**难度: 简单**

给定一个包含 [0, n] 中 n 个数的数组 nums ，找出 [0, n] 这个范围内没有出现在数组中的那个数。

进阶：

你能否实现线性时间复杂度、仅使用额外常数空间的算法解决此问题?

示例 1：

输入：nums = [3,0,1]

输出：2

解释：n = 3，因为有 3 个数字，所以所有的数字都在范围 [0,3] 内。2 是丢失的数字，因为它没有出现在 nums 中。

示例 2：

输入：nums = [0,1]

输出：2

解释：n = 2，因为有 2 个数字，所以所有的数字都在范围 [0,2] 内。2 是丢失的数字，因为它没有出现在 nums 中。

示例 3：

输入：nums = [9,6,4,2,3,5,7,0,1]

输出：8

解释：n = 9，因为有 9 个数字，所以所有的数字都在范围 [0,9] 内。8 是丢失的数字，因为它没有出现在 nums 中。

示例 4：

输入：nums = [0]

输出：1

解释：n = 1，因为有 1 个数字，所以所有的数字都在范围 [0,1] 内。1 是丢失的数字，因为它没有出现在 nums 中。

提示：

n == nums.length

1 <= n <= 104

0 <= nums[i] <= n

nums 中的所有数字都 独一无二

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/missing-number

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1:**

1). 先进行排序

2). 然后从0开始比较数组的值是否相等

3). 不想等就直接返回对应的下标

4). 到最后都没有返回,说明全部都是相同的,所以直接返回数组最后一个的值+1就好了

|  |
| --- |
| public int missingNumber(int[] nums) {  //先进行排序  Arrays.sort(nums);  for (int i = 0; i < nums.length; i++) {  if (nums[i] != i) {  return i;  }  }  return nums.length;  } |

**思路2:**

1). 先把原本的值进行相加

2). 把从0到数组长度+1的值相加

3). 把2)的值 减 1)的值 等到结果

|  |
| --- |
| int sum = 0;  for (int i = 0; i < nums.length; i++) {  sum += nums[i];  }    // int sum2 = 0;  // for (int i = 0; i < nums.length+1; i++) {  // sum2 += i;  // }  //高斯求和公式  int sum2 = (nums.length\*(nums.length+1))/2;  return sum2 - sum;  } |