#21: AVL Applications
October 16, 2020 · *G Carl Evans*

AVL – Proof of Runtime
Plan of Action:

Goal: Find a function that defines the lower bound on **n** given **h**.

Given the goal, we begin by defining a function that describes the smallest number of nodes in an AVL of height **h**:

Theorem:
An AVL tree of height **h** has at least ________________________.

I. Consider an AVL tree and let **h** denote its height.

II. Case: __________________

III. Case: ___________________

Inductive hypothesis (IH):

Proving our IH:

V. Using a proof by induction, we have shown that:

...and by inverting our finding:

Summary of Balanced BSTs:

Advantages	Disadvantages

AVL Trees	Red-Black Trees
Balanced BST Max height: 1.44 * lg(n) *Q: Why is our proof 2*lg(n)?* Rotations:	Balanced BST *Functionally equivalent to AVL trees; all key operations runs in O(h) time.* Max height: 2 * lg(n) Rotations:

- find: - insert: - remove:	- find: - insert: - remove:

*In CS 225, we learned **AVL trees** because they're intuitive and I'm certain we could have derived them ourselves given enough time. A red-black tree is simply another form of a balanced BST that is also commonly used.*

Summary of Balanced BSTs:
(Includes both AVL and Red-Black Trees)

Advantages	Disadvantages

Using a Red-Black Tree in C++
C++ provides us a balanced BST as part of the standard library:

```
std::map<K, V> map;
```

The map implements a dictionary ADT. Primary means of access is through the overloaded `operator[]`:

```
V & std::map<K, V>::operator[]( const K & )
```

This function can be used for both insert and find!

Removing an element:

```
void std::map<K, V>::erase( const K & );
```

Range-based searching:

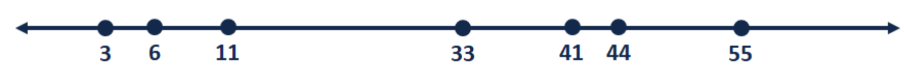
```
iterator std::map<K, V>::lower_bound( const K & );
iterator std::map<K, V>::upper_bound( const K & );
```

Running Time of Every Data Structure So Far:

	Unsorted Array	Sorted Array	Unsorted List	Sorted List
Find				
Insert				
Remove				
Traverse				

	Binary Tree	BST	AVL
Find			
Insert			
Remove			
Traverse			

Range-based Searches:
Q: Consider points in 1D: $p = \{p_1, p_2, \ldots, p_n\}$.
...what points fall in [11, 42]?



Range-based Searches:

Running Time: