Class 28 - Growing Trees

Schedule

Everyone should either have (1) completed the Blue Belt level and be working on Project 5 (target completion by **Monday, 11 April**), or (2) be making progress to completing the Blue Belt level. If you are stuck, not making progress, or unsure what you should be doing, make sure to check in with me today (in person, slack, or email) or at office hours tomorrow.

Logarithms

$log_b n = x$ means $b^x = n$

Changing bases

Why is it unnecessary to write the base in $\Theta(logN)$?

What is $log_2 1000000000$?

Growing Trees

Recall: A *List* is either (1) None, or (2) a pair whose second part is a *List*.

What is a *Tree*?

See Notes 27 for code for trees (and download link).

```
from binarytree import BinaryTree

class OrderedBinaryTree(BinaryTree):
    """
    A tree where the ordered invariant is preserved:
        if left_node is left of current_node,
            fcompare(left_node.value, current_node.value)
        must be True
    """
    ...
    def find_match(self, value):
        """
        Returns a node of self where node.value == value,
        or None if no such node exists.
        """
        if self._value == value:
            return self
        if self._fcompare(value, self._value):
            if self.left_child():
                return self.left_child().find_match(value)
            else:
                return None
        else:
            if self.right_child():
                return self.right_child().find_match(value)
            else:
                return None
```

What is the complexity of searching for a node in an OrderedBinaryTree?