LEC 13

CSE 123

Binary Trees

Questions during Class?
Raise hand or send here

sli.do #cse123

BEFORE WE START

Talk to your neighbors:

What is your favorite study spot on campus?

Respond on sli.do!

Instructor: Brett Wortzman

TAs:

Arohan	Jonah	Kavya	Eeshani	Trien
Ashar	Brice	Misha	Aidan	Evan
Sean	Chris	Kieran	Cora	Rena
Chloe	Elden	Sahana	Dixon	Katharine
Jenny	Ishita	Anirudh	Nhan	Anya
Nate	Kuhu	Crystal		

Now playing: 🎶 CSE 123 26wi Lecture Tunes 🎶

Lecture Outline

- **Announcements**
- Binary Tree Review
- Traversals
- Practice!

Announcements

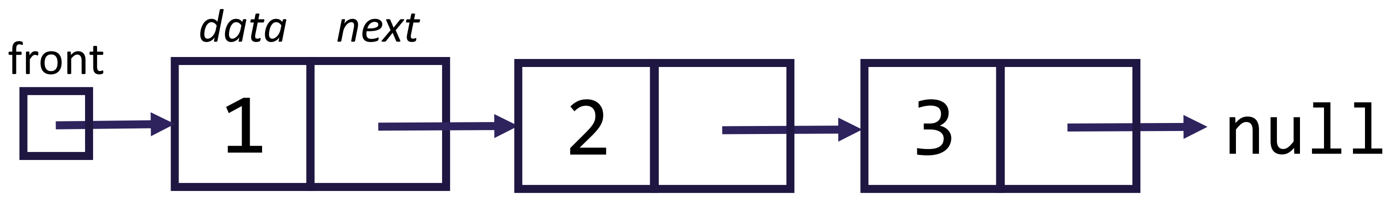
- Resubmission Cycle 4 is due tonight at 11:59pm
  - ***C1***, P1 eligible
- Programming Assignment 2 is out, due Wednesday (Nov 25)

Lecture Outline

- Announcements
- **Binary Tree Review**
- Traversals
- Practice!

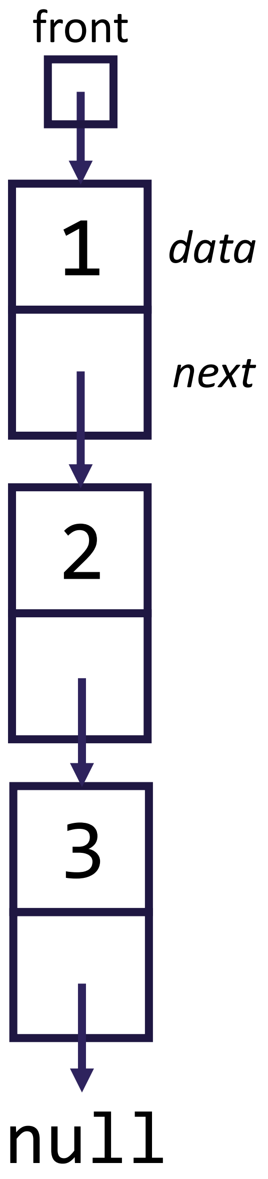
Binary Trees

- Last data structure of the quarter!
  - Very similar to `LinkedLists`…



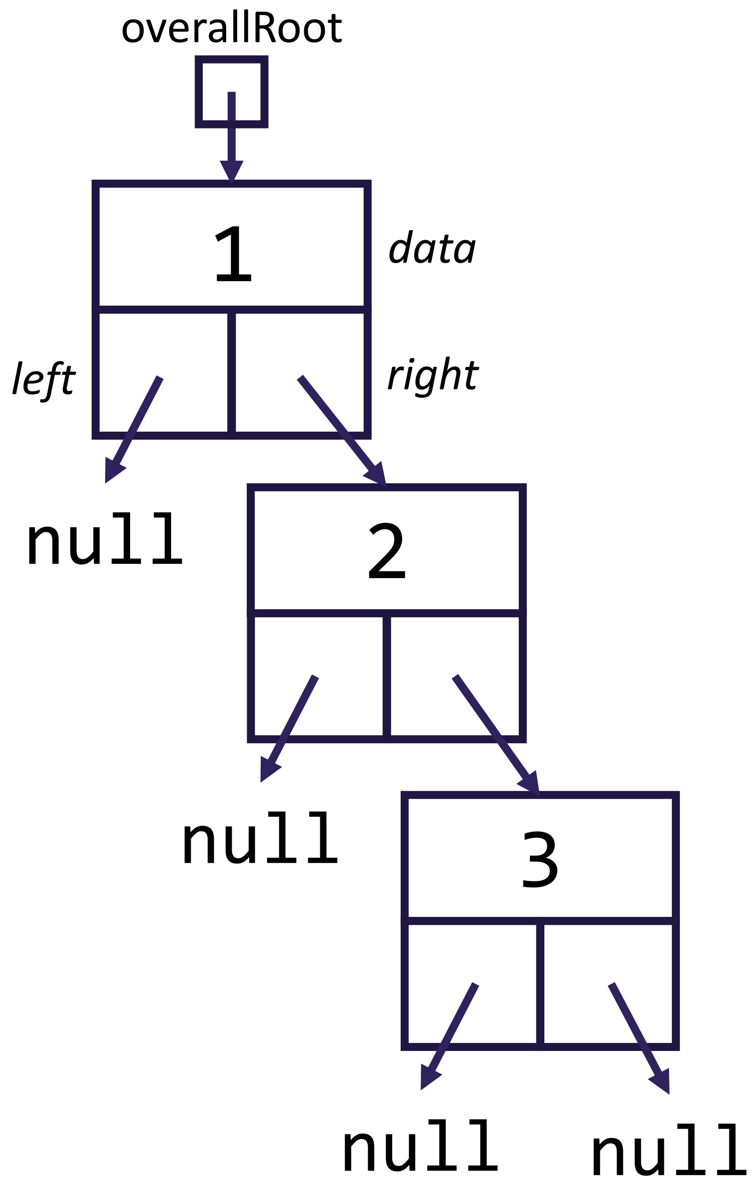
Binary Trees

- Last data structure of the quarter!
  - Very similar to `LinkedLists`…



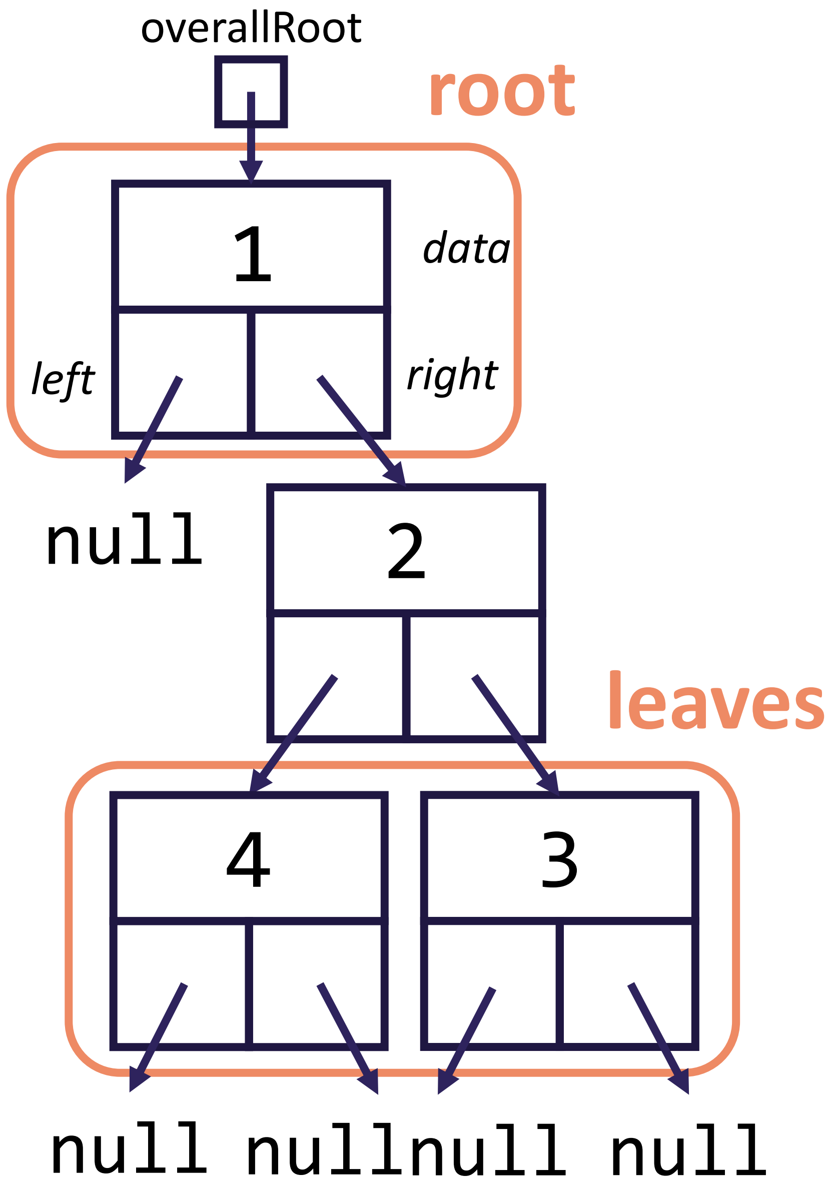
Binary Trees

- Last data structure of the quarter!
  - Very similar to `LinkedLists`…
- Linked `TreeNodes` w/ 3 fields:
  - int data, `TreeNode` left, `TreeNode` right
  - Doubly complicated!

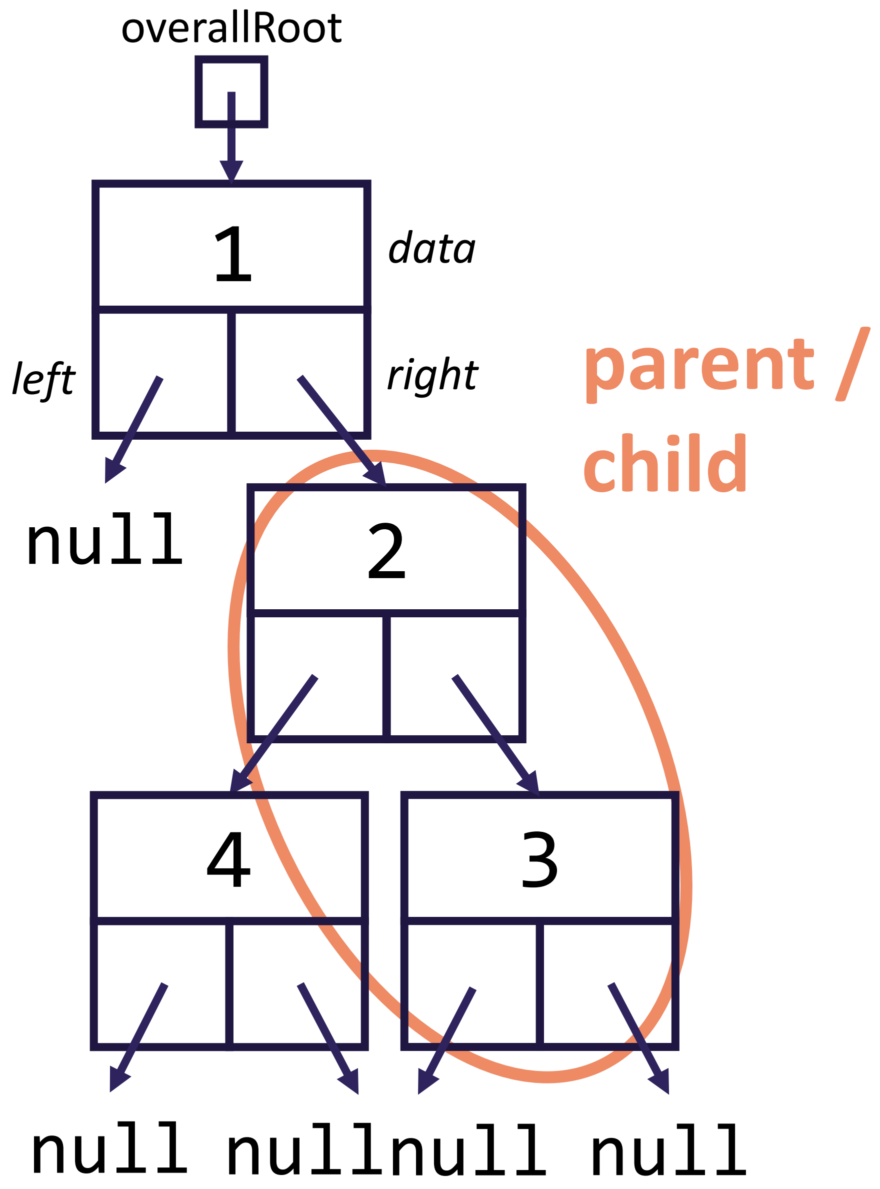


Binary Trees

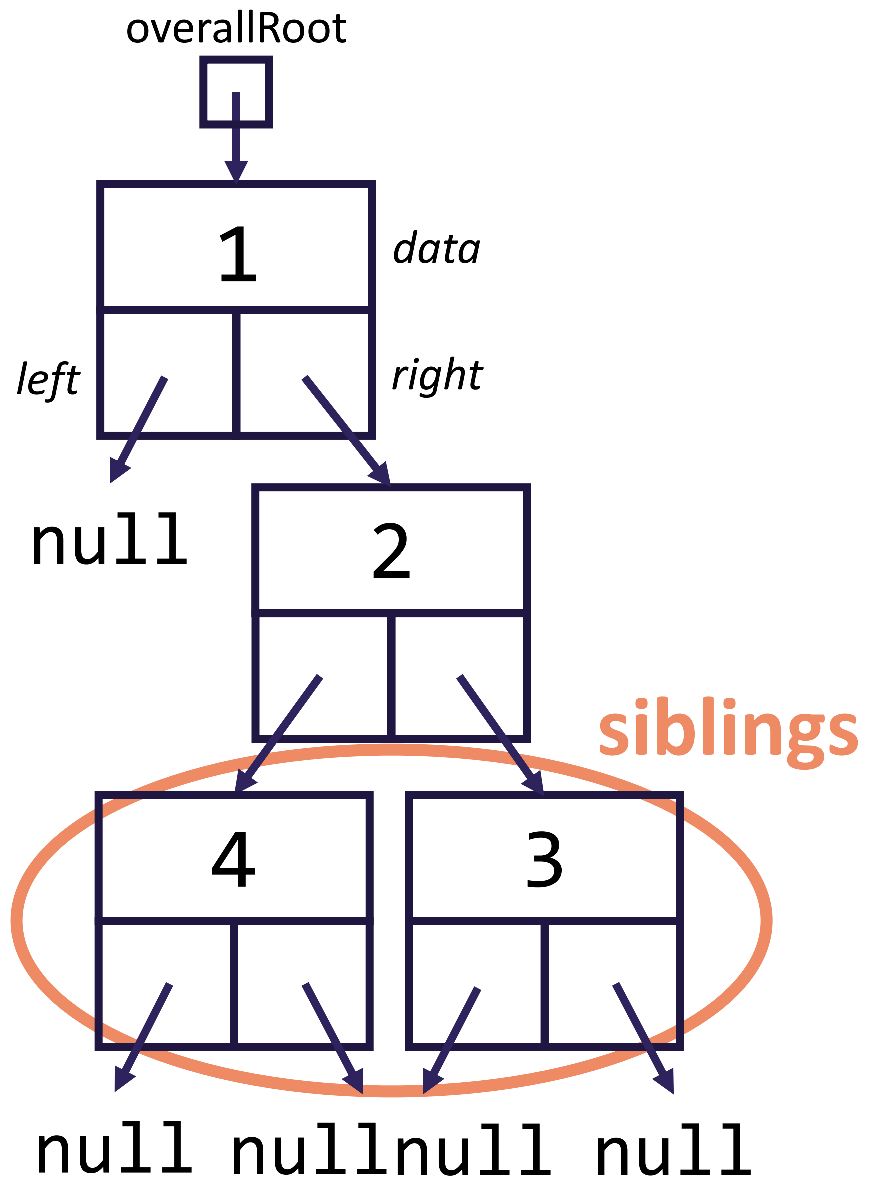
- Last data structure of the quarter!
  - Very similar to `LinkedLists`…
- Linked `TreeNodes` w/ 3 fields:
  - int data, `TreeNode` left, `TreeNode` right
  - Doubly complicated!
- Similar to trees?
  - Close enough!
  - Terminology: root / leaves
- Other terminology as well



Tree Terminology

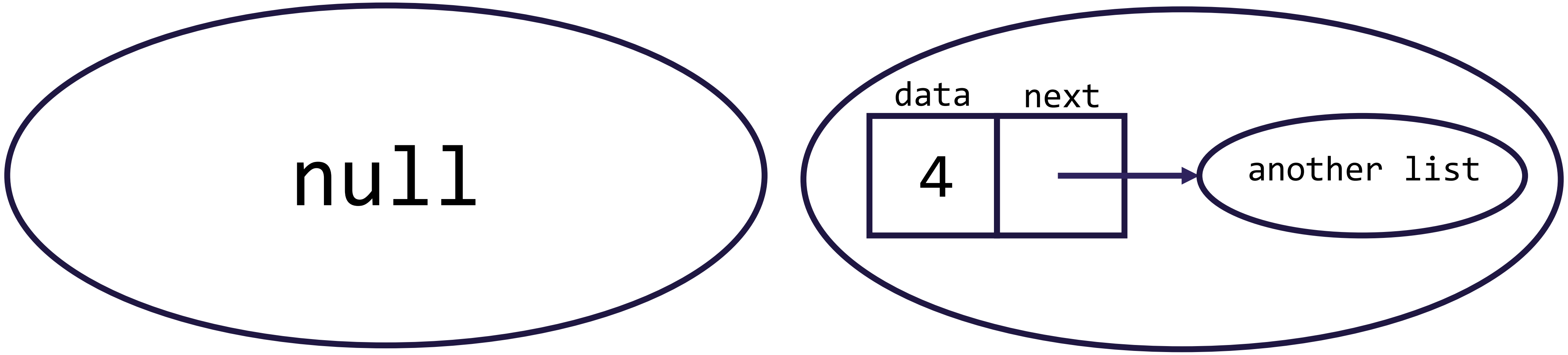


Tree Terminology



Linked Lists [Review]

- A linked list is either:



Empty list

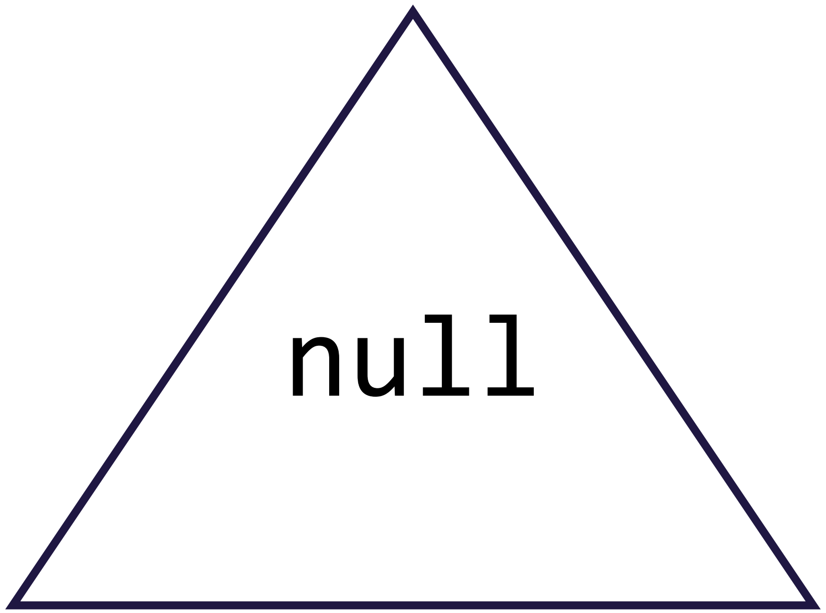
Node w/ another linked list

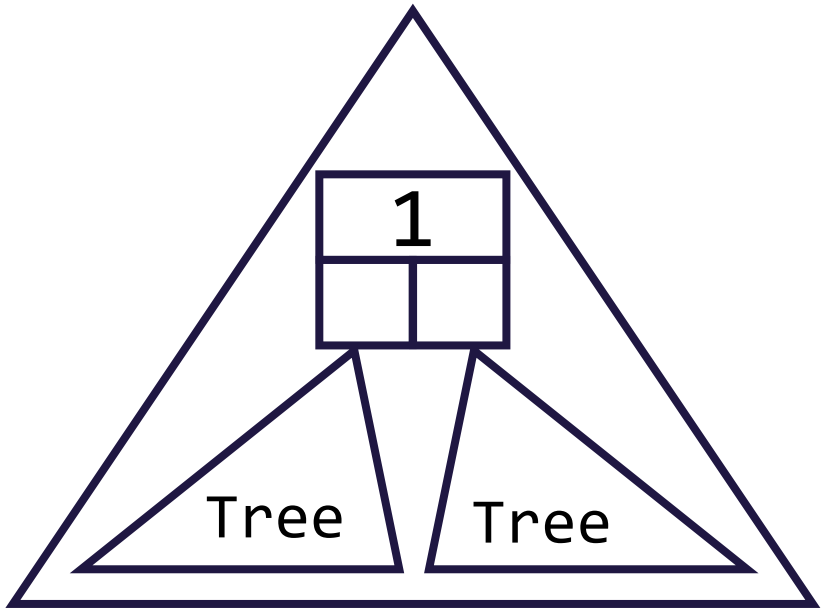
This is a recursive definition!

A list is either empty or a node with another list!

Binary Trees

- A Binary Tree is either:





Empty tree

Node w/ two subtrees

This is a recursive definition!
A tree is either empty or a node with two more trees!

Binary Tree Programming

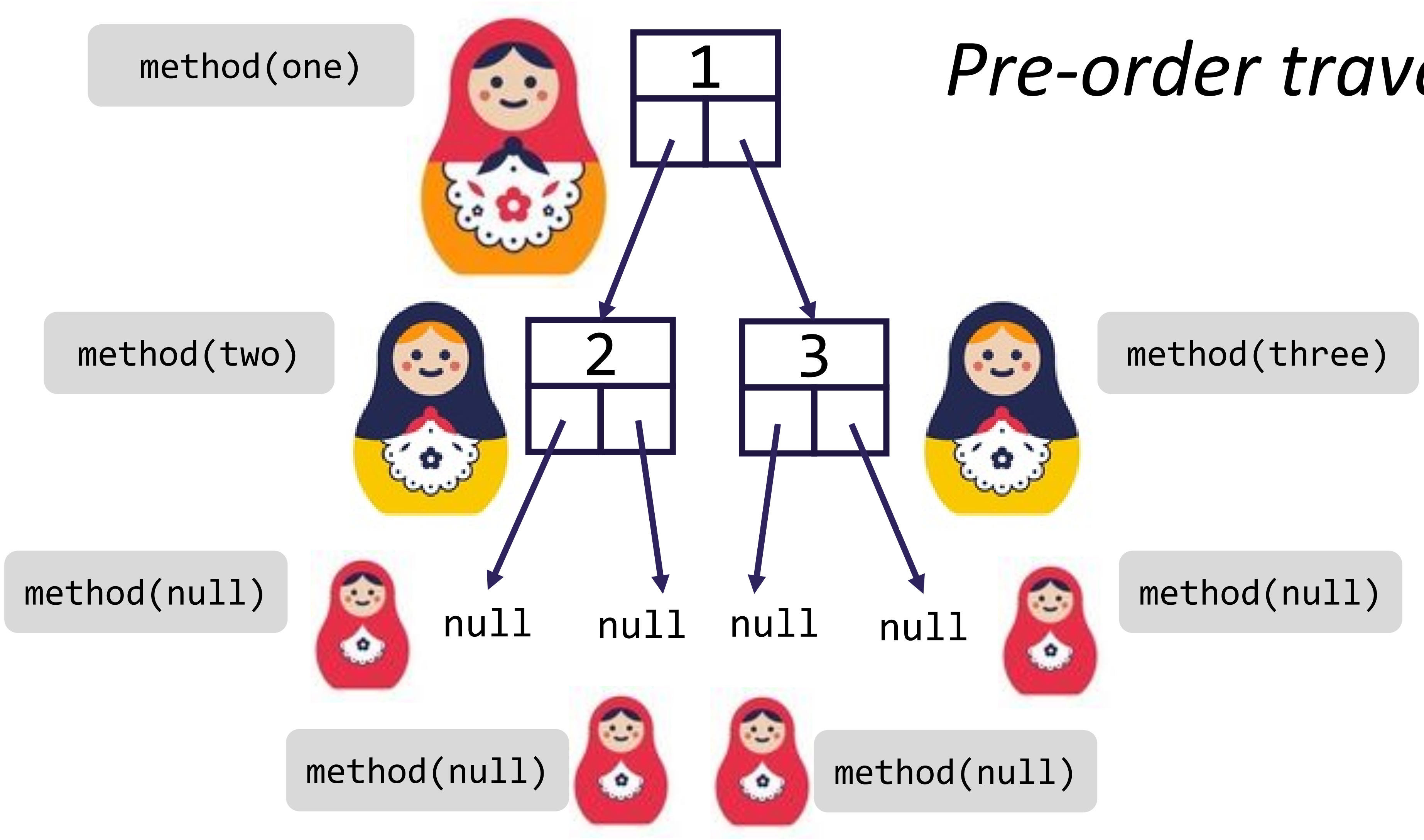
- Programs look very similar to Recursive LinkedList!

- Guaranteed base case: empty tree
  - Simplest possible input, should immediately know the return
- Guaranteed public / private pair
  - Need to know which subtree you're currently processing
- If modifying, we use `x = change(x)`
  - Don't stop early, return updated subtree (`IntTreeNode`)

- Let's trace through an example together…

Tracing Through Binary Tree Programming

Pre-order traversal



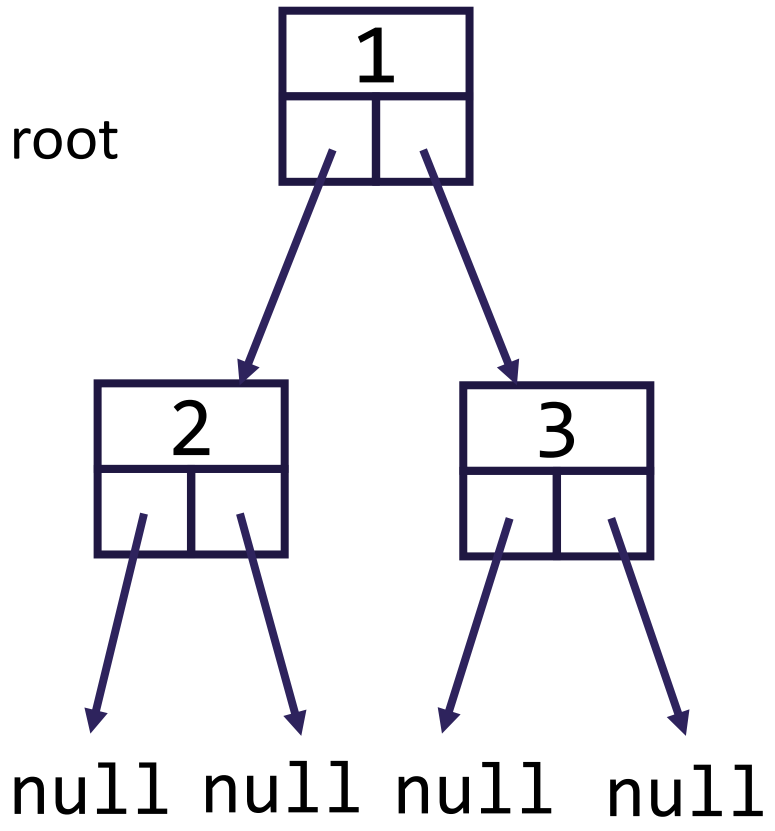
Lecture Outline

- Announcements
- Binary Tree Review
- **Traversals**
- Practice!

Binary Tree Traversals

- 3 different primary traversals
  - All concerned with when you process your current root

- Pre-order traversal:
  - Process **root**, left subtree, right subtree
- In-order traversal:
  - Process left subtree, **root**, right subtree
- Post-order traversal:
  - Process left subtree, right subtree, **root**

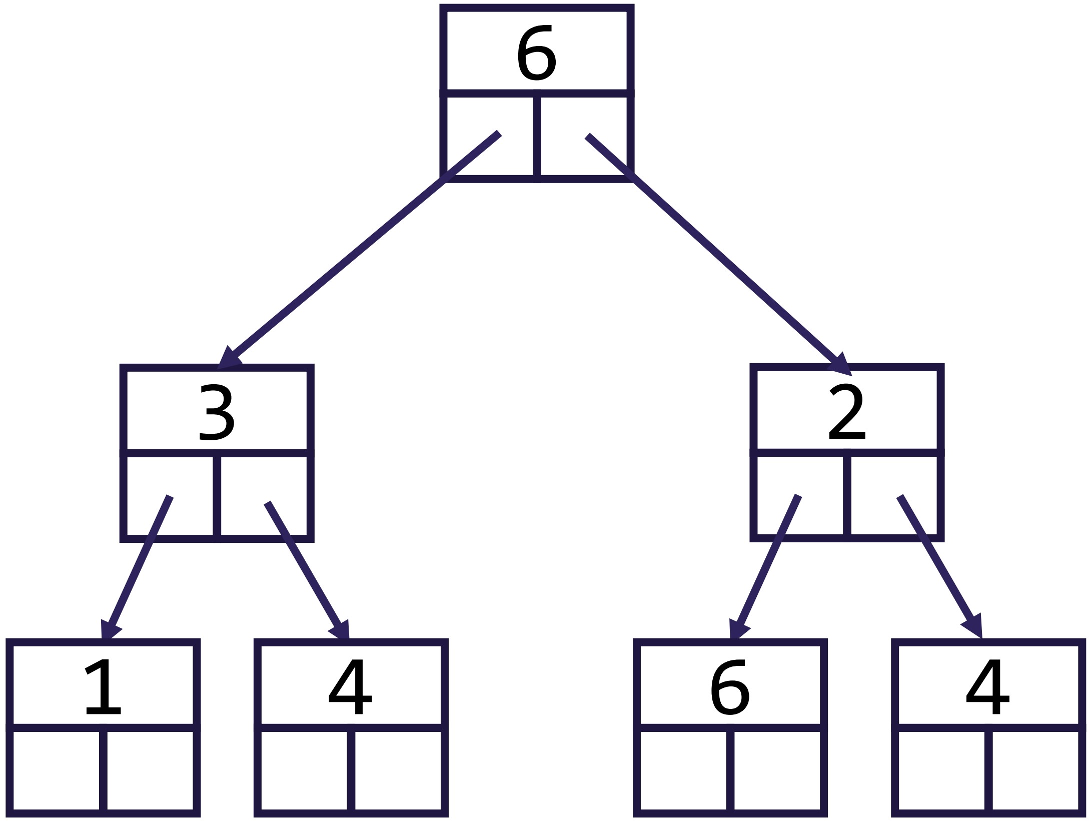


Sometimes different traversals yield different results

Practice : Think

sli.do #cse123

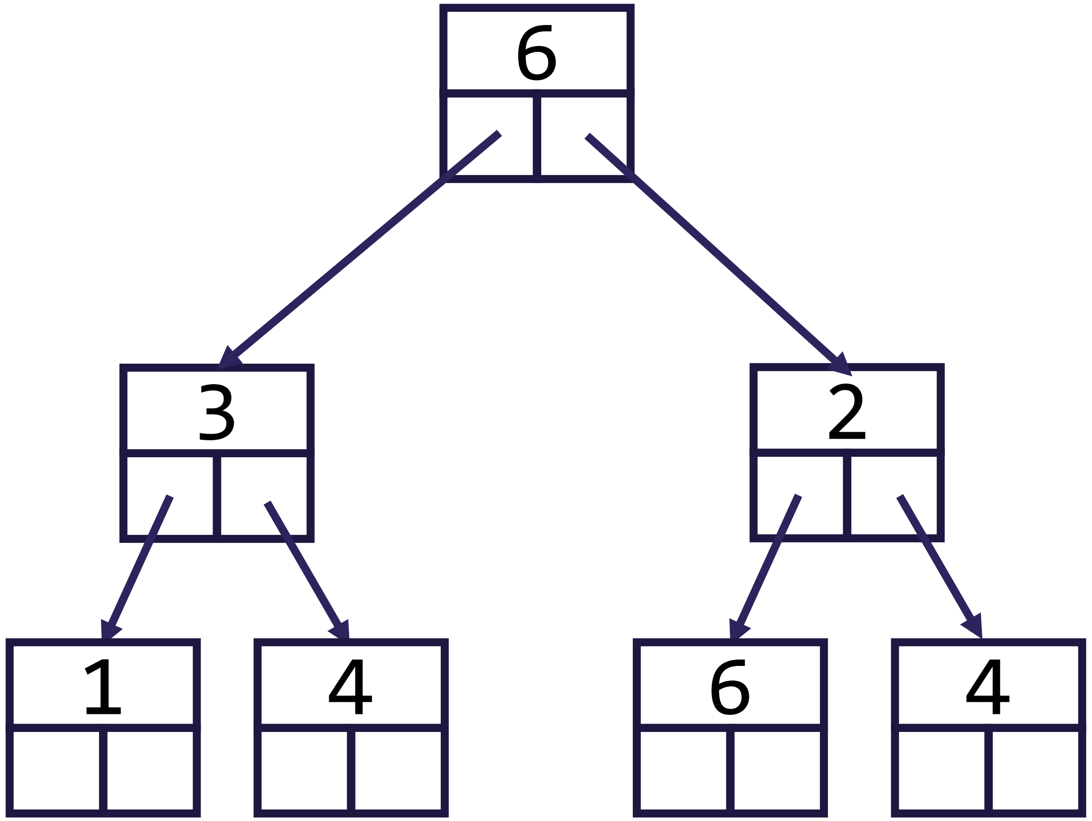
Enter the order in which the nodes of this tree would be visited in a pre-order traversal.



Practice : Pair

sli.do #cse123

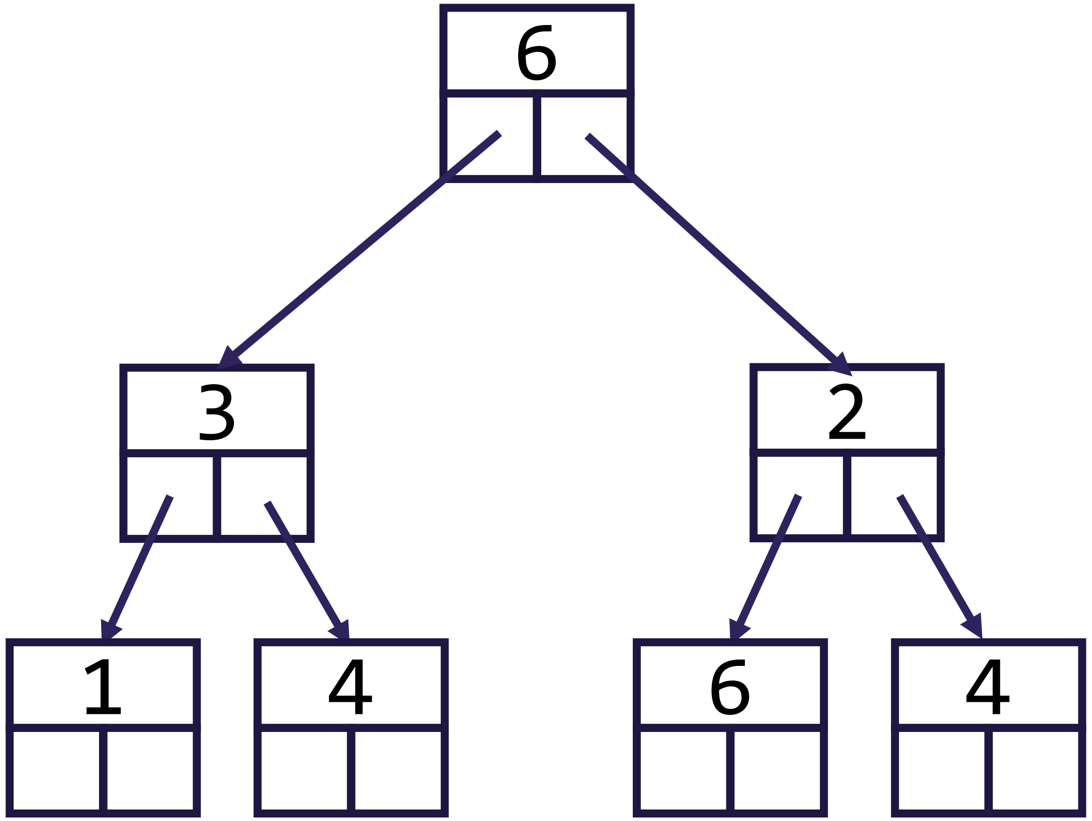
Enter the order in which the nodes of this tree would be visited in a pre-order traversal.



Practice : Pair

sli.do #cse123

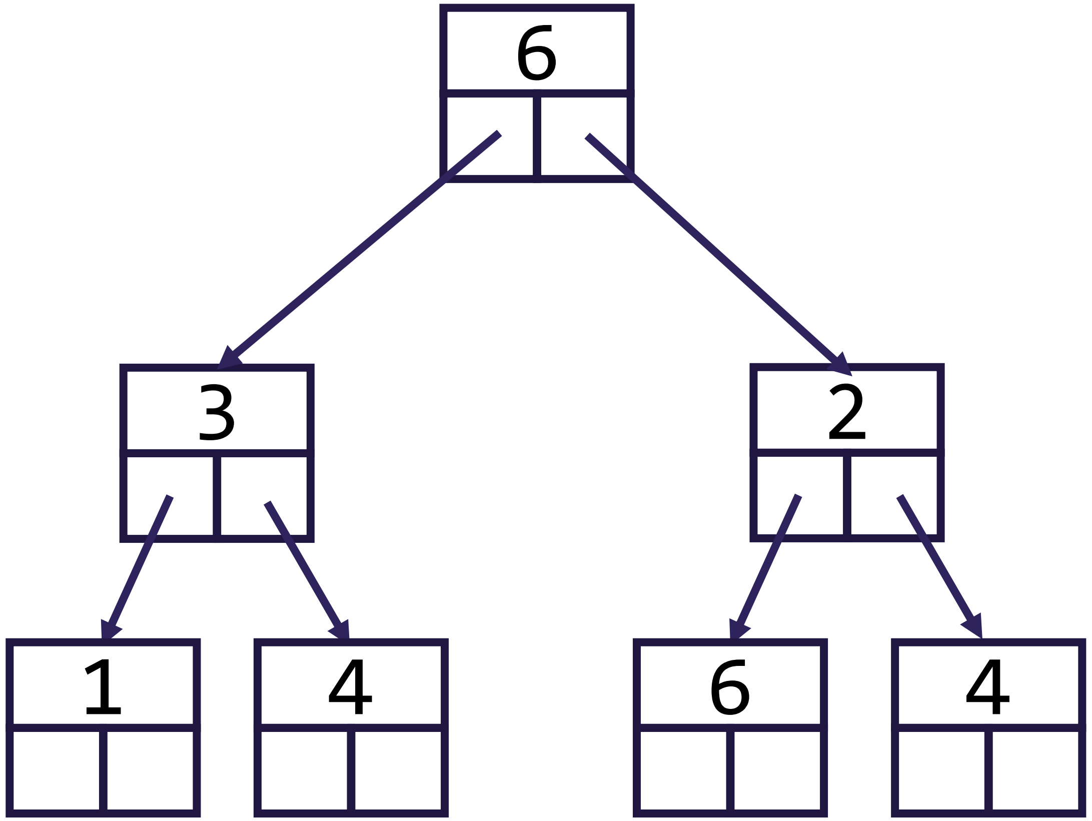
Enter the order in which the nodes of this tree would be visited in an <u>in-order</u> traversal.



Practice : Pair

sli.do #cse123

Enter the order in which the nodes of this tree would be visited in a post-order traversal.



Lecture Outline

- Announcements
- Binary Tree Review
- Traversals
- **Practice!**

Tracing through size

```java
public int size() {
    return size(overallRoot);
}

private int size(IntTreeNode currentRoot) {
    if (currentRoot == null) {
        return 0;
    } else {
        return 1 +
               size(currentRoot.left) +
               size(currentRoot.right);
    }
}
```