# 题目

给你一个未排序的整数数组nums ，请你找出其中没有出现的最小的正整数。

请你实现时间复杂度为O(n)并且只使用常数级别额外空间的解决方案。

示例 1：

输入：nums = [1,2,0]

输出：3

示例 2：

输入：nums = [3,4,-1,1]

输出：2

示例 3：

输入：nums = [7,8,9,11,12]

输出：1

提示：

1 <= nums.length <= 5 \* 10^5

-2^31 <= nums[i] <= 2^31 - 1

# 分析

## 方法一：哈希表

class Solution {

public:

int firstMissingPositive(vector<int>& nums) {

int n = nums.size();

for (int& num: nums) {

if (num <= 0) {

num = n + 1;

}

}

for (int i = 0; i < n; ++i) {

int num = abs(nums[i]);

if (num <= n) {

nums[num - 1] = -abs(nums[num - 1]);

}

}

for (int i = 0; i < n; ++i) {

if (nums[i] > 0) {

return i + 1;

}

}

return n + 1;

}

};

**复杂度分析：**

时间复杂度：O(N)，其中N是数组的长度。

空间复杂度：O(1)。

## 方法二：哈希表+置换

解题思路：

1、遍历数组并交换元素位置：第一个循环遍历数组，对于每个元素 nums[i]，如果它是一个正整数且在 `[1, n]` 范围内，且它所在的位置不是应该在的位置（即 nums[nums[i] - 1] != nums[i]），则将它与应该在的位置上的元素交换。这样，经过一轮遍历，数组中每个正整数 nums[i] 都被放置到了索引为 nums[i] - 1 的位置上（索引从 0 开始）。

2、寻找缺失的最小正整数：第二个循环遍历数组，查找第一个不在正确位置上的元素。如果 nums[i] != i + 1，则说明 i + 1 是缺失的最小正整数，直接返回即可。

3、返回 n + 1：如果数组中所有的正整数都在正确的位置上，则说明缺失的最小正整数是 n + 1。

这种方法的时间复杂度是 O(n)，其中 n 是数组的长度，因为每个元素最多只需要交换一次就可以放到正确的位置上。空间复杂度是 O(1)，因为只使用了常数个额外变量。

class Solution {

public:

int firstMissingPositive(vector<int>& nums) {

int n = nums.size();

for (int i = 0; i < n; ++i) {

while (nums[i] > 0 && nums[i] <= n && nums[nums[i] - 1] != nums[i]) {

swap(nums[nums[i] - 1], nums[i]);

}

}

for (int i = 0; i < n; ++i) {

if (nums[i] != i + 1) {

return i + 1;

}

}

return n + 1;

}

};

**复杂度分析：**

时间复杂度：O(N)，其中N是数组的长度。

空间复杂度：O(1)。

或：

**思路：**

可以通过将数组视为哈希表来解决这个问题（一般缺失的数字都可以使用哈希表）。

把nums[i]放在i处，遍历i处不为i的即为结果。如果1-n齐全，判断nums[0]即可。

**代码：**

class Solution {

public:

int firstMissingPositive(vector<int>& nums) {

int n=nums.size();

if(n==0)

return 1;

for(int i=0;i<n;i++){

while(nums[i]>0 && nums[i]<n && nums[i]!=nums[nums[i]])

//注意此处应为while循环，而不是if

// 这里只使用n大小数组即可，不需要按照数组最大值设置

// 因为在n个大小数组内必然会找到最小的正数

swap(nums[i],nums[nums[i]]);

// 把num[i]放在第num[i]位置上

}

for(int i=1;i<n;i++){

if(nums[i]!=i)

return i;

}

// 如果数组中第一个数 nums[0] 等于 n，说明数组中包含了从 1 到 n 的所有正整数，此时缺失的最小正整数是 n+1；否则，缺失的最小正整数是 nums[0]。

return nums[0]==n?n+1:n;

}

};