






Delete a Leaf Node in a Binary Search Tree

This is the first of three challenges where we will implement a more difficult operation in binary search trees: deletion. Deletion is difficult because removing nodes breaks links in the tree. These links must be carefully reestablished to ensure the binary tree structure is maintained. For some deletions, this means the tree must be rearranged. In general, you will encounter one of three cases when trying to delete a node: Leaf Node: The target to delete has zero children. One Child: The target to delete only has one child. Two Children: The target to delete has two child nodes. Removing a leaf node is easy, we simply remove it. Deleting a node with one child is also relatively easy, we simply remove it and link its parent to child of the node we deleted. Removing a node with two children is more difficult, however, because this creates two child nodes that need to be reconnected to the parent tree. We'll see how to deal with this case in the third challenge. Additionally, you need to be mindful of some edge cases when handling deletion. What if the tree is empty? What if the node to delete is the root node? What if there are only two elements in the tree? For now, let's handle the first case where we delete a leaf node.

Create a method on our binary tree called `remove`. We'll build the logic for our deletion operation in here. First, you'll want to create a function within `remove` that finds the node we are trying to delete in the current tree. If the node is not present in the tree, `remove` should return `null`. Now, if the target node is a leaf node with no children, then the parent reference to it should be set to `null`. This effectively deletes the node from the tree. To do this, you will have to keep track of the parent of the node we are trying to delete as well. It will also be useful to create a way to track the number of children the target node has, as this will determine which case our deletion falls under. We will handle the second and third cases in the next challenges. Good luck!

	The <code>BinarySearchTree</code> data structure should exist.
	The binary search tree should have a method called <code>remove</code> .

	Trying to remove an element from an empty tree should return <code>null</code> .
	Trying to remove an element that does not exist should return <code>null</code> .
	The <code>remove</code> method should remove leaf nodes from the tree.