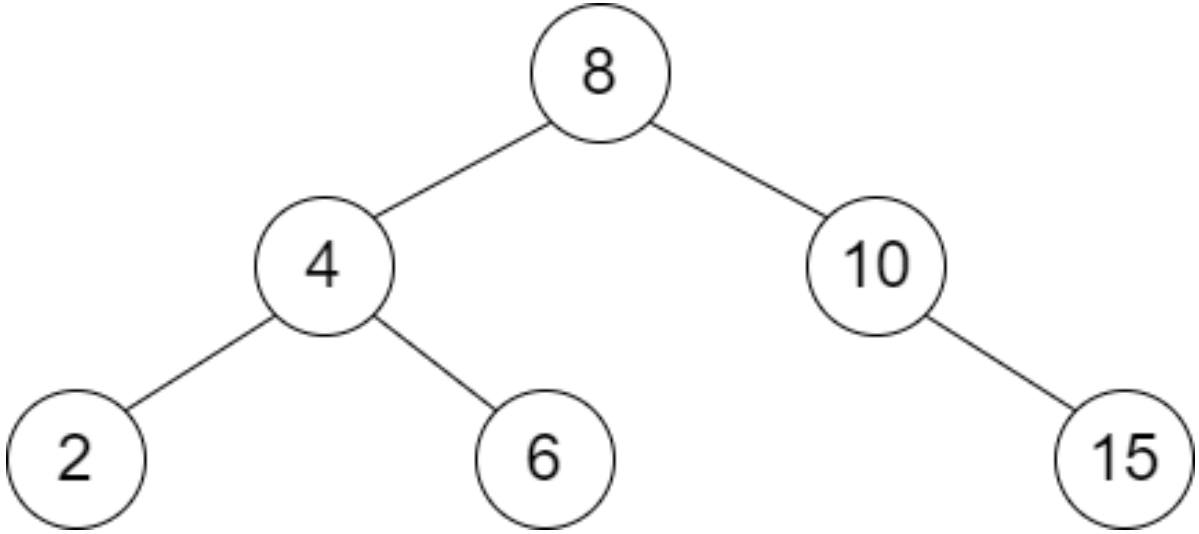
Today

- Binary Search Trees
- Verifying Structure
- Pre-Order Traversal
- Searching

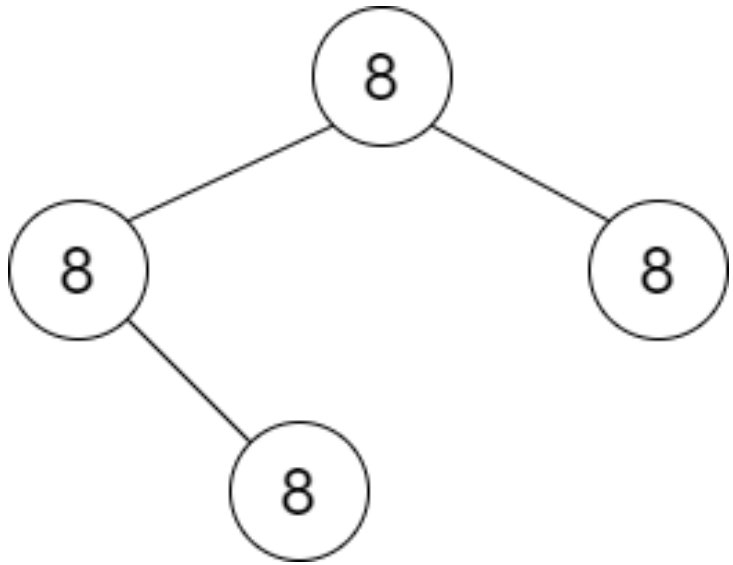
Binary Search Trees

- What is a binary search tree (BST)?
- Tree
  - Collection of nodes (or vertices) and edges connecting them
  - No cycles
- Binary
  - Each node has at most two children, left and right
- Search
  - Nodes store values
  - If a node has value v and u is a value of a node in the left subtree, $u \leq v$
  - If a node has value v and u is a value of a node in the right subtree, $u > v$

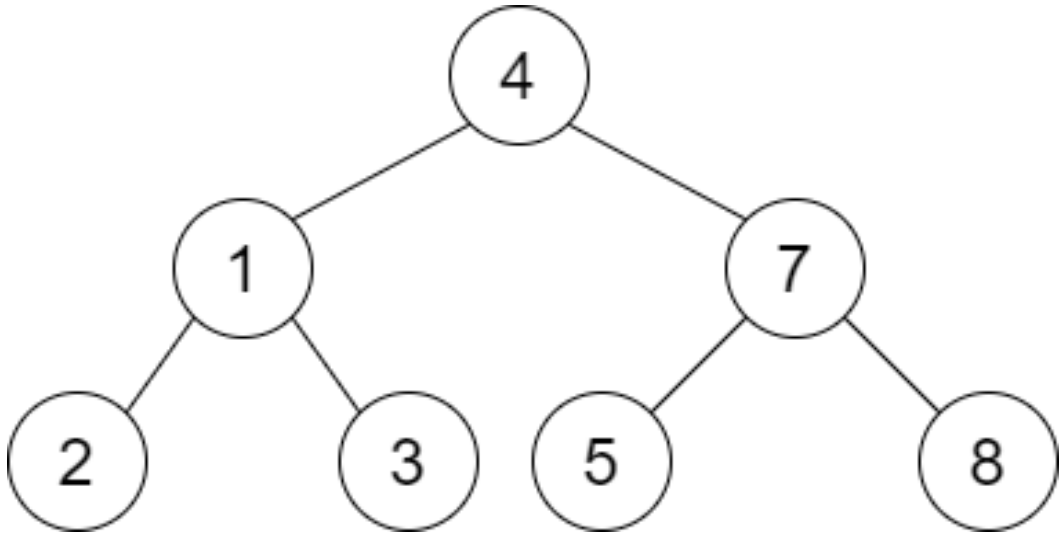
Binary Search Trees



Binary Search Trees



Binary Search Trees



Binary Search Trees

- If T is a BST
  - T has a root
  - Each node v has a value or key, $v.value$
  - Each node v has two children, $v.left$ and $v.right$
  - If a child is missing, it is set to *null*
  - $v.value \geq u.value$ for all u to the left of v
  - $v.value < u.value$ for all u to the right of v

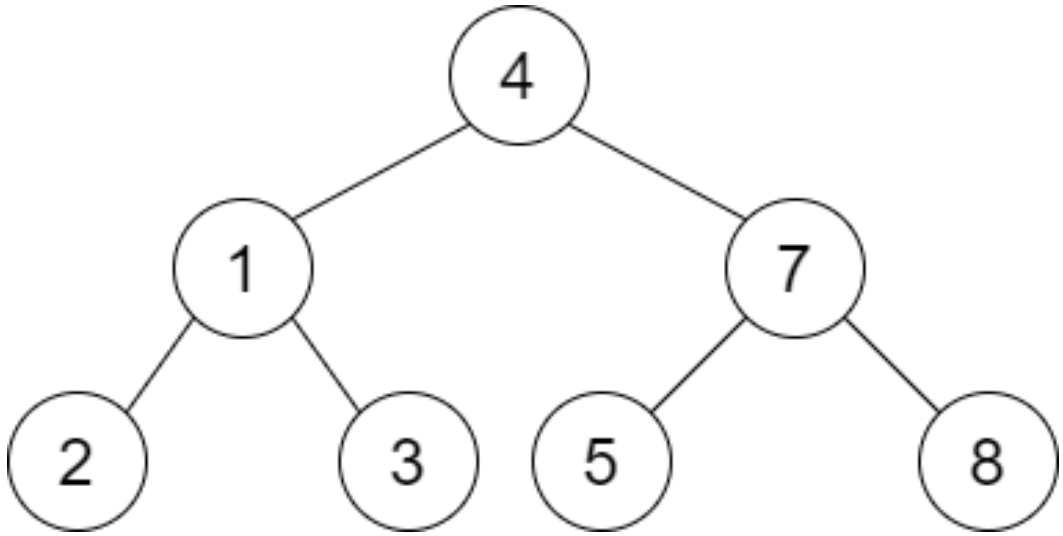
Verifying Structure

- Describe a recursive algorithm for checking that a tree is a BST
- We are given a root and a lower and upper bound on values in the tree
- What is the base case?
- How does being a BST depend on subtrees?

Verifying Structure

```
function isBST(v, low, high)
  if v == null
    return true
  if v.value < low or v.value  > high
    return false
  return isBST(v.left, low, v.value) and
         isBST(v.right, v.value, high)
```

Verifying Structure



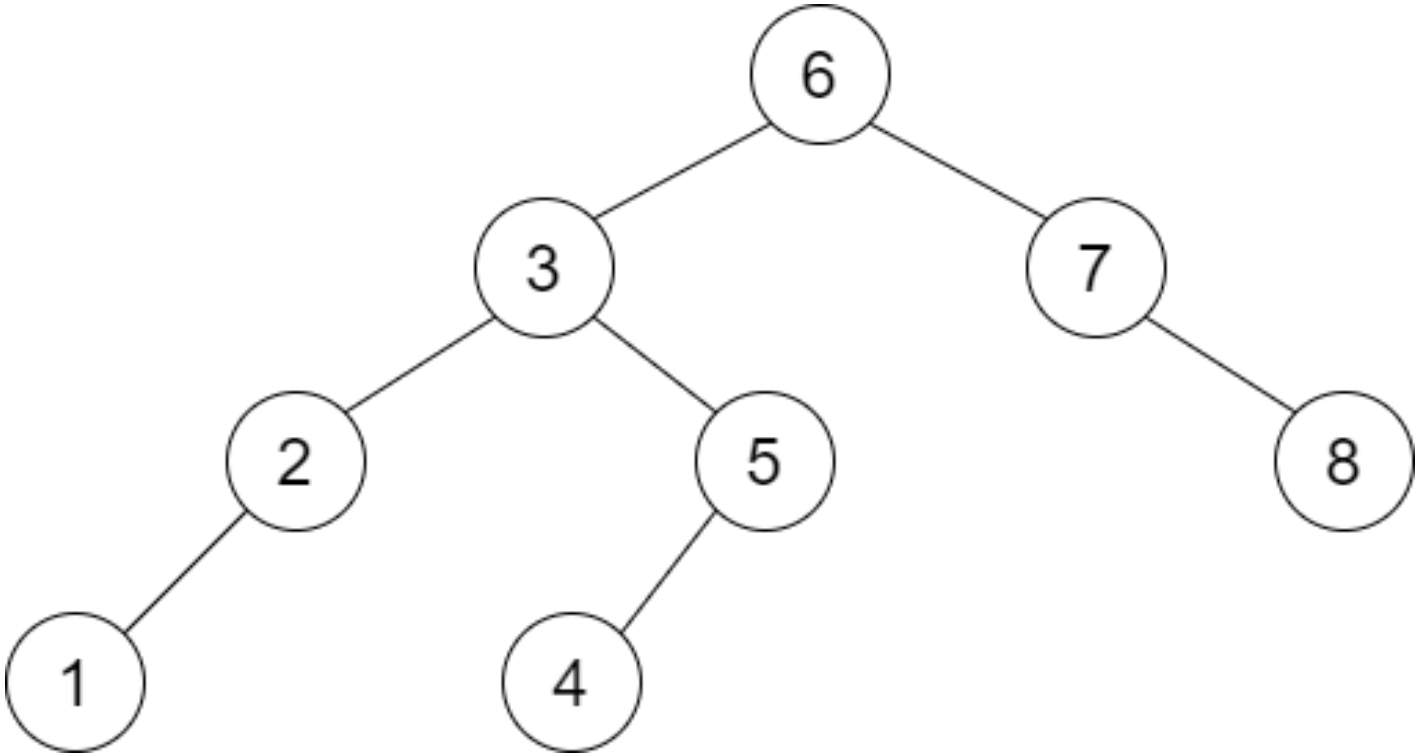
Pre-Order Traversal

- How can we print all the values of a BST?
- Consider a recursive approach
- If the node is not *null*, in what order should we consider the left subtree, the right subtree, and the value of this node itself?

Pre-Order Traversal

```
function preOrder(v)
  if v != null
    print(v.value)
    preOrder(left)
    preOrder(right)
```

Pre-Order Traversal



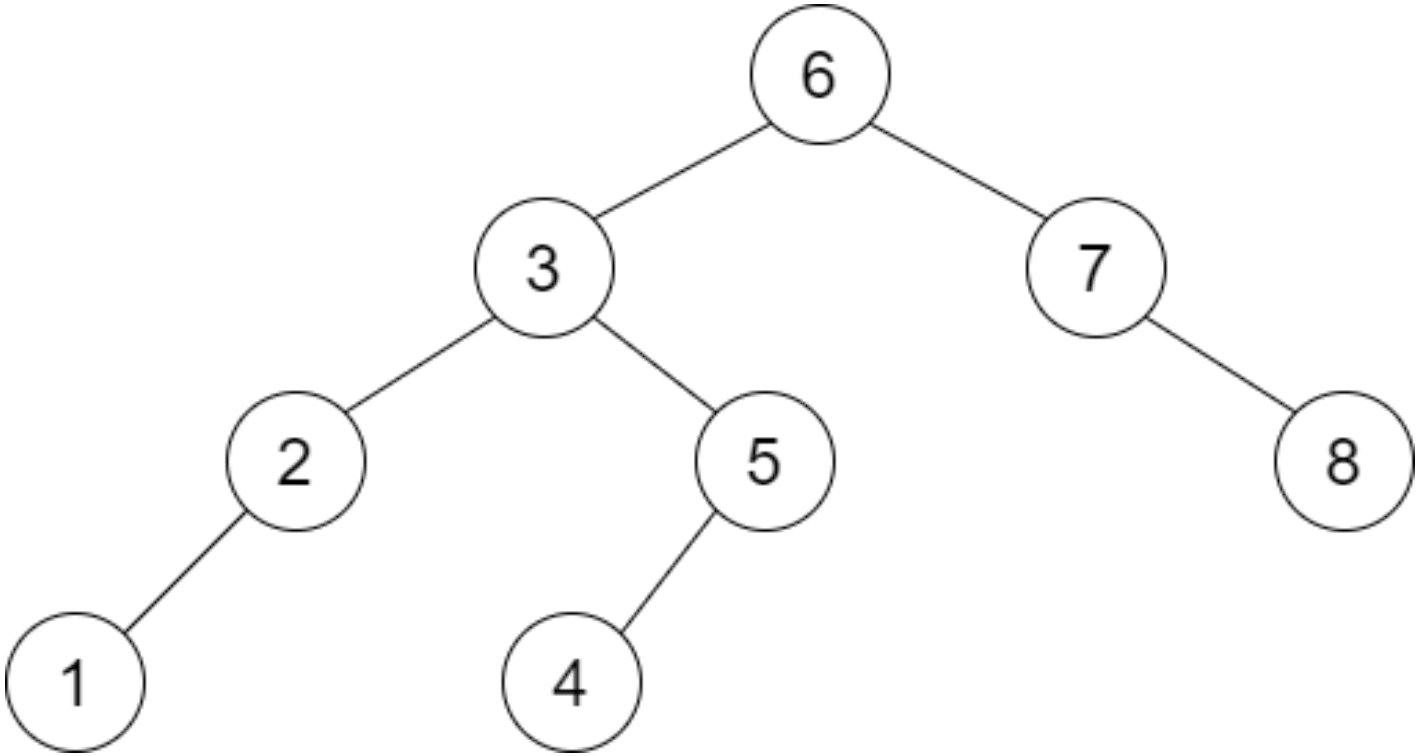
Searching

- Given a BST T, how can we determine whether or not a specific value is in the tree?
- Start with a node v and a target value m
- What is the base case?
- What should we do at each node?

Searching

```
function search(v, m)
  if v == null
    return null
  if v.value == m
    return v
  if v.value > m
    return search(v.left, m)
  if v.value < m
    return search(v.right, m)
```

Searching



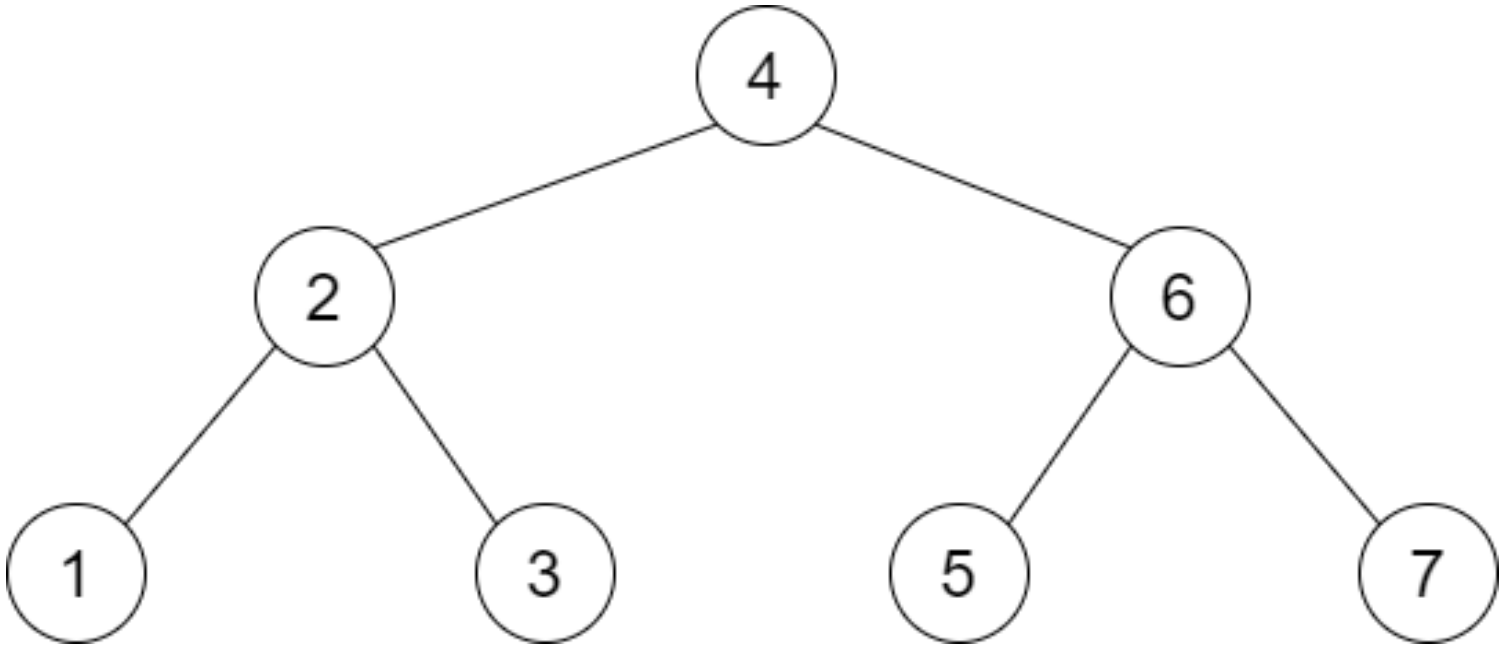
Searching

```
function search(v, m)
  if v == null
    return null
  if v.value == m
    return v
  if v.value > m
    return search(v.left, m)
  if v.value < m
    return search(v.right, m)
```

- What is the run time of this function?

Searching

A Full Binary Tree is a special type of binary tree in which every parent node/internal node has either two or no children

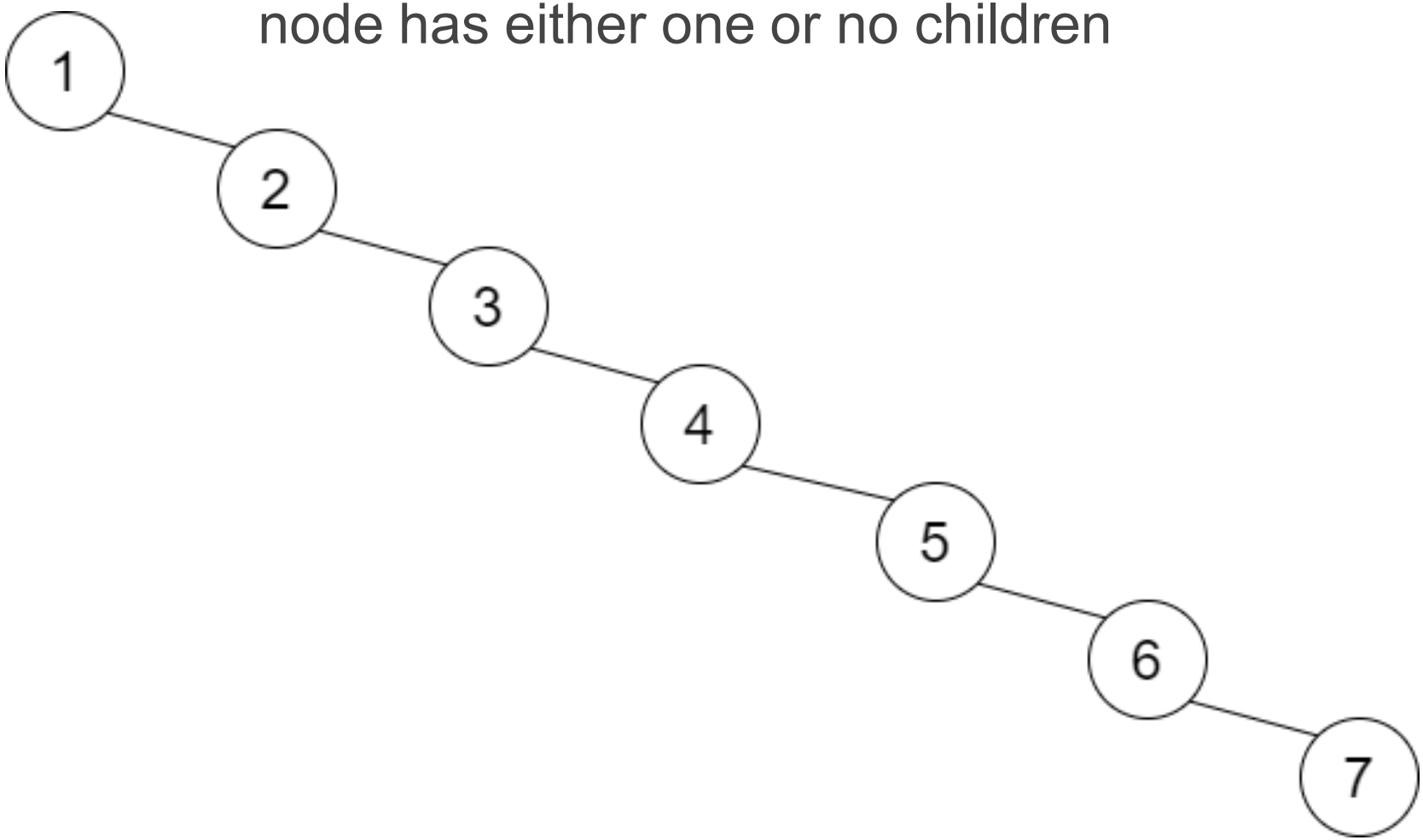


Best case: O(1)

Average Case: O(log(n))

Searching

A “Single-legged” Binary Tree is a special type of binary tree in which every parent node/internal node has either one or no children



worst case: O(n)

Searching