值几毛钱的两数之和黑盒子

1. 两数之和

简单 ♥ 相关标签 🔒 相关企业 🗘 提示 🗛

给定一个整数数组 nums 和一个整数目标值 target ,请你在该数组中找出 **和为目标值** target 的那 **两个** 整数,并返回它们的数组下标。你可以假设每种输入只会对应一个答案,并且你不能使用两次相同的元素。

你可以按任意顺序返回答案。

示例 1:

输入: nums = [2,7,11,15], target = 9

输出: [0,1]

解释: 因为 nums[0] + nums[1] == 9 , 返回 [0, 1] 。

示例 2:

输入: nums = [3,2,4], target = 6

输出: [1,2]

示例 3:

输入: nums = [3,3], target = 6

输出: [0,1]

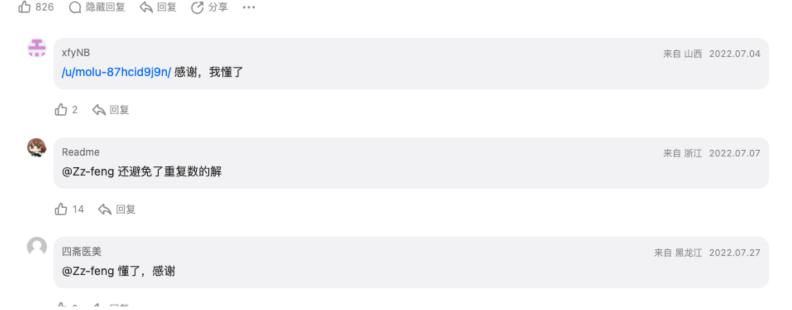
提示:

- 2 <= nums.length <= 104
- $-10^9 \le nums[i] \le 10^9$
- −10⁹ <= target <= 10⁹
- 只会存在一个有效答案

进阶: 你可以想出一个时间复杂度小于 0(n2) 的算法吗?

使用哈希表的方法,一开始简单的以为先遍历数组建立哈希表,再遍历数组在哈希表里找值,看了题解的代码,一度认为代码少了一次遍历,想了半 天才明白错的是我。

两个元素x,y必然是一前一后出现的,如果存在符合条件的解,在遍历到x时,哈希表里没有符合的y,此时把x加入到了哈希表里,当遍历到y时,就可以在哈希表里找到对应的x了,所以只需要一次遍历,妙啊。 原来LeetCode第一题都不会的那个人就是我。



代码

};

```
Java | C++ | C | Python3 | Golang
class Solution {
public:
   vector<int> twoSum(vector<int>& nums, int target) {
       unordered map<int, int> hashtable;
       for (int i = 0; i < nums.size(); ++i) {
           auto it = hashtable.find(target - nums[i]);
           if (it != hashtable.end()) {
               return {it->second, i};
           hashtable[nums[i]] = i; 感觉可以先把unordered
                                 map当作是二维数组
       return {};
```

初始nums:

值	2	7	11	15
下标	0	1	2	3

初始unordered_

map:

值		
键		

第一次循环后

unordered_map:

值	0		
键	2		

因为官方的关键差异化就是这么干的:

1. 两数之和

· 方法 2: 查找表法



map

value





1



2

输出: [0, 3]