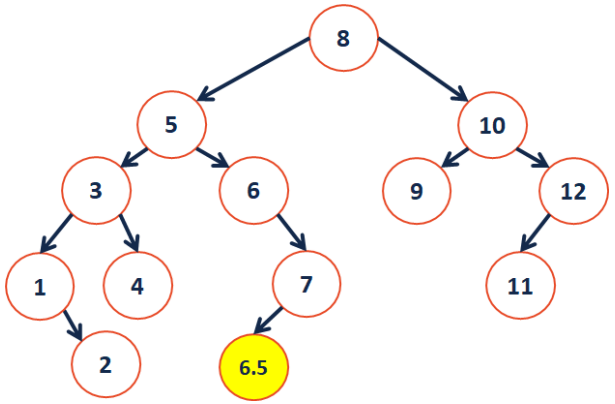
CS 225

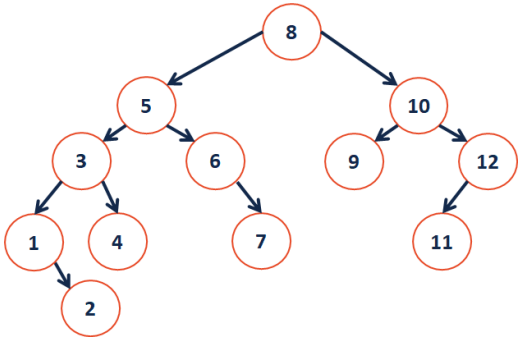
#20: AVL Analysis

March 2, 2018 · *Wade Fagen-Ulmschneider*

AVL Insertion



AVL Removal



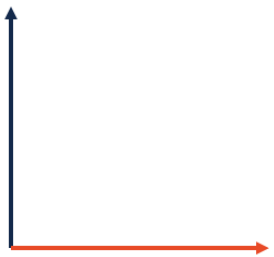
Running Times:

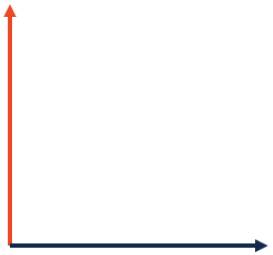
	AVL Tree
find	
insert	
remove	

Motivation:

Big-O is defined as:

Visually:





f(n), g(n) -- The graph above describes functions of the height (**h**) of an AVL tree given the number of nodes (**n**).

$f^{-1}(n)$, $g^{-1}(n)$ – Inverse functions describe the number of nodes in a tree (**n**) given a height (**h**).

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**:

State a 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: _____

IV. Case: _____

By an inductive hypothesis (IH):

We show that:

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

...and by inverting our finding:

Summary of Balanced BSTs:

Advantages	Disadvantages

Iterators + Usefulness
Three weeks ago, you saw that you can use an iterator to loop through data:

```
1  DFS dfs(...);
2  for ( ImageTraversal::Iterator it = dfs.begin();
         it != dfs.end(); ++it ) {
3    std::cout << (*it) << std::endl;
4  }
```

You will use iterators extensively in MP4, creating them in Part 1 and then utilizing them in Part 2. Given the iterator, you can use the for-each syntax available to you in C++:

```
1  DFS dfs(...);
2  for ( const Point & p : dfs ) {
3    std::cout << p << std::endl;
4  }
```

The exact code you might use will have a generic `ImageTraversal`:

```
1  ImageTraversal & traversal = /* ... */;
2  for ( const Point & p : traversal ) {
3    std::cout << p << std::endl;
4  }
```

CS 225 – Things To Be Doing:

1. Theory Exam 2 is ongoing!
2. MP4 extra credit submission ongoing – due Monday, March 5th!
3. lab_huffman is due on Sunday, March 4th
4. Daily POTDs are ongoing!