分发饼干

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.

解题思想：

贪心+排序

贪心思想尽可能满足多的孩子，则先满足需求小的孩子，则先排序

```
int cmp(int* a, int* b) {
    return *a - *b;
}
int findContentChildren(int* g, int gSize, int* s, int sSize){
    //贪心思想：尽量小的饼干满足小需求的孩子  先排序
    qsort(g,gSize,sizeof(int),cmp);
```

```
    qsort(s,sSize,sizeof(int),cmp);
    int n1=gSize;
    int n2=sSize;
    int cout=0;
    //当其中一个遍历完就结束
    for(int i=0,j=0;i<n1&&j<n2;i++,j++)
    {
        while(j<n2 && g[i]>s[j])//当前饼干不能满足孩子，则找下一块饼干
        {
            j++;
        }
        //此时，找打满足孩子饼干的位置，再判断饼干是否还有 如果有则满足之前那一个孩子
        if(j<n2)
        {
            cout++;
        }
    }
    return cout;
}
```