Interview Cake

Write a function to find the 2nd largest element in a binary search tree.

A **binary search tree** is a **binary tree** in which, for each node:

1. The node's value is greater than all values in the left subtree.
2. The node's value is less than all values in the right subtree.

BSTs are useful for quick lookups. If the tree is **balanced**, we can search for a given value in the tree in $O(\lg n)$ time.

Here's a sample binary tree node class:

Python

```
class BinaryTreeNode:

    def __init__(self, value):
        self.value = value
        self.left  = None
        self.right = None

    def insert_left(self, value):
        self.left = BinaryTreeNode(value)
        return self.left

    def insert_right(self, value):
        self.right = BinaryTreeNode(value)
        return self.right
```

Gotchas

Our first thought might be to do an in-order traversal of the BST and return the second-to-last item. This means looking at *every node in the* BST. That would take $O(n)$ time and $O(h)$ space, where h is the max *height* of the tree (which is $lg n$ if the tree is balanced, but could be as much as

n if not).

We can do better than $O(n)$ time and $O(h)$ space.

We can do this in *one* walk from top to bottom of our BST. This means $O(h)$ time (again, that's $O(\lg n)$ if the tree is balanced, $O(n)$ otherwise).

A clean recursive implementation will take $O(h)$ space in the call stack,

The **call stack** is what a program uses to keep track of what function it's currently running and what to do with that function's return value.

Whenever you call a function, a new **frame** gets pushed onto the call stack, which is popped off when the function returns. As functions call other functions, the stack gets taller. In recursive functions, the stack can get as tall as the number of times the function calls itself. This can cause a problem: the stack has a limited amount of space, and if it gets too big you can get a **stack overflow** error.

but we can bring our algorithm down to $O(1)$ space overall.