# LeetCode 第 1 号问题：两数之和

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 1 号问题：两数之和。题目难度为 Easy，目前通过率为 45.8% 。

### 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]
```

### 题目解析

使用查找表来解决该问题。

设置一个 map 容器 record 用来记录元素的值与索引，然后遍历数组 nums。

- 每次遍历时使用临时变量 complement 用来保存目标值与当前值的差值
- 在此次遍历中查找 record ，查看是否有与 complement 一致的值，如果查找成功则返回查找值的索引值与当前变量的值 i
- 如果未找到，则在 record 保存该元素与索引值 i

### 动画描述

### 代码实现

**C++**

```
// 1. Two Sum
// https://leetcode.com/problems/two-sum/description/
// 时间复杂度: O(n)
// 空间复杂度: O(n)
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> record;
        for(int i = 0 ; i < nums.size() ; i ++){

            int complement = target - nums[i];
            if(record.find(complement) != record.end()){
                int res[] = {i, record[complement]};
```

```
                return vector<int>(res, res + 2);
            }

            record[nums[i]] = i;
        }
        return {};
    }
};
```

### C

```c
// 1. Two Sum
// https://leetcode.com/problems/two-sum/description/
// 时间复杂度: O(n)
// 空间复杂度: O(n)
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* twoSum(int* nums, int numsSize, int target, int* returnSize){
    int *ans=(int *)malloc(2 * sizeof(int));
    int i,j;
    bool flag=false;
    for(i=0;i<numsSize-1;i++)
    {
        for(j=i+1;j<numsSize;j++)
        {
            if(nums[i]+nums[j] == target)
            {
                ans[0]=i;
                ans[1]=j;
                flag=true;
            }
        }
    }
    if(flag){
        *returnSize = 2;
    }
    else{
        *returnSize = 0;
    }
    return ans;
}
```

### Java

```java
// 1. Two Sum
// https://leetcode.com/problems/two-sum/description/
// 时间复杂度: O(n)
// 空间复杂度: O(n)
class Solution {
    public int[] twoSum(int[] nums, int target) {
```

```
        int l = nums.length;
        int[] ans=new int[2];
        int i,j;
        for(i=0;i<l-1;i++)
        {
            for(j=i+1;j<l;j++)
            {
                if(nums[i]+nums[j] == target)
                {
                    ans[0]=i;
                    ans[1]=j;
                }
            }
        }

        return ans;

    }
}
```

**Python**

```
# 1. Two Sum
# https://leetcode.com/problems/two-sum/description/
# 时间复杂度: O(n)
# 空间复杂度: O(n)
class Solution(object):
    def twoSum(self, nums, target):
        l = len(nums)
        print(nums)
        ans=[]
        for i in range(l-1):
            for j in range(i+1,l):
                if nums[i]+nums[j] == target:
                    ans.append(i)
                    ans.append(j)
                    print([i,j])
                    break
        return ans
```