#17: BST Remove

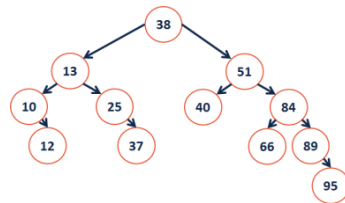October 4, 2019 · *G Carl Evans*

BST.cpp

```cpp
template <class K, class V>
void BST::_insert(TreeNode *& root, K & key, V & value) {
  TreeNode *t = _find(root, key);
  t = new TreeNode(key, value);
}
```

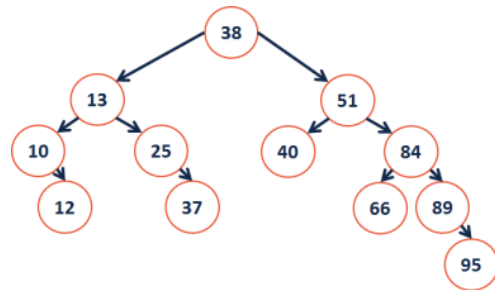**Running time?** _____   **Bound by?** _____

What happens when we run the bugged code above?

How do we fix the code?

Removing an element from a BST:

`_remove(40)`

`_remove(25)`

`_remove(10)`

`_remove(13)`

One-child Remove	Two-child remove

BinaryTree.cpp

```cpp
template <class K, class V>
void BST::_remove(TreeNode *& root, const K & key) {




}
```

BST Analysis:
Every operation we have studied on a BST depends on:

...what is this in terms of the amount of data, **n**?

BST – Simple Proofs

Q: Given a height **h**, what is the maximum number of nodes (**n**) in a valid BST of height **h**? Provide an outline of a proof.

Q: Given a height **h**, what is the minimum number of nodes (**n**) in a valid BST of height **h**? Provide an outline of a proof.

Final BST Analysis

For every height-based algorithm on a BST:

Lower Bound:

Upper Bound:

Why use a BST over a linked list?

operation	BST Avg. Case	BST Worst Case	Sorted Array	Sorted List
find				
insert				
delete				
traverse				

Q: How does our data determine the height?

$$1\ 3\ 2\ 4\ 5\ 7\ 6 \qquad \text{vs.} \qquad 4\ 2\ 3\ 6\ 7\ 1\ 5$$
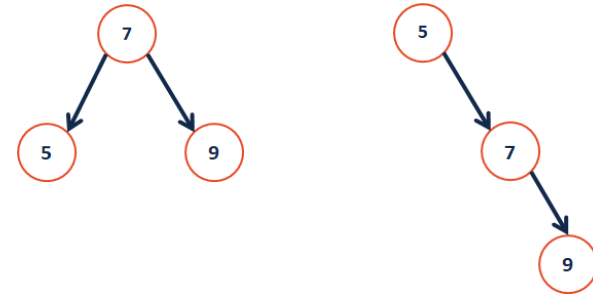
Q: How many different ways are there to insert data into a BST?

Q: What is the average height of every arrangement?

...what is the intuition here?

Height Balance on BST

What tree makes you happier?

We define the **height balance** (b) of a BST to be:

We define a BST tree T to be **height balanced** if: