

滑动窗口算法

作者: [wlgzs-sjl](#)

原文链接: <https://ld246.com/article/1583504881015>

来源网站: [链滴](#)

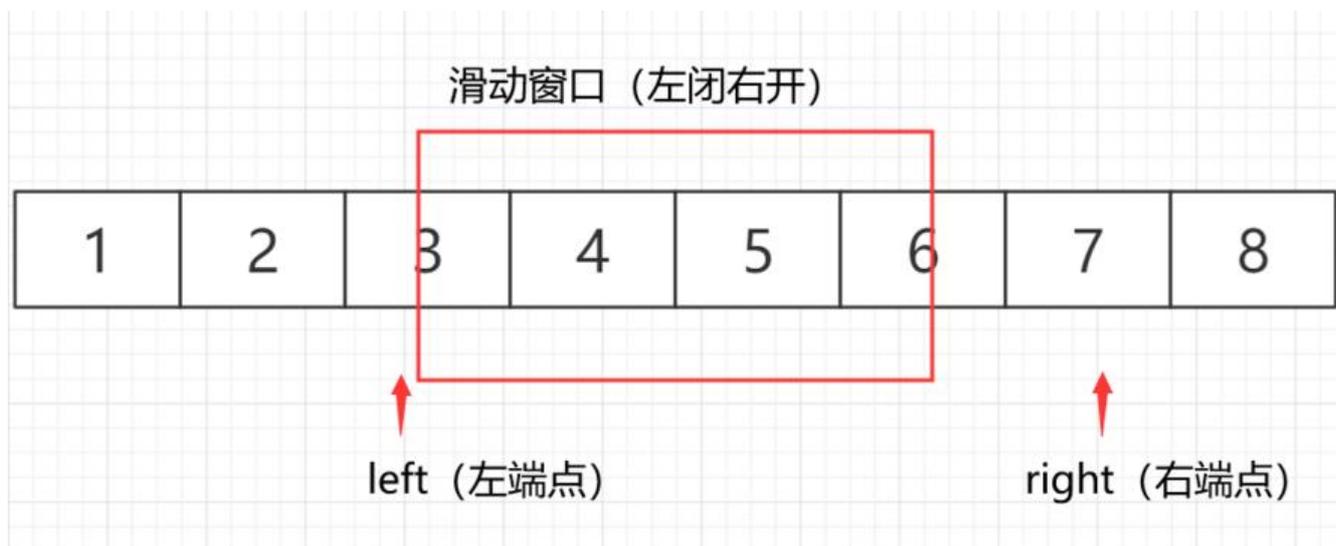
许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



1、什么是滑动窗口

  滑动窗口可以看成数组中框起来的一个部分。在一些数组类题目中，我们可用滑动窗口来观察可能的候选结果。当滑动窗口从数组的左边滑到了右边，我们就可以从所有的候选结果中找到最优的结果。

  我们设滑动窗口的左边界为 $left$ ，右边界为 $right$ ，则滑动窗口框起来的是一个左闭右开区间 $[left, right)$ 。为了编程的方便，滑动窗口一般表示成一个左闭右开区间。在一开始， $left=1, right=1$ ，滑动窗口位于序列的最左侧，窗口大小为零。



  滑动窗口的重要性质是：**窗口的左边界和右边界永远只能右移动，而不能向左移动。**这是为了保证滑动窗口的时间复杂度是 $O(n)$ 。如果左右边界向左移的话，这叫做“回溯”，算法的时间复杂度就可能不止 $O(n)$ 。

2、例题讲解

例题链接: [和为s的连续正数序列](#)

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 1:

输入: target = 9

输出: [[2,3,4],[4,5]]

示例 2:

输入: target = 15

输出: [[1,2,3,4,5],[4,5,6],[7,8]]

限制:

$1 \leq target \leq 10^5$

- 当窗口的和小于 target 的时候, 窗口的和需要增加, 所以要扩大窗口, 窗口的右边界向右移动
- 当窗口的和大于 target 的时候, 窗口的和需要减少, 所以要缩小窗口, 窗口的左边界向右移动
- 当窗口的和恰好等于 target 的时候, 我们需要记录此时的结果。设此时的窗口为 [left, right), 那我们已经找到了一个 left 开头的序列, 也是唯一一个 left 开头的序列, 接下来需要找 left+1 开头的序列, 所以窗口的左边界要向右移动

```
class Solution {
    public int[][] findContinuousSequence(int target) {
        int left = 1;
        int right = 1;
        int sum = 0;
        List<int[]> res=new ArrayList<>();
        while (left <= target/2) {
            if (sum<target){
                sum+=right;
                right++;
            }else if (sum>target){
                sum-=left;
                left++;
            }else {
                int[] arr=new int[right-left];
                for (int i = left; i <right ; i++) {
                    arr[i-left]=i;
                }
                res.add(arr);
                sum-=left;
                left++;
            }
        }
        return res.toArray(new int[res.size()][]);
    }
}
```

}

3、相关例题

1. [滑动窗口中位数](#)
2. [滑动窗口最大值](#)
3. [滑动谜题](#)