# 01.SlidingWindow

# 02.TwoPointersOrIterators

Squaring a sorted array (easy)

Triplets that sum to zero (medium)

Comparing strings that contain backspaces (medium)

# 03.FastAndSlowPointers

Linked List Cycle

Palindrome Linked List (medium)

Cycle in a Circular Array (hard)

# 04.MergeIntervals

Intervals Intersection (medium)

Maximum CPU Load (hard)

# 05.CyclicSort

Find the Missing Number (easy)

*function* findMissingNum (*arr*) {

*let* i = 0;

while(i<arr.length){

// swap the elements

*const* currEl = arr[i];

// 'currEl != i' so we dont increment i if we are in the right place

// currEl < arr.length because than it is out of the range (arr.length)

if(currEl != i && currEl < arr.length){

// ONCE THESE^ ARE TRUE WE SWAP

// newSpotVal is what is in the currEl so we set it to arr[currEl]

*const* newSpotVal = arr[currEl]

// NOW SWAP so arr[currEl] is now arr[i], which is defined as currEl

arr[currEl]=currEl;

// now where we are (arr[i]) is now what was in the currEl (newSpotVal

arr[i] = newSpotVal

}else {

// increment case when currEl is NOT EQUAL to i and within the range

i++

}

}

// THE CODE ABOVE WOULD RETURN THE ARRAYS SORTED with the nth number in the missing number's spot: [4,0,3,1] > [0,1,4,3]

// NOW WE JUST NEED TO FIND THE MISSING NUMBER

// create for loop with j

for (*let* j = 0; j < arr.length; j++){

// as soon so the number in the array is NOT EQUAL to the index, we return the index

if(arr[j] !== j) {

return j

}

}

// this is for the one case where missing number is the length of the array

return arr.length

}

// Test Cases

console.log(findMissingNum([4,0,3,1])) // 2

console.log(findMissingNum([9,6,4,2,3,5,7,0,1])) // 8

console.log(findMissingNum([4,0, 2, 5,3,1])) // 6

# 06.InPlaceReversalOfLinkedList

# 07.TreeBFS

# 08.TreeDFS

# 09.TwoHeaps

# 10.Subsets

# 11.ModifiedBinarySearch

# 12.TopKElements

# 13.K-WayMerge

# 14.TopologicalSort