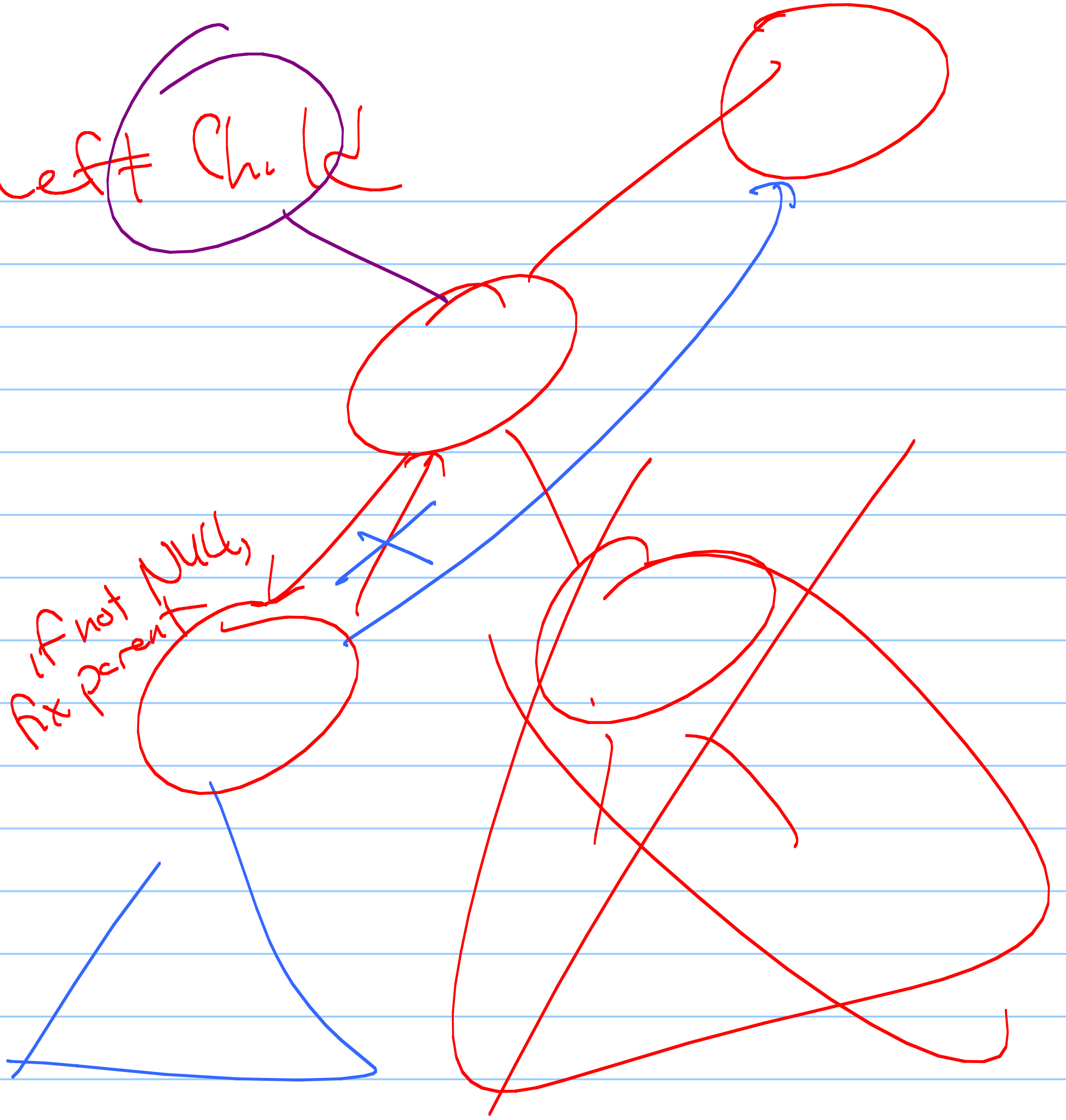
180 - AVL trees

Note Title 4/12/2011

Announcements

- Program is up (pair)
  - due next Monday
  - checkpoint Friday

Promote Left Child

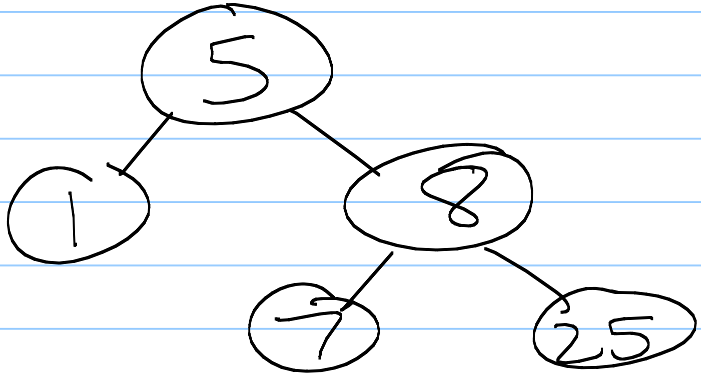


Recap: Search trees

What are they?

- "Sorted" trees: for every node v, left child is smaller & (right child) is greater.

⇒ Inorder traversal yields sorted list.



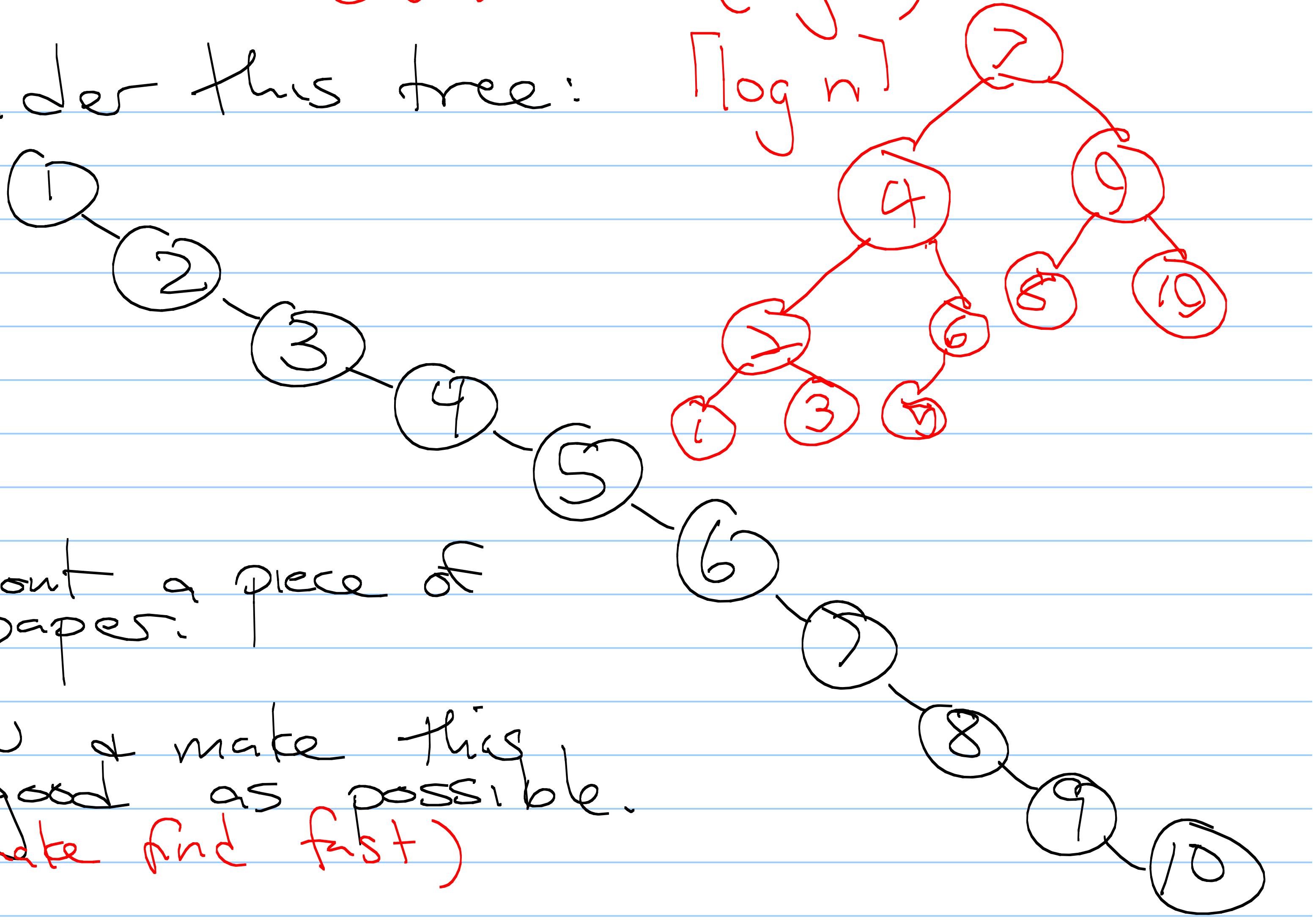
Runtimes

- insert: $O(n)$
- remove: $O(n)$
- find $O(n)$

These suck really - $O(n)$

$O(n) \rightarrow O(\log n)$

Consider this tree: $\lceil \log n \rceil$



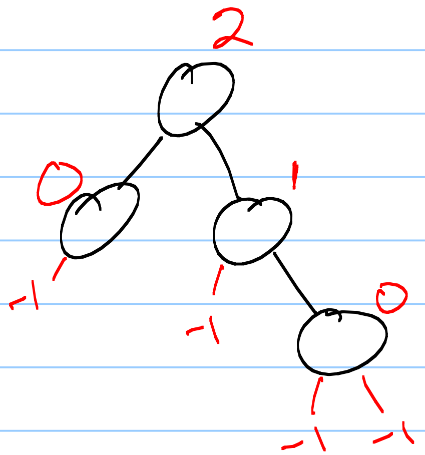
Take out a piece of paper.

Redraw & make this as good as possible.
(ie make find fast)

AVL Trees

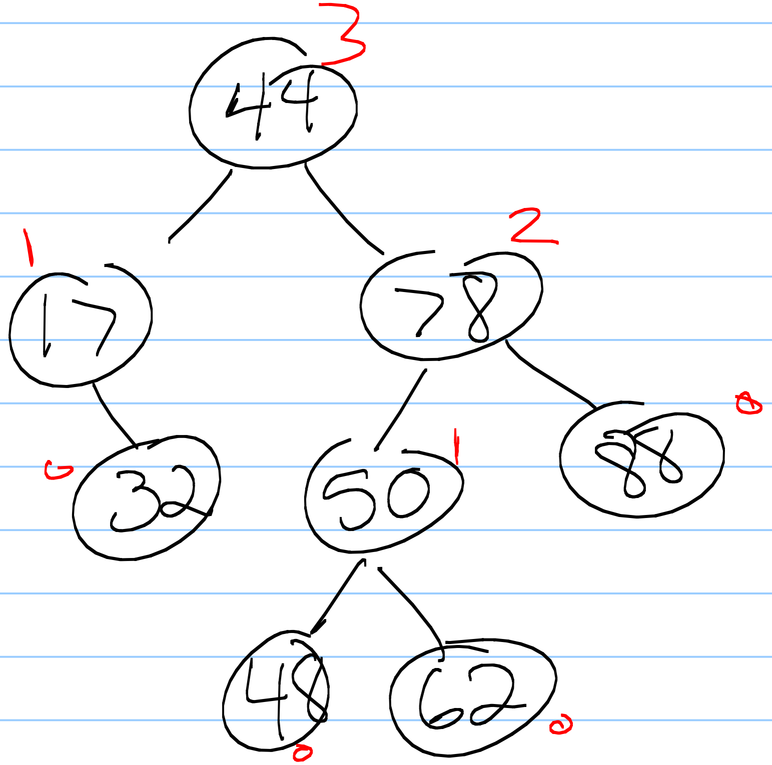
Height - Balance Property: for every node of T, the heights of the children differ by at most 1.

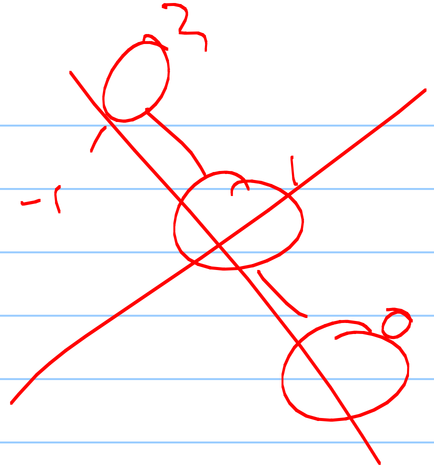
⇒ max height $\leq 2 \lceil \log_2 n \rceil$



(How do we calculate height again?)

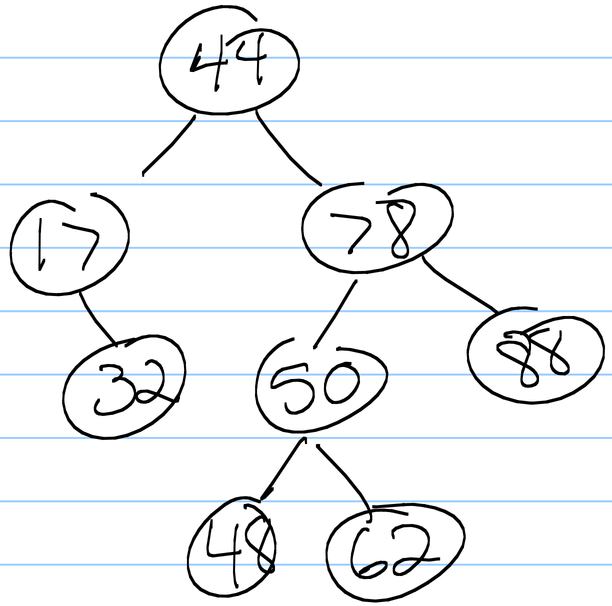
Ex:





Now: How can we mess this up?

(In other words, how can the height change?)

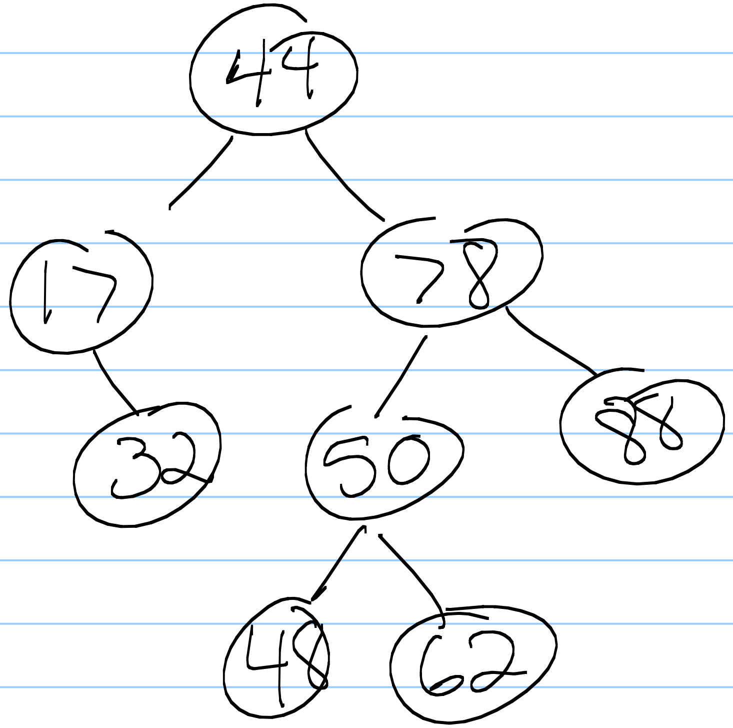


insert

remove

Insert:

insert(54)

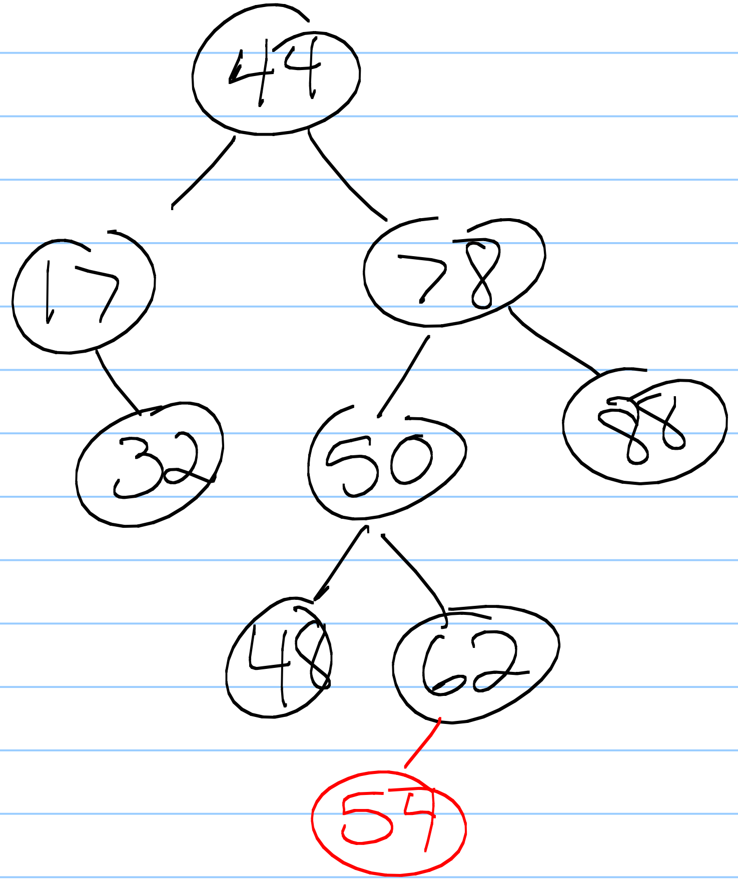


So: consider the lowest node which does not satisfy height-balance property — call this

Let be z's child with larger height.

Let be y's child with larger height.

Now — fix it!



What did you do?

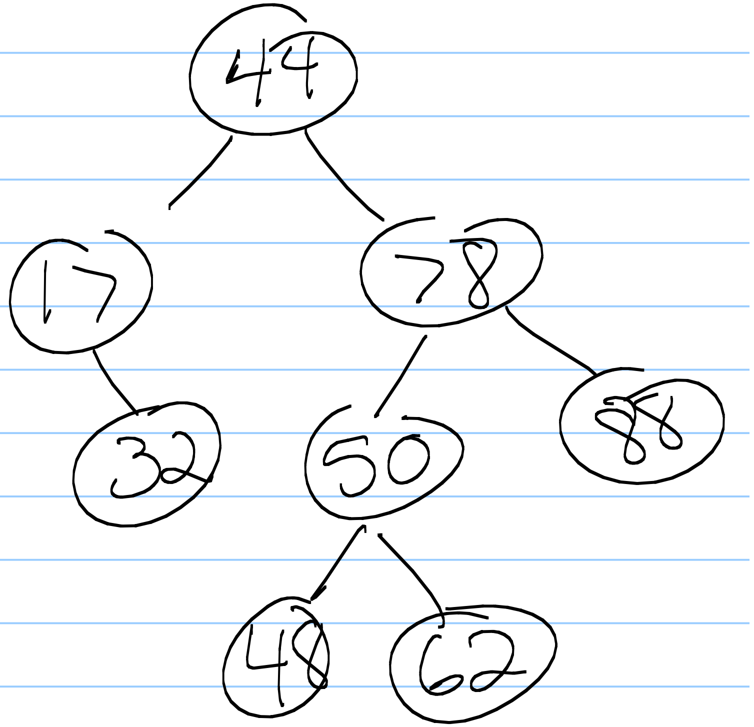
Another - insert (49)

So: consider the lowest node which does not satisfy height-balance property - call this

Let be z's child with larger height.

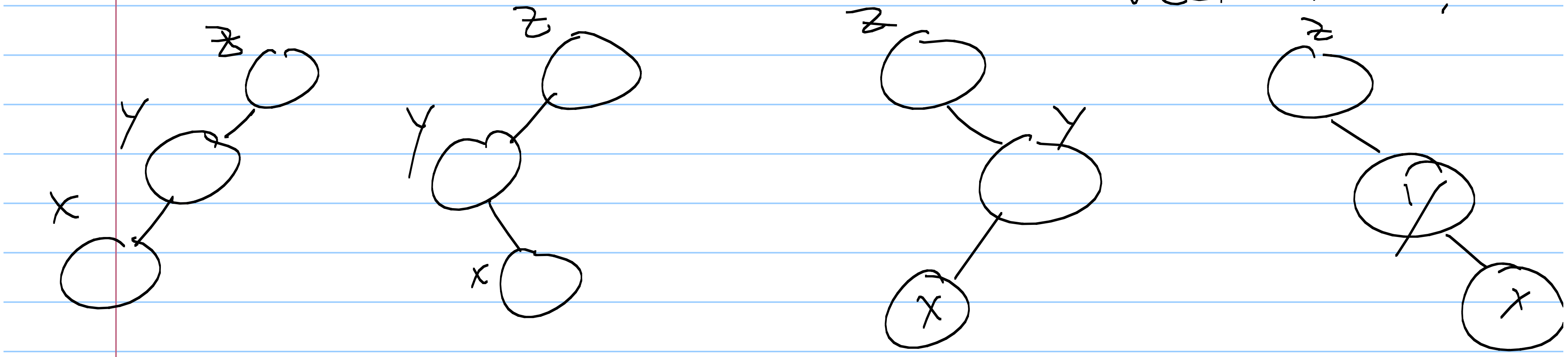
Let be y's child with larger height.

Now - fix it!



What did you do?

Generalize - Consider x, y, & z. How can we restructure?


z
y
x
z
y
x
z
y
x
z
y
x