Plan of Action:
Goal: Find a function that defines the lower bound on \( n \) given \( h \).

Given the goal, we begin by defining a function that describes the smallest number of nodes in an AVL of height \( h \):

Theorem:
An AVL tree of height \( h \) has at least ____________________________.

I. Consider an AVL tree and let \( h \) denote its height.

II. Case: ________________

III. Case: ________________

Inductive hypothesis (IH):

Proving our IH:

V. Using a proof by induction, we have shown that:

...and by inverting our finding:

Summary of Balanced BSTs:

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>

Using a Red-Black Tree in C++
C++ provides us a balanced BST as part of the standard library:

```cpp
std::map<K, V> map;
```

The map implements a dictionary ADT. Primary means of access is through the overloaded `operator[]`:

```cpp
V & std::map<K, V>::operator[]( const K & )
This function can be used for both insert and find!
```

Removing an element:

```cpp
void std::map<K, V>::erase( const K & );
```

Range-based searching:

```cpp
iterator std::map<K, V>::lower_bound( const K & );
iterator std::map<K, V>::upper_bound( const K & );
```
Running Time of Every Data Structure So Far:

<table>
<thead>
<tr>
<th></th>
<th>Unsorted Array</th>
<th>Sorted Array</th>
<th>Unsorted List</th>
<th>Sorted List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traverse</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Binary Tree</th>
<th>BST</th>
<th>AVL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traverse</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Range-based Searches:**

Q: Consider points in 1D: \( p = \{p_1, p_2, ..., p_n\} \).
...what points fall in \([11, 42]\)?

**Tree Construction:**

**BTree Motivation**

Big-O assumes uniform time for all operations, but this isn’t always true.

However, seeking data from the cloud may take 100ms+.
...an \( O(\lg(n)) \) AVL tree no longer looks great:

---

**BTree Motivations**

Knowing that we have long seek times for data, we want to build a data structure with two (related) properties:

1. 

2. 

---

**CS 225 – Things To Be Doing:**

1. mp_mosaics due Monday!
2. lab_trees due Sunday!
3. Find a team if you are going to do the Final Project
4. Daily POTDs are ongoing!

---

Running Time: