Problem 3086: Minimum Moves to Pick K Ones

Problem Information

Difficulty: Hard
Acceptance Rate: 20.90%
Paid Only: No
Tags: Array, Greedy, Sliding Window, Prefix Sum

Problem Description

You are given a binary array `nums` of length `n`, a **positive** integer `k` and a **non-negative** integer `maxChanges`.

Alice plays a game, where the goal is for Alice to pick up `k` ones from `nums` using the **minimum** number of **moves**. When the game starts, Alice picks up any index `aliceIndex` in the range `[0, n - 1]` and stands there. If `nums[aliceIndex] == 1` , Alice picks up the one and `nums[aliceIndex]` becomes `0`(this **does not** count as a move). After this, Alice can make **any** number of **moves** (**including** **zero**) where in each move Alice must perform **exactly** one of the following actions:

* Select any index `j != aliceIndex` such that `nums[j] == 0` and set `nums[j] = 1`. This action can be performed **at** **most** `maxChanges` times. * Select any two adjacent indices `x` and `y` (`|x - y| == 1`) such that `nums[x] == 1`, `nums[y] == 0`, then swap their values (set `nums[y] = 1` and `nums[x] = 0`). If `y == aliceIndex`, Alice picks up the one after this move and `nums[y]` becomes `0`.

Return _the**minimum** number of moves required by Alice to pick **exactly**_`k` _ones_.

Example 1:

Input: nums = [1,1,0,0,0,1,1,0,0,1], k = 3, maxChanges = 1

Output: 3

Explanation: Alice can pick up `3` ones in `3` moves, if Alice performs the following actions in each move when standing at `aliceIndex == 1`:

* At the start of the game Alice picks up the one and `nums[1]` becomes `0`. `nums` becomes `[1,**_0_** ,0,0,0,1,1,0,0,1]`. * Select `j == 2` and perform an action of the first type. `nums` becomes `[1,**_0_** ,1,0,0,1,1,0,0,1]` * Select `x == 2` and `y == 1`, and perform an action of the second type. `nums` becomes `[1,**_1_** ,0,0,0,1,1,0,0,1]`. As `y == aliceIndex`, Alice picks up the one and `nums` becomes `[1,**_0_** ,0,0,0,1,1,0,0,1]`. * Select `x == 0` and `y == 1`, and perform an action of the second type. `nums` becomes `[0,**_1_** ,0,0,0,1,1,0,0,1]`. As `y == aliceIndex`, Alice picks up the one and `nums` becomes `[0,**_0_** ,0,0,0,1,1,0,0,1]`.

Note that it may be possible for Alice to pick up `3` ones using some other sequence of `3` moves.

Example 2:

Input: nums = [0,0,0,0], k = 2, maxChanges = 3

Output: 4

Explanation: Alice can pick up `2` ones in `4` moves, if Alice performs the following actions in each move when standing at `aliceIndex == 0`:

* Select `j == 1` and perform an action of the first type. `nums` becomes `[**_0_** ,1,0,0]`. * Select `x == 1` and `y == 0`, and perform an action of the second type. `nums` becomes `[**_1_** ,0,0,0]`. As `y == aliceIndex`, Alice picks up the one and `nums` becomes `[**_0_** ,0,0,0]`. * Select `j == 1` again and perform an action of the first type. `nums` becomes `[**_0_** ,1,0,0]`. * Select `x == 1` and `y == 0` again, and perform an action of the second type. `nums` becomes `[**_1_** ,0,0,0]`. As `y == aliceIndex`, Alice picks up the one and `nums` becomes `[**_0_** ,0,0,0]`.

Constraints:

* `2 <= n <= 105` * `0 <= nums[i] <= 1` * `1 <= k <= 105` * `0 <= maxChanges <= 105` * `maxChanges + sum(nums) >= k`

Code Snippets

C++:

```
class Solution {
public:
long long minimumMoves(vector<int>& nums, int k, int maxChanges) {

}
};
```

Java:

```
class Solution {
public long minimumMoves(int[] nums, int k, int maxChanges) {

}
}
```

Python3:

```
class Solution:
def minimumMoves(self, nums: List[int], k: int, maxChanges: int) -> int:
```