0995. K 连续位的最小翻转次数

ITCharge 大约 3 分钟

- 标签：位运算、队列、数组、前缀和、滑动窗口
- 难度：困难

题目链接

- 0995. K 连续位的最小翻转次数 - 力扣

题目大意

描述：给定一个仅包含 0 和 1 的数组 $nums$，再给定一个整数 k。进行一次 k 位翻转包括选择一个长度为 k 的（连续）子数组，同时将子数组中的每个 0 更改为 1，而每个 1 更改为 0。

要求：返回所需的 k 位翻转的最小次数，以便数组没有值为 0 的元素。如果不可能，返回 -1。

说明：

- **子数组**：数组的连续部分。
- $1 <= nums.length <= 105$。
- $1 <= k <= nums.length$。

示例：

- 示例 1：

```py
输入：nums = [0,1,0], K = 1
输出：2
解释：先翻转 A[0]，然后翻转 A[2]。
```

- 示例 2：

```py
输入：nums = [0,0,0,1,0,1,1,0], K = 3
输出：3
解释：
翻转 A[0],A[1],A[2]: A变成 [1,1,1,1,0,1,1,0]
翻转 A[4],A[5],A[6]: A变成 [1,1,1,1,1,0,0,0]
翻转 A[5],A[6],A[7]: A变成 [1,1,1,1,1,1,1,1]
```

解题思路

思路 1：滑动窗口

每次需要翻转的起始位置肯定是遇到第一个元素为 0 的位置开始反转，如果能够使得整个数组不存在 0，即返回 ans 作为反转次数。

同时我们还可以发现：

- 如果某个元素反转次数为奇数次，元素会由 $0 \to 1$，$1 \to 0$。
- 如果某个元素反转次数为偶数次，元素不会发生变化。

每个第 i 位置上的元素只会被前面 $[i \quad +1, i-1]$ 的元素影响。所以我们只需要知道前面 $k-1$ 个元素翻转次数的奇偶性就可以了。

同时如果我们知道了前面 $k-1$ 个元素的翻转次数就可以直接修改 $nums[i]$ 了。

我们使用 $flip\_count$ 记录第 i 个元素之前 $k-1$ 个位置总共被反转了多少次，或者 $flip\_count$ 是大小为 $k-1$ 的滑动窗口。

- 如果前面第 $k-1$ 个元素翻转了奇数次，则如果 $nums[i] == 1$，则 $nums[i]$ 也被翻转成了 0，需要再翻转 1 次。
- 如果前面第 $k-1$ 个元素翻转了偶数次，则如果 $nums[i] == 0$，则 $nums[i]$ 也被翻转成为了 0，需要再翻转 1 次。

这两句写成判断语句可以写为：`if (flip_count + nums[i]) % 2 == 0:`。

因为 $0 <= nums[i] <= 1$，所以我们可以用 0 和 1 以外的数，比如 2 来标记第 i 个元素发生了翻转，即 `nums[i] = 2`。这样在遍历到第 i 个元素时，如果有 $nums[i-k] == 2$，则说明 $nums[i-k]$ 发生了翻转。同时根据 $flip\_count$ 和 $nums[i]$ 来判断第 i 位是否需要进行翻转。

整个算法的具体步骤如下：

- 使用 res 记录最小翻转次数。使用 $flip\_count$ 记录窗口内前 $k - 1 $ 位元素的翻转次数。
- 遍历数组 $nums$，对于第 i 位元素：
  - 如果 $i - k >= 0$，并且 $nums[i - k] == 2$，需要缩小窗口，将翻转次数减一。（此时窗口范围为 $[i - k + 1, i - 1]$）。
  - 如果 $(flip\_count + nums[i]) \mod 2 == 0$，则说明 $nums[i]$ 还需要再翻转一次，将 $nums[i]$ 标记为 2，同时更新窗口内翻转次数 $flip\_count$ 和答案最小翻转次数 ans。
- 遍历完之后，返回 res。

思路 1：代码

```py
class Solution:
    def minKBitFlips(self, nums: List[int], k: int) -> int:
        ans = 0
        flip_count = 0
        for i in range(len(nums)):
            if i - k >= 0 and nums[i - k] == 2:
                flip_count -= 1
            if (flip_count + nums[i]) % 2 == 0:
                if i + k > len(nums):
                    return -1
                nums[i] = 2
                flip_count += 1
                ans += 1
        return ans
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 为数组 $nums$ 的长度。
- **空间复杂度**：$O(1)$。