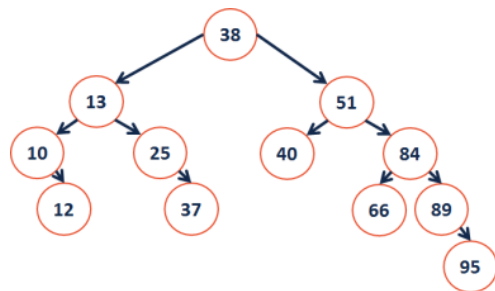


**Removing an element from a BST:**

`_remove(40)`  
`_remove(25)`  
`_remove(10)`  
`_remove(13)`



One-child Remove	Two-child remove
<code>BinaryTree.cpp</code>	
<pre> template &lt;class K, class V&gt; void BST::_remove(TreeNode *&amp; root, const K &amp; key) {  }                     </pre>	

**BST Analysis:**

Every operation we have studied on a BST depends on:

...what is this in terms of the amount of data, **n**?

**BST – Simple Proofs**

**Q:** Given a height **h**, what is the maximum number of nodes (**n**) in a valid BST of height **h**? Provide an outline of a proof.

**Q:** Given a height **h**, what is the minimum number of nodes (**n**) in a valid BST of height **h**? Provide an outline of a proof.

**Final BST Analysis**

For every height-based algorithm on a BST:

Lower Bound:

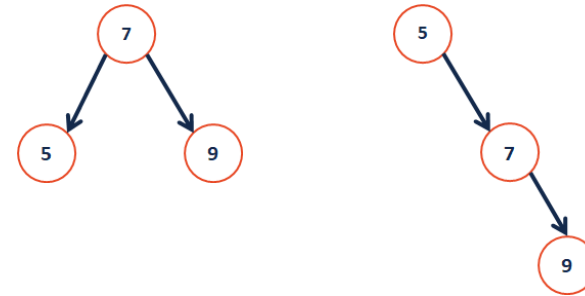
Upper Bound:

Why use a BST over a linked list?

**Q:** How does our data determine the height?

1 3 2 4 5 7 6      vs.      4 2 3 6 7 1 5

**Q:** How many different ways are there to insert data into a BST?



**Q:** What is the average height of every arrangement?

...what is the intuition here?

We define the **height balance** (b) of a BST to be:

We define a BST tree T to be **height balanced** if:

operation	BST Avg. Case	BST Worst Case	Sorted Array	Sorted List
find				
insert				
delete				
traverse				

---

---

### Height Balance on BST

What tree makes you happier?