滑动窗口——特殊的双指针法

2022年8月19日 10:20

数组操作除双指针外，另一个重要的方法就是滑动窗口
所谓滑动窗口，就是**不断地调节子序列的起始位置和终止位置，从而得到我们想要的结果。**
在暴力解法中，一个 for 循环滑动窗口的**起始位置**，另一个 for 循环滑动窗口的**终止位置**，用两个for来完成一个不断搜索区间的过程

若只用一个for循环，那么应该用来表示滑动窗口的**终止位置**，否则又会陷入暴力解法的怪圈

以leetcode209为例

给定一个含有 `n` 个正整数的数组和一个正整数 `target` 。

找出该数组中满足其和 $\geq$ `target` 的长度最小的 **连续子数组** $[nums_l, nums_{l+1}, \ldots, nums_{r-1}, nums_r]$ ，并返回其长度。如果不存在符合条件的子数组，返回 `0` 。

示例 1：

```
输入：target = 7, nums = [2,3,1,2,4,3]
输出：2
解释：子数组 [4,3] 是该条件下的长度最小的子数组。
```

在本题中实现滑动窗口，主要确定如下三点：

1. 窗口内是什么？
2. 如何移动窗口的起始位置？
3. 如何移动窗口的结束位置？

窗口是 满足其 和 >= s的长度最小的连续子数组
窗口的起始位置如何移动：如果当前窗口的值大于s了，窗口就要向前移动，也就是要**缩小了**
窗口的结束位置如何移动：窗口的结束位置就是遍历数组的指针，也就是for循环里的索引
关键在于窗口的起始位置如何移动

此块代码的精髓就是动态调节滑动窗口的起始位置

```
while (sum >= s) {
    subLength = (j - i + 1); // 取子序列的长度
    result = result < subLength ? result : subLength;
    sum -= nums[i++]; // 这里体现出滑动窗口的精髓之处，不断变更i（子序列的起始位置）
}
```

```
class Solution {
public:
    int minSubArrayLen(int target, vector<int>& nums) {
        int i=0,j=0;
        int sum=0;
        int result=INT_MAX,sublen=0;
        for(;i<nums.size();i++){
            sum+=nums[i];
            while(sum>=target){
                sublen=(i-j)+1;
                result=result>sublen?sublen:result;
                sum-=nums[j++];
            }
        }
        return result==INT_MAX ? 0:result;
    }
};
```

!