41. First Missing Positive

Given an unsorted integer array `nums`. Return the *smallest positive integer* that is *not present* in `nums`.

You must implement an algorithm that runs in `O(n)` time and uses `O(1)` auxiliary space.

Example 1:

Input: nums = [1,2,0]
Output: 3
Explanation: The numbers in the range [1,2] are all in the array.

Example 2:

Input: nums = [3,4,-1,1]
Output: 2
Explanation: 1 is in the array but 2 is missing.

Example 3:

Input: nums = [7,8,9,11,12]
Output: 1
Explanation: The smallest positive integer 1 is missing.

Constraints:

- `1 ≤ nums.length ≤ 105`
- `-231 ≤ nums[i] ≤ 231 - 1`

```
class Solution {
    private void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
    public int firstMissingPositive(int[] nums) {
        int n = nums.length;
        for(int i = 0; i<n; i++){
            while(nums[i] > 0 && nums[i] ≤ n && nums[i] ≠ nums[nums[i]
-1]){
                swap(nums, i, nums[i] -1);
            }
        }
        for(int i =0; i<n; i++){
            if(nums[i] ≠ i+1){
```

```
                return i+1;
            }
        }
        return n+1;
    }
}
```