<https://leetcode.com/problems/missing-number/description/>

Given an array nums containing n distinct numbers in the range [0, n], return *the only number in the range that is missing from the array.*

**Example 1:**

Input: nums = [3,0,1]

Output: 2

Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

**Example 2:**

Input: nums = [0,1]

Output: 2

Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

**Example 3:**

Input: nums = [9,6,4,2,3,5,7,0,1]

Output: 8

Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.

**Constraints:**

n == nums.length

1 <= n <= 104

0 <= nums[i] <= n

All the numbers of nums are **unique**.

**Follow up:** Could you implement a solution using only O(1) extra space complexity and O(n) runtime complexity?

**Attempt 1: 2023-09-17**

**Solution 1:  XOR (10 min)**

class Solution {

public int missingNumber(int[] nums) {

int n = nums.length;

// 我们可以把下标当成上边所说的完整的序列。因为下标没有 n，

// 所以初始化 result = n。然后把两个序列的数字依次异或即可

int result = n;

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

result ^= nums[i] ^ i;

}

return result;

}

}

**Refer to**

<https://leetcode.wang/leetcode-268-Missing-Number.html>

# **解法三**

又到了神奇的异或的方法了，[这里](https://leetcode.com/problems/missing-number/discuss/69791/4-Line-Simple-Java-Bit-Manipulate-Solution-with-Explaination) 的解法。

[136 题](https://leetcode.wang/leetcode-136-Single-Number.html) 详细的介绍了异或的一个性质，a ⊕ a = 0，也就是相同数字异或等于 0。

这道题的话，相当于我们有两个序列。

一个完整的序列， 0 到 n。

一个是 0 到 n 中缺少了一个数字的序列。

把这两个序列合在一起，其实就变成了[136 题](https://leetcode.wang/leetcode-136-Single-Number.html) 的题干——所有数字都出现了两次，只有一个数字出现了一次，找出这个数字。

假如合起来的数字序列是 a b a b c c d ，d 出现了一次，也就是我们缺失的数字。

如果我们把给定的数字相互异或会发生什么呢？因为异或满足交换律和结合律，所以结果如下。

a ⊕ b ⊕ a ⊕ b ⊕ c ⊕ c ⊕ d

= ( a ⊕ a ) ⊕ ( b ⊕ b ) ⊕ ( c ⊕ c ) ⊕ d

= 0 ⊕ 0 ⊕ 0 ⊕ d

= d

这样我们就找了缺失的数字了。

**代码的话，我们可以把下标当成上边所说的完整的序列。因为下标没有 n，所以初始化 result = n。**

**然后把两个序列的数字依次异或即可。**

public int missingNumber(int[] nums) {

int result = nums.length;

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

result = result ^ nums[i] ^ i;

}

return result;

}

**Refer to**

<https://leetcode.com/problems/missing-number/solutions/69791/4-line-simple-java-bit-manipulate-solution-with-explaination/>

The basic idea is to use XOR operation. We all know that a^b^b =a, which means two xor operations with the same number will eliminate the number and reveal the original number.

In this solution, I apply XOR operation to both the index and value of the array. In a complete array with no missing numbers, the index and value should be perfectly corresponding( nums[index] = index), so in a missing array, what left finally is the missing number.

public int missingNumber(int[] nums) {

int xor = 0, i = 0;

for (i = 0; i < nums.length; i++) {

xor = xor ^ i ^ nums[i];

}

return xor ^ i;

}

**Refer to**

<https://grandyang.com/leetcode/268/>

这题还有一种解法，使用位操作Bit Manipulation来解的，用到了异或操作的特性，相似的题目有[Single Number 单独的数字,](http://www.cnblogs.com/Given an array of integers, every element appears three times except for one. Find that single one.  Note: Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory) [Single Number II 单独的数字之二](http://www.cnblogs.com/grandyang/p/4263927.html)和[Single Number III 单独的数字之三](http://www.cnblogs.com/grandyang/p/4741122.html)。**那么思路是既然0到n之间少了一个数，我们将这个少了一个数的数组合0到n之间完整的数组异或一下，那么相同的数字都变为0了，剩下的就是少了的那个数字了**，参加代码如下：

class Solution {

public:

int missingNumber(vector<int>& nums) {

int res = 0;

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

res ^= (i + 1) ^ nums[i];

}

return res;

}

};