LEC 14

CSE 123

Binary Search Trees

Questions during Class?
Raise hand or send here

sli.do #cse123

BEFORE WE START

Talk to your neighbors:

Is water wet?

Respond on sli.do!

Instructor: Nathan Brunelle

TAs:

Cora	Chris	Crystal	Anya	Evan
Jonah	Nhan	Saachi	Ishita	Sahana
Chloe	Shiven	Nate	Renee	Yuntong
Amiya	Shashmit	Akshaya	Vihan	Shreedhaarini
Roshan	Adiyan	Nancy	Rachel	Ananya
Ryuta	Mimi	Isaac	Megan	

Lecture Outline: Announcements

- **Announcements**
- Binary Search Review
- Binary (Search) Trees Review
- More runtime!

Announcements

- CQ2 Ongoing!
- P3 released
  - Due Dec. 2

Lecture Outline: Binary Search Review

- Announcements
- **Binary Search Review** 
- Binary Search Trees

contains() on Collections

- contains() checks if an element is in a collection.
- How fast is contains() on an ArrayList? LinkedList? IntTree?
- Is it possible to be faster?

Looking through a dictionary

- How would you find your favorite word in the dictionary?
  - Look at every page?

- Can you find a word more quickly?

Looking through a dictionary pseudocode

- Assuming a *sorted order* of elements to search through `list`
- Suppose you're looking for a specific element `target`
- Return the index of the given `target`, or -1 if it's not in the `list`

```
begin with the dictionary, from the first to last word,
        looking for target

search(dictionary, left, right, target):
    if there are no more words to look through
        give up
    else
        pick a midpoint between left and right
        pick the word at that midpoint
        if target is that word
            found it!
        else if target comes before that word
            search(dictionary, left, midpoint-1, target)
        else (target comes after that word)
            search(dictionary, midpoint+1, right, target)
```

Binary Search

- Assuming a *sorted order* of elements to search through `list`
- Suppose you're looking for a specific element `target`
- Return the index of the given `target`, or -1 if it's not in the `list`

```
begin with search(list, 0, list.size() - 1, target)

search(list, left, right, target):
    if (left > right):
        return -1
    else:
        mid = (left + right) / 2
```

How would we add an element to the array while retaining our ability to do binary search?

-5	4	18	23	30	49	55	108	184

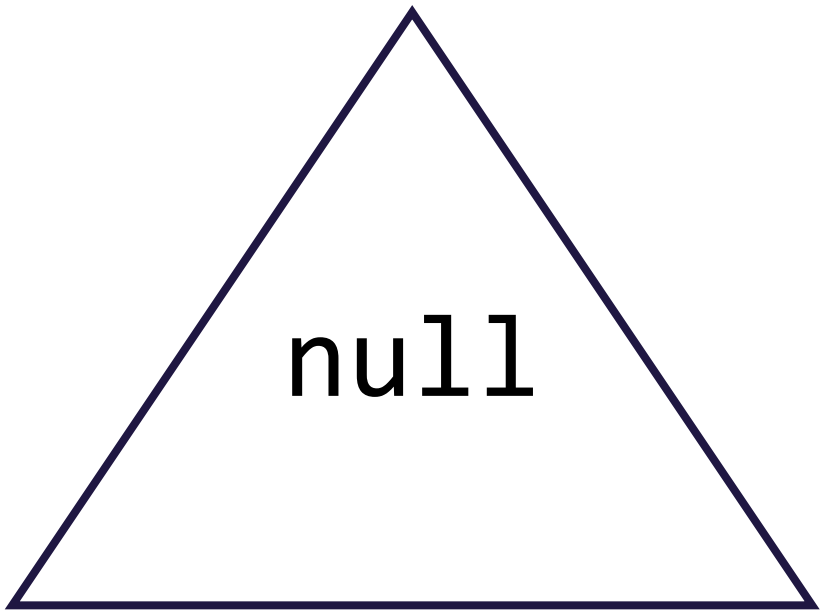
Lecture Outline: Binary (Search) Trees Review

- Announcements
- Binary Search Review
- **Binary Search Trees**



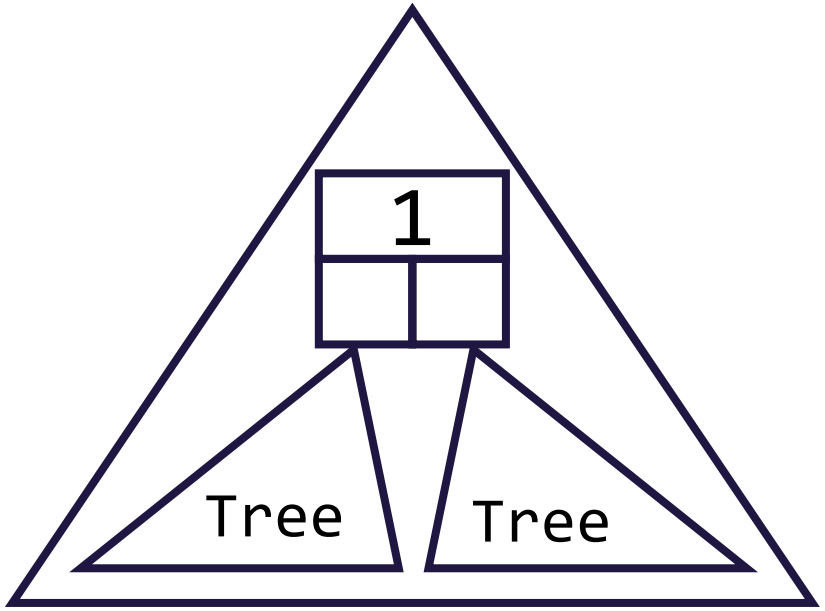
Binary Trees [Review]

- We'll say that any Binary Tree falls into one of the following categories:



Empty tree

`root == null`



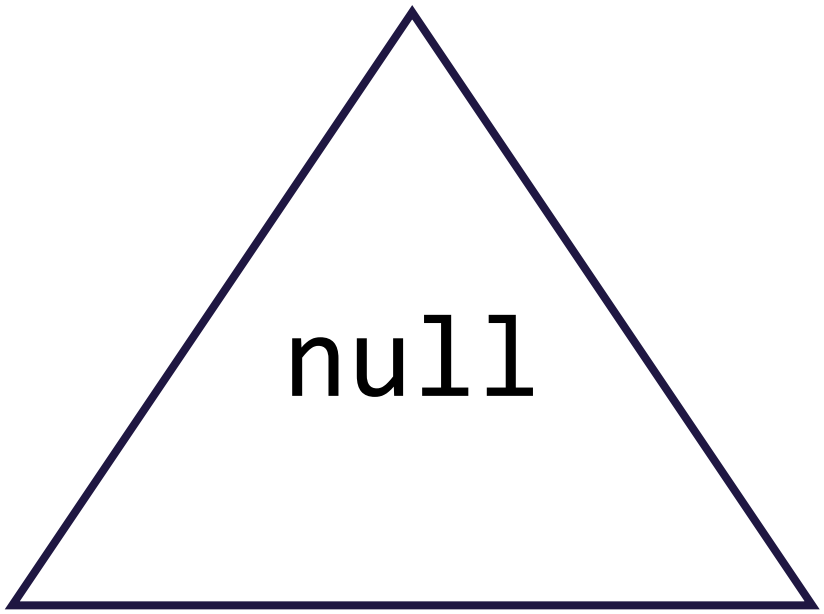
Node w/ two subtrees

```
root != null
root.left / root.right = Tree
```

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

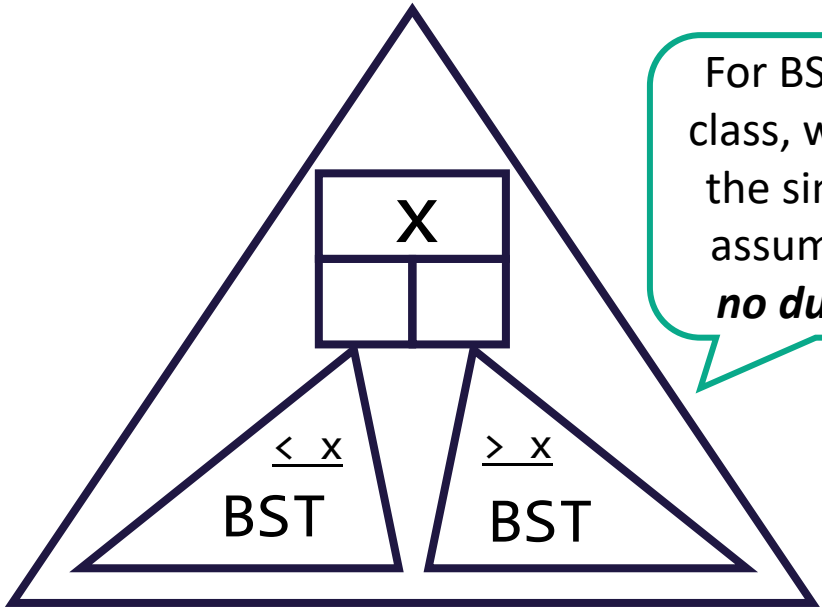
Binary Search Trees (BSTs)

- We'll say that any Binary Search Tree falls into the following categories:



Empty BST

```
root == null
```



For BSTs in this class, we'll make the simplifying assumption of ***no duplicates***

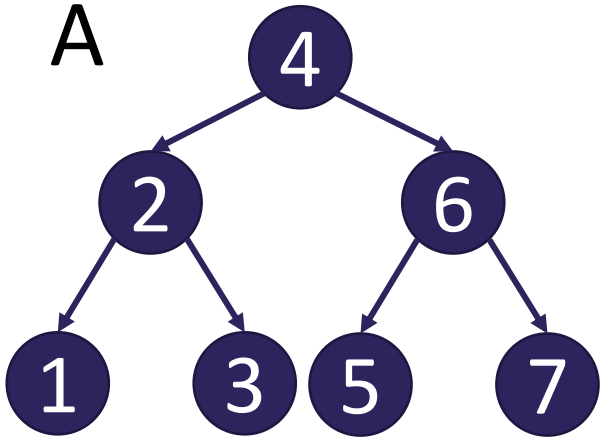
Node w/ two subBSTs

```
root != null
root.left / root.right = Tree
max(root.left) < x && min(root.right) > x
```

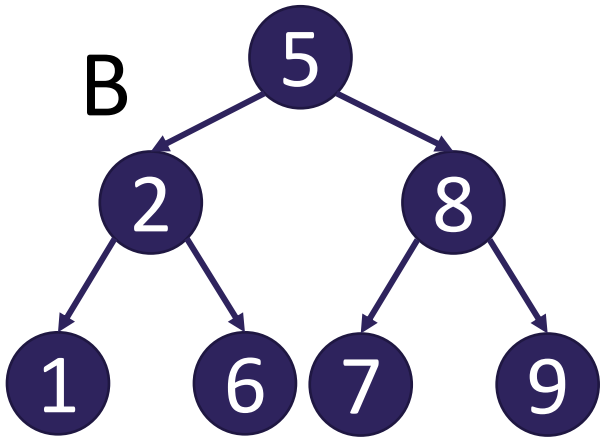
Note that not all Binary Trees are Binary Search Trees

Example Trees: Which are BSTs?

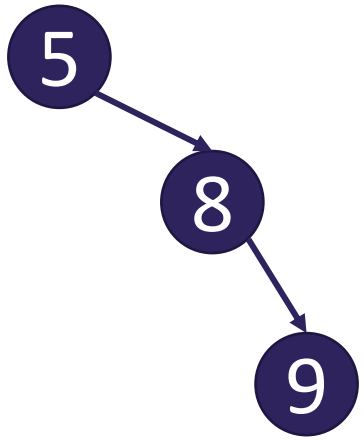
A



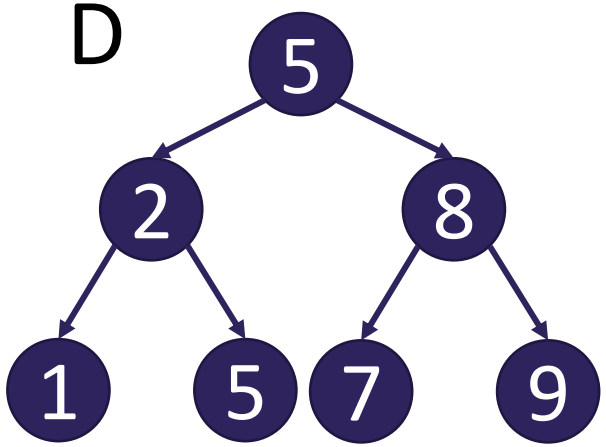
B



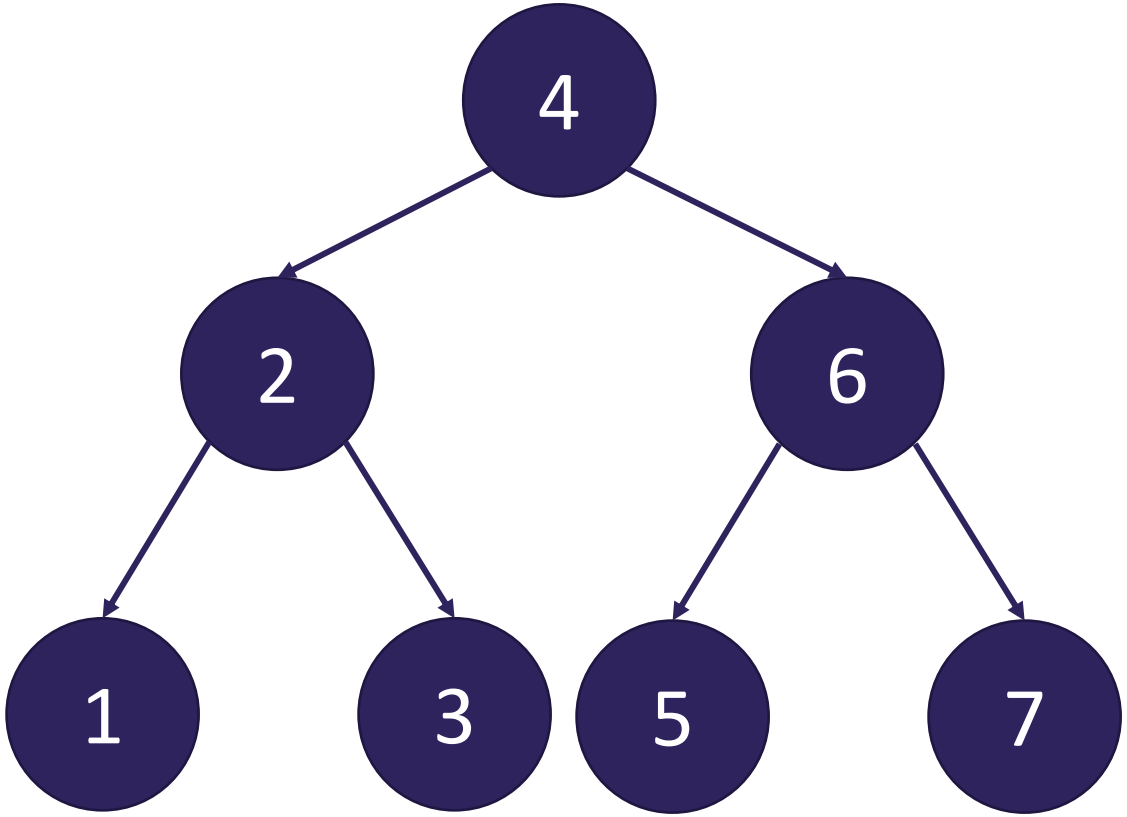
C



D



Example Method: contains



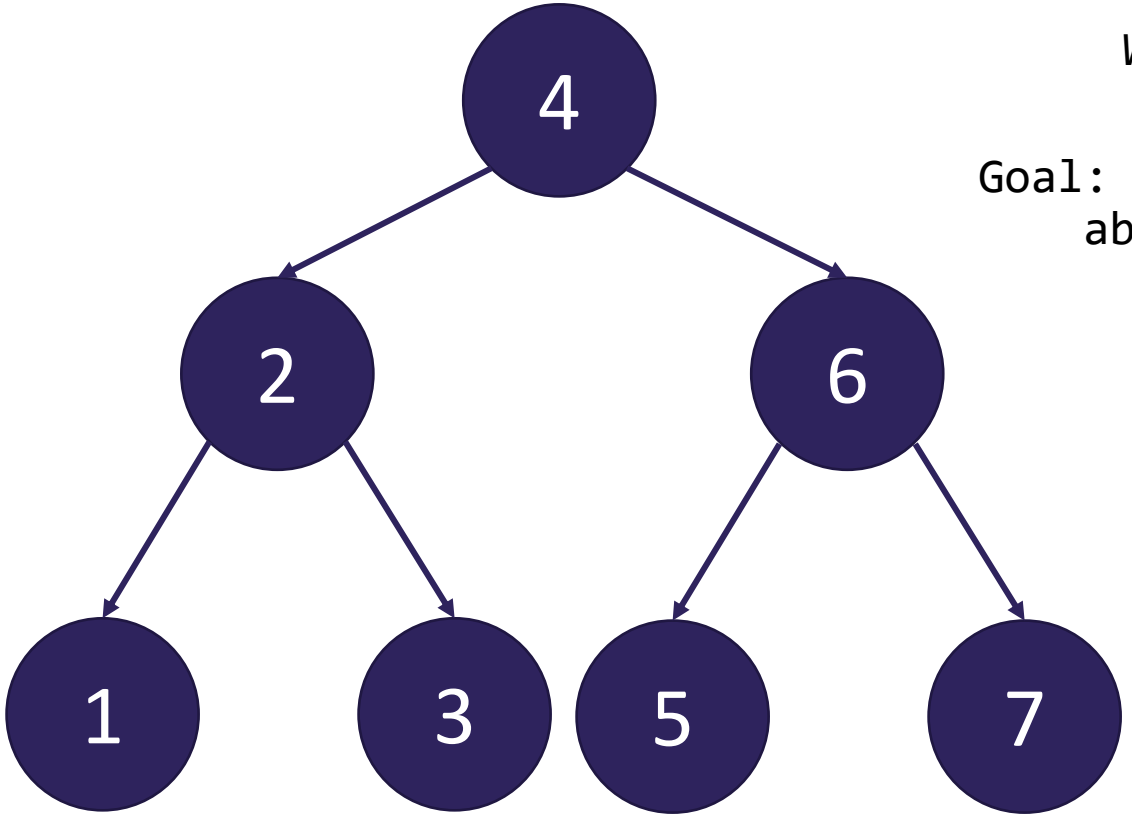
Why BSTs?

- Our `IntTree` implementation to `contains(int value)`

```
private boolean contains(int value, IntTreeNode root) {
    if (root == null) {
        return false;
    } else {
        return root.data == value ||
               contains(value, root.left) ||
               contains(value, root.right);
    }
}
```

- Which direction(s) do we travel if `root.data != value`?
  - Both left and right
- In a Binary Search Tree, should we check both sides?
  - Remember, additional constraint: `max(root.left) < root.data &&`
    `min(root.right) > root.data`

Example Method: add

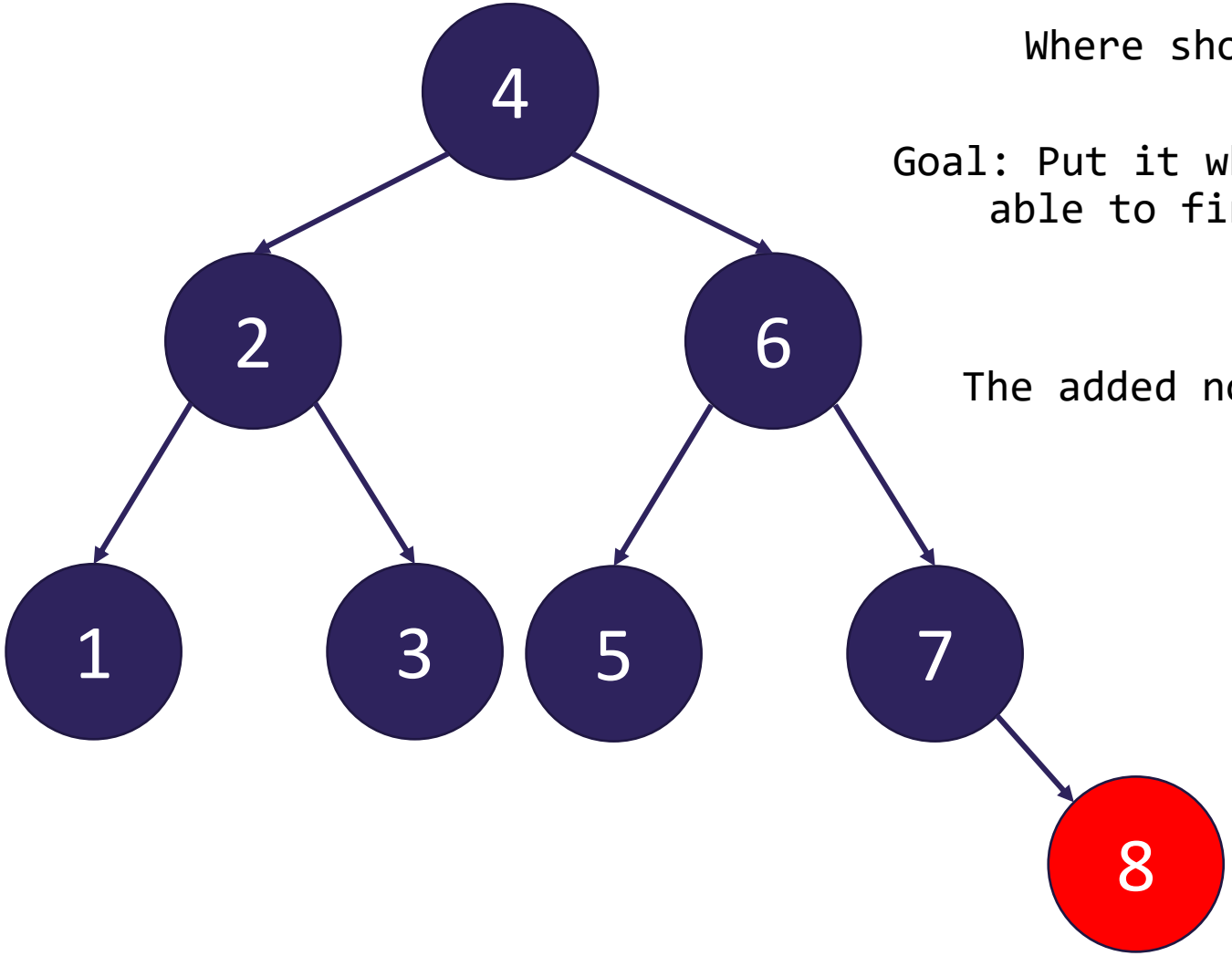


How would we add 8 to this tree?

Where should it go?

Goal: Put it where we would be able to find it later!

Example Method: add



How would we add 8 to this tree?

Where should it go?

Goal: Put it where we would be able to find it later!

The added node will always be a leaf!