CSC148 winter 2017

mutating BSTs

week 9

Danny Heap

heap@cs.toronto.edu / BA4270 (behind elevators)

http://www.teach.cs.toronto.edu/~csc148h/winter/

416-978-5899

March 13, 2017

Outline

term test #2

binary search tree operations

mutating binary search tree

test coverage

possible topics...

- **LinkedListNode** and **LinkedList**
- recursion on nested Python **list**
- recursion on class **Tree**
- recursion on class **BinaryTree**
- definitions for trees and binary trees, traversals (inorder, postorder, preorder, levelorder)

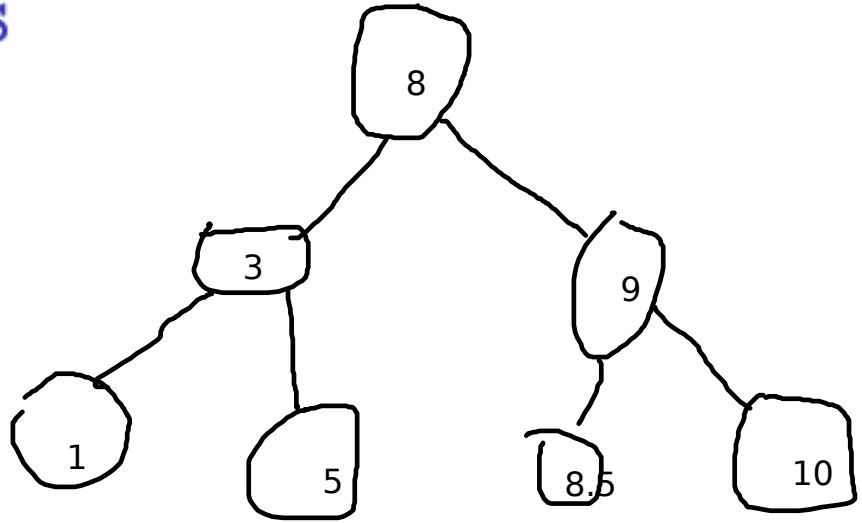
recall BinaryTree...

```
class BinaryTree

    def __init__(self, value, left=None, right=None):
        """
        Create BinaryTree self with value and children left and right.

        @param BinaryTree self: this binary tree
        @param object value: value of this node
        @param BinaryTree|None left: left child
        @param BinaryTree|None right: right child
        @rtype: None
        """
        self.value, self.left, self.right = value, left, right
```

bst_contains



same old contains, but more efficient...

... we only need to traverse a path from root to leaf...

insert must obey BST condition

example shows that we expect insert to ensure this is a binary search tree:

```
def insert(node, value):
    """
    Insert value in BST rooted at node if necessary, and return new roo

    Assume node is the root of a Binary Search Tree.

    @param BinaryTree node: root of a binary search tree.
    @param object value: value to insert into BST, if necessary.
    @rtype: BinaryTree
    >>> b = BinaryTree(5)
    >>> b1 = insert(b, 3)
    >>> print(b1)
    5
        3
    <BLANKLINE>
    """
```

We return the root in case it has changed...

deletion of value from BST rooted at node?

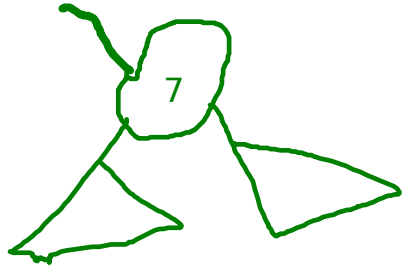
cases to consider...

- what return value?
- what to do if node is None?
- what if value to delete is less than that at node?
- what if it's more?
- what if the value equals this node's value and...
  - this node has no left child
  - ... no right child?
  - both children?

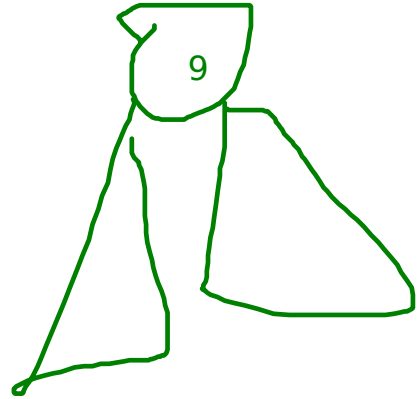
algorithm...

There are so many cases to consider (5!) that I would put the long implementation comment in my code *before* I start writing code.

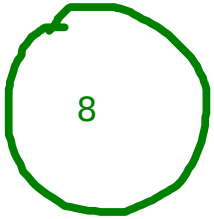
```
# Algorithm for delete:
# 1. If this node is None, return that
# 2. If value is less than node.value, delete it from left child an
#     return this node
# 3. If value is more than node.value, delete it from right child
#     and return this node
# 4. If node with value has fewer than two children,
#     and you know one is None, return the other one
# 5. If node with value has two non-None children,
#     replace value with that of its largest child in the left
#     subtree and delete that child, and return this node
```



delete from right

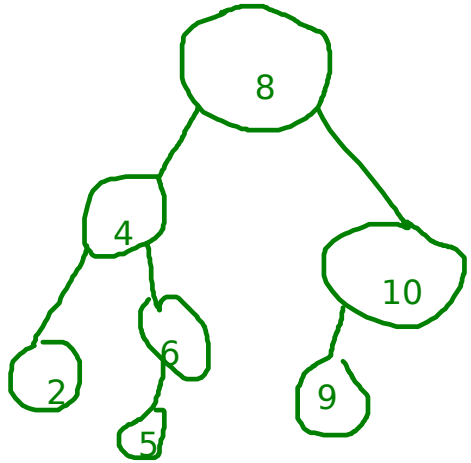


delete from left



? None

return ? (left)



notes