455-分发饼干

题述

455. 分发饼干

难度 简单 436

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。

示例 1:

```
输入: g = [1,2,3], s = [1,1]
输出: 1
解释:
你有三个孩子和两块小饼干，3个孩子的胃口值分别是：1,2,3。
虽然你有两块小饼干，由于他们的尺寸都是1，你只能让胃口值是1的孩子满足。
所以你应该输出1。
```

示例 2:

```
输入: g = [1,2], s = [1,2,3]
输出: 2
解释:
你有两个孩子和三块小饼干，2个孩子的胃口值分别是1,2。
你拥有的饼干数量和尺寸都足以让所有孩子满足。
所以你应该输出2.
```

思路

贪心算法的核心思路就是 **局部最优推出全局最优** 就这么简单是的！

这道题，为了满足更多的小孩，我们就应该尽可能地避免饼干尺寸的浪费！

那么，就应该尽可能满足以下规则：

- 大尺寸饼干优先满足大胃口孩子

那么，我们可以进行以下贪心操作：

1. 先将饼干和小孩数组排好序
2. 然后从后向前遍历小孩数组，用大饼干优先满足大胃口，并统计小孩数量

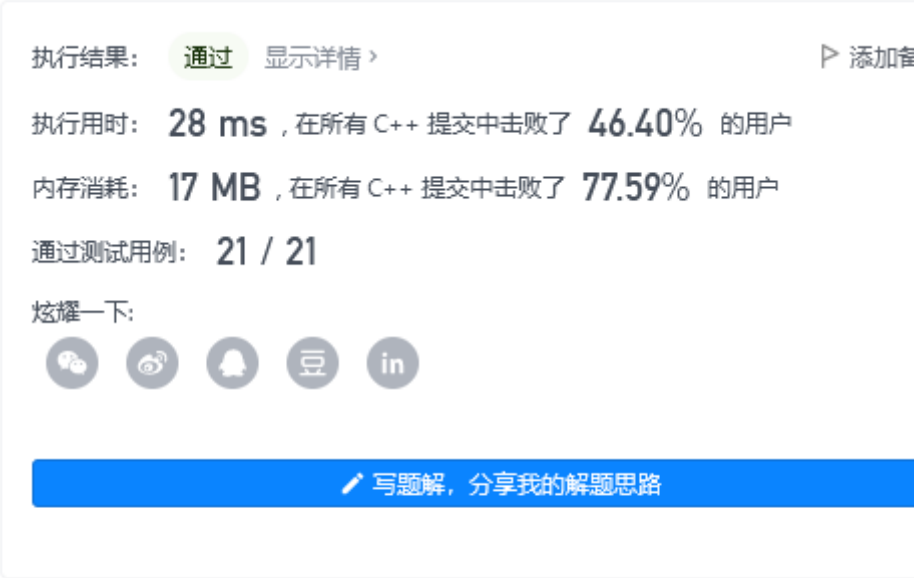
题解

大饼干优先满足大胃口

```
class Solution {
public:
    int findContentChildren(vector<int>& g, vector<int>& s)
    {
        //贪心--大饼干优先喂饱大胃口
        //局部最优推全局最优
        //g为胃口值    s为饼干
        sort(g.begin(),g.end());
        sort(s.begin(),s.end());
        int index=s.size()-1;   //饼干数组的下标
        int result=0;
        for(int i=g.size()-1;i>=0;i--)
        {
            //先遍历胃口较大的孩子
            if(index >= 0 && s[index] >= g[i])
            {
                result++;
                index--;
            }
        }
        return result;
    }
};
```

小饼干满足小胃口

```
class Solution {
public:
    int findContentChildren(vector<int>& g, vector<int>& s) {
        sort(g.begin(),g.end());
        sort(s.begin(),s.end());
        int index = 0;
        for(int i = 0;i < s.size();++i)
        {
            if(index < g.size() && g[index] <= s[i]){
                index++;
            }
        }
        return index;
    }
};
```



提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	28 ms	17 MB	C++	2022/02/05 20:29	

Python

```python
class Solution:
    def findContentChildren(self, g: List[int], s: List[int]) -> int:
        #贪心策略一 ： 优先考虑饼干尺寸
        # s为尺寸数组 g为胃口数组
        g.sort()
        s.sort()
        result = 0
        for i in range(len(s)):
            if result < len(g) and s[i] >= g[result]:
                #小饼干优先喂给小胃口
                result += 1
        return result
```

```python
class Solution:
    # 思路2：优先考虑胃口
    def findContentChildren(self, g: List[int], s: List[int]) -> int:
        g.sort()
        s.sort()
        start, count = len(s) - 1, 0
        for index in range(len(g) - 1, -1, -1): # 先喂饱大胃口
            if start >= 0 and g[index] <= s[start]:
                start -= 1
                count += 1
        return count
```

思考

想清楚局部最优，想清楚全局最优，感觉局部最优是可以推出全局最优，并想不出反例，那么就试一试贪心