CSCI-SHU 210 Data Structures

Assignment 8 Binary Search Trees

*Assignment 8 tasks are located at line 153 in BST.py

**I'll upload the standard LinkedQueue.py, LinkedBinaryTree.py on gradescope, which means as long as you don't modify those two files, they should behave exactly the same. (Running on gradescope & running on your computer)

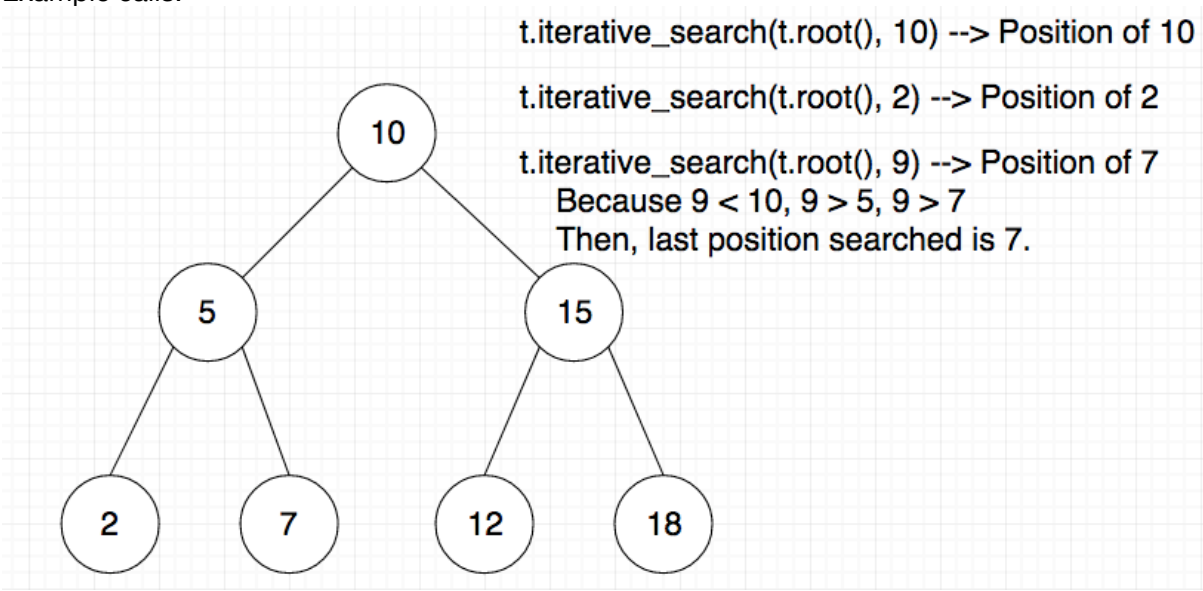
Problem 1: Iterative search in BST

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

"""Return Position of p's subtree having value v, or last Position searched."""

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

Example calls:



Important:

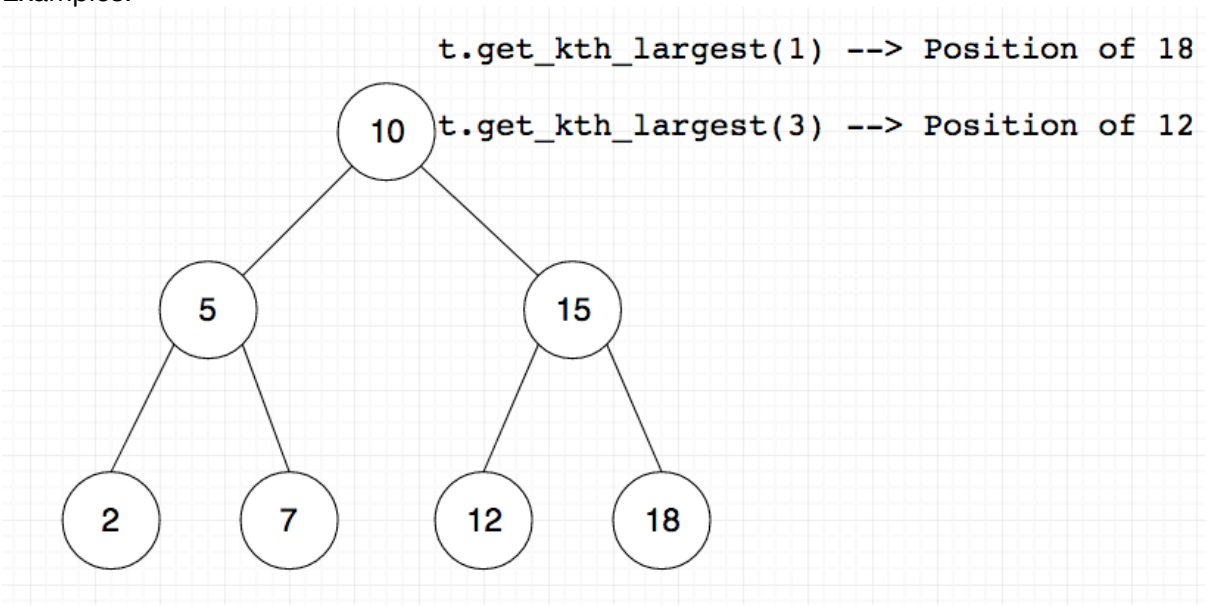
- Same job means, if same tree, same parameters are given, `iterative_search` should return the exact same position as `_subtree_search.`
- Your function should return a Position!
- You can reuse any function from Binary Tree, also any function Binary Search Tree.

Problem 2: Find k-th largest element in BST

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

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

Examples:



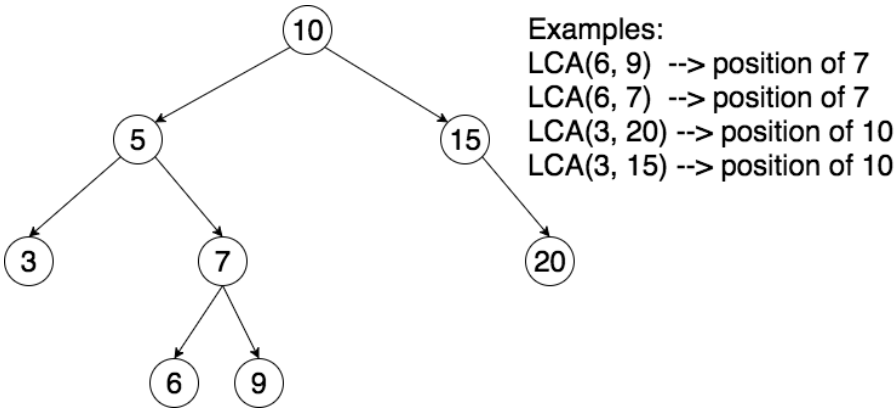
Important:

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

Problem 3: Lowest common ancestor in BST

Your task is to solve C-8.58:

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



Implement function `LCA(self, p1, p2)`. When called, it should return the **Position** of the lowest common ancestor.

Important:

- Make sure your return type is **Position**, so we can call Position.element() to test your code.
- You can reuse any function from Binary Tree, also any function Binary Search Tree.