Binary Search Trees

**BST Property**

E.g.

```
      5
     / \
    3   9
   /   / \
  1   4   11
```

**In-order Traversal**

```
1, 3, 4, 5, 9, 11
```

Recall: Heap property

```
X > all elements in A
X ≤ all elements in B
```

**BST Find**

```
Find: 4
```

```
Find: 11
```

```
Find: -1
```
BST Insertion

BST Removal & Successor

Remove 4

Remove 9

Remove 3

4 is 3's successor
Swap 3 and 4

Remove 3 as usual
More Ex. on Finding the Successor

Successor of 8 is 10

8 doesn't have a right child
Go up the parent chain, until you find a node that is the LEFT child of its parent. Its parent is the successor.

Successor of 20 is 21

1. 20 has a right child
2. So find the "left-most" node in its right subtree.

Successor of 10 is 14

Same as the prev. case.
Note: Don't turn right when you are going down the right subtree.

Successor of 7.2 is 8.

Note: Only a left child works when finding a successor up a parent chain.

Successor of 26 doesn't exist
**Height & Balance**

```
max steps: 3
```

```
"Good" BST
```

**AVL-Tree**

A self-balancing BST.

```
x is balanced if 
\[ |A-B| \leq 1 \]
```

```
\[ \rightarrow \max(A,B)+1 \]
```

```
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11
```

```
Degenerate into a linked list.
```

```
c < y < p
```
AVL Insert/Remove

1. Do a BST insert/remove

2. Look the parent of the node you touched.
   a. Is it still balanced?
   b. If it's still balanced, did its height change?
   
   If (a) is false, do a rotation.
   
   If (b) is true (height changed), return to step 1.

   If you did a rotation and its height has changed, also return to step 1.

Types of rotations you can do:

- **Left subtree is taller**
  
  \[ h + 2 \]
  
  \[ h \]

  \[ \text{rotation to the right} \]

  \[ x \geq y \rightarrow \text{Zig-Zig} \]

  \[ x < y \rightarrow \text{Zig-Zag} \]

  First rotate left child to the left
  Then rotate parent to the right
Example

Remove 4

Remove 3

Remove 1