1. 两数之和

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> unorderedMap;
        for (int i = 0; i < nums.size(); ++i) {
            auto  it=unorderedMap.find(target-nums[i]);
            if(it!=unorderedMap.end()){
                return {it->second,i};
            }else{
                unorderedMap[nums[i]]=i;
            }

        }
        return  {};
    }
};
```

49. 字母异位词分组

```
class Solution {
public:
    vector <vector<string>> groupAnagrams(vector <string> &strs) {
        unordered_map <string, vector<string>> unorderedMap;
        for (string &str: strs) {
            string tmp = str;
            sort(tmp.begin(), tmp.end());
            unorderedMap[tmp].emplace_back(str);
        }
        vector <vector<string>> ans;
        for (auto it: unorderedMap) {
            ans.emplace_back(it.second);
        }
        return ans;
    }
};


class Solution2 {
public:
    vector <vector<string>> groupAnagrams(vector <string> &strs) {
        vector <vector<string>> ans;
        map <string, vector<string>> hashTable;

        for (auto str: strs) {
            string sts = string(26, '0');
            for (auto c: str) {
                ++sts[c - 'a'];
```

```
            }
            hashTable[sts].emplace_back(str);
        }
        for (auto it: hashTable) {
            ans.emplace_back(it.second);
        }
        return ans;
    }
};
```

128. 最长连续序列

```
class Solution {
public:
    int longestConsecutive(vector<int> &nums) {
        unordered_set<int> numsSet;

        for (const int &num: nums) {
            numsSet.insert(num);
        }
        int longestStreak = 0;

        for (const int &num: nums) {

            if (!numsSet.count(num - 1)) {
                int currentNum = num;
                int currentStreak = 1;
                while (numsSet.count(currentNum + 1)) {
                    currentNum += 1;
                    currentStreak += 1;
                }
                longestStreak=max(longestStreak,currentStreak);
            }
        }
        return longestStreak;
    }
};
```

283. 移动零

```
class Solution {
public:
    void moveZeroes(vector<int>& nums) {
        int i=0,j=0;
        while(j<nums.size()){
            if(nums[j]){
                swap(nums[i++],nums[j]);
            }
            j++;
        }
    }
```

```
};
```

11. 盛最多水的容器

```
class Solution{
public:
    int maxArea(vector<int> &height) {
        int ans=0,i=0,j=height.size()-1;
        while (i<j){
            ans= height[i]< height[j] ? max(ans,(j-i)*height[i++]):max(ans,(j-
i)*height[j--]);
        }
        return ans;
    }
};
```

15. 三数之和

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        int n = nums.size();
        sort(nums.begin(), nums.end());
        vector<vector<int>> ans;
        // 枚举 a
        for (int first = 0; first < n; ++first) {
            // 需要和上一次枚举的数不相同
            if (first > 0 && nums[first] == nums[first - 1]) {
                continue;
            }
            // c 对应的指针初始指向数组的最右端
            int third = n - 1;
            int target = -nums[first];
            // 枚举 b
            for (int second = first + 1; second < n; ++second) {
                // 需要和上一次枚举的数不相同
                if (second > first + 1 && nums[second] == nums[second - 1]) {
                    continue;
                }
                // 需要保证 b 的指针在 c 的指针的左侧 和 b 和 c 的指针 大于 a
                while (second < third && nums[second] + nums[third] > target) {
                    --third;
                }
                // 如果指针重合，随着 b 后续的增加
                // 就不会有满足 a+b+c=0 并且 b<c 的 c 了，可以退出循环
                if (second == third) {
                    break;
```

```
                }
                if (nums[second] + nums[third] == target) {
                    ans.push_back({nums[first], nums[second], nums[third]});
                }
            }
        }
        return ans;
    }
};
```

42. 接雨水

```
```