题目描述

1. 两数之和

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。

Solu:

暴力解法(Brute Force)

```
class Solution {
public:
   vector<int> twoSum(vector<int>& nums, int target) {
       for (int i = 0; i < nums.size()-1; i++)
       {
           for (int j = i+1; j < nums.size(); j++)
           {
               if (nums[i] + nums[j]==target)
               {
                   return { i,j };
               }
           }
       }
       return {};
   }
};
```

优化思路:

观察：当我们检查数字7时，其实是在找 9-7=2 这个数是否存在于数组中

新思路：

创建一个"备忘录"（哈希表）记录已经见过的数字

对于每个数字，先检查"目标值减去当前数"的结果是否在备忘录中

如果在就直接返回，否则把当前数存入备忘录

优化解法(使用unorderd_map来实现哈希表)

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map <int, int>result;
        for (int i = 0; i < nums.size(); i++)
        {
            int remain = target - nums[i];
            if (result.count(remain))
            {
                return { result[remain],i };
            }
            result[nums[i]] = i;
        }
        return {};
    }
};
```

Points:

哈希表(使用unordered_map来实现哈希表)

```
#include<iostream>
#include <unordered_map>
using namespace std;
int main()
{
    unordered_map<string, int> phonebook;
    phonebook["Alice"] = 123456; // 像数组一样赋值
    phonebook.insert({ "Bob", 789012 });
    cout << "Alice: " << phonebook["Alice"] << endl;
    cout << "Bob: " << phonebook["Bob"] << endl;//验证
    return 0;
}
```

Comparison:

暴力法：时间O(n^2)，空间O(1)

优化法：时间O(n)，空间O(n)

关键点：

哈希表的查找速度极快（平均O(1)时间）

只需要遍历数组一次（O(n)时间）

空间换时间：用了额外O(n)空间存储哈希表