455 分发饼干

题目

假设你是一位很棒的家长，想要给你的孩子们一些小饼干。但是，每个孩子最多只能给一块饼干。

对每个孩子 i，都有一个胃口值 g[i]，这是能让孩子们满足胃口的饼干的最小尺寸；并且每块饼干 j，都有一个尺寸 s[j] 。如果 s[j] >= g[i]，我们可以将这个饼干 j 分配给孩子 i ，这个孩子会得到满足。你的目标是尽可能满足越多数量的孩子，并输出这个最大数值。

输入输出

```
输入: g = [1,2,3], s = [1,1]
输出: 1
```

思路

将当前满足该孩子饥饿度最小得饼干分配给该孩子，这样就可以尽可能多的满足饥饿度大的孩子。所以考虑如下步骤：

1. 设置两个数组的定位变量 index_s，index_g，用于数组定位。
2. 为两个数组从小到大排序，注意JS中的sort()函数只是将数组转为字符串排序，所以不适用与所有排序。所以要对sort()函数进行处理。
3. 判断两个索引有没有移动到数组尾，没有则循环处理。判断当前饥饿度是否能<饼干大小，如果小于则分配，并且饼干和人的数组索引都+1，如果饥饿度>饼干大小，则饼干索引+1。记录成功的次数。

代码

```
var findContentChildren = function(g, s) {
    let flag = new Array(s.length).fill(0);
    let index_s = 0;
    let index_g = 0;
    let res = 0;
    g.sort(function (a,b) {
        if (a < b ) {
            return -1;
        }
        if (a > b ) {
            return 1;
        }
        return 0;
    });
    s.sort(function (a,b) {
        if (a < b ) {
            return -1;
        }
        if (a > b ) {
            return 1;
         }
        return 0;
    });
    console.log(g,s)
```

```
    while(index_s!=s.length && index_g!=g.length){
        if(g[index_g] <= s[index_s]){
            console.log("饥饿",g[index_g],"饼干",s[index_s])
            index_g++;
            index_s++;
            res++;
        }else{
            index_s++;
        }
    }
    return res
};
```

435. 无重叠区间

题目

给定一个区间的集合，找到需要移除区间的最小数量，使剩余区间互不重叠。

可以认为区间的终点总是大于它的起点。
区间 [1,2] 和 [2,3] 的边界相互“接触”，但没有相互重叠。

输入输出

```
输入: [ [1,2], [2,3], [3,4], [1,3] ]
输出: 1
```

```
输入: [ [1,2], [1,2], [1,2] ]
输出: 2
```

```
输入: [ [1,2], [2,3] ]
输出: 0
```

思路

1. 先对二维数组每一个一维数组排序
2. 从第一个一维数组开始判断，如果当前数组的start小于当前的max_end则，证明区间重叠，count计数，如果当前数组的start大于，则max_end重新赋值

代码

```
var eraseOverlapIntervals = function(intervals) {
    let count = 0;
    for(let i = 1; i <= intervals.length-1; i++){
        for(let j= 1;j <= intervals.length-i;j++){
            let temp
            //console.log("比较end",intervals[i][1],intervals[i-1][1])
            if(intervals[j][1] < intervals[j-1][1]){
                temp = intervals[j-1];
                intervals[j-1] = intervals[j];
                intervals[j] = temp;
            }
        }
```

```
    }
    console.log(intervals)
    let max_end = intervals[0][1];
    for(let i = 1; i < intervals.length; i++){
        //console.log("intervals[i][0]=",intervals[i][0],"max_end",max_end)
        if(intervals[i][0] < max_end){
            count++;
        }else{
            max_end = intervals[i][1];
        }
    }
    return count
};
```

明日计划

1. js函数部分看完
2. 看番茄钟项目
3. leetcode