# CS 180 - AVL trees

Note Title 11/13/2009

Announcements

- Extra Credit exercise today!
  (take out a blank piece of paper
  and pen/pencil, or borrow one)

- New program out today, due in 1 week

Binary!

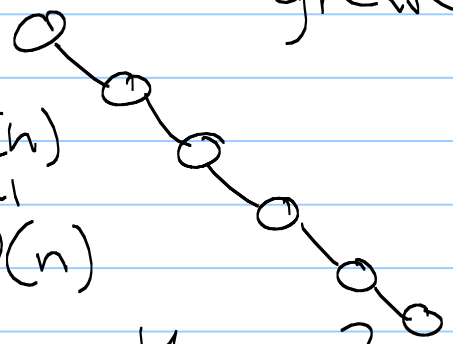
## Search trees

What are they? a tree that is "sorted", so
- inorder traversal results in sorted list

- For any node, left child is smaller + right is greater.

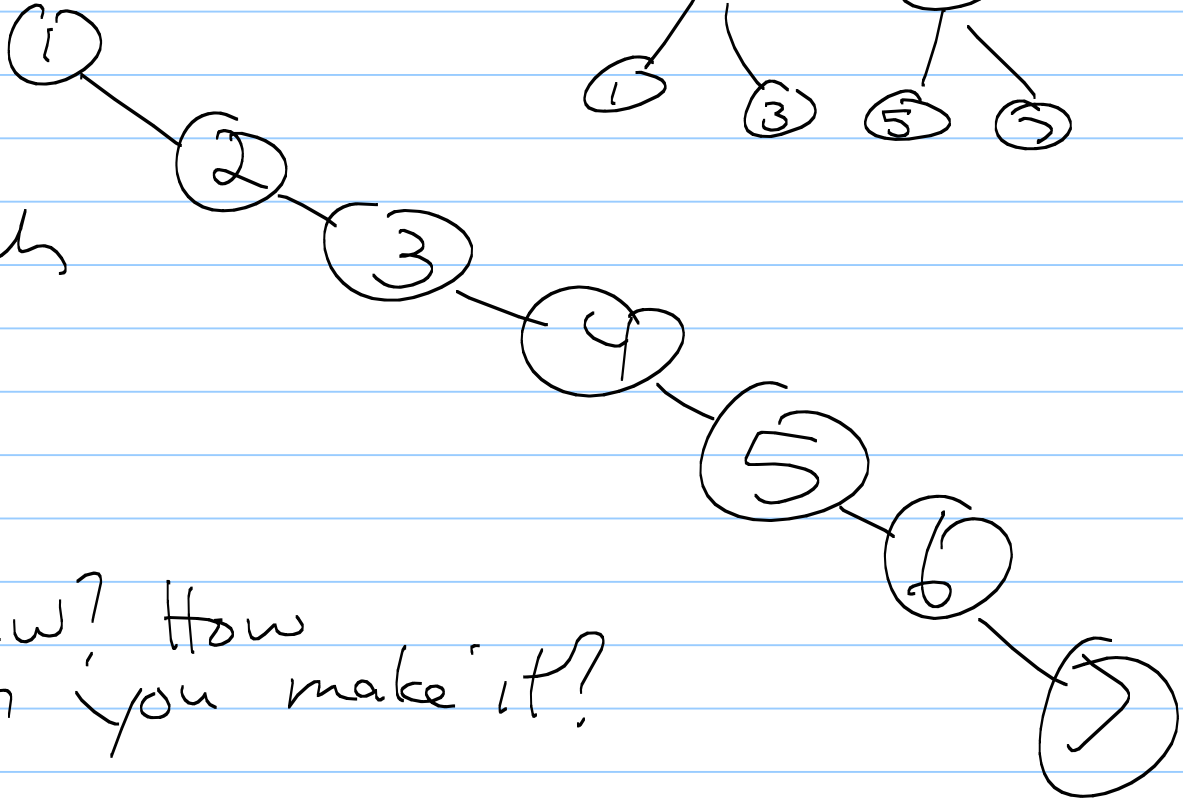
How fast is: search?
insert?
delete?

$$\left.\begin{array}{l}\text{search?}\\\text{insert?}\\\text{delete?}\end{array}\right] O(h) = O(n)$$

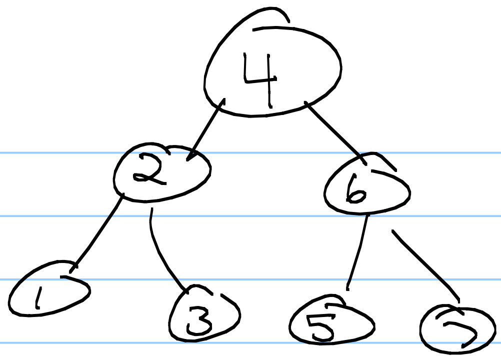


How could we improve these?
improve height = balance tree

Consider this tree:

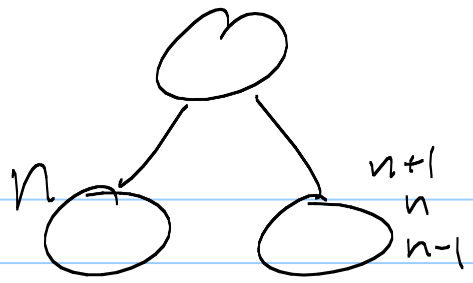


- Make a search tree that is balanced.



Can you redraw? How good can you make it?

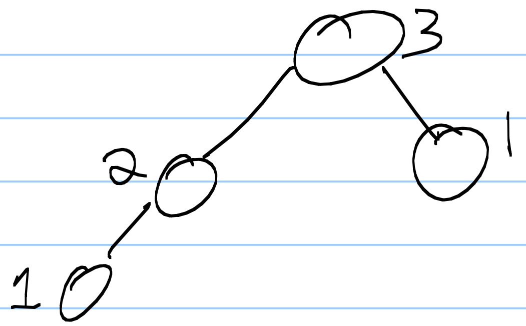
# AVL trees :



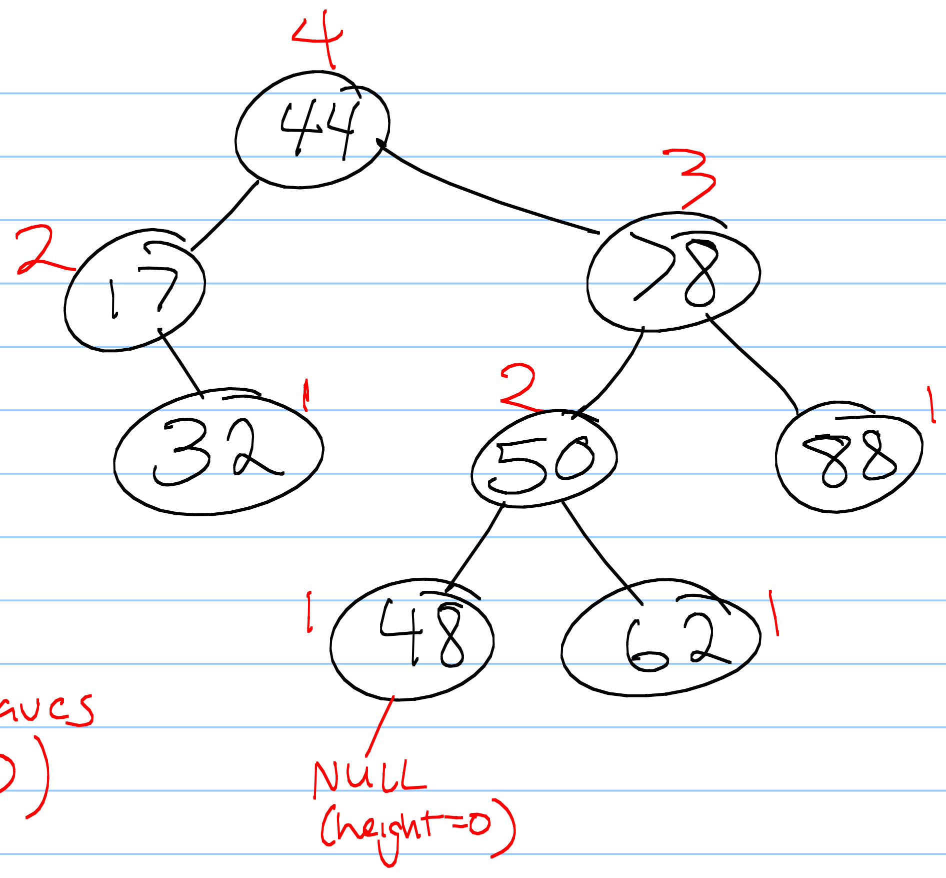
Height-Balance property:
For every (internal) node of T, the heights of the children differ by at most 1.

↳ height of tree $\leq 2 \cdot \log_2 n$

(Question: How do we calculate height?)



Ex:



what are
the heights?

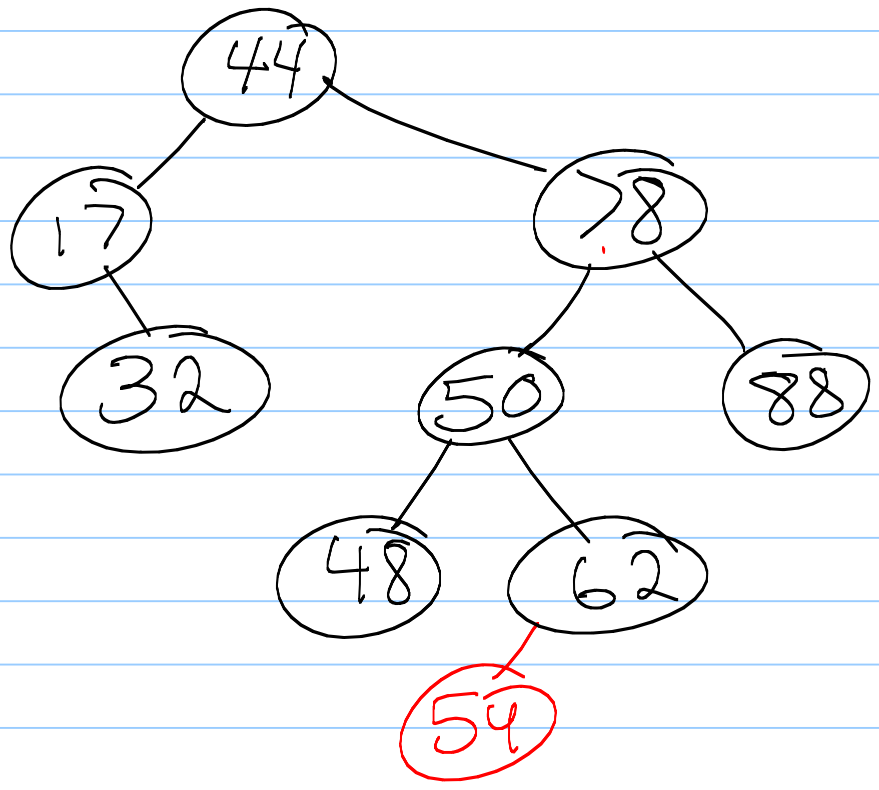
(assume NULL leaves
are height 0)

Now how can we mess this up?

(In other words how
can the height
change?)

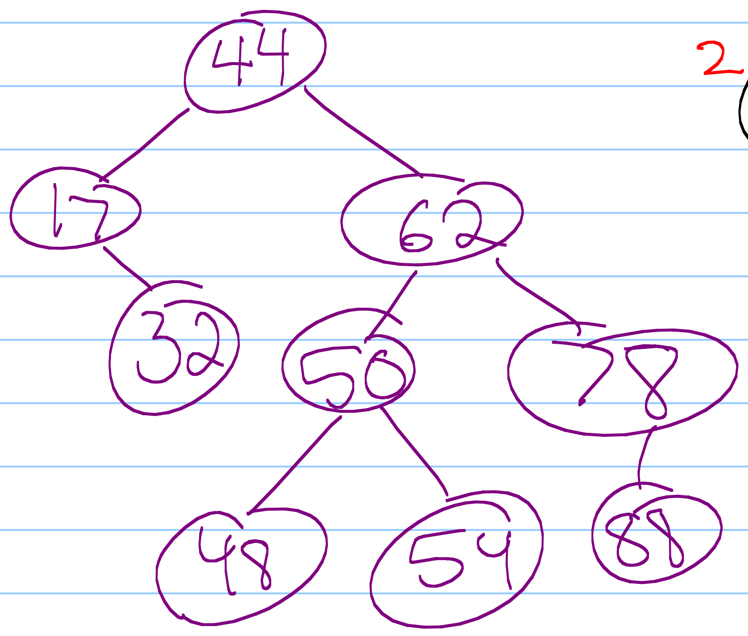
insert
and remove

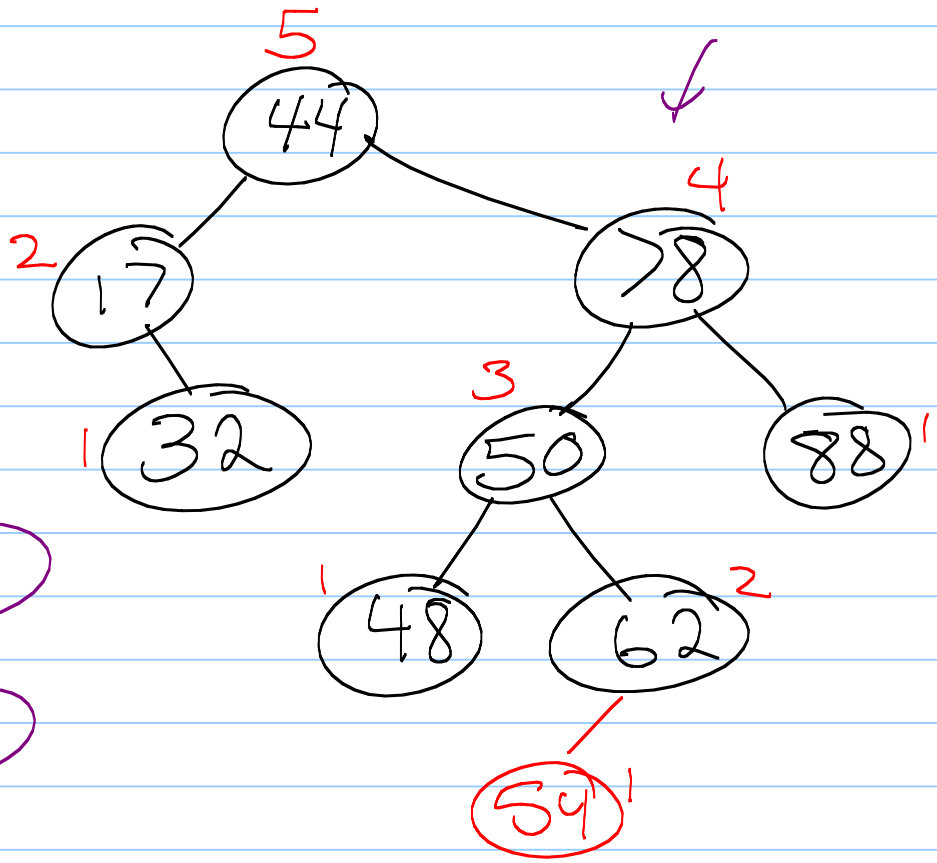
insert (54)



Insert :

insert(54)

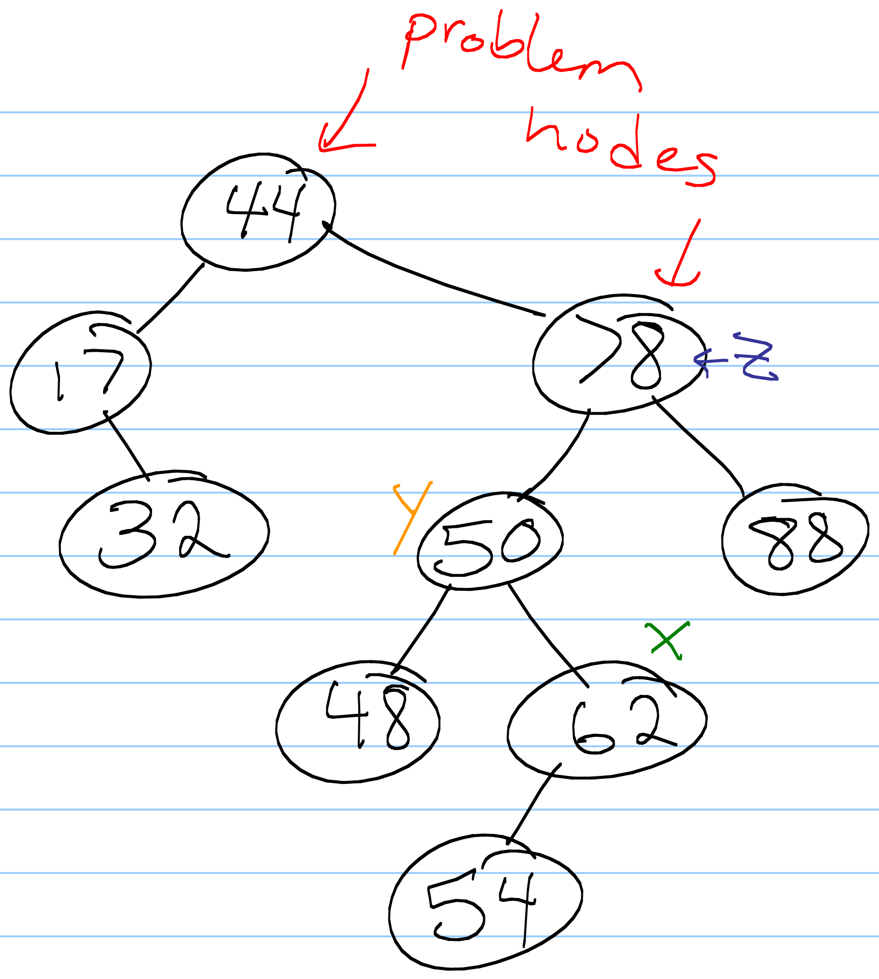




Consider the lowest node which does <u>not</u> satisfy height-balance property.
↳ call this z

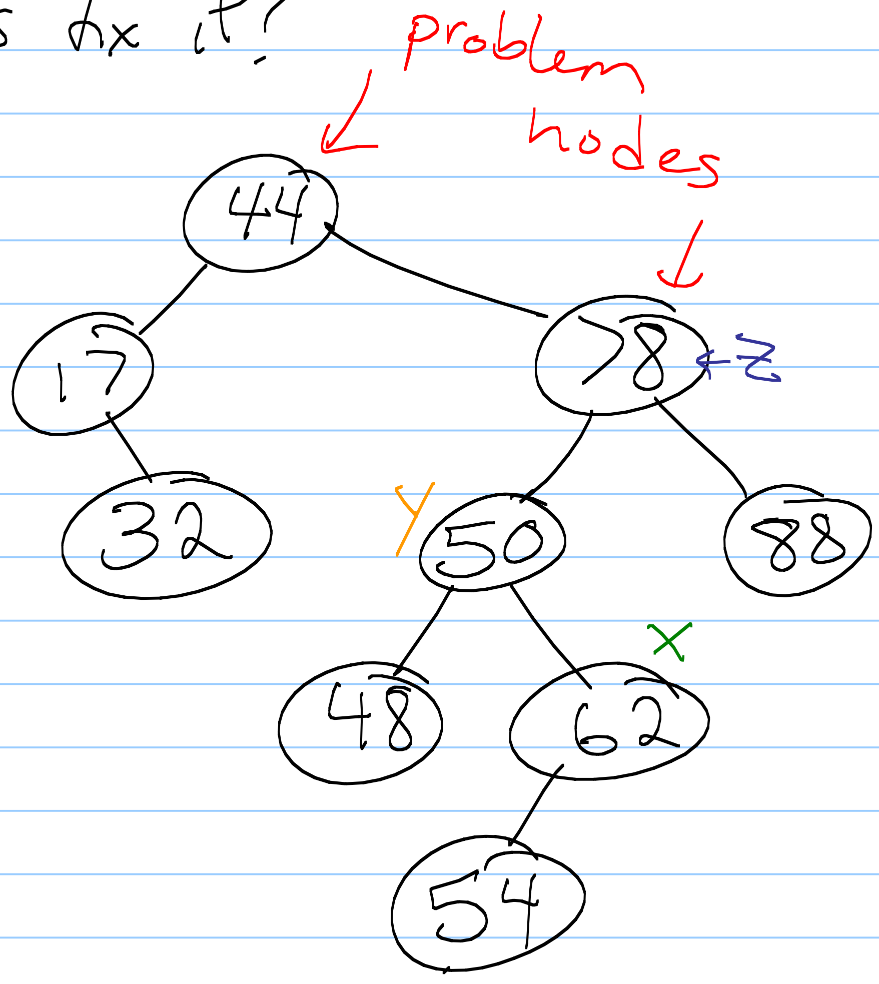
Let y be z's child with higher height.

Let x be y's child with higher height.



OK - how did you guys fix it?

Here, we promoted x.
(& demoted z)

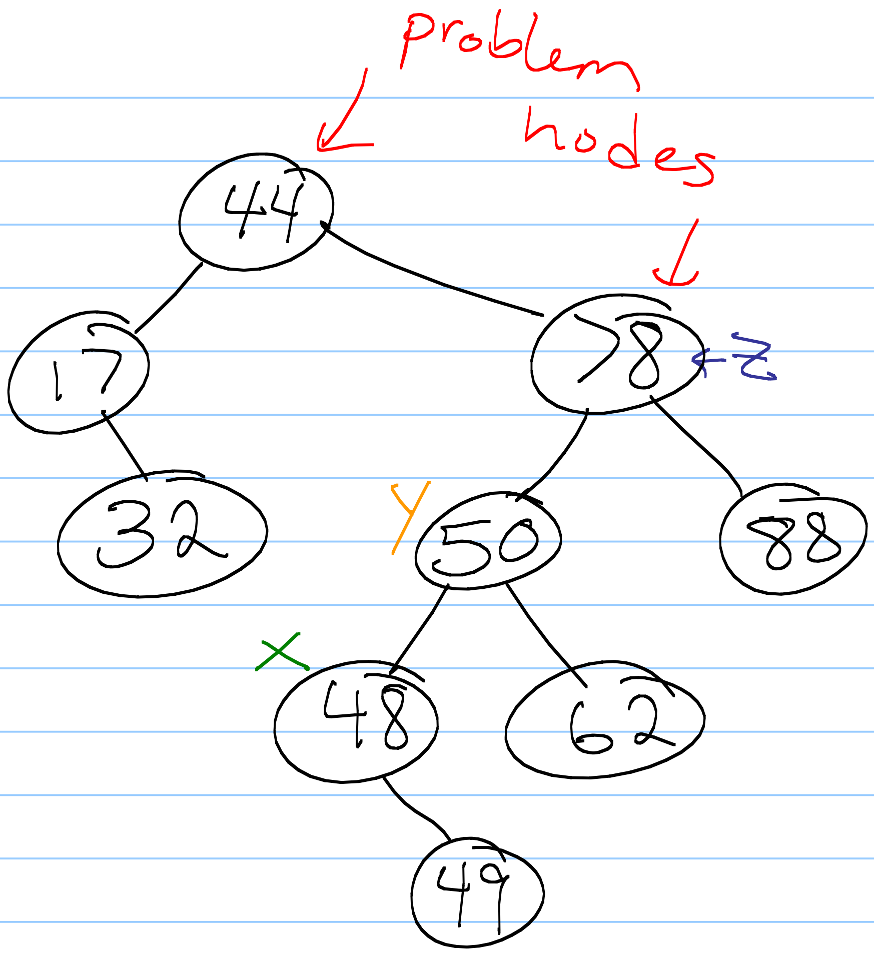


Another one: insert(49)

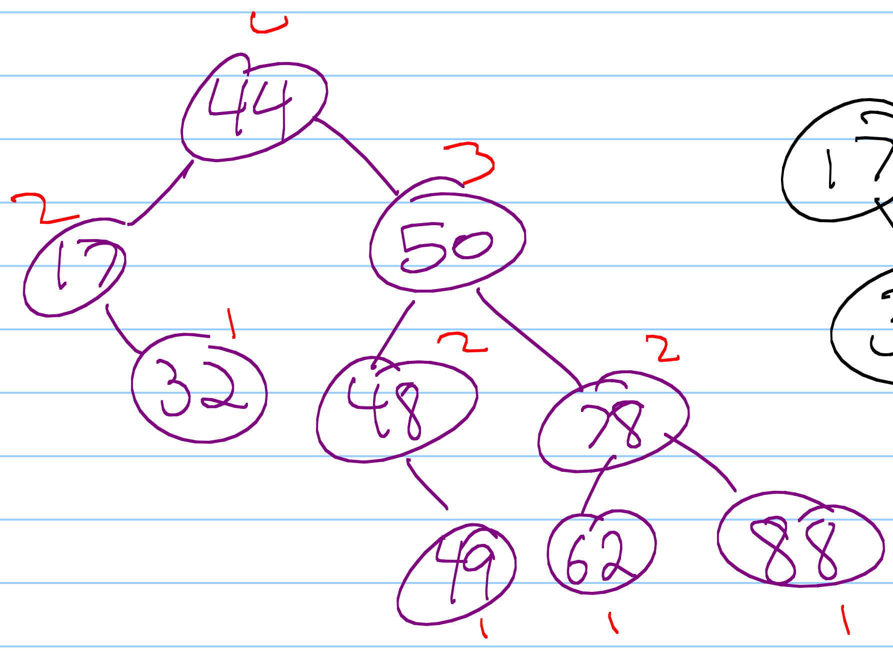
Consider the lowest node which does not satisfy height-balance property.
↳ call this $z$

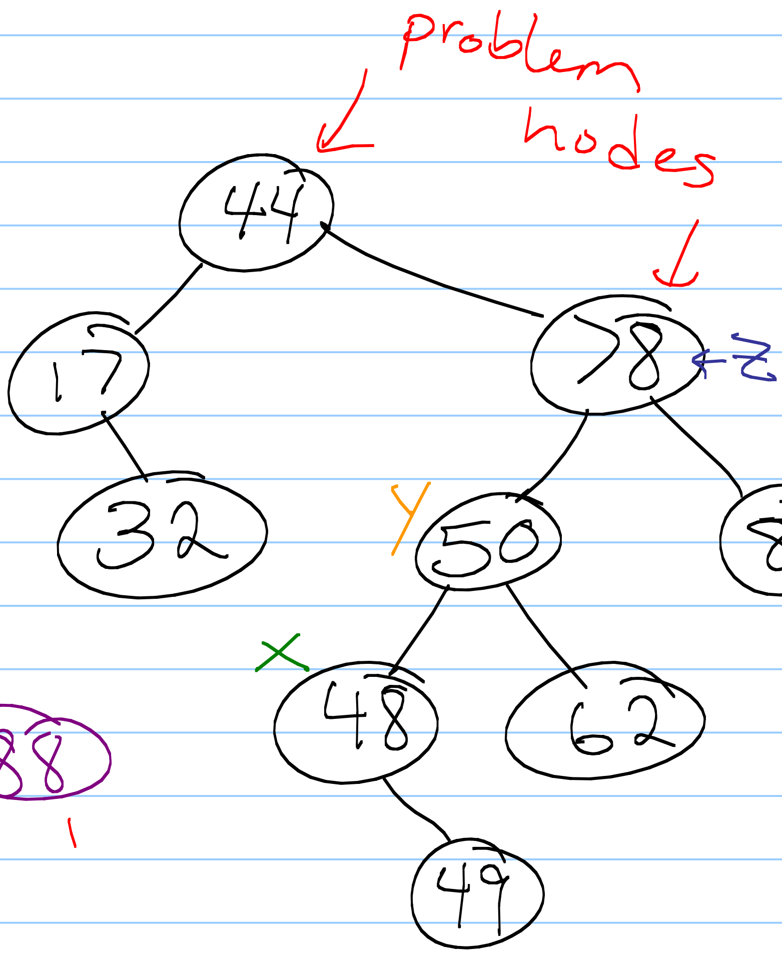
Let $y$ be $z$'s child with higher height.

Let $x$ be $y$'s child with higher height.

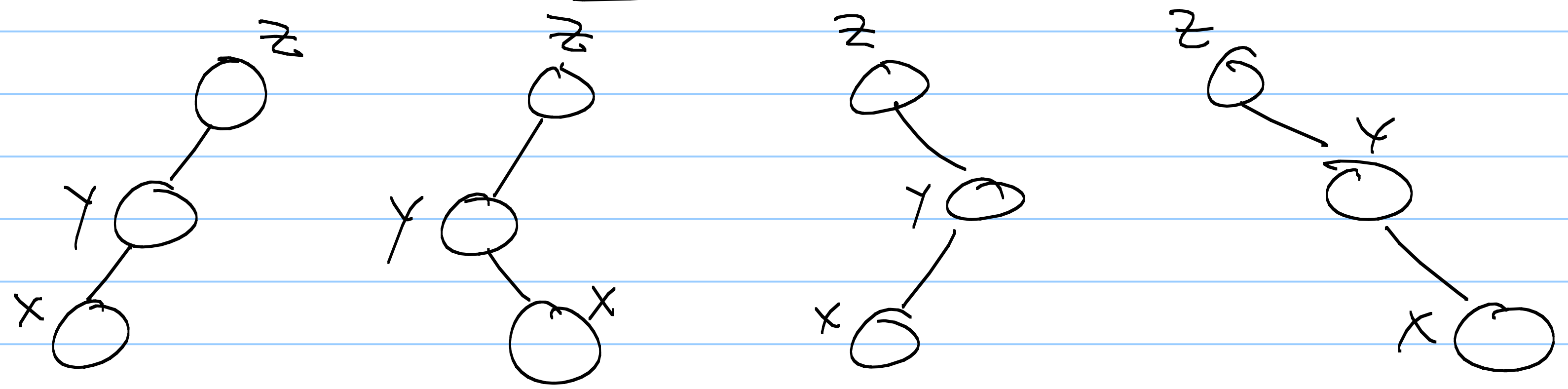


How to fix this one?

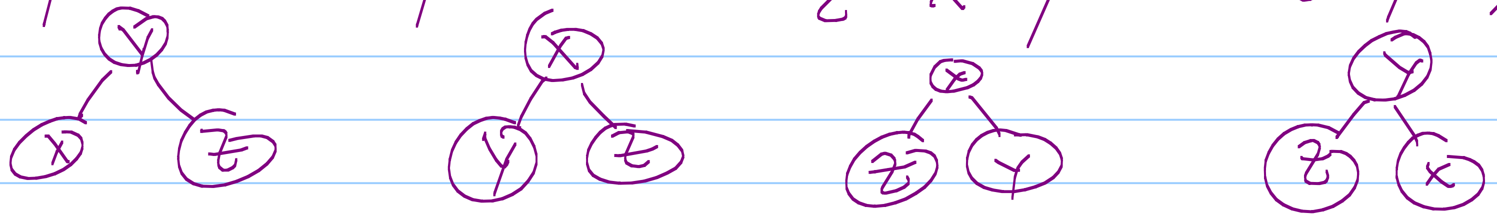




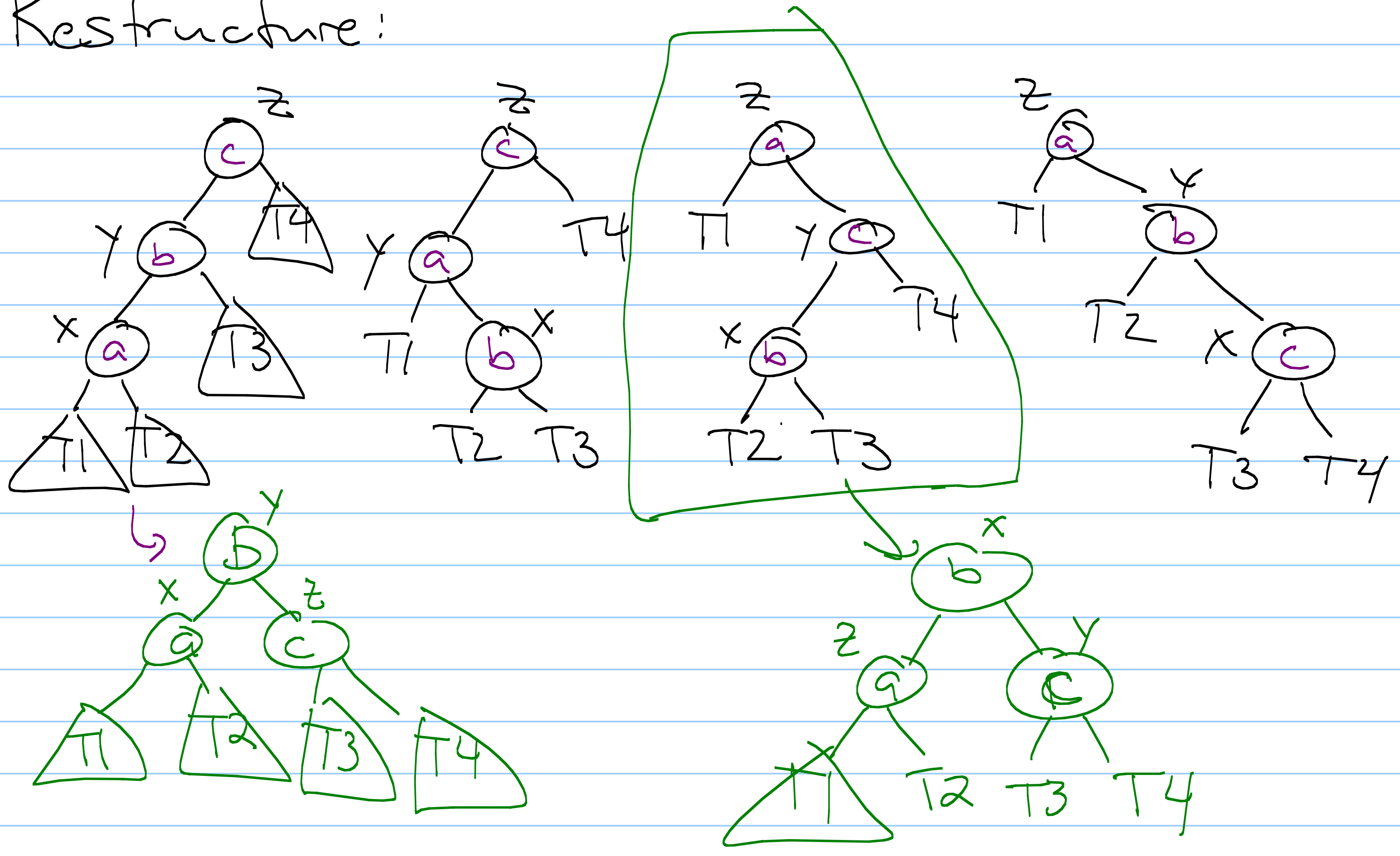
Generalize: Consider x, y, & z.
List them in an inorder traversal:



= x − y − z     = y − x − z     = z − x − y     = z − y − x



Restructure:



Any way you do it, b becomes the new root of this subtree!

↳ and a is left child, c is right child.

Any way you do it, a & c's children are T1, T2, T3, T4 (in that order)!

How long does this take?

$O(1)$

Why? 14 pointer manipulations to alter this root

$O(\log n)$ time to <u>find</u> z
$O(1)$ to fix z