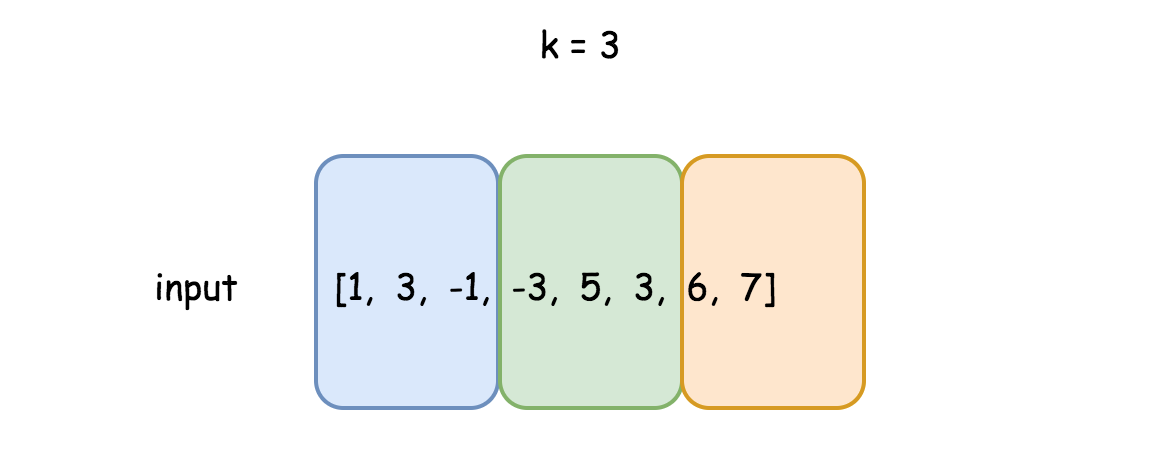
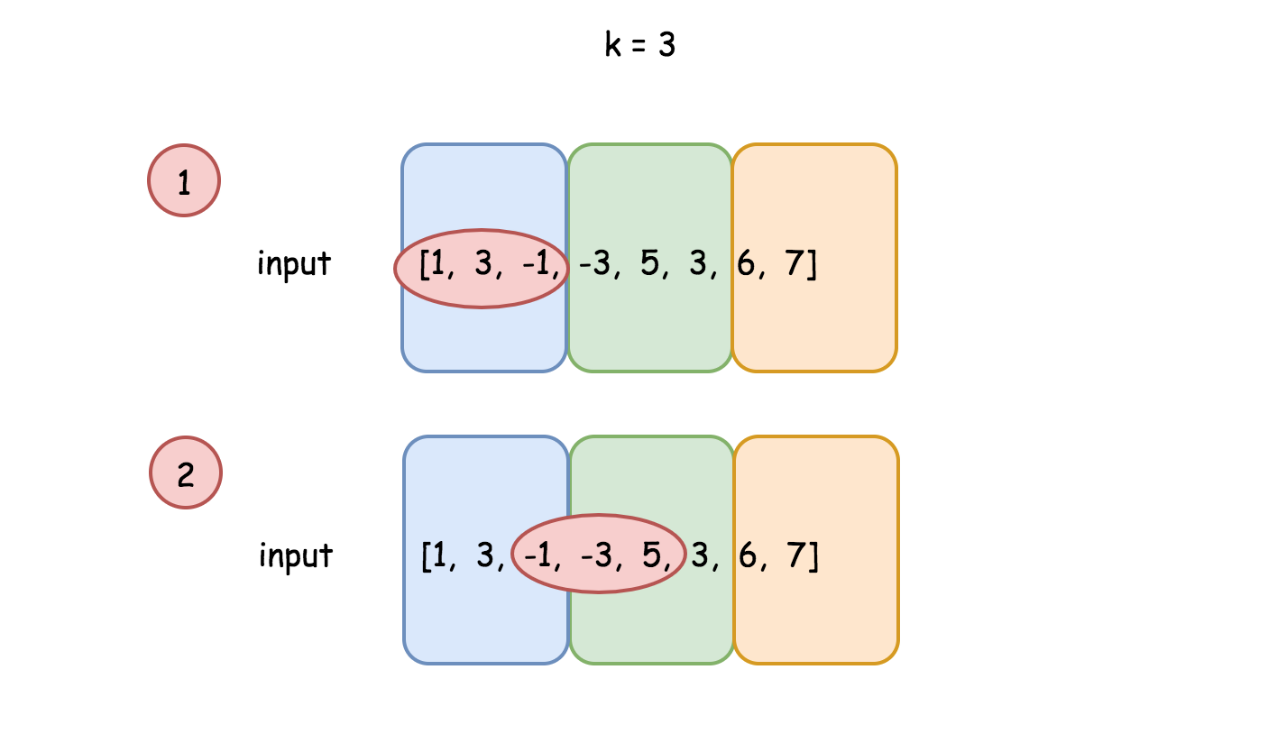
方法三: 动态规划

直觉

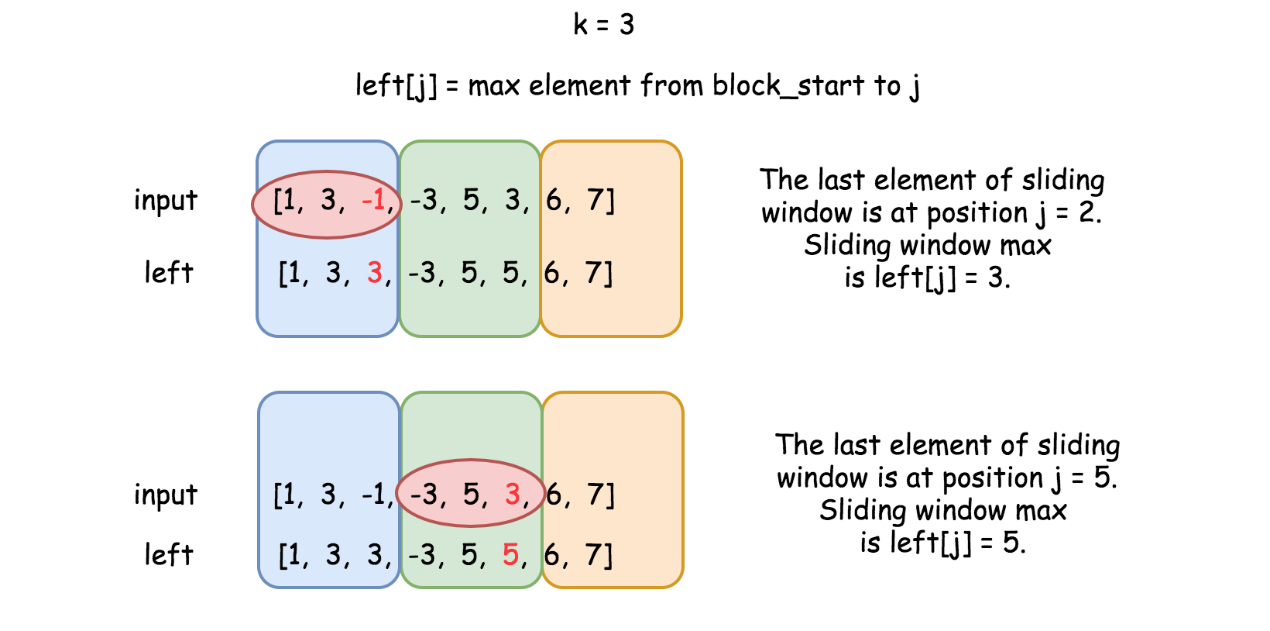
这是另一个 {O}(N)O(N) 的算法。本算法的优点是不需要使用 数组 / 列表 之外的任何数据结构。算法的思想是将输入数组分割成有 k 个元素的块。若 n % k != 0，则最后一块的元素个数可能更少。



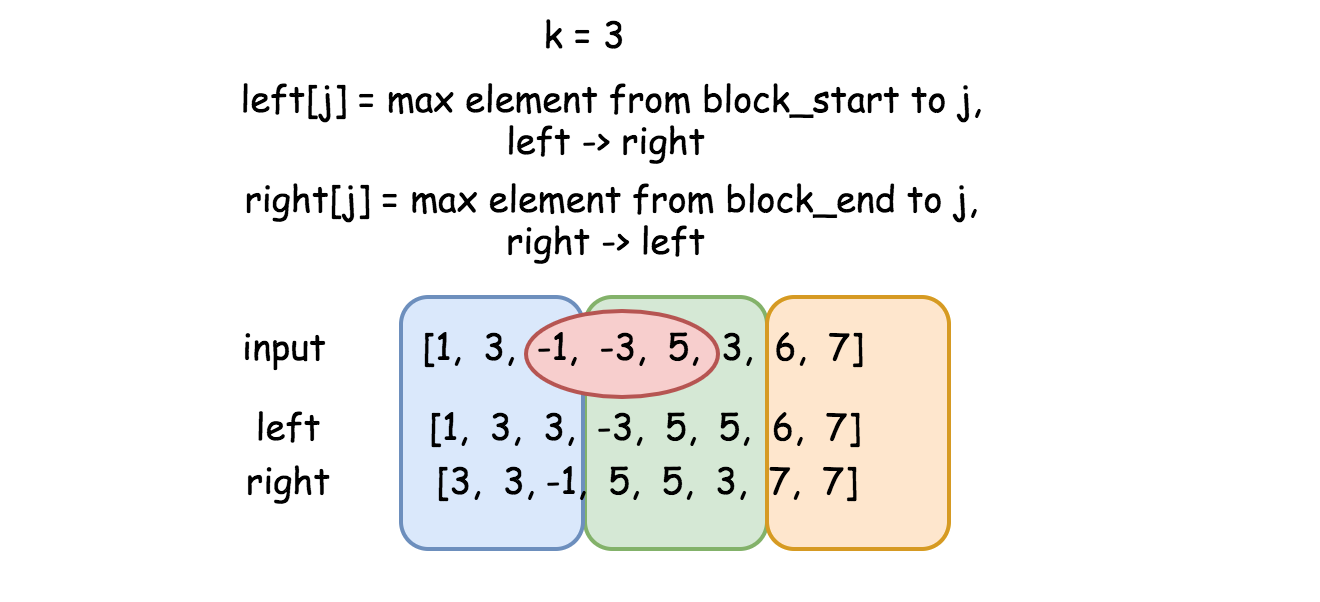
开头元素为 i ，结尾元素为 j 的当前滑动窗口可能在一个块内，也可能在两个块中。



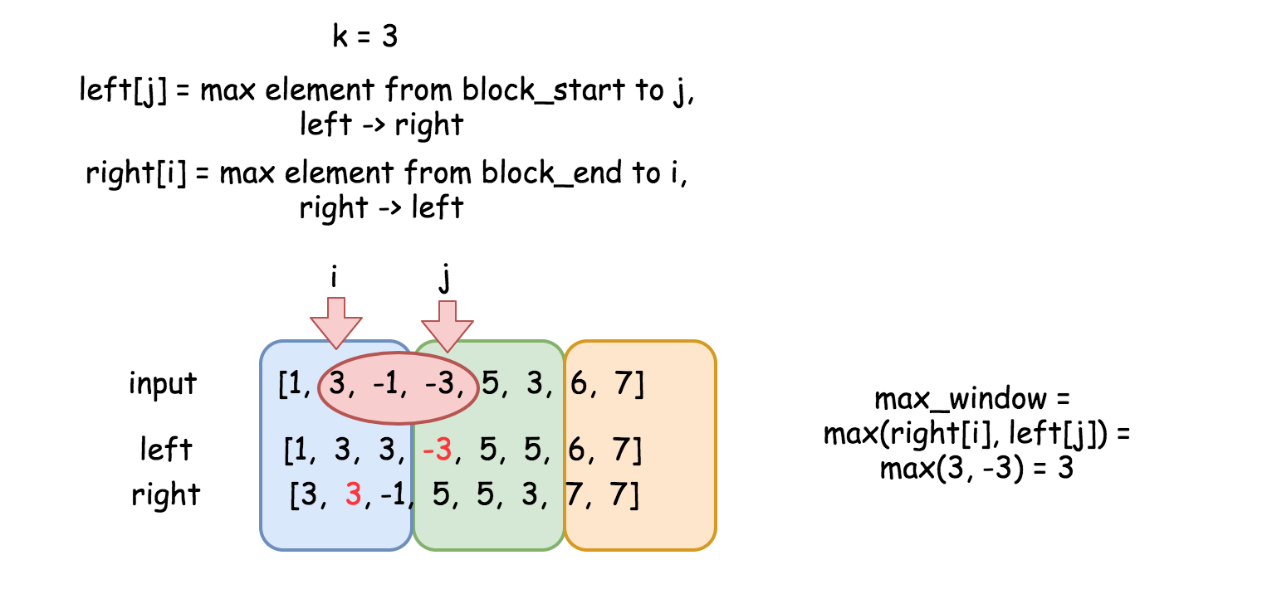
情况 1 比较简单。 建立数组 left， 其中 left[j] 是从块的开始到下标 j 最大的元素，方向 左->右。



为了处理更复杂的情况 2，我们需要数组 right，其中 right[j] 是从块的结尾到下标 j 最大的元素，方向 右->左。right 数组和 left 除了方向不同以外基本一致。



两数组一起可以提供两个块内元素的全部信息。考虑从下标 i 到下标 j的滑动窗口。 根据定义，right[i] 是左侧块内的最大元素， left[j] 是右侧块内的最大元素。因此滑动窗口中的最大元素为 max(right[i], left[j])。



算法

算法十分直截了当：从左到右遍历数组，建立数组 left。从右到左遍历数组，建立数组 right。

建立输出数组 max(right[i], left[i + k - 1])，其中 i 取值范围为 (0, n - k + 1)。

作者：LeetCode

链接：https://leetcode-cn.com/problems/sliding-window-maximum/solution/hua-dong-chuang-kou-zui-da-zhi-by-leetcode-3/

来源：力扣（LeetCode）

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。