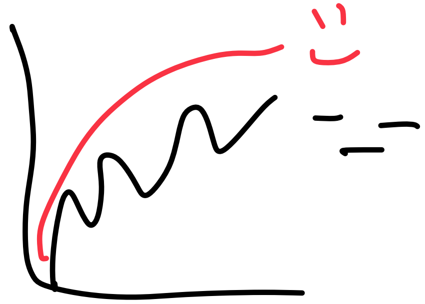
Data Structures

AVL Analysis

CS 225
Brad Solomon

September 30, 2024



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

No MP this week!

We will cover content necessary for mp_mosaics this week

An opportunity to catch up on work

An opportunity to complete the **Informal Early Feedback**

↳ +8 EC

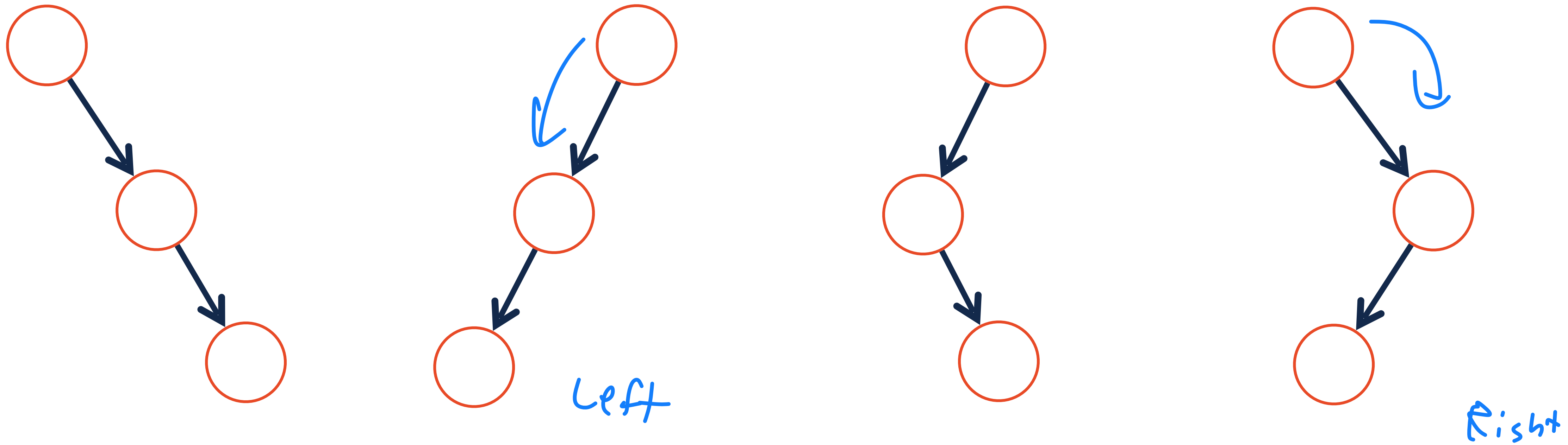
↳ 7 hours

Learning Objectives

Review AVL trees

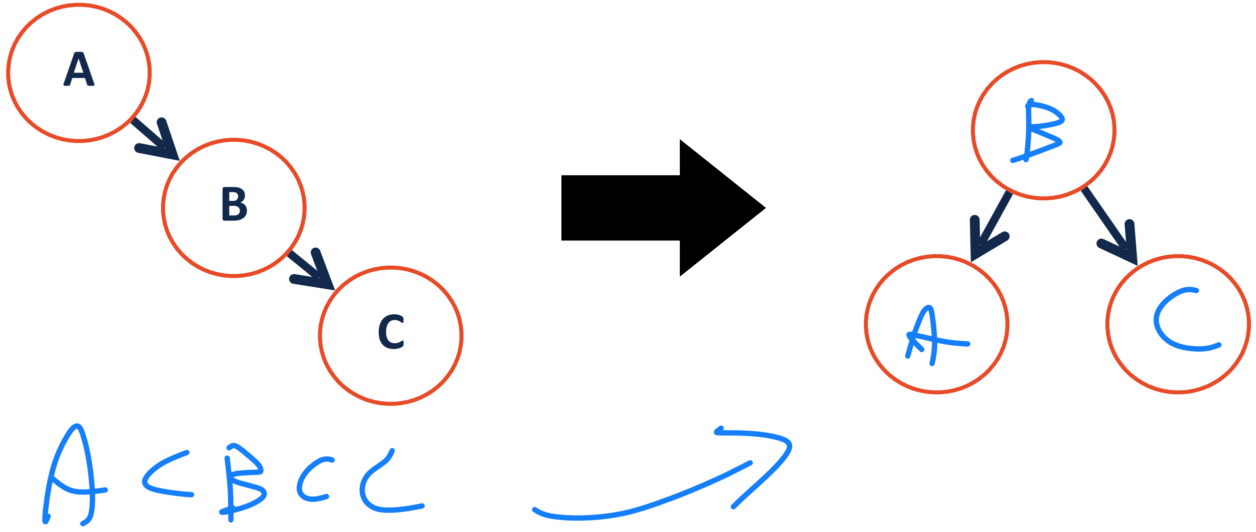
Prove that the AVL Tree speeds up all operations

AVL Rotations



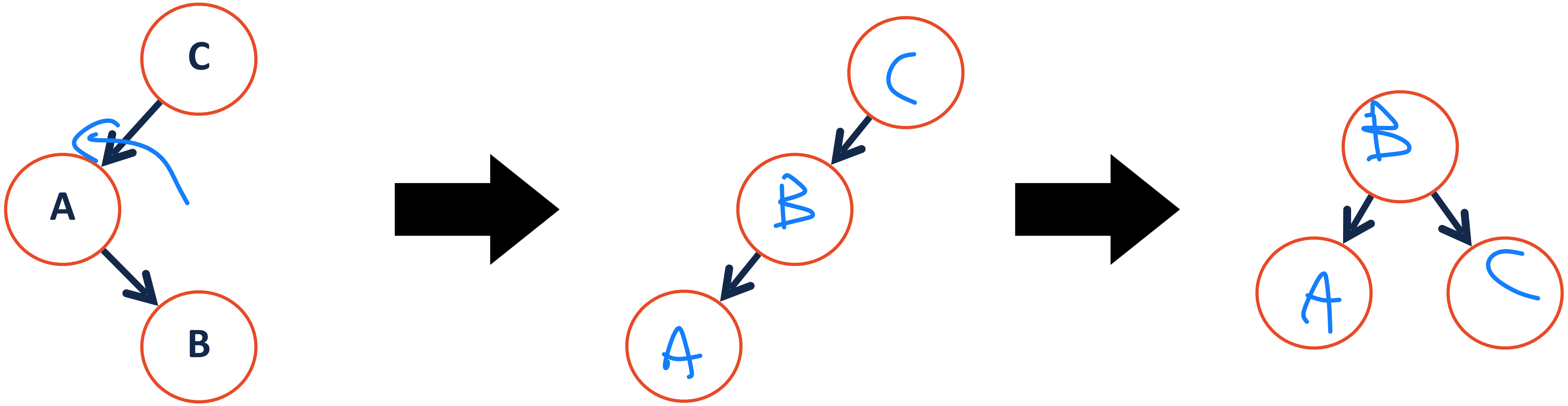
	Left	Right	LeftRight	RightLeft
Root Balance:	2	-2	-2	2
Child Balance:	1	-1	1	-1

AVL Tree Rotations



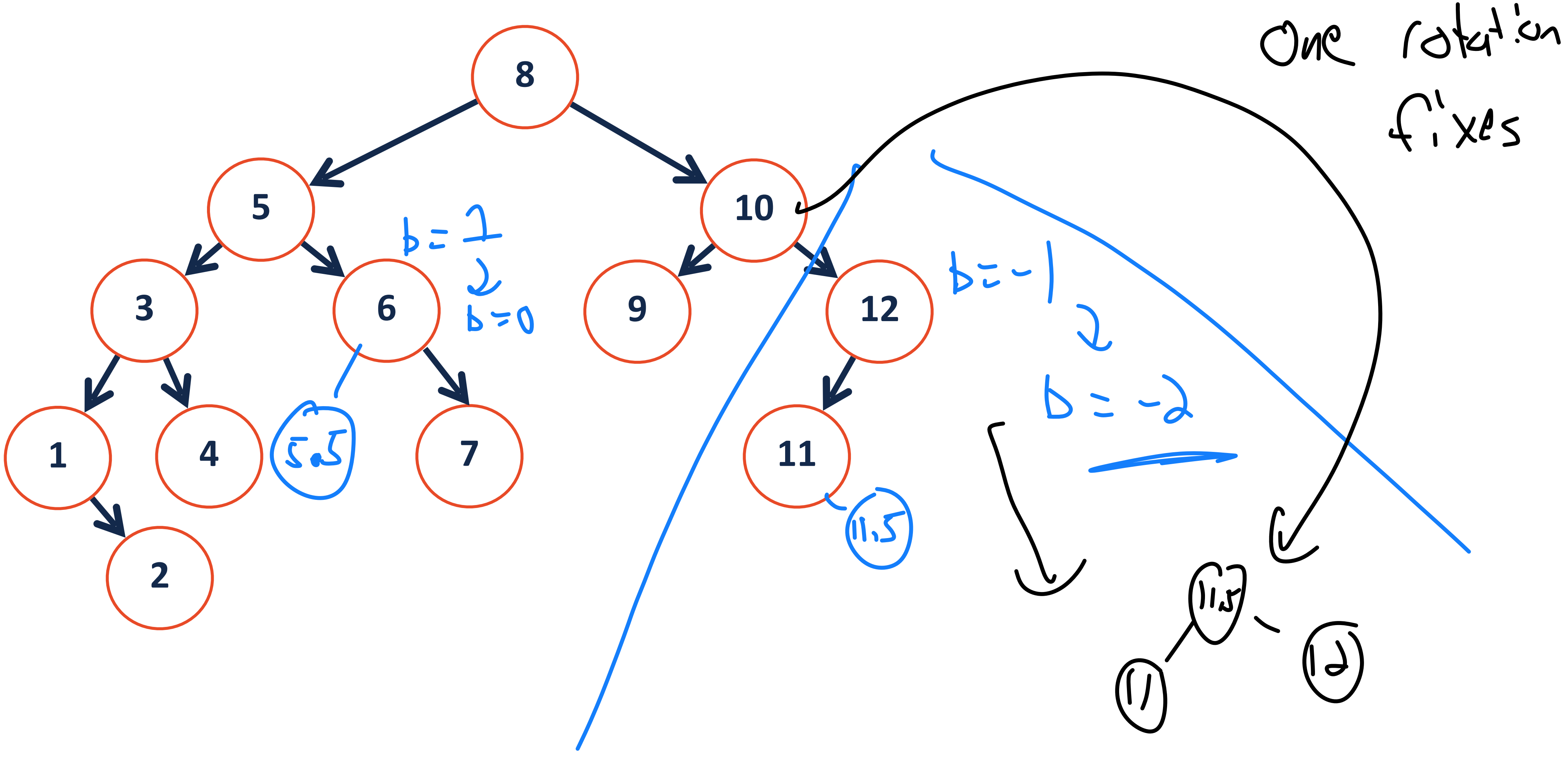
All rotations are O(1)

All rotations reduce subtree height by one



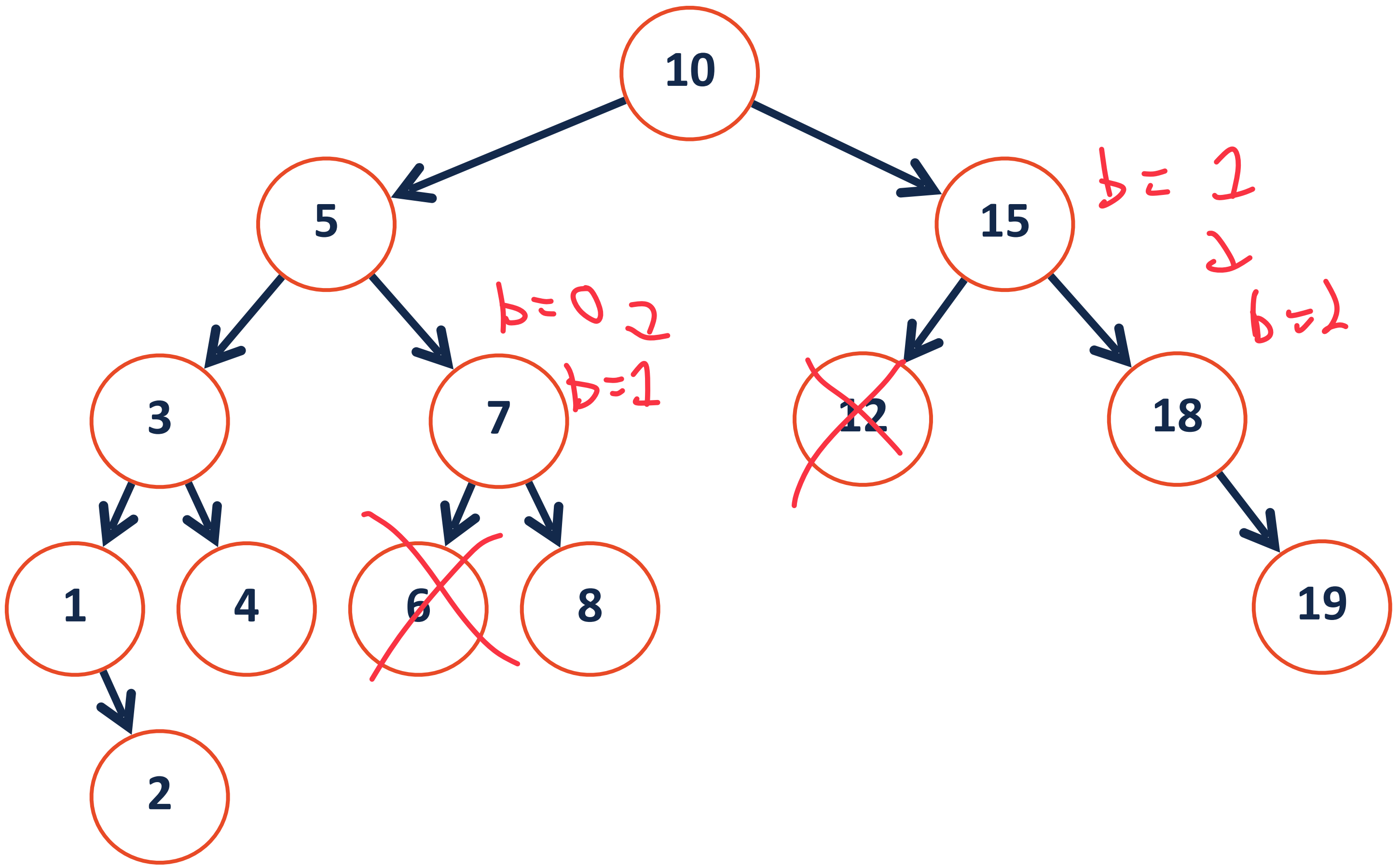
AVL Insertion

Given an AVL is balanced, insert can insert **at most** one imbalance



AVL Remove

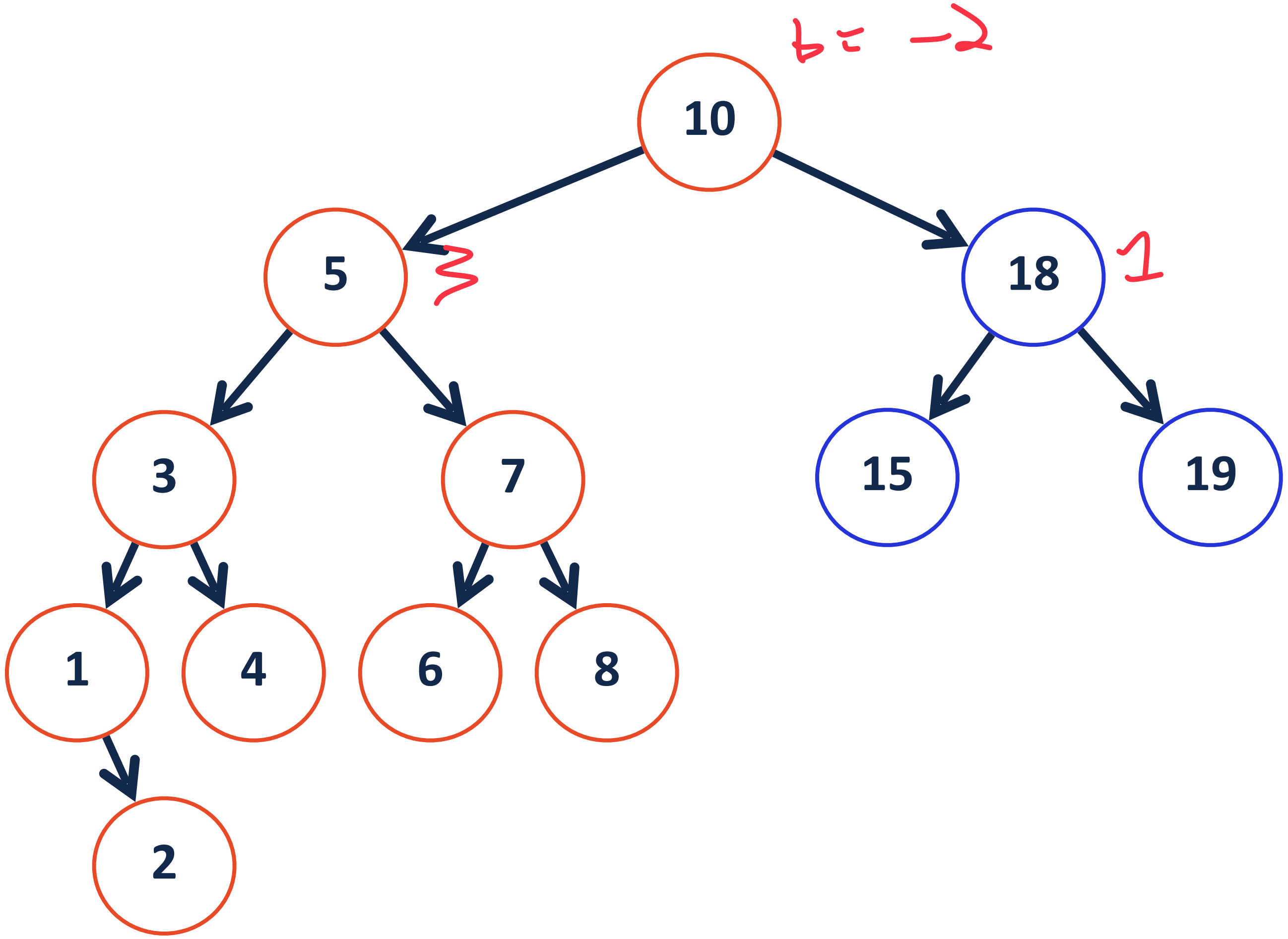
Remove can cause an imbalance at every level



AVL Remove

`_remove(12)`

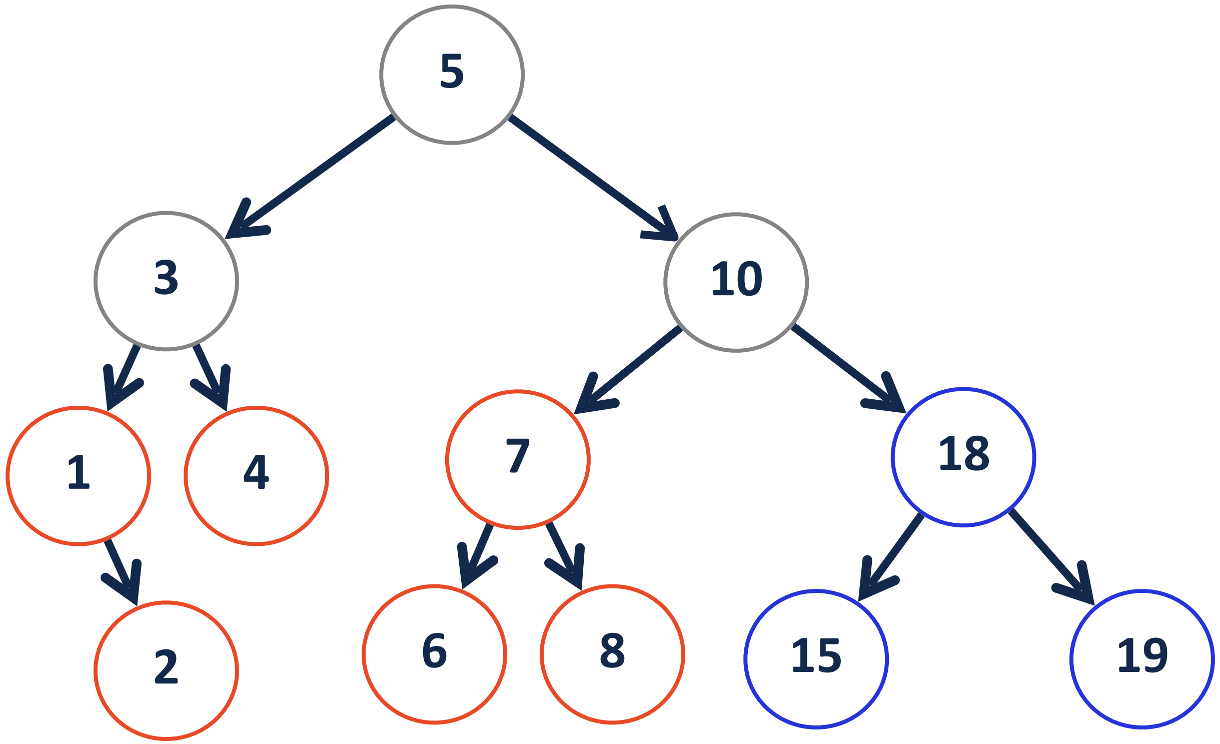
Remove can cause an imbalance at every level



AVL Remove

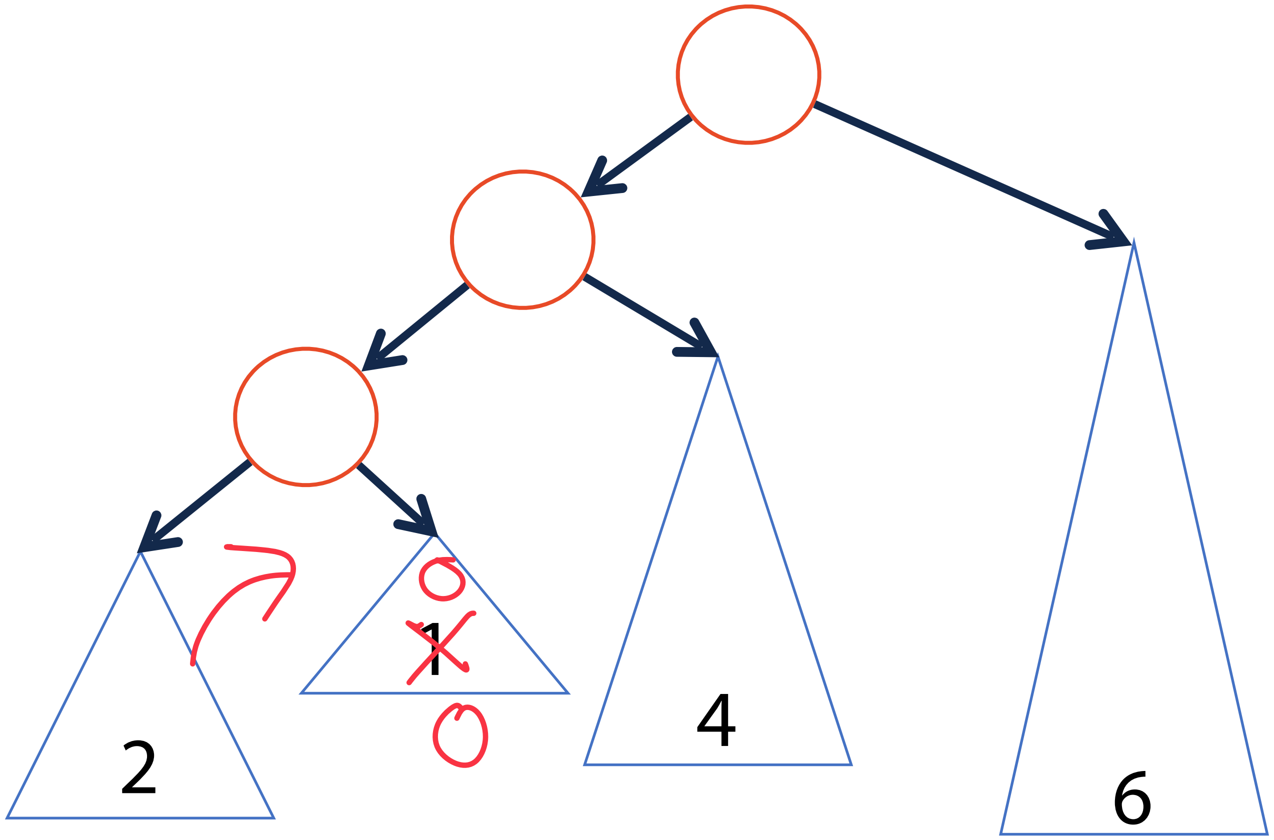
`_remove(12)`

Remove can cause an imbalance at every level



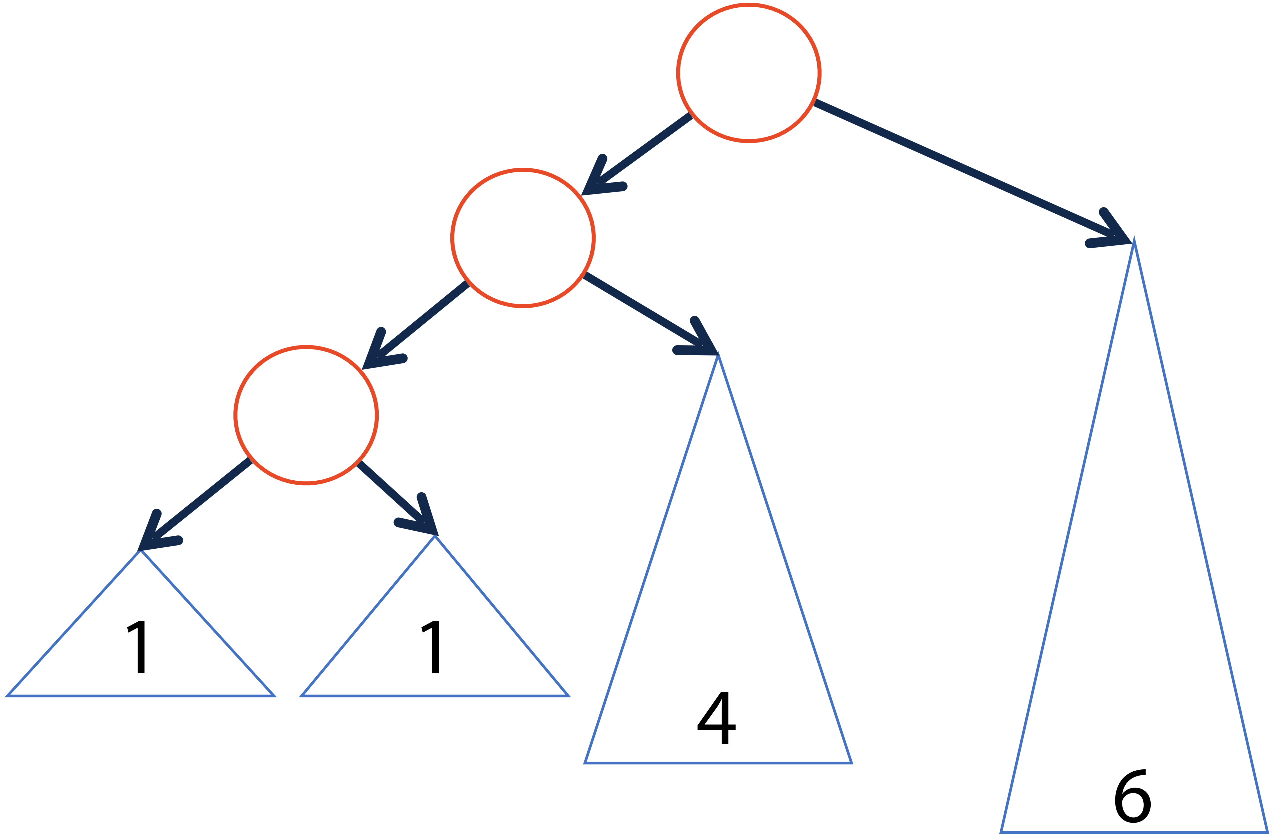
AVL Remove

Remove can cause an imbalance at every level



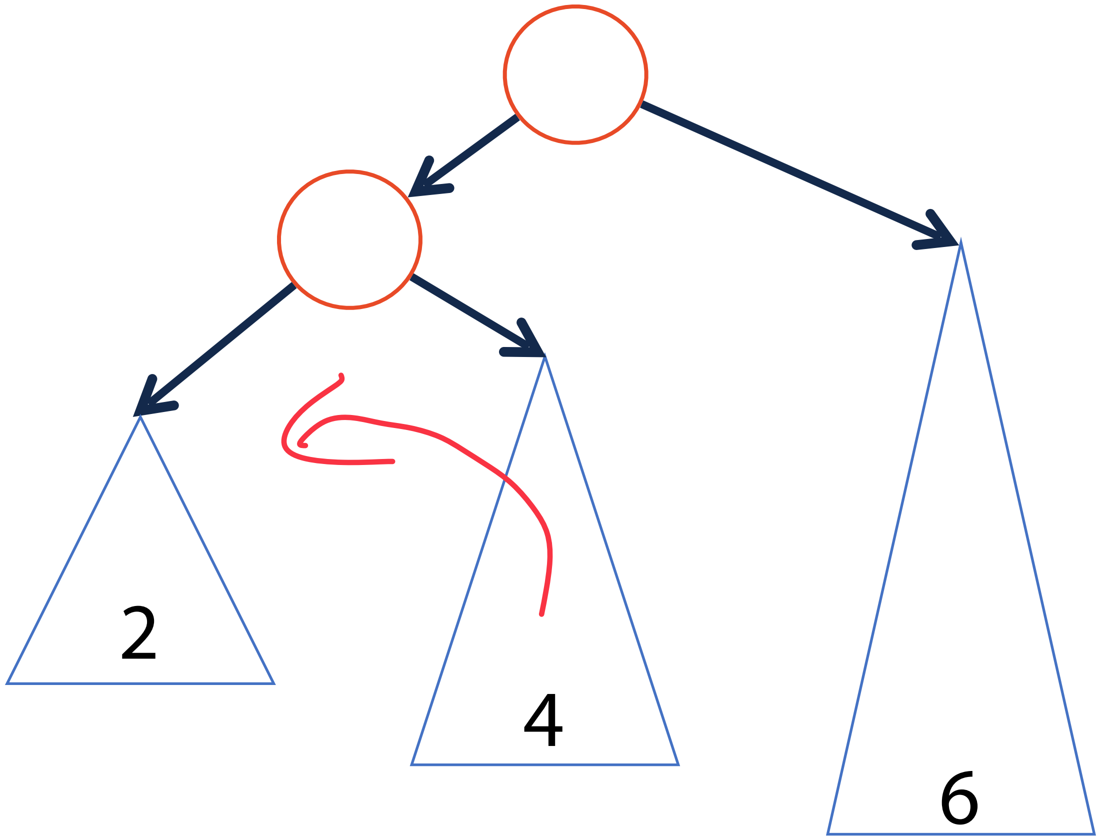
AVL Remove

Remove can cause an imbalance at every level



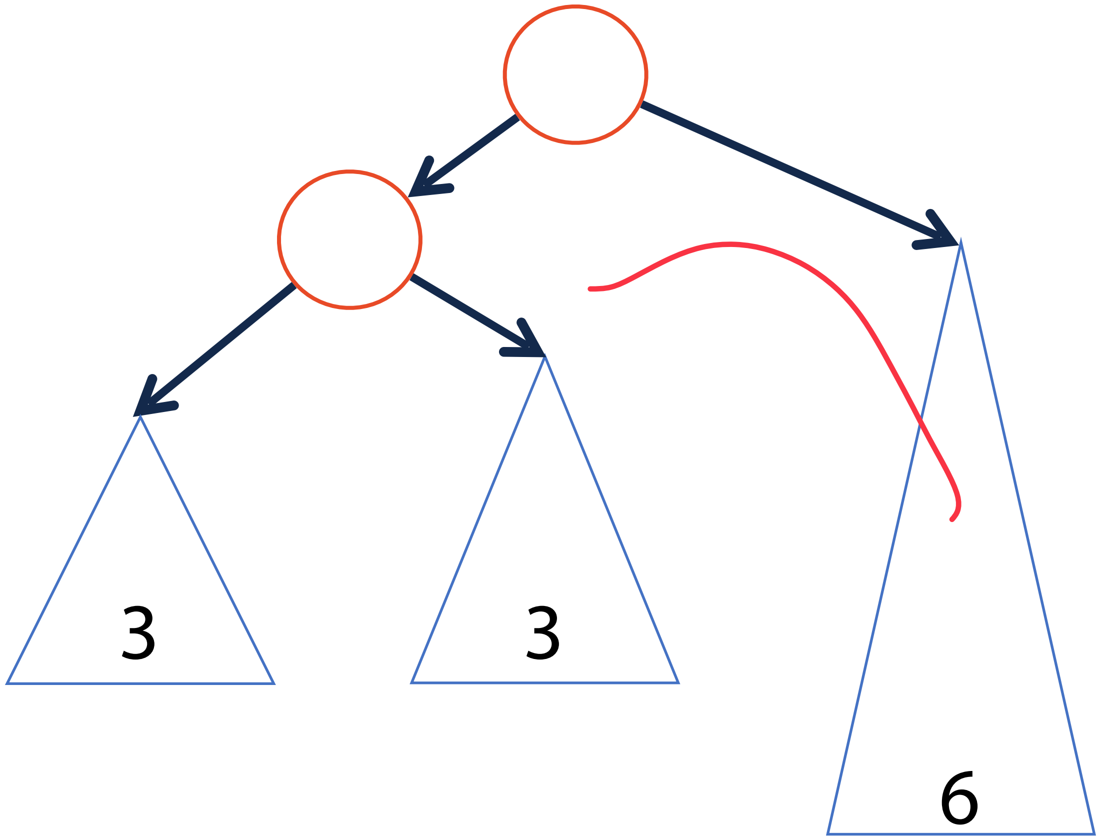
AVL Remove

Remove can cause an imbalance at every level



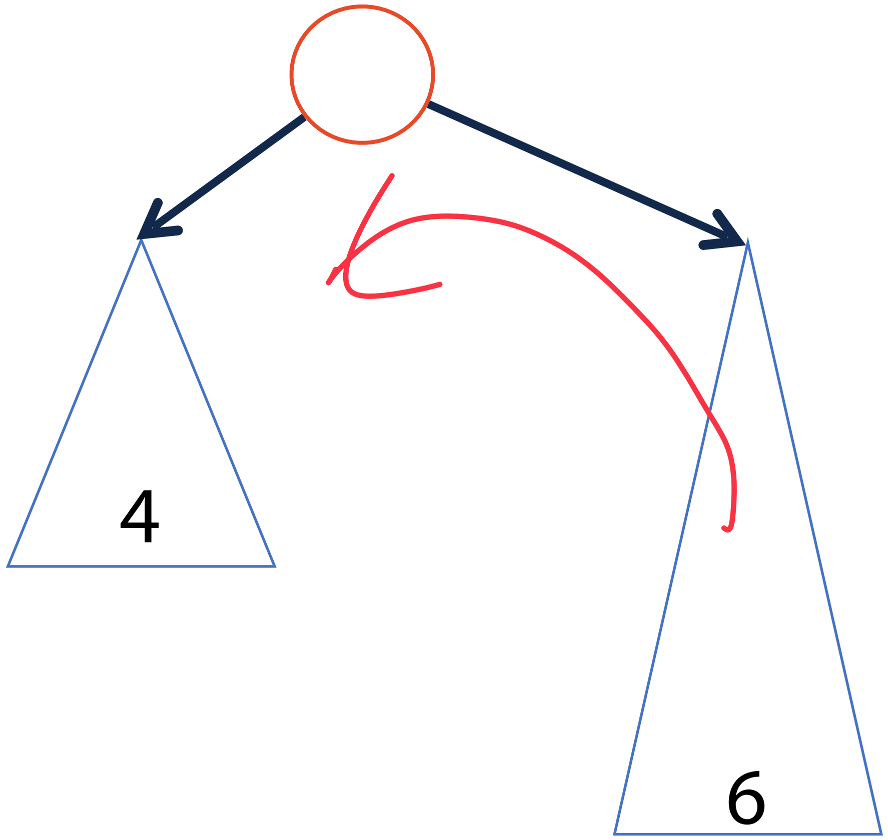
AVL Remove

Remove can cause an imbalance at every level



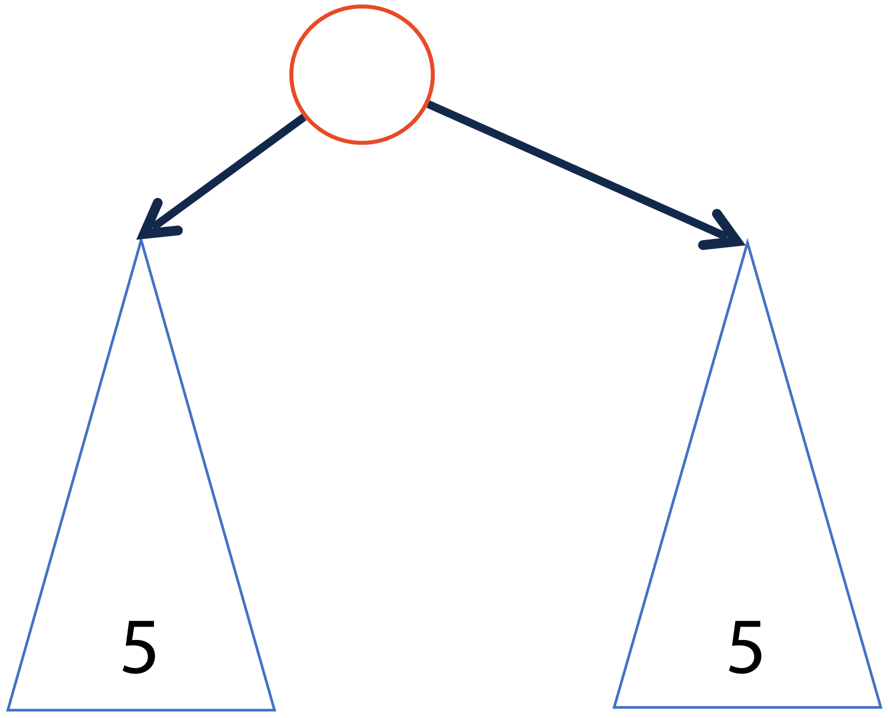
AVL Remove

Remove can cause an imbalance at every level



AVL Remove

Remove can cause an imbalance at every level



AVL Tree Analysis

Guarantee:

1) Tree is balanced

For an AVL tree of height h:

Find runs in: $O(h)$.

Insert runs in: $O(h)$.

Remove runs in: $O(h)$.

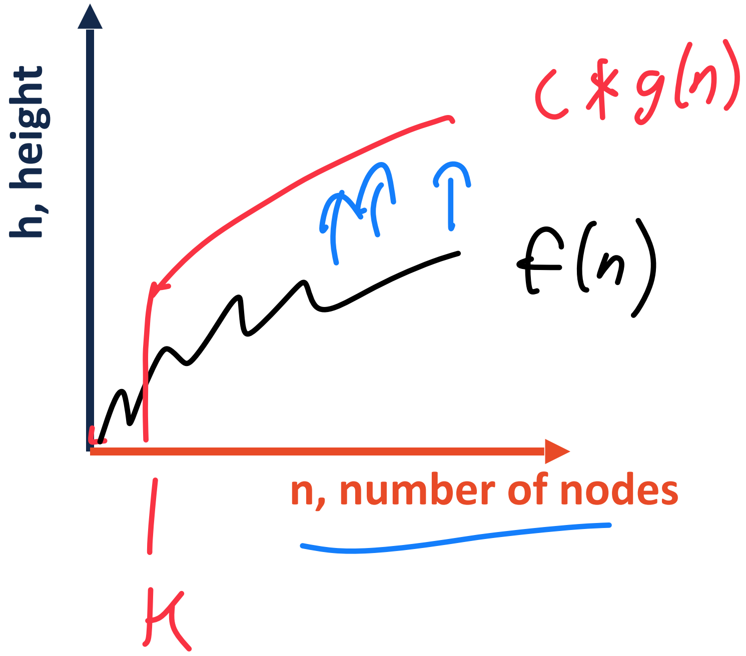
Claim: The height of the AVL tree with n nodes is: $O(\log n)$.

AVL Tree Analysis

Definition of big-O:

$$f(n) \text{ is } O(g(n)) \text{ iff } \exists c, k \text{ s.t. } f(n) \leq cg(n)\ \forall n > k$$

…or, with pictures:



— $h \leq \log(n)$

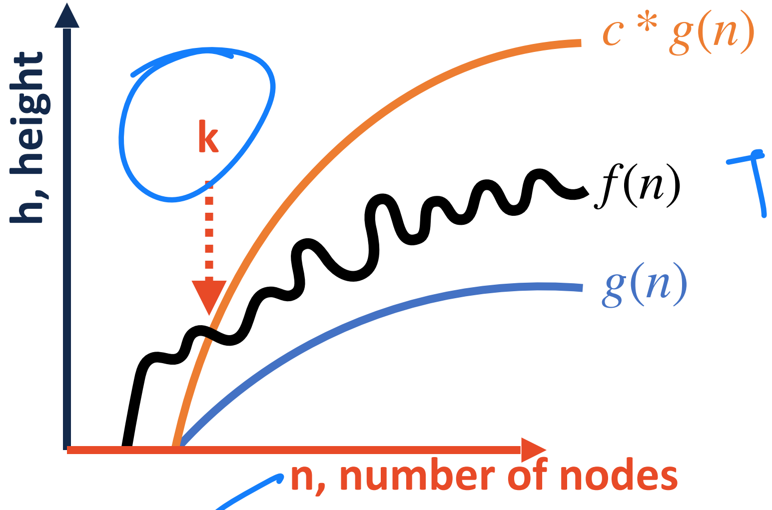
AVL Tree Analysis

We define!

$n = 2$

$n = 1000000 \ldots$

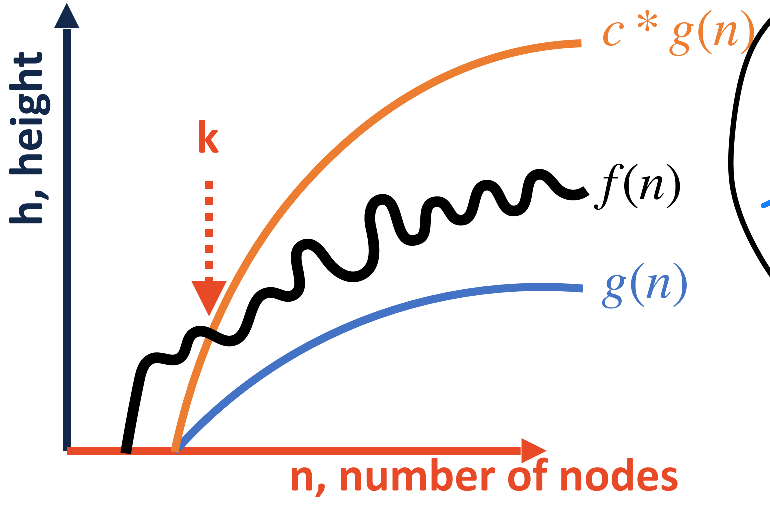
Tree height given n nodes



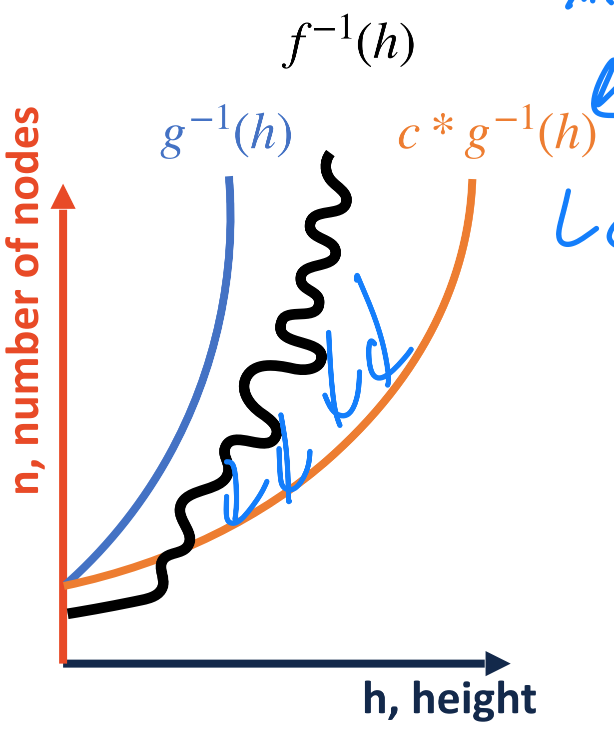
The height of the tree, **f(n)**, will always be <u>less than</u> **c × g(n)** for all values where **n > k**.

AVL Tree Analysis

Hard!



Invert



min nodes for tree of height h

Lower bound

mi

$f(n)$ = "Tree height given nodes"

$f^{-1}(h)$ = "Nodes in tree given height"

The number of nodes in the tree, $\mathbf{f^{-1}(h)}$, will always be <u>greater than</u> $\mathbf{c \times g^{-1}(h)}$ for all values where **n > k**.

Plan of Action

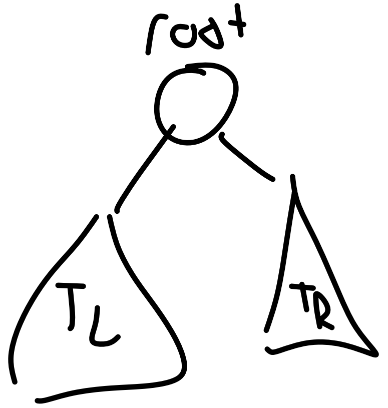
Since our goal is to find the lower bound on **n** given **h**, we can begin by defining a function given **h** which describes the smallest number of nodes in an AVL tree of height **h**:

Intuit

$N(h)$ = minimum number of nodes in an AVL tree of height h

$$N(h) = 1 + N(h-1) + N(h-2)$$

root



$T_L = h-1$ $T_R = h-2$

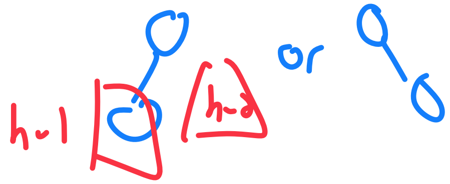
or

$T_R = h-2$ $T_L = h-1$

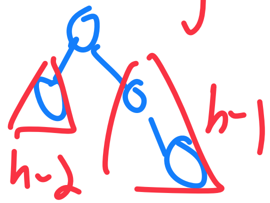
$N(-1) = 0$

$N(0) = 1$

$N(1) = 2$



$N(2) =$



Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

$> N(h-1) + N(h-2)$

$> 2N(h-2)$

$1 + N(x) > N(x)$

$N(h-1) > N(h-2)$

Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

$$N(h) > N(h-1) + N(h-2)$$

Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

$$N(h) > N(h-1) + N(h-2)$$

$$N(h) > 2N(h-2)$$

Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

$$N(h) > N(h-1) + N(h-2)$$

$$N(h) > 2N(h-2) \longrightarrow 2^{h/2}$$

1) Know characteristic equation? Get answer immediately!

Simplify the Recurrence

$N(h) = 1 + N(h-1) + N(h-2)$

$N(h) > N(h-1) + N(h-2)$

$N(h) > 2N(h-2)$

2) Unroll: $N(h) > 2N(h-2) = 2\left(2(N(h-4)\right) = 2^k\left(N(h-2k)\right)$

Simplify the Recurrence

$N(h) = 1 + N(h-1) + N(h-2)$

$N(h) > N(h-1) + N(h-2)$

$N(h) > 2N(h-2)$

2) Unroll: $N(h) > 2N(h-2) = 2\left(2(N(h-4)\right) = 2^k\left(N(h-2k)\right)$

When $h - 2k = 0, k = h/2$. Thus $N(h) > 2^{h/2}$

Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

$$N(h) > N(h-1) + N(h-2)$$

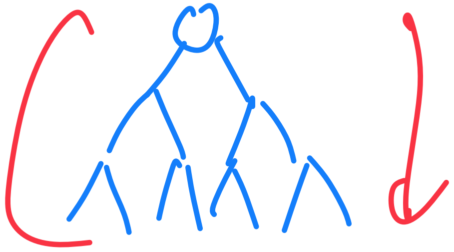
$$N(h) > 2N(h-2)$$

3) Intuit approximate shape from recursion

$h-1$

$h-2$

$N(h) \simeq 2N(h-1)$



$2^{h/2}$

Simplify the Recurrence

$$N(h) = 1 + N(h-1) + N(h-2)$$

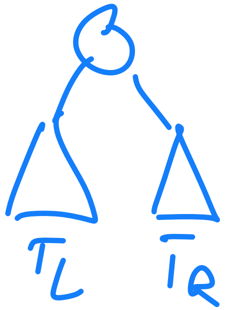
$$N(h) > N(h-1) + N(h-2)$$

$$N(h) > 2N(h-2)$$

By whatever strategy you like: $N(h) > 2^{h/2}$

Tip #1: Recurrence relation → Induction

(Binary) Tree →



Induction Step made easy

State a Theorem

$1 + N(h-1) + N(h-2)$

Theorem: An AVL tree of height h has at least $N(h) > 2^{h/2}$.

Proof by Induction:

I. Consider an AVL tree and let **h** denote its height.

II. Base Case: height = 1

$2^{1/2} \approx 1.41$

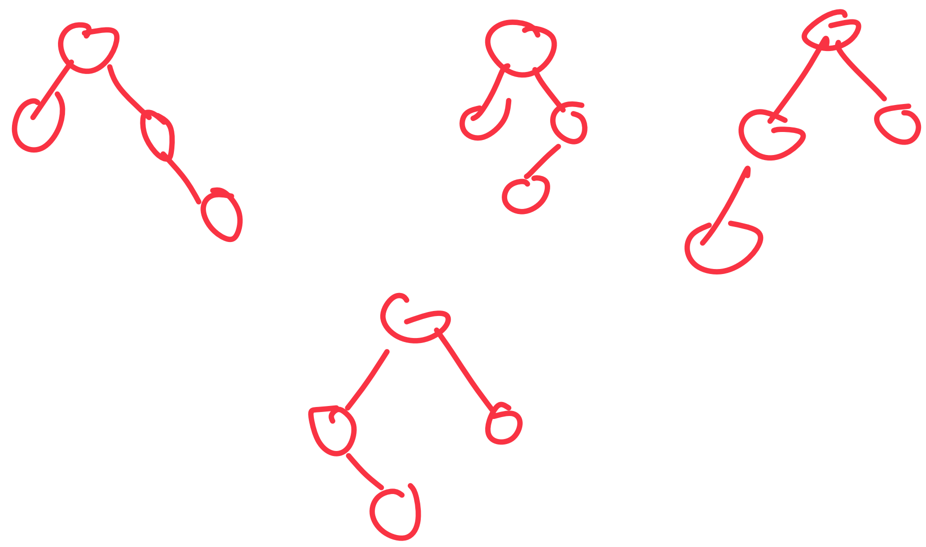
2 nodes!

This is true

An AVL tree of height 1 has at least 1.41 nodes.

Prove a Theorem

III. Base Case: height = 2



$$N(2) = 1 + N(h-1) + N(h-2)$$
$$= 1 + 2 + 1$$
$$= 4$$

min 4 nodes

has four nodes

$$f(h) = 2^{2/2} = 2^1 = 2$$

An AVL tree of height 2 has at least 2 nodes.

Prove a Theorem

IV. Induction Step: Assume for all heights $i < h$, $N(i) \geq 2^{i/2}$.

h-1

Prove that $N(h) \geq 2^{h/2}$

$N(h) = 1 + N(h-1) + N(h-2)$ simplify

$> 2N(h-2)$ plug in $N(h)$ By IH

$> 2 \cdot 2^{(h-2)/2}$ rewrite

$> \cancel{2} \cdot 2^{h/2 - 1 + 1}$

$> 2^{h/2}$

Prove a Theorem

IV. Induction Step: Assume for all heights $i < h, N(i) \geq 2^{i/2}$.

Prove that $N(h) \geq 2^{h/2}$

$$N(h) = 1 + N(h-1) + N(h-2)$$

simplify

$$N(h) > 2N(h-2)$$

sub by IH

$$N(h) > 2 * 2^{(h-2)/2}$$

rewrite

$$N(h) > 2 * 2^{h/2-1}$$

$$N(h) > 2^{h/2}$$

Prove a Theorem

V. Using a proof by induction, we have shown that:

Prove a Theorem

V. Using a proof by induction, we have shown that:

$N(h) \geq 2^{h/2}$, where $N(h)$ is the **min # of nodes of a tree of height h**

But we need to know n, the **# of nodes in any tree of height h**

$$n \geq N(h) \geq 2^{h/2}$$

$\log_2(\ )$

$$\log(n) \geq h/2$$

rewrite

$$h \leq 2\log(n)$$

Prove a Theorem

V. Using a proof by induction, we have shown that:

$N(h) \geq 2^{h/2}$, where $N(h)$ is the **min # of nodes of a tree of height h**

But we need to know n, the **# of nodes in any tree of height h**

$$n \geq N(h)$$

$$log(n) \geq \frac{h}{2}$$

$$h \leq 2\ log(n)$$

for all $h \geq 1$

AVL Runtime Proof

An upper-bound on the height of an AVL tree is **O(lg(n))**:

N(h) := Minimum # of nodes in an AVL tree of height h

$N(h) = 1 + N(h-1) + N(h-2)$

$> 1 + 2^{(h-1)/2} + 2^{(h-2)/2}$

$> 2 \times 2^{(h-2)/2} = 2^{(h-2)/2+1} = 2^{h/2}$

Theorem #1:

Every AVL tree of height h has at least $2^{h/2}$ nodes.

AVL Runtime Proof

An upper-bound on the height of an AVL tree is **O(lg(n))**:

# of nodes (n) $\geq N(h) > 2^{h/2}$

$n > 2^{h/2}$

$\lg(n) > h/2$

$2 \times \lg(n) > h$

$h < 2 \times \lg(n)$ ***, for $h \geq 1$***

Proved: The maximum number of nodes in an AVL tree of height h is less than $2 \times \lg(n)$.

Summary of Balanced BST

Pros:

Find is simple
↳ Easy to code
↳ Reasonably fast

Insert

Remove

O(h) ⇒ O(log n)

↳ O(log n) speed for all major functions

↳ Search trees are great for range find / nearest neighbor
↳ Iterators on trees

KD Tree

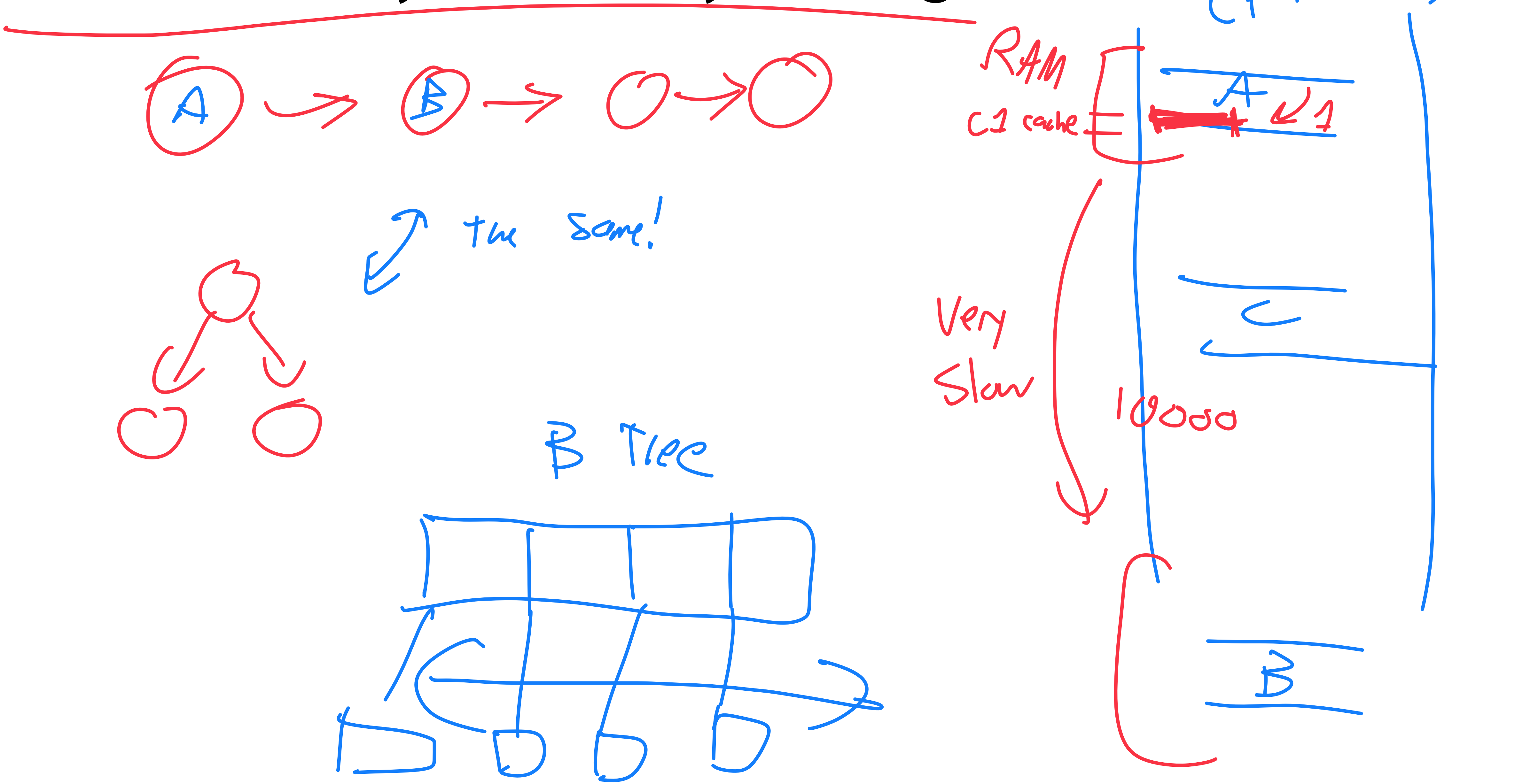
Cons:

↳ Runtime is O(log n)
↳ Maybe not optimal!
↳ O(1) is optimal
↳ O(0) :)

↳ Bad memory management (cache locality)
↳ Large-ish overload

BTree

Cache Locality / Memory Management



Every Data Structure So Far

	Unsorted Array	Sorted Array	Unsorted Linked List	Sorted Linked List	Binary Tree	BST	AVL
Find							
Insert							
Remove							
Traverse							