383. 赎金信

Monday, July 25, 2022 9:24 AM

https://leetcode.cn/problems/ransom-note/

简单

给你两个字符串：ransomNote 和 magazine ，判断 ransomNote 能不能由 magazine 里面的字符构成。

如果可以，返回 true ；否则返回 false 。

magazine 中的每个字符只能在 ransomNote 中使用一次。

示例 1：

输入：ransomNote = "a", magazine = "b"

输出：false

示例 2：

输入：ransomNote = "aa", magazine = "ab"

输出：false

示例 3：

输入：ransomNote = "aa", magazine = "aab"

输出：true

解法一：暴力，hashmap存magazine 再遍历ransomNote判断即可。

解法二：new int[26]存即可

解法一code：

```
class Solution {
    public boolean canConstruct(String ransomNote, String magazine) {
        HashMap<Character,Integer> mag = new HashMap<>();
        for(char c:magazine.toCharArray()){
            if(!mag.containsKey(c)){
                mag.put(c,1);
            }else{
                mag.put(c,mag.get(c)+1);
            }
        }
        for(char c:ransomNote.toCharArray()){
            if(!mag.containsKey(c)){
                return false;
            }else{
                if(mag.get(c)>0){
                    mag.put(c,mag.get(c)-1);
                }else{
                    return false;
                }
```

```
            }
        }
        return true;
    }
}
```

15. 三数之和
https://leetcode.cn/problems/3sum/
中等
给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？请你找出所有和为 0 且不重复的三元组。

注意：答案中不可以包含重复的三元组。

示例 1：
输入：nums = [-1,0,1,2,-1,-4]
输出：[[-1,-1,2],[-1,0,1]]
示例 2：
输入：nums = []
输出：[]
示例 3：
输入：nums = [0]
输出：[]
思路：
三指针，for循环作为第一个指针，中间while循环中left从i+1，right从nums.length-1开始；找三个指针和为0的情况，保存结果。
注意结果数组内部不能出现完全相同的三元组，如 [[1,2,1],[1,2,1]]。

```
class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
        List<List<Integer>> result = new ArrayList<>();
        if(nums==null||nums.length==0) //判空
            return result;
        Arrays.sort(nums); //排序目标数组
        //三指针遍历，i作为第一个指针
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] > 0) { //数组递增，第一个指针已大于0，得不到和为0的结果，返回res
                return result;
            }

            if (i > 0 && nums[i] == nums[i - 1]) { //i>0防止i-1越界，第一个指针与上一个数相同时，跳过，防止重复
                continue;
```

```
            }

            int left = i + 1; //第二个指针，左指针
            int right = nums.length - 1; //第三个指针，右指针
            while (right > left) {
                int sum = nums[i] + nums[left] + nums[right];
                if (sum > 0) {
                    right--;
                } else if (sum < 0) {
                    left++;
                } else {
                    result.add(Arrays.asList(nums[i], nums[left], nums[right])); //sum=0结果加
入列表

                    while (right > left && nums[right] == nums[right - 1]) right--; //右指针去
重
                    while (right > left && nums[left] == nums[left + 1]) left++; //左指针去重

                    right--;
                    left++;
                }
            }
        }
        return result;
    }
}
```