Search Algorithms

Binary Search

Time Complexity

Best Case:

O(1)

Average and Worst Case:

O(LogN)

Space Complexity

Iterative Approach:

No additional space required

(In-place algorithm)

O(1)

Recursive Approach:

O(N)

Array

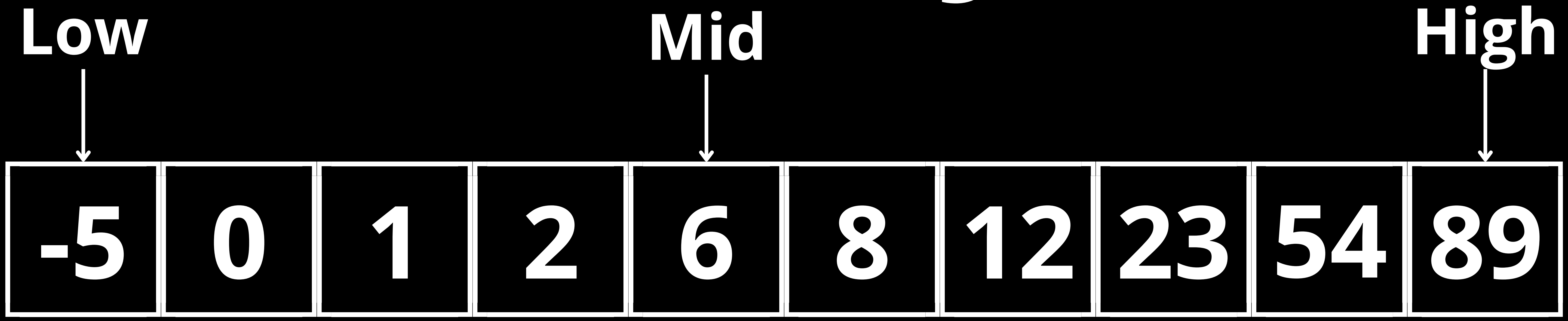
-5	0	1	2	6	8	12	23	54	89

Value to look for: 54

Steps to find the value:

- Find the lowest and highest index of the array
- Find the middle index of the array: (low+high)/2
- Check if the middle index element is the searched one
- If so, return the index, if not, check if its smaller or greater than the searched element
- If smaller, do a binary search on the upper half, if greater, do a binary search on the lower half

Array



Low

Mid

High

-5	0	1	2	6	8	12	23	54	89

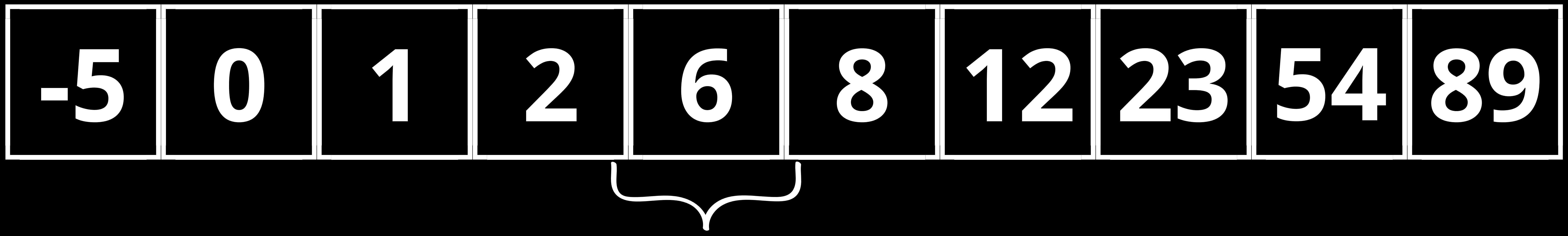
Lowest index: 0

Biggest index: 9

Middle index: (0+9)/2 = floor(4.5) = 4

Array

Index 4



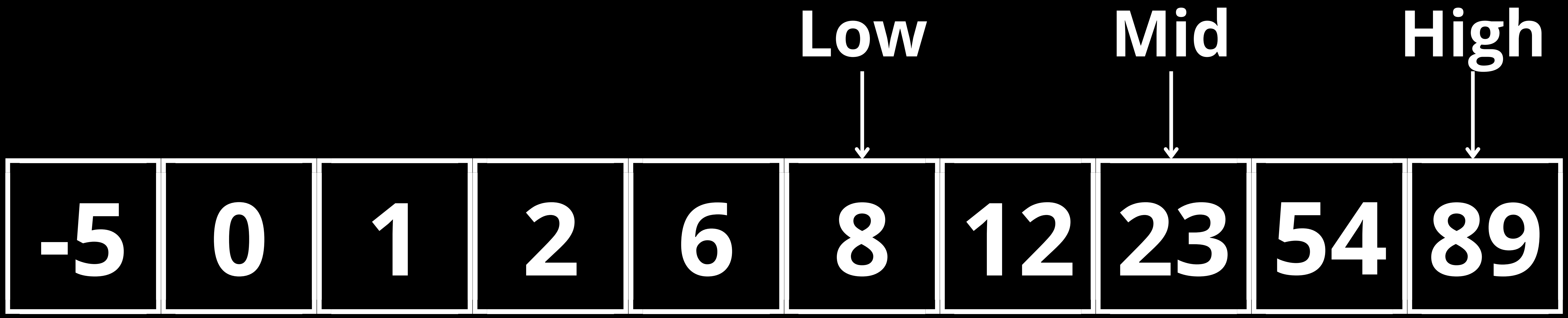
-5	0	1	2	6	8	12	23	54	89

Is 6 equal to 54? No!

Is 6 smaller than 54? Yes!

Search on the upper half

Array



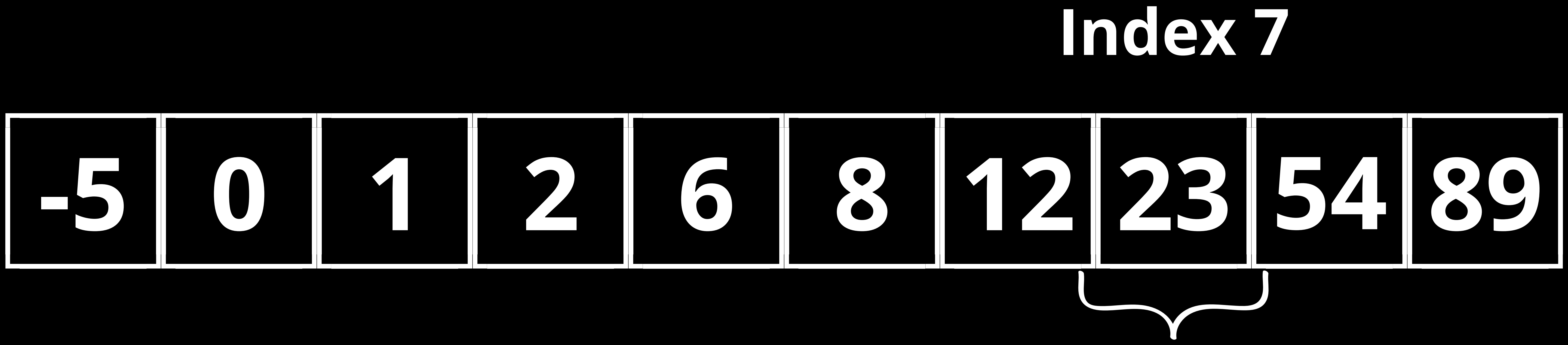
-5	0	1	2	6	8	12	23	54	89

Lowest index: mid+1 = 5

Biggest index: 9

Middle index: (5+9)/2 = floor(7) = 7

Array



Index 7

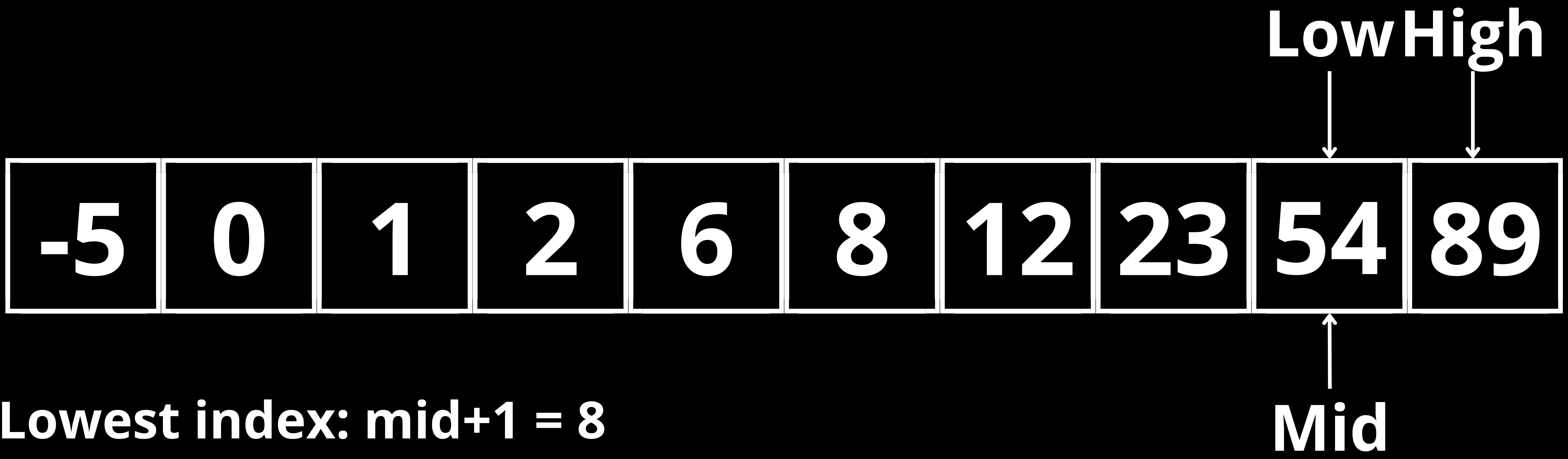
-5	0	1	2	6	8	12	23	54	89

Is 23 equal to 54? No!

Is 23 smaller than 54? Yes!

Search on the upper half

Array



Low High

-5	0	1	2	6	8	12	23	54	89

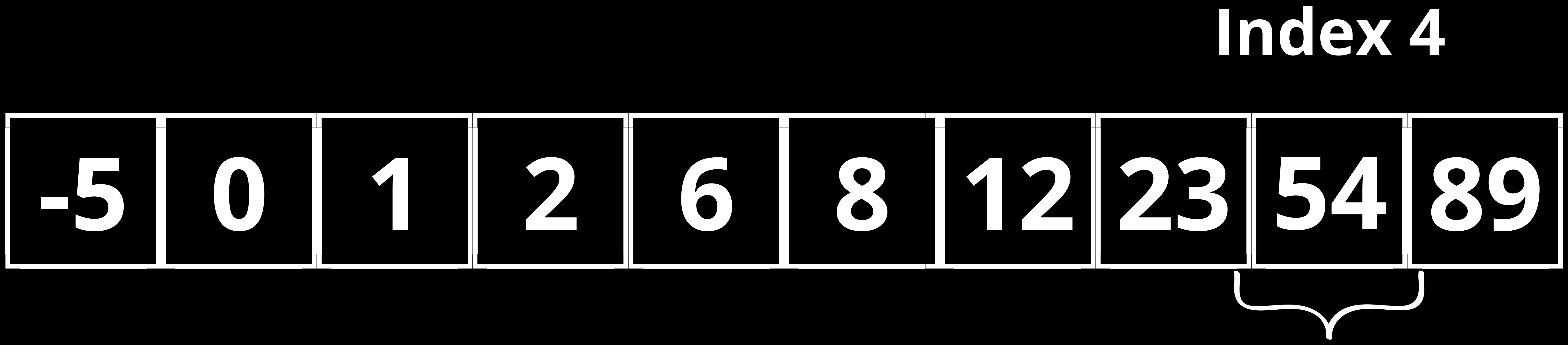
Mid

Lowest index: mid+1 = 8

Biggest index: 9

Middle index: (8+9)/2 = floor(8.5) = 8

Array



Index 4

-5	0	1	2	6	8	12	23	54	89

Is 54 equal to 54? Yes!

Return index 8

Done!