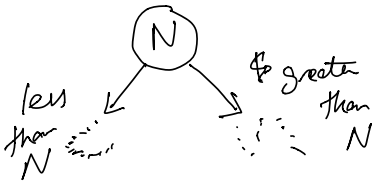
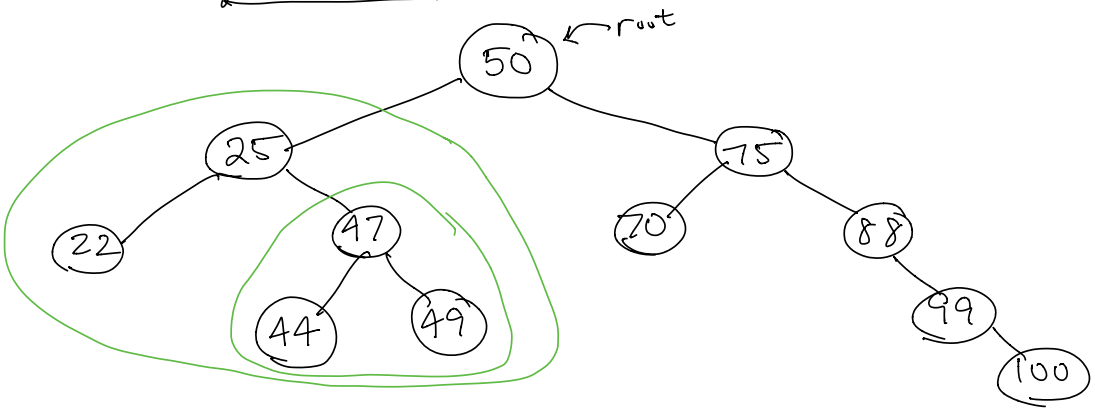
Binary Search Tree

- A binary search tree (BST) is a specific kind of binary tree.
- To be a BST, every node N in the tree must satisfy:
  - The left child of N and all further descendent nodes must have values less than N.
  - The right child of N and all of its descendents must have values greater than N.





Why are BST's so cool?

Imagine I want to know if 44 is in this BST.

Search

```
boolean search (node, searchKey) {
    if searchKey == node.data
        then return true;
    elseif searchKey < node.data
        return search (node.left, searchKey)
    else
        return search (node.right, searchKey)
```

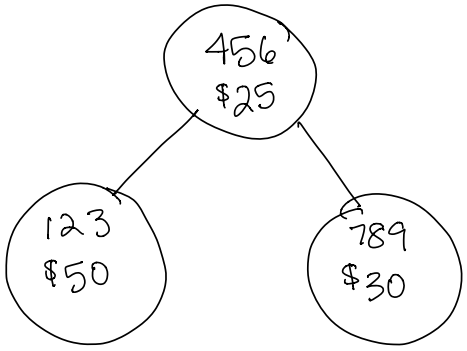
if node == null return false;

Implement a MAP w/ a BST

Example: Keys: Bank Acct #s
Values: balances ($)

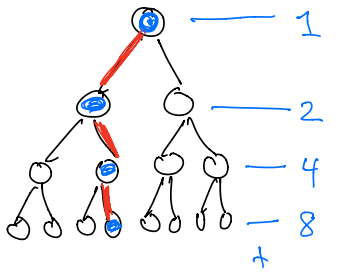
Keys become the numbers that you search for. These are the items that obey the BST rules.

Suppose I have 3 accts: 123 → $50
456 → $25
789 → $30



Why is searching a BST faster than searching an ArrayList or a LL?

ArrayLists: O(n)
LinkedLists: O(n)
BST: O(log n)

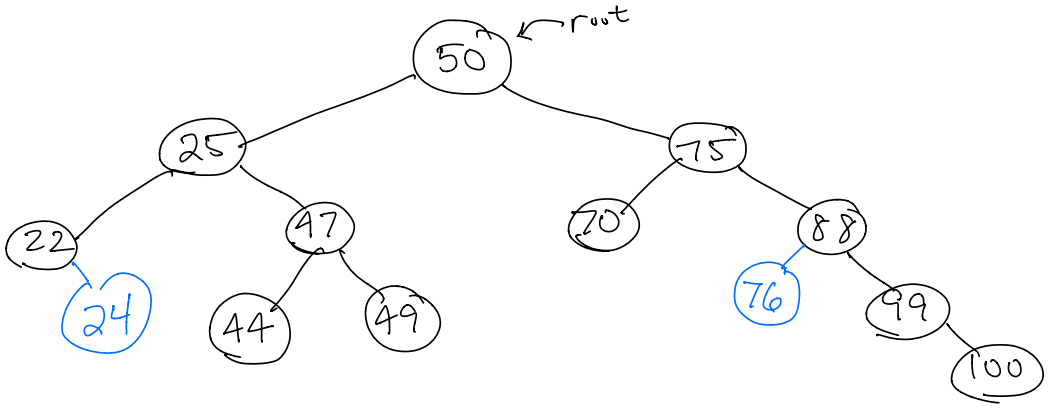


max # of comparisons = 4 ↙ ↘ n

$\log_2(n)$ where $n=16 \rightarrow 4$

$\log_2(n)$ where $n=32 \rightarrow 5$

Adding to a BST (put/insert)



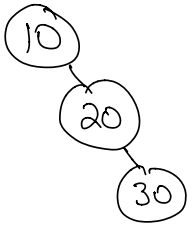
Add: 76

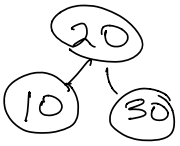
Add: 24

10, 20, 30 → Add these #s to an empty BST in order.

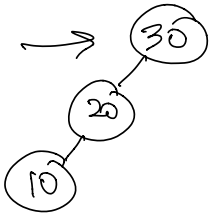
Add 20:

Add 30:





30, 20, 10 →



→ 20, 10, 30

→ 20, 30, 10