两数之和--哈希版本

题述

1. 两数之和

难度 简单 13279

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。但是，数组中同一个元素在答案里不能重复出现。

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]
```

思路

我们来看一下使用数组和set来做哈希法的局限：

- 数组的大小是受限制的，而且如果元素很少，而哈希值很大会造成内存空间的浪费
- set是一个集合，里面放的元素只能够是一个key，而两数之和这道题目，不仅要判断y是否存在，还要记录y的下标位置，因为要返回x和y的下标，所以set也不能用。

这时，我们就有必要选择另一种数据结构：map，map是一种key value存储结构，可以用key保存数值，用value来保存数值所在的下标

C++中map，有三种类型：

映射	底层实现	是否有序	数值是否可以重复	能否更改数值	查询效率	增删效率
std::map	红黑树	key有序	key不可重复	key不可修改	$O(\log n)$	$O(\log n)$
std::multimap	红黑树	key有序	key可重复	key不可修改	$O(\log n)$	$O(\log n)$
std::unordered_map	哈希表	key无序	key不可重复	key不可修改	$O(1)$	$O(1)$

std::unordered_map 底层实现为哈希表，std::map 和std::multimap 的底层实现是红黑树。

同理，std::map 和std::multimap 的key也是有序的（这个问题也经常作为面试题，考察对语言容器底层的理解）。 更多哈希表的理论知识请看关于哈希表，你该了解这些！ 。

这道题目中并不需要key有序，选择std::unordered_map 效率更高！

题解

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target)
    {
        //这道题我们也可以使用map来解决
        //map是一种key value的存储结构 可以用key保存数值，用value保存数值所在的下标

        std::unordered_map<int,int> map;
        for(int i=0;i<nums.size();i++)
        {
            auto iter=map.find(target-nums[i]); //智能指针
            if(iter!=map.end())
            {
                return {iter->second,i};
            }
            map.insert(pair<int,int>(nums[i],i));
        }
        return {};
    }
};
```

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	8 ms	10.4 MB	C++	2022/01/27 17:53	添
通过	268 ms	9.9 MB	C++	2021/09/21 16:00	添
通过	360 ms	9.8 MB	C++	2021/08/28 14:58	添
通过	272 ms	9.8 MB	C++	2021/08/23 14:53	添
通过	276 ms	9.9 MB	C++	2021/08/23 14:52	添
通过	12 ms	9 MB	C++	2020/12/06 19:54	添
解答错误	N/A	N/A	C++	2020/12/06 19:53	添
解答错误	N/A	N/A	C++	2020/12/06 19:52	添

思考

理解底层STL