

CSE 100: HASH TABLES, MARKOV, AVL

Announcements

- Midterm 1 grade published
- PA2
 - Checkpoint Deadline: 11:59pm on Tuesday, 11/6 (not slip day eligible)
 - Final Deadline: 11:59pm on Tuesday, 11/13 (slip day eligible)

Applications of Hashing – The Heavy Hitters Problem

1	2	1	1	1	2	4	1	4	2	4	1
---	---	---	---	---	---	---	---	---	---	---	---

Given a set of elements of size n , and some (much) smaller value k
determine which elements occur at least n/k times.

In the example above for:

$n=12$

$k=3$

Which elements should be returned?

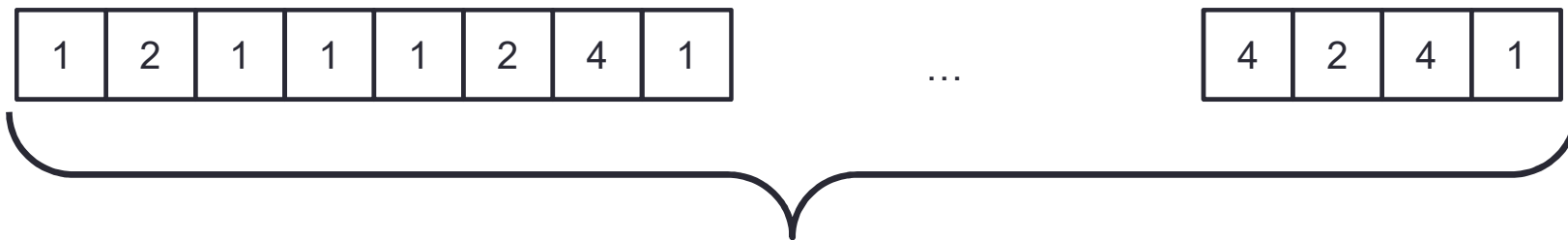
A. {1}

B. {1, 4}

C. {1, 4, 2}

D. {}

The Heavy Hitters Problem

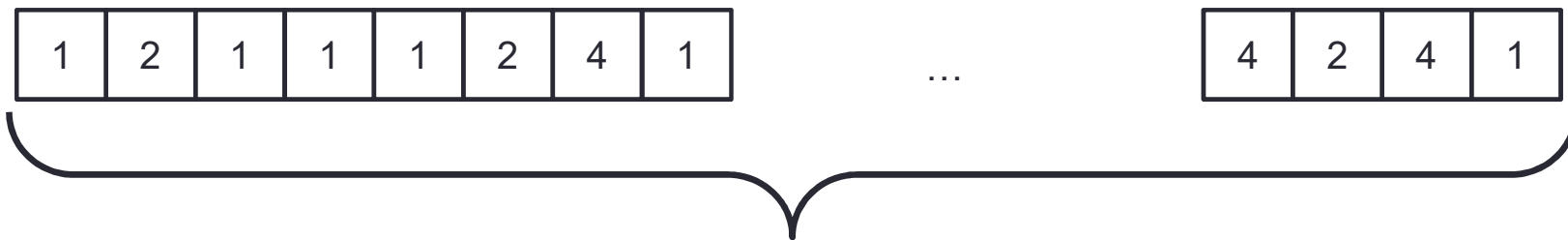


n is generally VERY LARGE (millions or billions)
You can't store it in memory, or even perhaps on disk!

Given a set of elements of size n , and some (much) smaller value k
determine which elements occur at least n/k times.

What are the real-world applications of this problem?

The Heavy Hitters Problem: A sad fact



n is generally VERY LARGE (millions or billions)
You can't store it in memory, or even perhaps on disk!

Given a set of elements of size n , and some (much) smaller value k
determine which elements occur at least n/k times.

There is no single pass algorithm that can solve this problem in a sublinear amount of auxiliary space!

Building Toward Count-Min Sketches: Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k
determine which elements occur at least n/k times.

1	35	10	1	1	10	1	35	1	35	16	1
1	3	2	1	1	2	1	3	1	3	0	1

Example:
 $n=12$
 $k=2$

Idea: Use a small Hash Map (of size $b \ll n$) to store counts

Example: $b = 4$, $H(k) = k \bmod 4$

1	6	2	3
---	---	---	---

What's the problem here?

Building Toward Count-Min Sketches: Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k
determine which elements occur at least n/k times.

1	35	10	17	9	14	1	35	17	35	16	9
---	----	----	----	---	----	---	----	----	----	----	---

$H(k) = k \bmod 4$: 1 3 2 1 1 2 1 3 1 3 0 1

Example:
 $n=12$
 $k=2$

Idea: Use a small Hash Map (of size $b \ll n$) to store counts

Example: $b = 4$, $H(k) = k \bmod 4$

1	6	2	3
---	---	---	---

Any ideas on how to improve this?

Count-Min Sketch: Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k
determine which elements occur at least n/k times.

1	35	10	17	9	14	1	35	17	35	16	9
---	----	----	----	---	----	---	----	----	----	----	---

Example:
 $n=12$
 $k=2$

Idea: Use ℓ small Hash Maps (of size $b \ll n$) to store counts

Example: $\ell=3$, $b=4$

$$H_0(k) = (k \% 5) \% 4$$

$$H_1(k) = (k \% 7) \% 4$$

$$H_2(k) = (k \% 13) \% 4$$

0

1

2

3

Count in each hash function, advance slides to get to the example

Count-Min Sketch:

Idea: Use ℓ small Hash Maps (of size b much less than n) to store counts

Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k determine which elements occur at least n/k times.

Example:
 $n=12$
 $k=2$

	1	35	10	17	9	14	1	35	17	35	16	9
$H_0(k) = (k\%5) \% 4$												
$H_1(k) = (k\%7) \% 4$												
$H_2(k) = (k\%13) \% 4$												

Have to compute hash for each number...

Example: $\ell=3$, $b=4$

$H_0(k) = (k\%5) \% 4$				
$H_1(k) = (k\%7) \% 4$				
$H_2(k) = (k\%13) \% 4$				
	0	1	2	3

Count-Min Sketch:

Idea: Use ℓ small Hash Maps (of size b much less than n) to store counts

Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k determine which elements occur at least n/k times.

Example:
 $n=12$
 $k=2$

	1	35	10	17	9	14	1	35	17	35	16	9
$H_0(k) = (k\%5) \% 4$	1	0	0	2	0	0	1	0	2	0	1	0
$H_1(k) = (k\%7) \% 4$	1	0	3	3	2	0	1	0	3	0	2	2
$H_2(k) = (k\%13) \% 4$	1	1	2	0	1	1	1	1	0	1	3	1

Start tallying and stop ~1/2 done

Example: $\ell=3$, $b=4$

$H_0(k) = (k\%5) \% 4$				
$H_1(k) = (k\%7) \% 4$				
$H_2(k) = (k\%13) \% 4$				
	0	1	2	3

Count-Min Sketch:

Idea: Use ℓ small Hash Maps (of size b much less than n) to store counts

Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k determine which elements occur at least n/k times.

Example:
 $n=12$
 $k=2$

	1	35	10	17	9	14	1	35	17	35	16	9
$H_0(k) = (k\%5) \% 4$	1	0	0	2	0	0	1	0	2	0	1	0
$H_1(k) = (k\%7) \% 4$	1	0	3	3	2	0	1	0	3	0	2	2
$H_2(k) = (k\%13) \% 4$	1	1	2	0	1	1	1	1	0	1	3	1

Example: $\ell=3$, $b = 4$

$H_0(k) = (k\%5) \% 4$	7	3	2	0
$H_1(k) = (k\%7) \% 4$	4	2	3	3
$H_2(k) = (k\%13) \% 4$	2	8	1	1
	0	1	2	3

What is the estimate for 1?

- A. 1
- B. 2
- C. 3
- D. 4
- E. Other

Count-Min Sketch:

Idea: Use ℓ small Hash Maps (of size b much less than n) to store counts

Sacrificing Exact Solutions for Large Savings!

Given a set of elements of size n , and some (much) smaller value k determine which elements occur at least n/k times.

Example:
 $n=12$
 $k=2$

	1	35	10	17	9	14	1	35	17	35	16	9
$H_0(k) = (k\%5) \% 4$	1	0	0	2	0	0	1	0	2	0	1	0
$H_1(k) = (k\%7) \% 4$	1	0	3	3	2	0	1	0	3	0	2	2
$H_2(k) = (k\%13) \% 4$	1	1	2	0	1	1	1	1	0	1	3	1

Example: $\ell=3$, $b = 4$

$H_0(k) = (k\%5) \% 4$	7	3	2	0
$H_1(k) = (k\%7) \% 4$	4	2	3	3
$H_2(k) = (k\%13) \% 4$	2	8	1	1
	0	1	2	3

Which of the following values are overestimated?

- A. 10
- B. 17
- C. 9
- D. 16
- E. More than one of these

Which value is overestimated by the most?

Count-Min Sketch

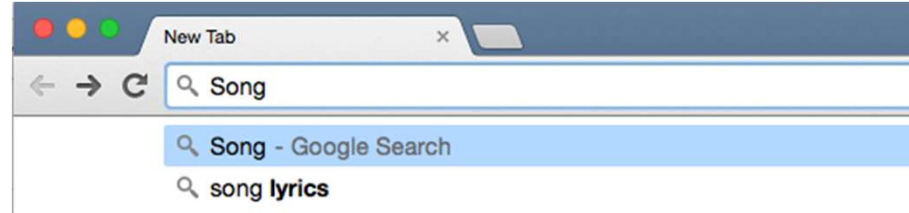
- ℓ (number of hash functions) and b (size of hash tables) can be much smaller than n . Size of storage depends on k not n !
- Updating in the table is fast: $O(1)$, assuming hash functions are fast
- There is a very high probability that the overestimate of a number will be "small"
- Count-Min Sketches can solve an approximation of the Heavy Hitters problem (think about how)

PA2 background

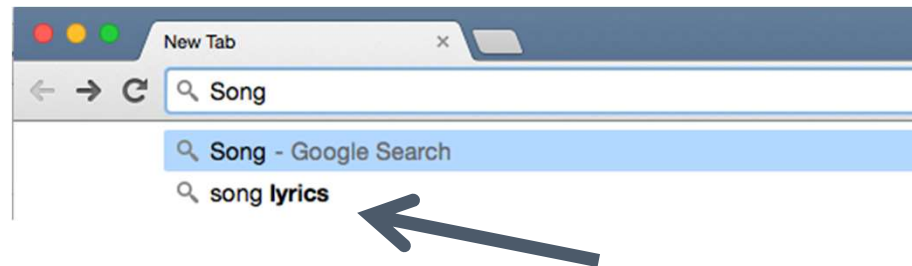
- Explain how Markov processes can be used to generate text
- Distinguish between training on text and generating text
- Create Markov models to generate text

Note: Punctuation is part of a word in my examples here. In the assignment, you may be asked to separate out ",", ". ", "!" among others.

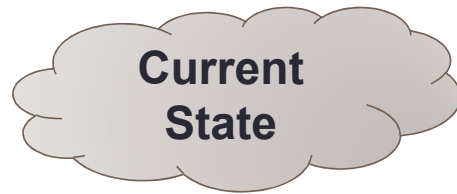
Predicting the future



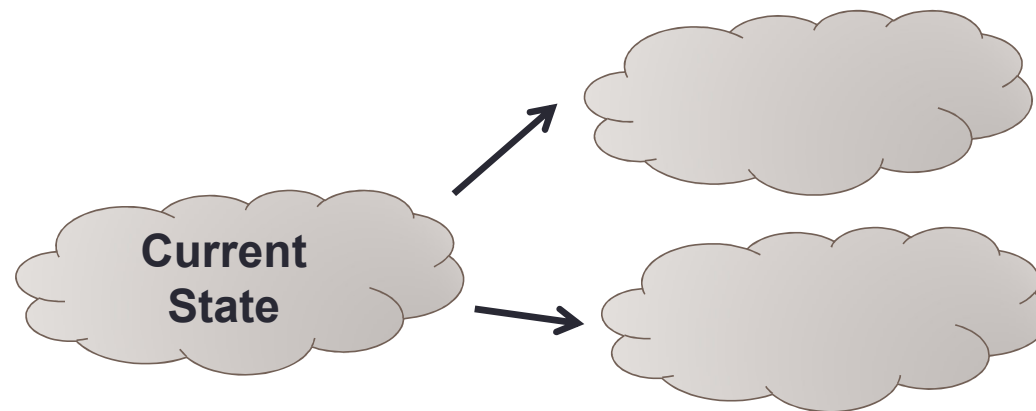
Predicting the future



Predicting the future: Markov processes

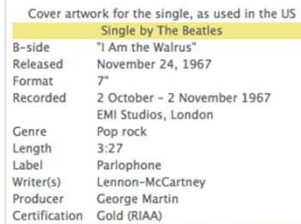



Predicting the future: Markov processes



SONG

You say yes, I say no,
You say stop and I say go, go, go.





**You say hello. I don't know why you say hello,
hello. I say goodbye. Oh no. You say no, You**

Browser window showing the Beatles website: www.thebeatles.com/song/hello-goodbye

THE BEATLES | EXPLORE | ALBUMS | SONGS | NEWS | STORE | LOGIN/

Hello, Goodbye

SONG

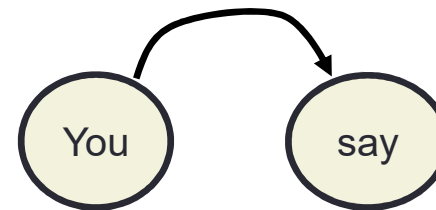
Oh yes, I say no,
You say stop, and I say go, go, go,
Oh yes,
You say goodbye and I say hello, hello, hello,
I don't know what you say goodbye, I say hello,
hello, hello,
I don't know what you say goodbye, I say hello.
I say high, you say low,
You say why, and I say I don't know.
Oh yes,
You say goodbye and I say hello, hello, hello,
I don't know what you say goodbye, I say hello,
hello, hello,
I don't know what you say goodbye, I say hello.
Why, why, why, why, why, why,
Do you say goodbye.
Oh yes,
You say goodbye and I say hello, hello, hello,
I don't know what you say goodbye, I say hello,
hello, hello,
I don't know what you say goodbye, I say hello.
Oh yes,
You say yes, I say no,
You say stop and I say go, go, go.

WIKIPEDIA
The Free Encyclopedia
"Hello, Goodbye"

THE BEATLES
Hello Goodbye
I Am The Walrus
Album

Cover artwork for the single, as used in the US

Single by The Beatles	
B-side	"I Am the Walrus"
Released	November 24, 1967
Format	7"
Recorded	2 October – 2 November 1967
	EMI Studios, London
Genre	Pop rock
Length	3:27
Label	Parlophone
Writer(s)	Lennon–McCartney
Producer	George Martin
Certification	Gold (RIAA)



Browser window showing the Beatles website: www.thebeatles.com/song/hello-goodbye

THE BEATLES

EXPLORE ALBUMS SONGS NEWS STORE LOGIN/

Hello, Goodbye

SONG

You say yes, I say no,
You say stop, and I say go, go, go,

WIKIPEDIA
The Free Encyclopedia

**You say hello. I don't know why you say hello,
hello. I say goodbye. Oh no. You say no, You**

I say high, you say low,
You say why, and I say I don't know.
Oh no.
You say goodbye and I say hello, hello, hello.
I don't know why you say goodbye, I say hello,
hello, hello,
I don't know why you say goodbye, I say hello.

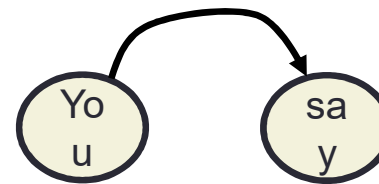
Why, why, why, why, why, why,
Do you say goodbye.
Oh no.
You say goodbye and I say hello, hello, hello.
I don't know why you say goodbye, I say hello,
hello, hello,
I don't know why you say goodbye, I say hello.

You say yes, I say no,
You say stop and I say go, go, go.

Cover artwork for the single, as used in the US

Single by The Beatles

B-side	"I Am the Walrus"
Released	November 24, 1967
Format	7"
Recorded	2 October – 2 November 1967
	EMI Studios, London
Genre	Pop rock
Length	3:27
Label	Parlophone
Writer(s)	Lennon–McCartney
Producer	George Martin
Certification	Gold (RIAA)



Browser: Hello, Goodbye | The Beatles | www.thebeatles.com/song/hello-goodbye

THE BEATLES | EXPLORE | ALBUMS | SONGS | NEWS | STORE | LOGIN/

Hello, Goodbye

SONG


You say yes, I say no,
 You say stop, and I say go, go, go,
 Oh no.
 You say goodbye and I say hello, hello, hello,
 I don't know why you say goodbye, I say hello,
 hello, hello,
 I don't know why you say goodbye, I say hello.

I say high, you say low,
 You say why, and I say I don't know.
 Oh no.
 You say goodbye and I say hello, hello, hello.
 I don't know why you say goodbye, I say hello,
 hello, hello,
 I don't know why you say goodbye, I say hello.

Why, why, why, why, why, why,
 Do you say goodbye.
 Oh no.
 You say goodbye and I say hello, hello, hello.
 I don't know why you say goodbye, I say hello,
 hello, hello,
 I don't know why you say goodbye, I say hello.

You say yes, I say no,
 You say stop and I say go, go, go.

WIKIPEDIA
 The Free Encyclopedia
 "Hello, Goodbye"



Cover artwork for the single, as used in the US

Single by The Beatles

B-side	"I Am the Walrus"
Released	November 24, 1967
Format	7"
Recorded	2 October – 2 November 1967
	EMI Studios, London
Genre	Pop rock
Length	3:27
Label	Parlophone
Writer(s)	Lennon–McCartney
Producer	George Martin
Certification	Gold (RIAA)

Suppose "hello." is followed by:
 "I", "I", "Why", "I", "You", and "You"

What should be the Markov Model for the word "hello."?

[Hello, Goodbye | The Beatles](#)
[www.thebeatles.com/song/hello-goodbye](#)

THE BEATLES
 EXPLORE ALBUMS SONGS NEWS STORE LOGIN/

Hello, Goodbye

SONG


You say yes, I say no,
 You say stop, and I say go, go, go,
 Oh no.
 You say goodbye and I say hello, hello, hello,
 I don't know why you say goodbye, I say hello,
 hello, hello,
 I don't know why you say goodbye, I say hello.

I say high, you say low,
 You say why, and I say I don't know.
 Oh no.
 You say goodbye and I say hello, hello, hello.
 I don't know why you say goodbye, I say hello,
 hello, hello,
 I don't know why you say goodbye, I say hello.

Why, why, why, why, why, why,
 Do you say goodbye.
 Oh no.
 You say goodbye and I say hello, hello, hello.
 I don't know why you say goodbye, I say hello,
 hello, hello,
 I don't know why you say goodbye, I say hello.

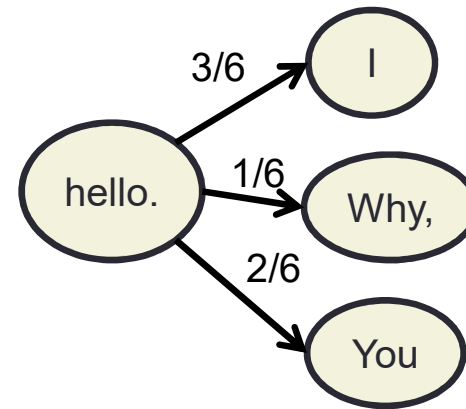
You say yes, I say no,
 You say stop and I say go, go, go.

WIKIPEDIA
The Free Encyclopedia
"Hello, Goodbye"



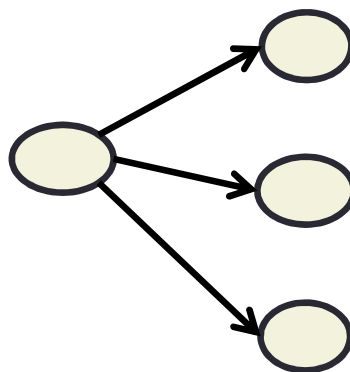
Cover artwork for the single, as used in the US

Single by The Beatles	
B-side	"I Am the Walrus"
Released	November 24, 1967
Format	7"
Recorded	2 October – 2 November 1967
	EMI Studios, London
Genre	Pop rock
Length	3:27
Label	Parlophone
Writer(s)	Lennon–McCartney
Producer	George Martin
Certification	Gold (RIAA)



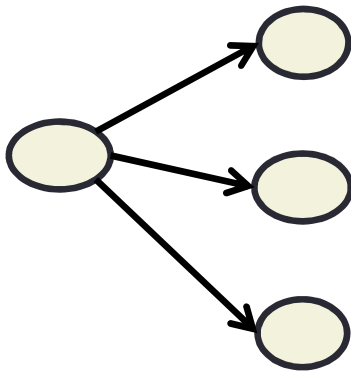
Stage 1: Train

Build model based on data



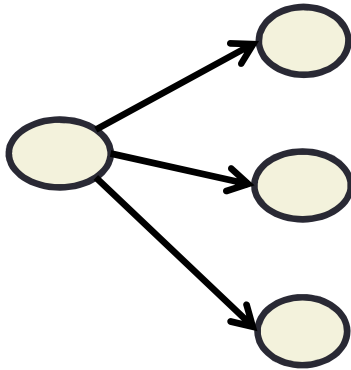
Stage 1: Train

Build model based on data input String



Stage 1: Train

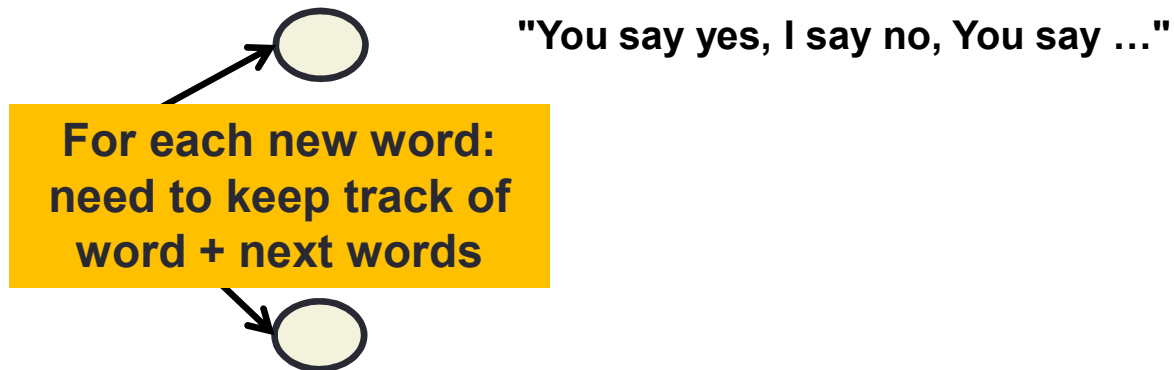
Build model based on data input String



"You say yes, I say no, You say ..."

Stage 1: Train

Build model based on data input String



Stage 1: Train

"You"	"say"	"yes,"	"I"	"say"	"no,"	"You"	"say"
-------	-------	--------	-----	-------	-------	-------	-------



Stage 1: Train

For each new word:
need to keep track of
word + next words

"You"	"say"	"yes,"	"I"	"say"	"no,"	"You"	"say"
-------	-------	--------	-----	-------	-------	-------	-------

word → "You"

nextWords ➔ "say"

Stage 1: Train

For each new word:
need to keep track of
word + next words

"You"	"say"	"yes,"	"I"	"say"	"no,"	"You"	"say"
-------	-------	--------	-----	-------	-------	-------	-------

word → "You"
nextWords → "say"

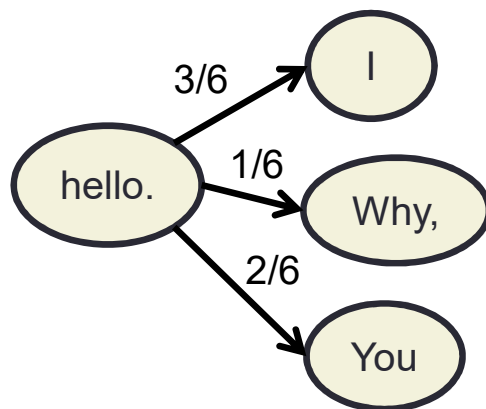
word → "say"
nextWords → "yes,"

Keep going through the
next "say" to show
duplicates

What about
probabilities?

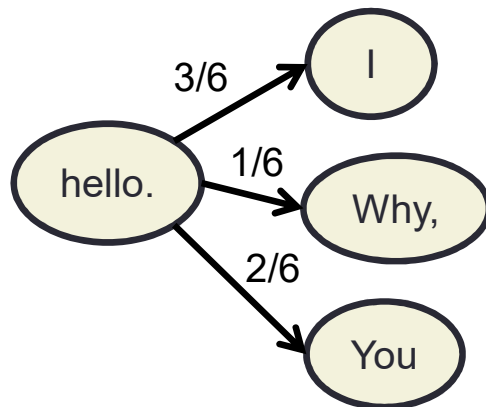
Stage 1: Train

"I don't know why you say goodbye, I say hello. I say high, you say low, ...



Stage 1: Train

... I say hello, hello, hello. I don't know why you say goodbye ...



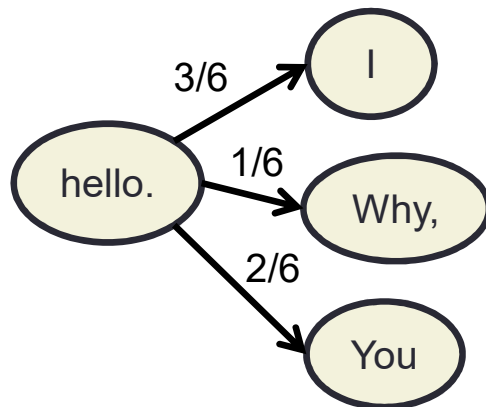
word → "hello."
"

nextWords → "I", "I"

What about
probabilities?

Stage 1: Train

... I say hello. Why, why, why, why, why, why, ...

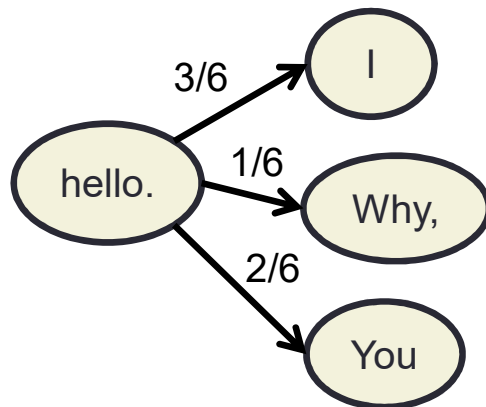


word → "hello."
" "
nextWords ➤ "I", "I", "Why,"

What about
probabilities?

Stage 1: Train

... hello, hello, hello. I don't know ...

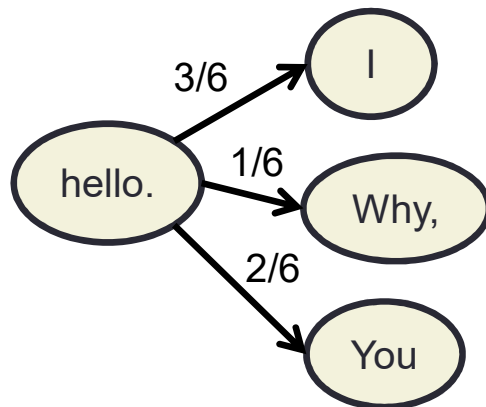


word → "hello."
" "
nextWords → "I", "I", "Why,", "I"

What about
probabilities?

Stage 1: Train

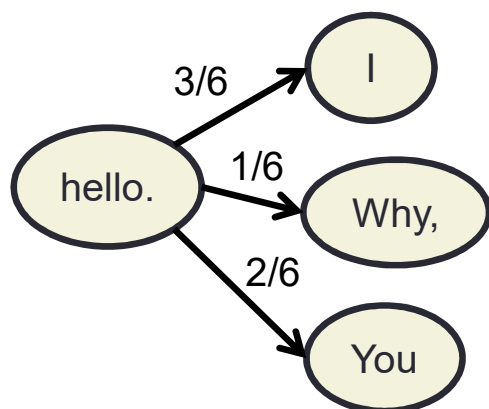
... I say hello. You say yes, I say no, ...



word → "hello."
" "
nextWords → "I", "I", "Why,", "I", "You"

Stage 1: Train

etc.

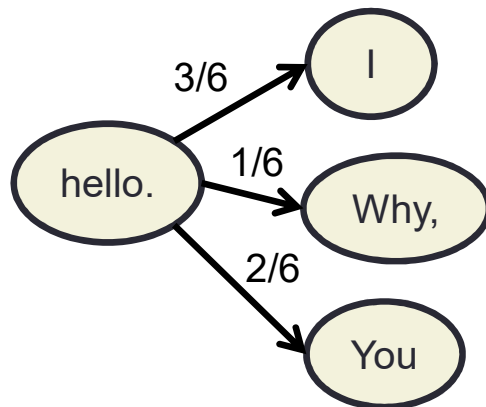


What about
probabilities?

word → "hello."
" "
nextWords → "I", "I", "Why,", "I", "You",
"You"

Stage 2: Generate

What about probabilities?



word → "hello."
" "
nextWords → "I", "I", "Why,", "I", "You",
"You"

Pick Randomly
But we could do better
than a list, right?

Stage 2: Generate

Until we have enough words:

- Find current word as the word of some node in present word list
- Generate a random number between 0 and the size of
nextWords list of this node
- Print the word at that index,
- Repeat

Given the text below, build your structure

"I like my cat. I like to see my dog. My dog likes to see me."

Note – treat a "." as a separate word in this case.

Given the text below, build your structure

"I like my cat. I like to see my dog. My dog likes to see me."

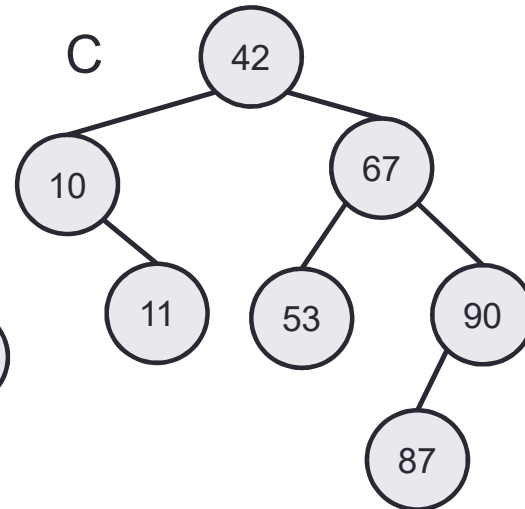
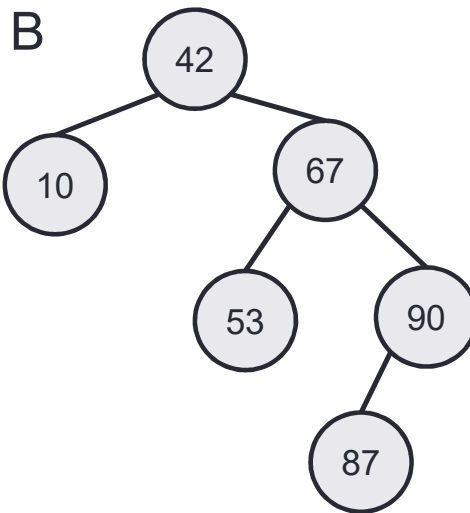
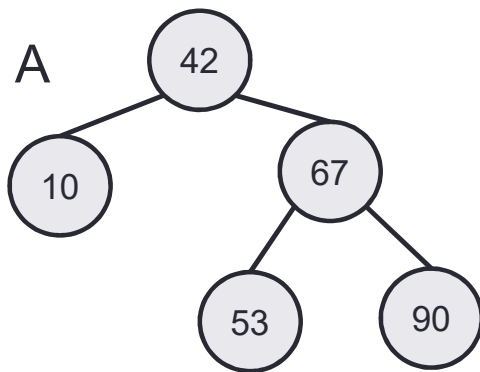
Note – treat a "." as a separate word in this case.

Which of the following Strings could not have been created with the text above?

- A. I like to see my cat.
- B. I like to see me.
- C. I like my cat.
- D. My dog likes to see my cat.
- E. All of the above are possible.

Which of the following is/are balanced trees?

And thus can become AVL trees by adding the balance factors



D. A&C

E. A&B&C

Annotate the trees with balance factors

An AVL Tree is *worst case* $O(\log N)$ to find an element!

How would you prove this?

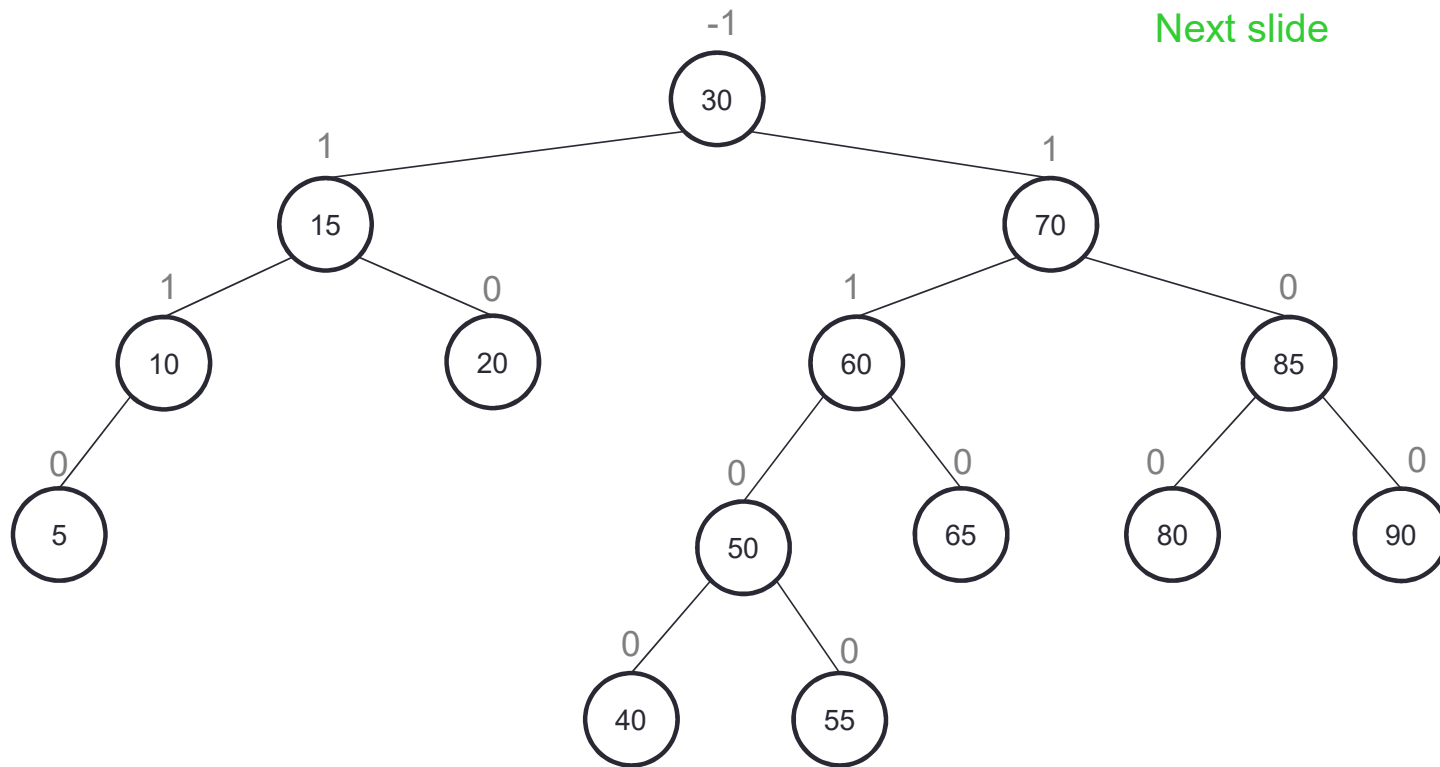
Come up with a formula that shows that the height of the tallest AVL tree with N nodes is never bigger than $c \cdot \log N + k$, for some c and k (assuming large N).

The key to this proof is showing that the height stays “small”, no matter how legally “unbalanced” the tree is.

But how does the tree stay balanced??

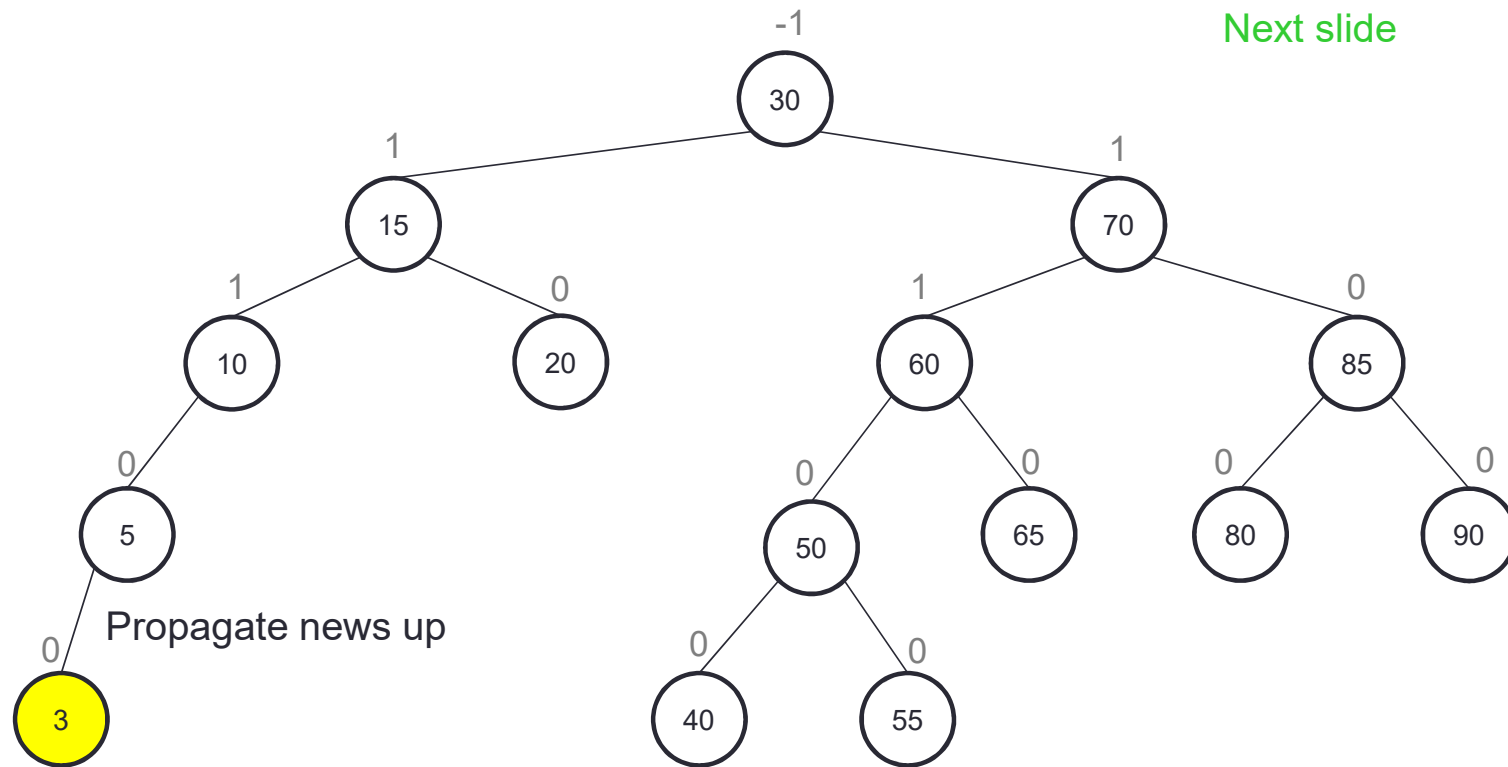
Inserting and rebalancing

Next slide



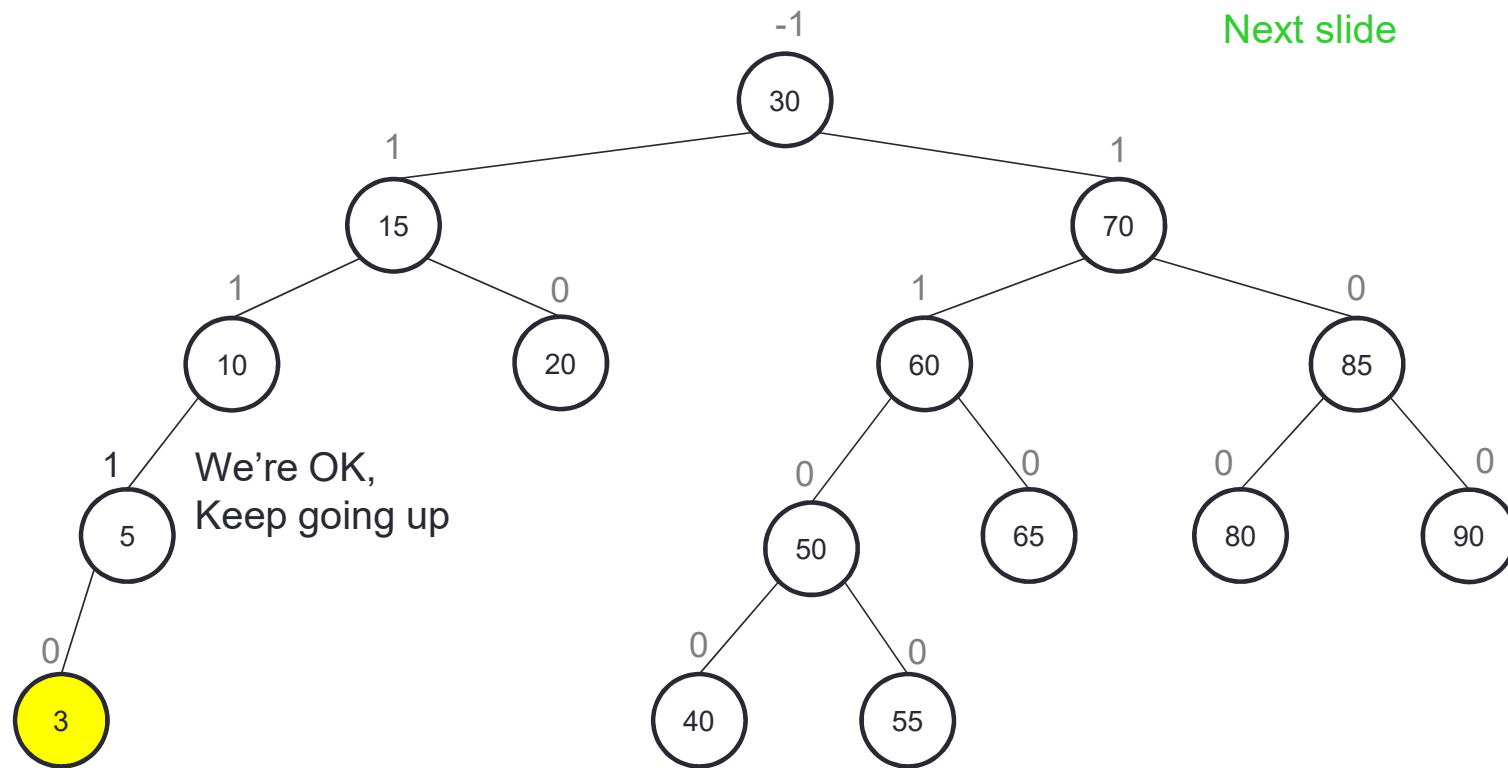
Insert 3

Inserting and rebalancing



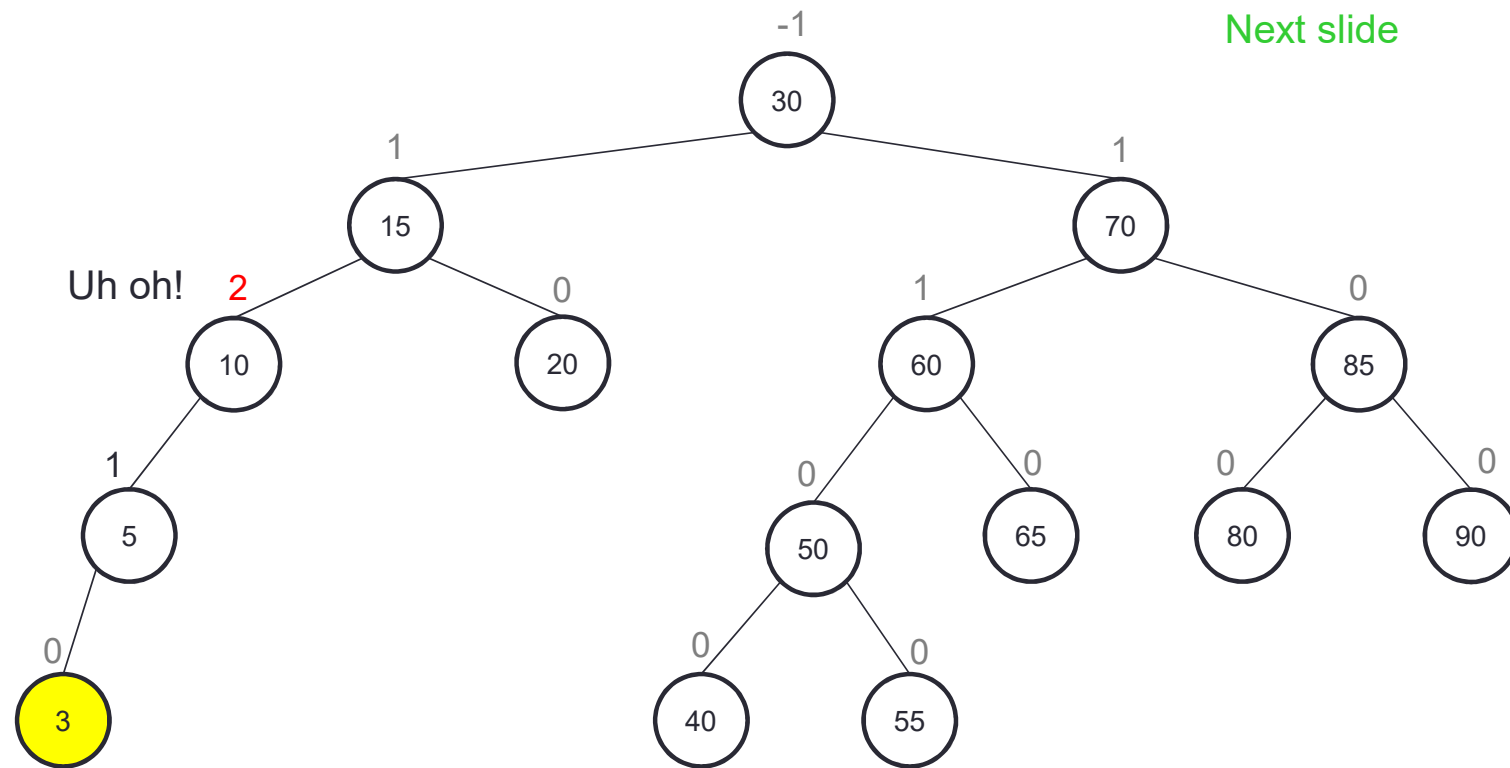
Insert 3

Inserting and rebalancing



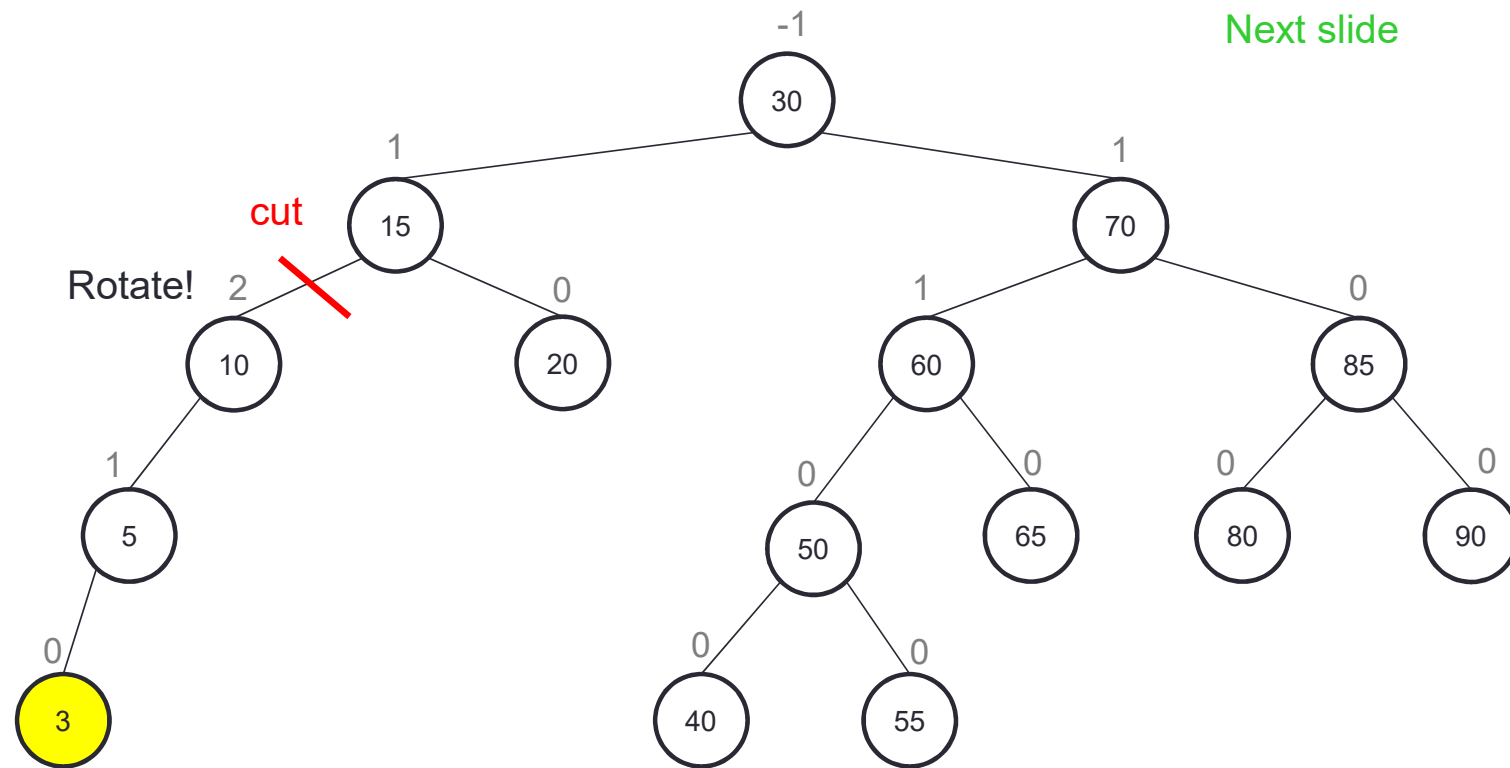
Insert 3

Inserting and rebalancing

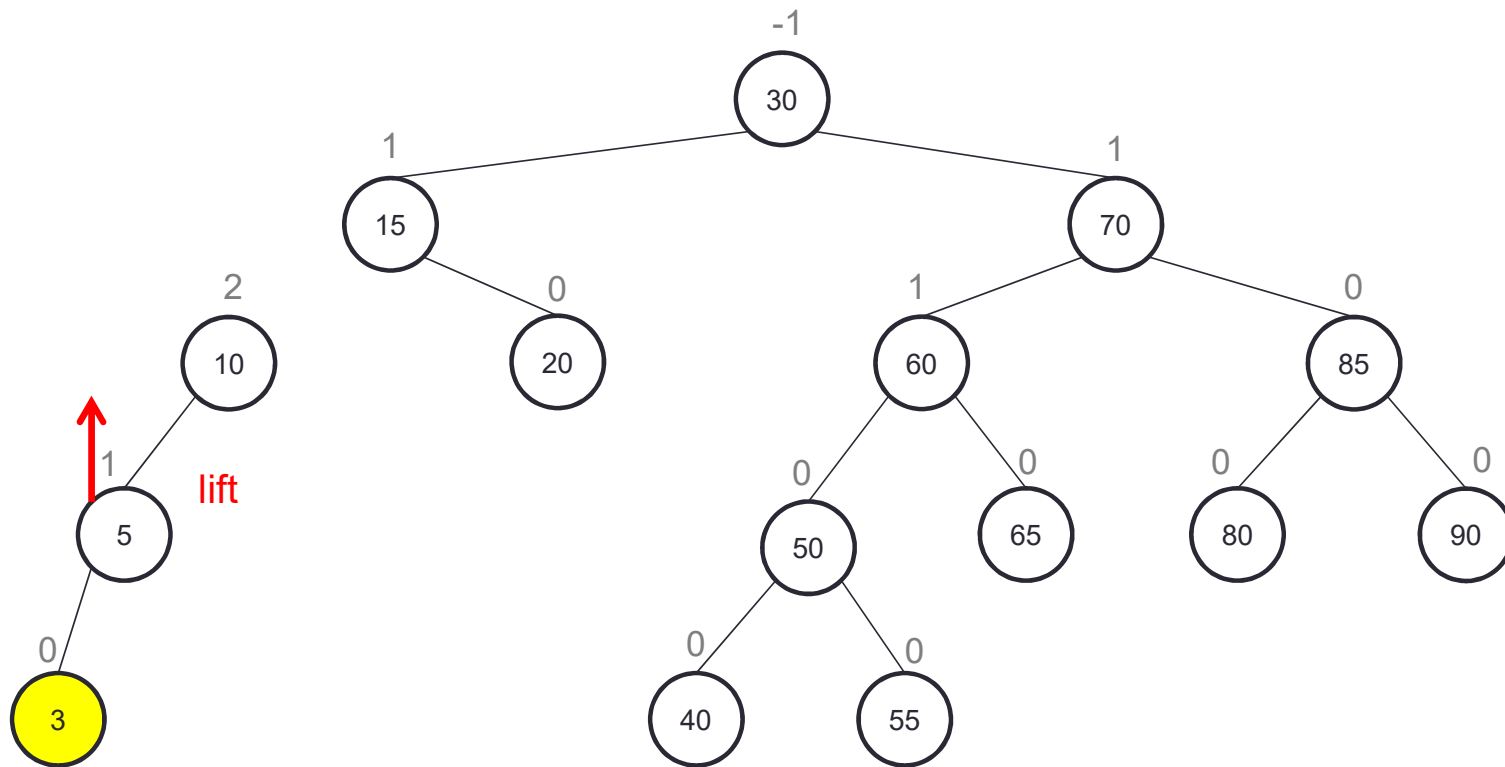


Insert 3

Inserting and rebalancing

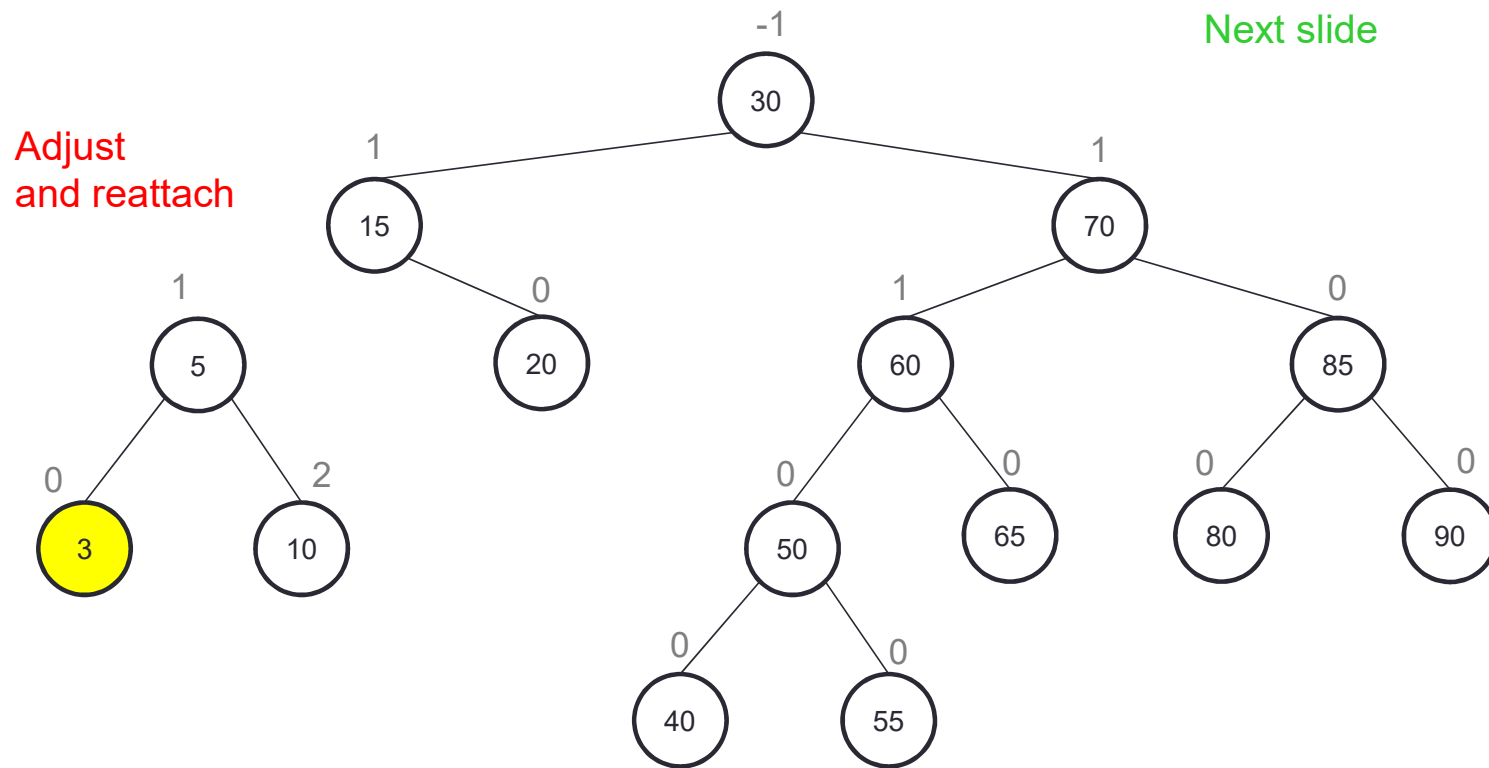


Inserting and rebalancing



Insert 3

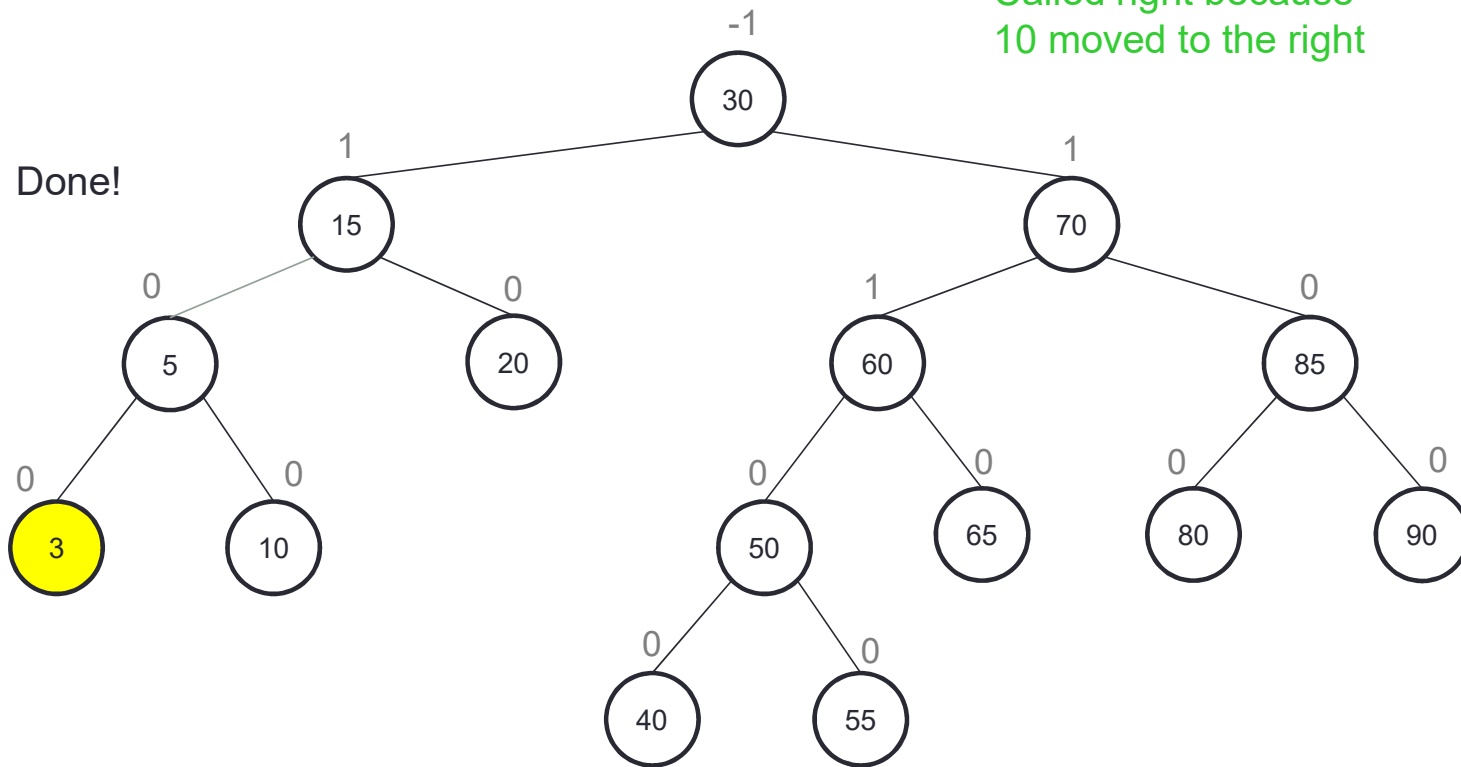
Inserting and rebalancing



Insert 3

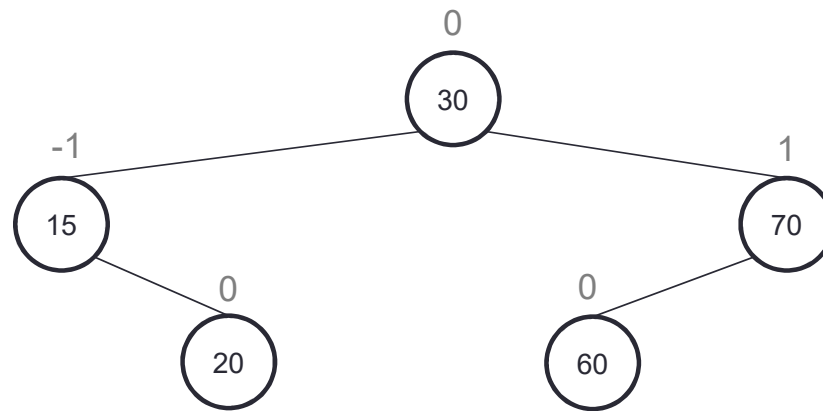
We just did a single right rotation at 10

Called right because
10 moved to the right



Insert 3

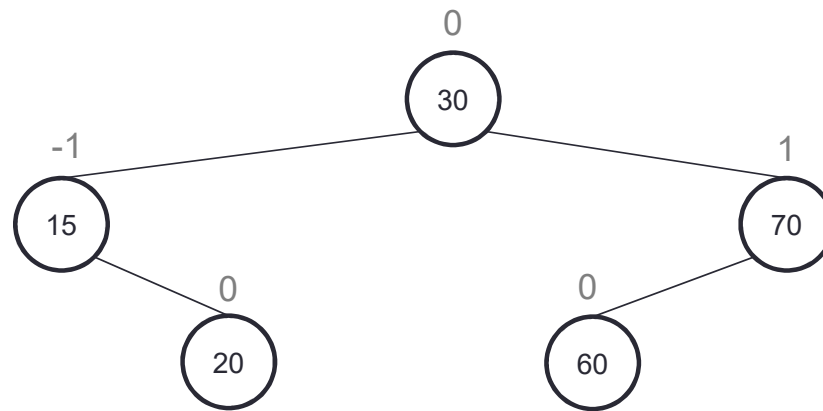
Single rotation practice



What could you insert into this AVL tree that would result in a single rotation?

- A. 71
- B. 10
- C. 50
- D. 66

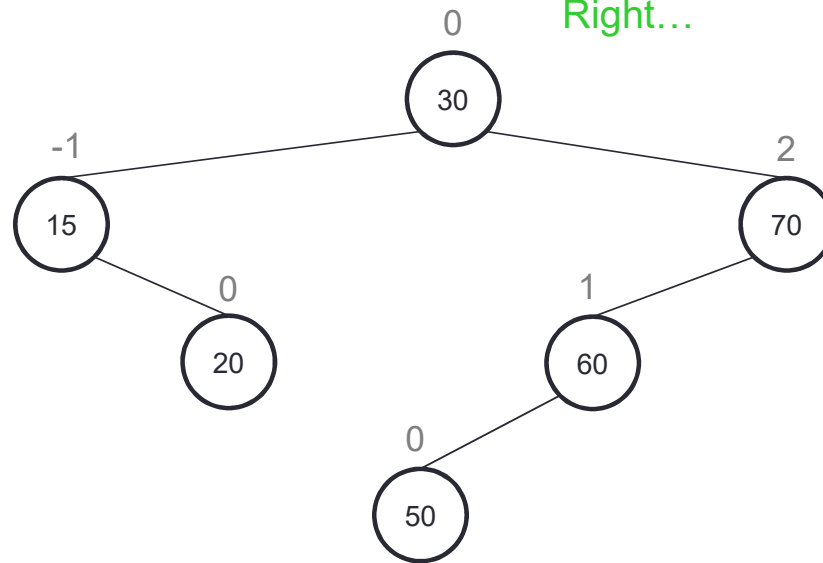
Single rotation practice



Insert 50. Draw the resulting AVL tree. (Don't peek)

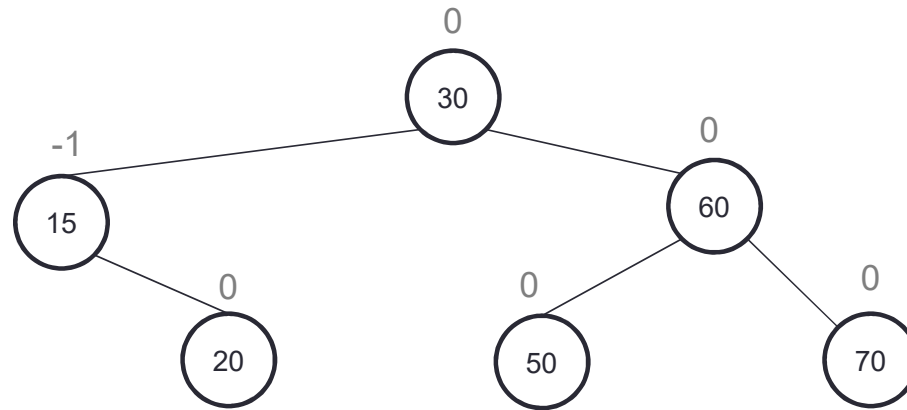
Single rotation practice

What rotation do we need to do?
Right...



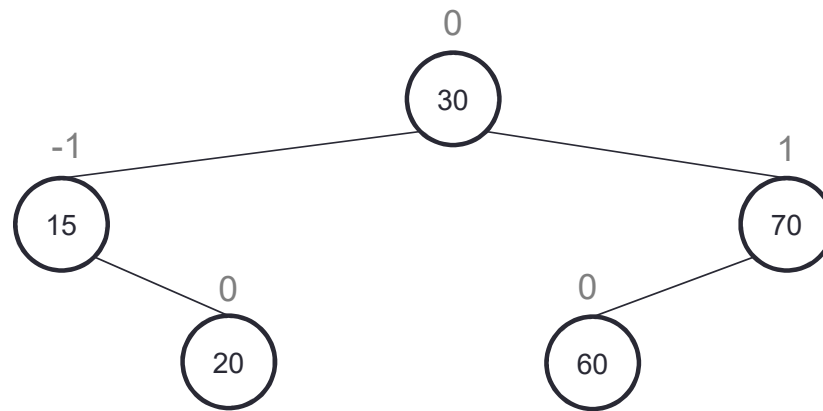
After insertion

Single rotation practice



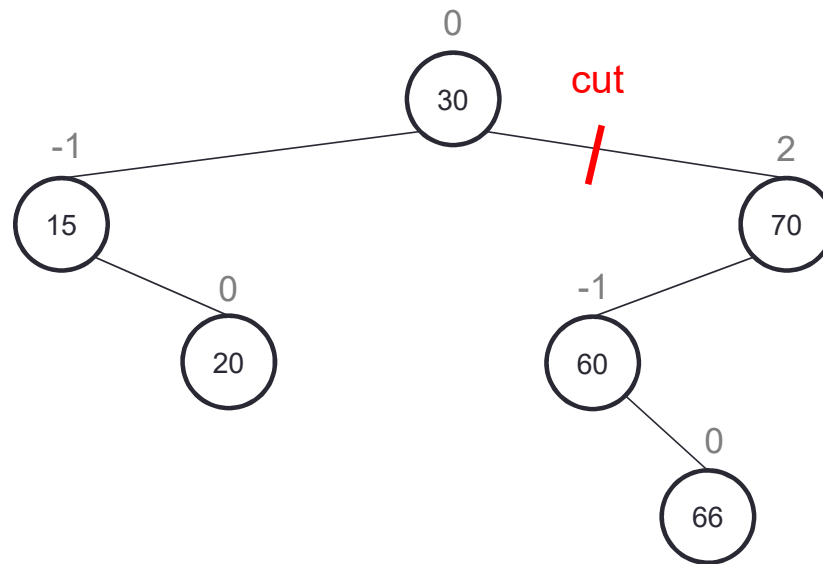
After rotation

Single rotation is not enough



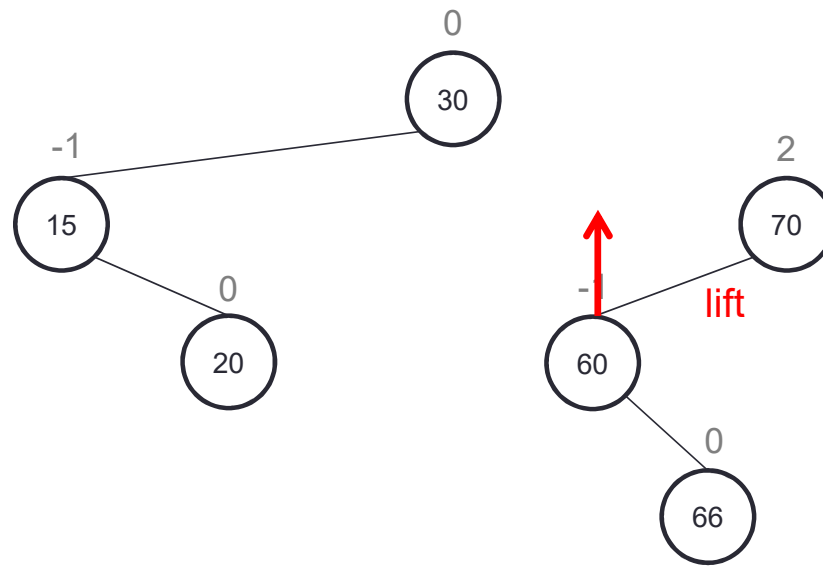
What happens if we insert 66?

Single rotation is not enough

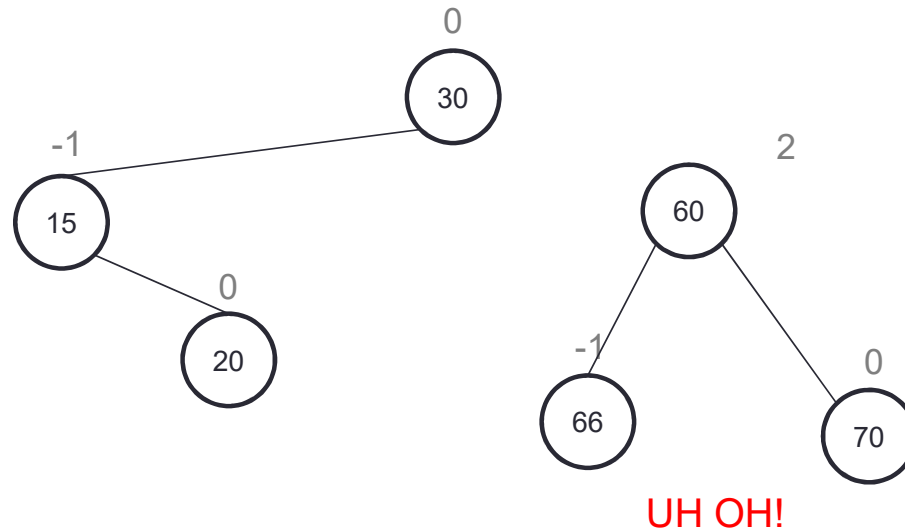


Why won't a single rotation work? Try it.

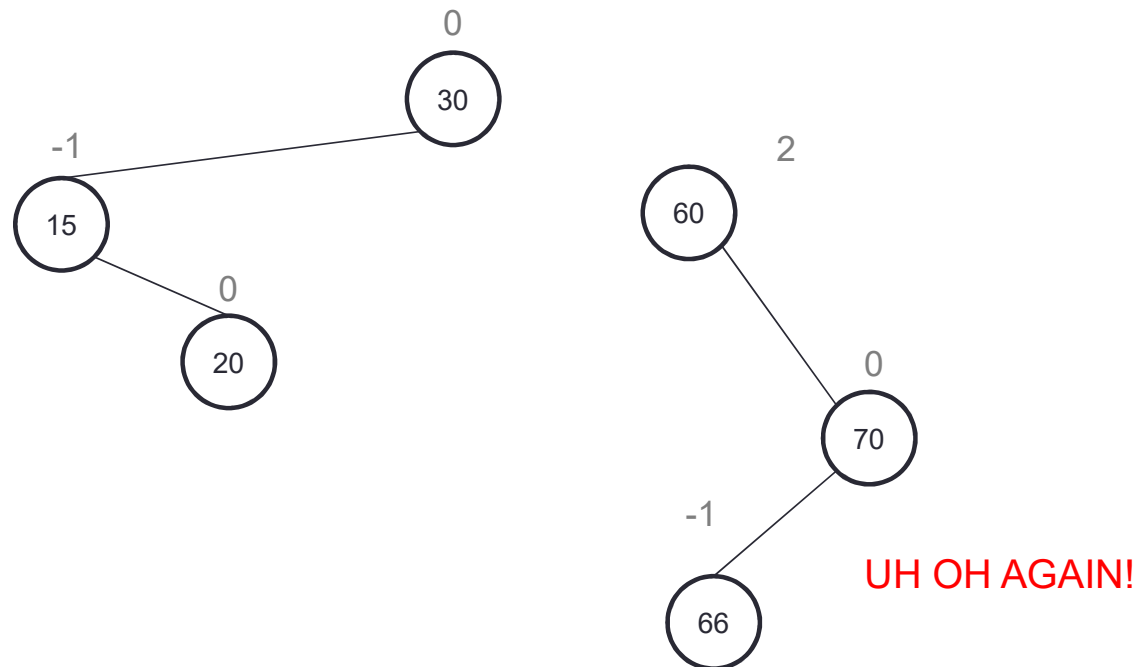
Single rotation is not enough



Single rotation is not enough



Single rotation is not enough

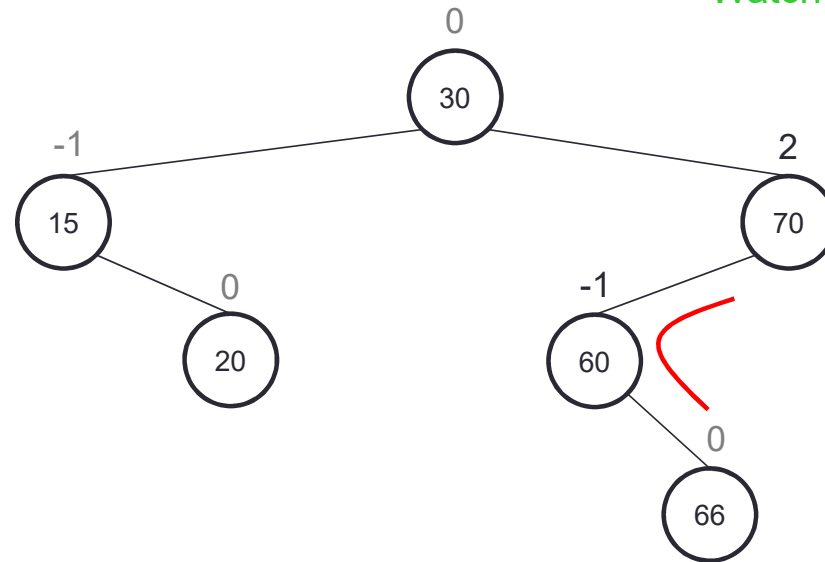


Reattaching 66 here will always work with respect to the BST properties, and we know that 66 will always fit here because 60 used to be 70's left child.

The problem is that this won't fix the balance issue!

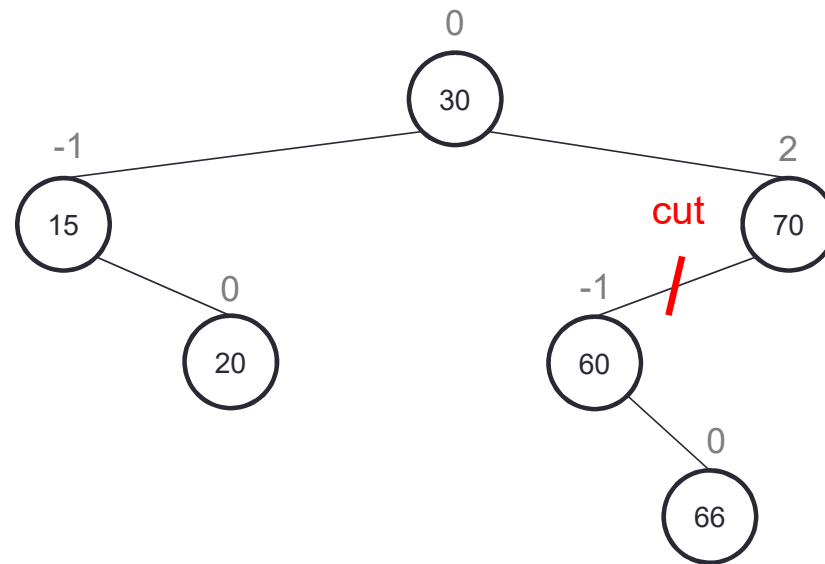
Double rotation to the rescue

Watch for this curve/kink



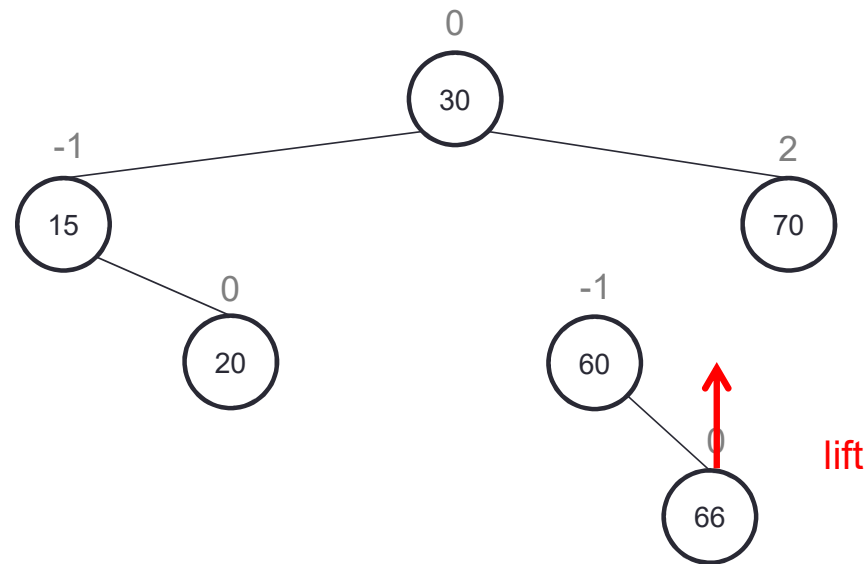
Single rotations only work to balance the tree when involved nodes are “in a line”
This is not the case here. We want 66 to be the top node, not 60.
So we will first rotate left at 60 to get 66 in the middle, then we can rotate right at 70.

Double rotation to the rescue



Single rotations only work when involved nodes are “in a line”
So we will first rotate left at 60, then we can rotate right at 70.

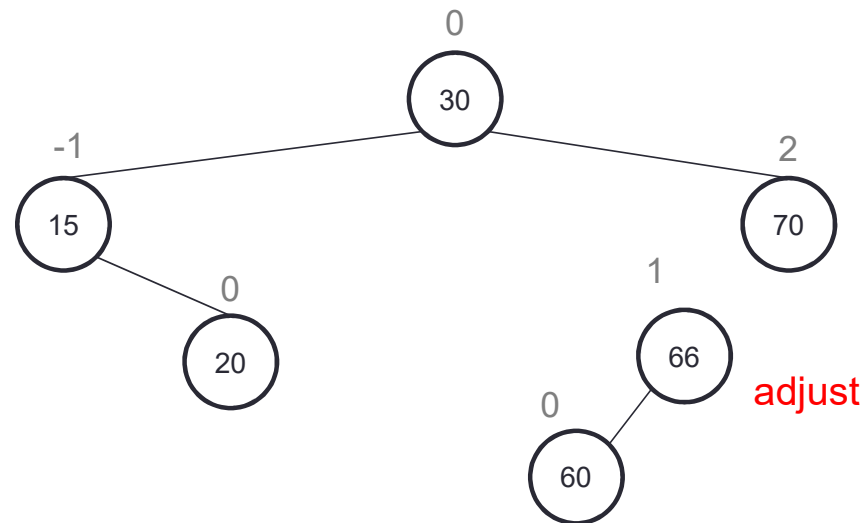
Double rotation to the rescue



Single rotations only work when involved nodes are “in a line”

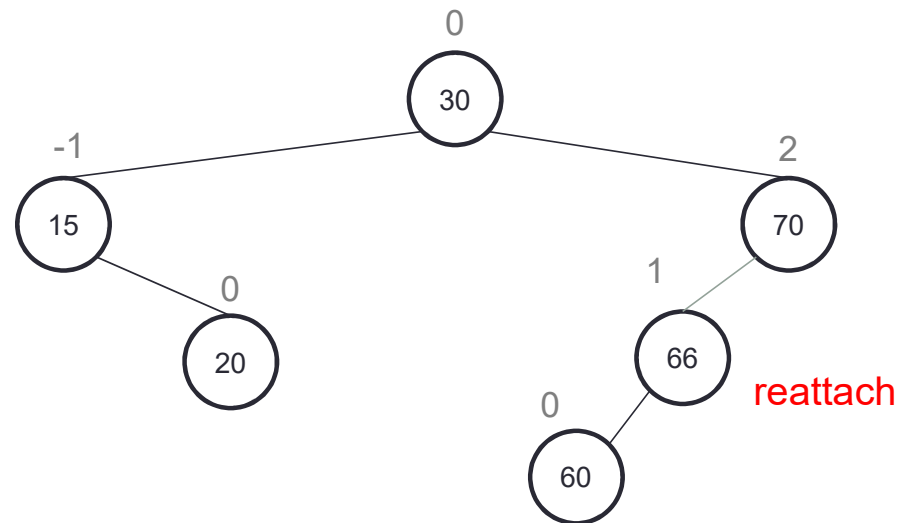
So we will first rotate left around 60, then we can rotate right around 70.

Double rotation to the rescue



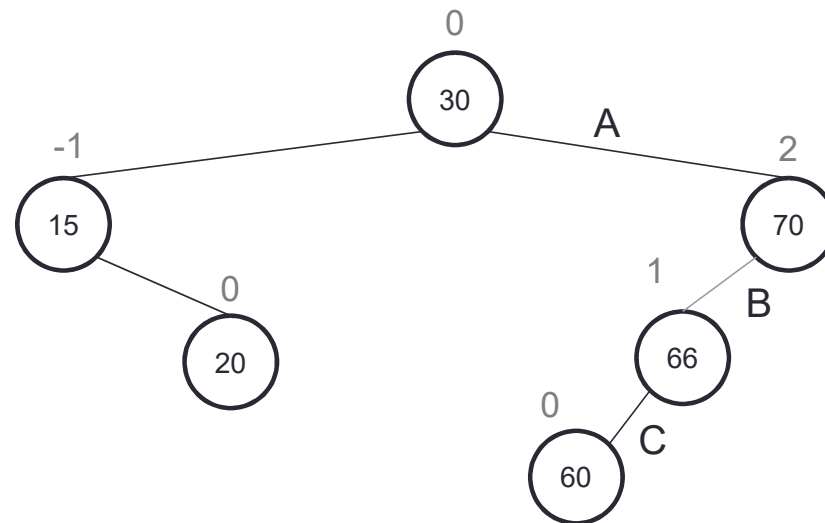
Single rotations only work when involved nodes are “in a line”
So we will first rotate left at 60, then we can rotate right at 70.

Double rotation to the rescue



Single rotations only work when involved nodes are “in a line”
So we will first rotate left at 60, then we can rotate right at 70.

Double rotation to the rescue

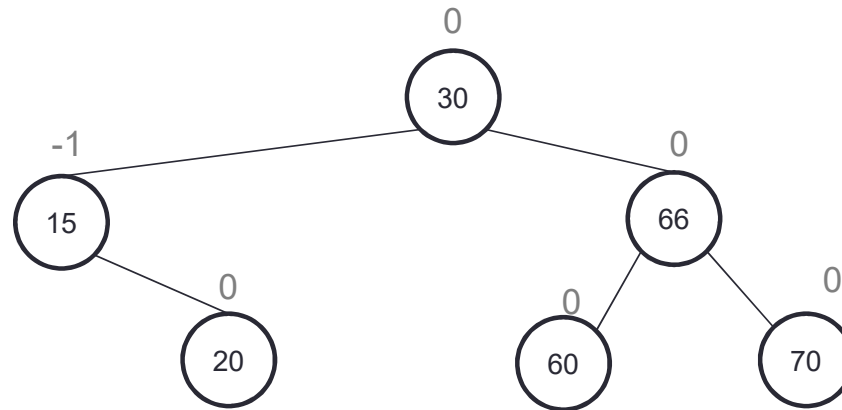


Single rotations only work when involved nodes are “in a line”

So we will first rotate left around 60, **then we can rotate right around 70.**

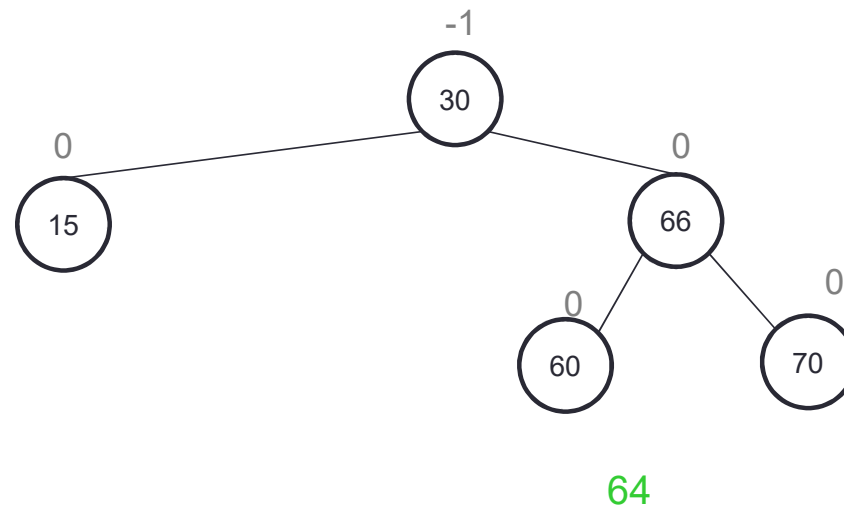
Where in the tree above should I cut to start the second rotation?

Double rotation to the rescue



Single rotations only work when involved nodes are “in a line”
So we will first rotate left at 60, **then we can rotate right at 70.**

It's sometimes even more complicated



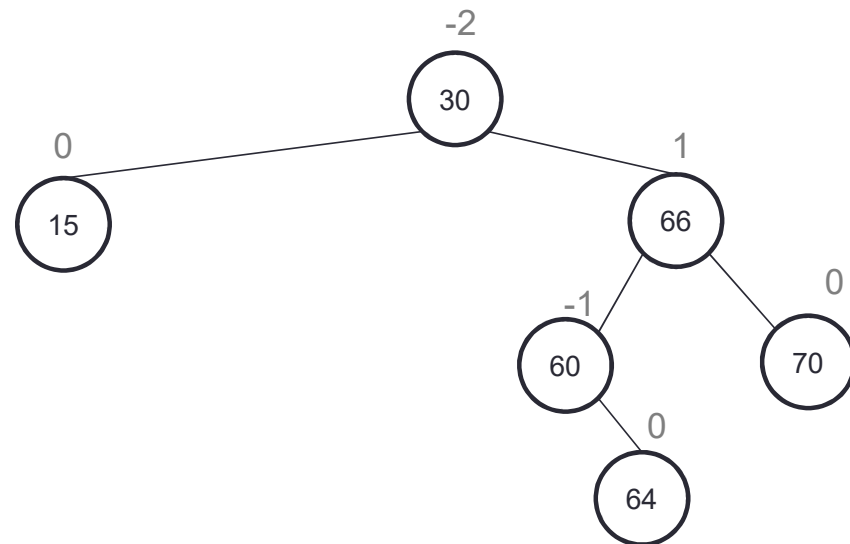
Insert 64... do we need a double or a single rotation?

A. Double

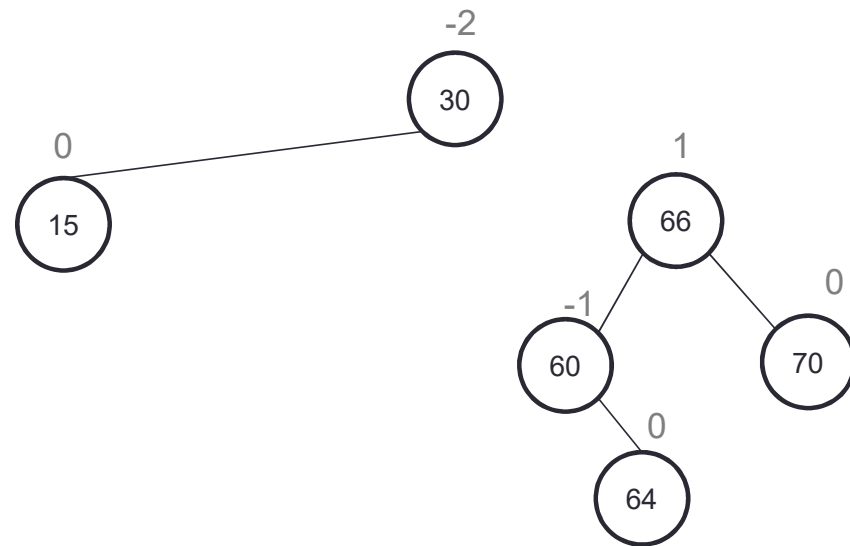
B. Single

C. No rotation needed

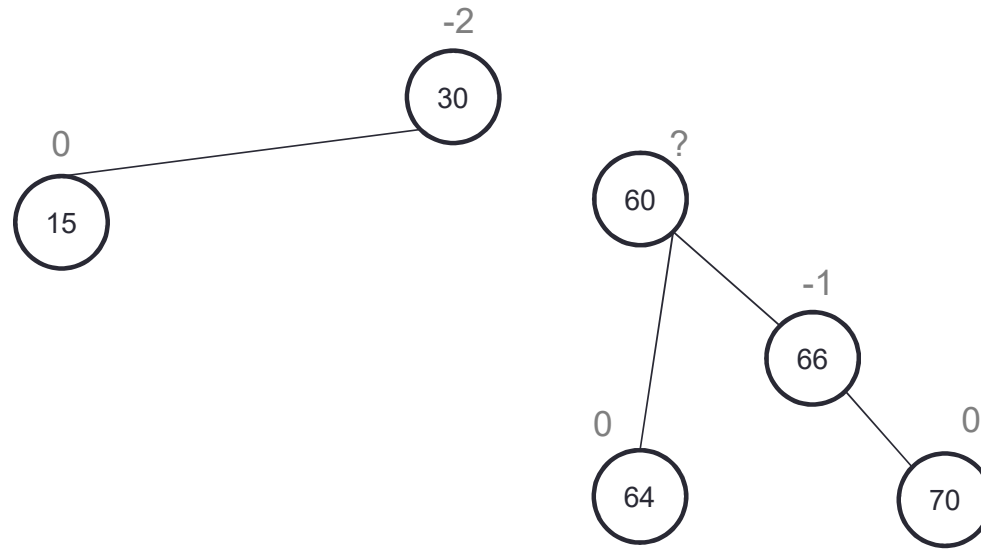
Rotate right at 66 to make a straight line



Rotate right at 66 to make a straight line

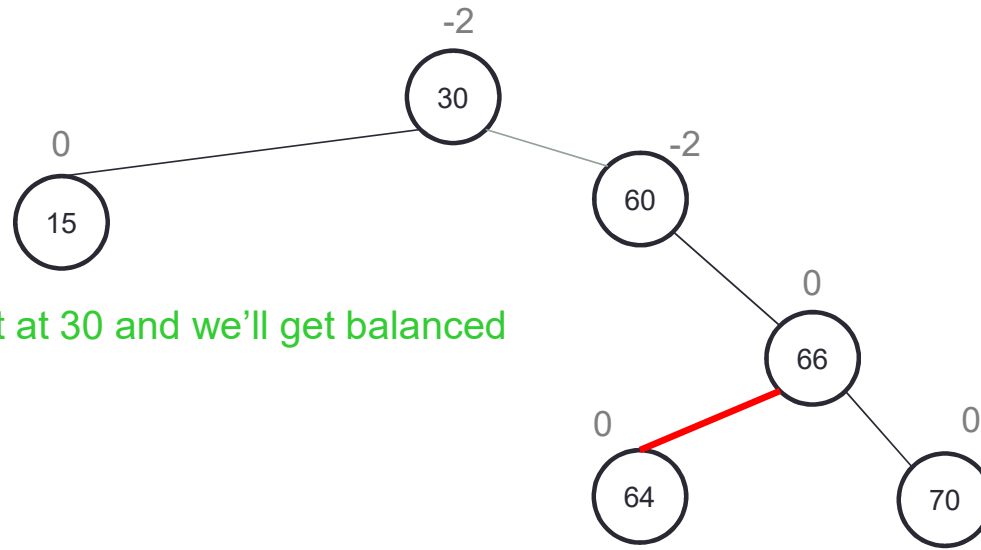


Rotate right at 66 to make a straight line



UH OH! Where do we put 64??
Are we stuck?

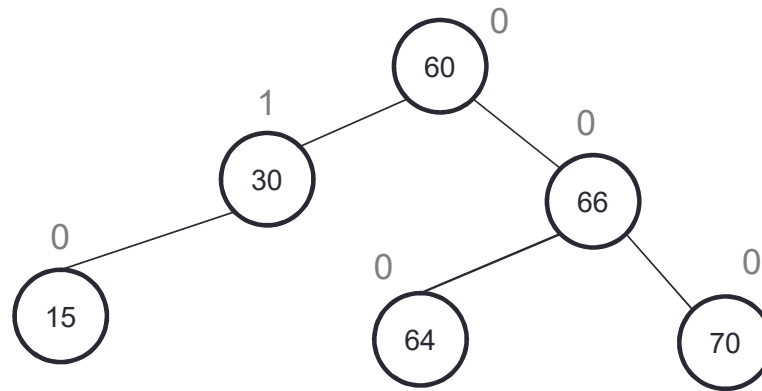
Rotate right at 66 to make a straight line



Now rotate left at 30 and we'll get balanced

Will 64 always reattach there?
Yes! Discuss why...

Finishing the rotation to balance the tree



Summary

- AVLs are balanced because we enforce balance at insertion (and removal)
 - Rotations allow us to achieve this and are constant time operations
- By enforcing balance, we ensure $O(\log n)$ find operations