CS 225

Data Structures

March 1 – AVL Analysis

Wade Fagen-Ulmschneider

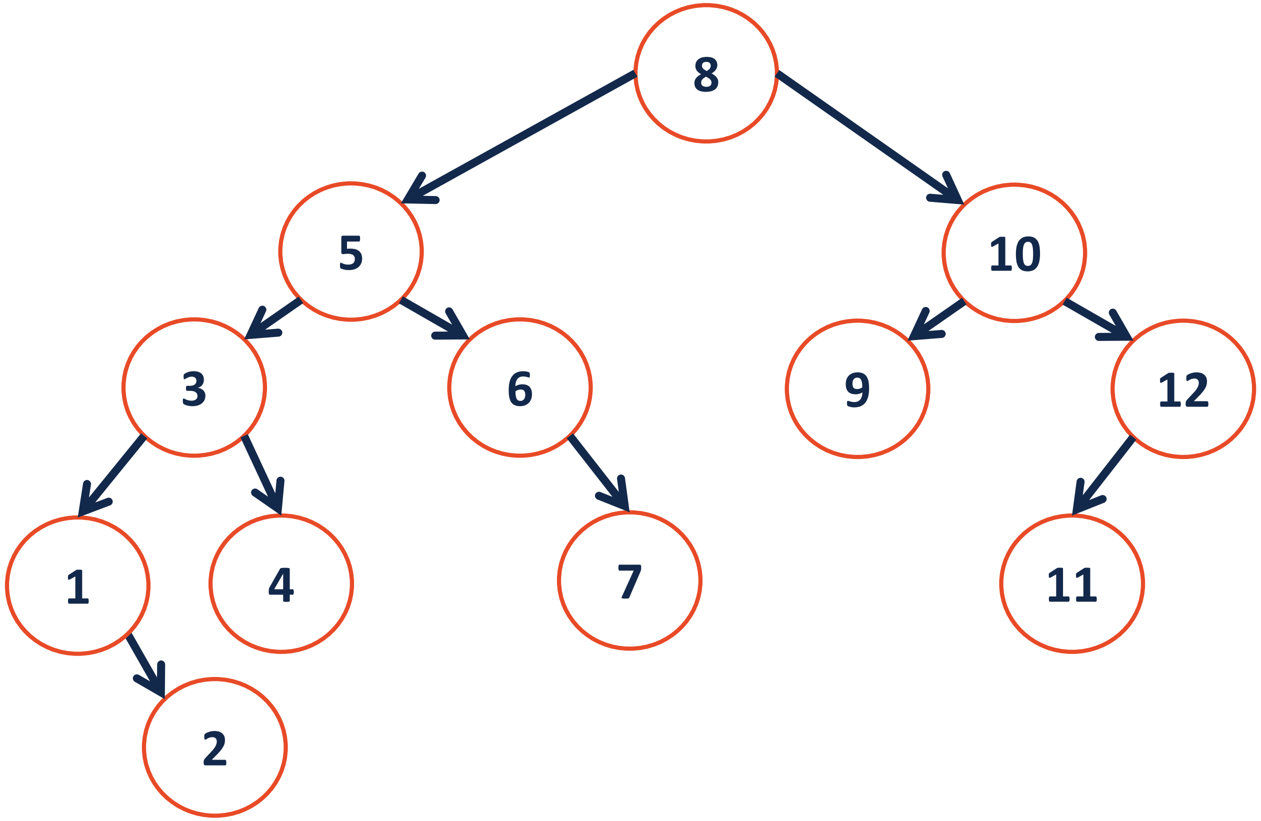
Insertion into an AVL Tree

`_insert(6.5)`

Insert (pseudo code):

1: Insert at proper place
2: Check for imbalance
3: Rotate, if necessary
4: Update height

```
struct TreeNode {
  T key;
  unsigned height;
  TreeNode *left;
  TreeNode *right;
};
```

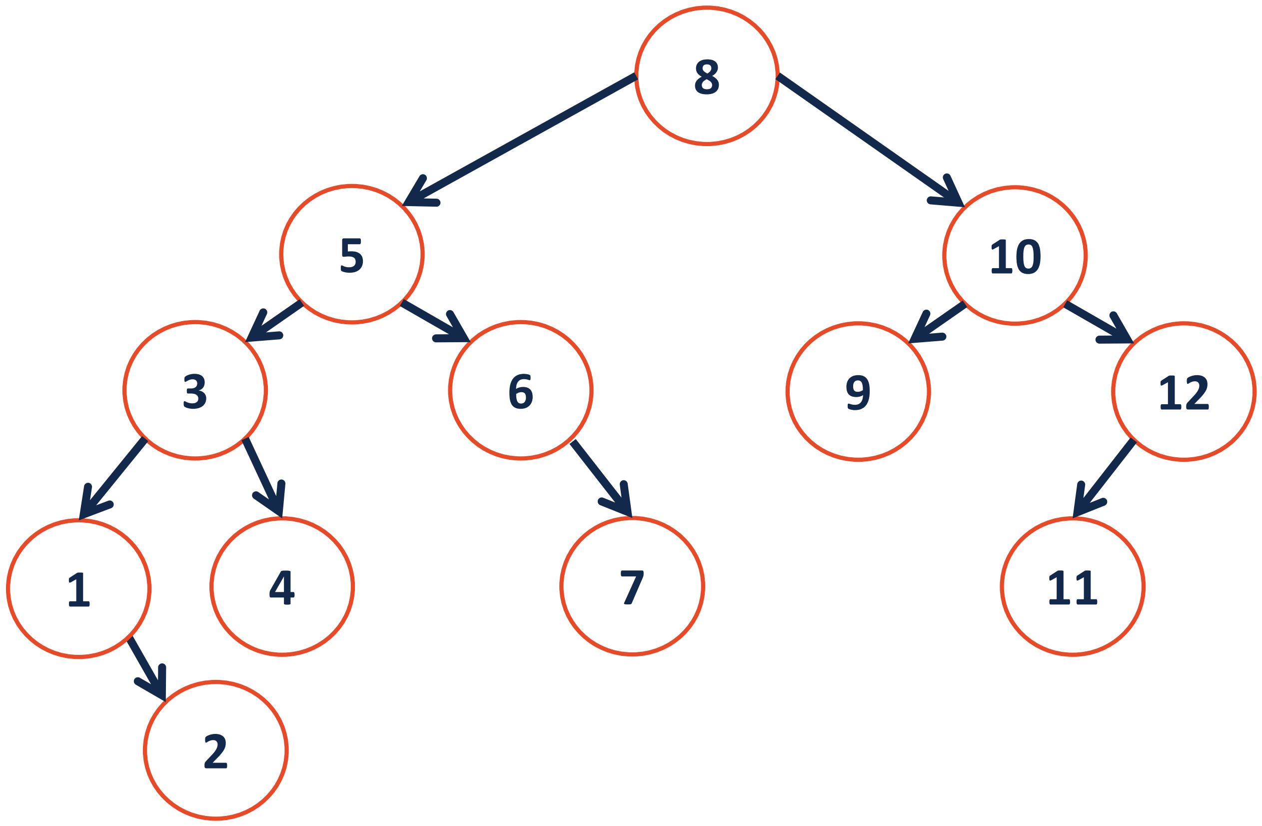


```
template <class T> void AVLTree<T>::_insert(const T & x, treeNode<T> * & t ) {
  if( t == NULL ) {
    t = new TreeNode<T>( x, 0, NULL, NULL);
  }

  else if( x < t->key ) {
    _insert( x, t->left );
    int balance = height(t->right) - height(t->left);
    int leftBalance = height(t->left->right) - height(t->left->left);
    if ( balance == -2 ) {
      if ( leftBalance == -1 ) { rotate_____________( t ); }
      else                     { rotate_____________( t ); }
    }
  }

  else if( x > t->key ) {
    _insert( x, t->right );
    int balance = height(t->right) - height(t->left);
    int rightBalance = height(t->right->right) - height(t->right->left);
    if( balance == 2 ) {
      if( rightBalance == 1 ) { rotate_____________( t ); }
      else                    { rotate_____________( t ); }
    }
  }

  t->height = 1 + max(height(t->left), height(t->right));
}
```


8
5
10
3
6
9
12
1
4
7
11
2

AVL Tree Analysis

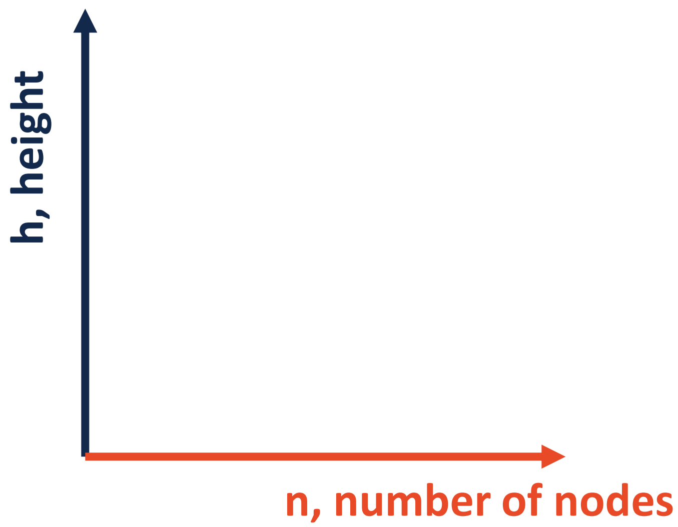
We know: insert, remove and find runs in: ____________.

We will argue that: h is ___________.

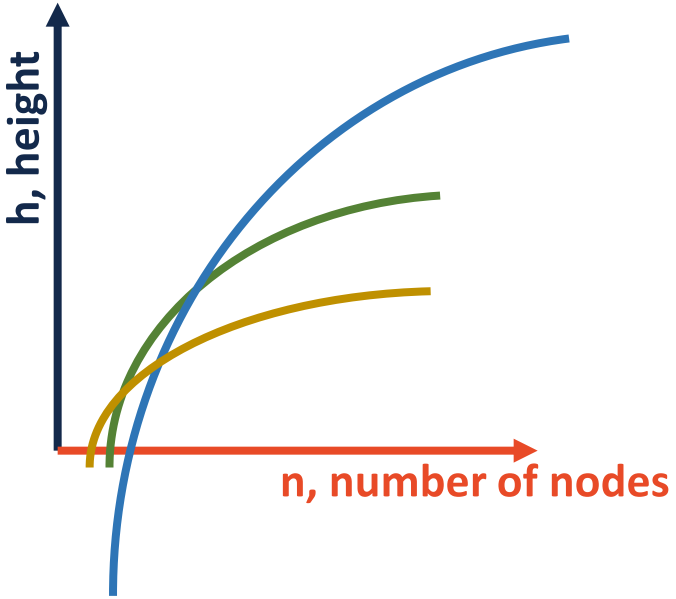
AVL Tree Analysis

Definition of big-O:

…or, with pictures:

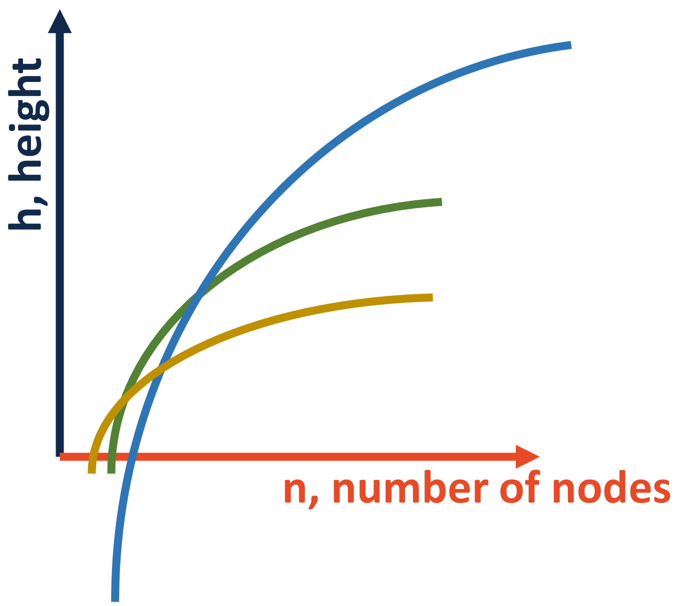


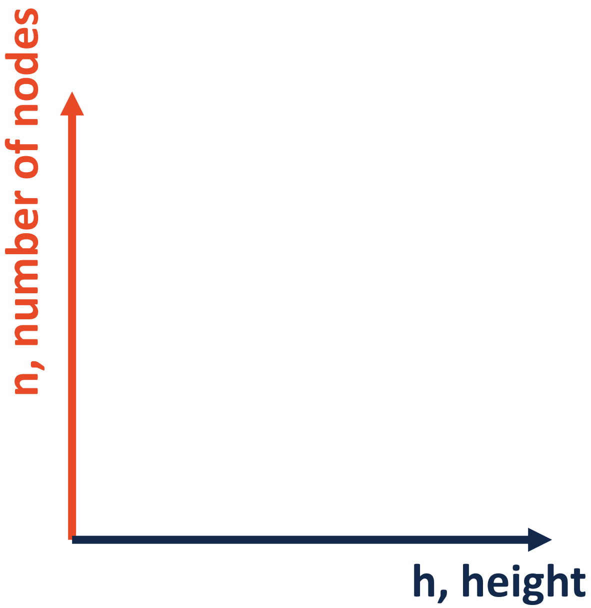
AVL Tree Analysis



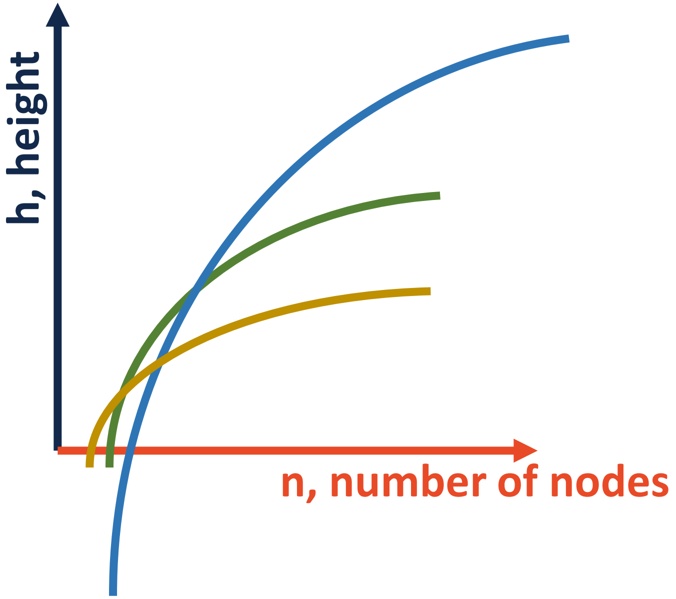
- The height of the tree, **f(n)**, will always be <u>less than</u> **c × g(n)** for all values where **n > k**.

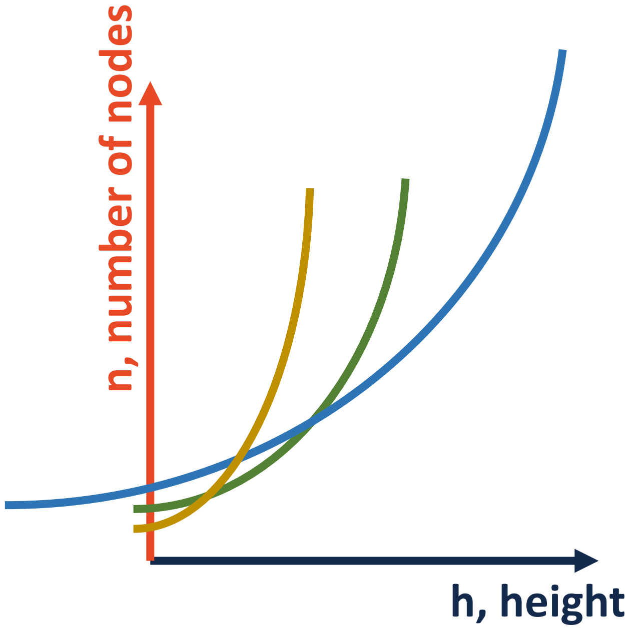
AVL Tree Analysis





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

$\mathbf{N(h)} = 1 + N(h - 1) + N(h - 2)$

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

IV. Case: ________________

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: