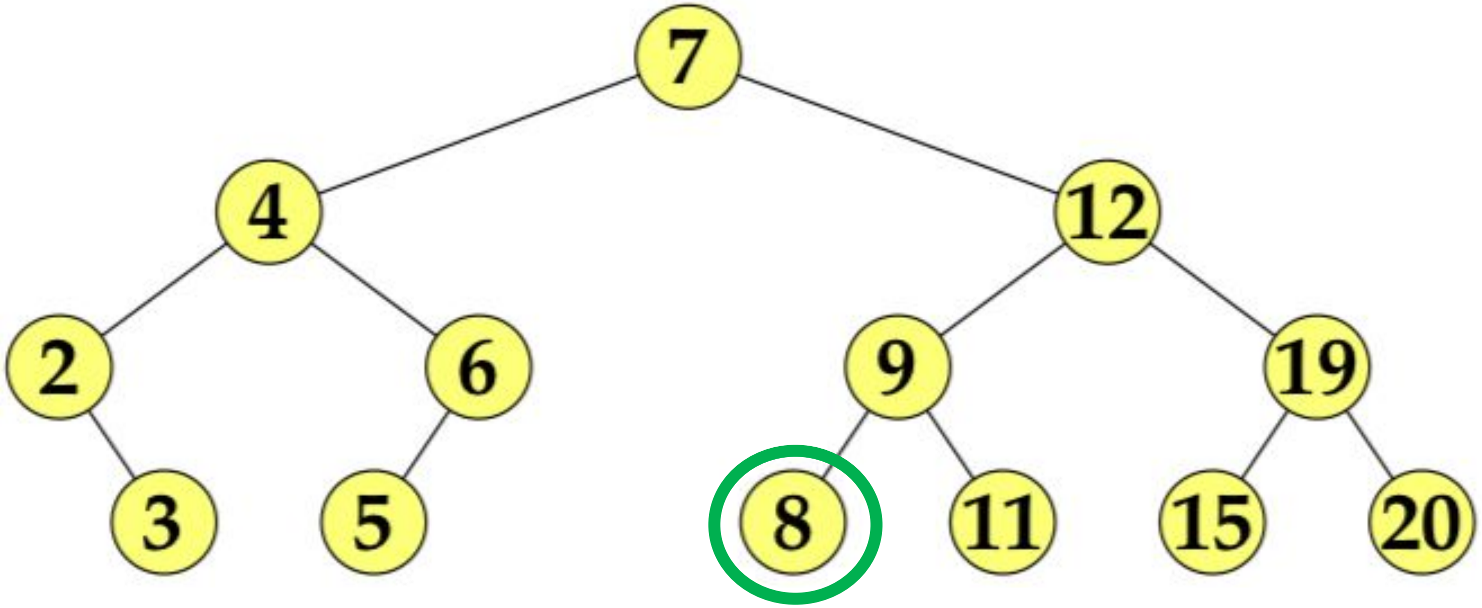


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8

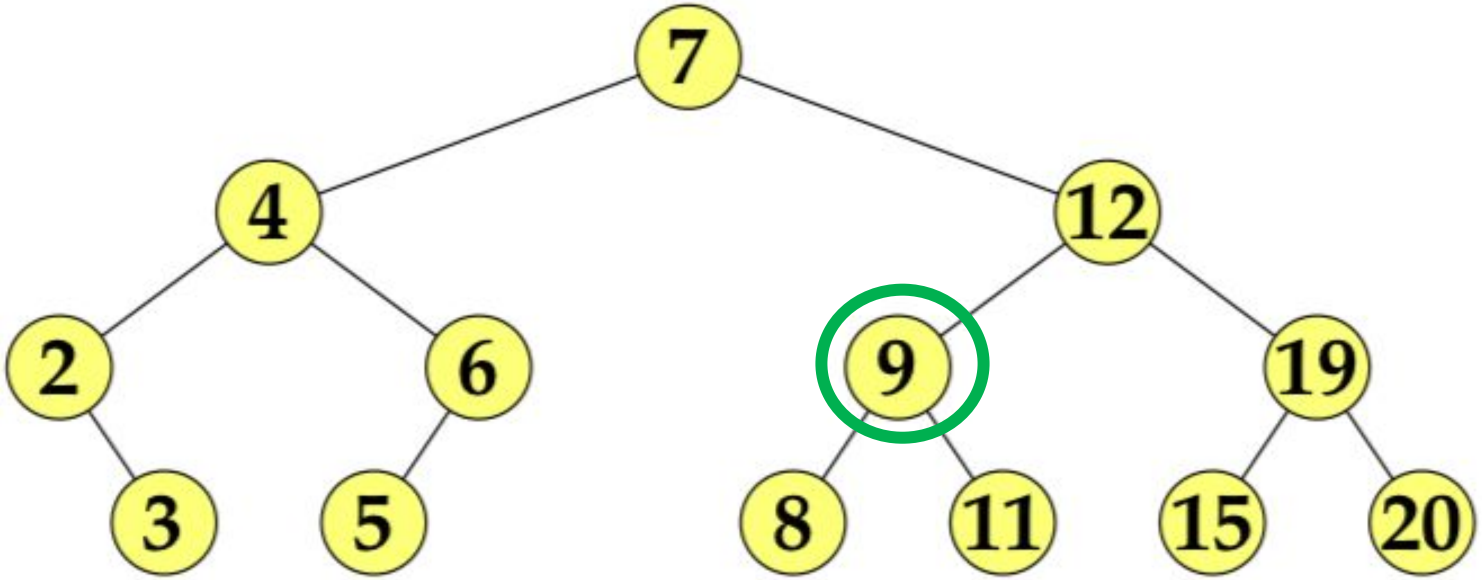


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8

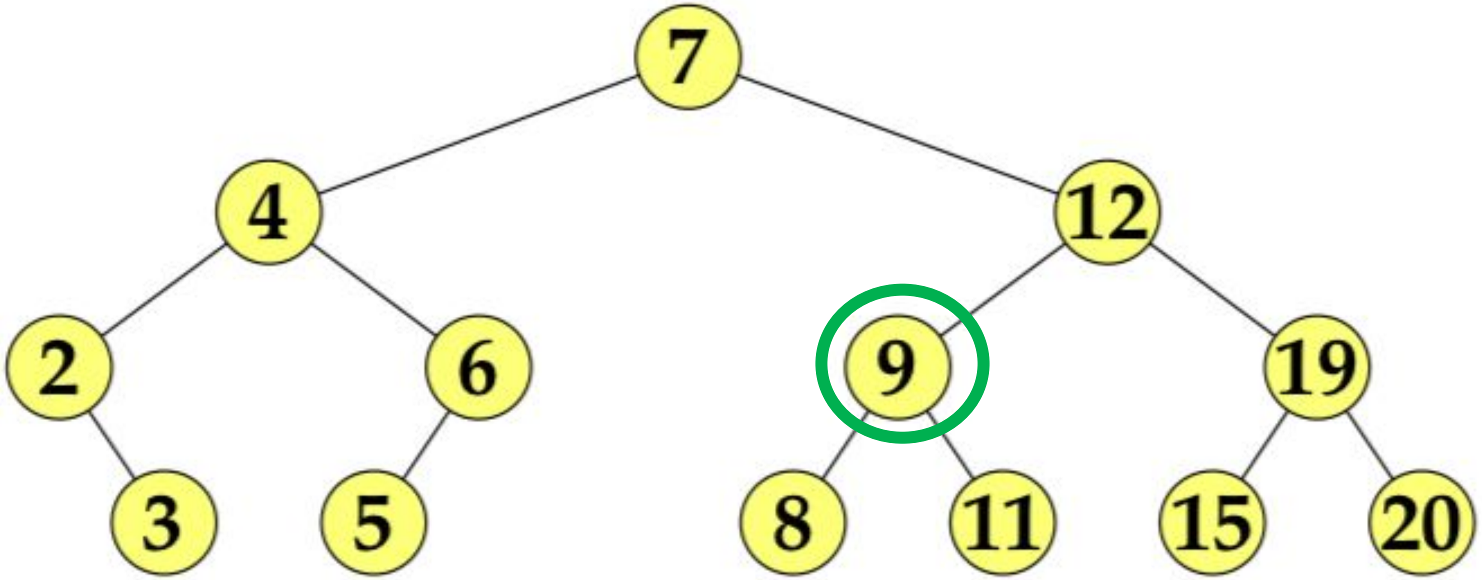


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9

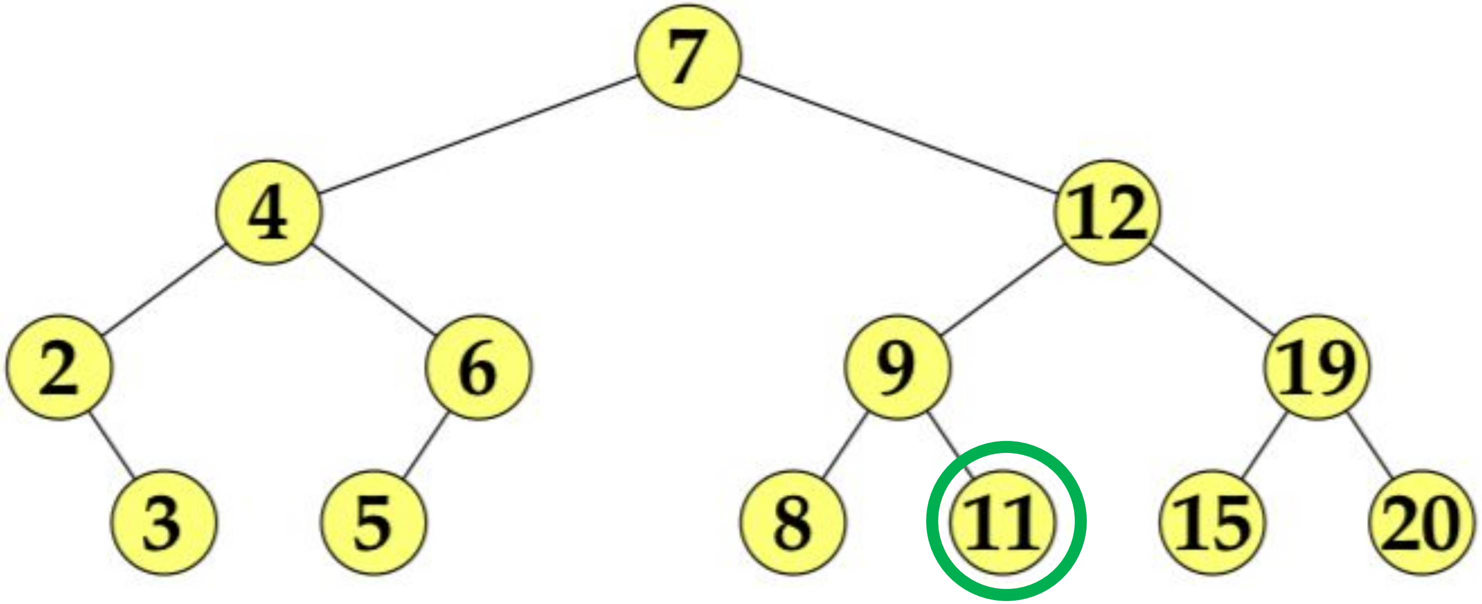


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9

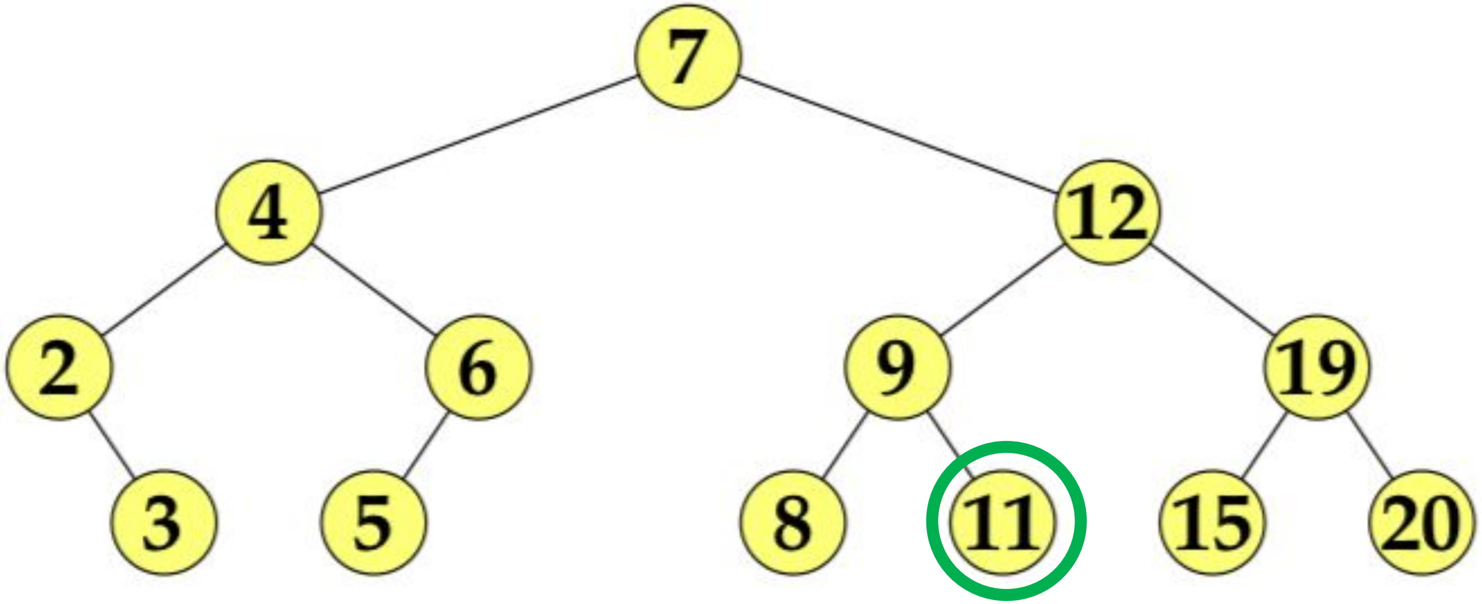


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11

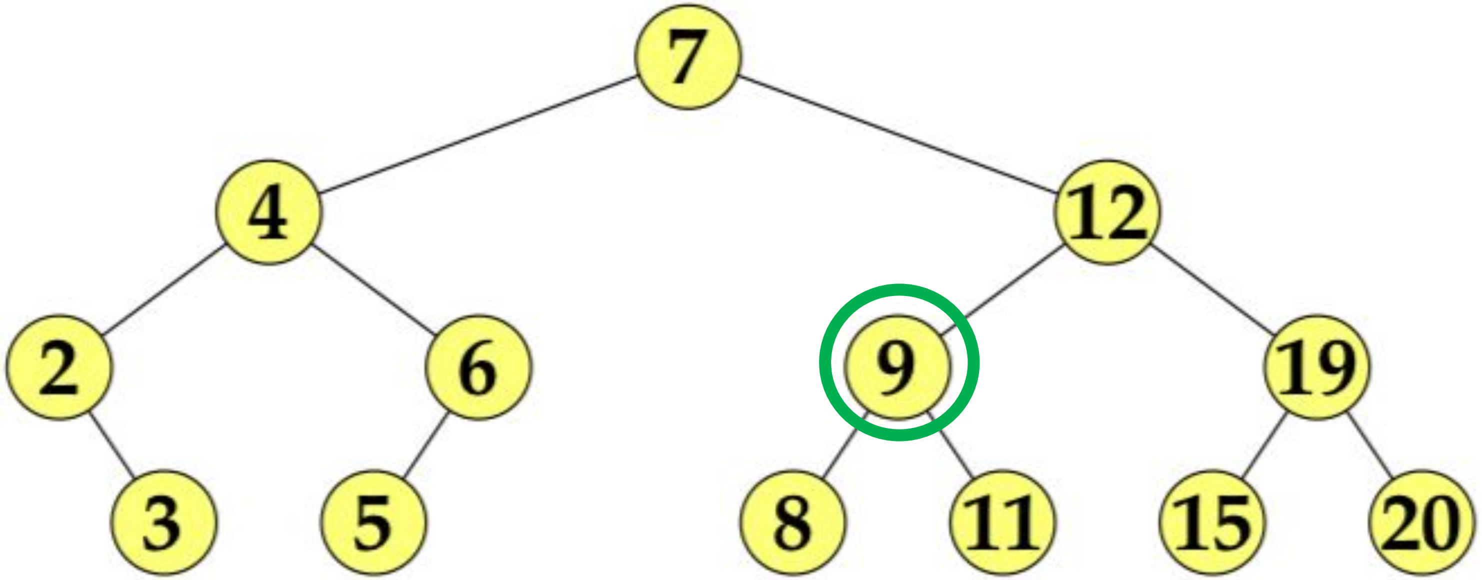


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11

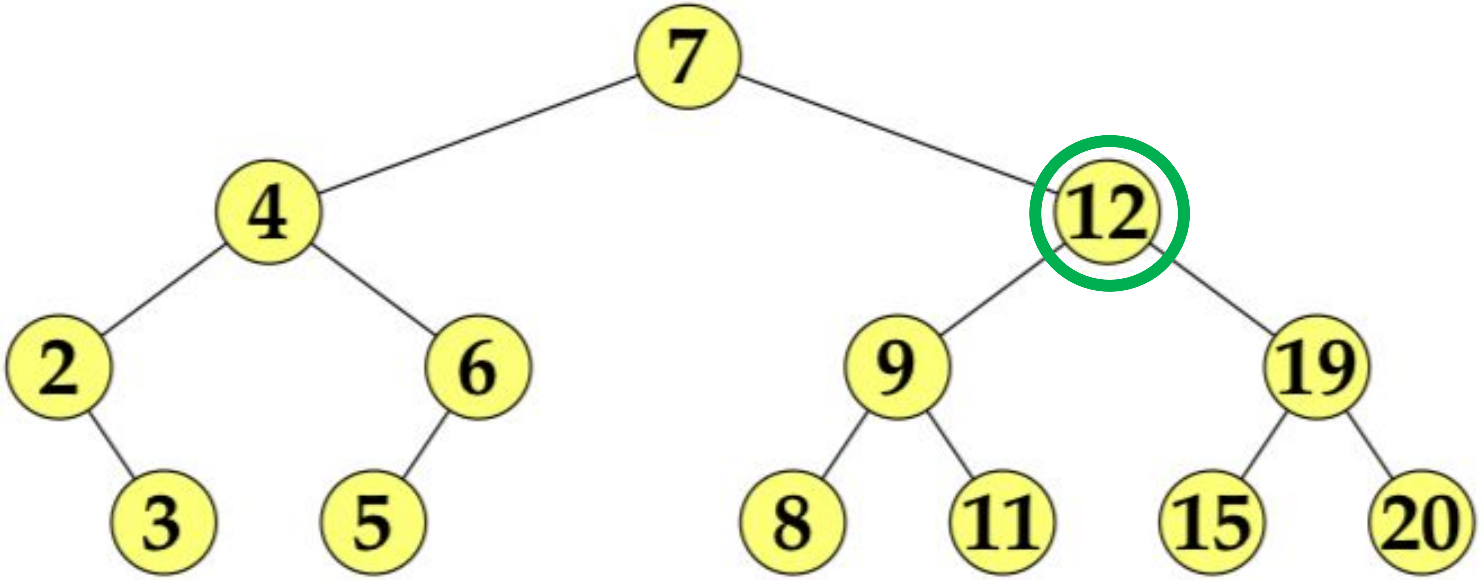


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11

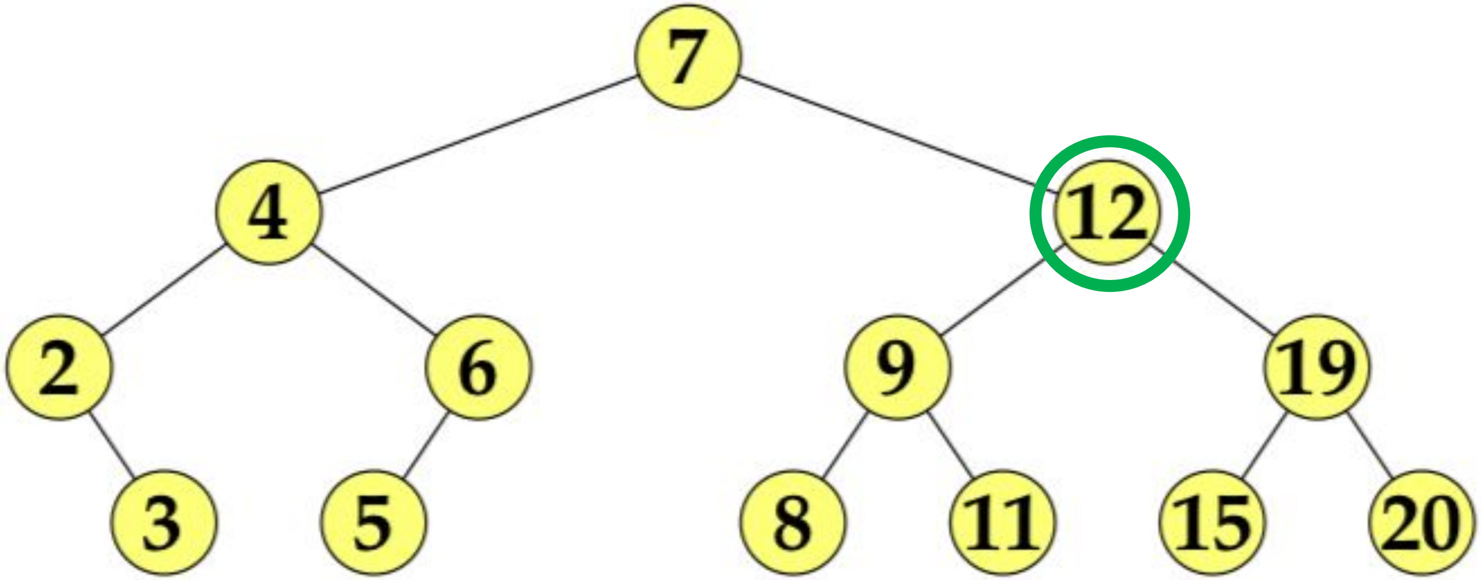


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12

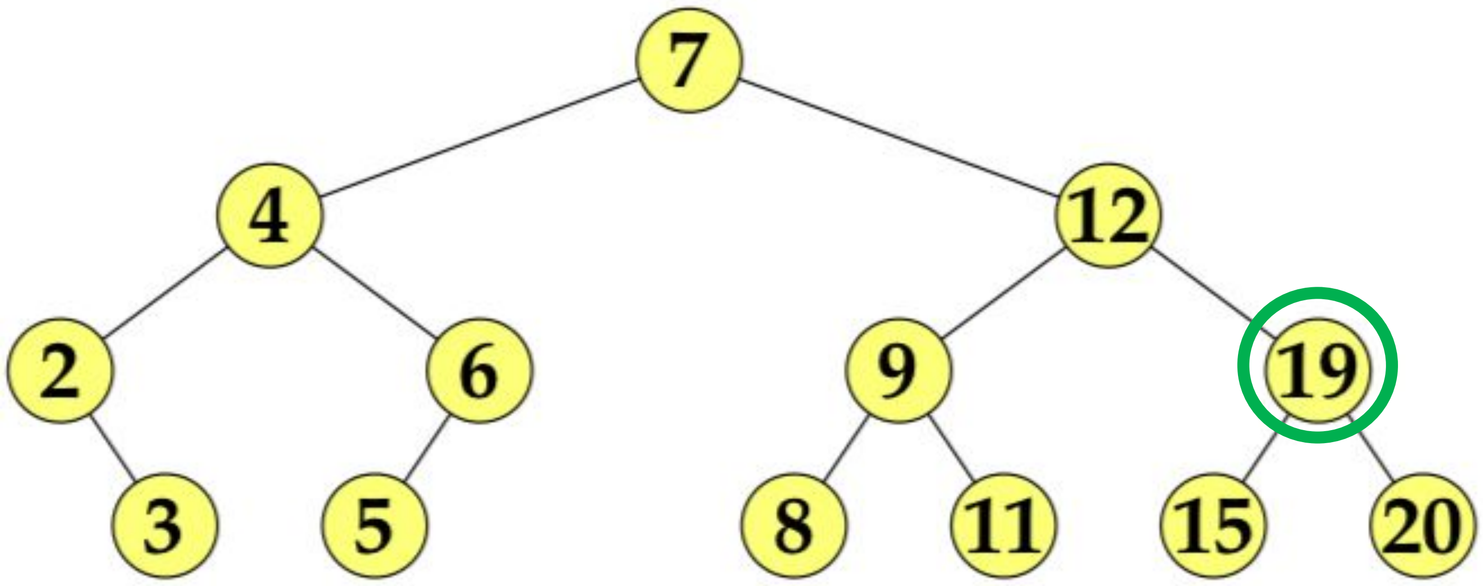


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12

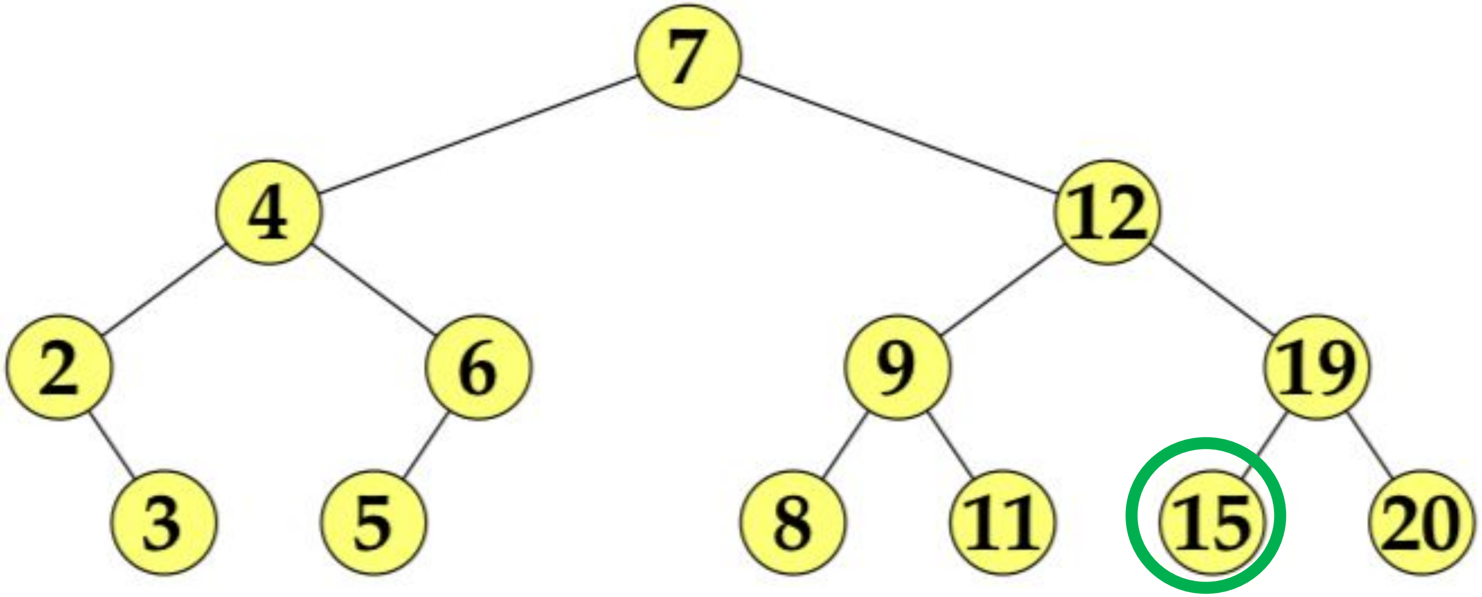


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12

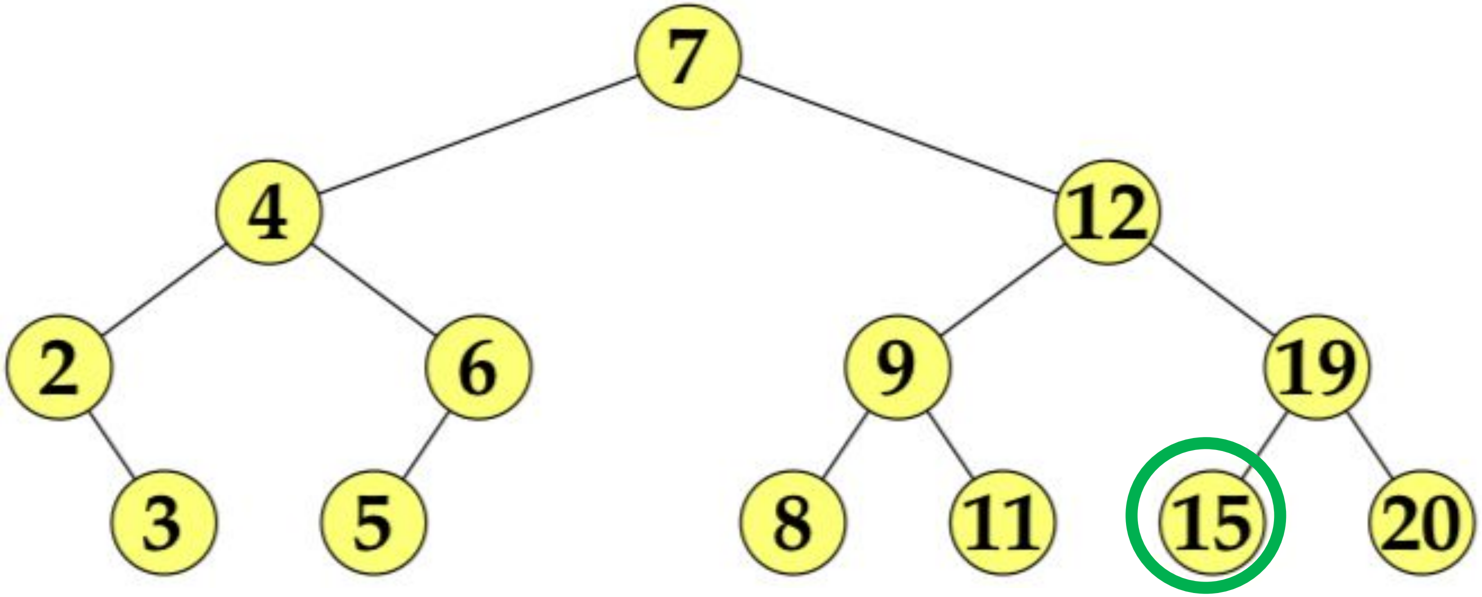


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15

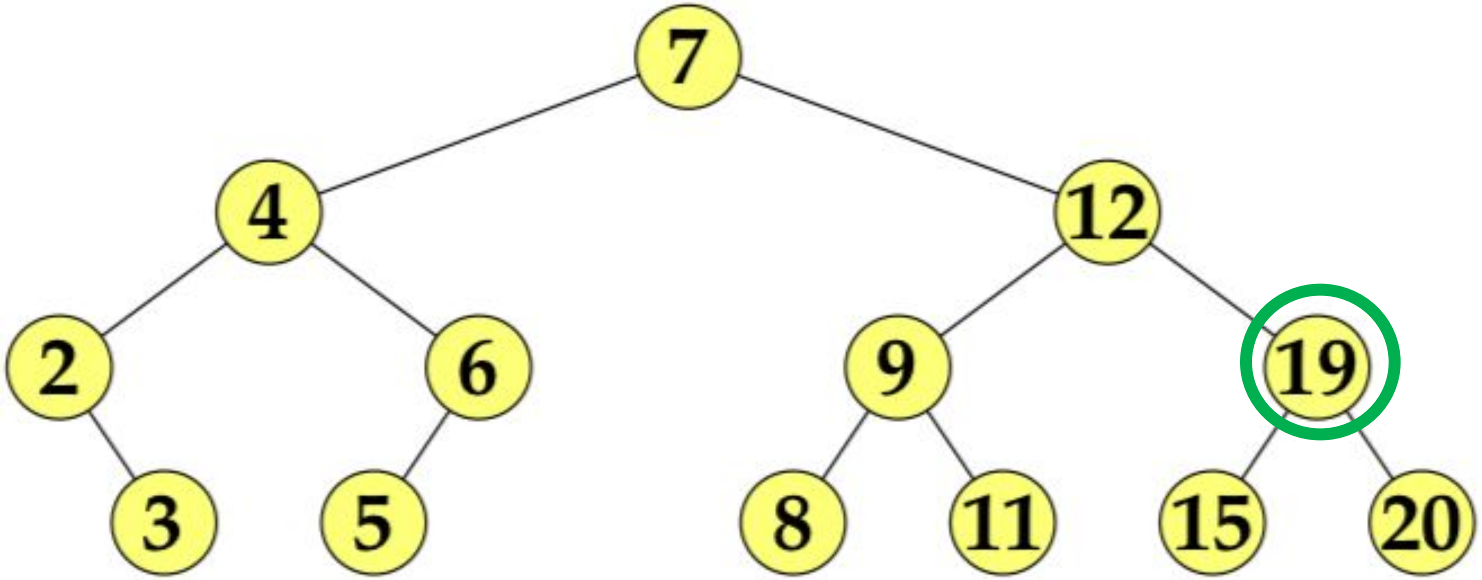


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15

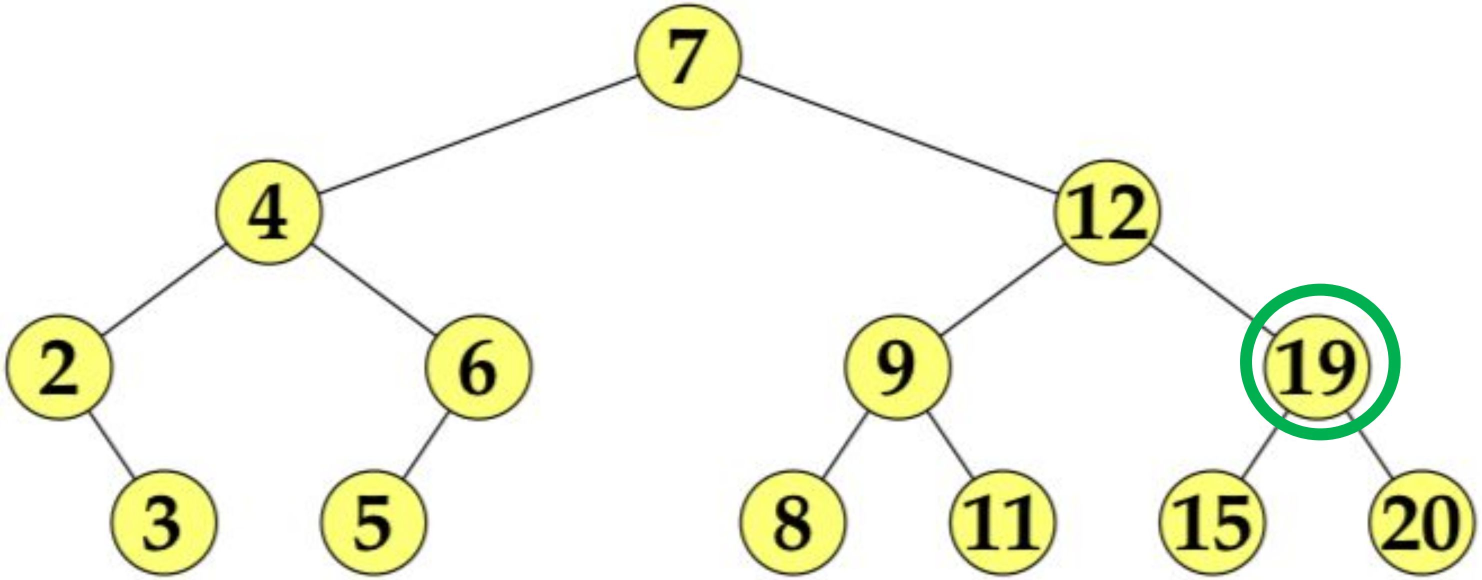


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, *19*

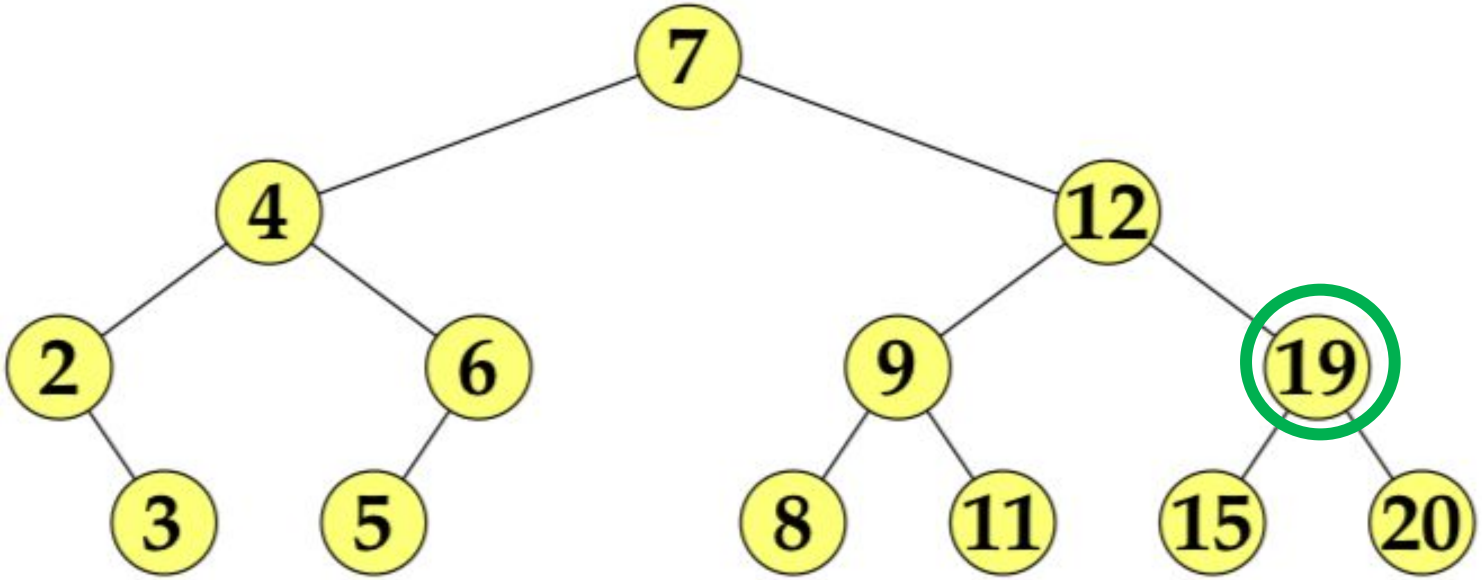


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 19

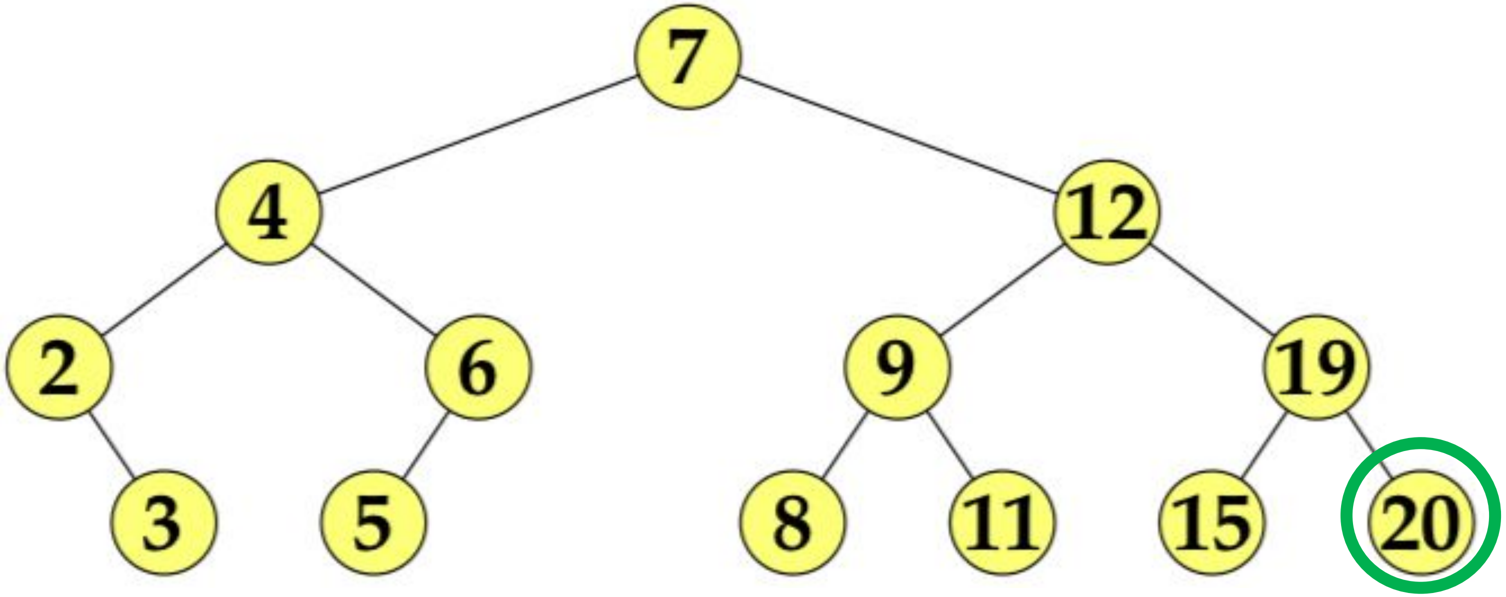


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 19

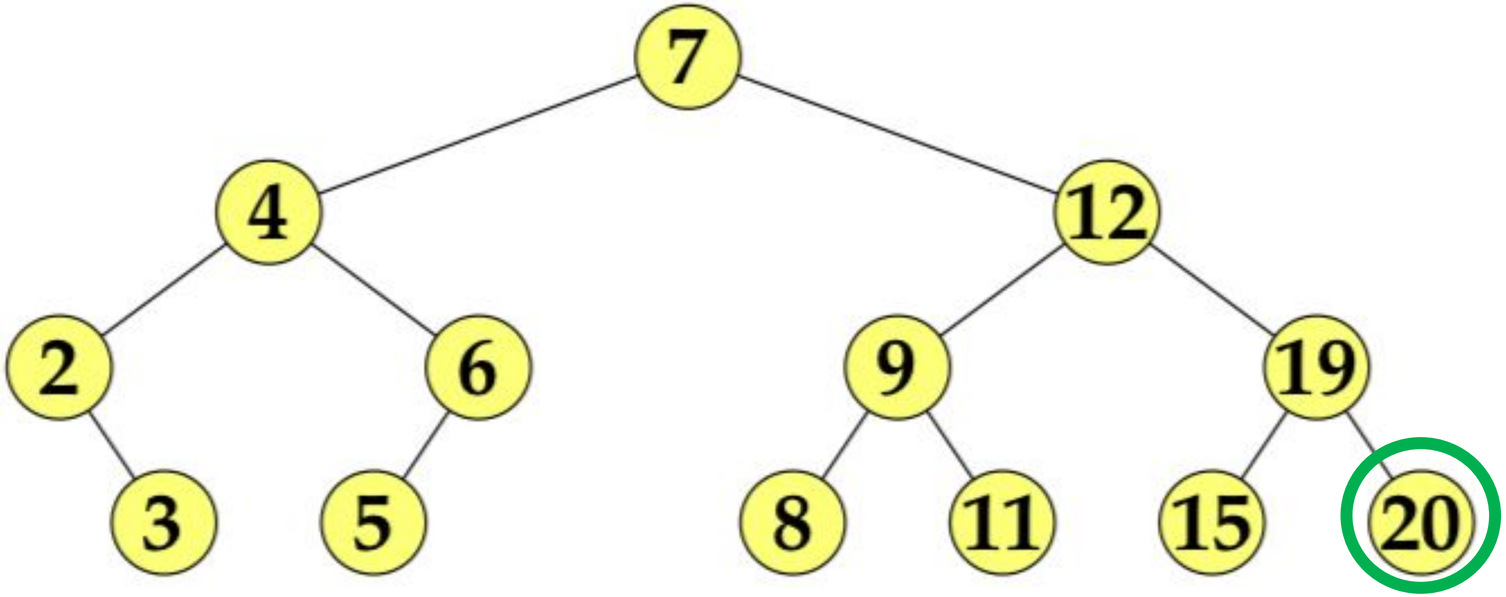


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 19, 21

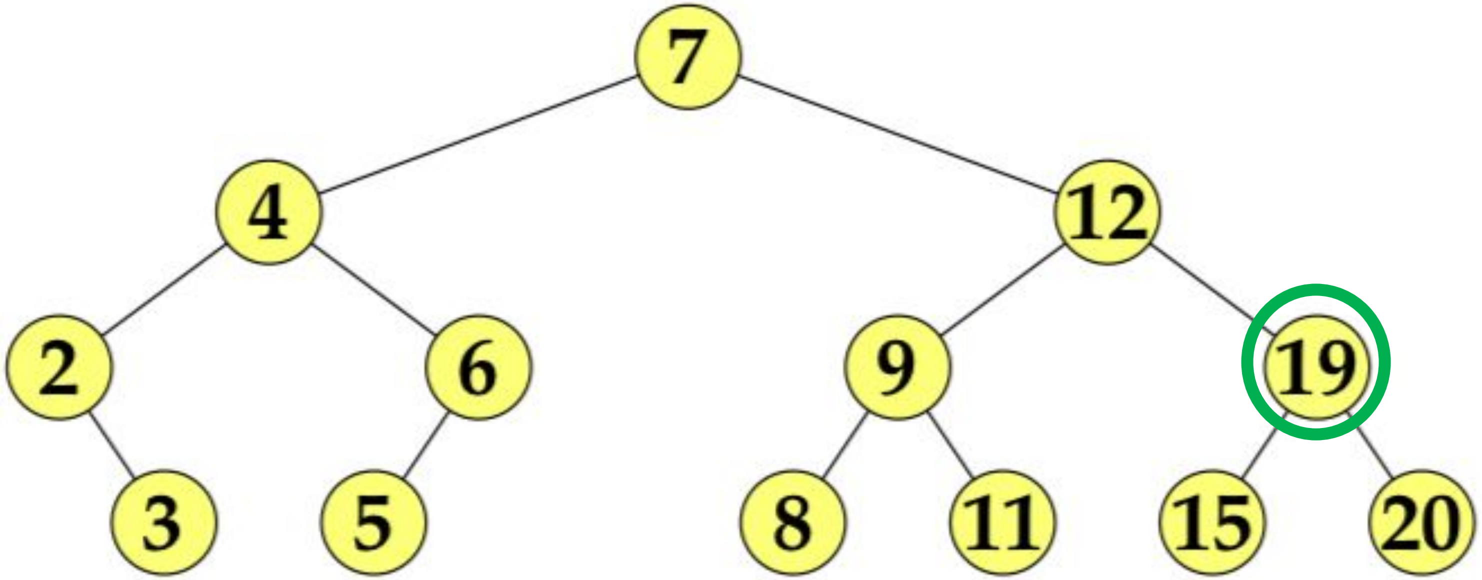


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 19, 21

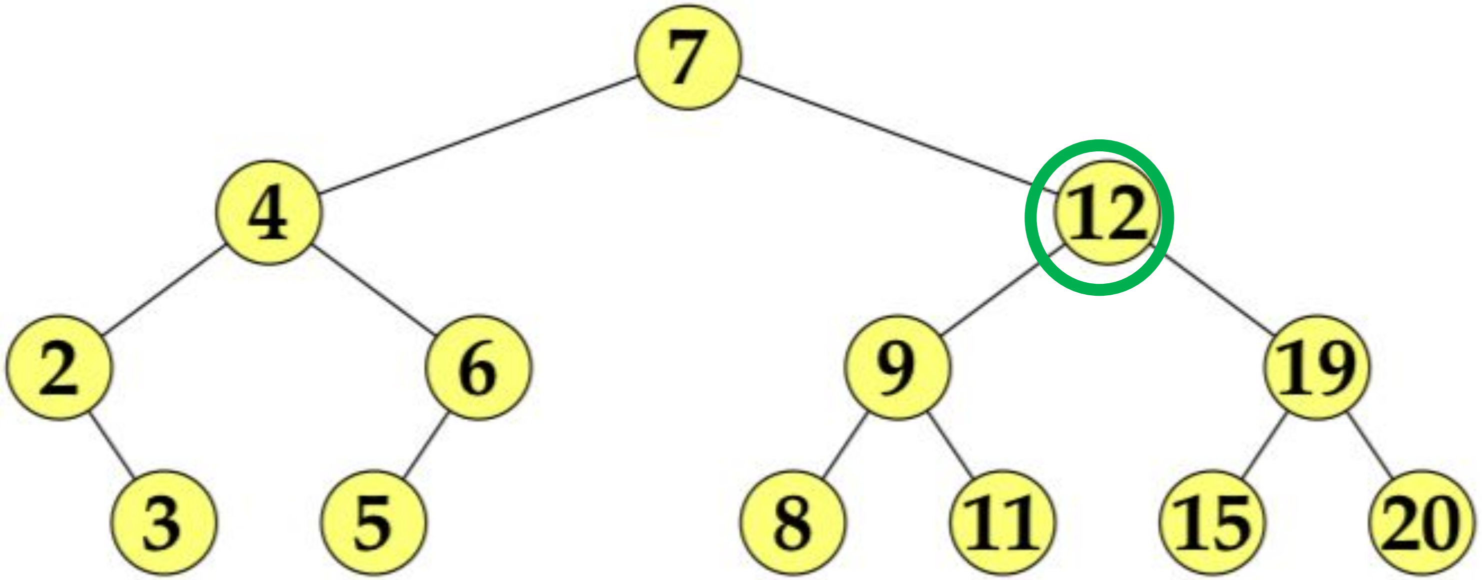


Inorder

What would the in-order traversal be here?

left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 19, 21

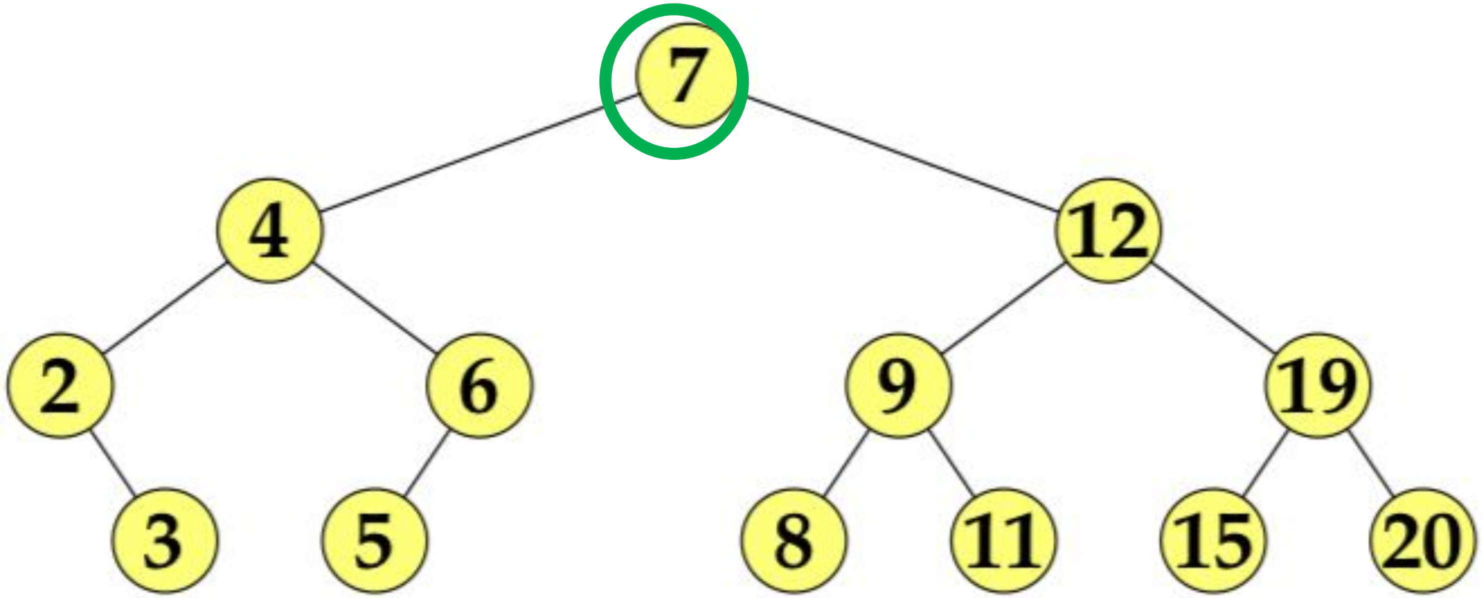


Inorder

What would the in-order traversal be here?

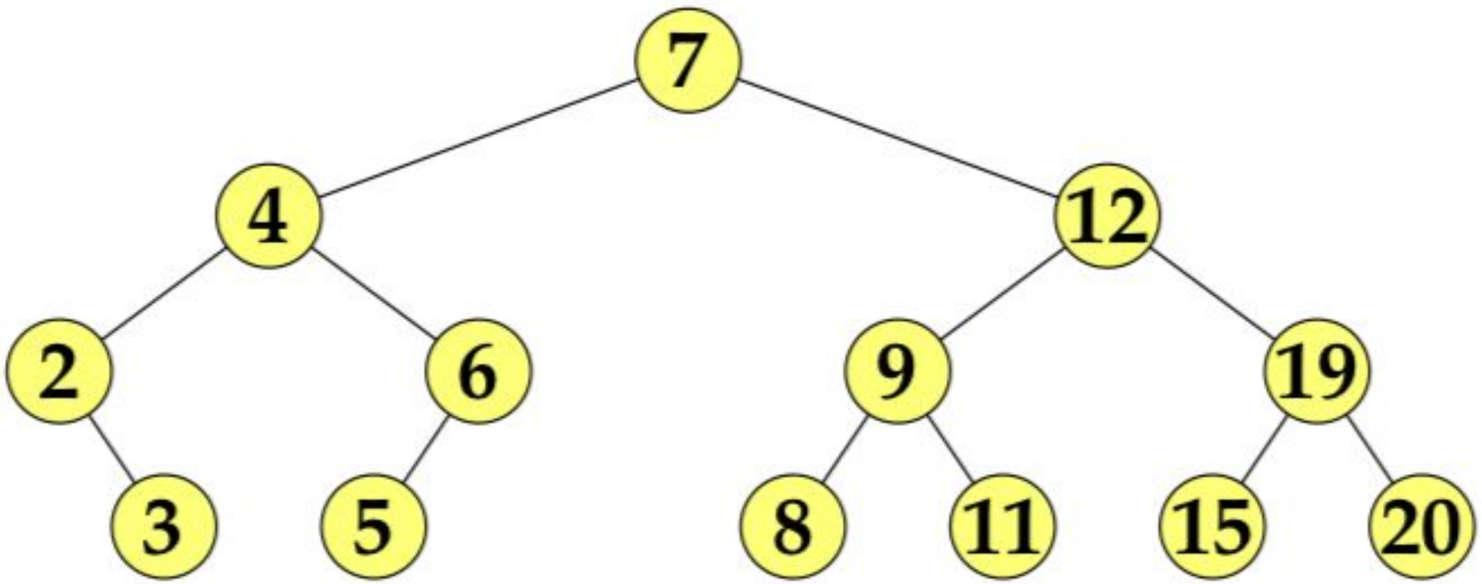
left subtree, current, right subtree

2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 19, 21



Inorder

- 2, 3, 4, 5, 6, 7, 8 , 9, 11, 12, 15, 19, 20



In Order Traversal Implementation

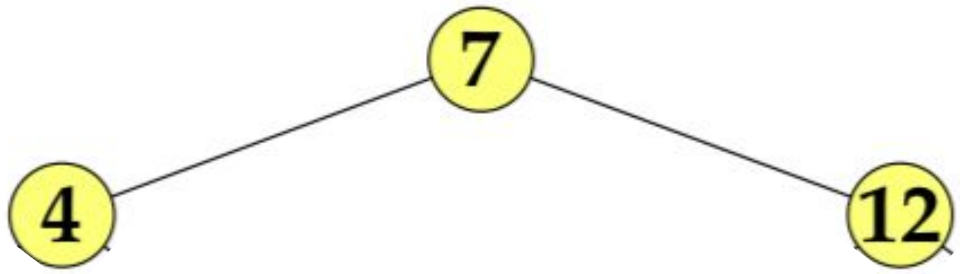
Pre Order Traversal

1. Print the current node
2. Move left
3. Move right

Pre order Example 1

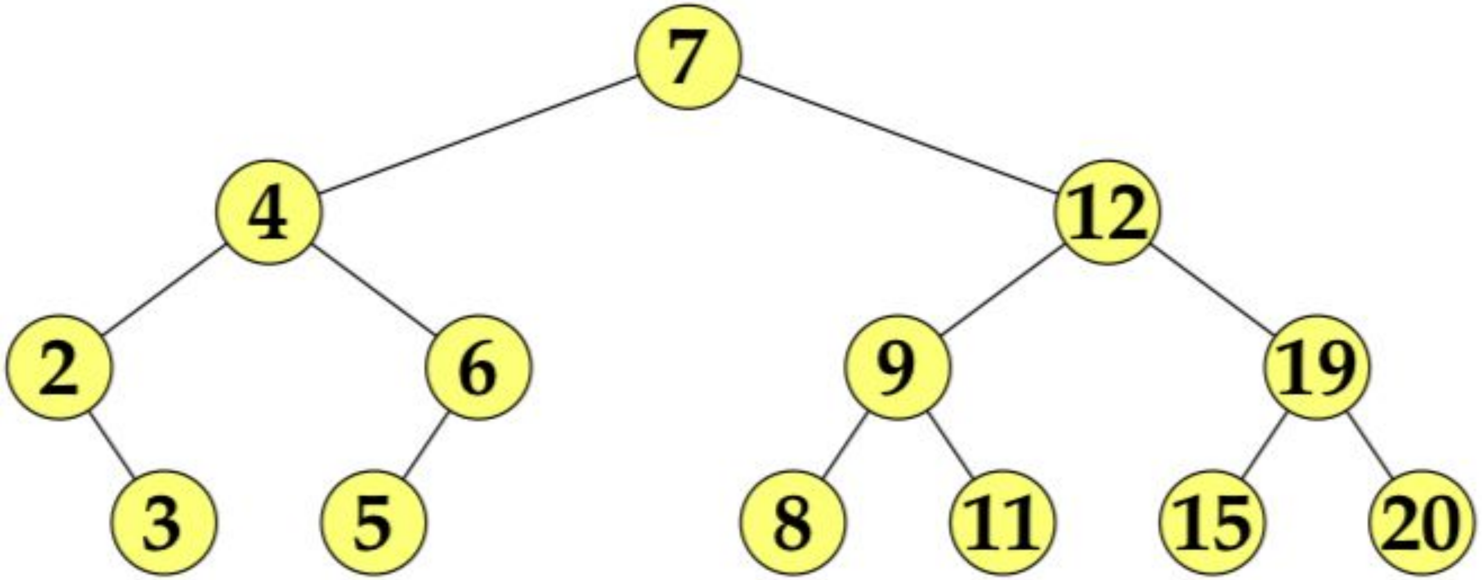
What would the pre-order traversal be here?

current, left subtree, right subtree



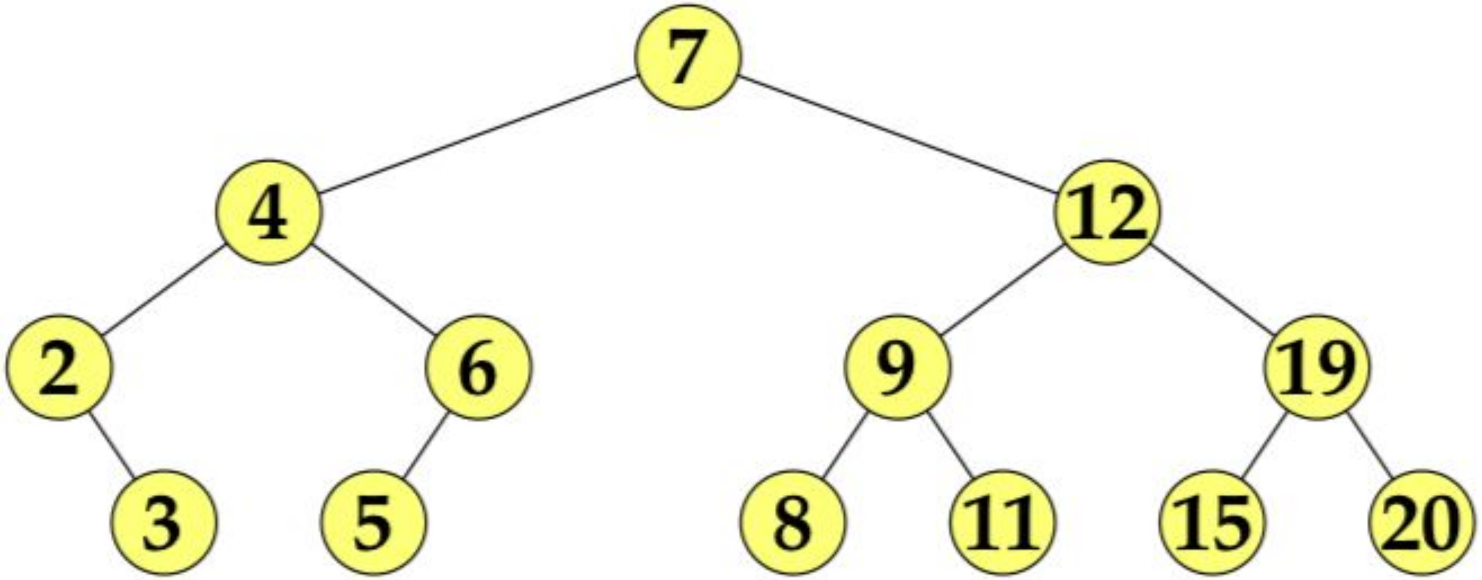
Preorder

Current, left, right



Preorder

- 7, 4, 2, 3, 6, 5, 12, 9, 8, 11, 19, 15, 20



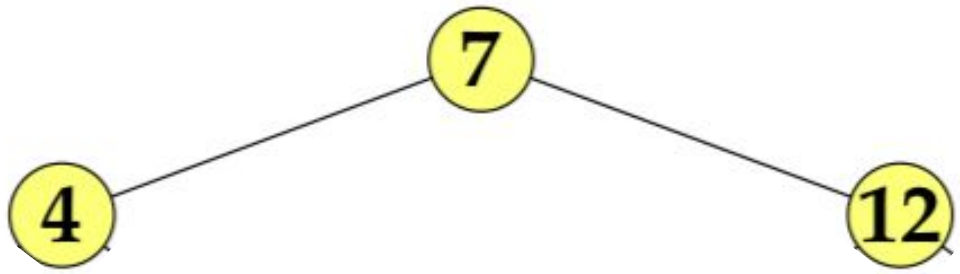
Post Order Traversal

1. Move left
2. Move right
3. Print the current node

Post order Example 1

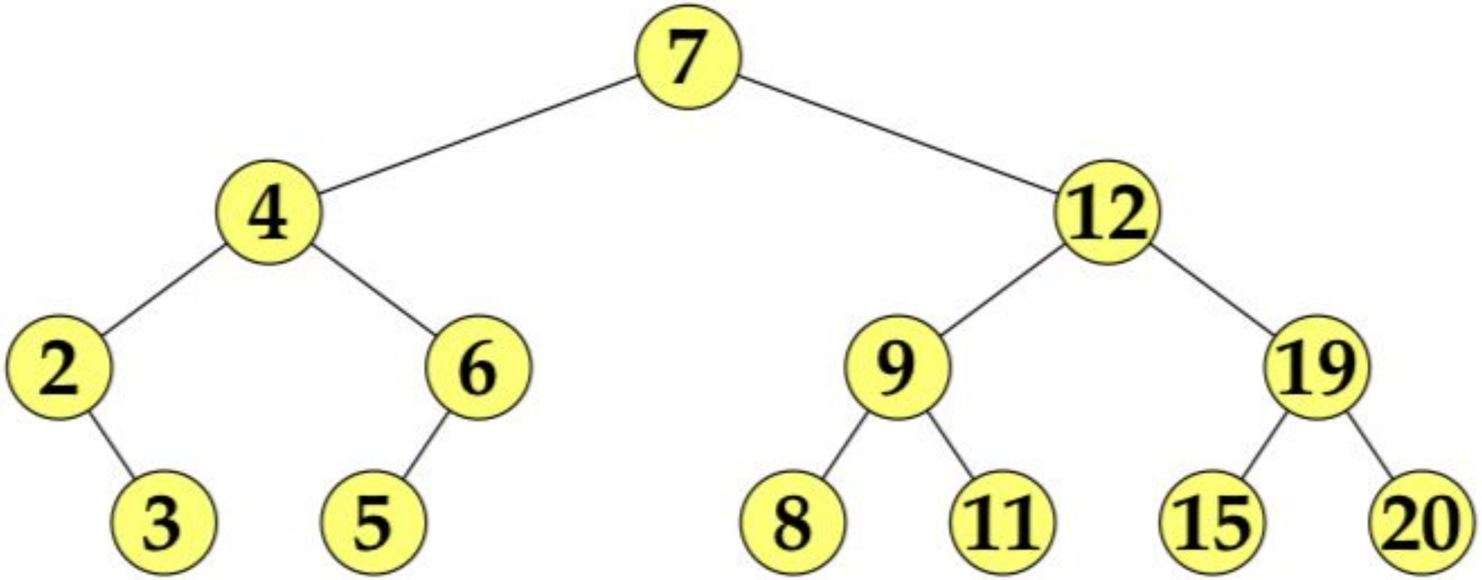
What would the pre-order traversal be here?

left subtree, right subtree, current



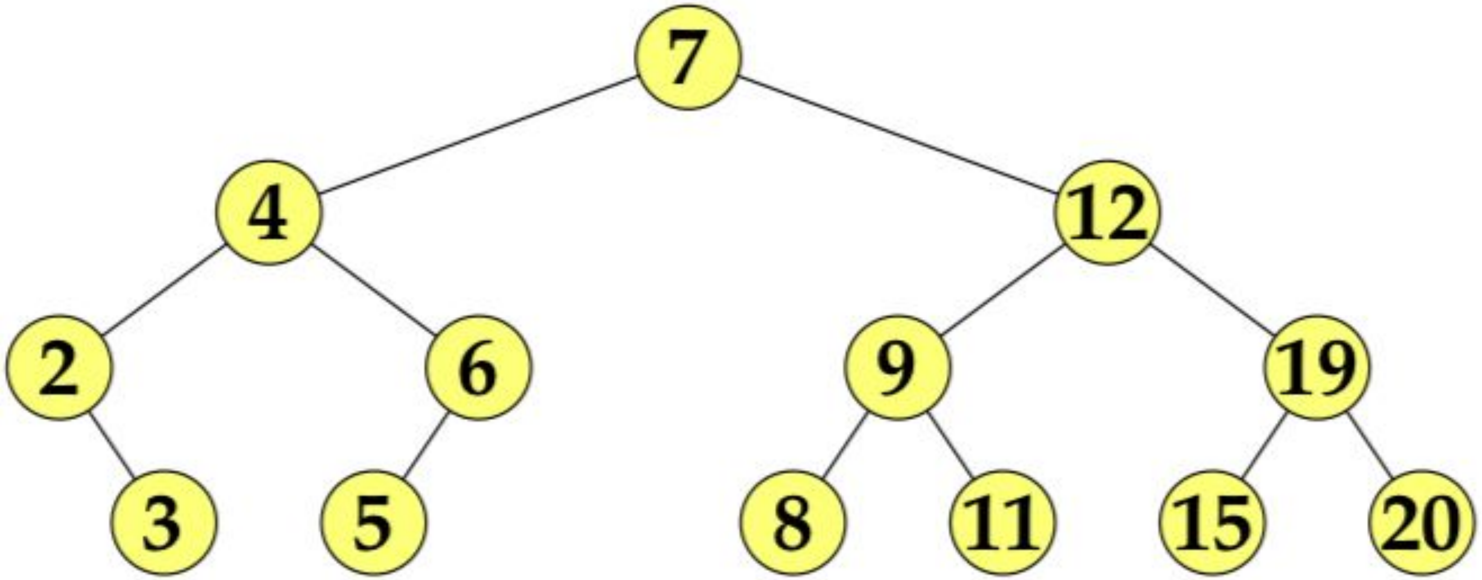
Postorder

Left, right, current



Postorder

- 3, 2, 5, 6, 4, 8, 11, 9, 15, 20, 19, 12, 7



Interface you will implement in homework

Performance of BST

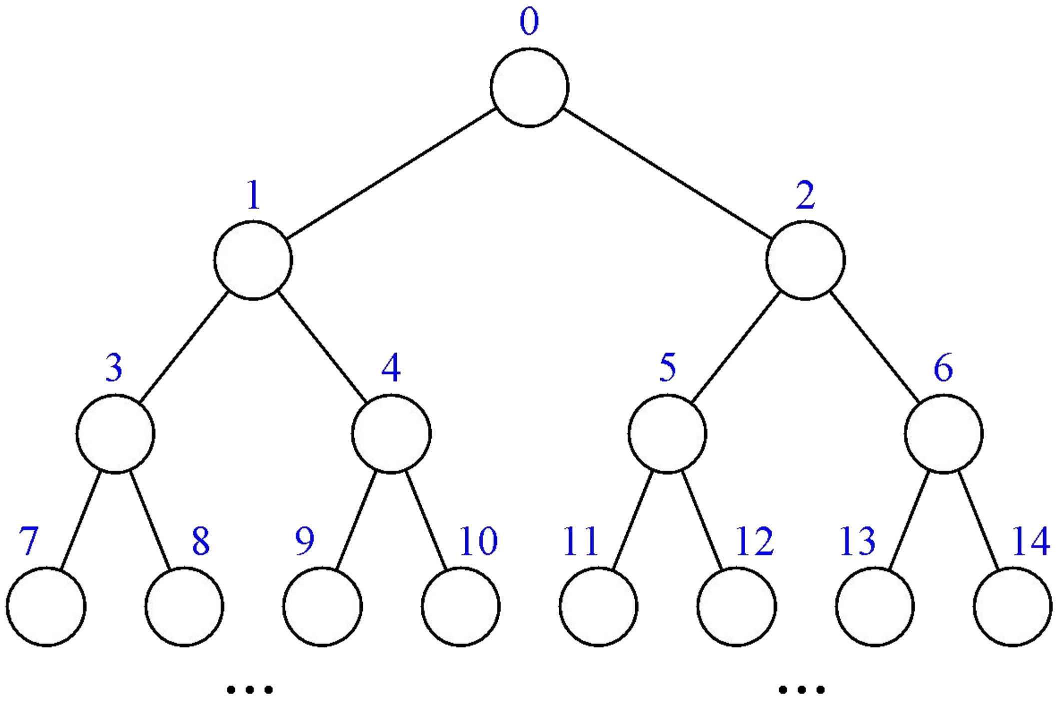
	BST balanced	BST worst
search		
insert		
remove		
min/max		

Array-based Implementation

- BinaryTrees can be implemented in different ways
  - Linked nodes - what you'll do in your homework
  - Array

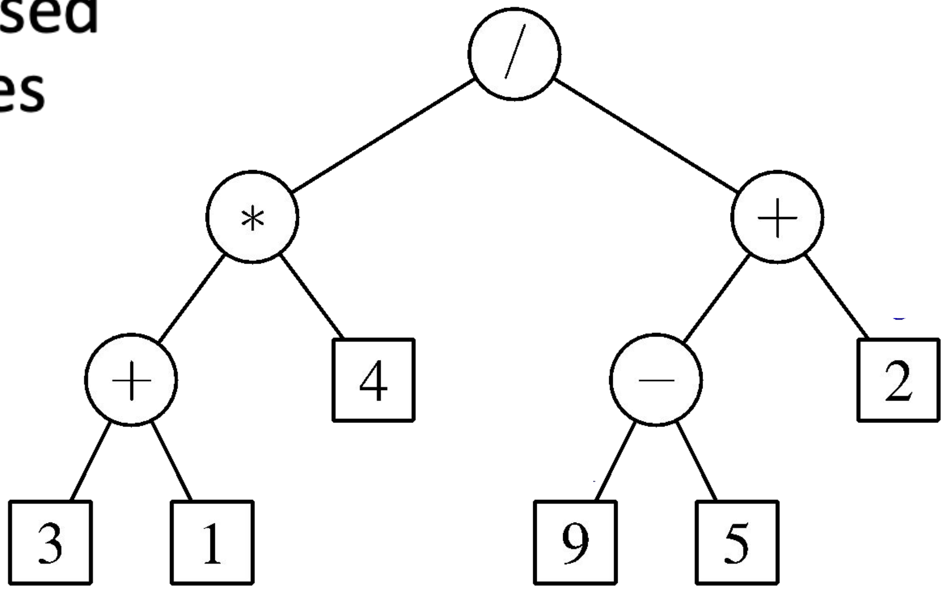
Array-based Implementation

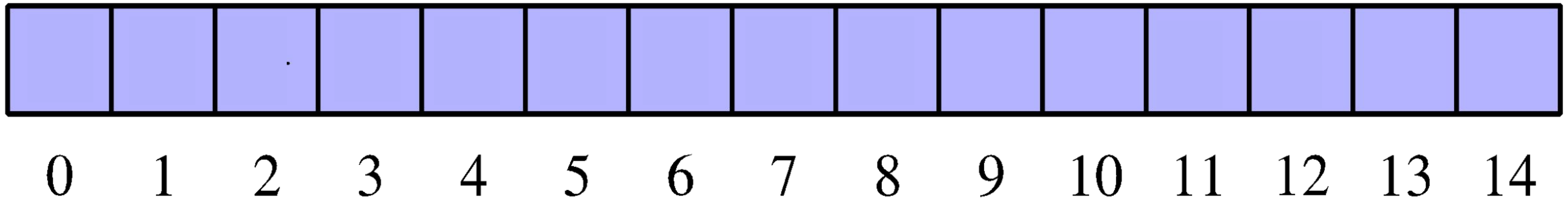
- Number nodes level-by-level, left-to-right
- $f(root) = 0$
- $f(l) = 2f(p) + 1$
- $f(r) = 2f(p) + 2$
- Numbering is based on all positions, not just occupied positions



Array-based Binary Tree

- The numbering can then be used as indices for storing the nodes directly in an array
- $f(root) = 0$
- $f(l) = 2f(p) + 1$
- $f(r) = 2f(p) + 2$





Array-based Binary Tree

- The numbering can then be used as indices for storing the nodes directly in an array