AVL Tree Analysis

Definition of big-O:

...or, with pictures:
An upper bound on the height $h$ for a tree of $n$ nodes ...is the same as...

A lower bound on the number of nodes $n$ in a tree of height $h$
Plan of Action

Begin by defining a function that defines the least number of nodes in an AVL tree of height $h$.

$N(h)$: The least 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: ______________

anavl tree of height ____ has at least ____ nodes.
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:
Iterators

Iterators give client code access to traverse the data!

Operators:
- `operator++`
- `operator==`
- `operator!=`
- `operator=`
- `operator*`

Types of iterators:
- Forward
- Backward
- Bidirectional
Iterators encapsulate access to our data:
const PNG & png;
const PNG & png;

Point start(0,3);
const PNG & png;

Point start(0,3);

ImageTraversal *traversal = NULL;
const PNG & png;

Point start(0, 3);

ImageTraversal traversal = ;

ImageTraversal::Iterator

DFS
RFS
CFS
BFS
Iterators

Why do we care?
Iterators

Why do we care?

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

Why do we care?

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

```cpp
DFS dfs(...);
for ( const Point & p : dfs ) {
    std::cout << p << std::endl;
}
```
Iterators

Why do we care?

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

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

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

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

```cpp
std::list<Sphere> sphereList;
...
for (const Sphere & s : sphereList) {
    ...
}
```

```cpp
std::vector<Sphere> sphereList;
...
for (const Sphere & s : sphereList) {
    ...
}
```

```cpp
std::map<std::string, Sphere> sphereMap;
...
for (const std::pair<std::string, Sphere> & kv : sphereMap) {
    ...
}
```
Exam 6 (Programming, Lists/Trees) is ongoing!
More Info: https://courses.engr.illinois.edu/cs225/fa2017/exams/

MP4: Available now!
Due: Monday, Oct. 23 at 11:59pm

Labs: lab_avl
Implement an AVL tree in lab!

POTD
Every Monday-Friday – Worth +1 Extra Credit /problem (up to +40 total)