CS 225

Data Structures

September 28 – AVL Trees

G Carl Evans

mp_traversals AMA

Thursday at 6pm on zoom

https://illinois.zoom.us/j/82051109289?pwd=bzNBbFZnWG5GRU9iZmxTWTYwajBtZz09

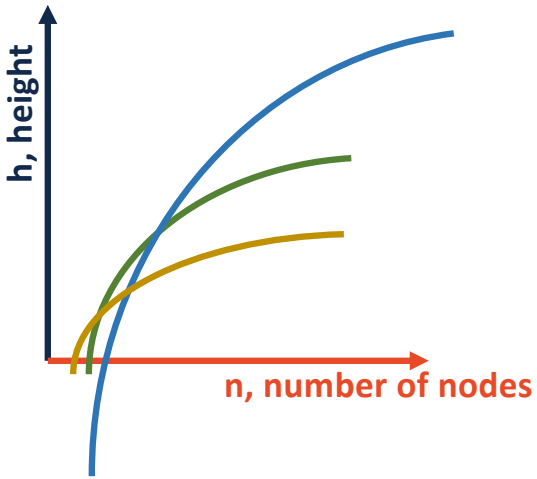
It will also be recorded and added to the lecture stream on mediaspace later.

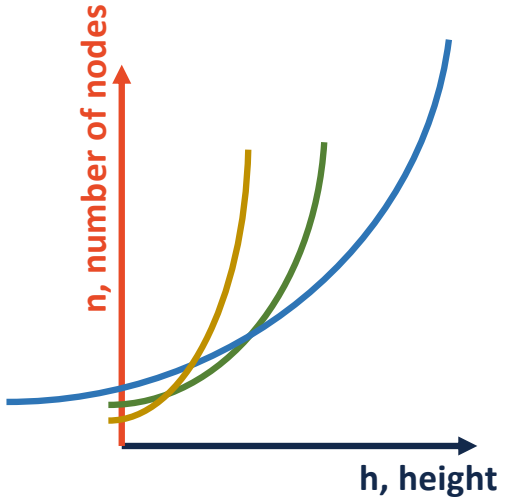
AVL Tree Analysis

We know: insert, remove and find runs in: _____.

We will argue that: h is _____.

AVL Tree Analysis





- The number of nodes in the tree, $\mathbf{f^{-1}(h)}$, will always be <u>greater than</u> $\mathbf{c \times g^{-1}(h)}$ for all values where $\mathbf{n > k}$.

Plan of Action

Since our goal is to find the lower bound on **n** given **h**, we can begin by defining a function given **h** which describes the smallest number of nodes in an AVL tree of height **h**:

Simplify the Recurrence

N(h) = 1 + N(h - 1) + N(h - 2)

State a Theorem

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

Proof:

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

II. Case: _____

An AVL tree of height ____ has at least ____ nodes.

Prove a Theorem

III. Case: _____

An AVL tree of height ____ has at least ____ nodes.

Prove a Theorem

By an Inductive Hypothesis (IH):

We will show that:

An AVL tree of height ____ has at least ____ nodes.

Prove a Theorem

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

…and inverting:

Summary of Balanced BST

Red-Black Trees

- Max height: 2 * lg(n)
- Constant number of rotations on insert, remove, and find

AVL Trees

- Max height: 1.44 * lg(n)
- Rotations:

Summary of Balanced BST

Pros:

- Running Time:

  - Improvement Over:

- Great for specific applications:

Summary of Balanced BST

Cons:

- Running Time:

- In-memory Requirement:

Red-Black Trees in C++

C++ provides us a balanced BST as part of the standard library:

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

Red-Black Trees in C++

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

Red-Black Trees in C++

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

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

Red-Black Trees in C++

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

Every Data Structure So Far

	Unsorted Array	Sorted Array	Unsorted List	Sorted List	Binary Tree	BST	AVL
Find							
Insert							
Remove							
Traverse							