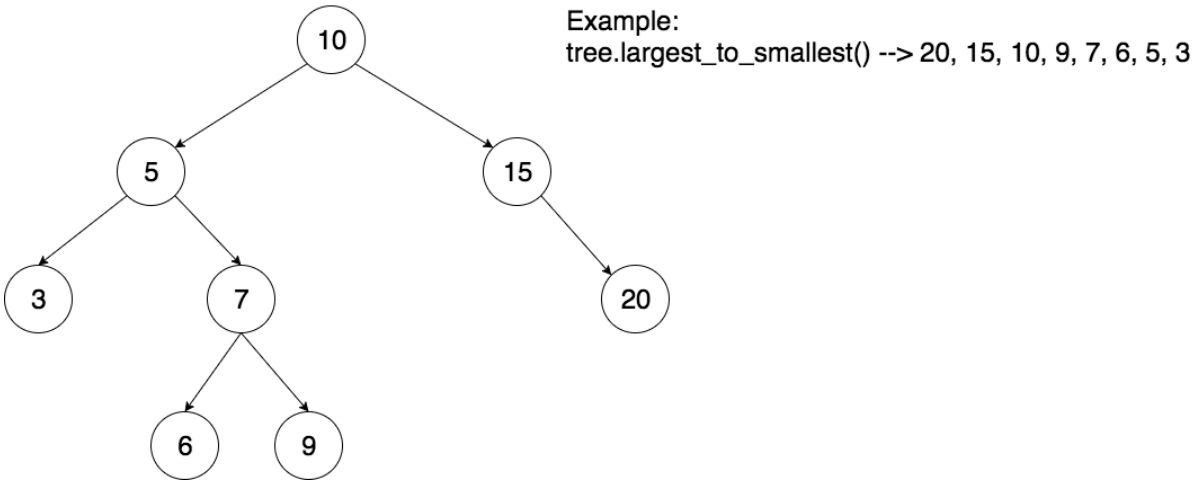
CSCI-SHU 210 Data Structures

Assignment 8 Binary Search Trees

*Assignment 8 tasks are located at line 408 in Assignment8.py

Problem 1: Largest to smallest for a BST



Implement function `largest_to_smallest(self)`. When called, it should return a list of values from self BST, from the largest value to the smallest value order.

Important:

- Your implementation's time complexity should be O(N). Where N is the size of self BST.
- You can define helper functions with additional parameters to perform recursion tasks.

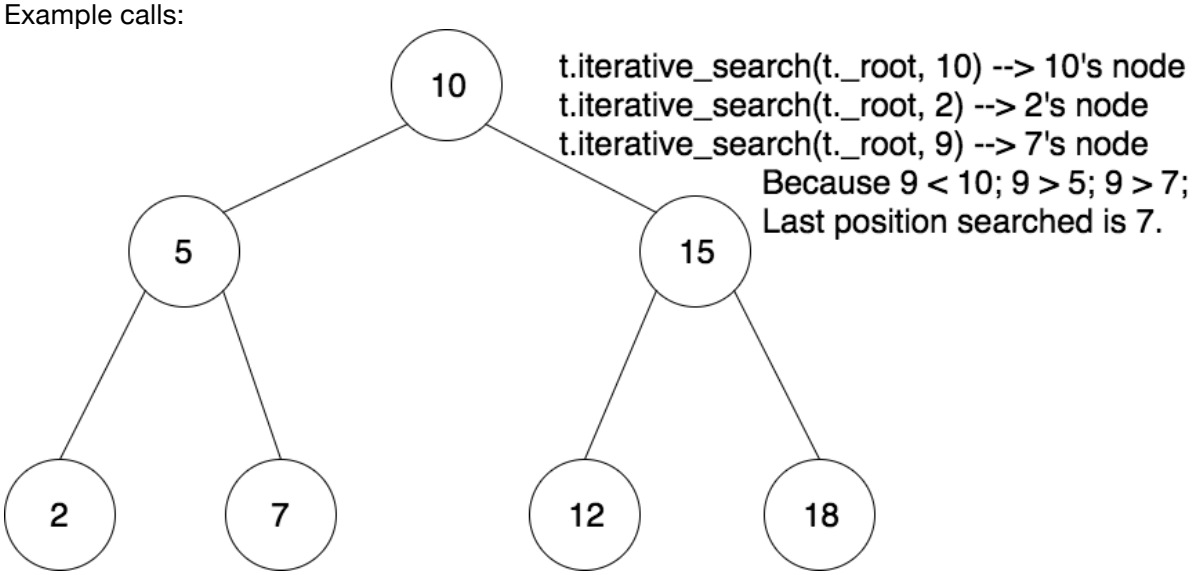
Problem 2: Iterative search in BST

In class BinarySearchTree, our search function `_subtree_search(self, node, v)` is implemented recursively.

"""Return the node having value v, or last node searched."""

Your task: Implement function `iterative_search(self, node, v)`, which performs same job as _subtree_search, but iteratively.

Example calls:



Important:

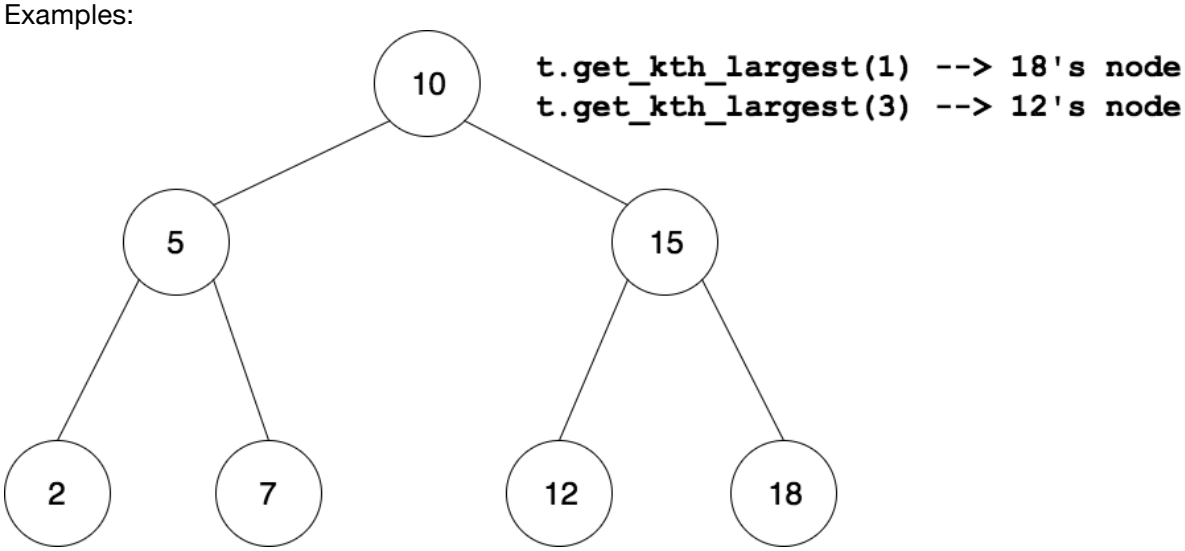
- Same job means, for the same tree, same parameters are given, `iterative_search` should return the exact same **TreeNode** as `_subtree_search.`
- Your function should return a **TreeNode**!
- You cannot use recursion.
- You can reuse any function from Binary Tree, also reuse any function from Binary Search Tree.

Problem 3: Find k-th largest element in BST

Implement function `get_kth_largest(self, k)`, which returns the k-th largest node within self Binary Search Tree.

If k is too large, return the smallest element's node within the tree.
If k is too small, return the largest element's node within the tree.

Examples:



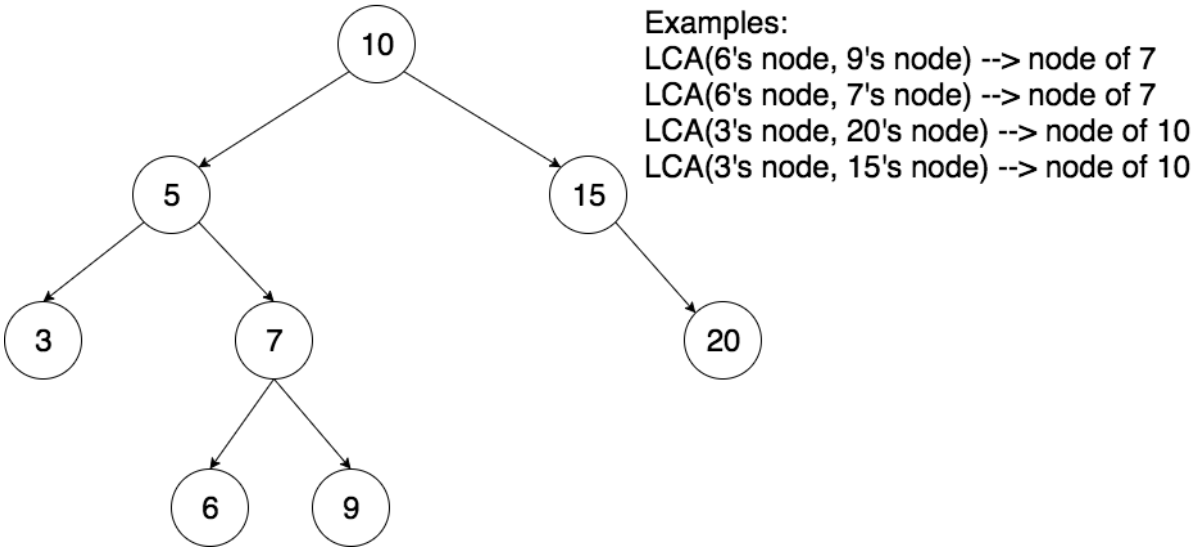
Important:

- Your function should return a **TreeNode**!
- You can reuse any function from Binary Tree, also reuse any function from Binary Search Tree.

Problem 4: Lowest common ancestor in BST

Your task is to solve C-8.58:

*Let T be a tree with n nodes. Define the **lowest common ancestor** (LCA) between two nodes p and q as the lowest node in tree T that has both p and q as descendants (where we allow a node to be a descendant of itself). Given two nodes p and q, describe an efficient algorithm for finding the LCA of p and q.*



Implement function `LCA(self, node1, node2)`. When called, it should return the **TreeNode** of the lowest common ancestor.

Important:

- Make sure your return type is **TreeNode**, so I can call **TreeNode**._element to test your code.
- You can reuse any function from Binary Tree, or any function Binary Search Tree.