Binary Search

Searching in Sorted Arrays

A card for you to write.

Linear Search

- Searching is looking for the location of the item (index).
- Algorithm: Start at element 0. Continue until (a) you find it > return index (b) you reach the end > return -1.
- Speed: O(n).
- Trade-off: Slower search than binary, however it works on unsorted data.

Linear Search #1

https://youtu.be/xDBxVzkUk7E

Start at the beginning.
Look at each element.
Stop when:

- You find it.
- You get to the end. Return -1. It isn't there.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
1	91	3	5	8	7	21	4	33	9	0	10	2

Linear Search #2

https://youtu.be/Wo-mLpXL9aE

Start at the beginning.
Look at each element.
Stop when:

- You find it.
- You get to the end. Return -1. It isn't there.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Z	X	B	N	M	P	O	J	K	F	R

A card for you to write.

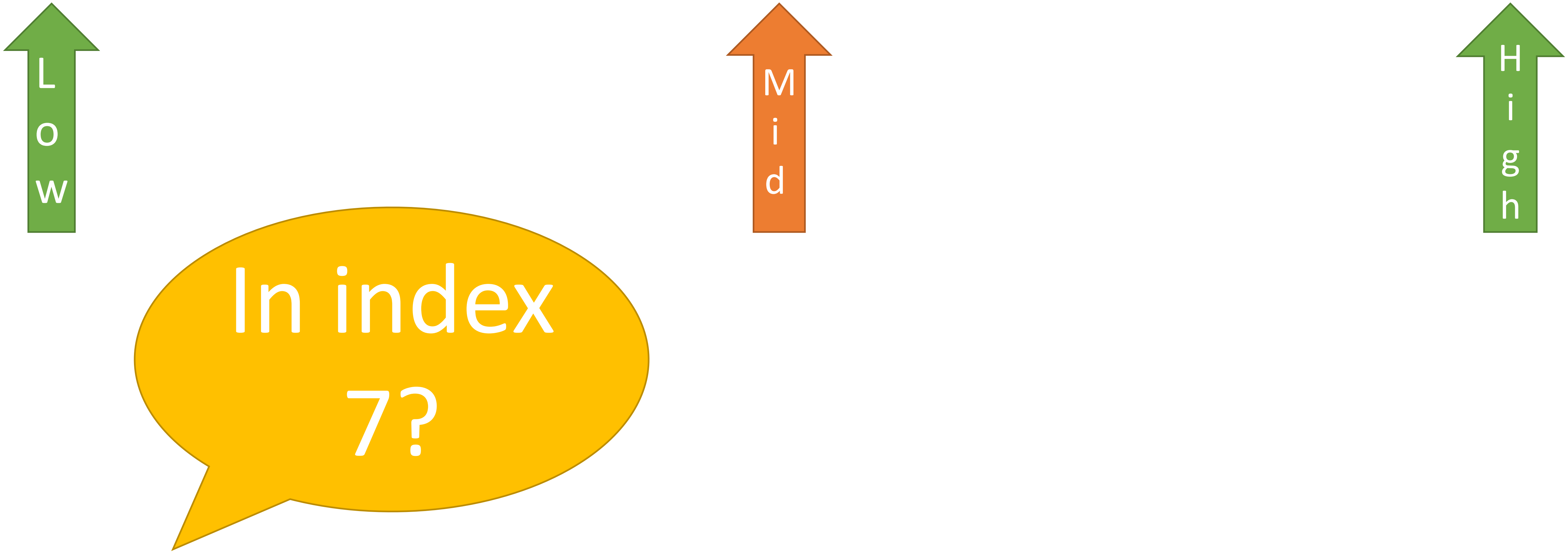
Binary Search

- Can be coded recursively.
- Algorithm: Track the lowest and highest spot where the item might be. Search halfway, adjust.
- Speed: O(log n).
- Trade-off: Much faster search than linear search, however only works on sorted data.
- Uses the order of the data to speed up the search.

14

Looking for 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



Looking for 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24

So, we can adjust our low boundary.

Mid

Low

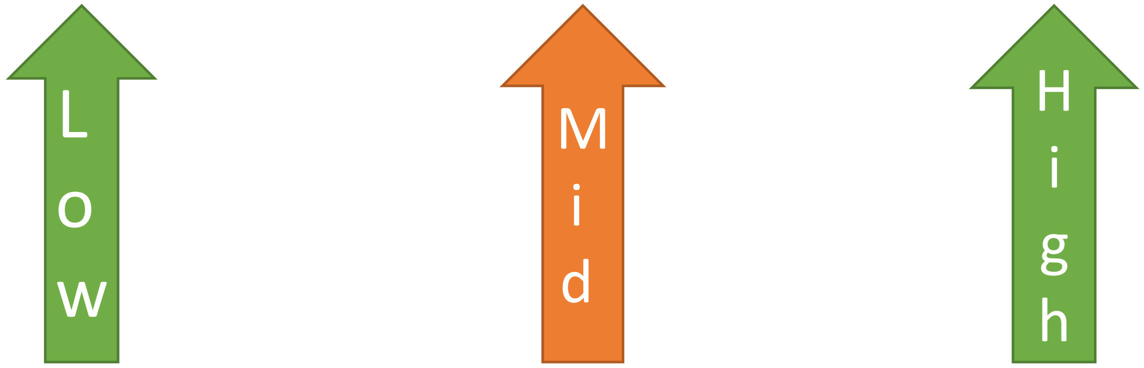
High

In index 7?

Too Low.

Looking for 14

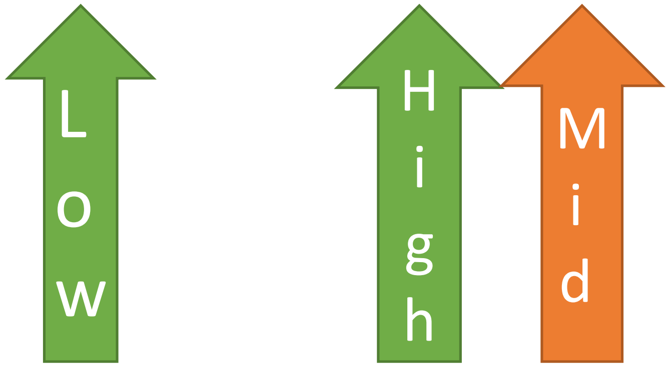
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



In index 11?

Looking for 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



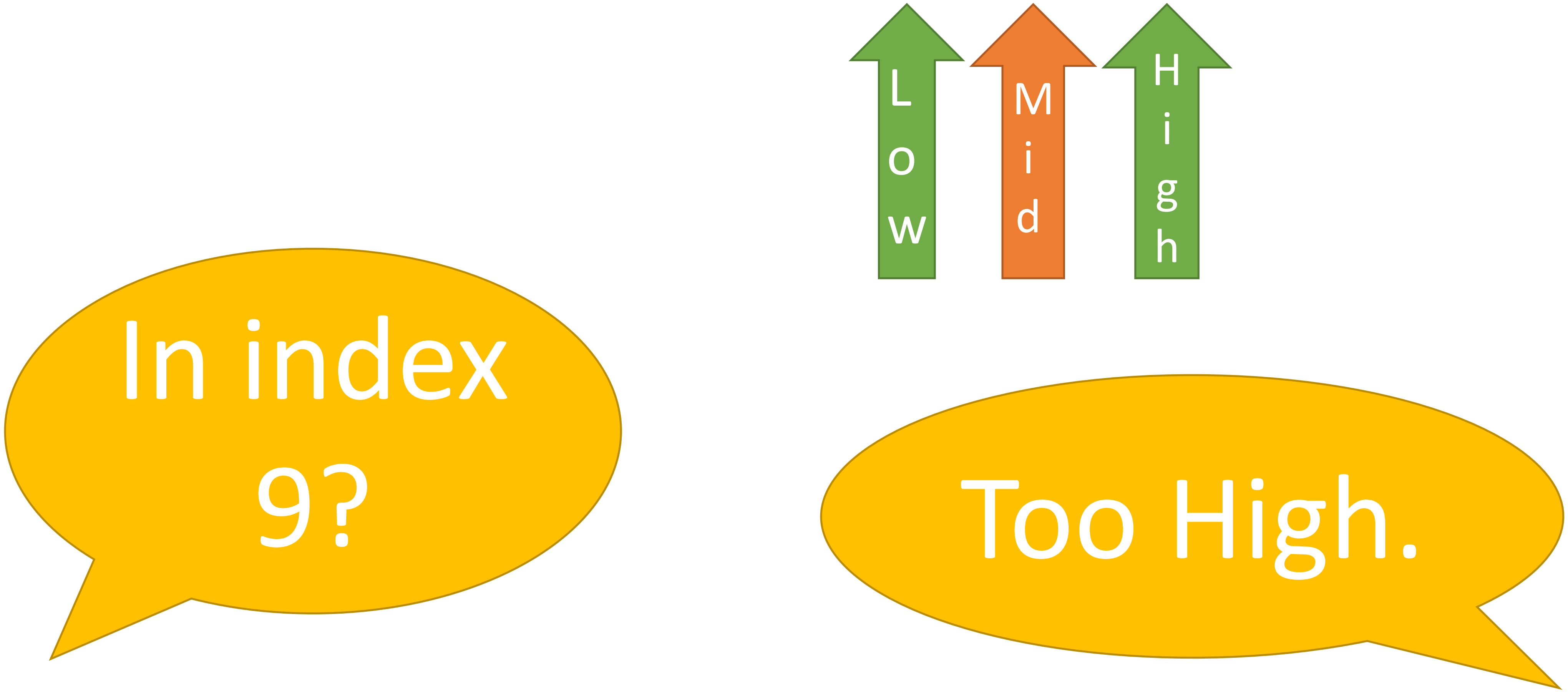
So, we can adjust our high boundary.

In index 11?

Too High.

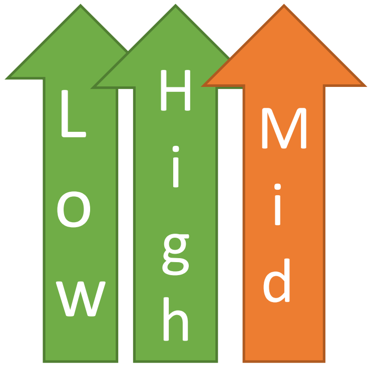
Looking for 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



Looking for 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



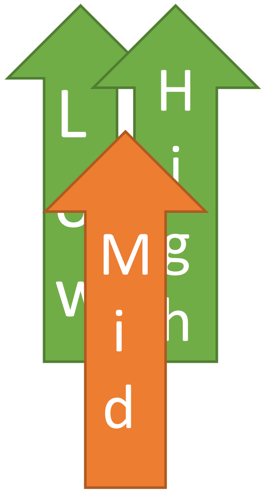
So, we can adjust our high boundary.

In index 9?

Too High.

Looking for 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



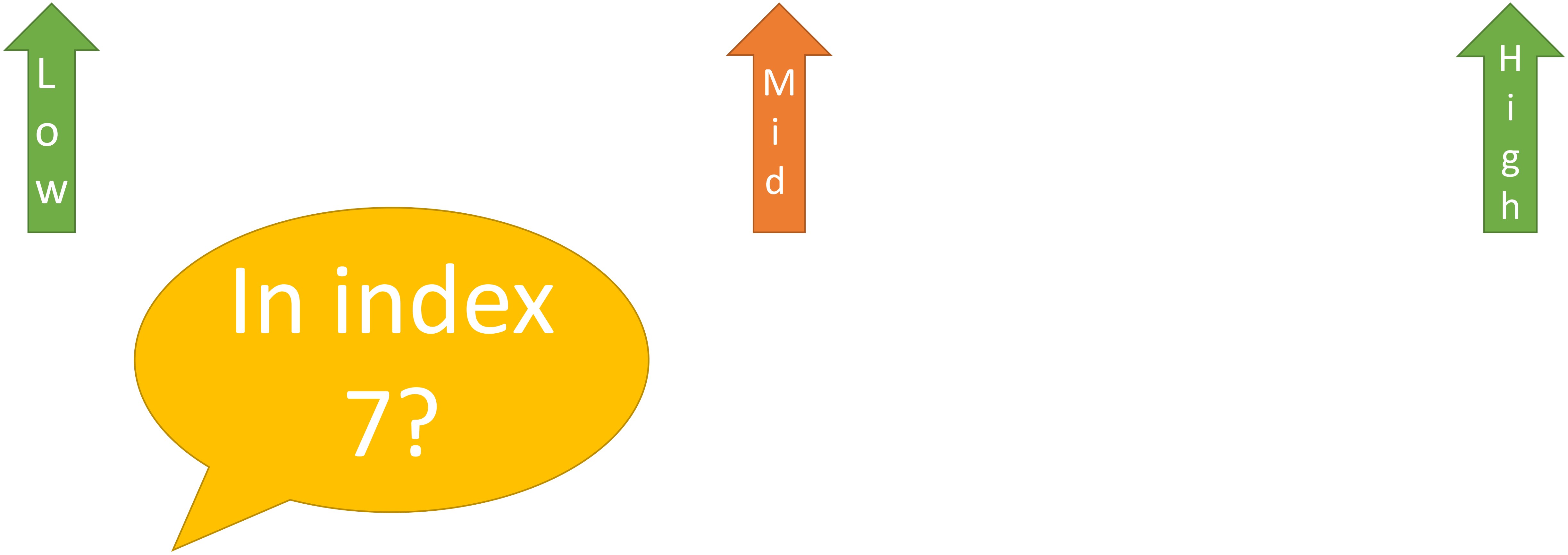
In index 8?

Yes.


13

Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24

So, we can adjust our low boundary.

Mid

Low

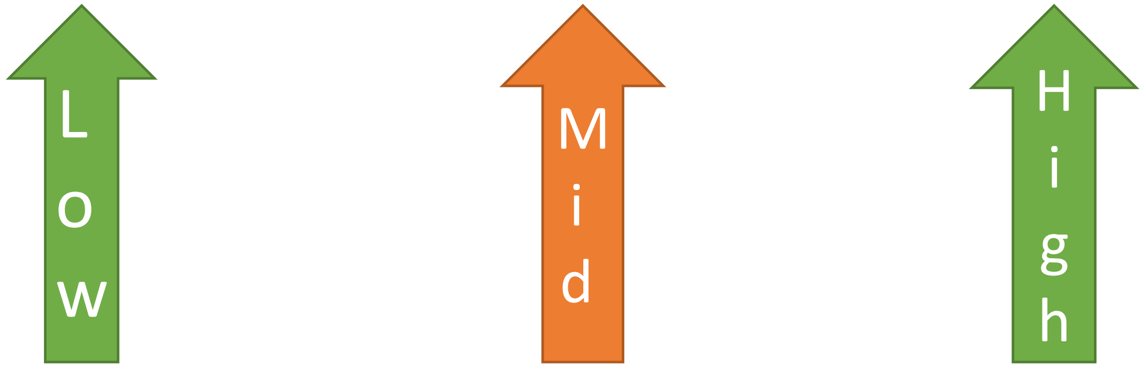
High

In index 7?

Too Low.

Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24

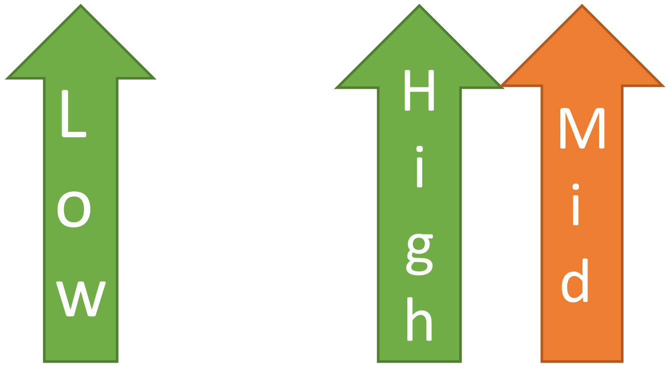


In index 11?

Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24

So, we can adjust our high boundary.

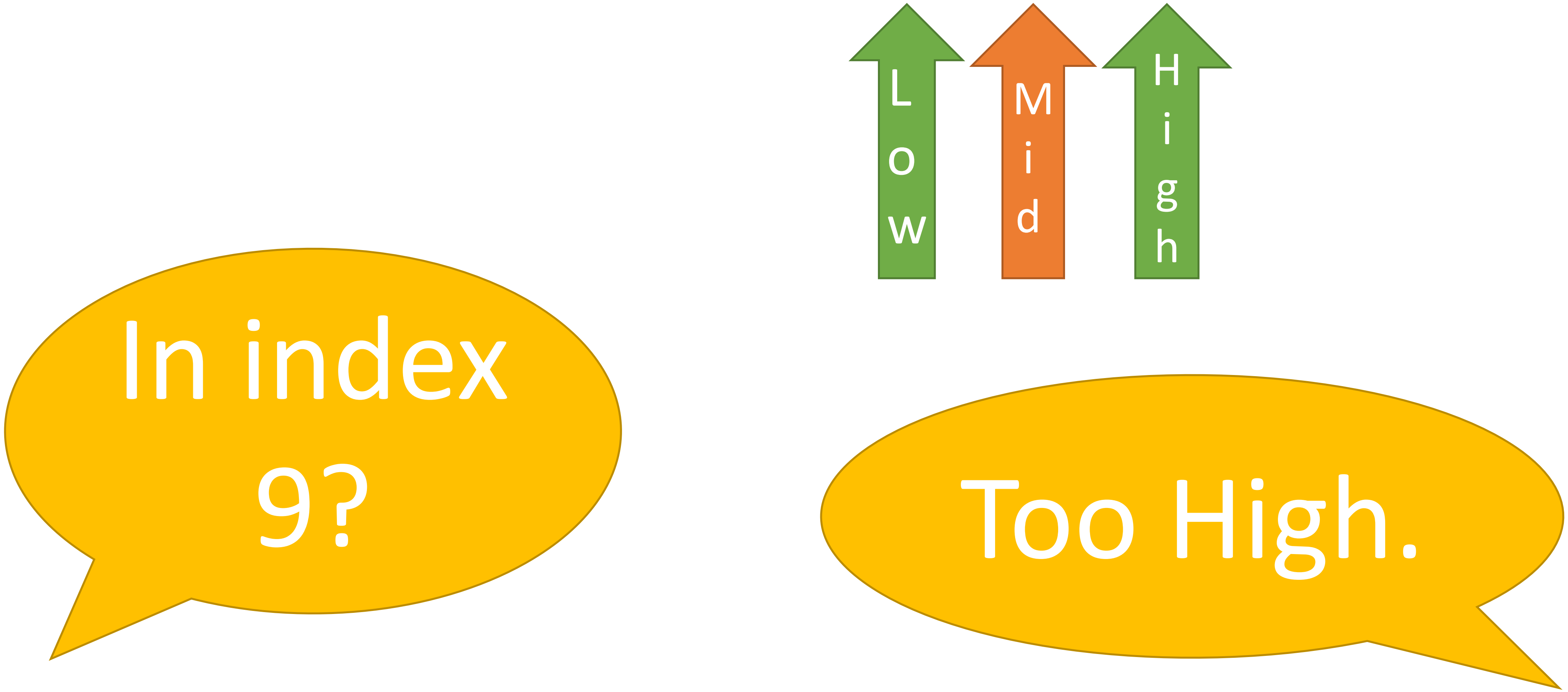


In index 11?

Too High.

Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24

So, we can adjust our high boundary.

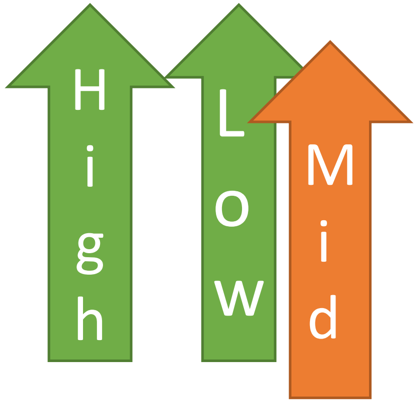
Low High Mid

In index 9?

Too High.

Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



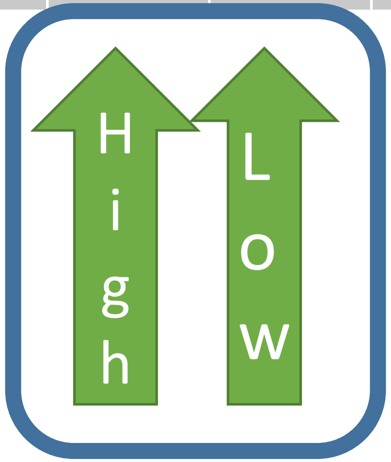
So, we can adjust our high boundary.

In index 8?

No.

Looking for 13

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-2	-1	4	6	8	9	10	12	14	16	17	18	20	21	24



Out of order. It’s not in either half. It’s not there.

Is

No.

Binary Search #1

Start Low at 0. Start High at a.length.
Find mid. Look in that position.

- If too low, adjust low to (mid + 1)
- If too high, adjust high to (mid -1)

Stop when:

- Mid is the right position.
- Low > High. Return -1. It isn't there.

https://youtu.be/NjGSKXnaFz8

Low	High	Mid

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
-45	-32	-12	-6	0	1	3	5	7	10	14	21

Binary Search #2

https://youtu.be/79hlCibvntQ

Start Low at 0. Start High at a.length.
Find mid. Look in that position.

- If too low, adjust low to (mid + 1)
- If too high, adjust high to (mid -1)

Stop when:

- Mid is the right position.
- Low > High. Return -1. It isn’t there.

Low	High	Mid

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Apple	Bee	Carrot	Egg	Jam	Mitt	Pet	Zebra

Binary Search #3

Start Low at 0. Start High at a.length.
Find mid. Look in that position.

- If too low, adjust low to (mid + 1)
- If too high, adjust high to (mid -1)

Stop when:

- Mid is the right position.
- Low > High. Return -1. It isn't there.

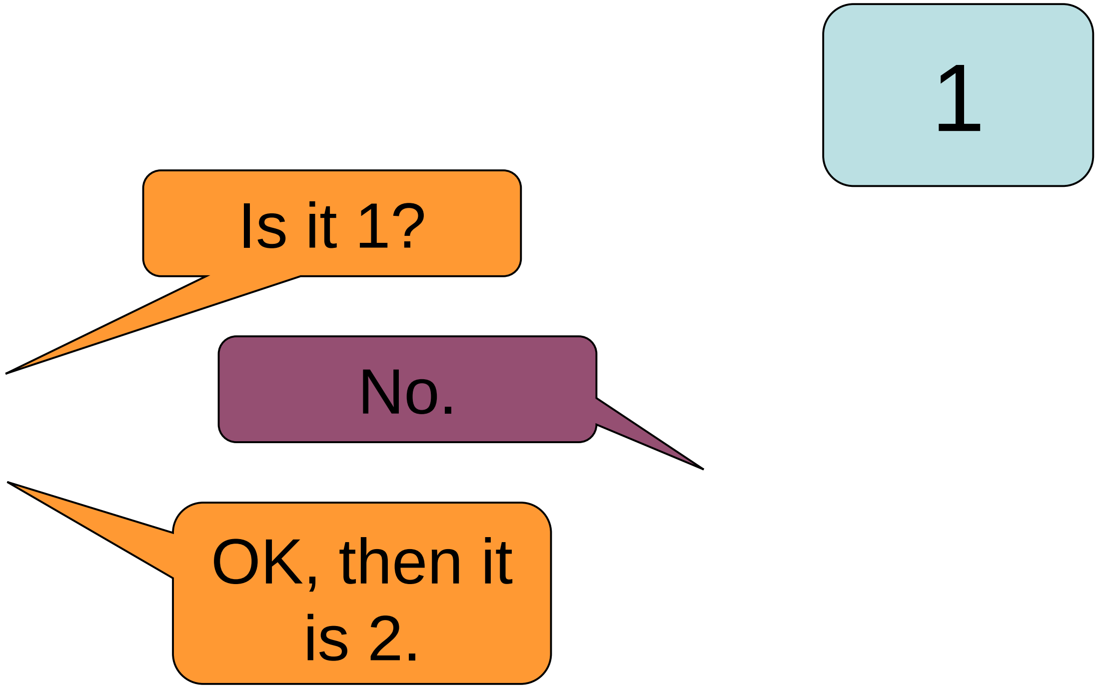
https://youtu.be/Wo-mLpXL9aE

Low	High	Mid

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
A	B	D	E	G	I	K	M	N	Q	S	U	Z

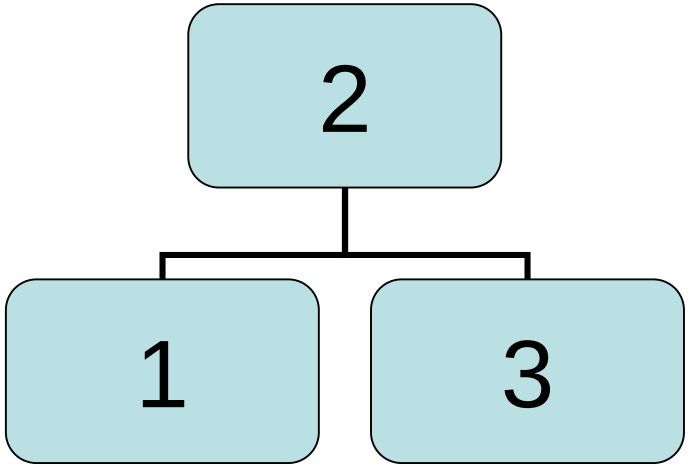
Why is Binary Search O(logn)?

And why do you keep saying that is really fast?



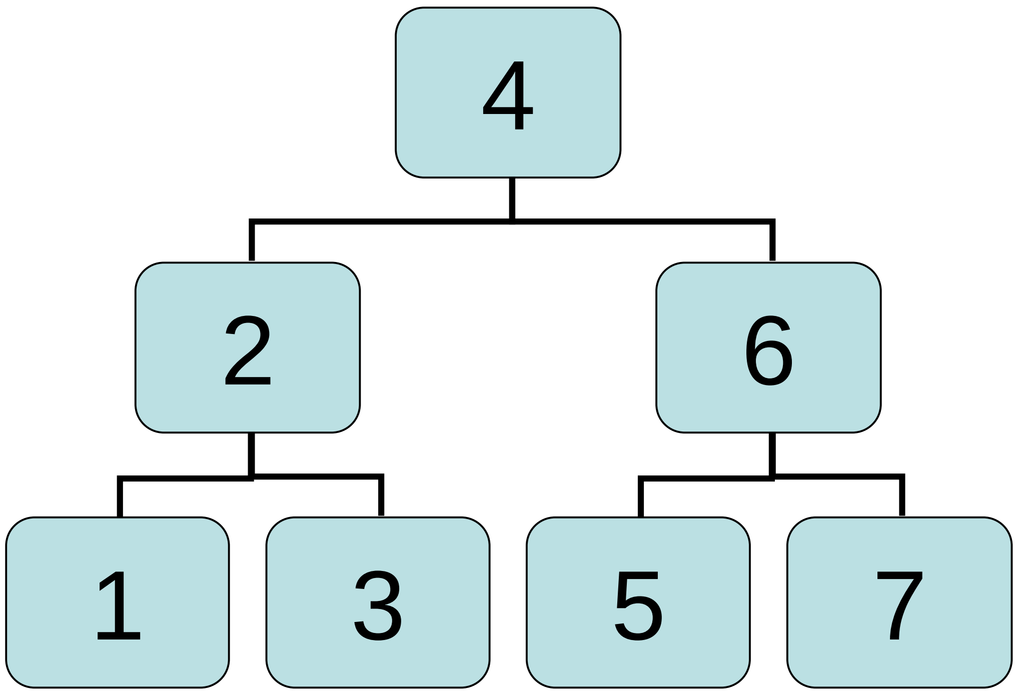
Numbers 1 to 2 = 1 guesses.

$2^1 = 2$



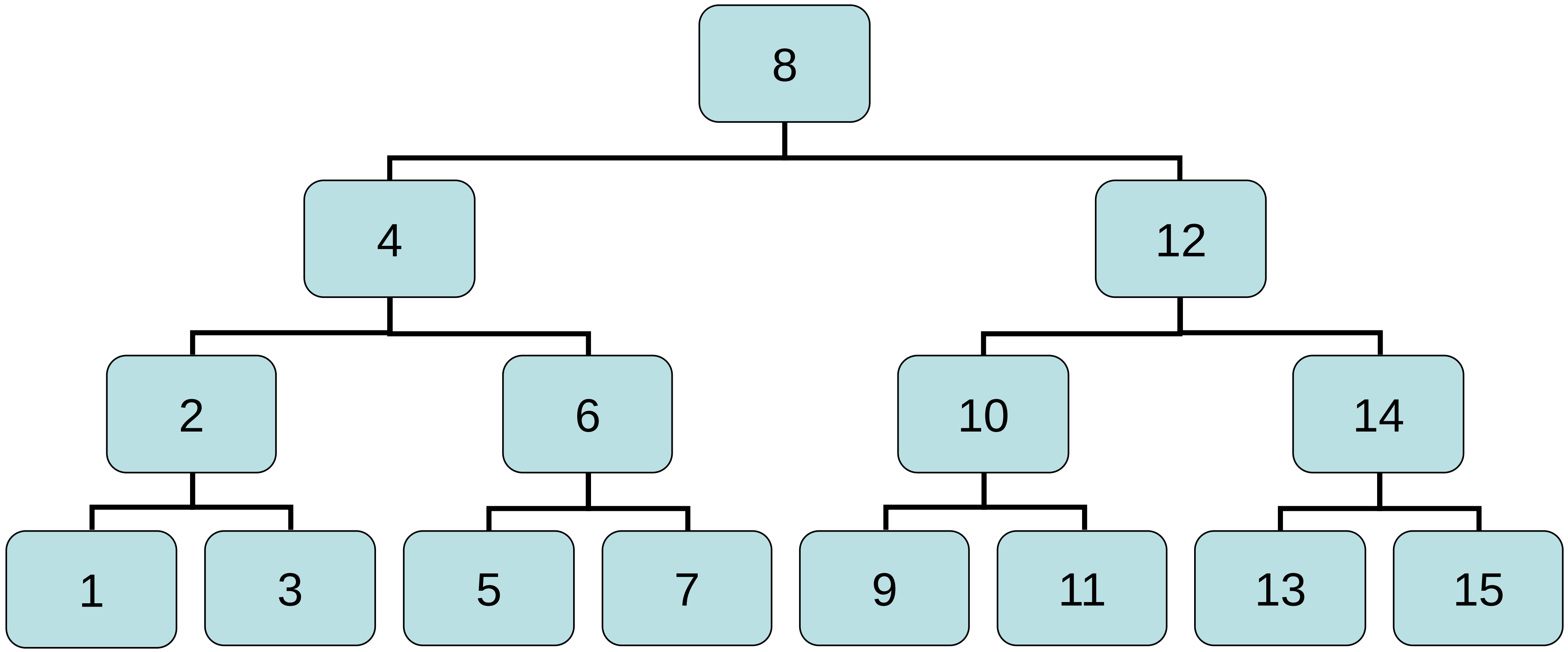
Numbers 1 to 3 = 2 guesses.

$2^2 = 4$



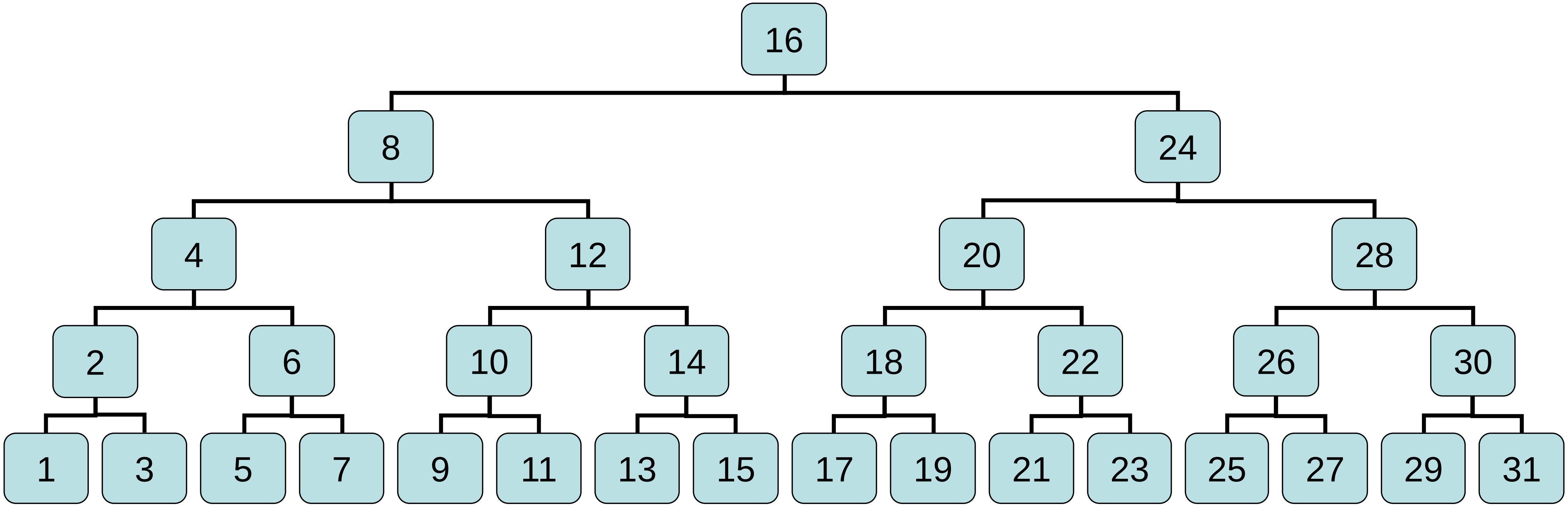
Numbers 1 to 7 = 3 guesses.

$2^3 = 8$



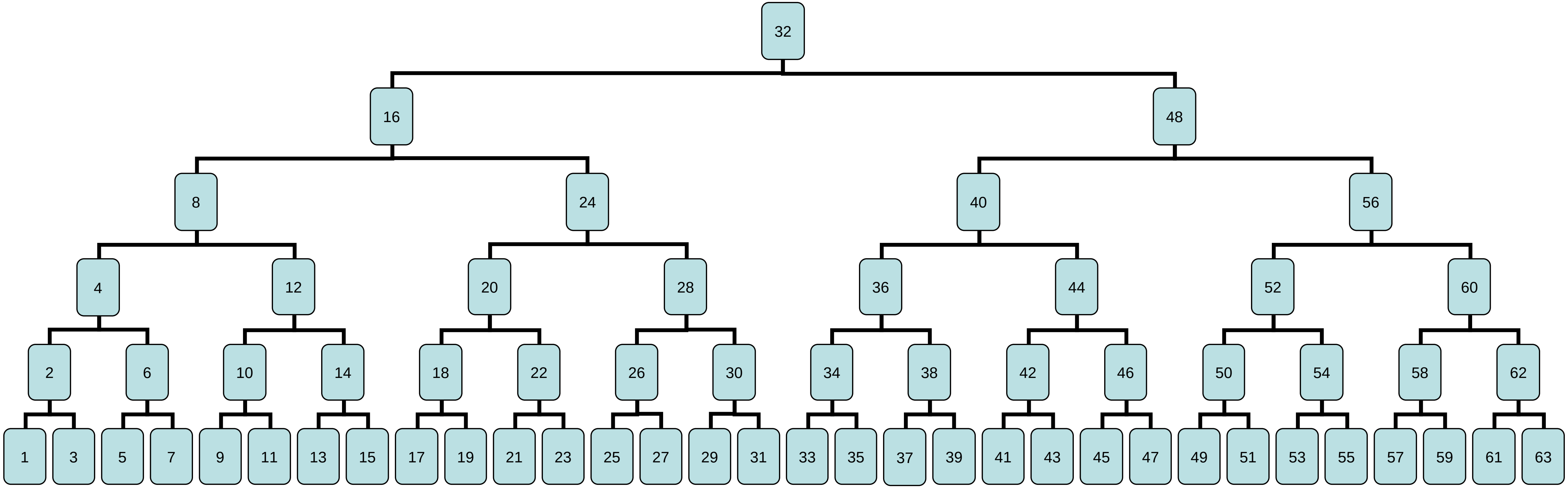
Numbers 1 to 15 = 4 guesses.

$2^4 = 16$



Numbers 1 to 31 = 5 guesses.

$2^5 = 32$



Numbers 1 to 63 = 6 guesses.

$2^6 = 64$

Binary Search is Logarithmic

Highest Number (n)	Binary Expression	Number of Searches	
2	$2^1 = 2$	1	$\log_2(2)=1$
3	$2^2 = 4$	2	$\log_2(4)=2$
7	$2^3 = 8$	3	$\log_2(8)=3$
15	$2^4 = 16$	4	$\log_2(16)=4$
31	$2^5 = 32$	5	$\log_2(32)=5$
63	$2^6 = 64$	6	$\log_2(64)=6$
127	$2^7 = 128$	7	$\log_2(128)=7$

Binary Search is a lot faster

C3 =CEILING(LOG(B3,2),1)+1

	A	B	C
1	# of Records	Avg Linear	Max Binary
2	1	1	1
3	10	5	4
4	100	50	7
5	1000	500	10
6	10000	5000	14
7	100000	50000	17
8	1000000	500000	20
9	10000000	5000000	24
10	100000000	50000000	27
11	1000000000	500000000	30
12	10000000000	5000000000	34

BINÄRY SEARCH

idea-instructions.com/binary-search/

IDEA