[0001] 两数之和

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

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，你不能重复利用这个数组中同样的元素。

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**]

Related Topics
- 数组
- 哈希表

题目代码

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {

    }
};
```

题目剖析&信息挖掘

题目中有以下信息未明确：

- nums的长度
- 数组中整数的范围
- 题目没有明确指出一定可以找到2数之和等于目标数

以上问题都要事先明确。

本题是一个搜索题，主要方法有暴力枚举(复杂度高)，哈希查找，排序后用双指针法查找。

解题思路

方法一 哈希表查找法

分析

- 这道题是一个查找问题。可以通**过哈希表**加速
- 可以通过遍历数组中的每个元素`nums[i]`，查找`target-nums[i]`
- 查找`target-nums[i]`，可以遍历整个数组`nums`，总的时间复杂度为`$O(n^2)$`；也可以使用哈希表记录已查元素及其索引，中的时间复杂度为`$O(n)$`

思路

- 创建哈希表：`<nums[i], index>`。用于存储元素及其索引
- 遍历数组的每个元素`nums[i]`。检查哈希表中是否存在`target-nums[i]`，如果存在，返回两个元素的索引；否则，将当前元素存储到哈希表中，处理下一个元素
- 如果遍历完所有元素，查找失败，返回空数组

注意

- 边界检查：数组为空
- 处理查找失败情况

知识点

- 数组
- 查找
- 哈希表

复杂度

- 时间复杂度：O(n)
- 空间复杂度：O(n)

参考

- https://www.cnblogs.com/grandyang/p/4130379.html

代码实现

```
//#include <unordered_map>
//
//using namespace std;

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        // 输入检查
        if(nums.empty())
            return {};
        // 创建哈希表 <num, index>, 加速查找
        unordered_map<int, int> m;
```

```
        for(int i=0; i<nums.size(); i++){
            if(m.count(target - nums[i]))
                return {i, m[target-nums[i]]};
            m[nums[i]] = i; // <num, index>
        }
        return {};
    }
};
```

方法二 双指针法

分析

- 先对数组的排序，再用双指针法从两端开始慢慢往中间移
- 复杂度：排序为$O(nlogn)$，遍历数据为$O(n)$，总体为$O(nlogn)$

思路

- 先对数组（带下标信息）排序
- 初始i=0, j=len(arr)-1
- 终止条件 i>=j
- 如果 arr[i].val+arr[j].var=target 返回 {arr[i].ind, arr[j].ind} 作为结果
- 如果arr[i].val+arr[j].var<target, 由于arr[i].val < arr[j].var, 故i++
- 如果arr[i].val+arr[j].var>target, 由于arr[i].val > arr[j].var, 故j--

注意

- 题目要的是下标，所以数字与下标要关联排序
- 加减过程中结果有可能超出int32
- 排序后，arr[i].ind, arr[j].ind 大小关系不是一定的，在最后返回时要做判断

知识点

- 数组
- 排序
- 双指针法

复杂度

- 时间复杂度：$O(nlogn)$
- 空间复杂度：$O(n)$

参考

代码实现

```
typedef long long lld;

class Num {
public:
```

```
    int ind;
    lld val; // 使用长整型，防止加法溢出
    Num(int i, int v) : ind(i), val(v) {}
};

bool cmp(Num a, Num b) {
    return a.val < b.val;
}

class Solution {
public:
    vector<int> twoSum(vector<int> &nums, int target) {
        vector<Num> arr;

        // 1. 将数字和下标同步排序
        for (int i = 0; i < nums.size(); i++) {
            arr.push_back(Num(i, nums[i]));
        }
        sort(arr.begin(), arr.end(), cmp);

        // 2. 用双指针法查找
        vector<int> res(2, 0);
        for (int i = 0, j = arr.size() - 1; i < j;) {
            lld sum = arr[i].val + arr[j].val;
            if (sum == target) { // 找到答案
                res[0] = arr[i].ind;
                res[1] = arr[j].ind;
                break;
            }

            if (sum < target) i++;
            else j--;
        }


        if (res[0] > res[1]) {// 下标按小到大给出
            swap(res[0], res[1]);
        }
        return res;
    }
};

/*
[2,7,11,15]
9
[2147483647,1, -1, 0]
2147483647
[2147483646, 1, 2, 3]
3
```

```
[ 1, 2, 3,2147483646]
3
[2147483647, 1, 2, 3]
3
*/
```

相关题目

https://leetcode-cn.com/problems/3sum/