Algorithms and Data Structures for Data Science

AVL Analysis

CS 277
Brad Solomon

October 27, 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Final Project Proposal Resubmissions

There is no official resubmission deadline

It is in your best interest to resubmit this week

Remember that November 8th - 12th is mid-project checkins

Final project expectations will not change even if you start ‘late’

Learning Objectives

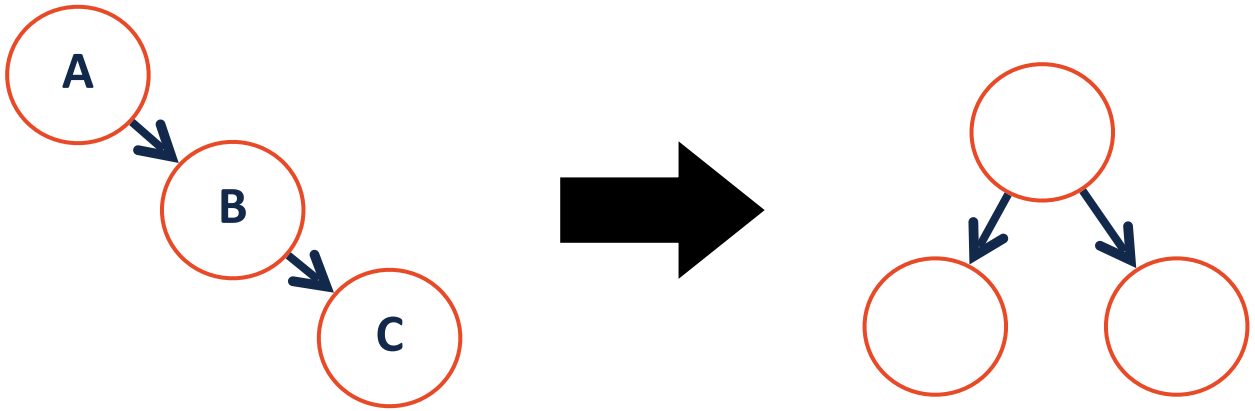
Finish discussing AVL functions (remove)

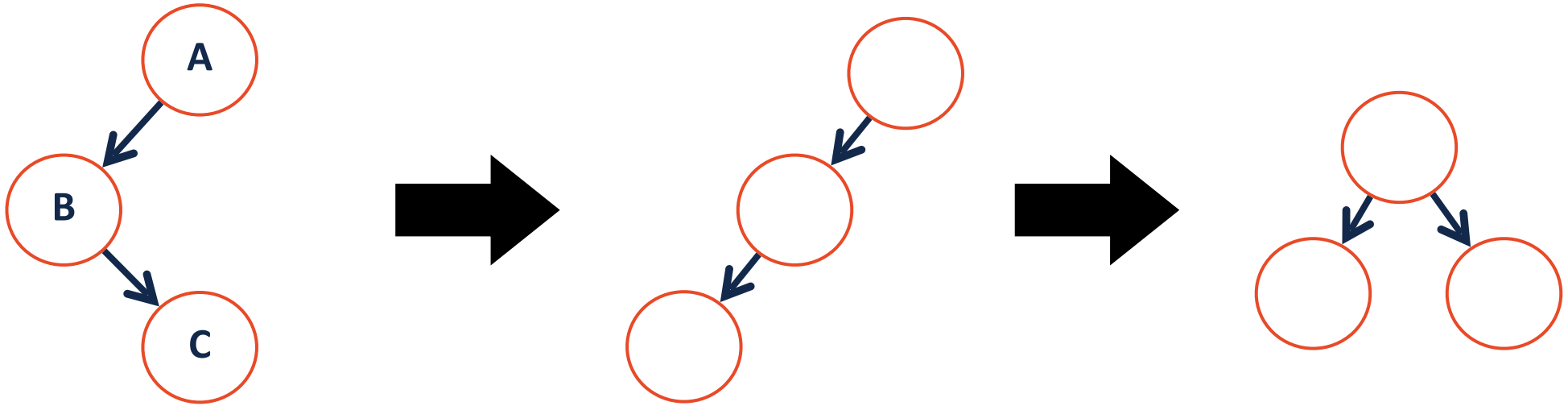
Prove that the AVL tree's height is bounded

AVL Tree Rotations

All rotations are O(1)

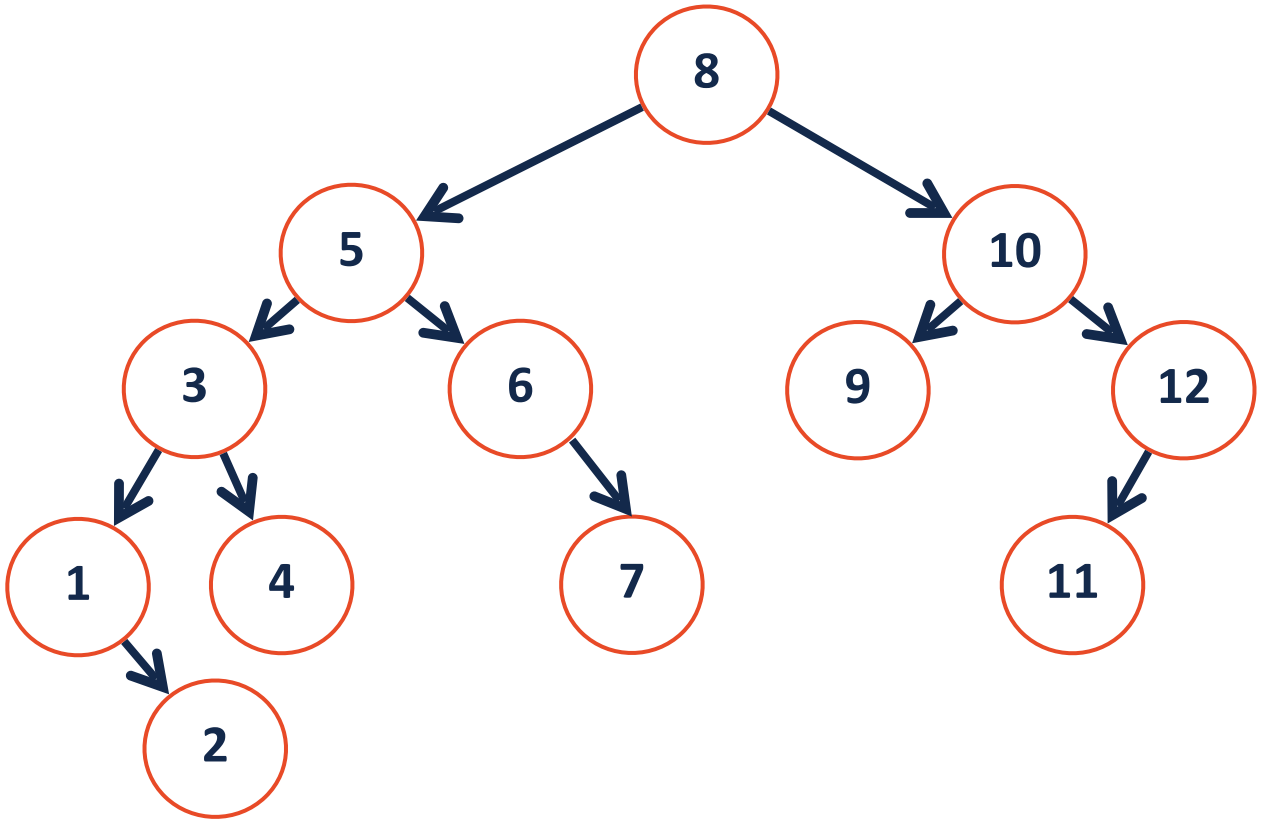
All rotations reduce subtree height by one





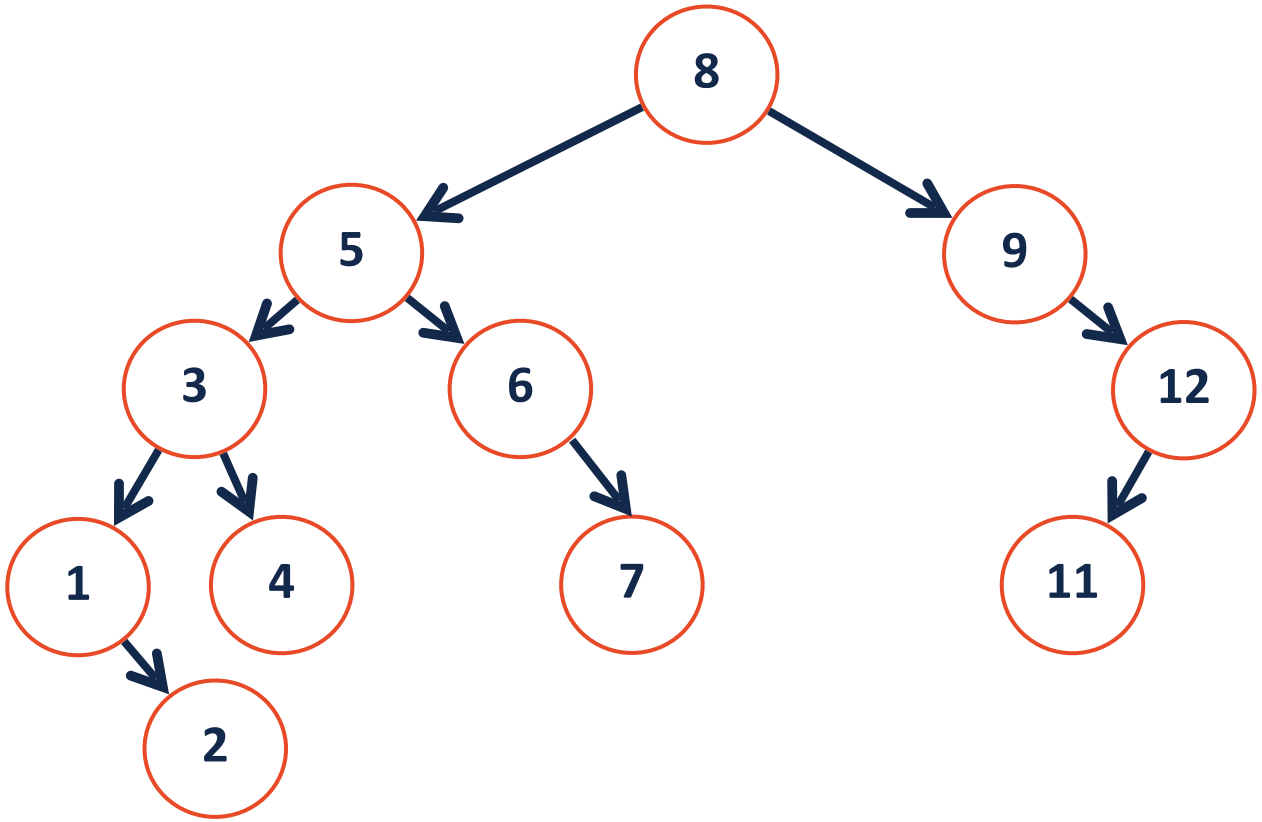
AVL Remove

`_remove(10)`



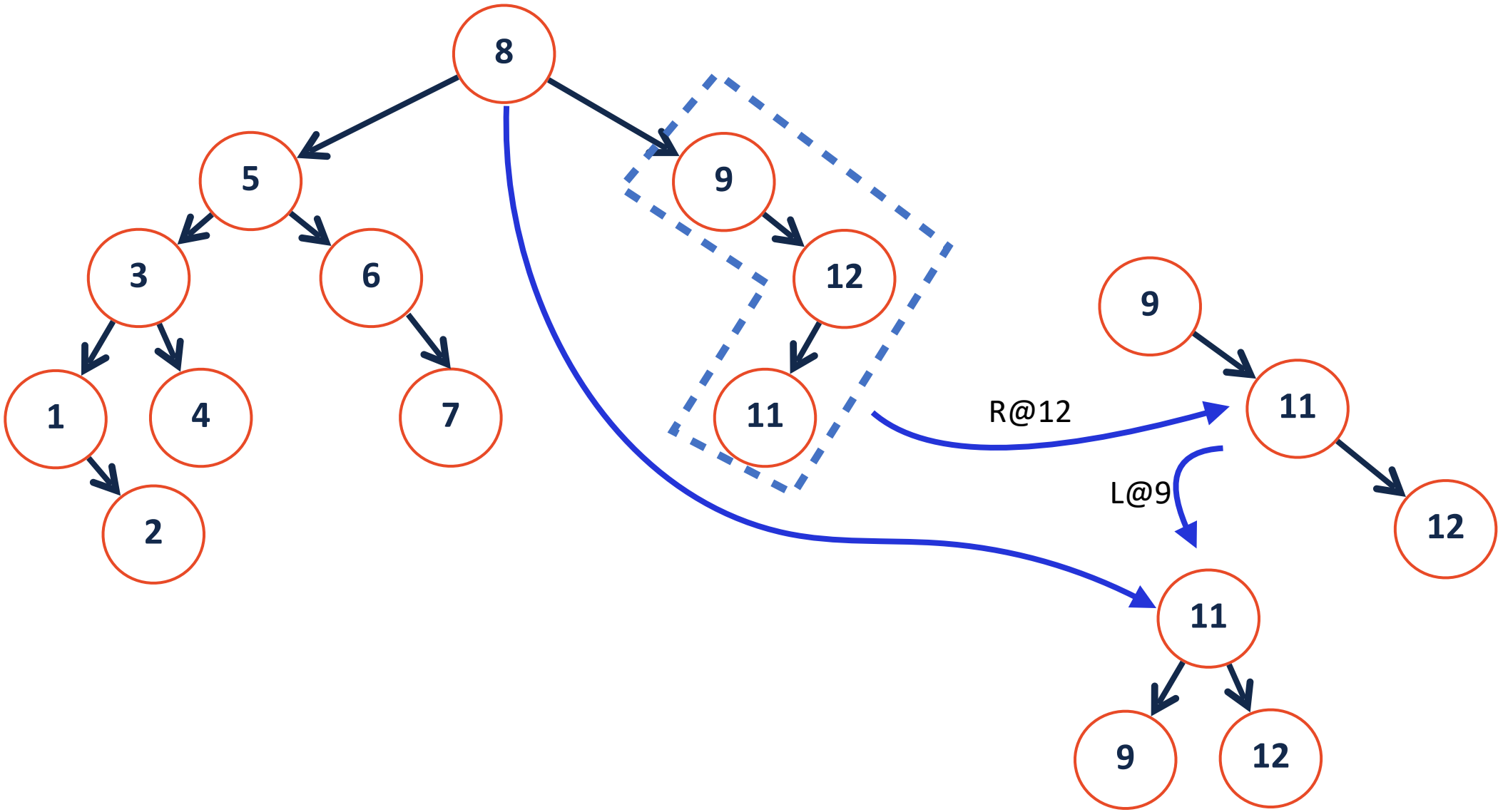
AVL Remove

`_remove(10)`



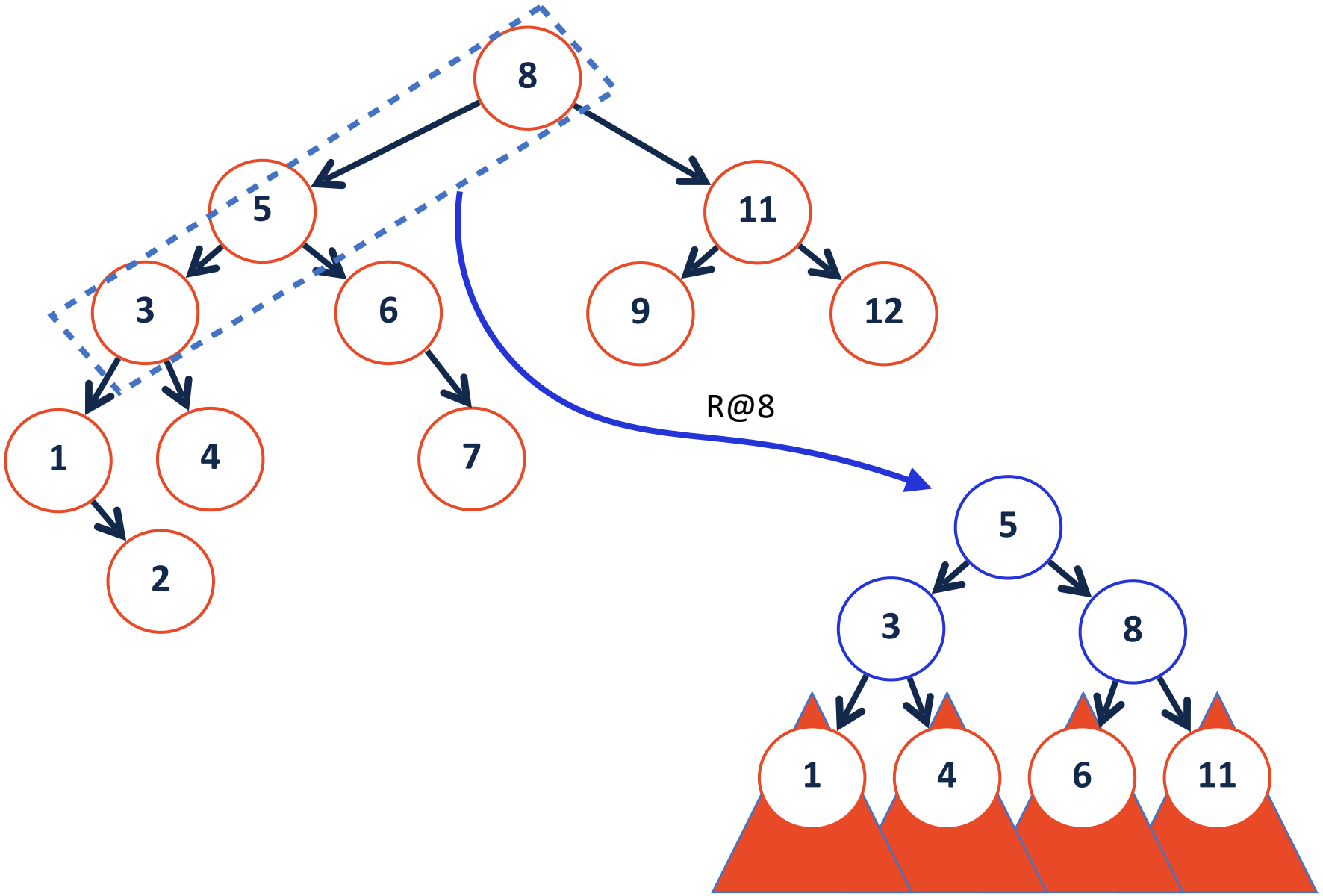
AVL Remove

`_remove(10)`



AVL Remove

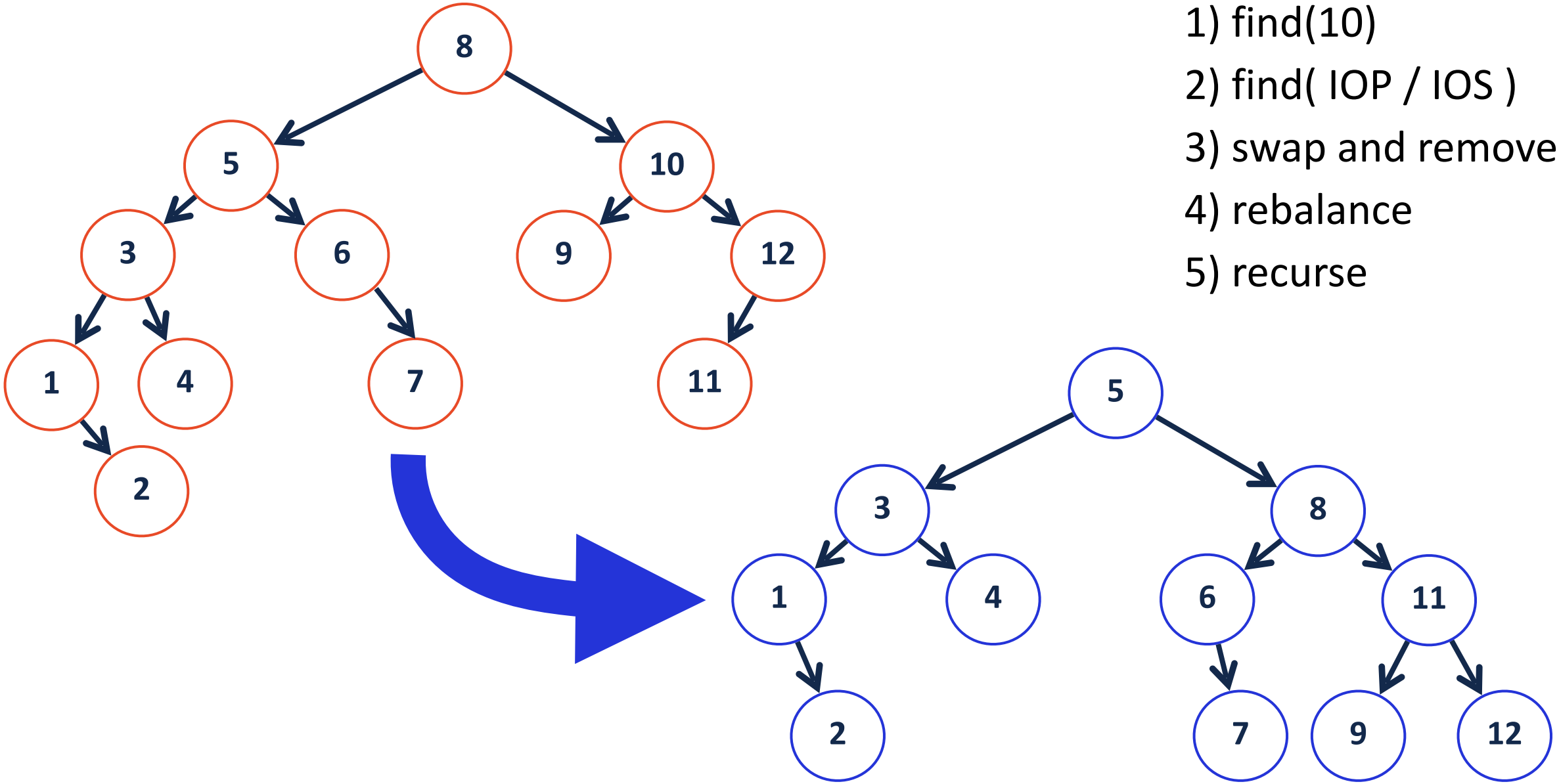
`_remove(10)`



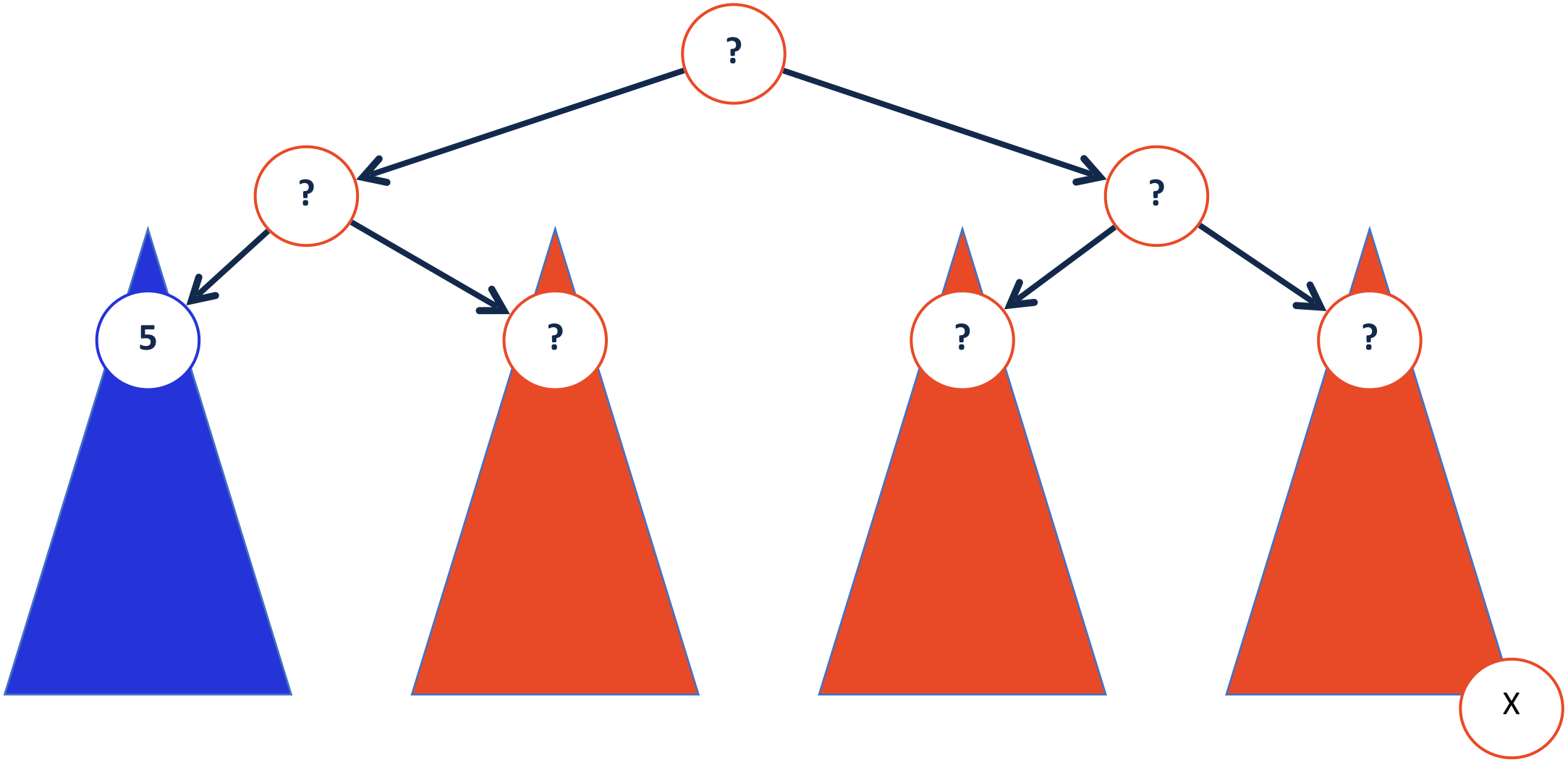
AVL Remove

`_remove(10)`

1) find(10)
2) find(IOP / IOS)
3) swap and remove
4) rebalance
5) recurse



AVL Remove



AVL Tree Analysis

For AVL tree of height h, we know:

find runs in: _____.

insert runs in: _____.

remove runs in: _____.

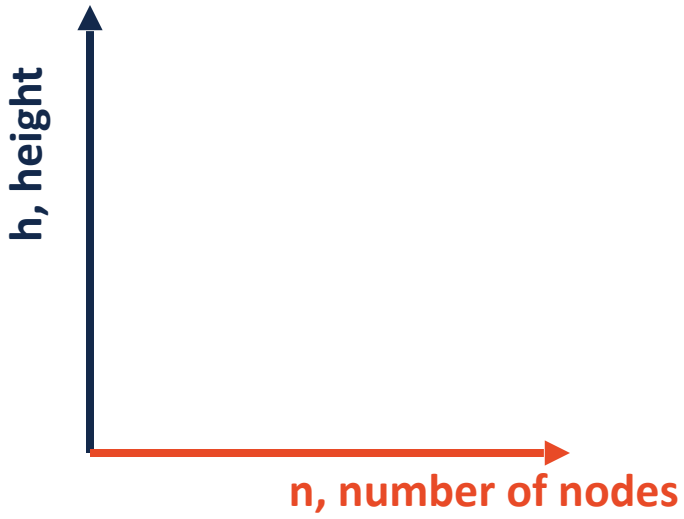
We will argue that: h is _____.

AVL Tree Analysis

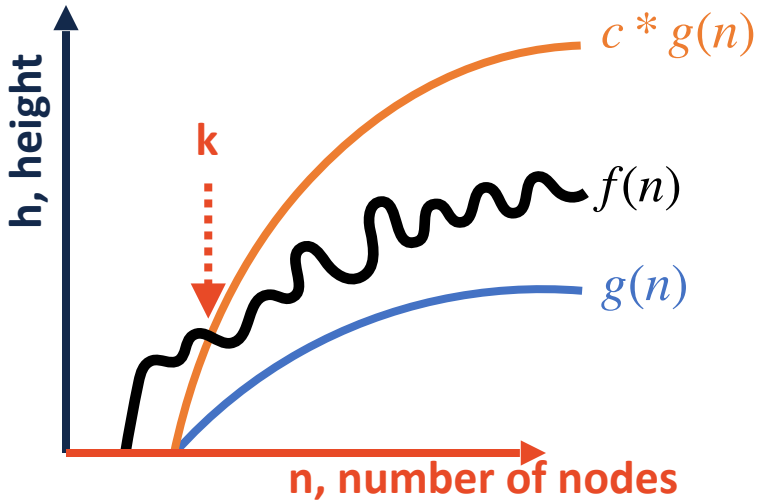
Definition of big-O:

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

…or, with pictures:

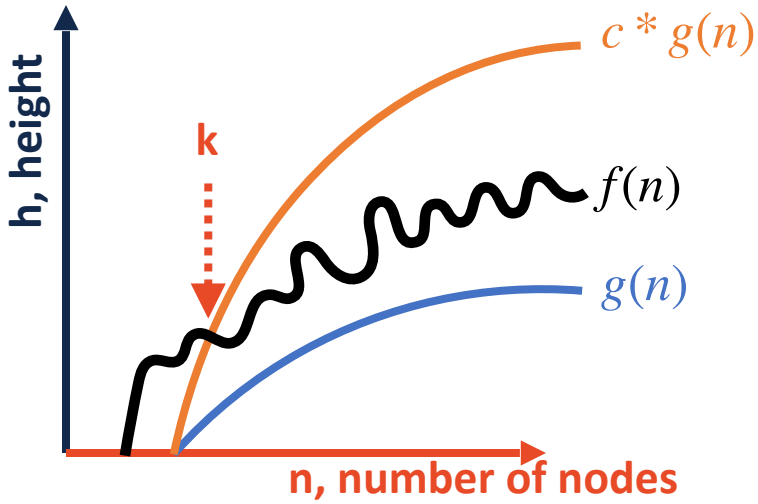


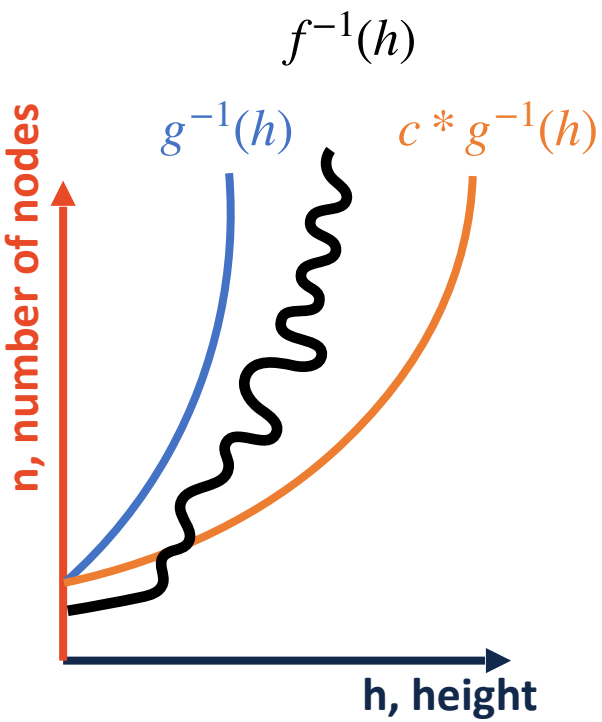
AVL Tree Analysis



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





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

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

The number of nodes in the tree, **$f^{-1}(h)$**, will always be <u>greater than</u> **$c \times g^{-1}(h)$** 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**:

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

Simplify the Recurrence

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

$$N(h) \geq N(h) - 1$$

State a Theorem

Theorem: An AVL tree of height h has at least _____.

Proof by Induction:

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

II. Base Case: _____

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

Prove a Theorem

III. Base Case: _____

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

Prove a Theorem

IV. Induction Case: _____

If for all heights $i < h, N(i) \geq 2^{i/2}$

then we must show for height h that $N(h) \geq 2^{h/2}$

Prove a Theorem

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

…and inverting:

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) ≥ 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 × lg(n).

Summary of Balanced BST

AVL Trees

- Max height: 1.44 * lg(n)
- Rotations:

Summary of Balanced BST

AVL Trees

- Max height: 1.44 * lg(n)
- Rotations:

  Zero rotations on find
  One rotation on insert
  O(**h**) == O(**lg(n)**) rotations on remove

Red-Black Trees

- Max height: 2 * lg(n)
- Constant number of rotations on insert (max 2), remove (max 3).

Summary of Balanced BST

Pros:

- Running Time:

  - Improvement Over:

- Great for specific applications:

Summary of Balanced BST

Cons:

- Running Time:

- In-memory Requirement: