How are AVL Trees constructed?

AVL TREE INVARIANTS:

+ Search Tree Invariant
+ Height invariant.

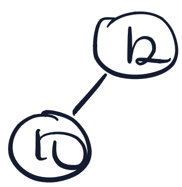
Q2 from tutes

"Show how an AVL tree would be constructed if the following values were inserted in order"

1.


height = 1

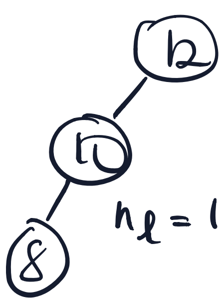
2.




$|h_l - h_r| > 1$ ?

$h_l = 1$ $h_r = 0$

3.

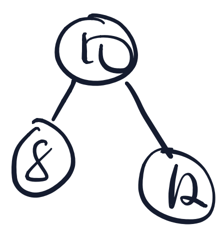



$h_l = 2$ $h_r = 0$

$h_l = 1$ $h_r = 0$ ✓

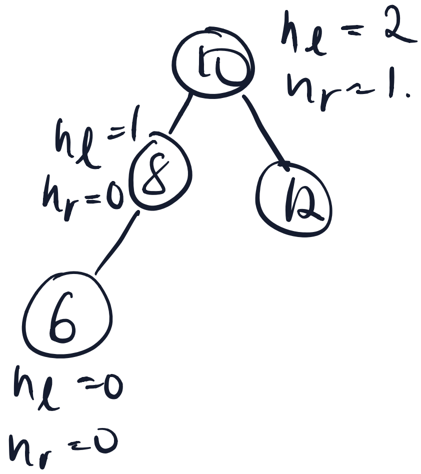
$h_l = 0$ $h_r = 0$ ✓

rotate → right. (counter leftheavy)



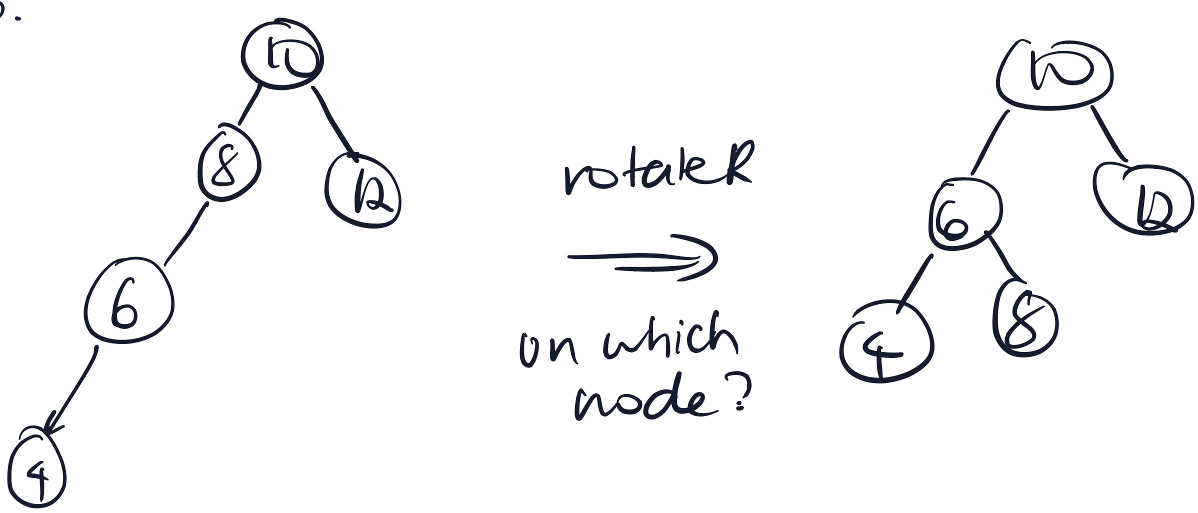

We only ever do one rotate and then its a balanced tree, why is that?

4.




BALANCED.

5.



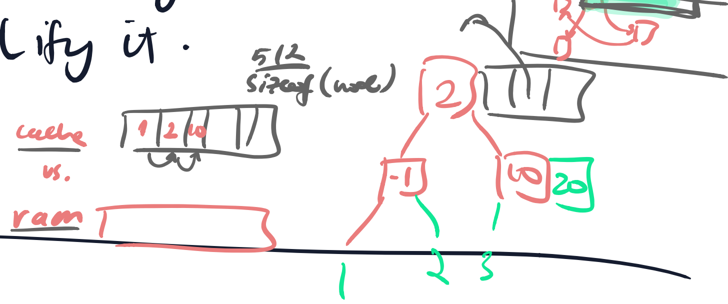

6. 2 balanced.

We only ever do a maximum of one rotate / node, and then we move on. How can we guarantee that it's balanced at the end.

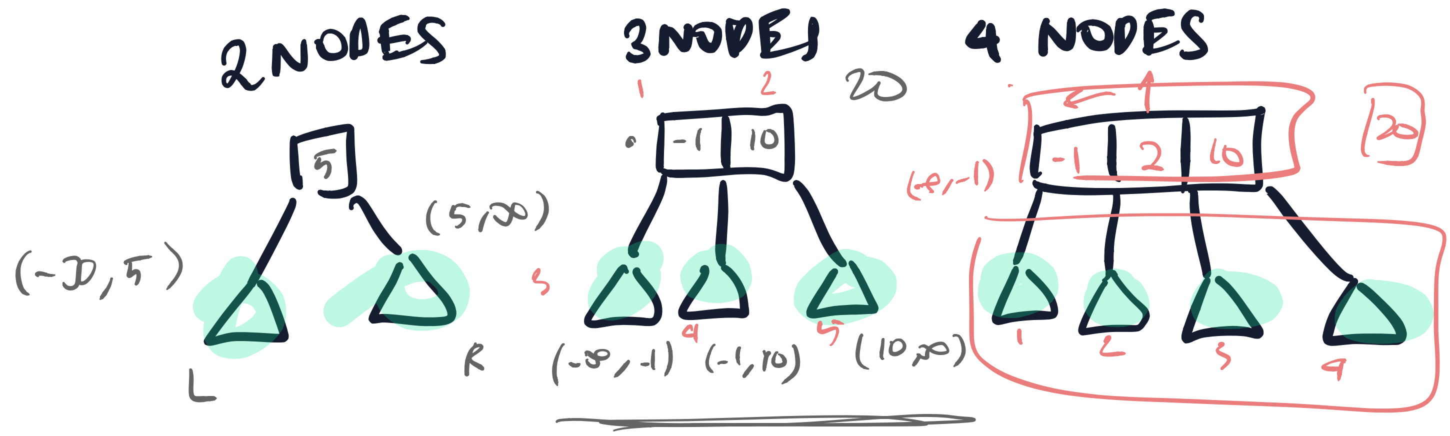
Develop an algorithm that checks if a tree is an AVL tree.

> what does it mean to be an AVL tree?

Say we have a BST that may or may not be an AVL tree. Develop an algorithm to AVLify it.




# 2-3-4 TREES.




Invariants:

- Every non-leaf node is a 2/3 node
- All leaves are on the same level.

Why? Checkout insert.

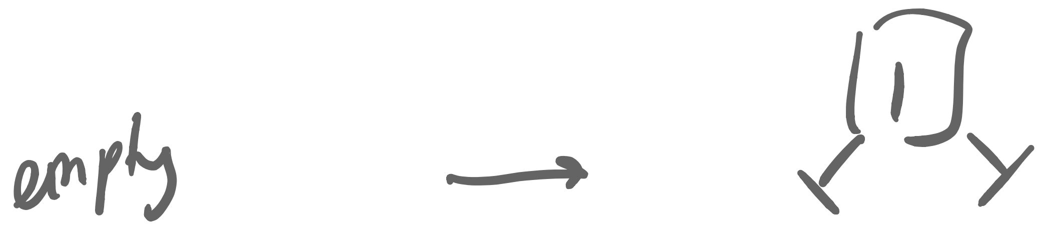
(1) Find the position that you'd like to insert the value x.

(2) If we have overflow remove original middle node and push up. Insert our element as a leaf.

(3) Repeat overflow check for parent, doing steps (2-3) on parent.

If root, the median node is the new root.

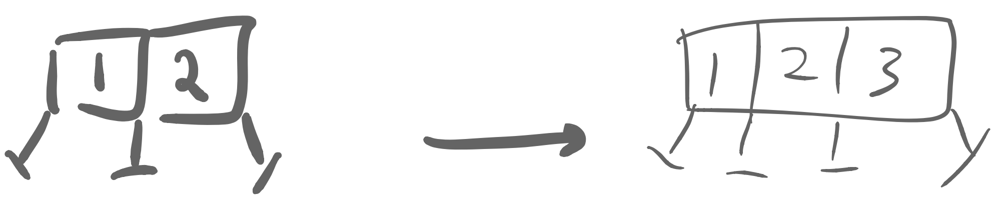
---

1. 


---

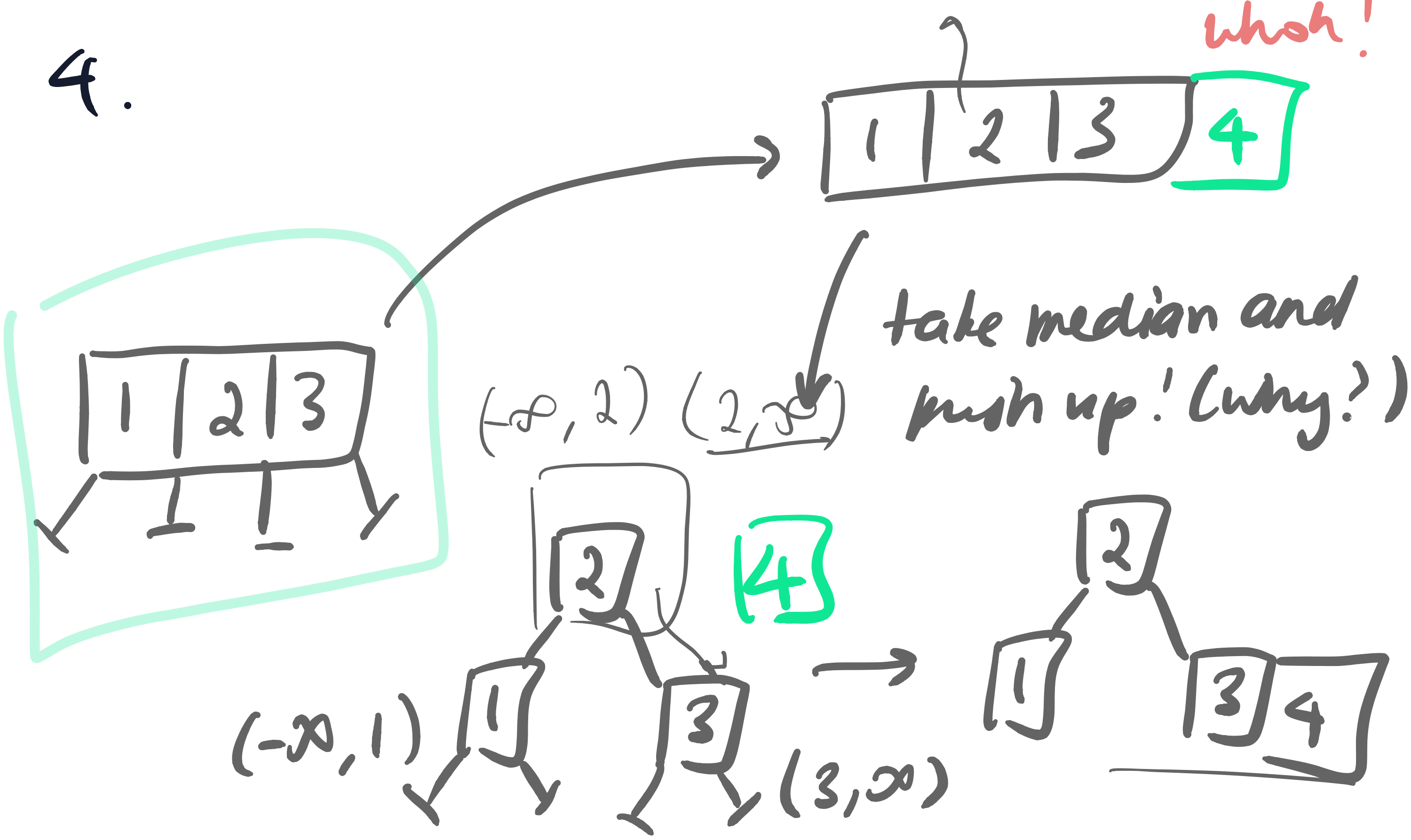
2. 


---

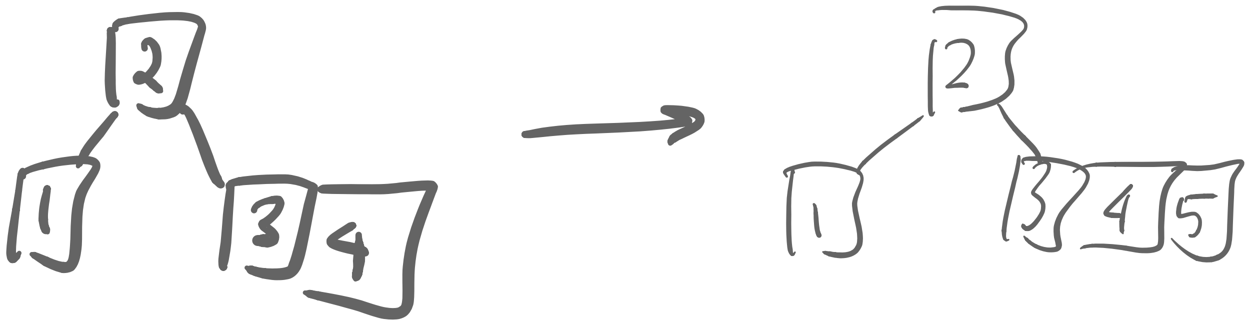
3. 


---

4.


whoa!
1 | 2 | 3 | 4
1 | 2 | 3
take median and push up! (why?)
(-∞, 2) (2, ∞)
2
4
(-∞, 1)
1
3
(3, ∞)
2
1
3 | 4

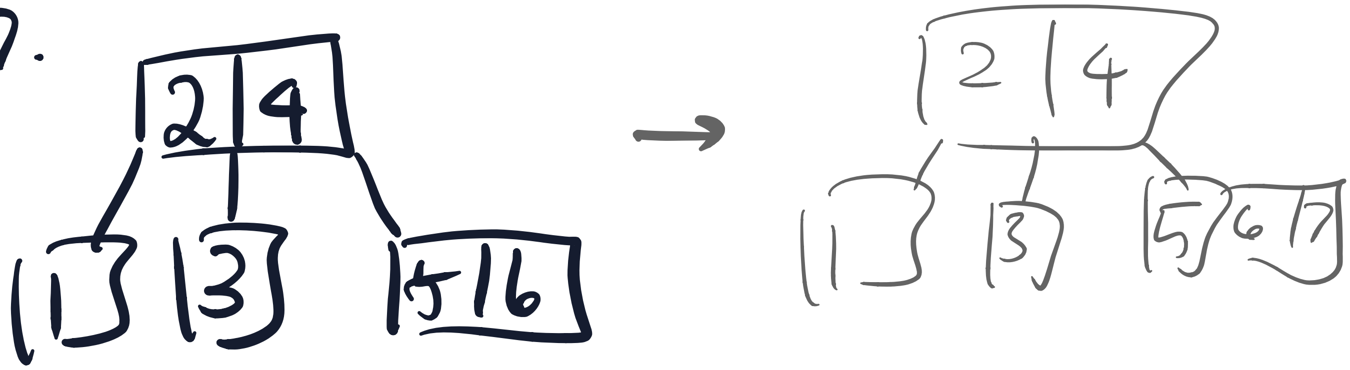
5.


2
1
3 | 4
2
1
3 | 4 | 5

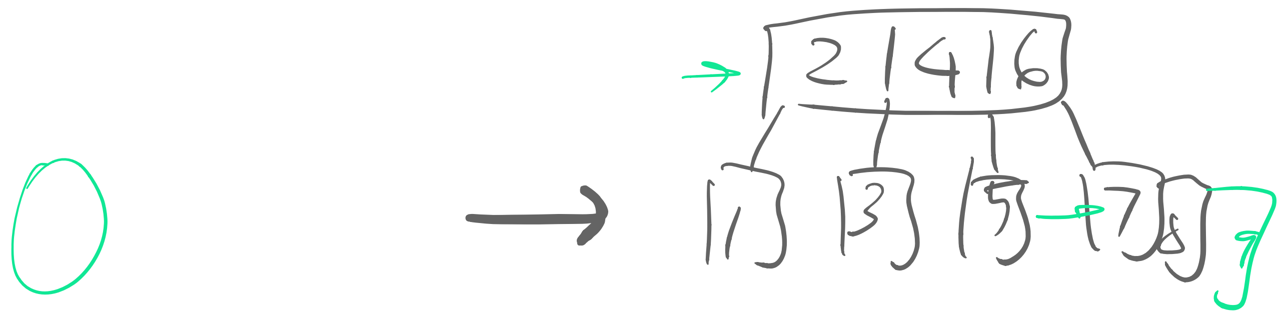
6.


2
1
3 | 4 | 5
6
2 | 4
1
3
5 | 6

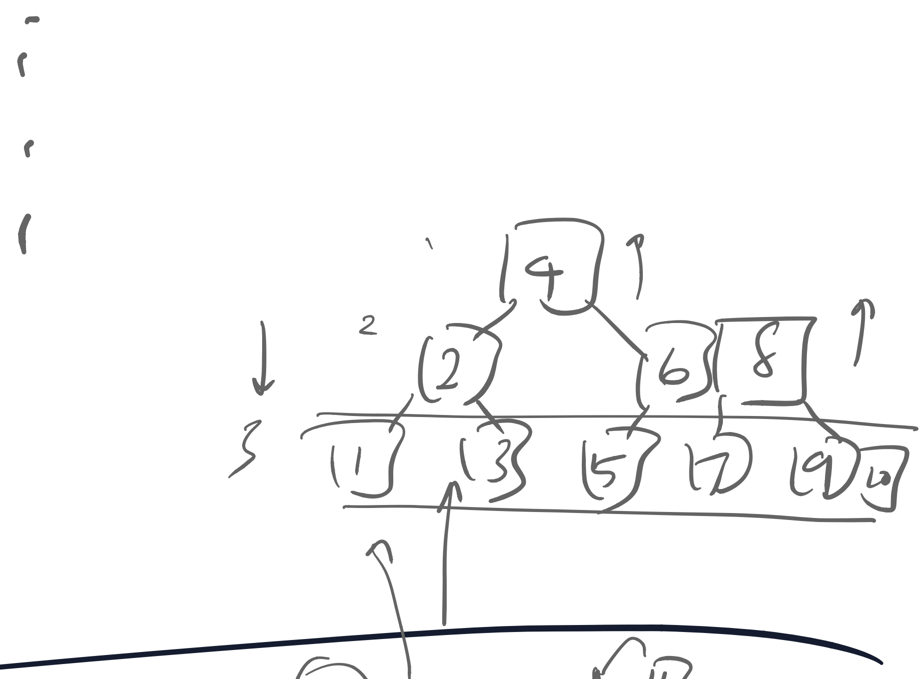
7.



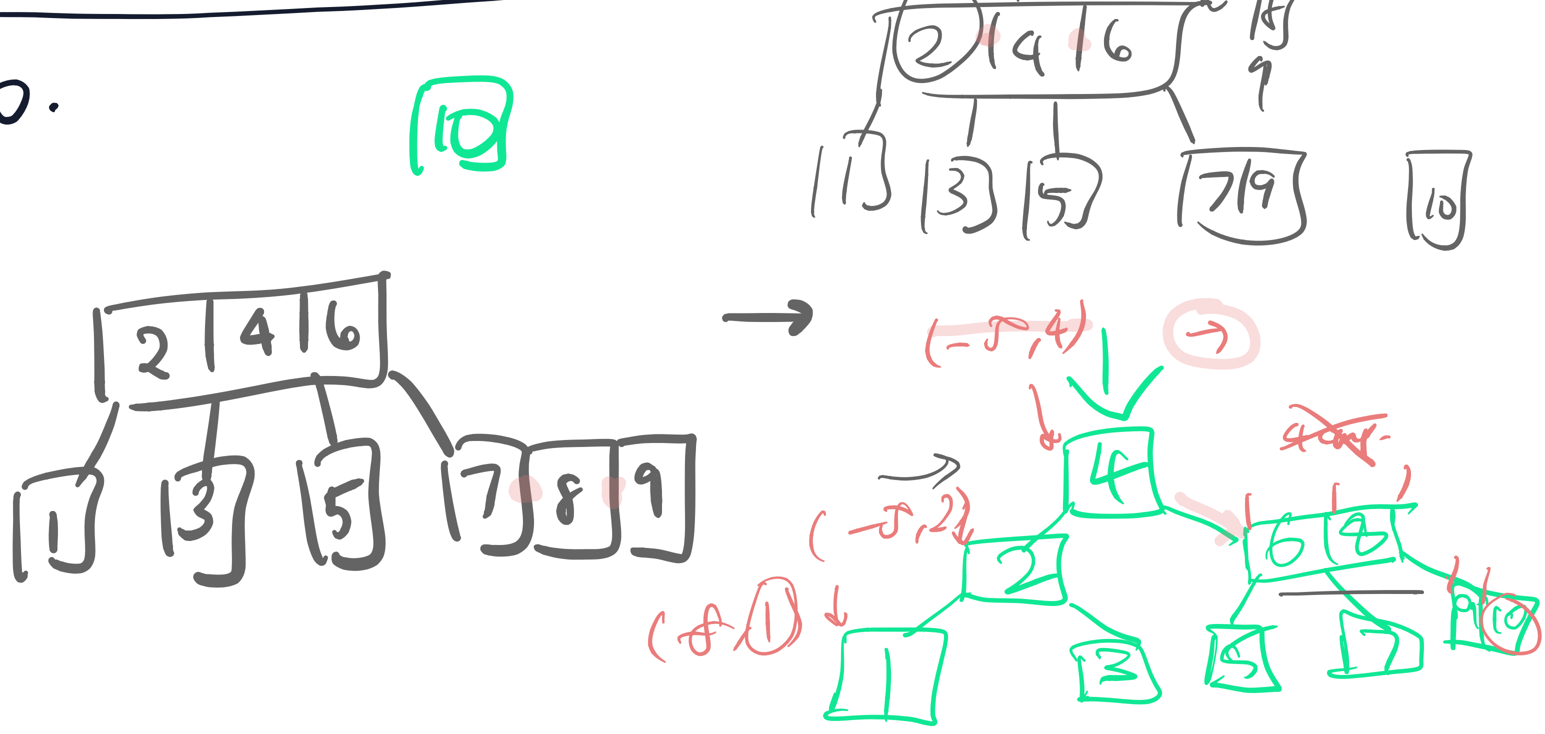

8.




9.



10.



search for 1

in interval $(-\infty, 4]$? YES!

in interval $(-\infty, 2]$? YES!

in interval $(-\infty, 1]$? YES!

3 comp.

search for 7

can check if 2 node or explicit range check.

1/ $(-\infty, 4]$? × $[4, \infty)$? ✓

2/ $(-\infty, 6]$ × $(6, 7]$ ✓. 3 checks.

$$\lfloor \log_2(10) \rfloor = 3.$$

3/7

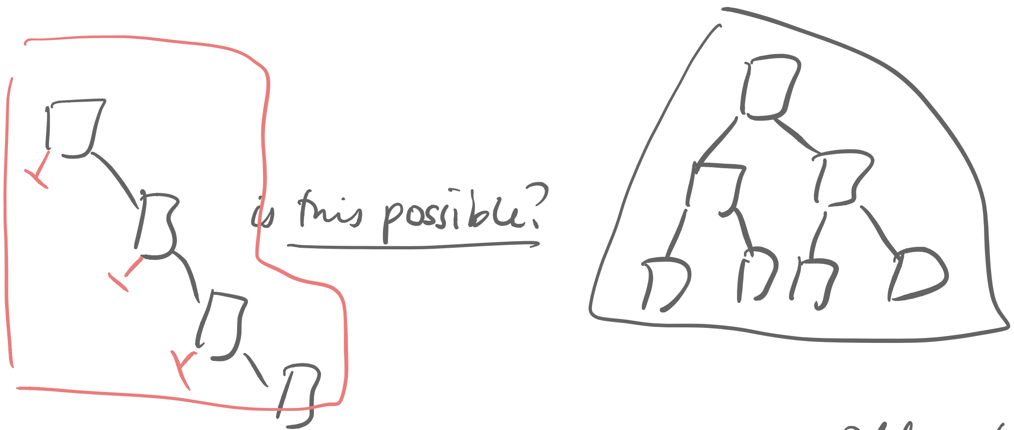
4/8

Some math to derive the worst case time complexities.

① Suppose I had a B tree with n nodes, then in order to find worst case (find the worst possible height of this tree w.r.t. n.

② Each node has at most 4 children or at worst 2 children.

③ The worst case is when each node has 2 children. (why?)




④ If every node has exactly 2 children in worst case ⇒ $O(\log_2(n))$ height. (balanced tree).

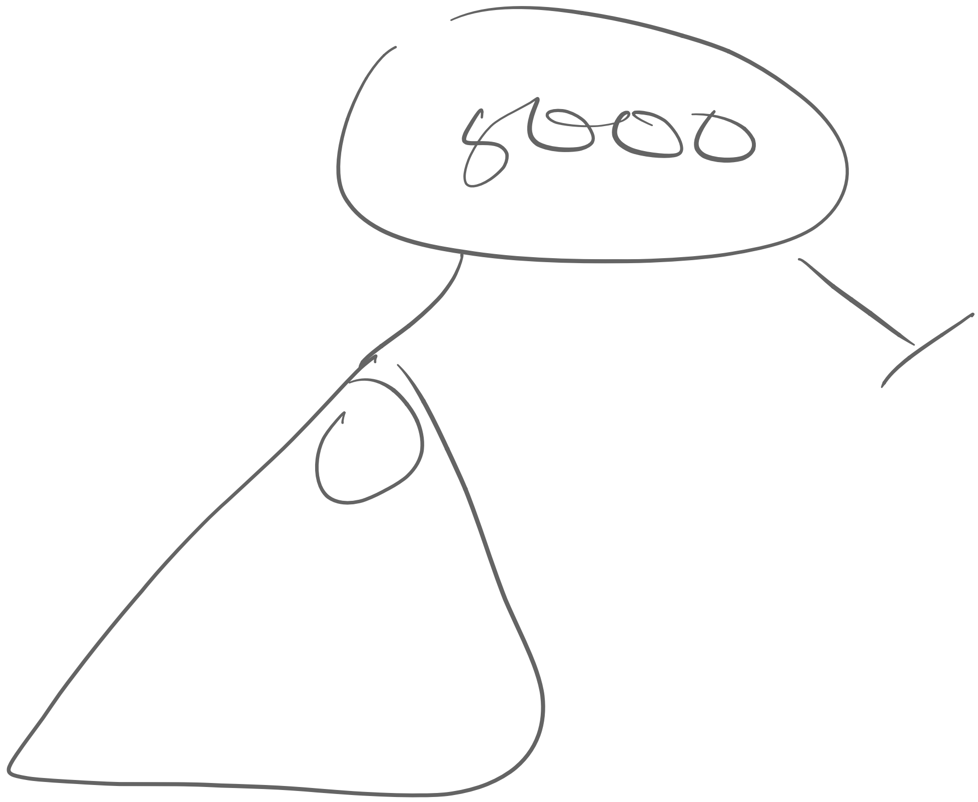
All LEAVES same level:

push one up -> leaves go one down the tree

-> subtrees go down one level only by construction..

1, 2, 3, 4

2500, 1250, . . . . . , $\boxed{5000}$




1 2 3, 4 5