Solution Review: Problem Challenge 2

We'll cover the following

Find the Smallest Missing Positive Number (medium)

Given an unsorted array containing numbers, find the **smallest missing positive number** in it.

Example 1:

```
Input: [-3, 1, 5, 4, 2]
Output: 3
Explanation: The smallest missing positive number is '3'
```

Example 2:

```
Input: [3, -2, 0, 1, 2]
Output: 4
```

Example 3:

```
Input: [3, 2, 5, 1]
Output: 4
```

Solution

This problem follows the **Cyclic Sort** pattern and shares similarities with Find the Missing Number with one big difference. In this problem, the numbers are not bound by any range so we can have any number in the input array.

However, we will follow a similar approach though as discussed in Find the Missing Number to place the numbers on their correct indices and ignore all numbers that are out of the range of the array (i.e., all negative numbers and all numbers greater than or equal to the length of the array). Once we are done with the cyclic sort we will iterate the array and the first index that does not have the correct number will be the smallest missing positive number!

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```python
def find_first_smallest_missing_positive(nums):
  i, n = 0, len(nums)
  while i < n:
    j = nums[i] - 1
    if nums[i] > 0 and nums[i] <= n and nums[i] != nums[j]:
      nums[i], nums[j] = nums[j], nums[i]  # swap
    else:
      i += 1

```

```
  for i in range(n):
    if nums[i] != i + 1:
      return i + 1

  return len(nums) + 1


def main():
  print(find_first_smallest_missing_positive([-3, 1, 5, 4, 2]))
  print(find_first_smallest_missing_positive([3, -2, 0, 1, 2]))
  print(find_first_smallest_missing_positive([3, 2, 5, 1]))


main()

```

Run Save Reset

Time complexity

The time complexity of the above algorithm is $O(n)$.

Space complexity

The algorithm runs in constant space $O(1)$.

Back

Problem Challenge 2

Next

Problem Challenge 3

Completed

Report an Issue Ask a Question