


CS2100

AVL trees:

balancing



Recap:

- Update makefile:

> make BST

for HW9: make AVL

(assuming you use same names!)

- BST HW: due Tuesday at
11:59pm

(won't extend past Wednesday)

- Next Tuesday: lecture

Balanced BSTs

Many kinds:

- Red-black trees : $1.4 \lceil \log_2 n \rceil$

- Splay trees :

- AVL trees : $2 \lceil \log_2 n \rceil$

⋮

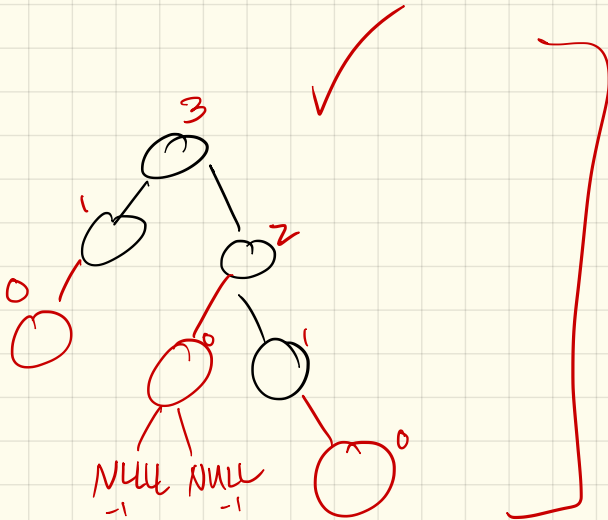
Goal of them all :

$O(\log_2 n)$

AVL trees :

Height balance property :

For every node x in T ,
the heights of its children differ
at most 1.

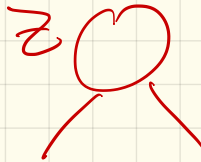
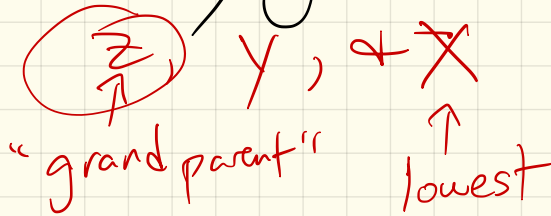


$$\Rightarrow \text{max height} \leq 2 \lceil \log_2 n \rceil$$

Note: NULL is considered -1

So: algorithm to insert:

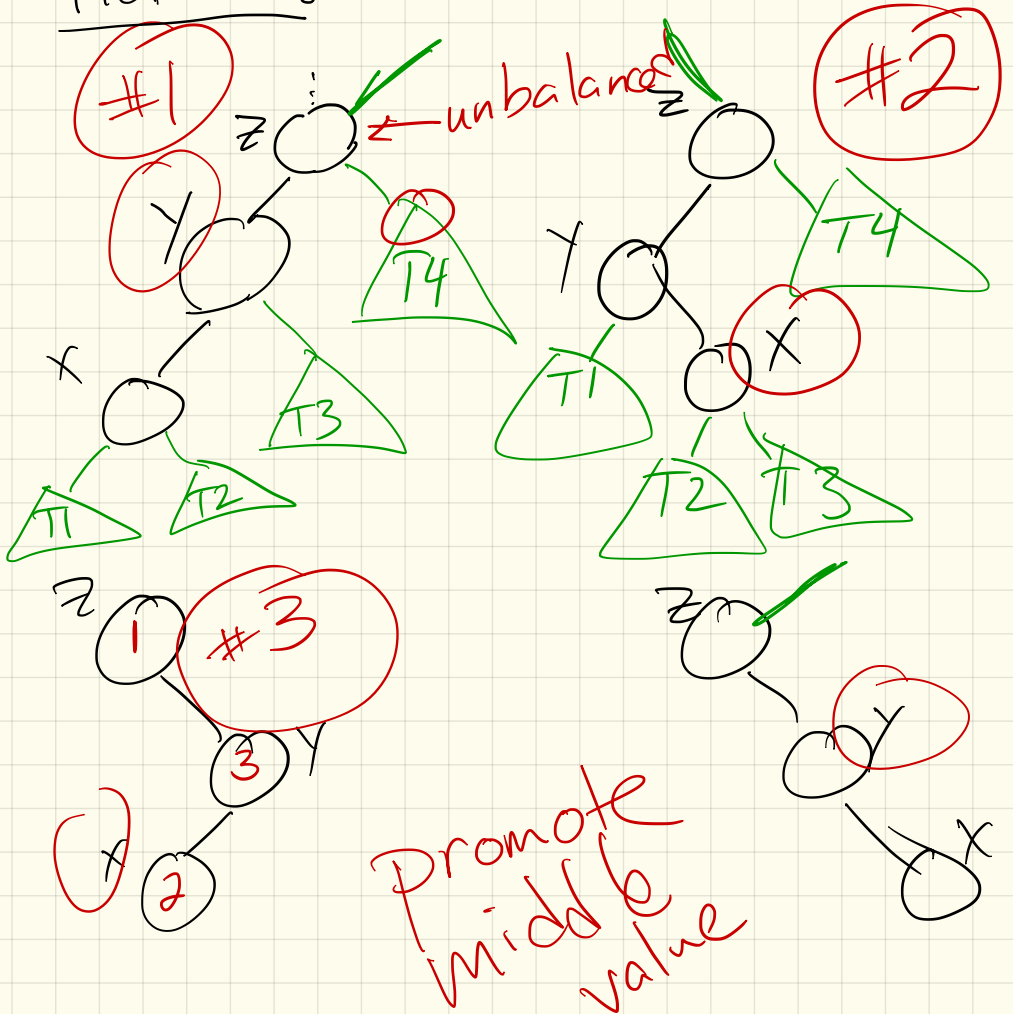
- First run BST insert
- Then find lowest unbalanced node Z , \uparrow deeper child / grand child.



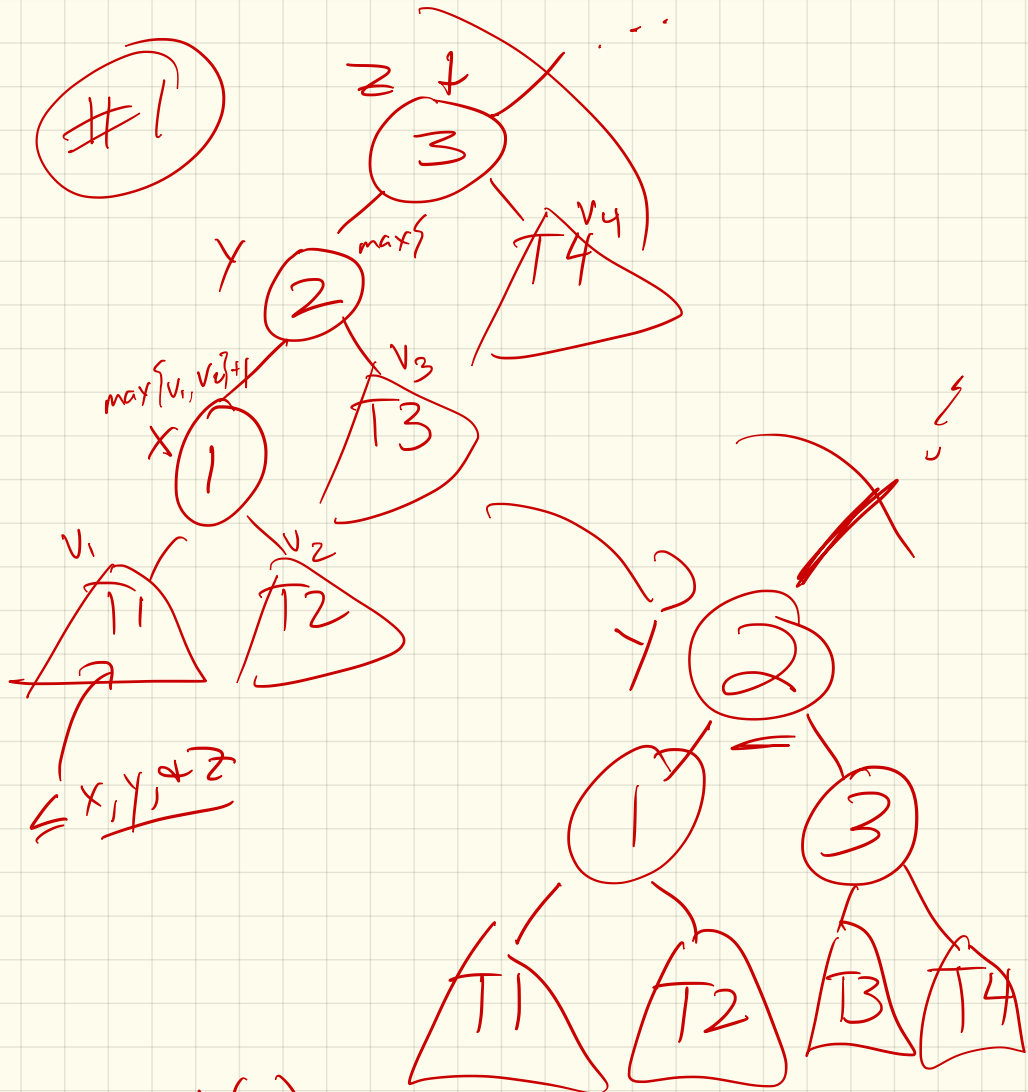
diff in height of left + right ≥ 2

- balance

Pictures:

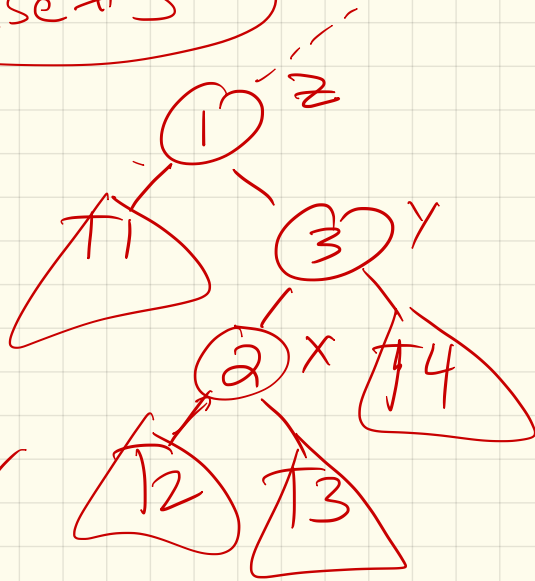


#1

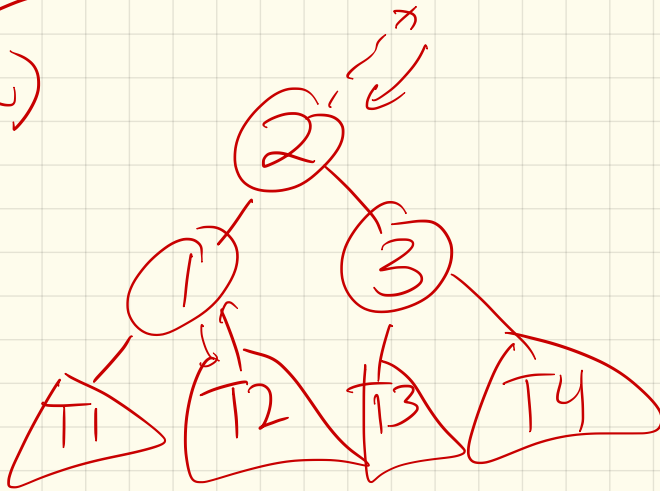


pivot(y)

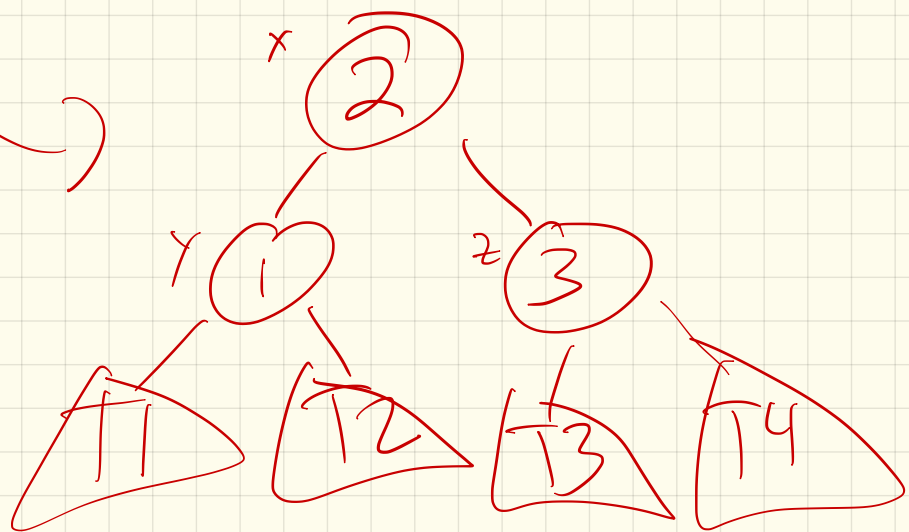
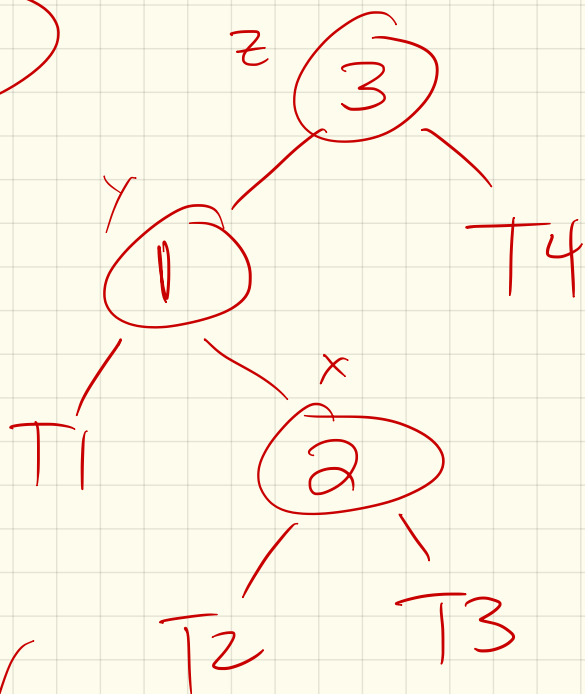
Case #3



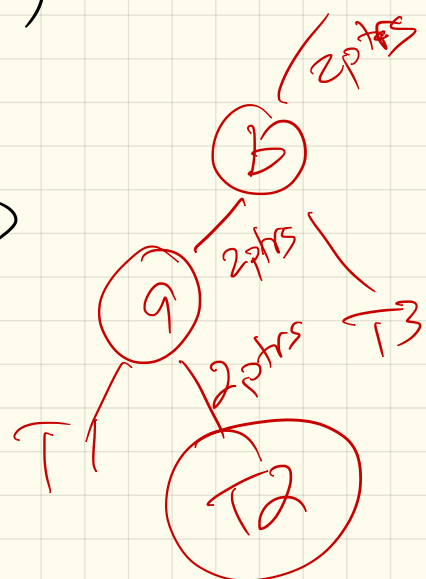
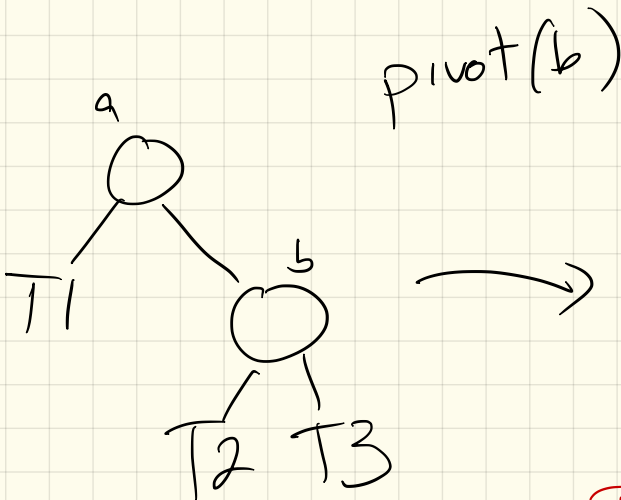
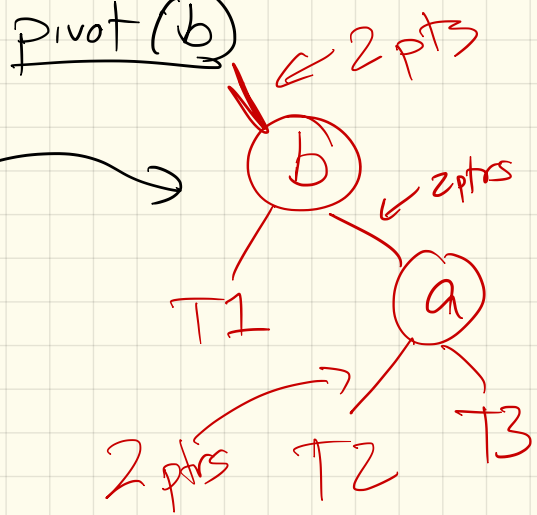
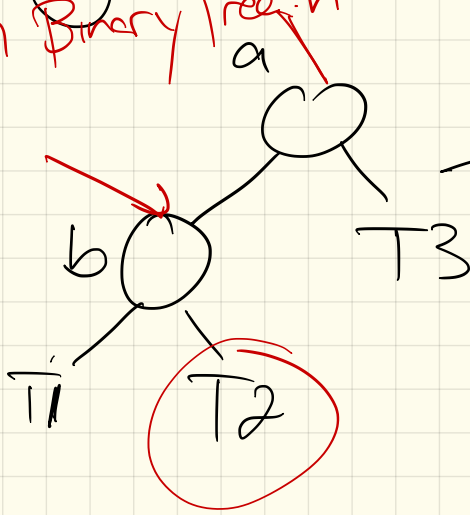
fix : make x the root



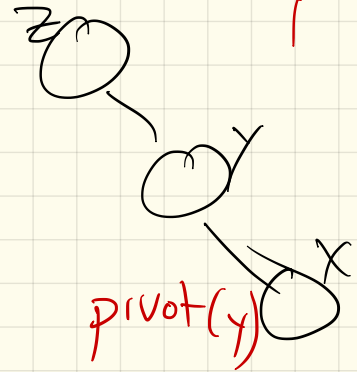
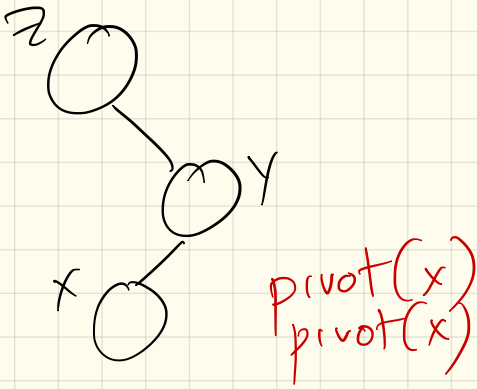
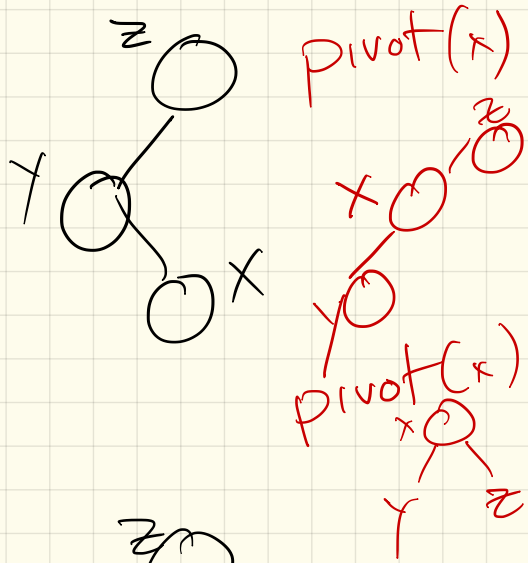
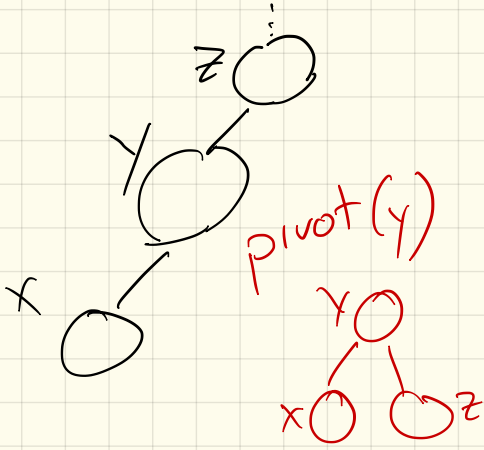
#2



Key operation: pivot (iterator)
 In Binary Tree: h



Then: implement w/ pivot!
 In AVL: balance



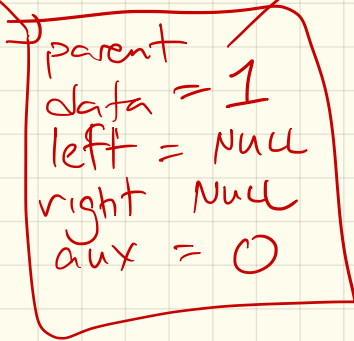
Bigger example:

Insert : 1, 2, 3, 4, 5, 6, 7

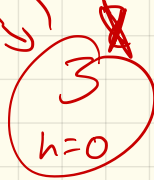
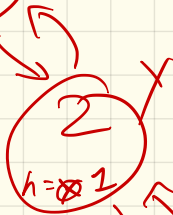
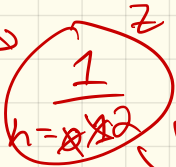
AVL:

root

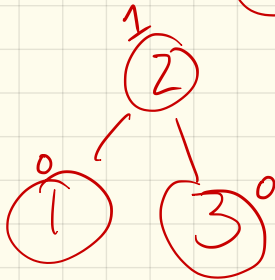
NULL

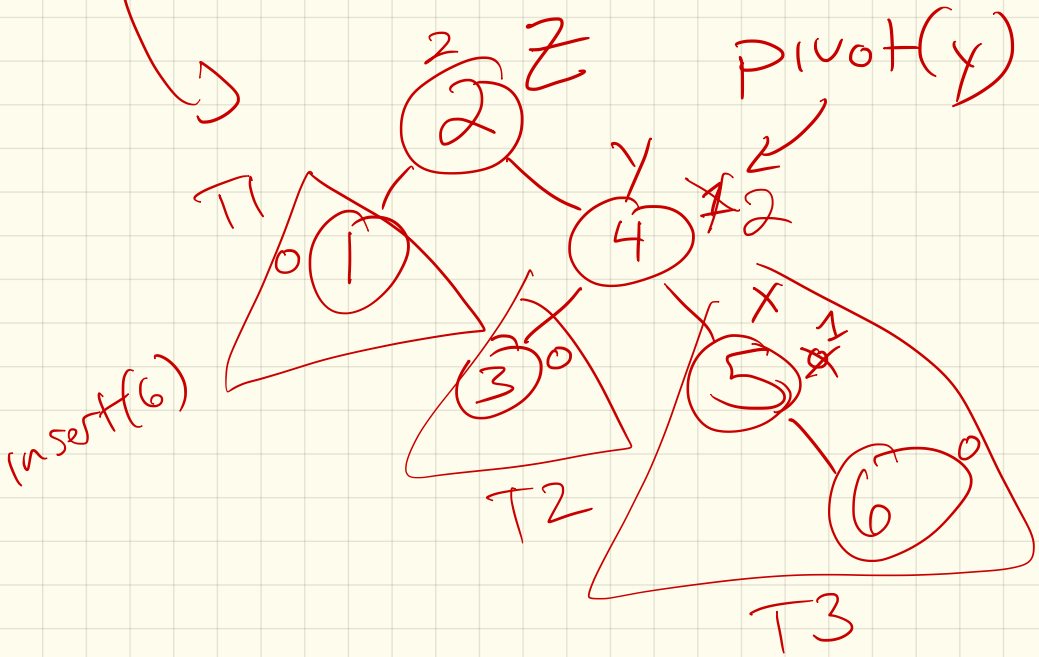
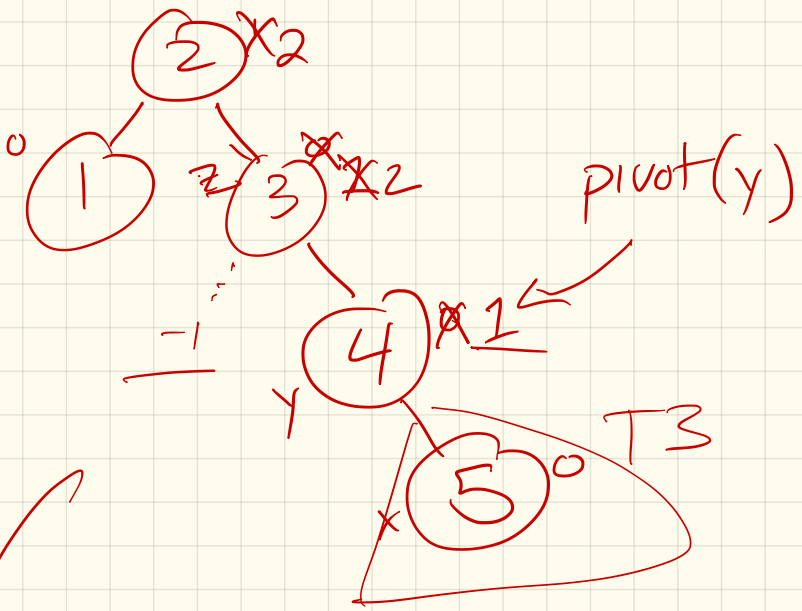


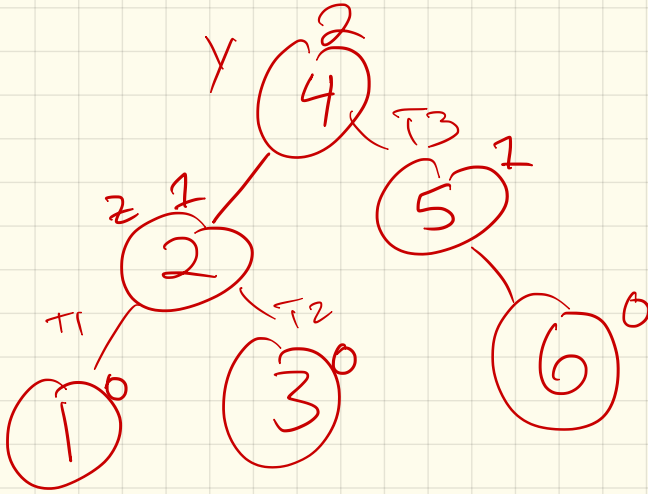
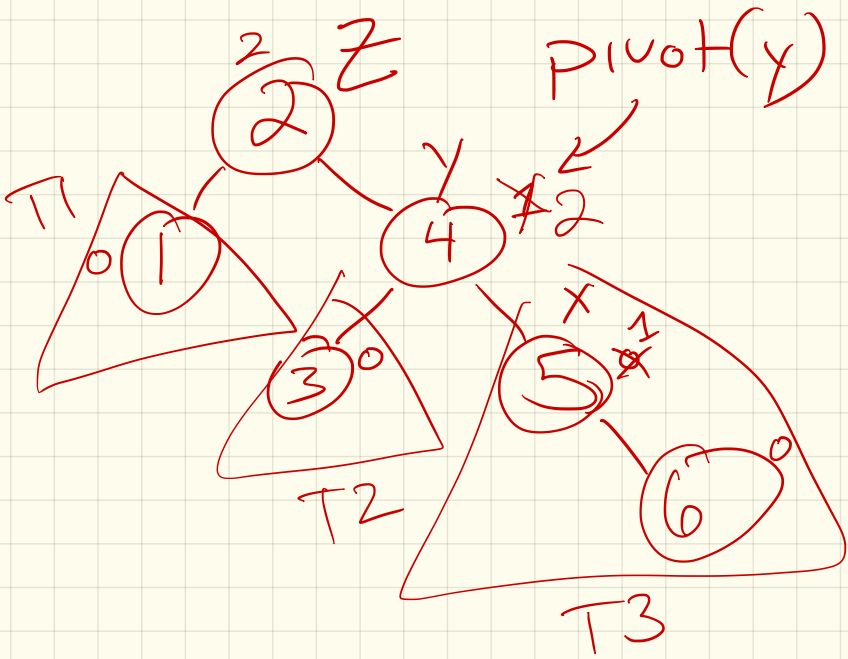
root



pivot (Y)







Remove:

- Do binary search tree remove

Then:

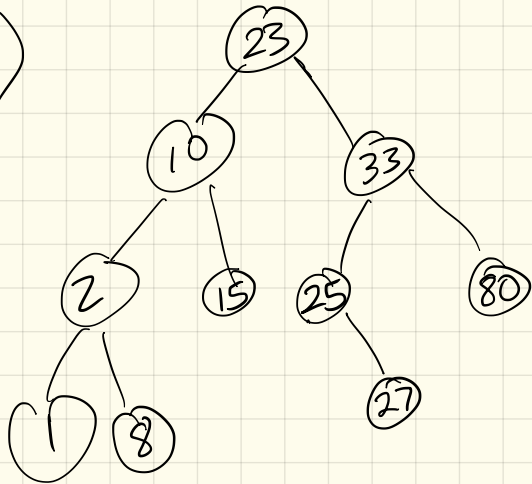
- Start from deleted node's position

travel up

reset heights

rebalance if needed

Ex: remove (23)



Next things:

- Code!

- in BinaryTree.h

- ~~+~~ in AVLtree.h

- Go over remove