

Remember
can suggest
you
music!

Data Structures

AVL Analysis

CS 225

Brad Solomon

October 5, 2026



UNIVERSITY OF
ILLINOIS
URBANA - CHAMPAIGN

Department of Computer Science

Feedback Survey EC

IEF: ~60% participation — last day is today!

Informed Consent Form: Still available for ~1 more week

Surveys are important! Please do them and get points!

Exam FRQ Regrade Requests

Use the regrade request form! *← exam page*

Go to office hours to go over exam!

Write a clear explanation for why you disagree with grade

Exam 1 regrade request deadline: 11/02/26

Exam 2 regrade request deadline: 11/18/26

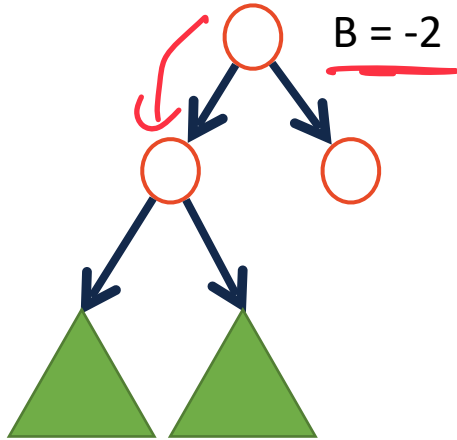


Learning Objectives

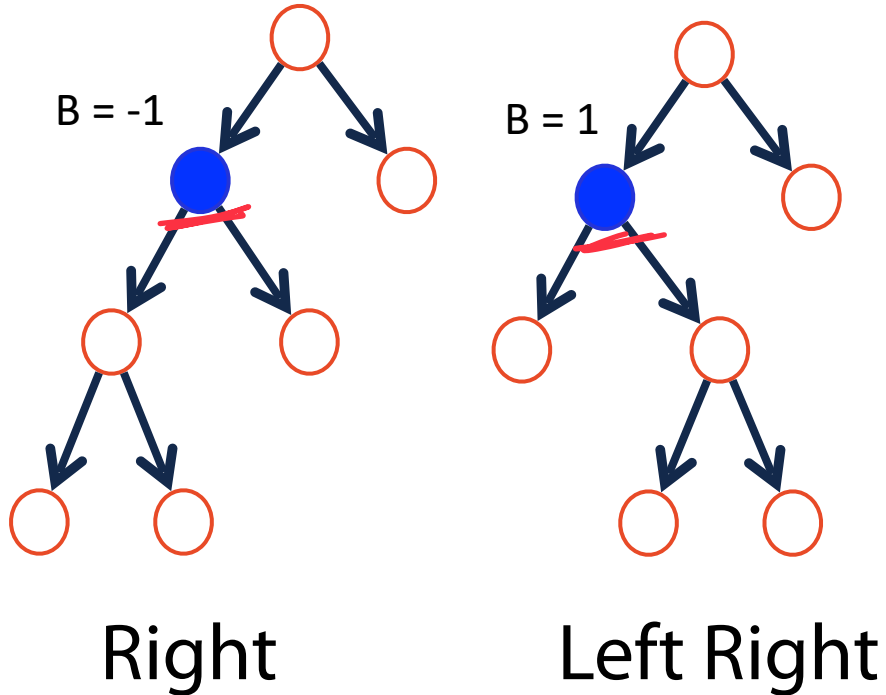
Review AVL trees

Prove that the AVL Tree speeds up all operations

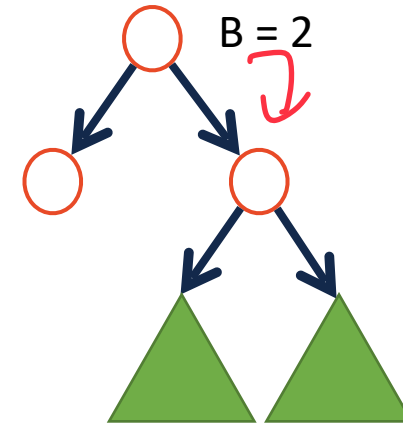
Left Imbalance



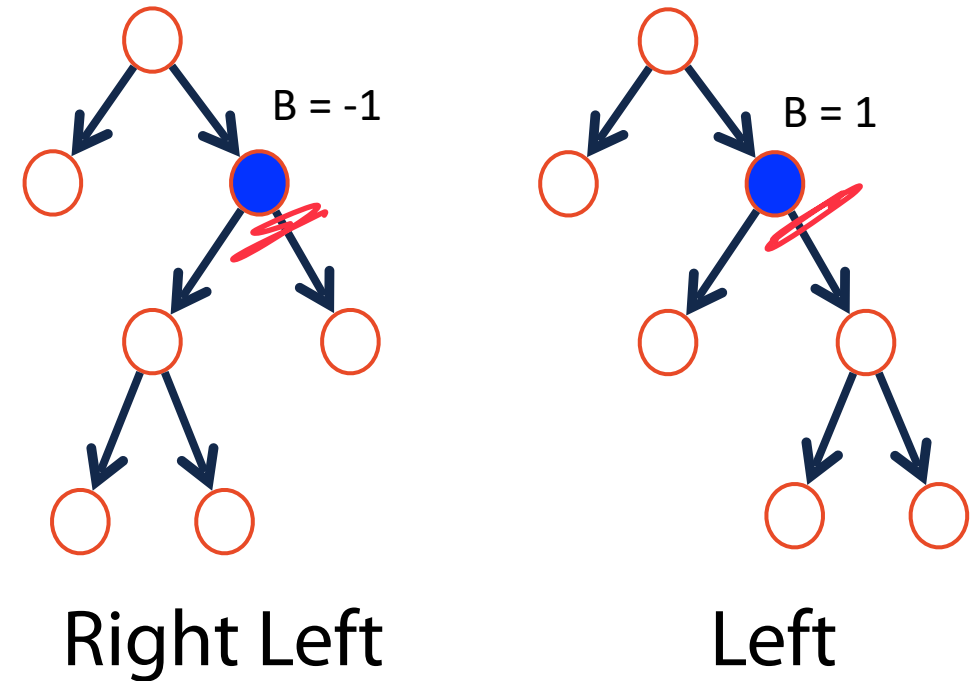
Check left subtree:



Right Imbalance



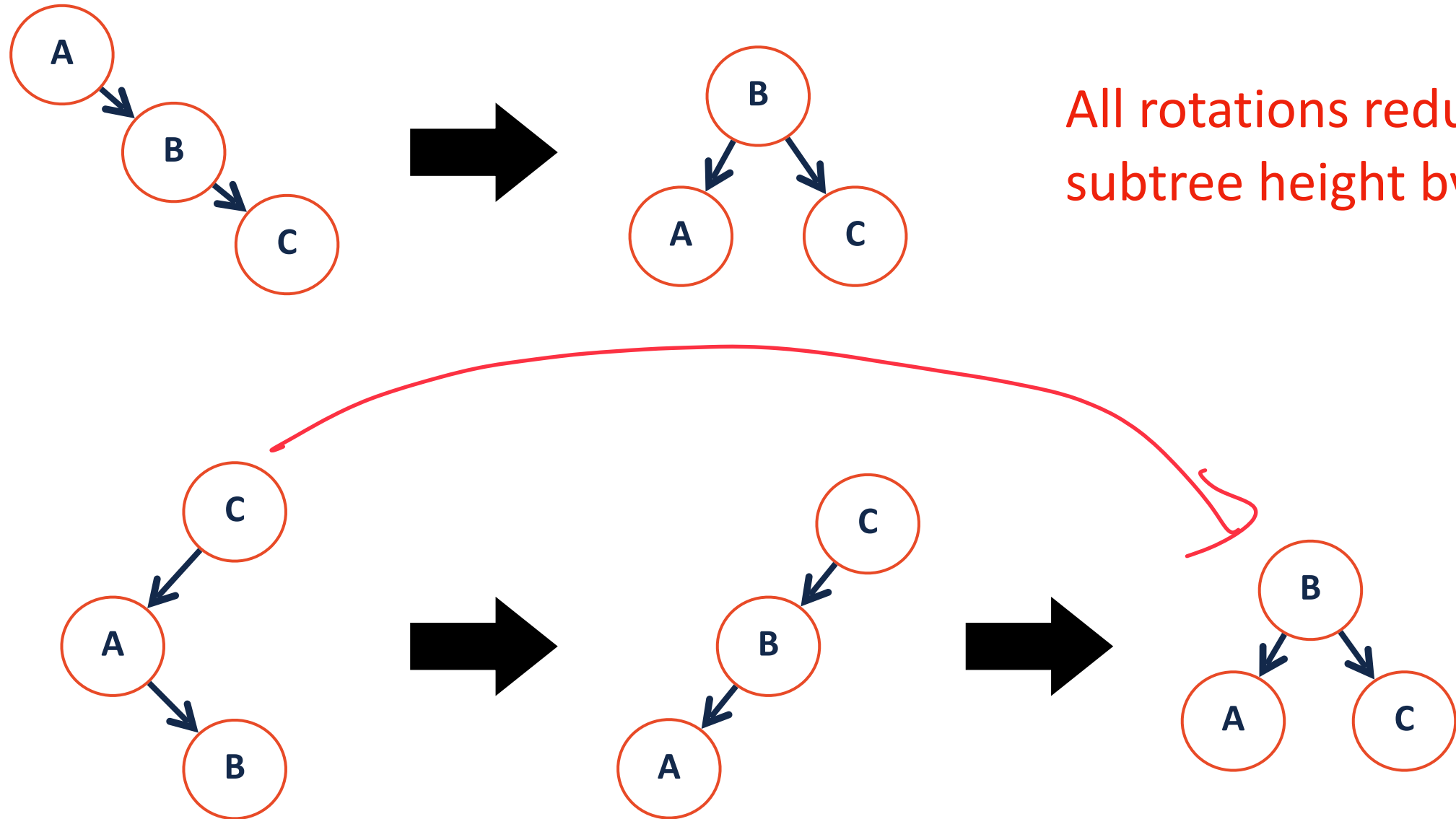
Check right subtree:



AVL Tree Rotations

All rotations are $O(1)$

All rotations reduce subtree height by one



AVL Tree Analysis

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: **$O(\log n)$**

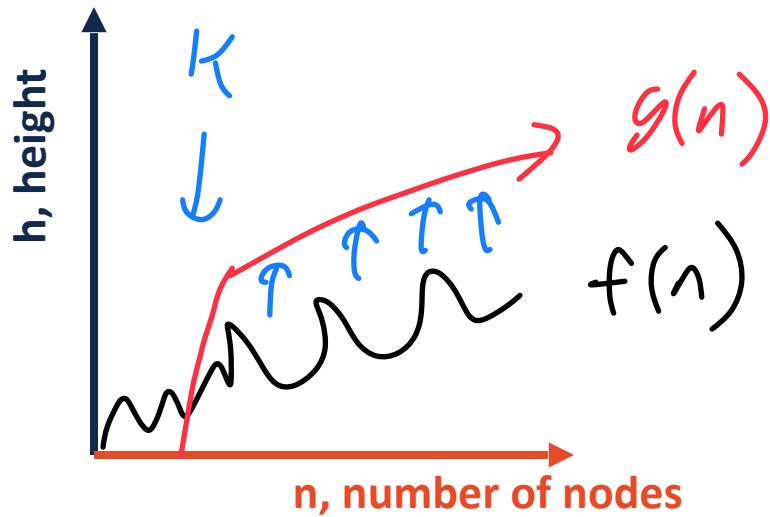


AVL Tree Analysis

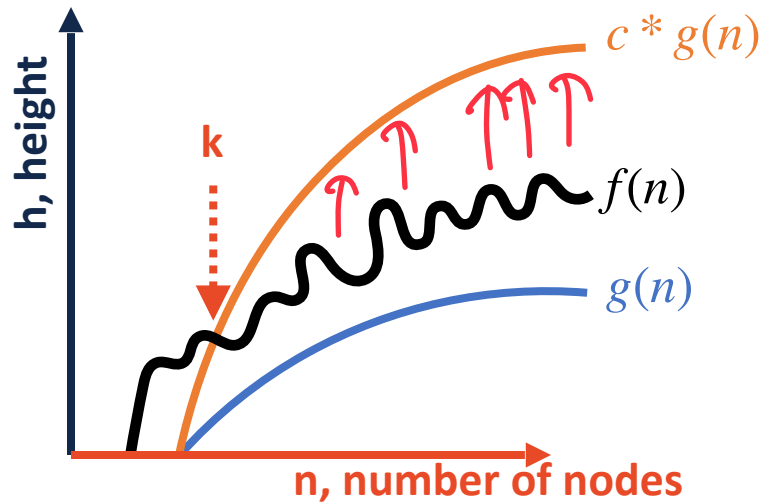
Definition of big-O:

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

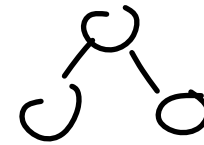
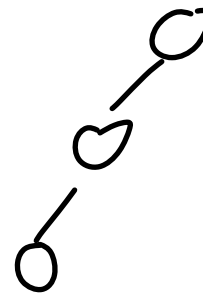
...or, with pictures:



AVL Tree Analysis

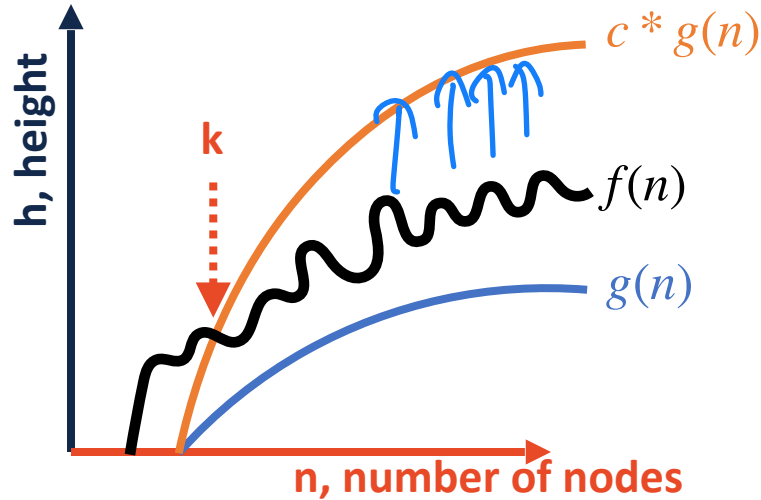


height of tree given n nodes

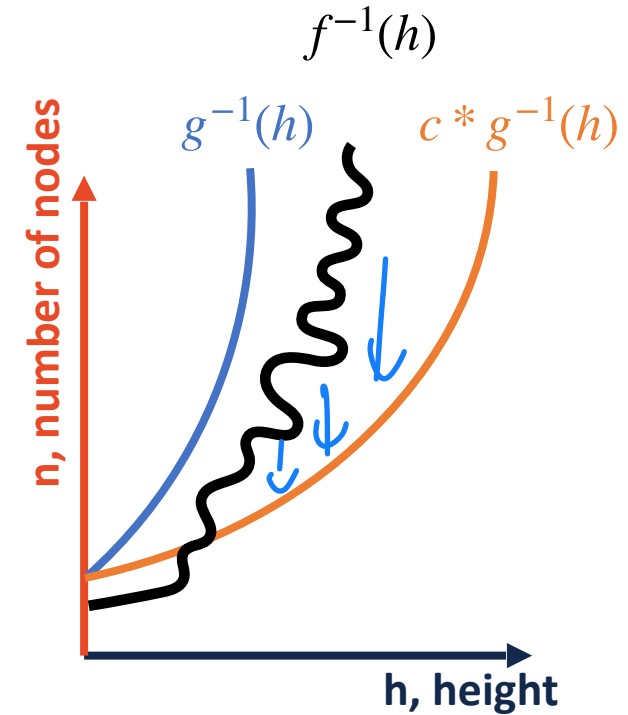


The height of the tree, $f(n)$, will always be less than $c \times g(n)$ for all values where $n > k$.

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 greater than $c \times g^{-1}(h)$ for all values where $n > k$.

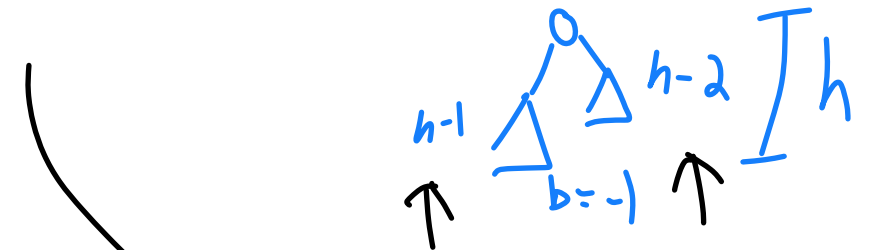
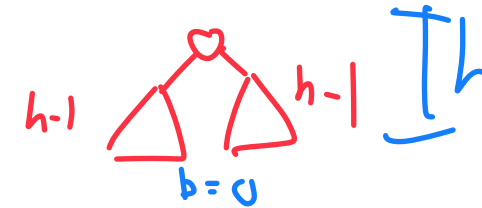


Plan of Action

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

$$\begin{aligned} &= 1 + 2N(h-1) && 42\% \leftarrow \\ &= 1 + 2N(h-2) \times && 10\% \\ \text{Best} \rightarrow &= 1 + N(h-1) + N(h-2) && 30\% \leftarrow \\ &= 2N(h-1) && 8\% \\ &= N(h-1) + N(h-2) && -\% \end{aligned}$$



This shape implies proof by induction

Simplify the Recurrence

$$N(h) = \cancel{1} + N(h - 1) + N(h - 2)$$

Simplify the Recurrence

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

$$N(h) > \underline{N(h - 1)} + N(h - 2)$$

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

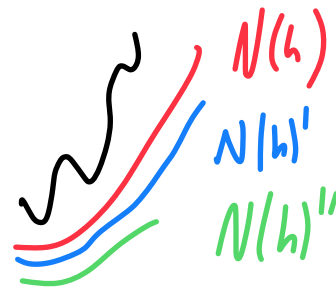
 $N(h)$

Simplify the Recurrence

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

$$N(h)^{\textcolor{blue}{|}} > N(h - 1) + N(h - 2)$$

$$N(h)^{\textcolor{green}{''}} > 2N(h - 2)$$



Simplify the Recurrence

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

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

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

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

1) Know characteristic equation? Get answer immediately!

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

Simplify the Recurrence

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

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

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

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

$$h - 2k = 0, \quad k = h/2$$

$$\rightarrow N(h) = 2^{h/2}$$

Simplify the Recurrence

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

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

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

$$2) \text{ Unroll: } N(h) > 2N(h - 2) = 2(2(N(h - 4))) = 2^k (N(h - 2k))$$

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

Simplify the Recurrence

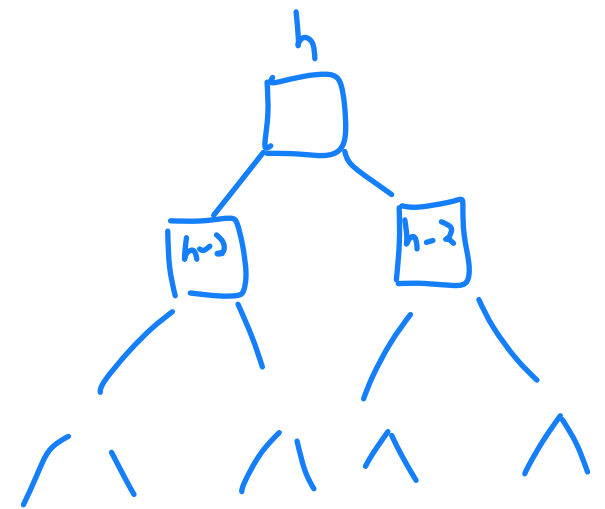
$$N(h) = 1 + N(h - 1) + N(h - 2)$$

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

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

3) Intuit approximate shape from recursion

if $N(h-1) \geq \frac{h}{2}$



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

State a Theorem

h is at least $h=1$

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

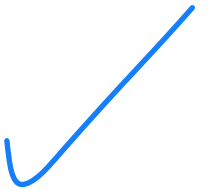
Proof by Induction:

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

II. Base Case: $h=0$
 $h=1$

$N=2$
 $2^{0/2}=1$ X

 or 2 nodes

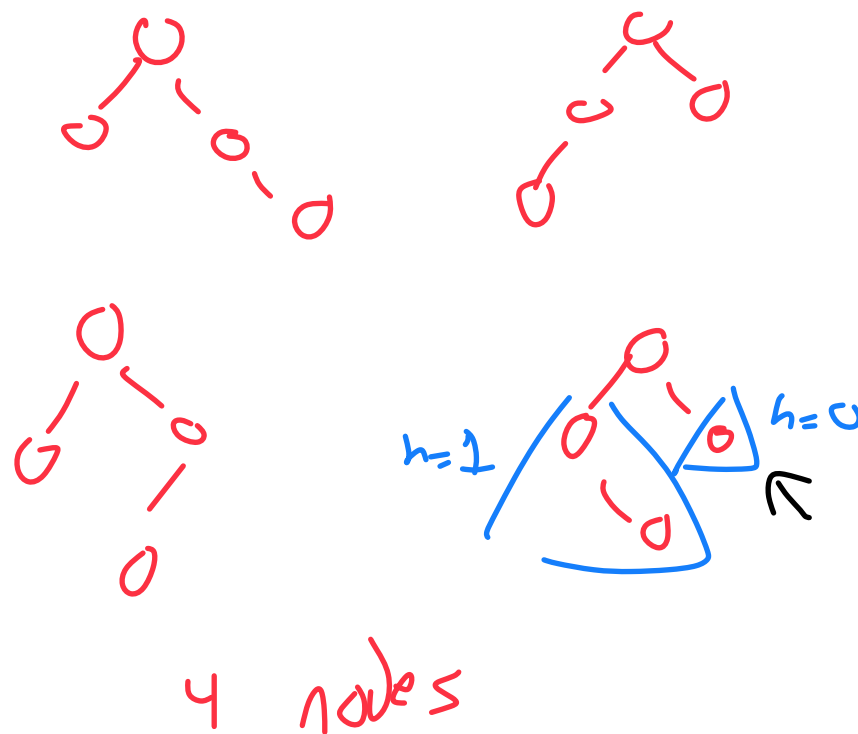
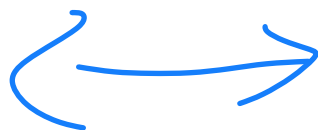


An AVL tree of height 2 has at least 4 nodes.

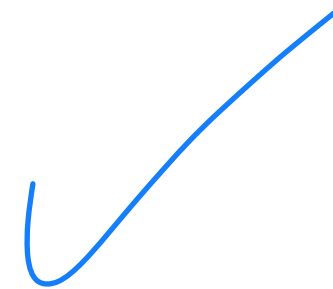
Prove a Theorem

III. Base Case: $h = 2$

$$\begin{aligned} N(2) &= 1 + N(h-1) + N(h-2) \\ &= 1 + 2 + 1 \\ &= 4 \end{aligned}$$



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



An AVL tree of height 4 has at least 2 nodes.

Prove a Theorem

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

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

$$\begin{aligned} N(h) &= 1 + N(h-1) + N(h-2) \\ &> 2 N(h-2) \end{aligned}$$

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

plus in for $N(i)$

$$> 2 \cdot 2^{(h-2)/2} = \underline{2} \cdot \underline{2^{h/2-1}} \text{ cancels out } = 2^{h/2}$$

Prove a Theorem

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

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

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

substitute $N(h-1)$

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

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

substitute eq

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

simplify

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

Prove a Theorem

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

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

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

↳ Actual # of nodes is equal to or larger than min

$$n \geq N(h)$$

$$\log_2 n \geq \log_2 2^{h/2} \quad \text{drop the top term}$$

$$2 \times \log_2 n \geq h/2 \times 2$$

$$h \leq 2 \log n$$

Wow!

Any

AVL tree is bounded by
 h of at most $\log n$



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**
 $\uparrow$

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

$$n \geq N(h)$$

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

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

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 $O(\lg(n))$:

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

$$n > 2^{h/2}$$

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

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

$$h < 2 \times \lg(n) \quad , \text{ for } h \geq 1$$

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



Summary of Balanced BST

Best at what?

Pros:

Runtime is $O(\log n)$

↳ for every operation

↳ find specifically is nice

Nearest Neighbor / Range Search
is efficient!

when is it not good?

Cons:

Runtime is $O(\log n)$

↳ Insert is $\uparrow$

↳ Remove is $\downarrow$

Harder to implement than other trees

↳ Rotations

↳ Memory Management

↳ Balance calcs

↳ storing height

why Not O(1) find???

Summary of Balanced BST

AVL Trees



- Max height: 1.44 * lg(n)

- Rotations:

Better proof not shown!

Summary of Balanced BST

AVL Trees

- Max height: $1.44 * \lg(n)$

- Rotations:

Zero rotations on find

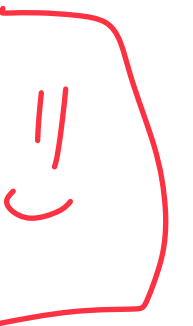
One rotation on insert

$O(h) == O(\lg(n))$ rotations on remove

Red-Black Trees

- Max height: $2 * \lg(n)$

- Constant number of rotations on insert (max 2), remove (max 3).



Red-Black Trees in C++

C++ provides us a balanced BST as part of the standard library:

```
std::map<K, V> map;
```

```
V & std::map<K, V>::operator[] ( const K & )
```

```
std::map<K, V>::erase( const K & )
```

Red-Black Trees in C++

```
iterator std::map<K, V>::lower_bound( const K & );  
iterator std::map<K, V>::upper_bound( const K & );
```

↑
Nearest neighbor search!

Every Data Structure So Far

Join Code: 225



	Unsorted Array	Sorted Array	Unsorted Linked List	Sorted Linked List	Binary Tree	BST	AVL
Find							
Insert		You fill it out! We will go over it					
Remove		in a review session!					
Traverse							

Now we must learn to pick one and justify why it is best!

Summary of Balanced BST

Pros:

- Running Time: $O(\log n)$
- Improvement Over: All lists / kinds
- Great for specific applications:

We talked about all of this
in the pro / con slide!

This was backup space for questions

Nearest Neighbor

Summary of Balanced BST

Cons:

- Running Time:
- In-memory Requirement:

Engineering vs Theory Efficiency

	Time x1 billion	Like
L1 cache reference	0.5 seconds	Heartbeat 💖
Branch mispredict	5 seconds	Yawn 🥱
L2 cache reference	7 seconds	Long yawn 🥱 🥱 🥱
Mutex lock/unlock	25 seconds	Make coffee ☕
Main memory reference	100 seconds	Brush teeth
Compress 1K bytes	50 minutes	TV show 📺
Send 2K bytes over 1 Gbps network	5.5 hours	(Brief) Night's sleep 🛏
SSD random read	1.7 days	Weekend
Read 1 MB sequentially from memory	2.9 days	Long weekend
Read 1 MB sequentially from SSD	11.6 days	2 weeks for delivery 📦
Disk seek	16.5 weeks	Semester
Read 1 MB sequentially from disk	7.8 months	Human gestation 🐣
Above two together	1 year	🌍 ☀️
Send packet CA->Netherlands->CA	4.8 years	Ph.D. 🎓

(Care of <https://gist.github.com/hellerbarde/2843375>)