Left and right subtrees **recursively** have heights differing by at most one.

**Definition (balance)**

\[ \text{balance}(n) = \text{abs}(\text{height}(n.\text{left}) - \text{height}(n.\text{right})) \]

**Definition (AVL Balance Property)**

An AVL tree is balanced when:

For every node \( n \), \( \text{balance}(n) \leq 1 \)

- This ensures a small depth
- It’s relatively easy to maintain
AVL Trees

**AVL Tree**

**Structure Property:**
0, 1, or 2 children

**BST Property:**
Keys in **Left Subtree** are smaller
Keys in **Right Subtree** are larger

**AVL Balance Property:**
**Left** and **Right** subtrees have heights that differ by at most one.

That is, **all AVL Trees are BSTs**, but the reverse is not true.

AVL Trees rule out **unbalanced BSTs**.
When we insert 3, we violate the AVL Balance condition. What to do?
This “fix” is called a rotation. We’re “rotating” the child node “up”:

This is the only fundamental of AVL Trees!

You can either look at this as “the only way to correctly rearrange the subtrees” or it’s helpful to think of it as gravity.
Inserting 16

Is the result an AVL tree? If not, how do we fix it?
More Complicated Now...

Inserting 16

Is the result an AVL tree? If not, how do we fix it?

This is just the same rotation in the other direction!
AVL Rotations... Are We Done?

We Want...

Cases We’ve Handled

Cases To Handle
AVL Rotations... Are We Done?

We Want...

Cases We’ve Handled

Cases To Handle
Another Case

Second Case

1. When we insert 2, we violate the AVL Balance condition. What to do?

There's only one tree with the BST Property and the Balance Property:

FIXING: Second Case
It Doesn’t Look Like a Single Rotation Will Do…

Double Rotation

First, we rotate b.

First, we rotate b.
It Doesn’t Look Like a Single Rotation Will Do... Double Rotation

First, we rotate $b$.

Now, we're back to the line case. And now it's balanced!
We must **guarantee** that the AVL property gives us a small enough tree.

Our approach: Find a big **lower bound** on the number of nodes necessary to make a tree with height $h$.

$$n \rightarrow h$$

$h \sim \log n$
We must **guarantee** that the AVL property gives us a small enough tree. Our approach: Find a big **lower bound** on the number of nodes necessary to make a tree with height $h$.

What is the **smallest** number of nodes to get a height $h$ AVL Tree?

- **For $h = 0$**
  
  - 1 node

- **For $h = 1$**
  
  - 2 nodes

This is not an AVL tree!
We must **guarantee** that the AVL property gives us a small enough tree. Our approach: Find a big **lower bound** on the number of nodes necessary to make a tree with height $h$.

**What is the smallest number of nodes to get a height $h$ AVL Tree?**

<table>
<thead>
<tr>
<th>For $h = 0$</th>
<th>For $h = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram for $h = 0$" /></td>
<td><img src="image2.png" alt="Diagram for $h = 1$" /></td>
</tr>
</tbody>
</table>
We must **guarantee** that the AVL property gives us a small enough tree. Our approach: Find a big **lower bound** on the number of nodes necessary to make a tree with height \( h \).

What is the **smallest** number of nodes to get a height \( h \) AVL Tree?

For \( h = 0 \)

\[
\begin{array}{c}
\text{This is not an AVL tree!}
\end{array}
\]

For \( h = 1 \)

For \( h = 2 \)

\( \star (3) \)
Does an AVL Tree Work?

What is the **smallest** number of nodes to get a height \( h \) AVL Tree?

The general number of nodes to get a height of \( h \) is:

\[
f(h) = f(h-2) + f(h-1) + 1
\]

We break down where each term comes from. We want a tree that has the **smallest** number of nodes where each branch has the AVL Balance condition.

- \( f(h-1) \): To force the height to be \( h \), we take the smallest tree of height \( h-1 \) as one of the children
- \( f(h-2) \): We are allowed to have the branches differ by one; so, we can get a smaller number of nodes by using \( f(h-2) \)
- +1 comes from the root node to join together the two branches
So, now we solve our recurrence. How?

**Ratio Between Terms**

A good way of solving a recurrence that we expect to be of the form \( f(n) \) is to look at the ratio between terms. If \( \frac{f(h+1)}{f(h)} > X \), then

\[
\frac{f(h+1)}{f(h)} > X \frac{f(h)}{f(h)} \geq X \frac{f(h-1)}{f(h)} \geq \cdots > X^h
\]

So, we evaluate these ratios and see the following:

```
>> 2.0
>> 2.0
>> 1.75
>> 1.7142857142857142
>> 1.6666666666666667
>> 1.65
>> 1.6363636363636365
>> 1.6296296296296295
>> 1.625
>> 1.6223776223776223
>> 1.6206896551724137
>> 1.6196808510638299
>> 1.619047619047619
>> 1.618661257606491
>> 1.618421052631579
>> ...
```