- Test rooms posted on course web site
- test during usual time for lecture

CSC148 winter 2016

mutating BSTs

week 9

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/148/W14/
416-978-5899

March 18, 2016

Computer Science
UNIVERSITY OF TORONTO

Outline

test coverage

we provide an API with common Python functions and classes you will need

- **LinkedListNode** and **LinkedList** ← process iteratively
- recursion on nested Python **list** [.. [[...]..] ..]
- recursion on class **Tree** → arbitrary # of children, no empty
- recursion on class **BinaryTree** – two children max, None ≡ empty
- definitions for trees and binary trees nodes, root, leaf, path, height, etc.

recall ~~BTNode~~ BinaryTree

```
class BinaryTree

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

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

insert must obey BST condition

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

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

    Assume node is the root of a Binary Search Tree.

    @param BinaryTree node: root of a binary search tree.
    @param object data: data to insert into BST, if necessary.

    >>> b = BinaryTree(5)
    >>> b1 = insert(b, 3)
    >>> print(b1)
    5
        3
    <BLANKLINE>
    """
```

] turned sideways

deletion of data from BST rooted at node?

- what return value? may change, depending on which node is deleted.
- what to do if node is None?
  nothing to delete - return None
- what if data to delete is less than that at node?
  - call delete on left subtree, return un-modified root
- what if it's more?
  - " ______ " right " ______ "
- what if the data equals this node's data and...
  - this node has no left child — return right child whatever it is
  - ... no right child?
    ↳ return left child, whatever it is
  - both children? are non-None?
    replace this node's data with max data in left subtree

algorithm...

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

code here!

code here!

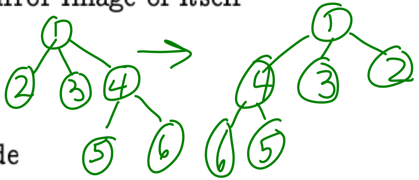
code here!

code here!

a helper, find_max is useful here.

more mutation — reflection!

change a Tree so that it is a mirror image of itself



this changes every internal node

- children gets re-ordered.

order of changes is critical

t.children[i] = t.children[n-1-i]

look out !