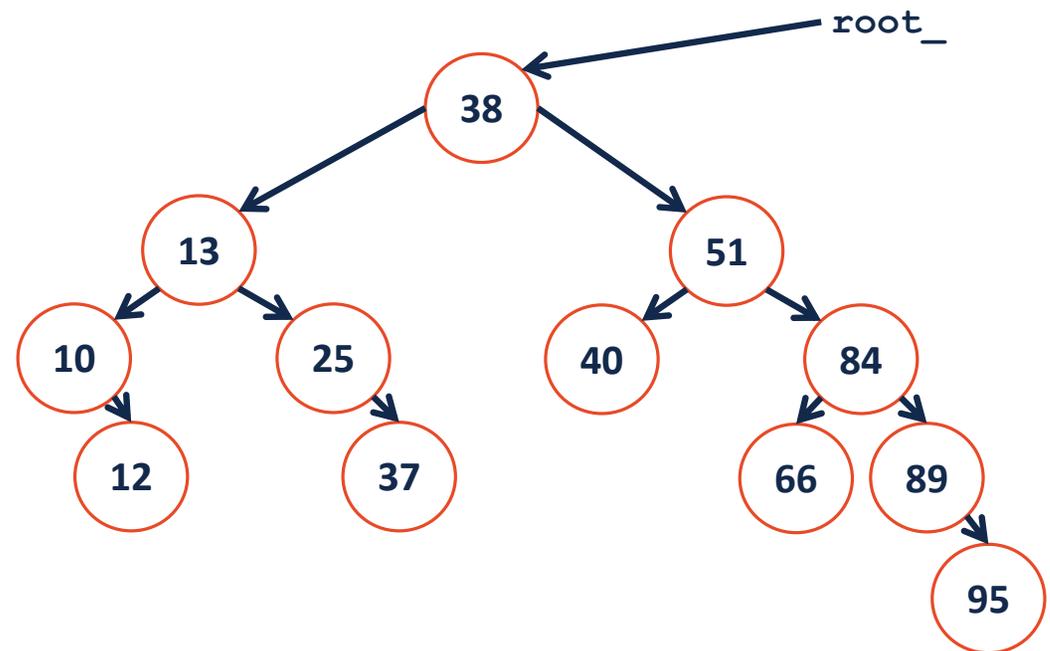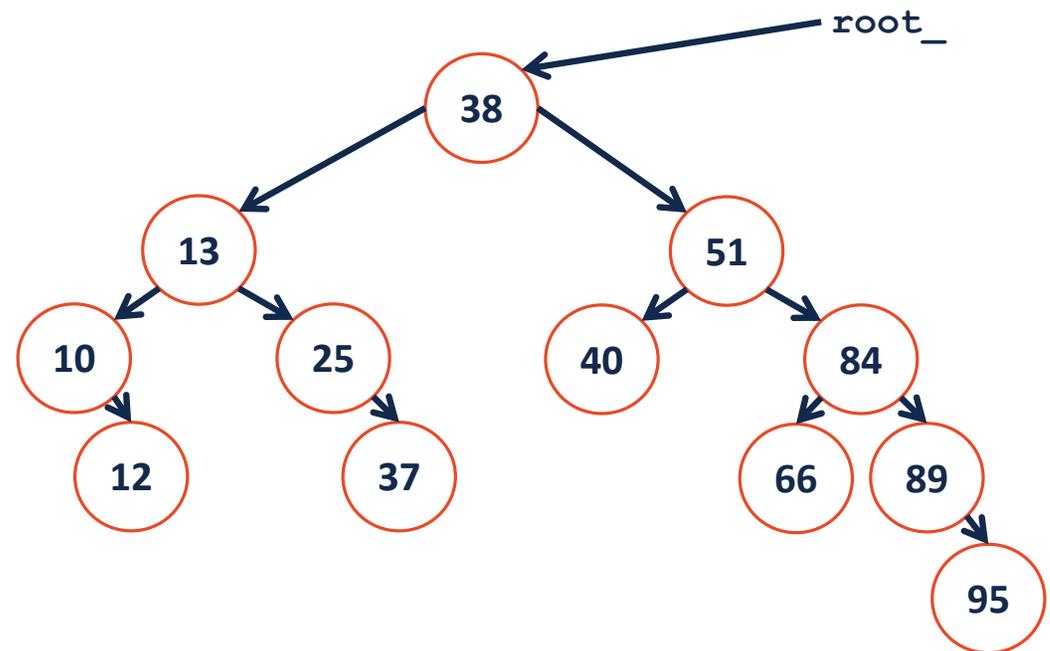# CS 225

**Data Structures**

*Feb. 22 – BST Remove*
*Wade Fagen-Ulmschneider, Craig Zilles*

```cpp
template<typename K, typename V>

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

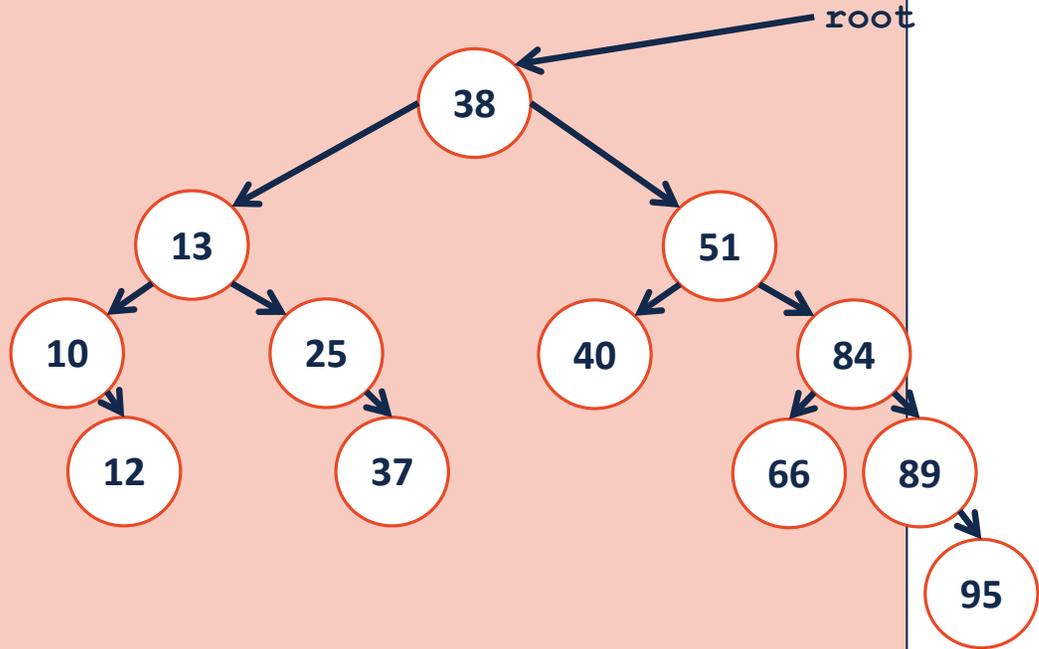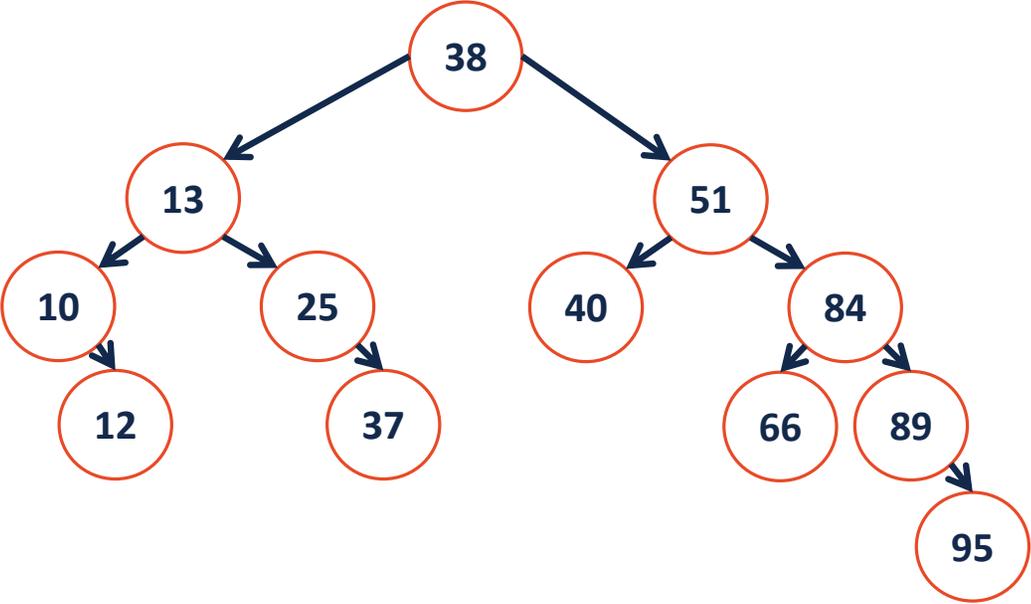```cpp
template<typename K, typename V>

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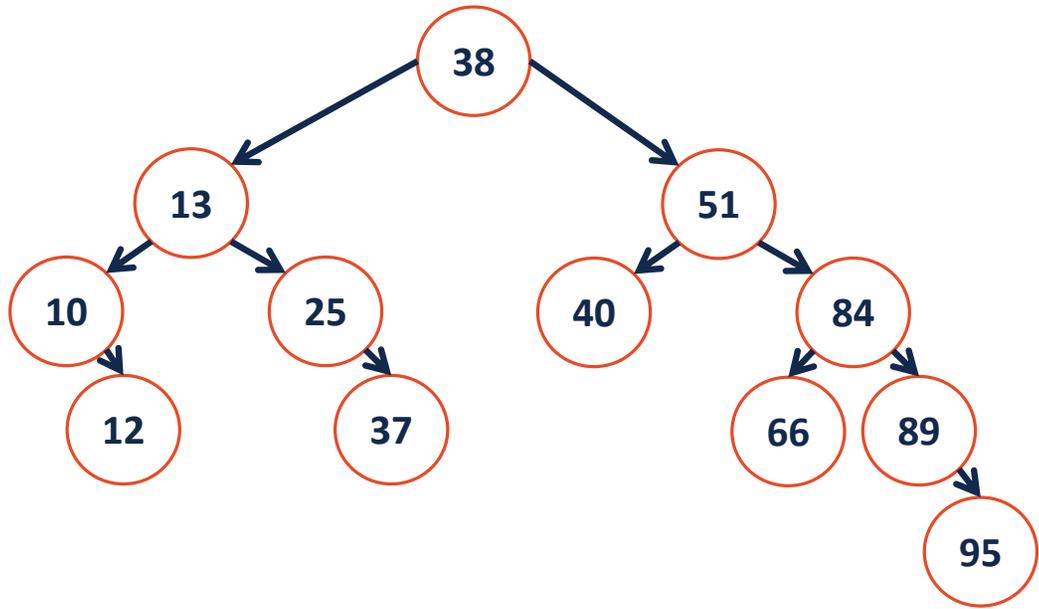
```
template<typename K, typename V>

_____ _remove(TreeNode *& root, const K & key) {



}
```

```
remove(40);
```
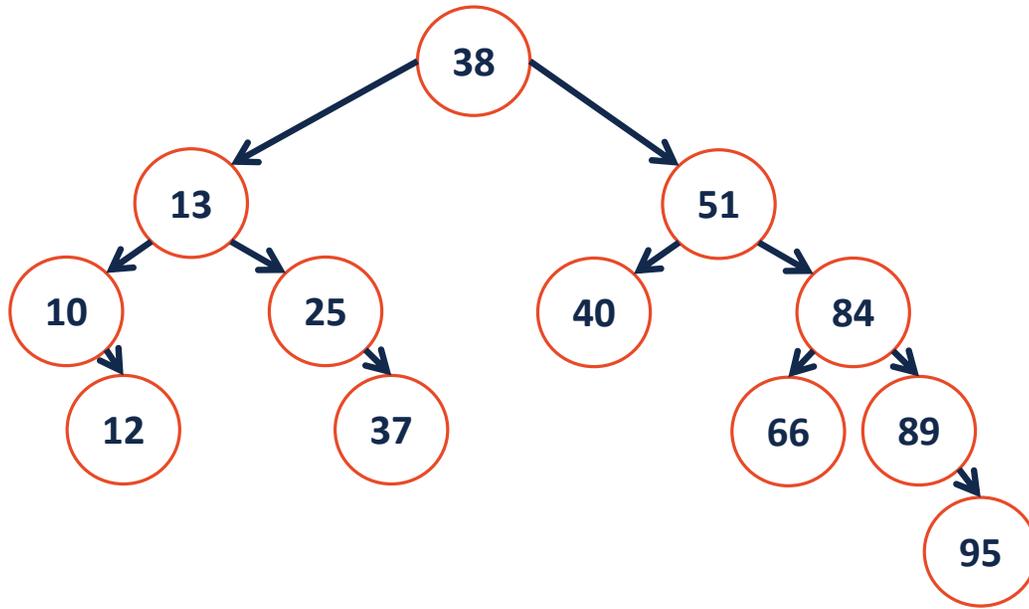
remove(25);

```
remove(10);
```
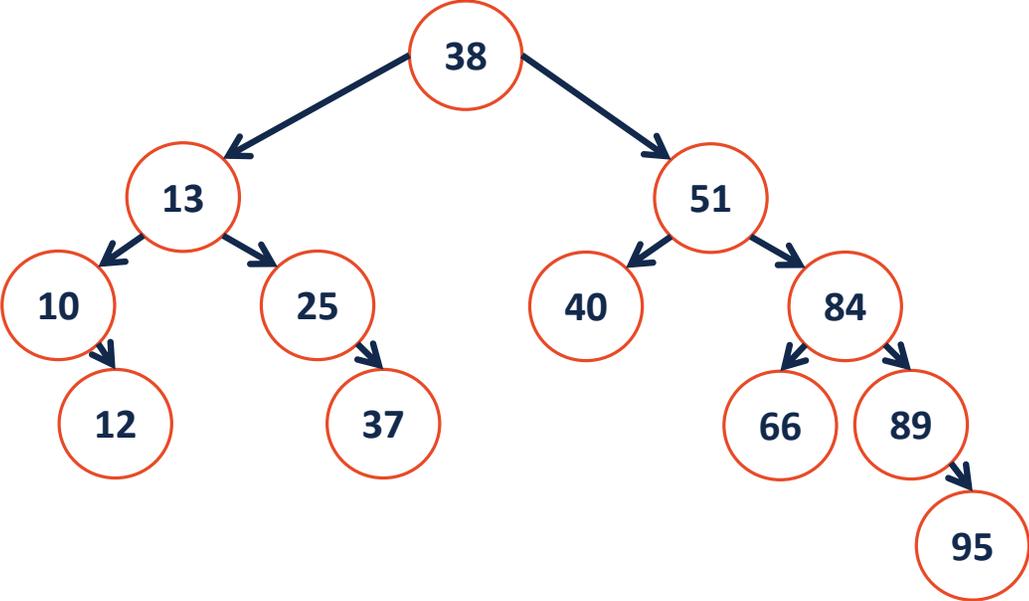
remove(13);

# BST Analysis – Running Time

| Operation | BST Worst Case |
|---|---|
| find | |
| insert | |
| delete | |
| traverse | |

# BST Analysis

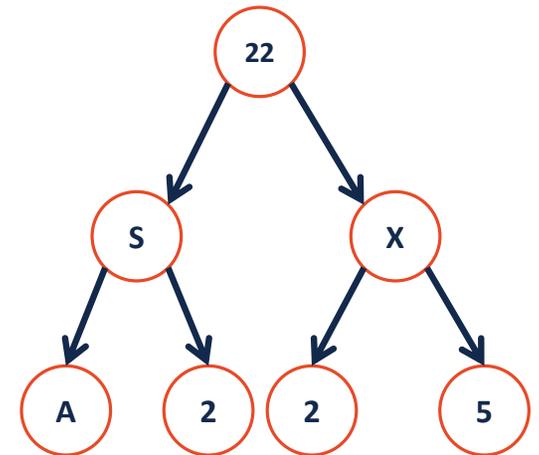Every operation that we have studied on a BST depends on the height of the tree: **O(h)**.

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

We need a relationship between **h** and **n**:
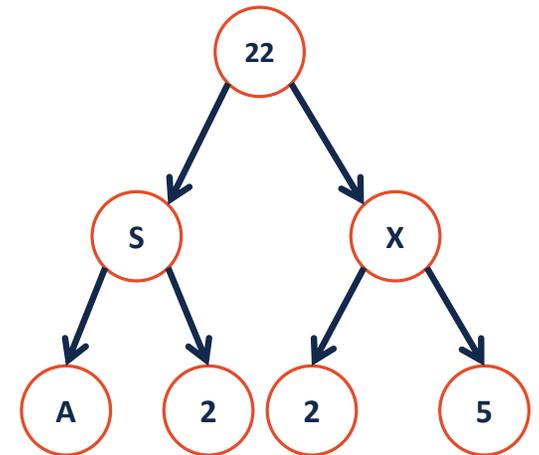
**f(h) ≤ n ≤ g(h)**

# BST Analysis

**Q:** What is the maximum number of nodes in a tree of height **h**?

# BST Analysis

**Q:** What is the minimum number of nodes in a tree of height **h**?

What is the maximum height for a tree of **n** nodes?

# BST Analysis

Therefore, for all BST:

**Lower bound:**

**Upper bound:**

# BST Analysis

The height of a BST depends on the order in which the data is inserted into it.

**ex: 1 3 2 4 5 7 6          vs.          4 2 3 6 7 1 5**

**Q:** How many different ways are there to insert keys into a BST?

**Q:** What is the average height of all the arrangements?

# BST Analysis

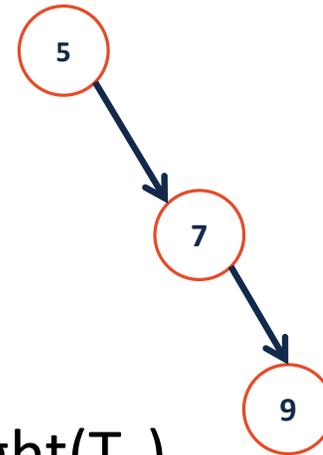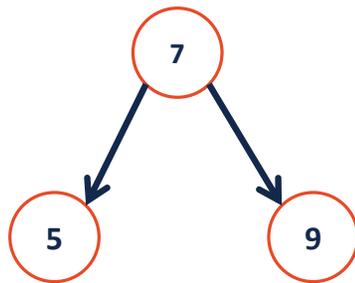**Q:** How many different ways are there to insert keys into a BST?

**Q:** What is the average height of all the arrangements?

# BST Analysis – Running Time

| Operation | BST Average case | BST Worst case | Sorted array | Sorted List |
|-----------|------------------|----------------|--------------|-------------|
| find | | | | |
| insert | | | | |
| delete | | | | |
| traverse | | | | |

# Height-Balanced Tree

What tree makes you happier?



Height balance:   $b = height(T_L) - height(T_R)$

A tree is height balanced if: