剑指 Offer 57 - II. 和为s的连续正数序列

Thursday, June 16, 2022 8:49 AM

https://leetcode.cn/problems/he-wei-sde-lian-xu-zheng-shu-xu-lie-lcof/
简单

输入一个正整数 target ，输出所有和为 target 的连续正整数序列（至少含有两个数）。
序列内的数字由小到大排列，不同序列按照首个数字从小到大排列。

示例 2：

输入：target = 15
输出：[[1,2,3,4,5],[4,5,6],[7,8]]

意思是：输出序列，序列有序，连续，不能完全重复，至少两个数。

我的思路

暴力遍历，双循环，以 i 为起点，往后遍历，没有则 i+1 下一轮

code:

```
class Solution {
    public int[][] findContinuousSequence(int target) {
        List<int[]> res=new ArrayList<>(); //存结果
        int top=(target)/2; //因为至少两个数，所以 i遍历到 i/2 就够了
        for(int i=1;i<=top;i++){
            int sum=0;
            for(int j=i;j<=top+1;j++){ // j每次从I 出发，但上界要比i大1. 15/2=7, j可以取8
                sum+=j;
                if(sum>target){ // 大于目标数了，退出 j循环，开始下一轮 i循环
                    break;
                }else if(sum==target){ // 等于目标数，遍历 i-j 加入数组。
                    int[] cur=new int[j-i+1];
                    for(int k=0;k<cur.length;k++){
                        cur[k]=i+k;
                    }
                    res.add(cur);
                }
                 // 小于目标数，直接下轮 j循环。
            }
        }
        return res.toArray(new int[res.size()][]);
    }
}
```

剑指 Offer 58 - I. 翻转单词顺序

https://leetcode.cn/problems/fan-zhuan-dan-ci-shun-xu-lcof/

简单

输入一个英文句子，翻转句子中单词的顺序，

但单词内字符的顺序不变。

为简单起见，标点符号和普通字母一样处理。

例如输入字符串"I am a student. "，则输出"student. a am I"。

示例 2：

输入: " hello world! "

输出: "world! hello"

解释: 输入字符串可以在前面或者后面包含多余的空格，但是反转后的字符不能包括。

示例 3：

输入: "a good   example"

输出: "example good a"

解释: 如果两个单词间有多余的空格，将反转后单词间的空格减少到只含一个。

我的思路：

双指针，初始化至末位，从后往前遍历，一个遇到的空格，保存本段内容；

两个指针再放到前一位

...

code:

```
class Solution {
    public String reverseWords(String s) {
        s = s.trim(); // 删除首尾空格
        int j = s.length() - 1, i = j; // 初始化均指向末位
        StringBuilder res = new StringBuilder(); //保存结果
        while(i >= 0) {
            while(i >= 0 && s.charAt(i) != ' ')
                i--; // 搜索首个空格
            res.append(s.substring(i + 1, j + 1) + " "); // 添加单词
            while(i >= 0 && s.charAt(i) == ' ')
                i--; // 跳过单词间空格
            j = i; // j 指向下个单词的尾字符
        }
        return res.toString().trim(); // 转化为字符串并返回
    }
}
```