# 题目

给你一个含n个整数的数组nums ，其中nums[i]在区间[1, n]内。请你找出所有在[1, n]范围内但没有出现在nums中的数字，并以数组的形式返回结果。

示例 1：

输入：nums = [4,3,2,7,8,2,3,1]

输出：[5,6]

示例 2：

输入：nums = [1,1]

输出：[2]

提示：

n == nums.length

1 <= n <= 105

1 <= nums[i] <= n

进阶：你能在不使用额外空间且时间复杂度为O(n)的情况下解决这个问题吗? 你可以假定返回的数组不算在额外空间内。

# 分析

## 方法一：哈希表

**思路：**

我们可以用一个哈希表记录数组nums中的数字，由于数字范围均在[1,n]中，记录数字后我们再利用哈希表检查[1,n]中的每一个数是否出现，从而找到缺失的数字。

由于数字范围均在[1,n]中，我们也可以用一个长度为n的数组来代替哈希表。这一做法的空间复杂度是O(n)的。我们的目标是优化空间复杂度到O(1)。

注意到nums的长度恰好也为n，能否让nums充当哈希表呢？

由于nums的数字范围均在[1,n]中，我们可以利用这一范围之外的数字，来表达「是否存在」的含义。

具体来说，遍历nums，每遇到一个数x，就让nums[x−1]增加n。由于nums 中所有数均在[1,n]中，增加以后，这些数必然大于n。最后我们遍历nums，若nums[i]未大于n，就说明没有遇到过数i+1。这样我们就找到了缺失的数字。

注意，当我们遍历到某个位置时，其中的数可能已经被增加过，因此需要对n取模来还原出它本来的值。

代码：

class Solution {

public:

vector<int> findDisappearedNumbers(vector<int>& nums) {

int n = nums.size();

for (auto& num : nums) {

int x = (num - 1) % n;

nums[x] += n;

}

vector<int> ret;

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

if (nums[i] <= n) {

ret.push\_back(i + 1);

}

}

return ret;

}

};

复杂度分析：

时间复杂度：O(n)。其中n是数组nums的长度。

空间复杂度：O(1)。返回值不计入空间复杂度。

另一种写法：

class Solution {

public:

vector<int> findDisappearedNumbers(vector<int>& nums) {

vector<int> res;

if(nums.empty()) return nums;

for(int i=0;i<nums.size();i++)

{

int index=(nums[i]-1)%nums.size();

nums[index]+=nums.size();

}

for(int i=0;i<nums.size();i++)

{

if(nums[i]<=nums.size())

res.push\_back(i+1);

}

return res;

}

};

## 方法二：优先队列

这种空间复杂度较高，为O(n)。