题目地址(两数之和)

https://leetcode-cn.com/problems/two-sum

入选理由

1. 两数之和的经典程度不用我多说了，大家都应该知道。

2. 这道题不仅是入门题目，而且和后面我们要讲的双指针有联系。

题目描述

给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。
你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。
示例：

给定 nums = [2, 7, 11, 15], target = 9

因为 nums[0] + nums[1] = 2 + 7 = 9
所以返回 [0, 1]

标签

- 哈希表
- 双指针

难度

- 简单

前置知识

- 哈希表

思路 - 暴力

思路很简单，遍历数据，对每一个出现的 num 判断其另一半 `target - num` 是否也出现在数组中即可

代码

代码支持：Java

```
public class Solution {

    public int[] twoSum(int[] nums, int target) {

        for(int i = 0; i < nums.length; i++)
            for(int j = i + 1; j < nums.length; j++)
                if(nums[i] + nums[j] == target)
                    return new int[]{i, j};

        return new int[]{-1, -1};
    }
}
```

复杂度分析

- 空间复杂度: $O(1)$
- 时间复杂度：$O(n^2)$, n为数组长度

思路

上面是用于搜索整个数组的方式来判断 `target - num` 是否也存在 `nums`，我们也可以用哈希表记录所有已经遍历过的数字，判断 `target - num` 是否出现时，直接查表即可。

代码

哈希表是非常常用的时间换空间的方式

代码支持：Java

```
class Solution {

    public int[] twoSum(int[] nums, int target) {

        Map<Integer, Integer> map = new HashMap<>();
```

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

            if (map.containsKey(nums[i]))
                return new int[]{map.get(nums[i]), i};

            map.put(target - nums[i], i);
        }

        return new int[]{};
    }
}
```

复杂度分析

- 空间复杂度: $O(n)$
- 时间复杂度: $O(n)$

上一页 下一页