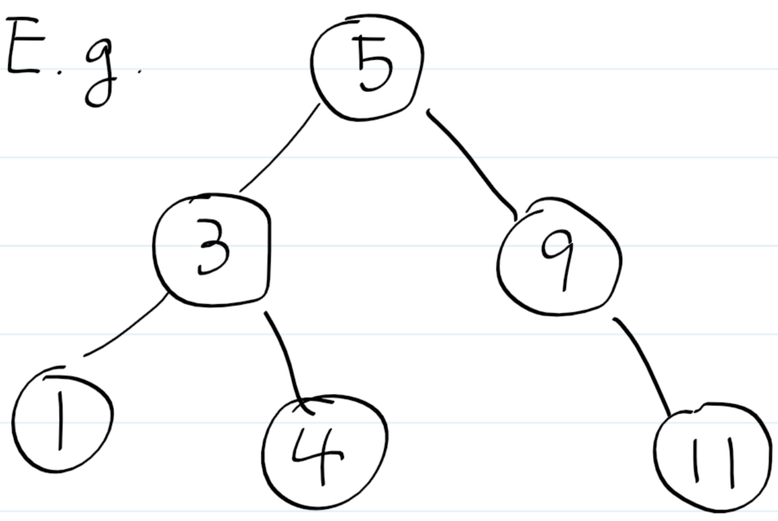
Binary Search Trees

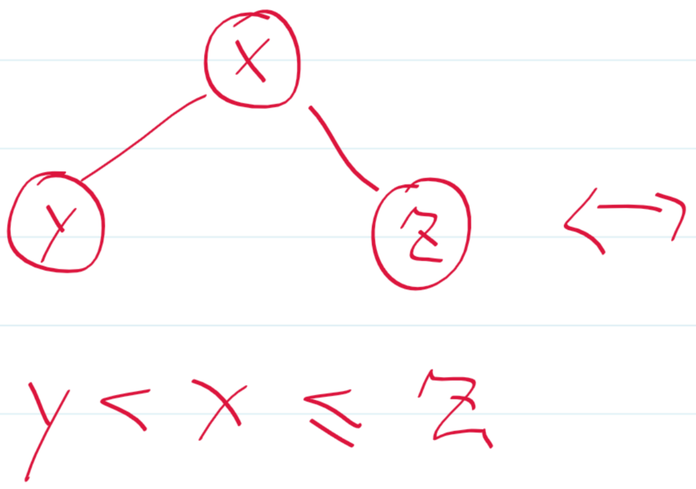
BST Property

E.g.



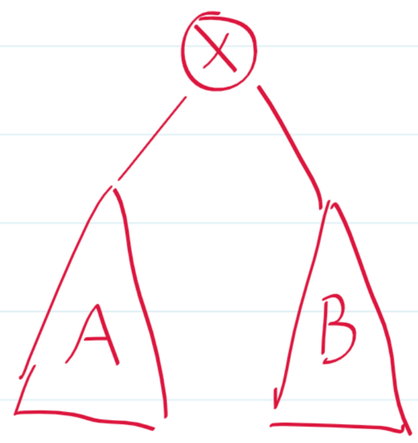
In-order Traversal

1, 3, 4, 5, 9, 11



$y < x \leq z$

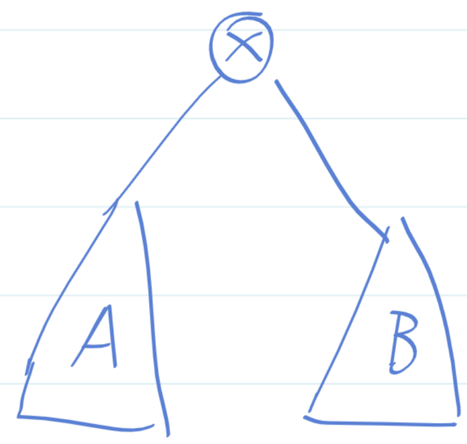
for every node x



x > all elements in A

$x \leq$ all elements in B

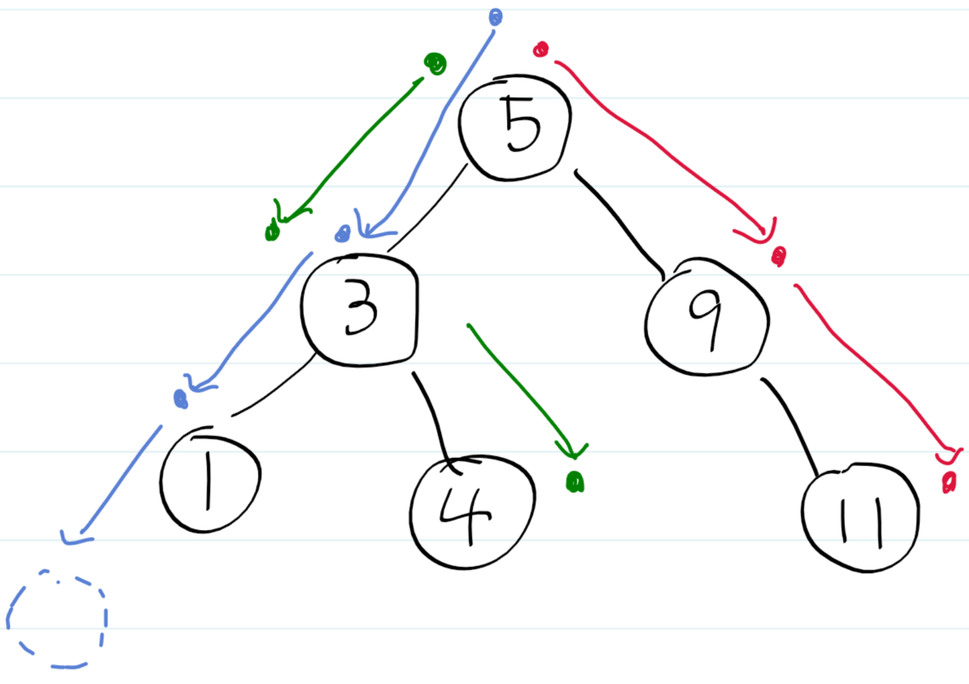
Recall: Heap property



$x \geq$ all A

$x \geq$ all B

BST Find

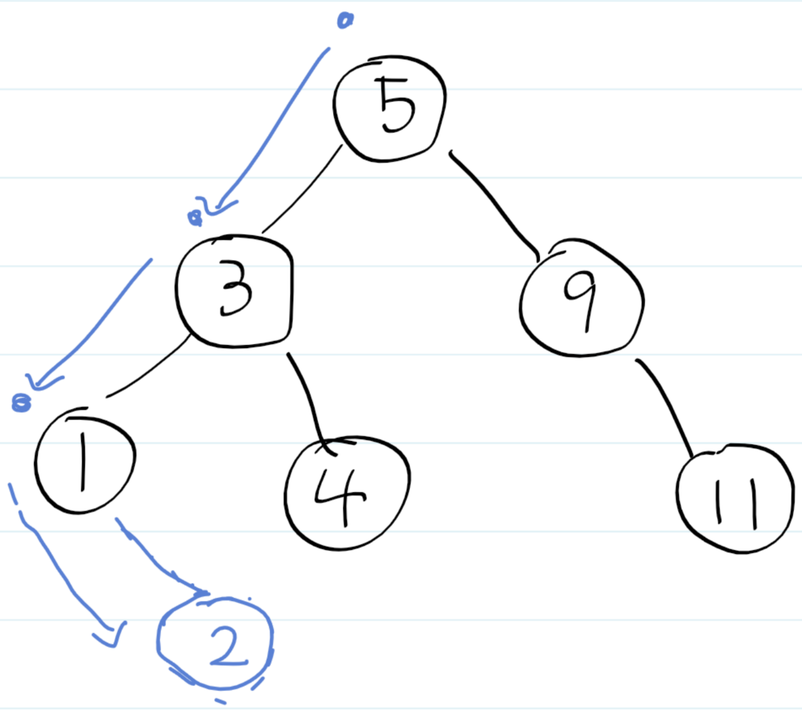


Find: 4

Find: 11

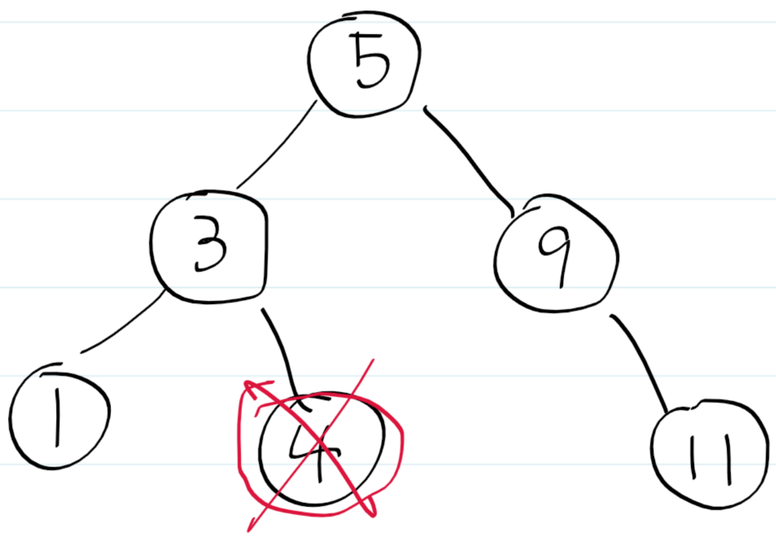
Find: -1

BST Insertion

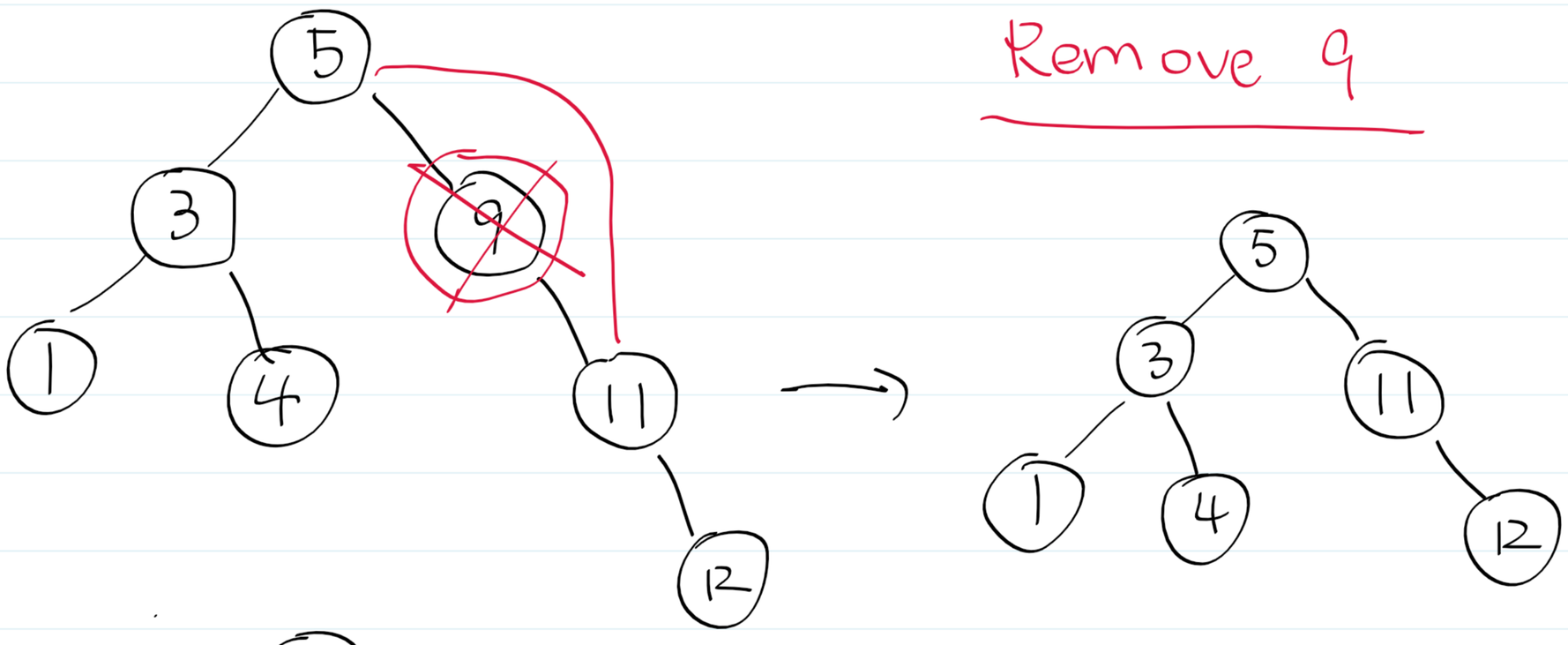


Insert 2

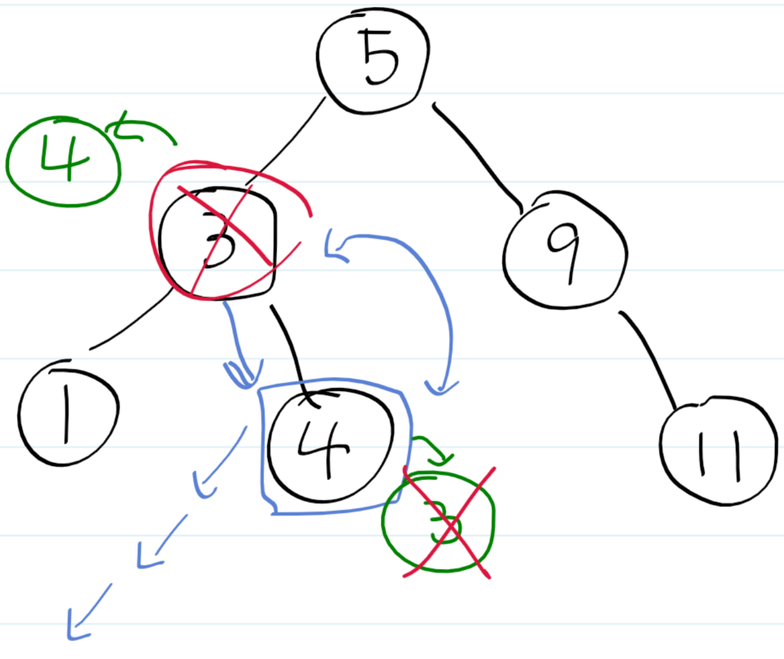
BST Removal & Successor



Remove 4



Remove 9



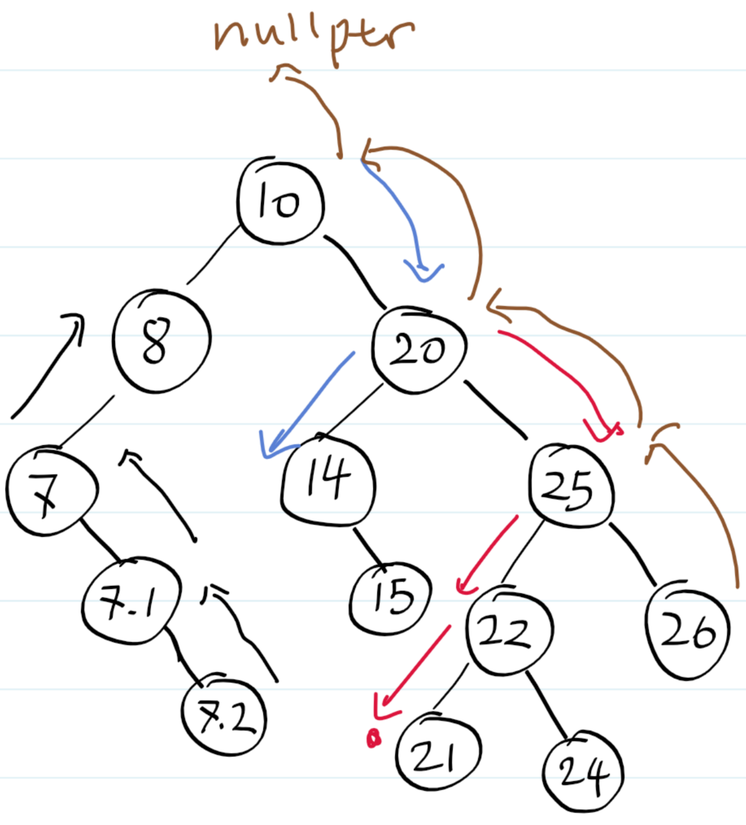
Remove 3

4 is 3's successor

Swap 3 and 4

Remove 3 as usual

More Ex. on Finding the Successor



Successor of 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 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.

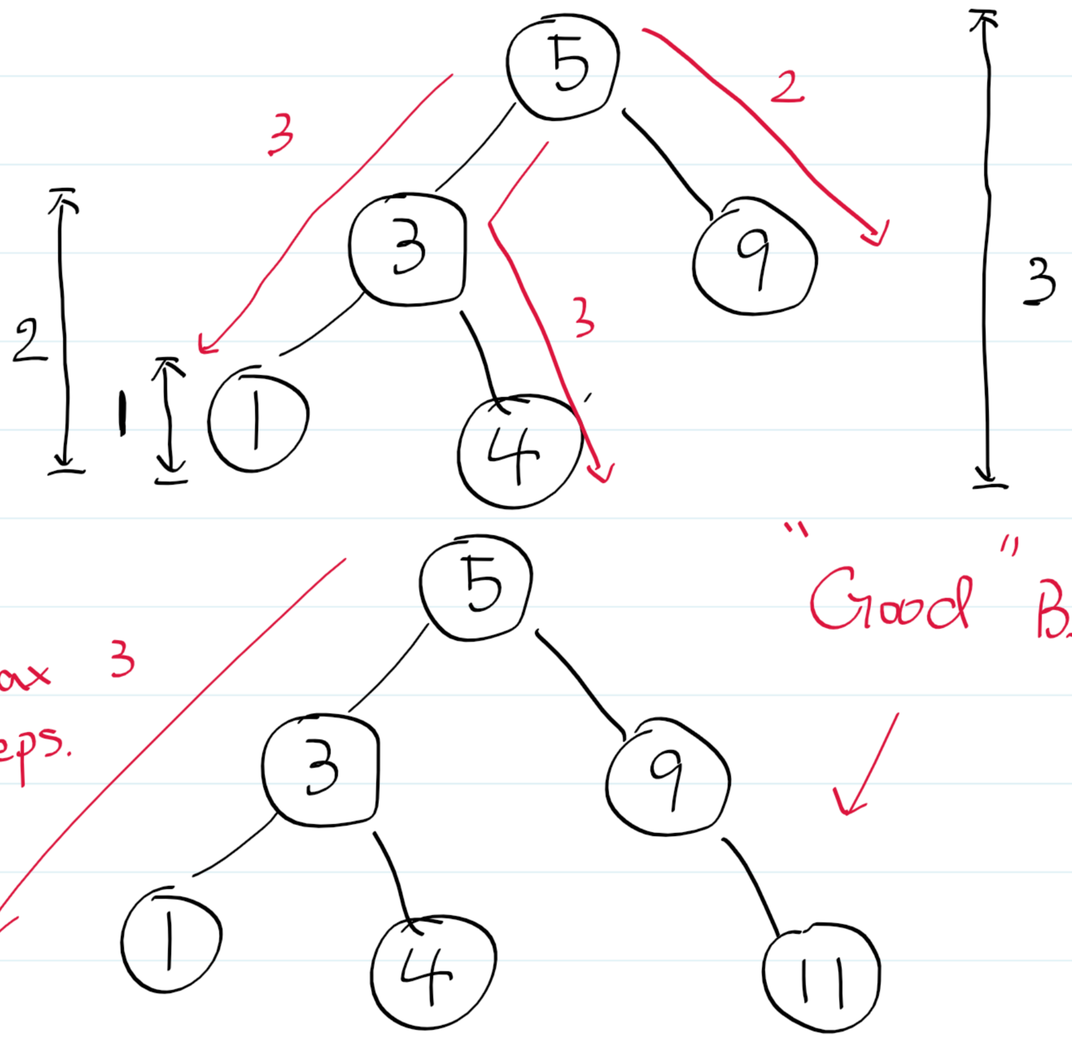
Its parent is the successor.

Successor of 7.2 is 8.

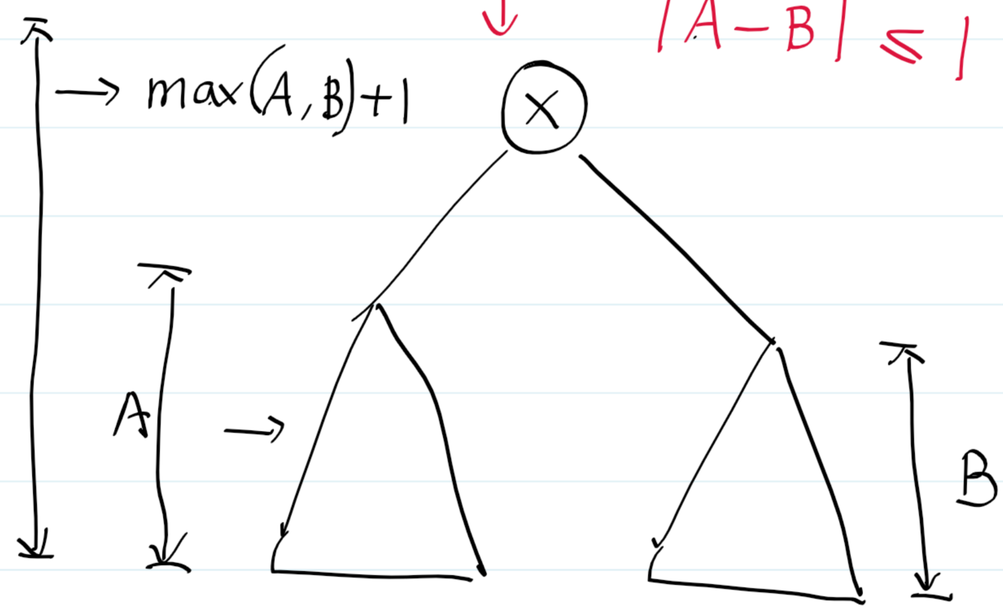
NOTE: Only a left child works when finding a successor up a parent chain.

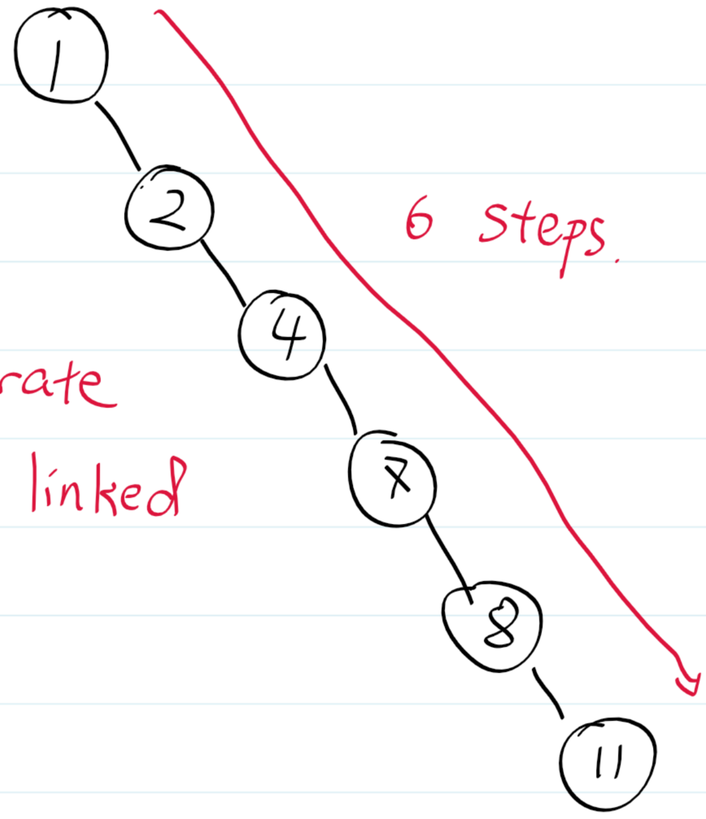
Successor of 26 doesn't exist

Height & Balance



x is balanced if $|A-B| \leq 1$

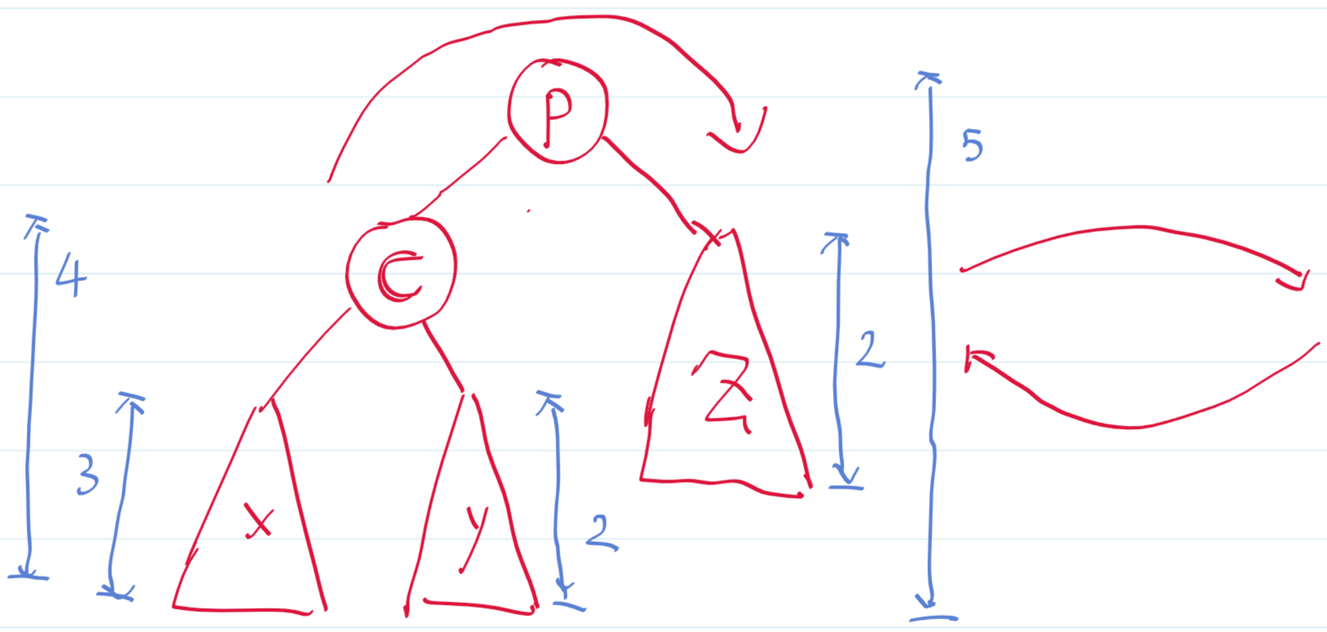




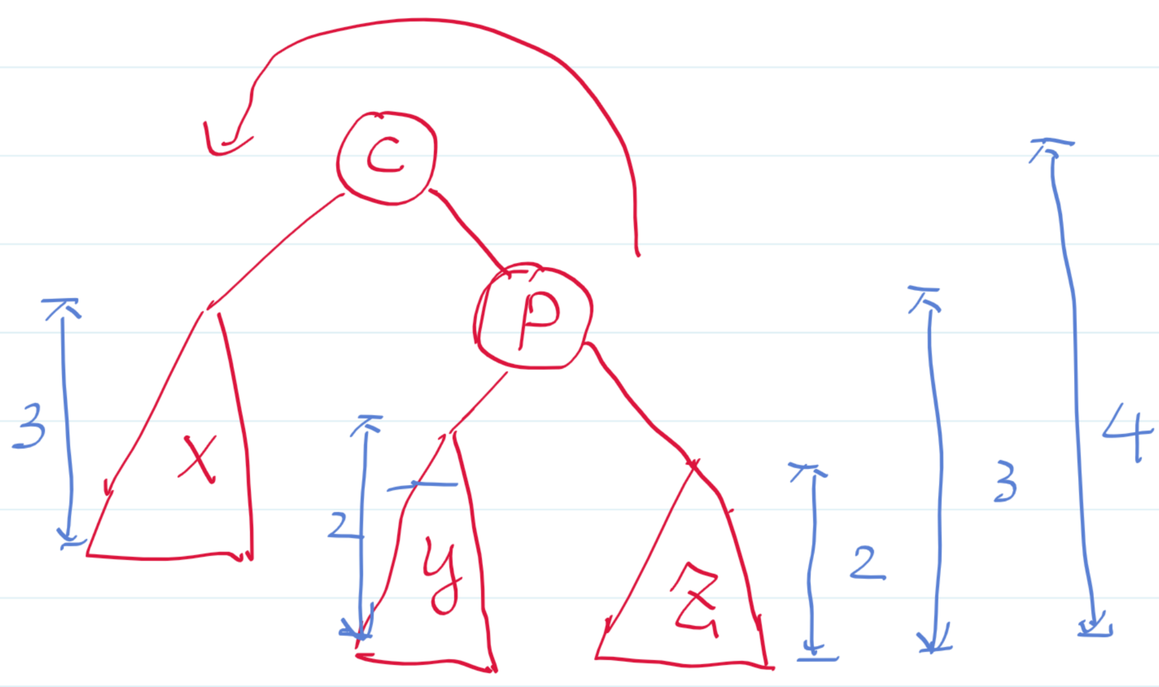
Degenerate into a linked list.

AVL-Tree

A self-balancing BST.



$C < y < P$



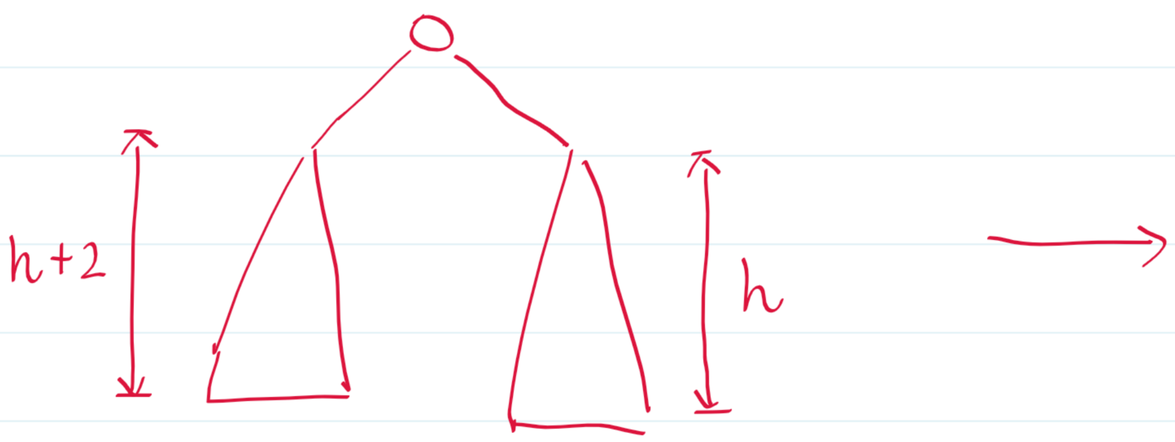
$C < y < P$

AVL Insert / Remove

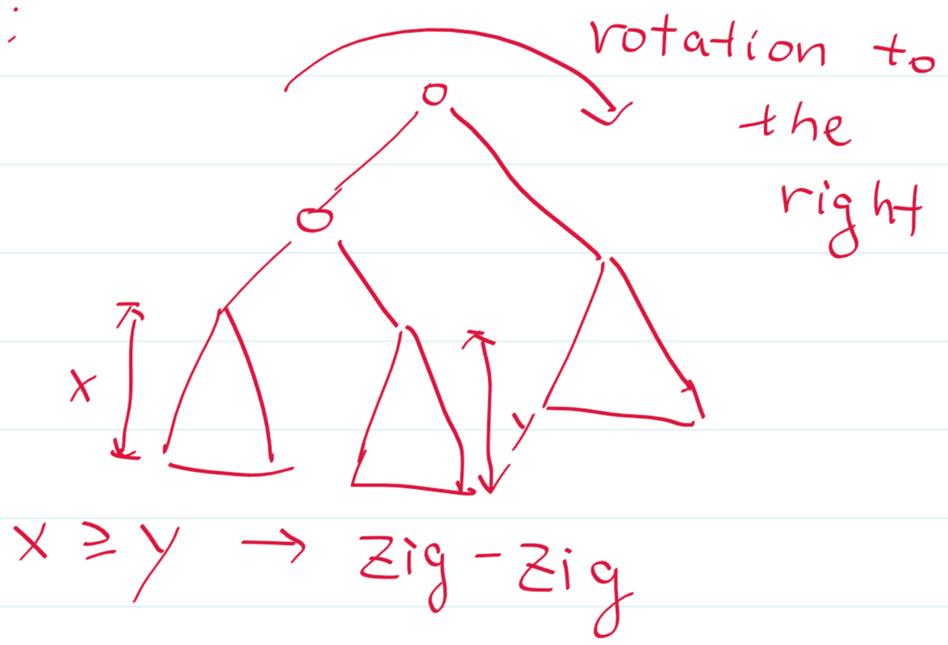
1. Do a BST insert / remove
2. Look the parent of the node you touched.
   a. Is it still balanced?
   b. If it's still balanced, did its height change?

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



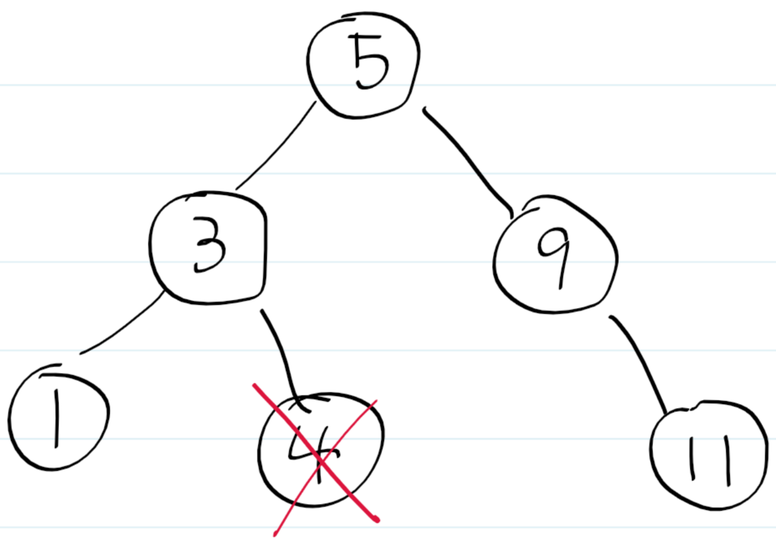
$x \geq y$ → zig-zig

$x < y$ → zig-zag

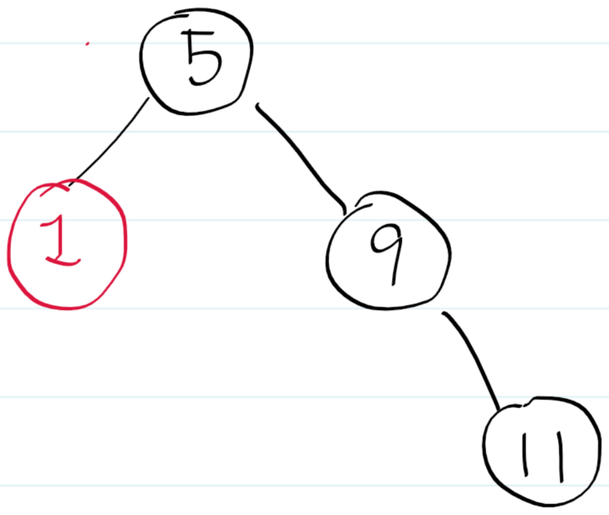
First rotate left child to the left
Then rotate parent to the right

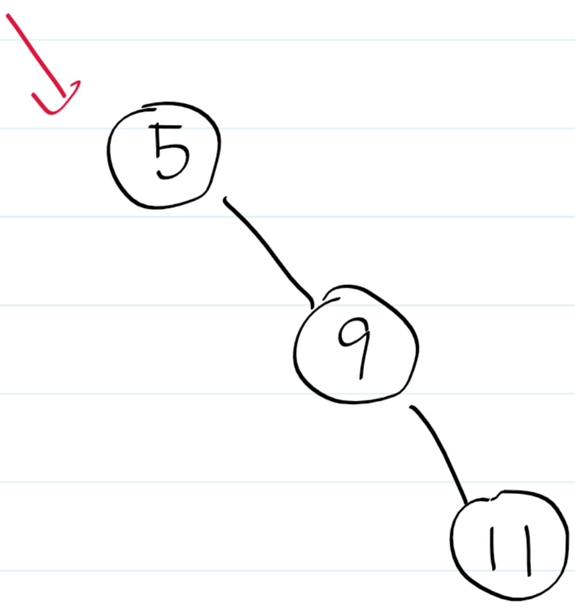
Example

remove 4



Remove 3





Remove 1