Sorting Algorithms

QuickSort

Time Complexity

Best and Average Case:

$$O(N * LogN)$$

Worst Case:

$$O(N^2)$$

Space Complexity

(Not Considering recursive Stack Space)

Constant:

O(1)

(Considering recursive Stack Space)

Worst Case:

O(N)

PIVOT

Compare with pivot

Counter for pivot swap

PIVOT

pink < pivot?

no!

pink++

pink < pivot?
yes!
swap pink and yellow
pink++
yellow++

PIVOT

pink < pivot?
yes!
swap pink and yellow
pink++
yellow++

PIVOT

PIVOT

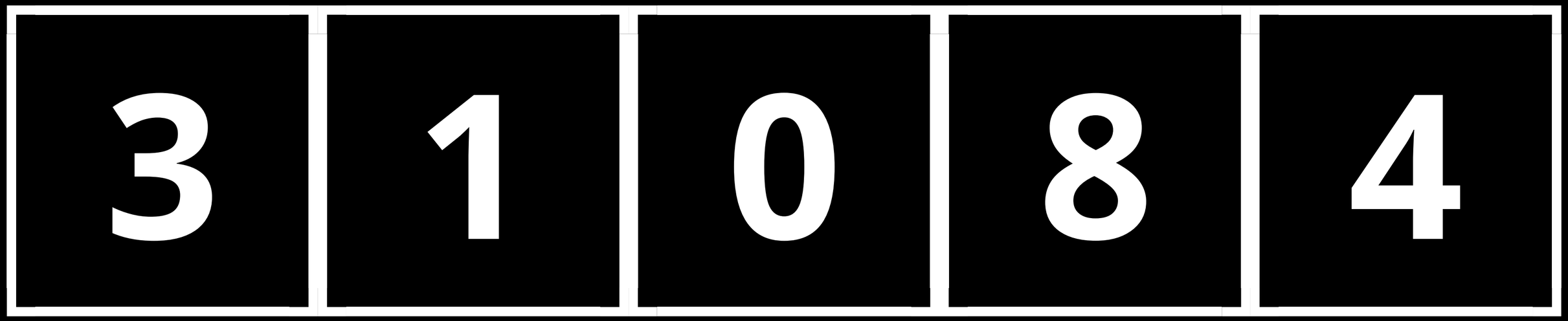
pink < pivot?
yes!
swap pink and yellow
pink++
yellow++

pink index == pivot index?

PIVOT

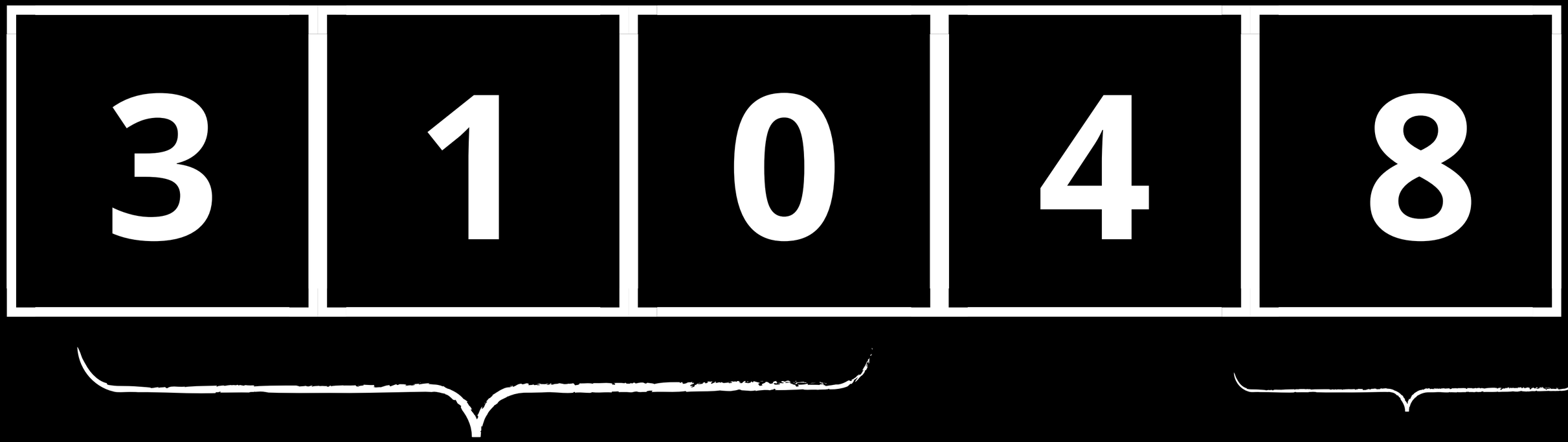
yes!

Swap Pivot and Yellow



PIVOT

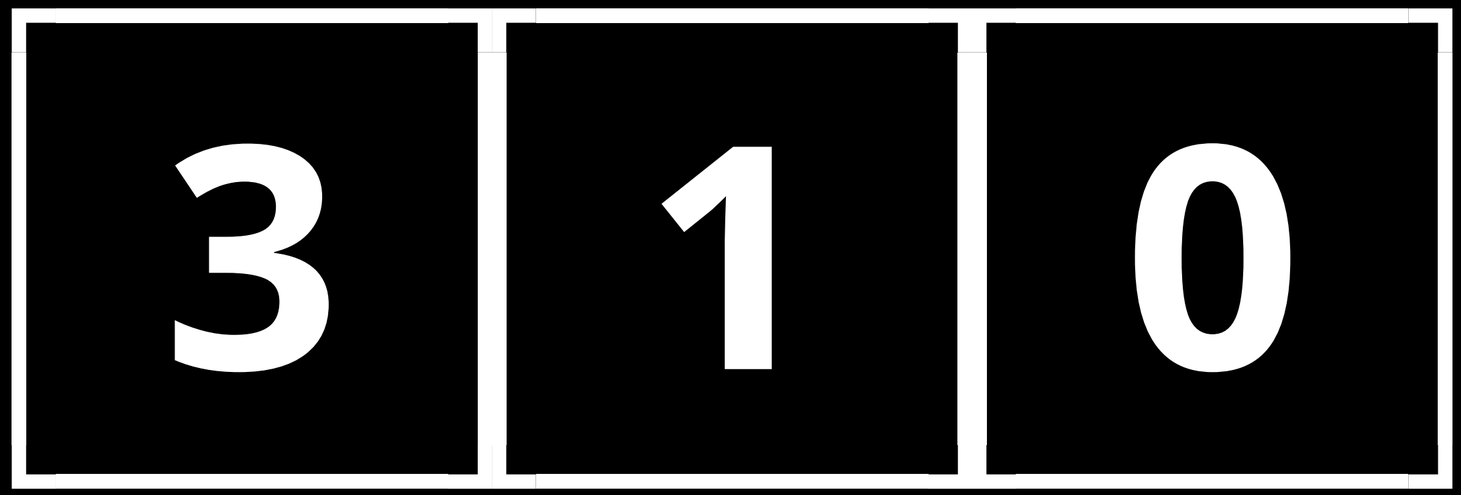
All elements to left are < pivot and all elements to right are > pivot!



Now we repeat the process recursively on the left and right sub-arrays

Applying to the left sub-array

PIVOT



Compare with pivot

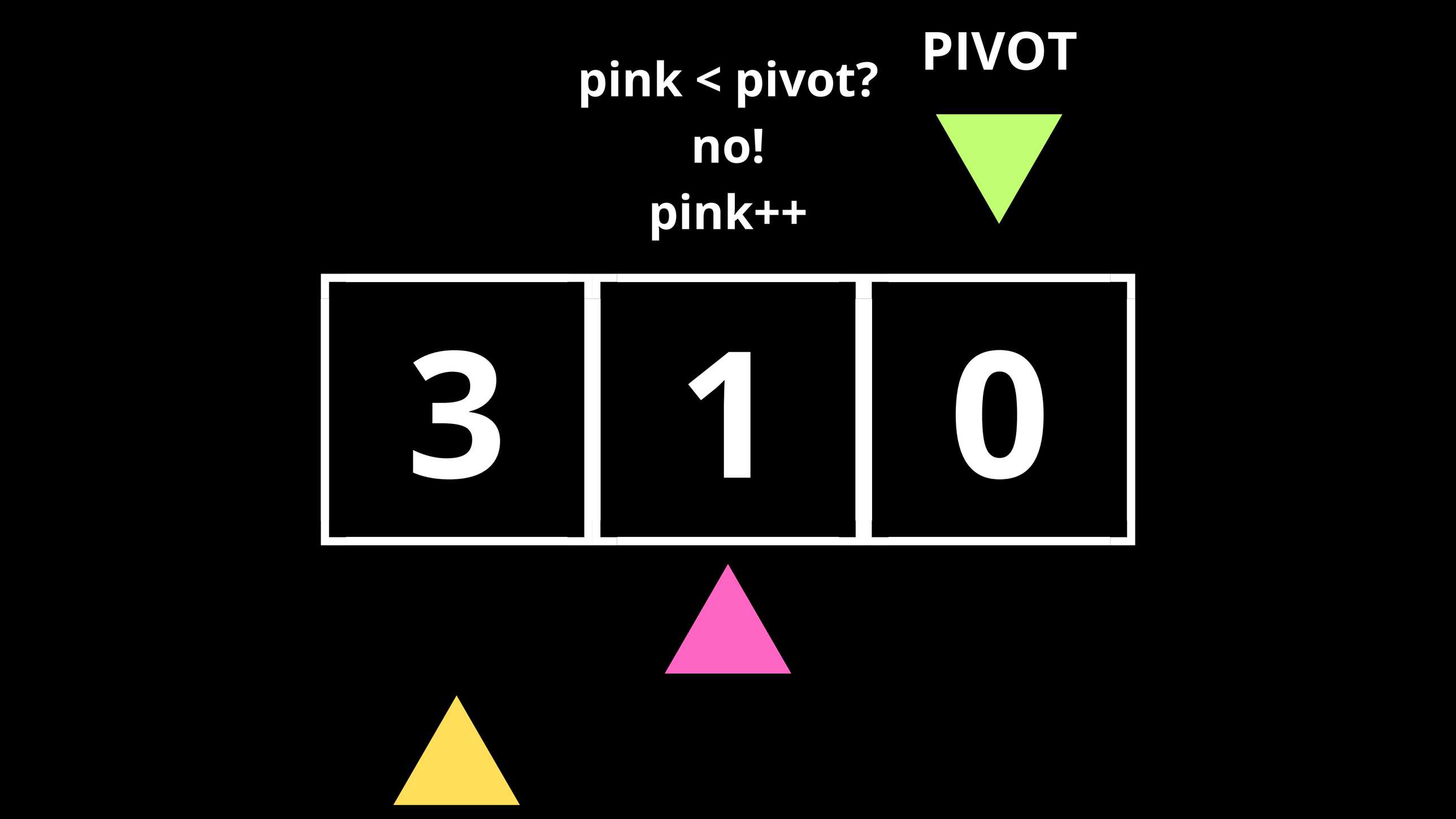
Counter for pivot swap

pink < pivot?

no!

pink++

PIVOT

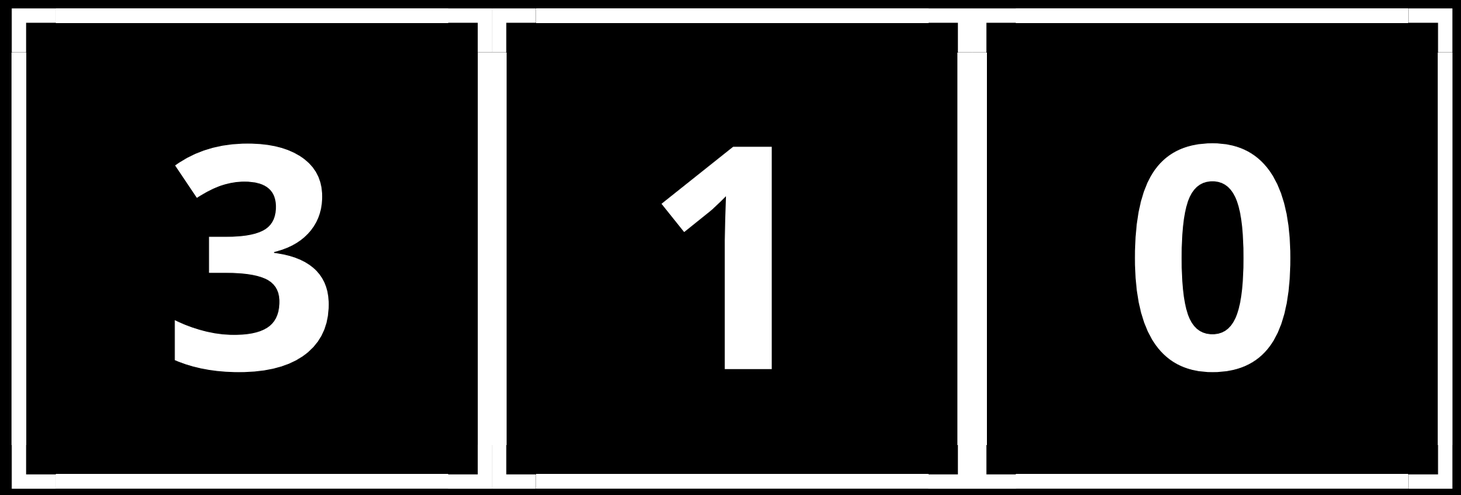

pink < pivot?
no!
pink++
PIVOT
3
1
0

pink index == pivot index?

PIVOT

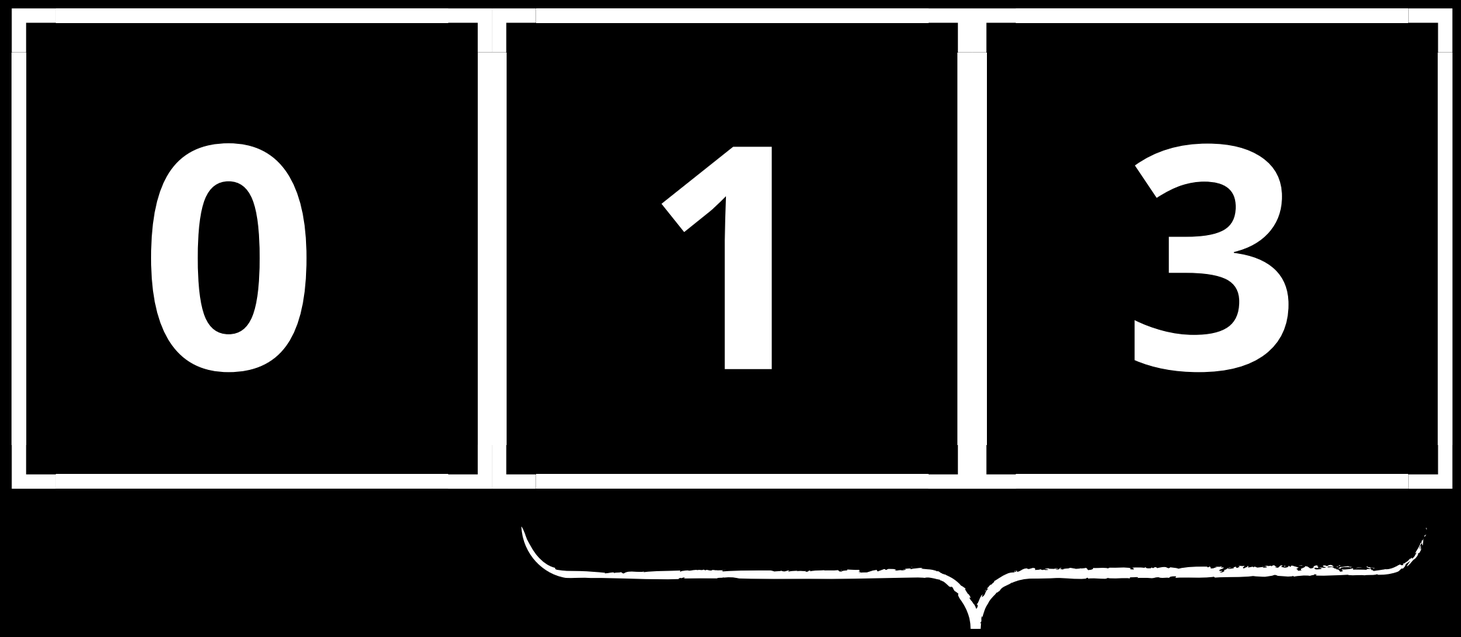
yes!

Swap Pivot and Yellow



PIVOT

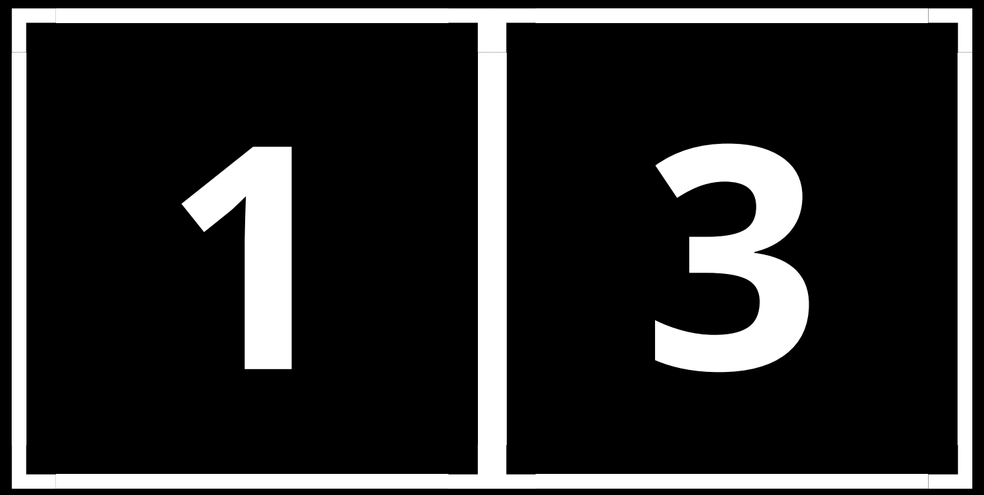
All elements to left are < pivot and all elements to right are > pivot!



Now we repeat the process recursively on the left and right sub-arrays

Applying to the right sub-array

PIVOT

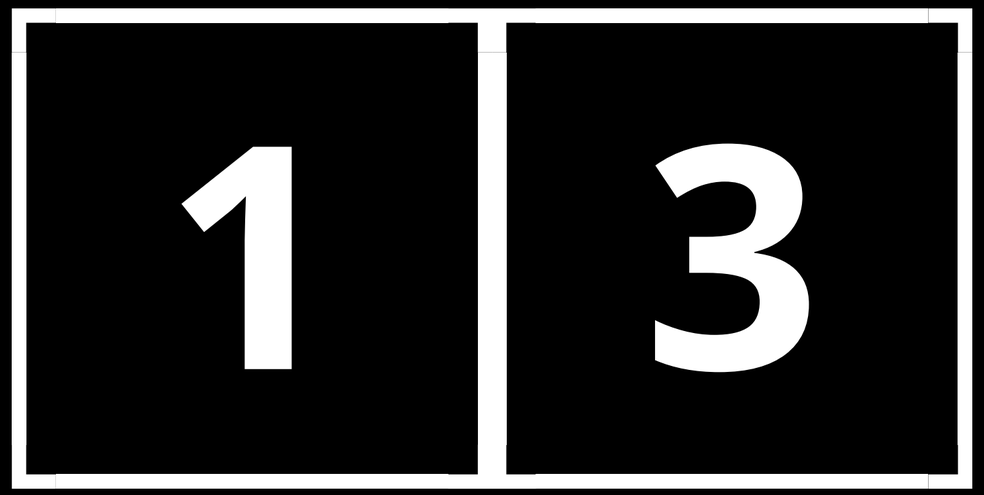


Compare with pivot

Counter for pivot swap

pink < pivot?
yes!
swap pink and yellow
pink++
yellow++

PIVOT

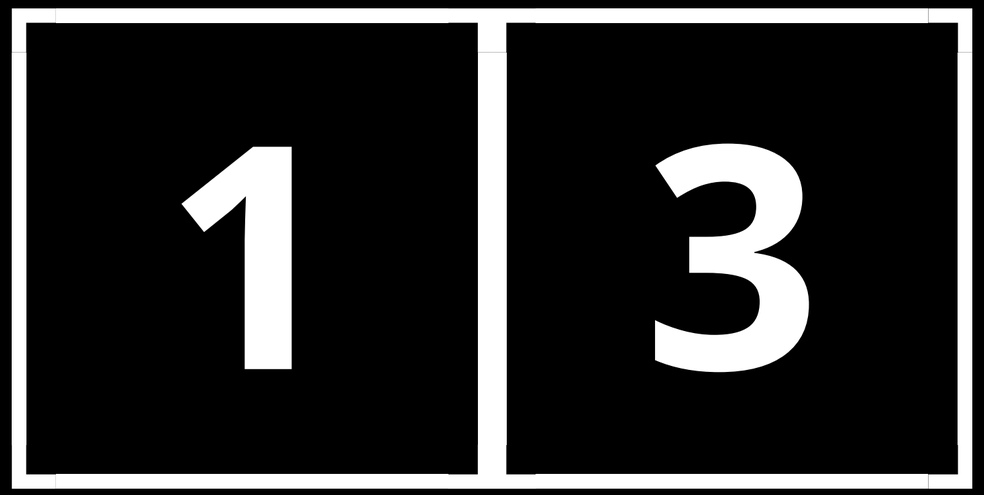


pink index == pivot index?

yes!

Swap Pivot and Yellow

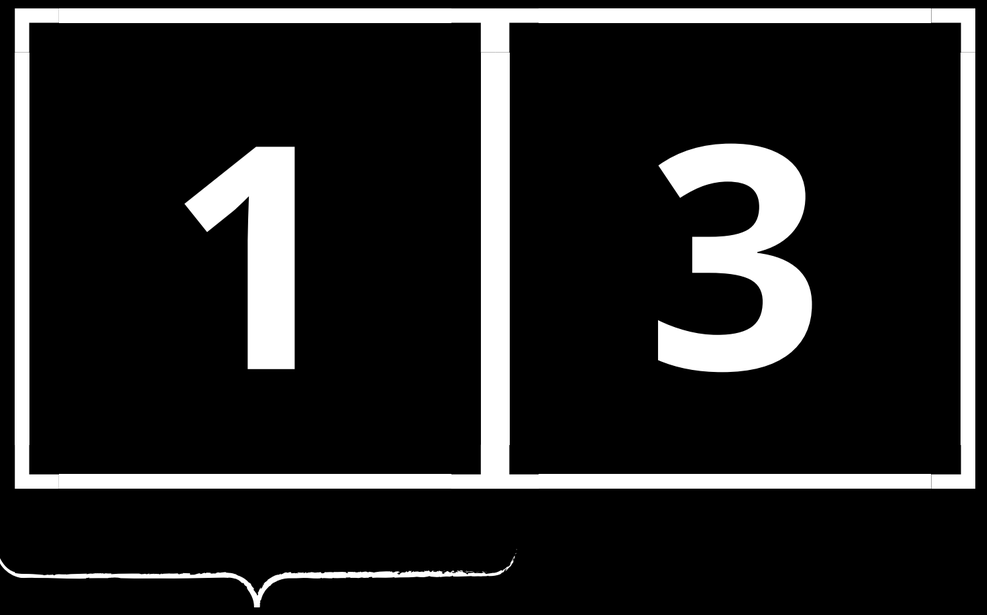
PIVOT



As the index of Yellow == index of Pivot, we just ignore the swap

All elements to left are < pivot and all elements to right are > pivot!

PIVOT



Now we repeat the process recursively on the left and right sub-arrays

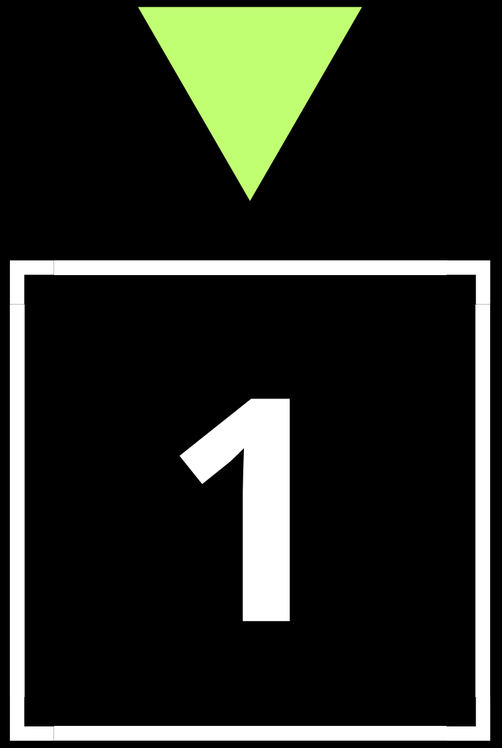
Applying to the left sub-array

PIVOT

Compare with pivot

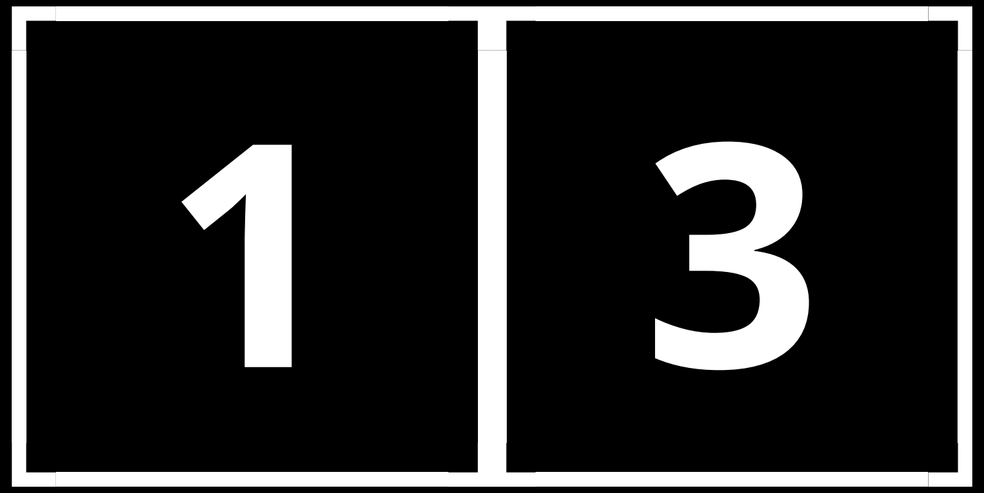
Counter for pivot swap

PIVOT



As we only have one element left, there is no need to sort anything, we just return the sub-array

This is the next sub-array we will return, as it has been already sorted



This is the next sub-array we will return, as it has been already sorted

This is the original array after we have applied the first iteration and applied recursively to the left side and returned it

Now we apply it to the right side sub-array, the same way we’ve done with the left side

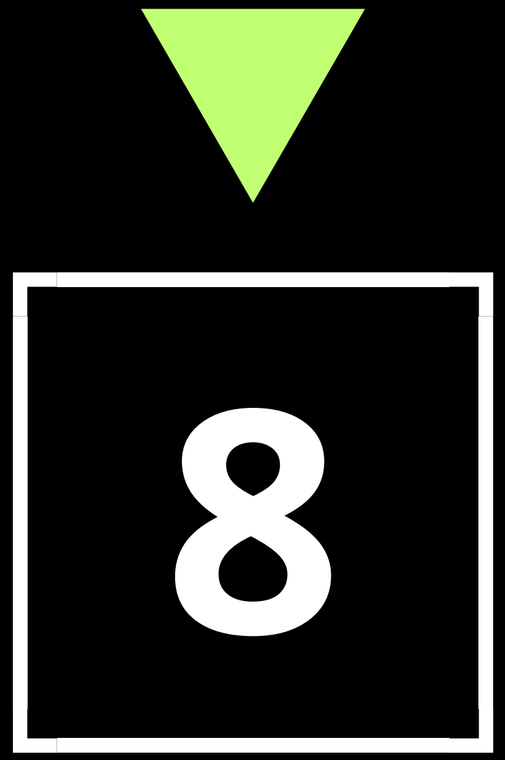
Applying to the right sub-array

PIVOT

Compare with pivot

Counter for pivot swap

PIVOT



As we only have one element left, there is no need to sort anything, we just return the sub-array

Ans this is the final sorted array after we've applied all iterations on all sub-arrays