CS 225

Data Structures

October 14 – AVL Analysis

G Carl Evans
Insertion into an AVL Tree

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;
};
```

`_insert(6.5)`
template <typename K, typename V>
void AVL<K, D>::_insert(const K & key, const V & data, TreeNode * & cur) {
    if (cur == NULL) { cur = new TreeNode(key, data); }
    else if (key < cur->key) { _insert(key, data, cur->left); }
    else if (key > cur->key) { _insert(key, data, cur->right); }
    _ensureBalance(cur);
}
```cpp
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));
}
```
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 less than $c \times g(n)$ for all values where $n > k$. 
AVL Tree Analysis

- The number of nodes in the tree, $f^{-1}(h)$, will always be greater than $c \times g^{-1}(h)$ for all values where $n > k$. 
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.
An AVL tree of height \( h \) has at least \( 2^h - 1 \) 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: $2 \times \lg(n)$
- Constant number of rotations on insert, remove, and find

AVL Trees
- Max height: $1.44 \times \lg(n)$
- Rotations:
Summary of Balanced BST

Pros:
- Running Time:
  - Improvement Over:
- Great for specific applications:
Summary of Balanced BST

**Cons:**
- Running Time:
  - In-memory Requirement: