CS180 - AVL trees

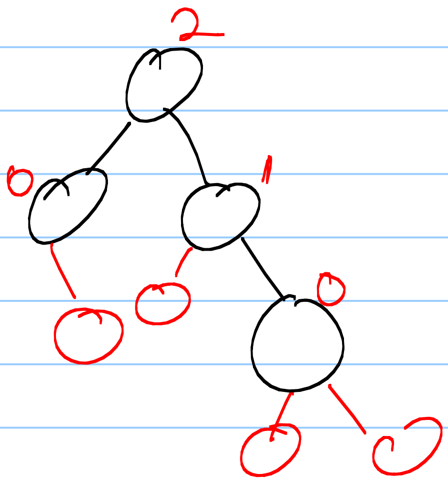
Note Title 10/31/2012

Announcements

AVL Trees :

Height - Balance Property :
For every node of T, the heights of the children differ by at most 1.

$\Rightarrow$ max height $\leq \boxed{2 \lceil \log_2 n \rceil}$

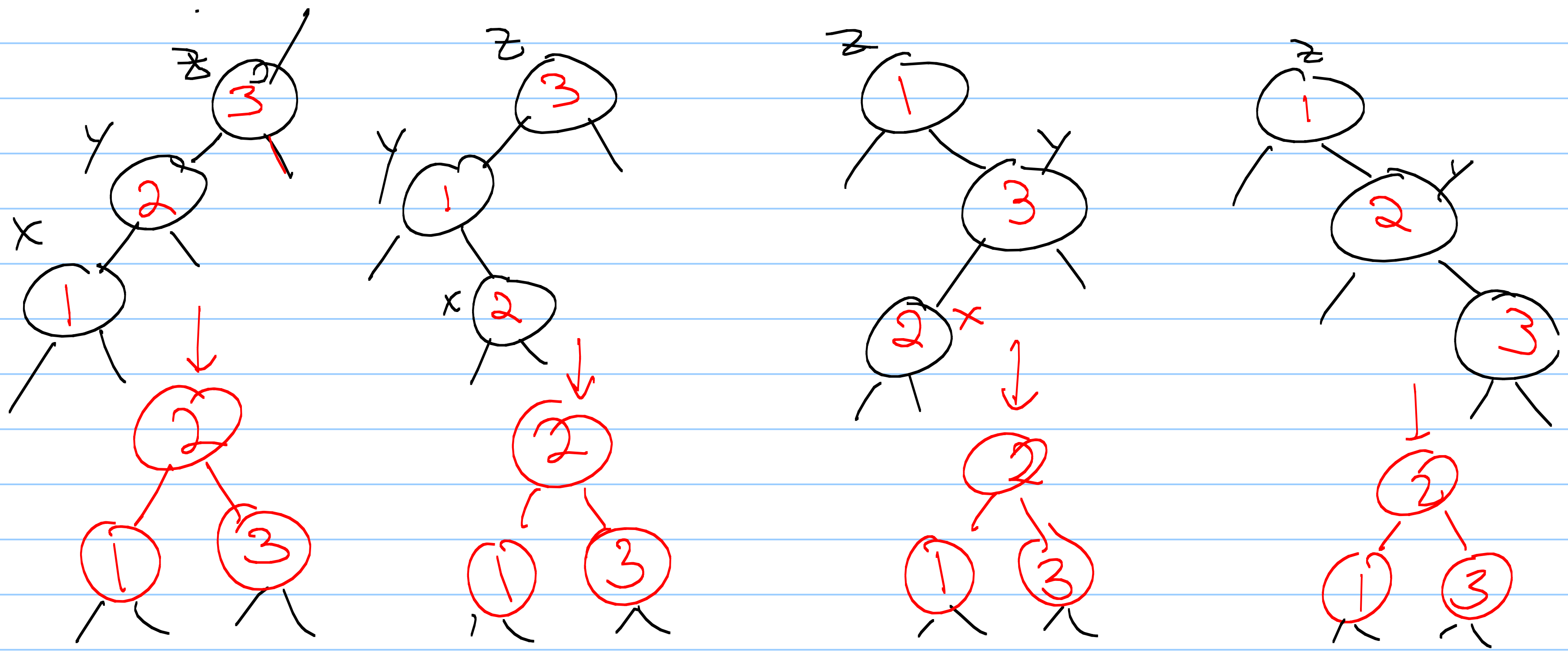


(How do we calculate height again?)

recursive - look at 2 children, take max, +1

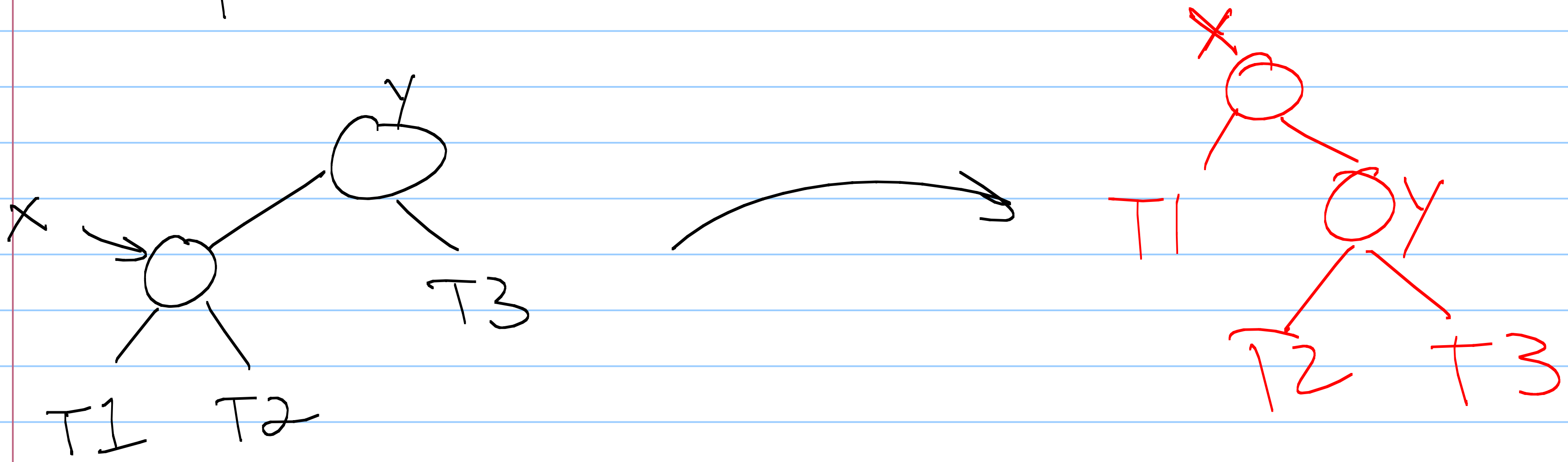
Insert: Do BST insert.

Then find lowest unbalanced node z:

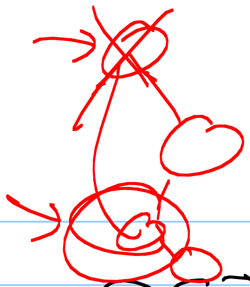


Key operation:

- pivot (x)



Removing in AVL trees



Step 1: Remove - just like in BST

Step 2: Re-balance (if removal violated H-B property.)

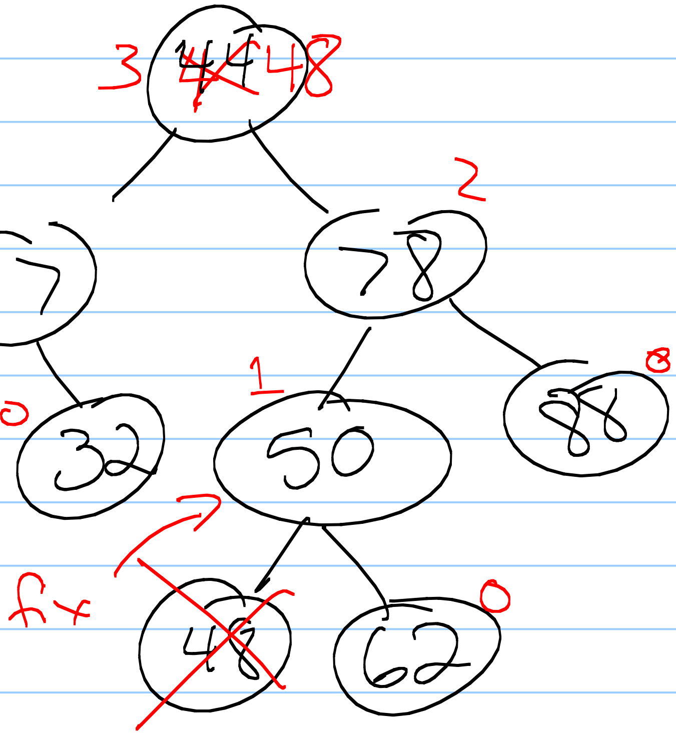
↳ start from actual deleted node

Note: Unlike insert, remove could actually unbalance all the way to the root.

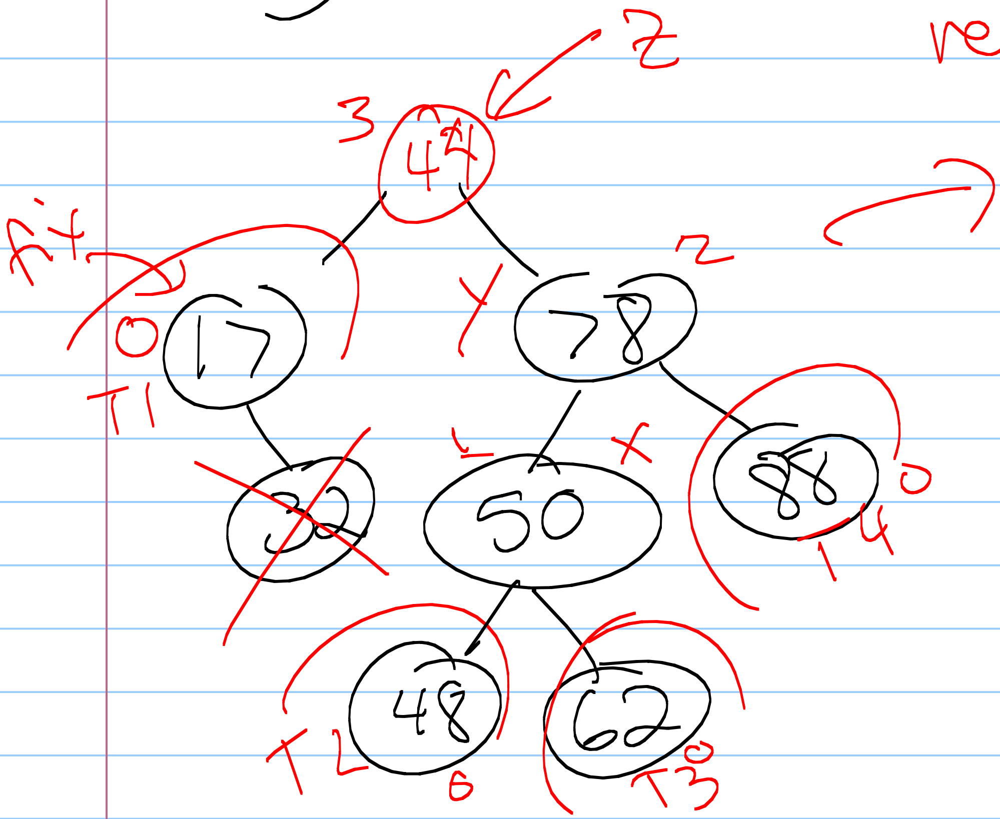
Example:

remove(44)

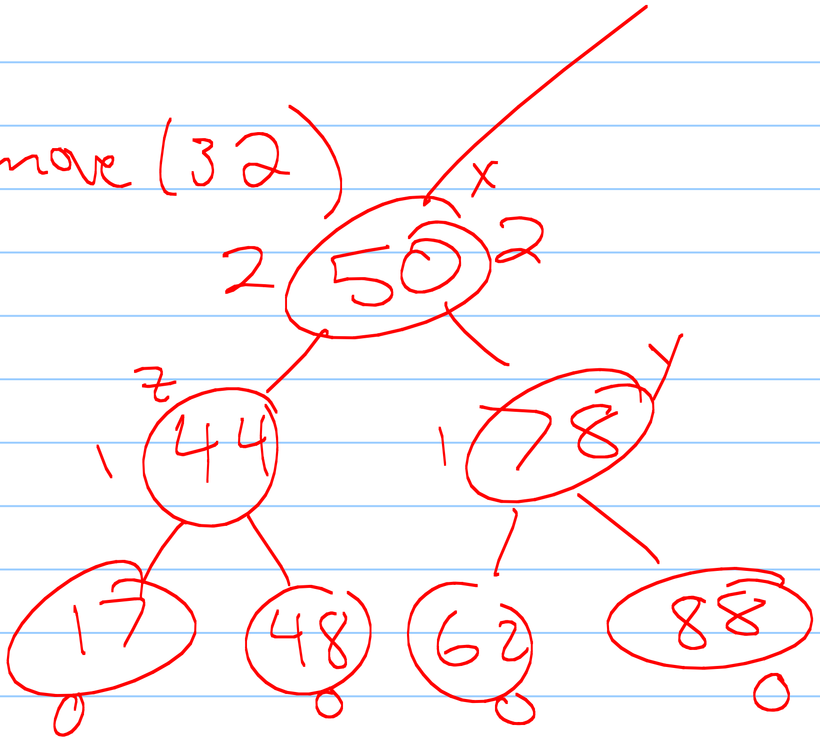
remove



Fixing the tree



remove (32)



Algorithm to remove

- Remove as in BST
- Track lower node that was removed.
- Travel up tree, searching for unbalanced nodes (+ fixing) until you reach the root.

Performance

For insert & delete, follow root to leaf path at most 3 times:

- find
- next in inorder (for remove)
- travel back up tree balancing

At each node:
- reset height
- ≤ 2 pivots
- reset heights again (if pivot)

at most 60 operations: $O(1)$

Total time: $O(\log n)$

Next HW:

Remove in an AVL tree.

Caution: Testing will be a significant portion of your grade!

(Lots of cases, since imbalances could propagate all the way up the tree.)

I'll post this, but it won't be due until a week after the exam.