Test #1 on Friday in class. If you need special considerations (extra time, quiet space, different time, etc.) talk to me today, either via email or in office hours.

Test topics:

- advanced Java:
  - abstract classes and interfaces, inheritance, exceptions, javadoc, jUnit
- algorithms analysis:
  - definition of big-Oh, big-Omega, little-Oh, little-omega, and big Theta
  - determine the running time of various algorithms (including log n ones)
- linear and binary search, selection, insertion and merge sort
- data structures: operations, implementations, running times.
  - arraylists, stacks, queues, linked lists, doubly linked lists, binary trees
- iterators
We’ve seen some recursively defined data structures: LL, BT’s, BST’s, AVL trees.

Suppose we want to prove some property about all instances of a structure.

Ex: suppose I want to prove some boolean property $P(T)$ is true for all AVL trees $T$, such as: $(\text{nodes}(T) \geq f_{h(T)}+1)$, where $f_i$ is the $i$’th Fibonacci number, where the Fibonacci numbers are $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

Think of it this way: suppose we’ve defined a boolean method $P(T)$:

$$P(\text{AVL } T )$$
return $(\text{nodes}(T) \geq f_{h(T)}+1)$

And suppose I suspect that this method $P(T)$ returns true no matter what tree I give it. How could I convince you of this?

I could start by showing it’s true for trees of height $= -1$ (empty trees).

Then I could show it’s true for bigger and bigger trees...

But to really, truly convince you, I’d have to try all of the trees, of which there are an infinite number. So, I don’t have time for that.

We’ll use the recursive definition of our structures to help us prove these things.
Structural Induction

Structural induction is a proof technique used to prove boolean properties. That is, it’s used to prove that a given boolean method always returns true. To use structural induction, you need

- a recursively defined structure S with both
  - a base case, and
  - a recursive case
- a boolean property / method P(S) defined on any instance of structure S

To prove that P(S) returns / is true for all S that satisfy your definition, just

- prove that P(S) is true when S is one of the base cases, and
- prove that P(S) is true in the recursive case IF it’s true for the substructures

Ex: An AVL tree T is either

- an empty tree
- a root node r with left and right (sub) AVL trees whose heights differ by \( \leq 1 \)

For a AVL T, let P(T) be the boolean property that \( \text{nodes}(T) \geq f_{h(T)} + 1 \)

To show that P(T) is always true we just need to show it is true for both cases.
An AVL tree $T$ is either
- an empty tree
- a root node $r$ with left and right (sub) AVL trees whose heights differ by $\leq 1$

For a AVL $T$, let $P(T)$ be the boolean property that $\text{nodes}(T) \geq f_{h(T)+1}$

We will show that $P(T)$ is true for all AVL’s $T$ “by structural induction” on $T$.

**base case:** $T$ is an empty tree

We want to show (WTS) that $P(T)$ is true when $T$ is an empty tree, i.e. that $\text{nodes}(T) \geq f_{h(T)+1}$ holds when $T$ is an empty tree.

Well what do we know about empty trees? $\text{nodes}(T) = 0$ and $h(T) = -1$

Then plugging these in, we get $f_{h(T)+1} = f_{-1+1} = f_0 = 0 \leq 0 = \text{nodes}(T)$.

So the base case is true, because $P(T)$ is true when $T$ is an empty AVL tree.
For a AVL T, let P(T) be the boolean property that \( \text{nodes}(T) \geq f_{h(T)} + 1 \)

recursive case (aka “inductive step”): T is a root node r with left and right (sub) AVL trees L & R, respectively, whose heights differ by \( \leq 1 \)

Our induction hypothesis (IH) assumes that P(L) and P(R) are true, i.e. that

\[
\text{nodes}(L) \geq f_{h(L)} + 1 \quad \text{and} \quad \text{nodes}(R) \geq f_{h(R)} + 1
\]

Given these assumptions, we need to show that \( \text{nodes}(T) \geq f_{h(T)} + 1 \)

To do this, we need to find a connection between

- \( \text{nodes}(T) \) and \( \text{nodes}(L) \), \( \text{nodes}(R) \): \( \text{nodes}(T) = 1 + \text{nodes}(L) + \text{nodes}(R) \)
- \( f_{h(T)} + 1 \) and \( f_{h(L)} + 1, f_{h(R)} + 1 \): \( f_{h(T)} + 1 = f_{h(T)} + f_{h(T)} - 1 \leq 1 + f_{h(L)} + f_{h(R)} + 1 \) because \( h(R), h(L) \) differ by \( \leq 1 \)

Then \( \text{nodes}(T) = 1 + \text{nodes}(L) + \text{nodes}(R) \)

\[
\geq 1 + f_{h(L)} + 1 + f_{h(R)} + 1 \\
\geq 1 + f_{h(T)} + f_{h(T)} - 1 \\
= 1 + f_{h(T)} + 1
\]

(by the IH)

So the recursive case is true, and we are done with our proof!
Structural Induction Example 2

A natural number n is either
• 0, or
• n’+1, where n’ is a natural number

For a nat num n, let P(n) be the boolean property that \( f_{k+2} \geq 2^{k/2} \) for all \( 0 \leq k \leq n \)

We will show that P(n) is true for all natural numbers n, by structural induction.

**base case:** n is 0

We WTS that P(n) is true when n is 0, i.e. that \( f_{k+2} \geq 2^{k/2} \) for all \( 0 \leq k \leq 0 \)

That should be easy, since there is only one value of k to check, k=0...

\[
f_{k+2} = f_2 = 1 = 2^0 = 2^{k/2}
\]

because k = 0
by definition of fibonacci sequence
by definition of exponentiation
because k = 0

So the base case is true, because P(n) is true when n is 0.
Structural Induction Example 2

For a nat num \( n \), let \( P(n) \) be the boolean property that \( f_{k+2} \geq 2^{k/2} \) for all \( 0 \leq k \leq n \)

recursive case (aka “inductive step”): \( n = n' + 1 \) where \( n' \) is a nat number

Our induction hypothesis (IH) assumes that \( P(n') \) is true, i.e. that

\[
f_{k+2} \geq 2^{k/2} \text{ for all } 0 \leq k \leq n' (= n - 1)
\]

Given these assumptions, we need to show that \( f_{k+2} \geq 2^{k/2} \) for all \( 0 \leq k \leq n \)

Our assumptions get us all values of \( k \) up to \( n-1 \). Only \( f_{n+2} \geq 2^{n/2} \) remains.

To do this, we need to find a connection between

- \( f_{n+2} \) and \( f_{k+2} \) for \( k \leq n - 1 \):
- \( 2^{n/2} \) and \( 2^{k/2} \) for \( k \leq n - 1 \):

Then \( f_{n+2} = f_{n+1} + f_n = f_{(n-1)+2} + f_{(n-2)+2} \)

\[
\geq 2^{(n-1)/2} + 2^{(n-2)/2} \\
\geq 2 \cdot 2^{(n-2)/2} \\
= 2^{(n-2)/2+1} \\
= 2^{n/2}
\]

(by the IH)

So the recursive case is true, and we are done with our proof!
For every AVL tree $T$, we know that

Also, for every natural number $n$, we know that

Let’s put those together and see what we get:

Take the log (base-2) of both sides (to get rid of the exponent on the right):

Multiply both sides by two then add one to both to get

Namely, $\text{height}(T)$ is $O(\log n)$