Week 9: Tree Mutation

Arnamoy Bhattacharyya

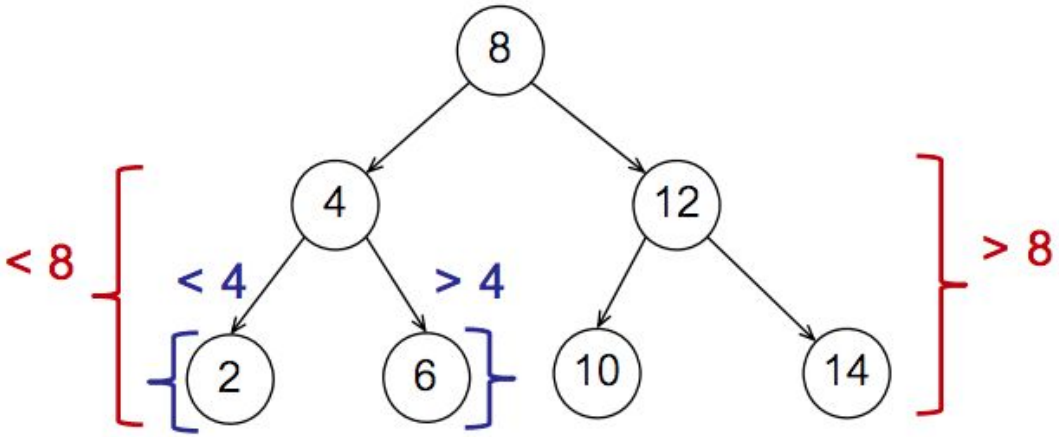
Agenda

1. A bunch of exercises
2. Insert a node in BST

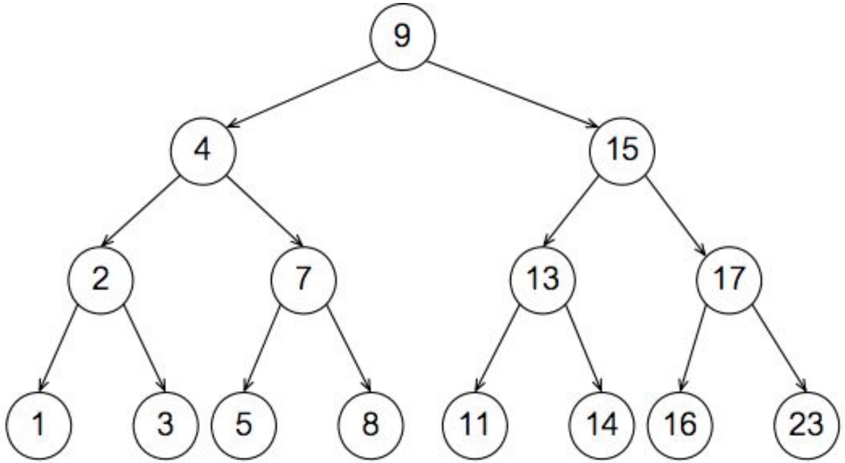
Refresher BST:

Definition

- Add ordering conditions to a binary tree:
  - data are *comparable*
  - data in **left** subtree are **less** than node.data
  - data in **right** subtree are **more** than node.data



Find a value in a BST



How many nodes do we visit (say, in preorder) to find out the following:

- Find value 5, if present...
- Find value 13, if present...
- Find value 12, if present...

Why binary search trees?

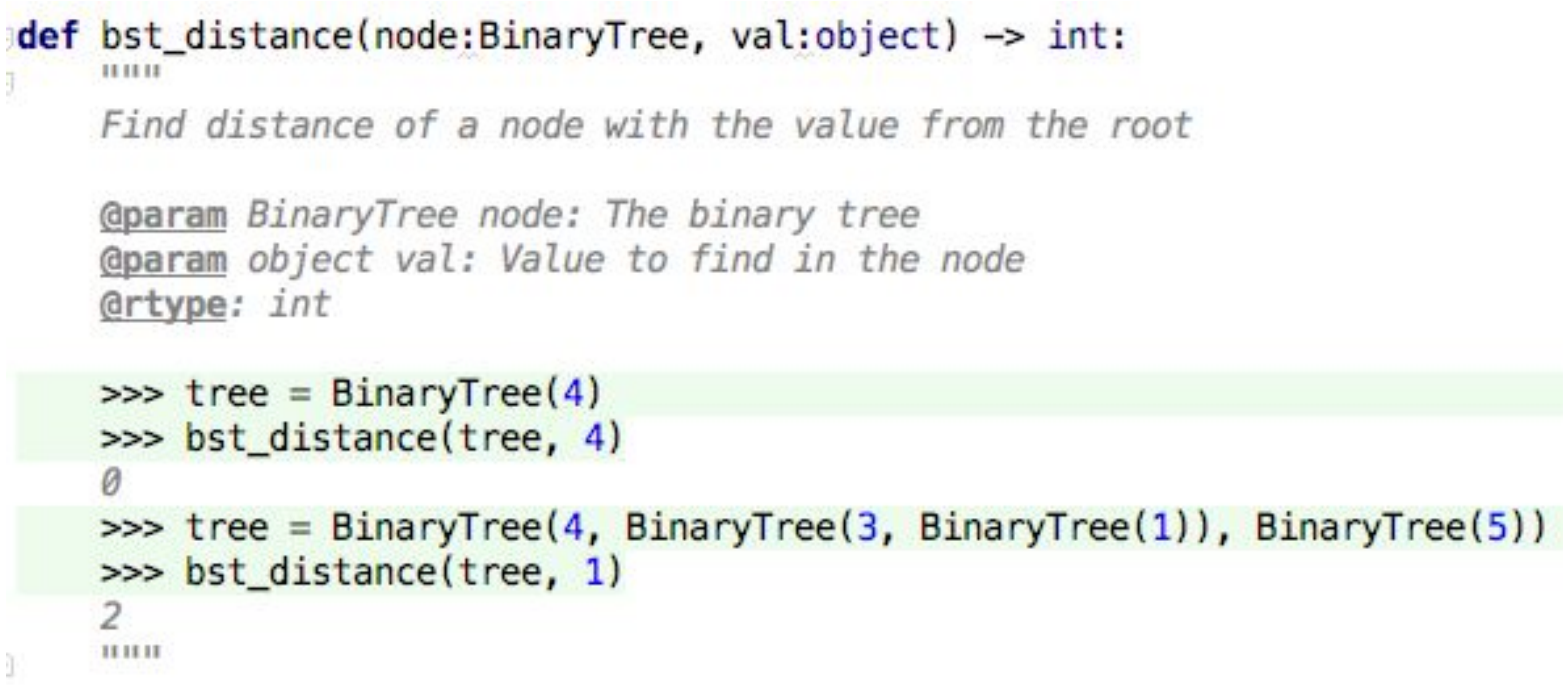
Searches that are directed along a single path are efficient:

- a BST with 1 node has height 1
- a BST with 3 nodes may have height 2
- a BST with 7 nodes may have height 3
- a BST with 15 nodes may have height 4
- a BST with n nodes may have height `log`$_2$ `n`
  - **`1,000,000 nodes => height < 20!`**

If the BST is “balanced”, then we can check whether an element is present in about `lg n` node accesses

`lg` is eqv to `log`$_2$

Warm up Exercise:



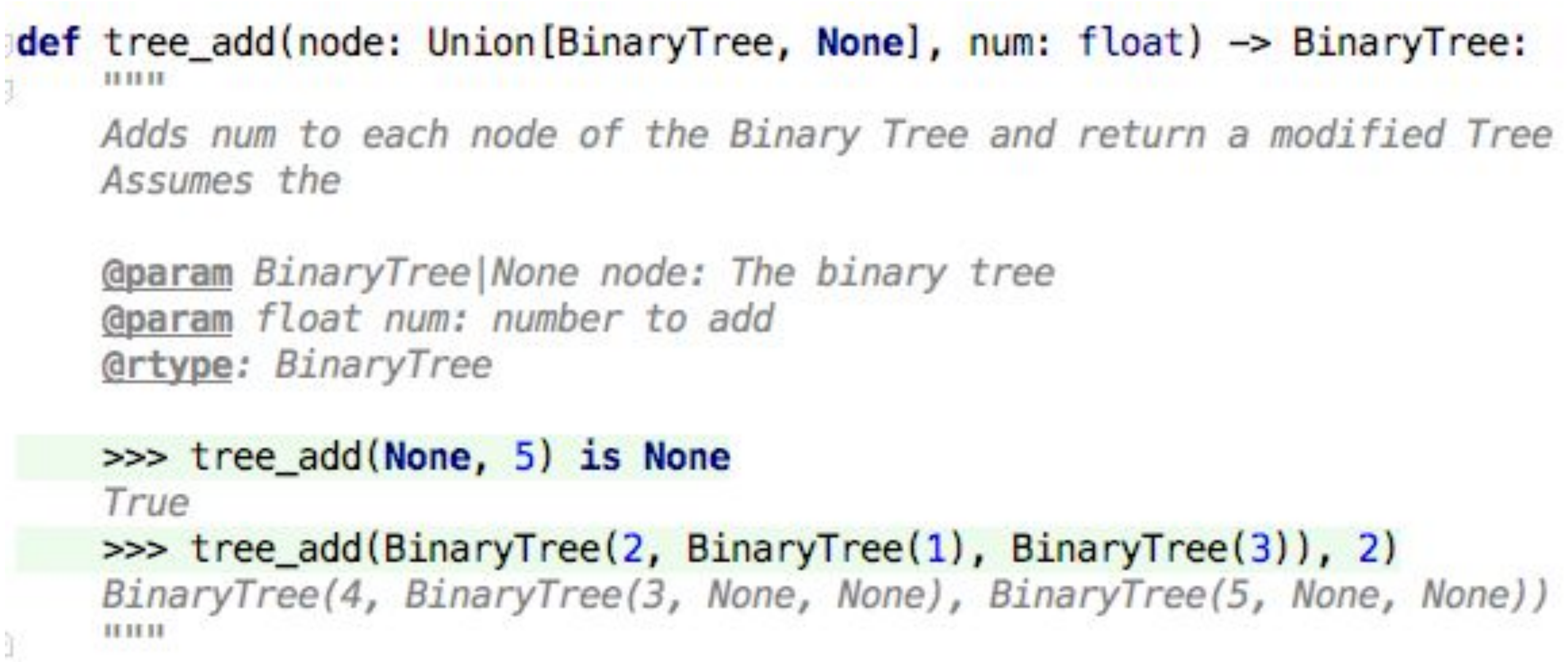
```
def bst_distance(node:BinaryTree, val:object) -> int:
    """
    Find distance of a node with the value from the root

    @param BinaryTree node: The binary tree
    @param object val: Value to find in the node
    @rtype: int

    >>> tree = BinaryTree(4)
    >>> bst_distance(tree, 4)
    0
    >>> tree = BinaryTree(4, BinaryTree(3, BinaryTree(1)), BinaryTree(5))
    >>> bst_distance(tree, 1)
    2
    """
```

Exercise 2: Code Testing

Problem:

Check whether a given Binary Tree is a BST

Exercise 3: Mutation



```
def tree_add(node: Union[BinaryTree, None], num: float) -> BinaryTree:
    """
    Adds num to each node of the Binary Tree and return a modified Tree
    Assumes the

    @param BinaryTree|None node: The binary tree
    @param float num: number to add
    @rtype: BinaryTree

    >>> tree_add(None, 5) is None
    True
    >>> tree_add(BinaryTree(2, BinaryTree(1), BinaryTree(3)), 2)
    BinaryTree(4, BinaryTree(3, None, None), BinaryTree(5, None, None))
    """
```

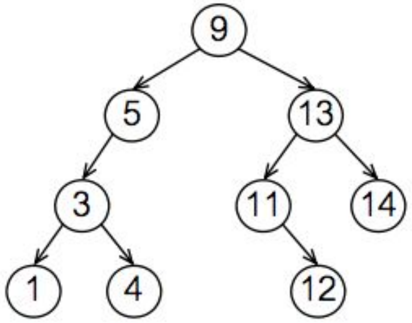
Mutation: BST Insert

Insert must ensure BST condition holds:

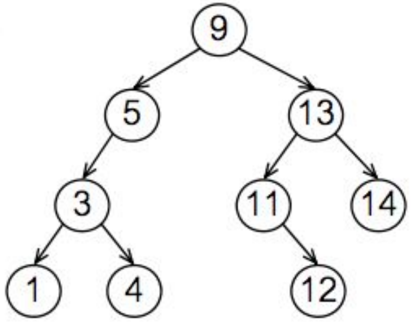
- Left subtree of node < node.data
- Right subtree of node > node.data

How would insert work?

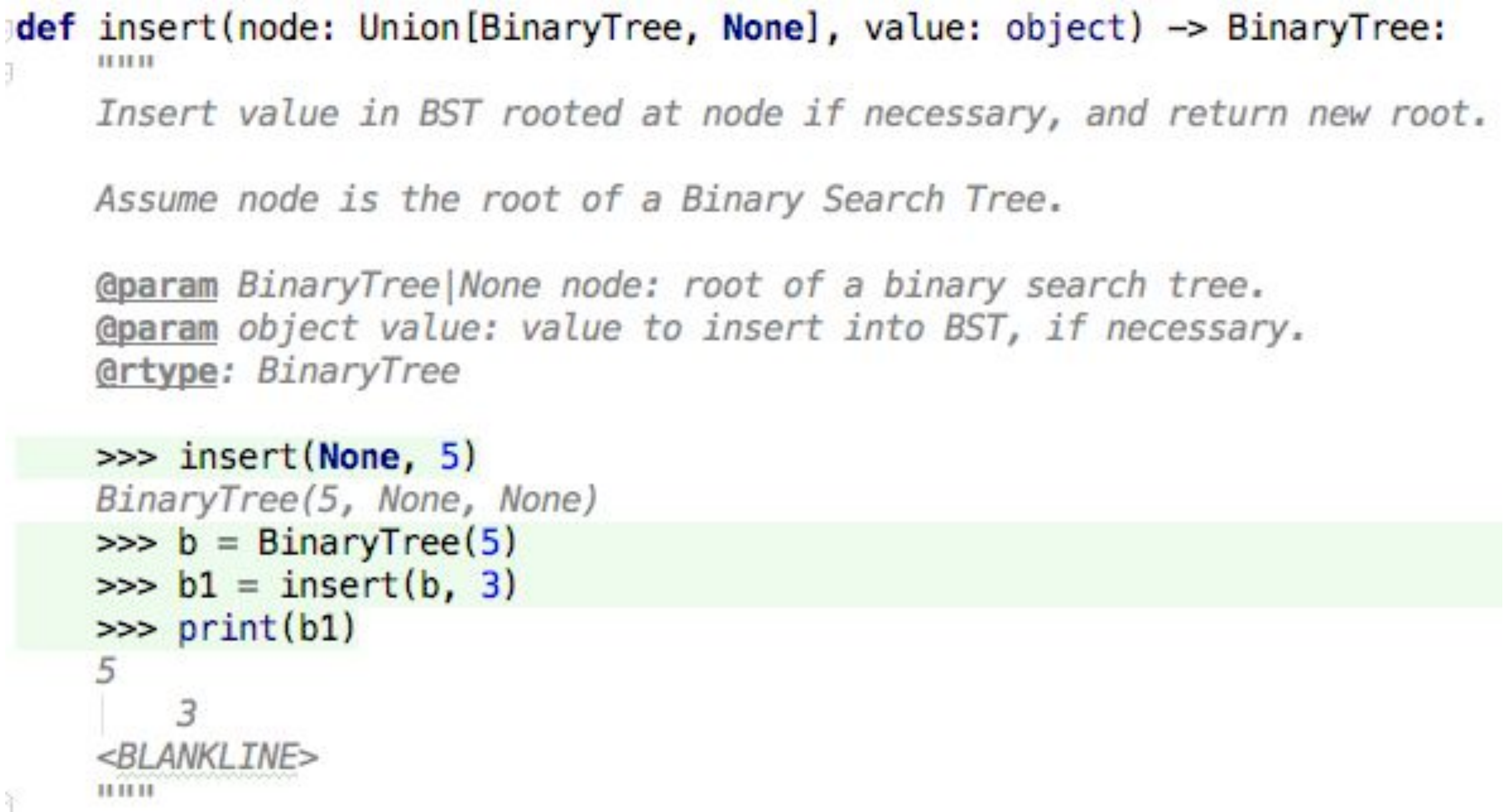
insert value 7?



insert value 10?



Insert implementation



```
def insert(node: Union[BinaryTree, None], value: object) -> BinaryTree:
    """
    Insert value in BST rooted at node if necessary, and return new root.

    Assume node is the root of a Binary Search Tree.

    @param BinaryTree|None node: root of a binary search tree.
    @param object value: value to insert into BST, if necessary.
    @rtype: BinaryTree

    >>> insert(None, 5)
    BinaryTree(5, None, None)
    >>> b = BinaryTree(5)
    >>> b1 = insert(b, 3)
    >>> print(b1)
    5
        3
    <BLANKLINE>
    """
```