## 41. First Missing Positive Feb. 2, 2019 | 46.9K views

Average Rating: 4.12 (49 votes)

Example 1:

Given an unsorted integer array, find the smallest missing positive integer.

### Input: [1,2,0]

Example 2:

## Output: 3

Input: [3,4,-1,1] Output: 2

Example 3:

## Output: 1

Input: [7,8,9,11,12]

Your algorithm should run in O(n) time and uses constant extra space.

## Solution

when the first missing positive is equal to n + 1 will be treated separately.

Approach 1: Index as a hash key.

 $[1, 2, 3, 4, 5, 6, 7, 8] \longrightarrow 9$ [1, 2, 48, 14, 15, 16, 17, 18] --> 3  $[1, 2, 3, 4, 5, 6, 7, 18] \longrightarrow 8$ 

number of elements is

n = 8

# max possible first missing number is n + 1 = 9

What does it mean - to get rid of, if one has to keep  $\mathcal{O}(N)$  time complexity and hence could not pop unwanted elements out? Let's just replace all these by 1 s. Data clean up : replace by 1s :

- negative numbers

- zeros

[3, 4, 1, 1, 1, 5, 1, 1, 2, 1] To ensure that the first missing positive is not 1, one has to verify the presence of 1 before proceeding to

# That would be simple, if one would be allowed to have a hash-map positive number -> its presence

 $[3, 4, 1, 1, 1, 5, 1, 1, 2, 1] \longrightarrow$ 

"O(1) space complexity" solution

with string

 $[3, 4, 1, 1, 1, 5, 1, 1, 2, 1] \longrightarrow$ 

number 6 is

missing

with hash-map

Sort of "dirty workaround" solution would be to allocate a string hash\_str with n zeros, and use it as a sort of hash map by changing hash\_str[i] to 1 each time one meets number i in the array.

"1111100000" number 5 is

present

positive sign of nums[3] element means that number 3 is not present (missing) in nums. To achieve that let's walk along the array (which after clean up contains only positive numbers), check each element value elem and change the sign of element nums[elem] to negative to mark that number elem is present in **nums**. Be careful with duplicates and ensure that the sign was changed only once. O(1) space complexity solution  $[3, 4, 1, 1, 1, 5, 1, 1, 2, 1] \longrightarrow$ 

because nums[5] < 0Now everything is ready to write down the algorithm.

• Walk along the array. Change the sign of a-th element if you meet number a. Be careful with

• Walk again along the array. Return the index of the first positive element.

duplicates : do sign change only once. Use index 0 to save an information about presence of number

• If on the previous step you didn't find the positive element in nums, that means that the answer is n +

1. Check if 1 is present in nums: yes

1/4

**С**ору

Next

Sort By -

Post

A Report

number 5 is present

[3, -4, -1, -1, -1, -5, 1, 1, 2, 1, 1]

number 6 is missing

because nums[6] > 0

Python Java class Solution:

# For example, if nums[1] is negative that means that number `1`

• Space complexity :  $\mathcal{O}(1)$  since this is a constant space solution.

Type comment here... (Markdown is supported)

coder\_xyzzy ★ 140 ② March 28, 2019 5:49 AM

destroying the contents of the array (though changing its order).

• Time complexity :  $\mathcal{O}(N)$  since all we do here is four walks along the array of length N.

# is present in the array.

M M def firstMissingPositive(self, nums): :type nums: List[int] :rtype: int n = len(nums)# Base case. if 1 not in nums: return 1 # nums = [1]if n == 1: return 2 # Replace negative numbers, zeros, # and numbers larger than n by 1s. # After this convertion nums will contain # only positive numbers. for i in range(n): if nums[i] <= 0 or nums[i] > n: nums[i] = 1# Use index as a hash key and number sign as a presence detector.

class Solution { nublic: Read More 32 A V 🕏 Share 🕈 Reply **SHOW 7 REPLIES** karbayev ★ 26 ② May 5, 2020 7:50 PM A Report Solution with a string of length of n is not O(1) space complexity, it's O(n) Haomin0817 ★30 ② April 27, 2020 5:08 AM Another O(n) solution use cycle sort: def firstMissingPositive(self, nums: List[int]) -> int: n = len(nums)Read More 11 A V C Share Reply **SHOW 1 REPLY** totsubo \* 730 • June 25, 2019 6:02 AM A Report One could get rid of all numbers larger than n as well, since the first missing positive is for sure smaller or equal to n + 1This isn't obvious and could use some more explanation. Read More 7 A V C Share Share **SHOW 2 REPLIES** manchesterunited 🛊 5 🗿 February 3, 2019 9:23 AM A Report Should the following statement: If array contains only one element and it's **not** 1, the answer is 2. be: If array contains only one element and it's 1, the answer is 2.

Read More

Read More

Read More

Nice solution, but it's possible to do this in one pass, considering each value at most twice, and not

2 A V C Share Reply **SHOW 4 REPLIES** nqm149 🛊 13 🧿 May 7, 2020 11:43 AM we can apply cyclic replacement which cost O(n) to swap and put all numbers in their correct positions in array

nublic int firstMissingPositive(int[] nums) {

manchesterunited \*5 @ February 3, 2019 9:57 AM

And ArrayIndexOutOfBoundsException while processing {1,3,3};

3 ∧ ∨ ♂ Share ¬ Reply

else if (nums[a] > 0)

**SHOW 2 REPLIES** 

Should:

parag\_meshram 🛊 6 ② April 24, 2020 11:32 PM The problem should specify that it is ok if you alter the existing array because to my surprise the LeetCode solution doesn't care about modifying the existing array. 1 A V C Share Reply 

every element is at most swapped 1 time, so o(n) time complexity Read More

Note:

First of all let's get rid of negative numbers and zeros since there is no need of them. One could get rid of all numbers larger than n as well, since the first missing positive is for sure smaller or equal to n + 1. The case

Data clean up

this operation. How to solve in-place Now there we have an array which contains only positive numbers in a range from 1 to n, and the problem is to find a first missing positive in  $\mathcal{O}(N)$  time and constant space.

O(N) space complexity solution

{1: 6, 2: 1, 3: 1, 4: 1, 5: 1, 6: missing}

The final idea is to use index in nums as a hash key and sign of the element as a hash value which is presence detector. For example, negative sign of nums[2] element means that number 2 is present in nums. The

Algorithm • Check if 1 is present in the array. If not, you're done and 1 is the answer. • If nums = [1], the answer is 2.

n since index n is not available.

[3, 4, -1, -2, 1, 5, 16, 0, 2, 0]

• If nums[0] > 0 return n.

1.

Implementation

> 18 19

20 21

22

23

24 25

26

27

**Complexity Analysis** 

O Previous

Comments: 35

Rate this article: \* \* \* \* \*

Preview

2409akshay 🛊 1 🗿 May 6, 2020 5:27 AM This isn't really O(1) solution. The thing is you're using that O(n) space of the input array itself. two pointers

- numbers larger than n = 10  $[3, 4, -1, -2, 1, 5, 16, 0, 2, 0] \longrightarrow$ 

for the array.

Let's not use this solution, but just take away a pretty nice idea to use index as a hash-key for a positive number.

Replace negative numbers, zeros, and numbers larger than n by 1 s.

1 2 3

nums[a] \*= -1;3 A V C Share Reply

1 A V C Share Reply SHOW 1 REPLY

(1234)

for each position i, find if i+1 exists in the array j increases from i to len(nums), for each step it tries to find i+1 by swaping 1 A V C Share Reply