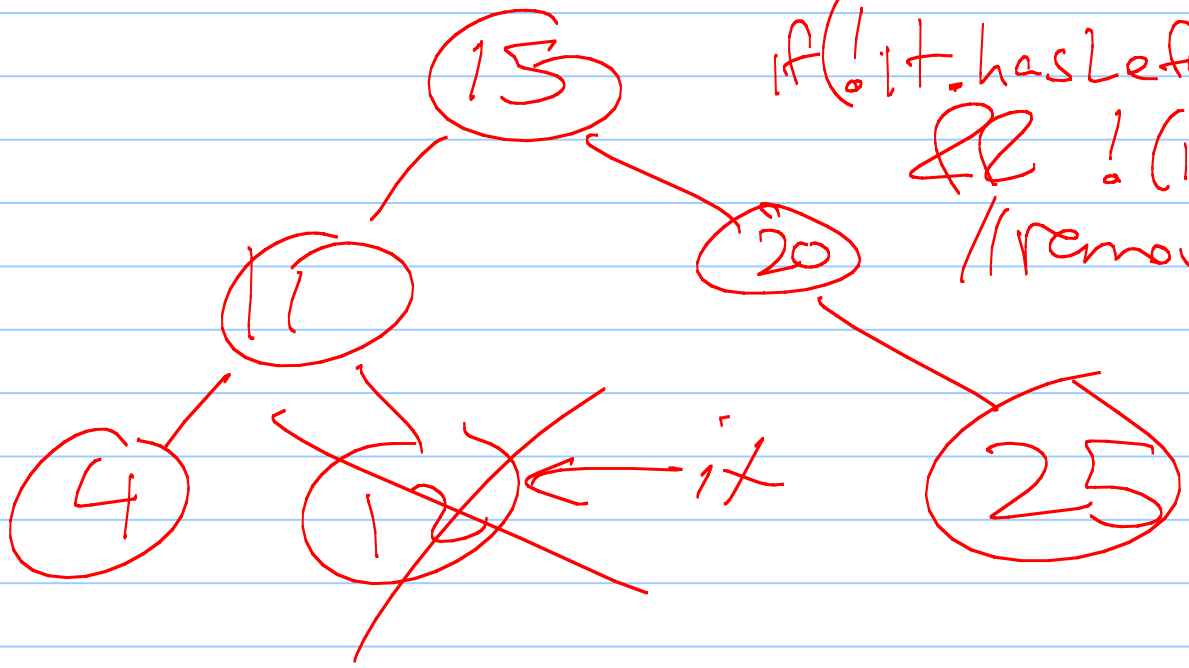
CS2100 - AVL Trees

Note Title 10/19/2012

Announcements

- scholarship deadline next week

Remove: cases

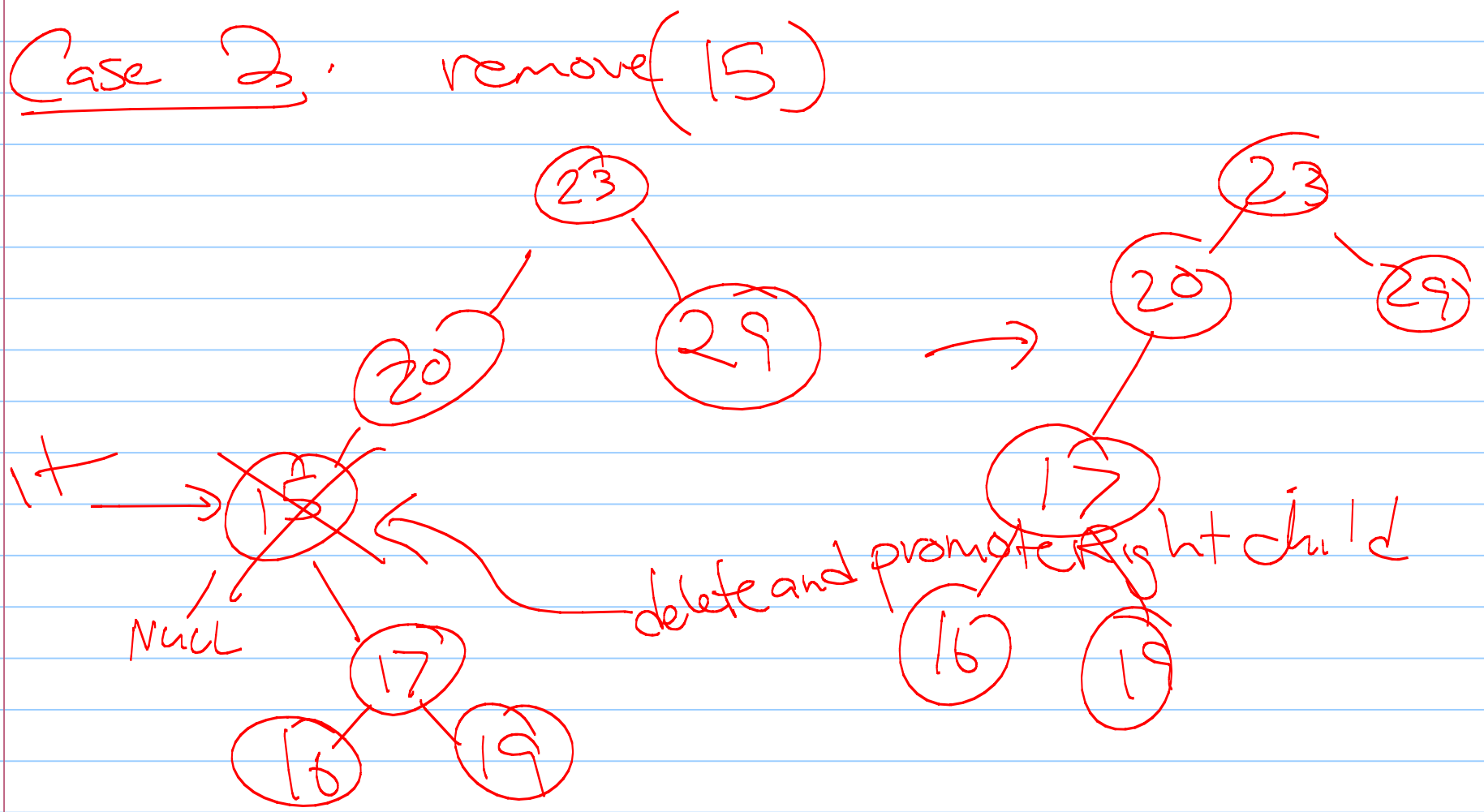


remove (12)

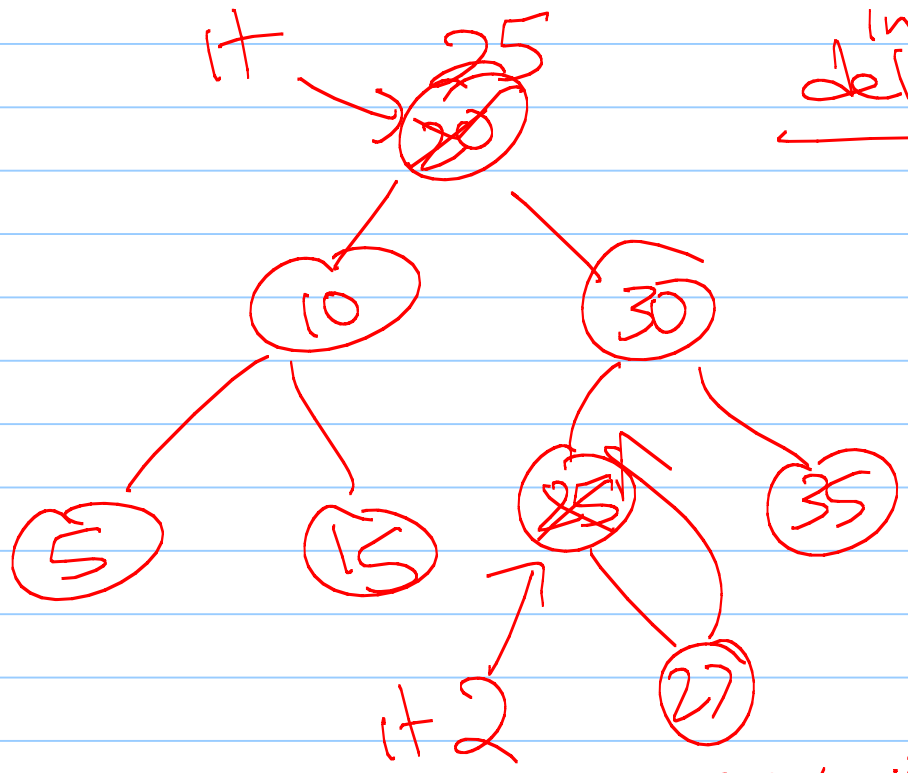
```
if(!it.hasLeftChild()
   && !(it.haveRightChild
   //remove node
```

remove a leaf - easy!

Case 3: remove(15)



Case 3: 2 children : find next node in an inorder traversal (in sorted order)

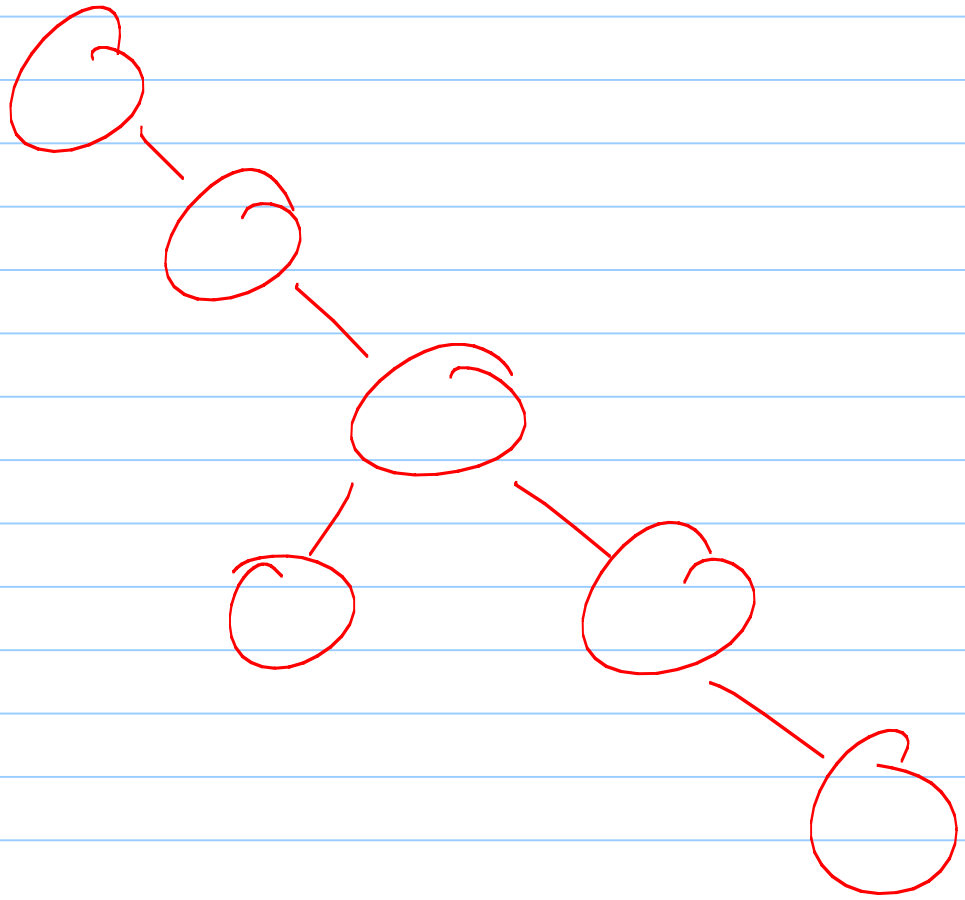


delete(20)

it2 = it++;

only has (at most) 1 child - on right

copy *it2 into *it & delete it2 promote right child of it2



Recap: BST

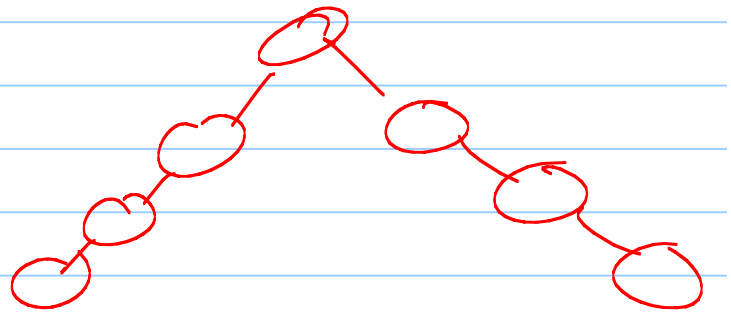
Runtimes:

insert
remove
find

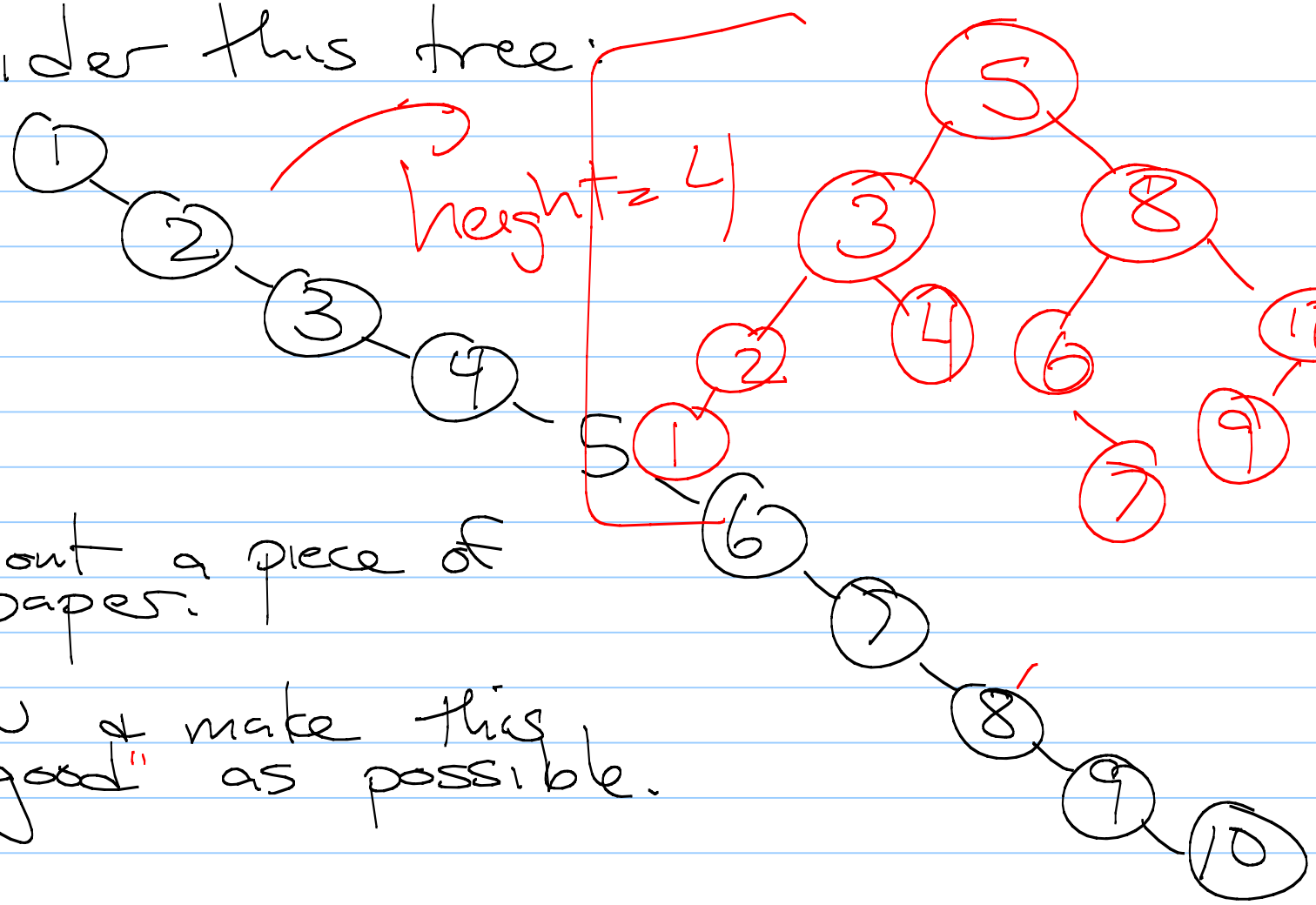
worst case, need to travel from root to some leaf

→ $O(\text{height})$

here: $O(n)$



Consider this tree:



Take out a piece of paper.

Redraw & make this as "good" as possible.

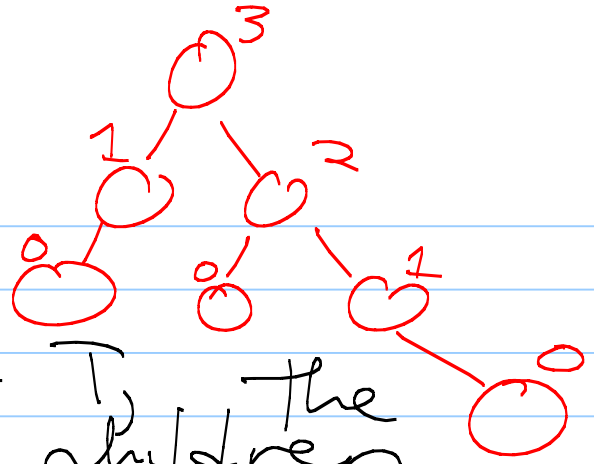
What did you do?

Balanced Binary Search Tree

- Red-black trees ← 1.2 log n
- Splay Trees
- AVL trees ← ~~1.44~~ log n
- :

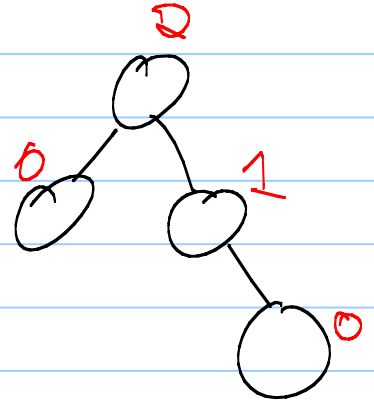
Goal of all: O(log n)

AVL Trees



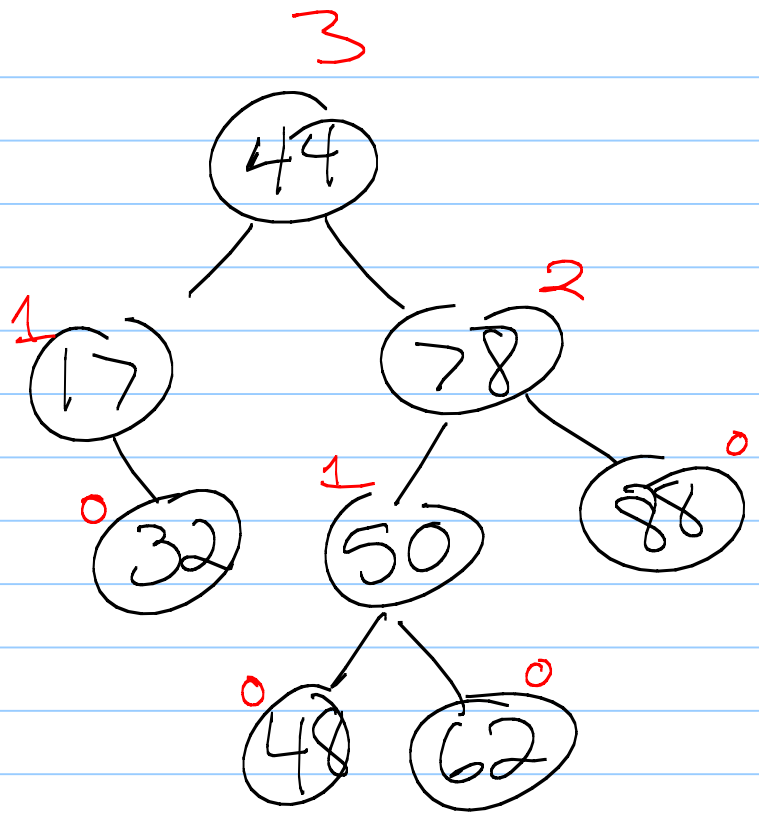
Height - Balance Property:
For every node of T, the heights of the children differ by at most 1.

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



(How do we calculate height again?)

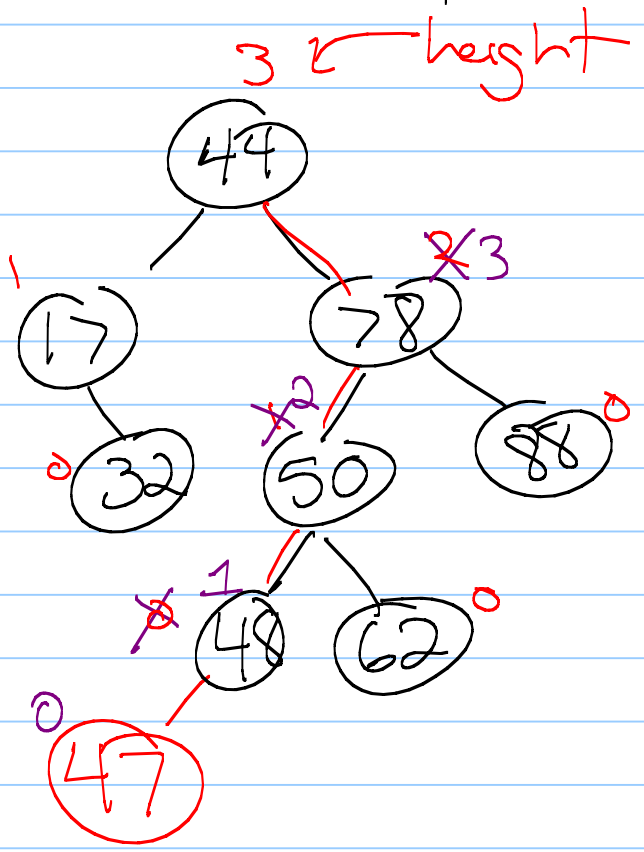
Ex:



Now: How can we mess this up?

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

Insert(47)

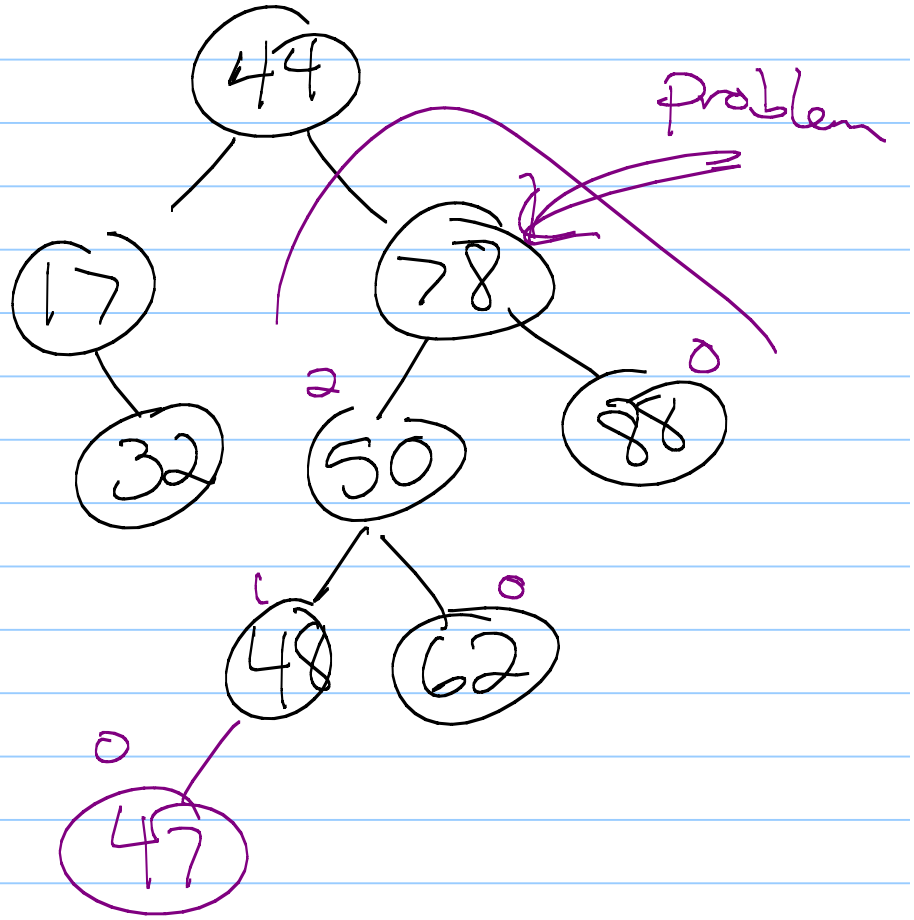


Insert:

insert(~~54~~ 47)

Goal: height-balance property

Fix it!

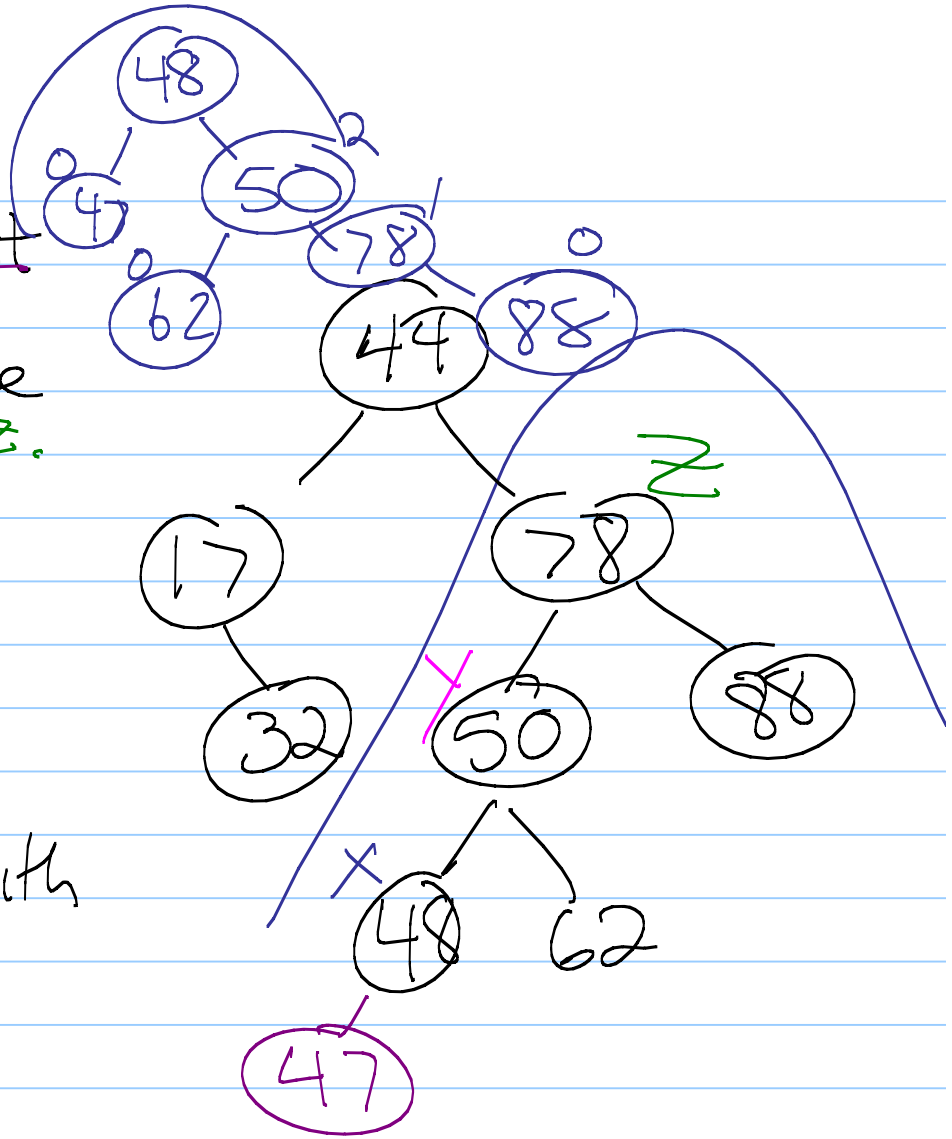


So: consider the lowest node which does not satisfy height-balance property -- call this z.

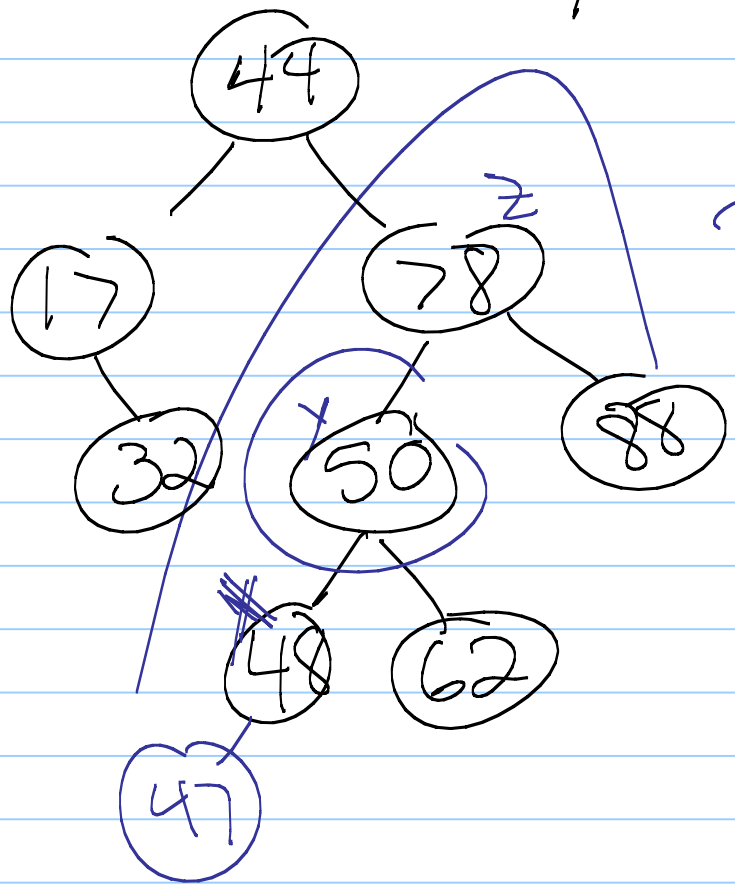
Let y be z's child with larger height.

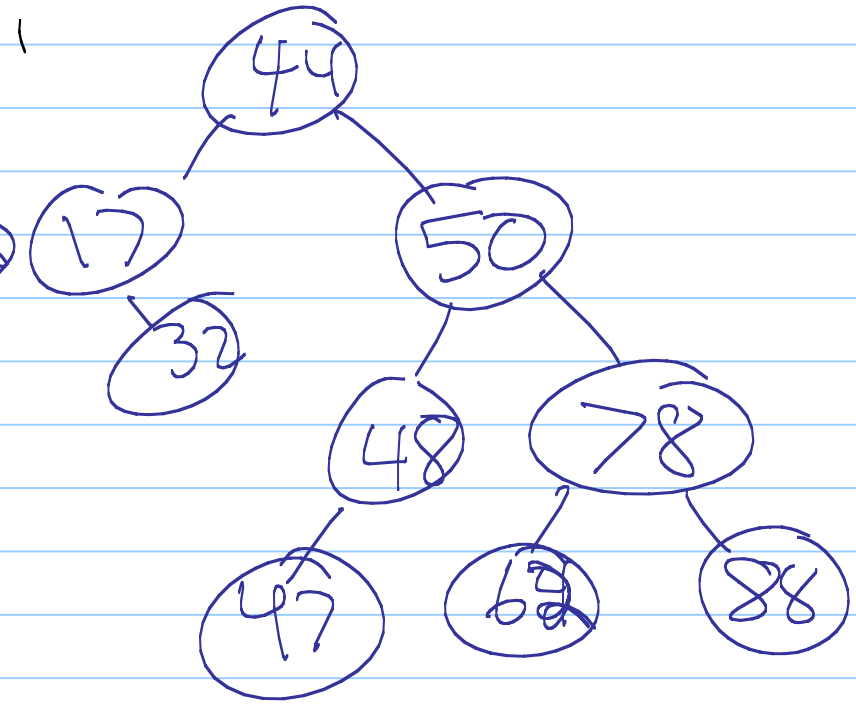
Let x be y's child with larger height.

Now - fix it!



What did you do?





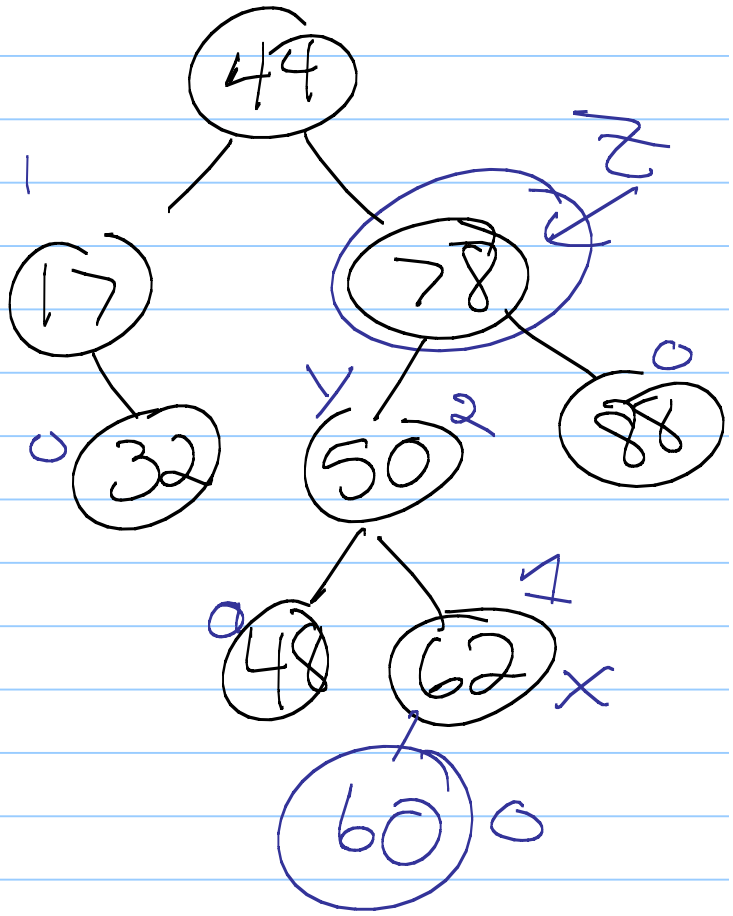
Another - insert 60

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

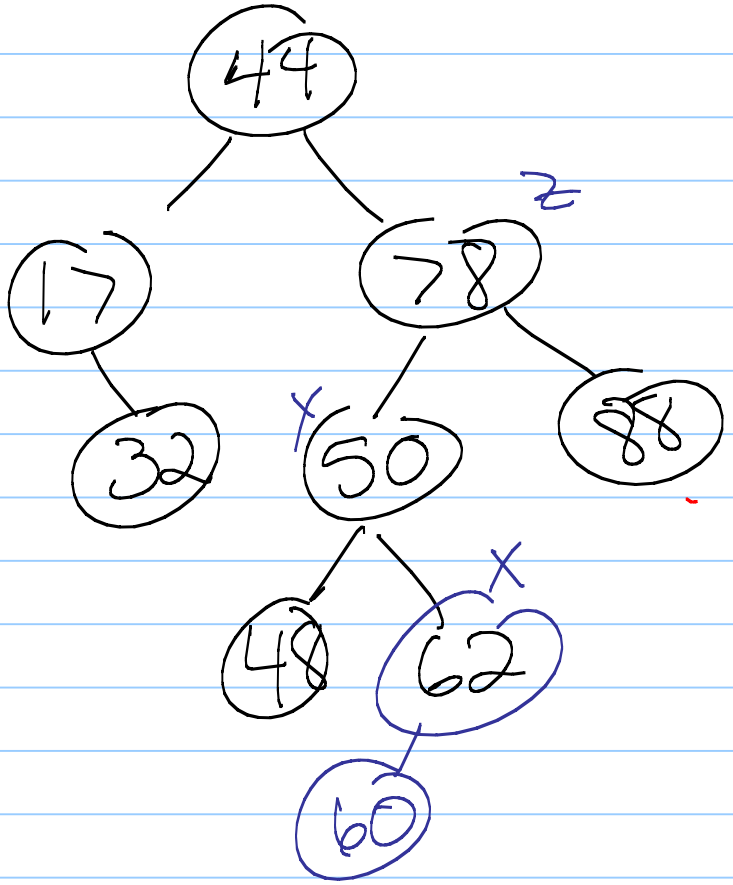
Let y be z's child with larger height.

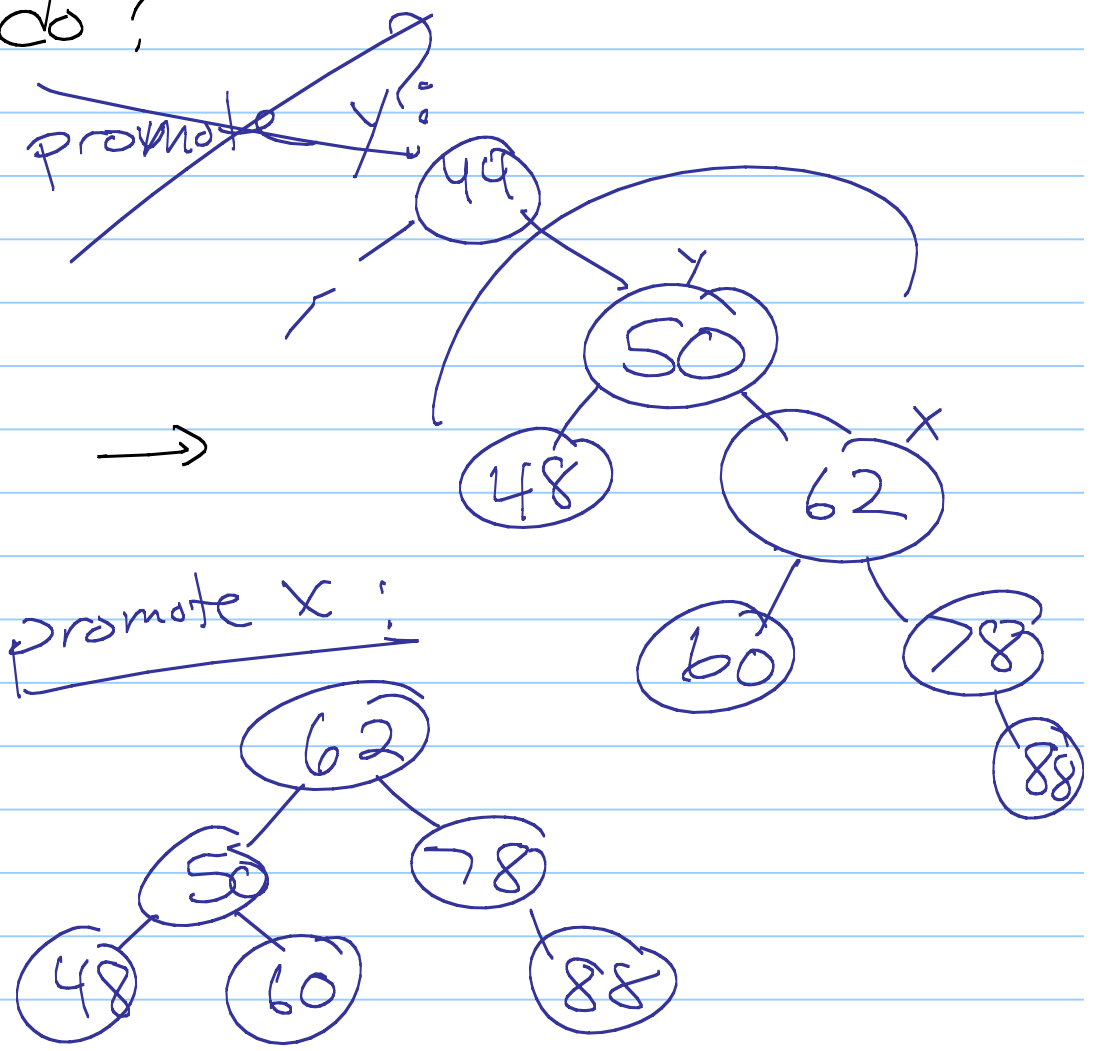
Let x be y's child with larger height.

Now - fix it!



What did you do?

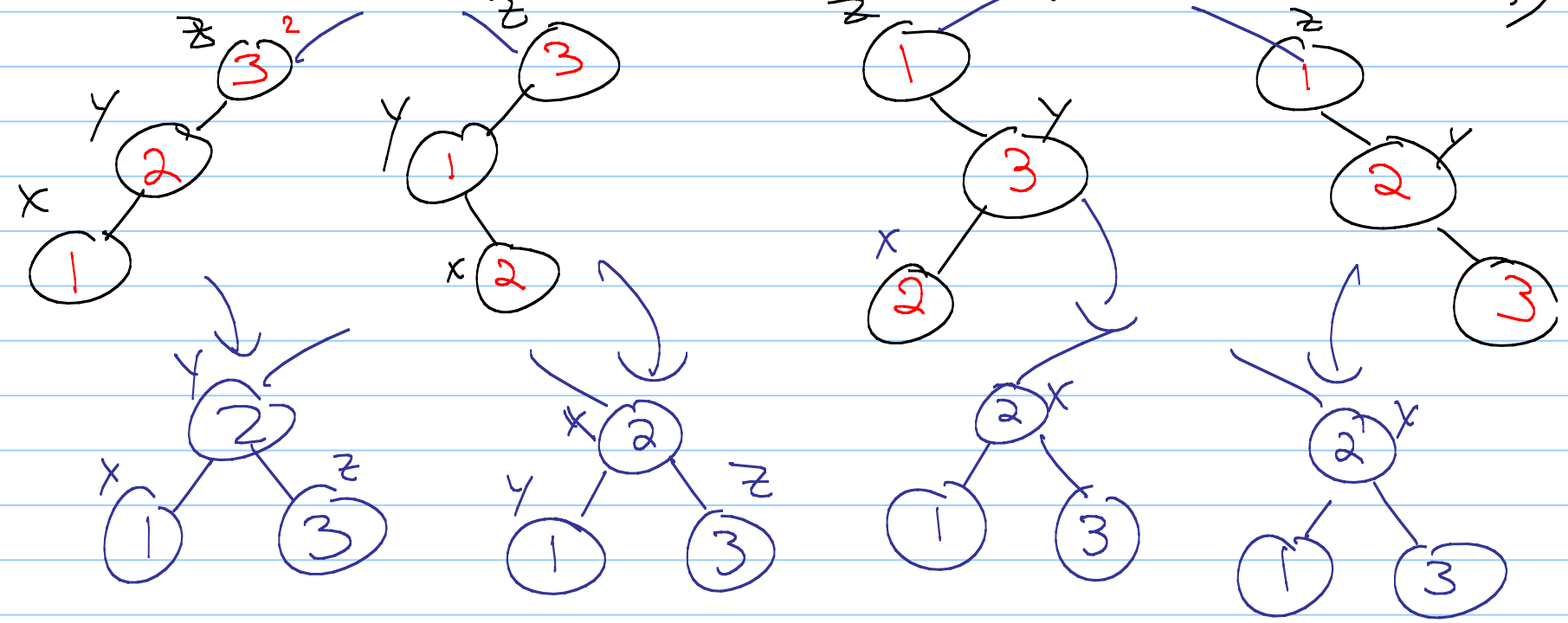




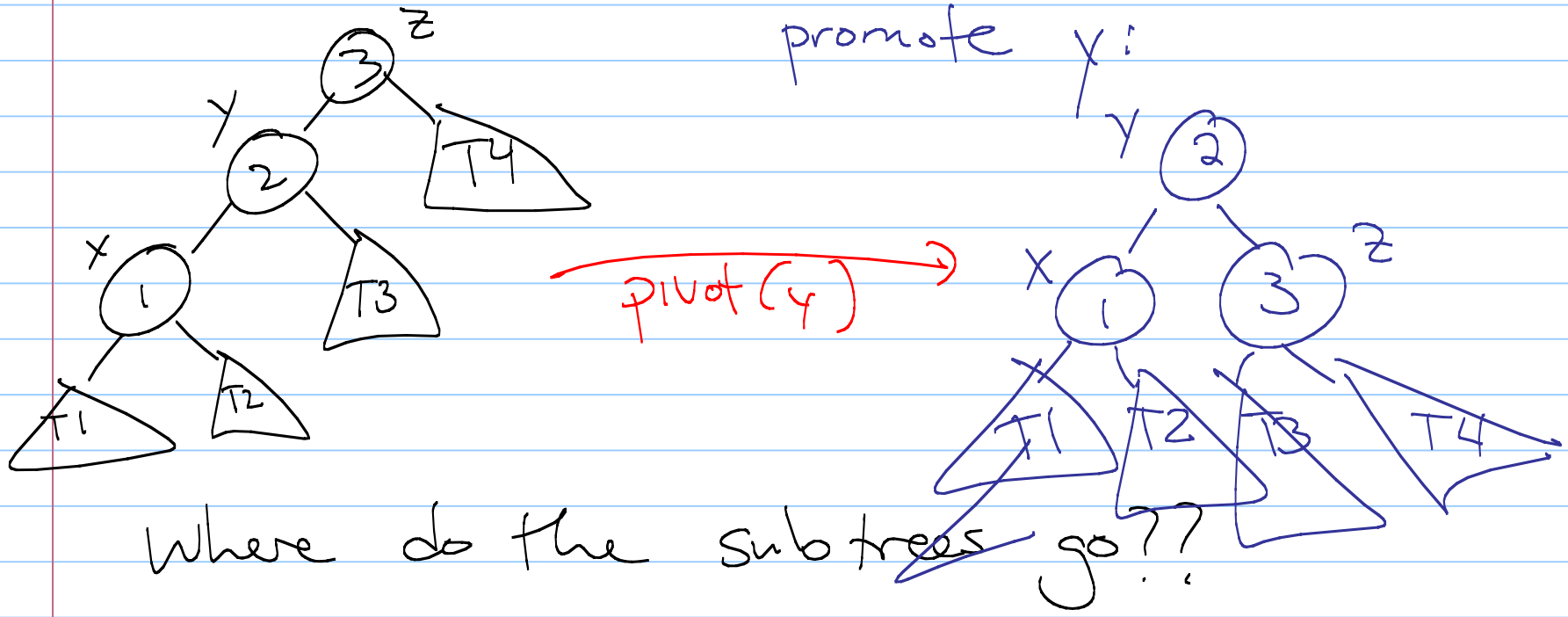
pick middle element

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

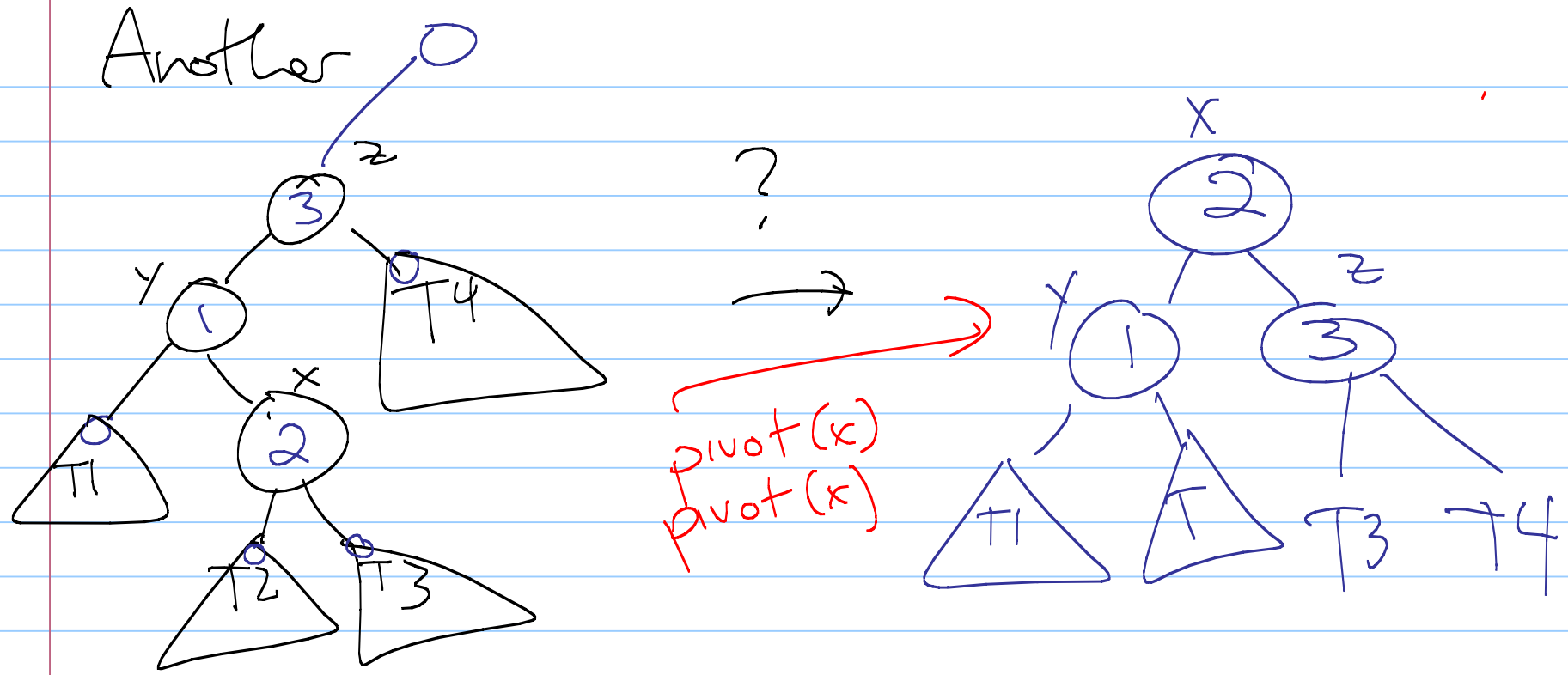
(Hint: What is inorder traversal of these in each case?)



Actual picture:



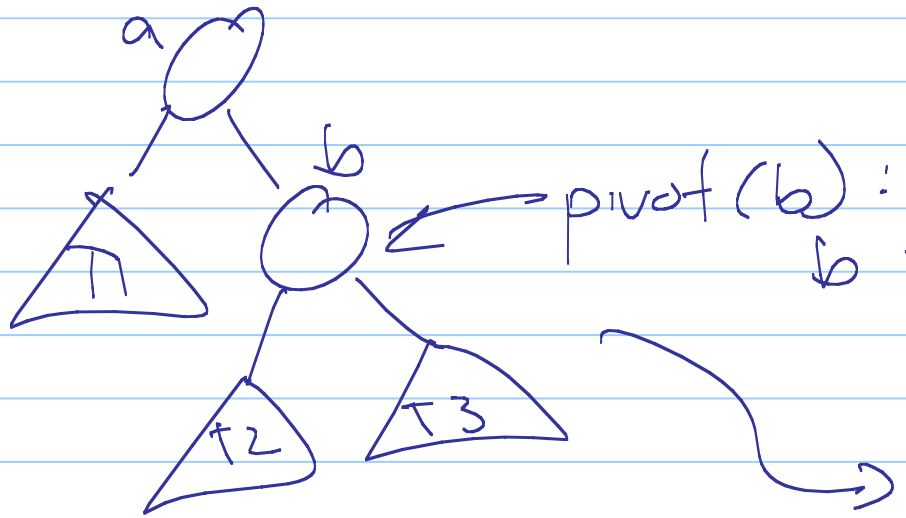
Where do the subtrees go??


Another
z
3
y
1
T4
x
2
T1
T2
T3
?
pivot(x)
pivot(x)
X
2
y
1
z
3
T1
T
T3
T4

Any way you do this, "2" becomes the root of the new subtree, with "1" to the left & "3" to the right!

What about T1, T2, T3, & T4?

Key operation: Pivot



pivot(b): exchange b ↔ its parent