1004. Max Consecutive Ones III

Given a binary array `nums` and an integer `k`, return *the maximum number of consecutive* `1`*'s in the array if you can flip at most* `k` `0`'s.

Example 1:

```
Input: nums = [1,1,1,0,0,0,1,1,1,1,0], k = 2
Output: 6
Explanation: [1,1,1,0,0,1,1,1,1,1,1]
Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.
```

Example 2:

```
Input: nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3
Output: 10
Explanation: [0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1]
Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.
```

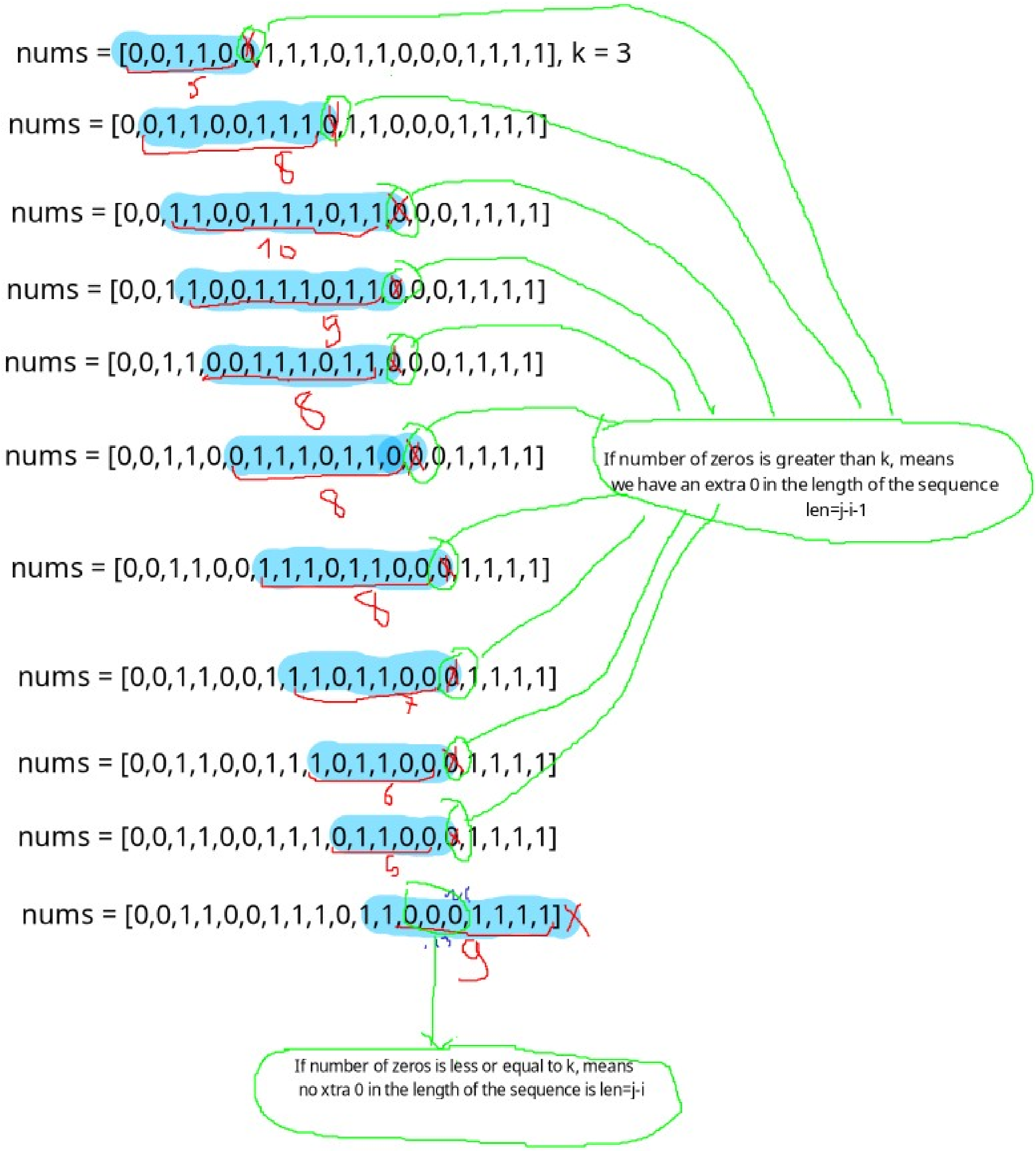
Constraints:

- $1 \leq nums.length \leq 10^5$
- `nums[i]` is either `0` or `1`.
- `0 <= k <= nums.length`

1004. Max Consecutive Ones III

Brute force approach

Try all possible windows (subarrays)



```
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1]
```

If number of zeros is greater than k, means we have an extra 0 in the length of the sequence len=j-i-1

If number of zeros is less or equal to k, means no xtra 0 in the length of the sequence is len=j-i

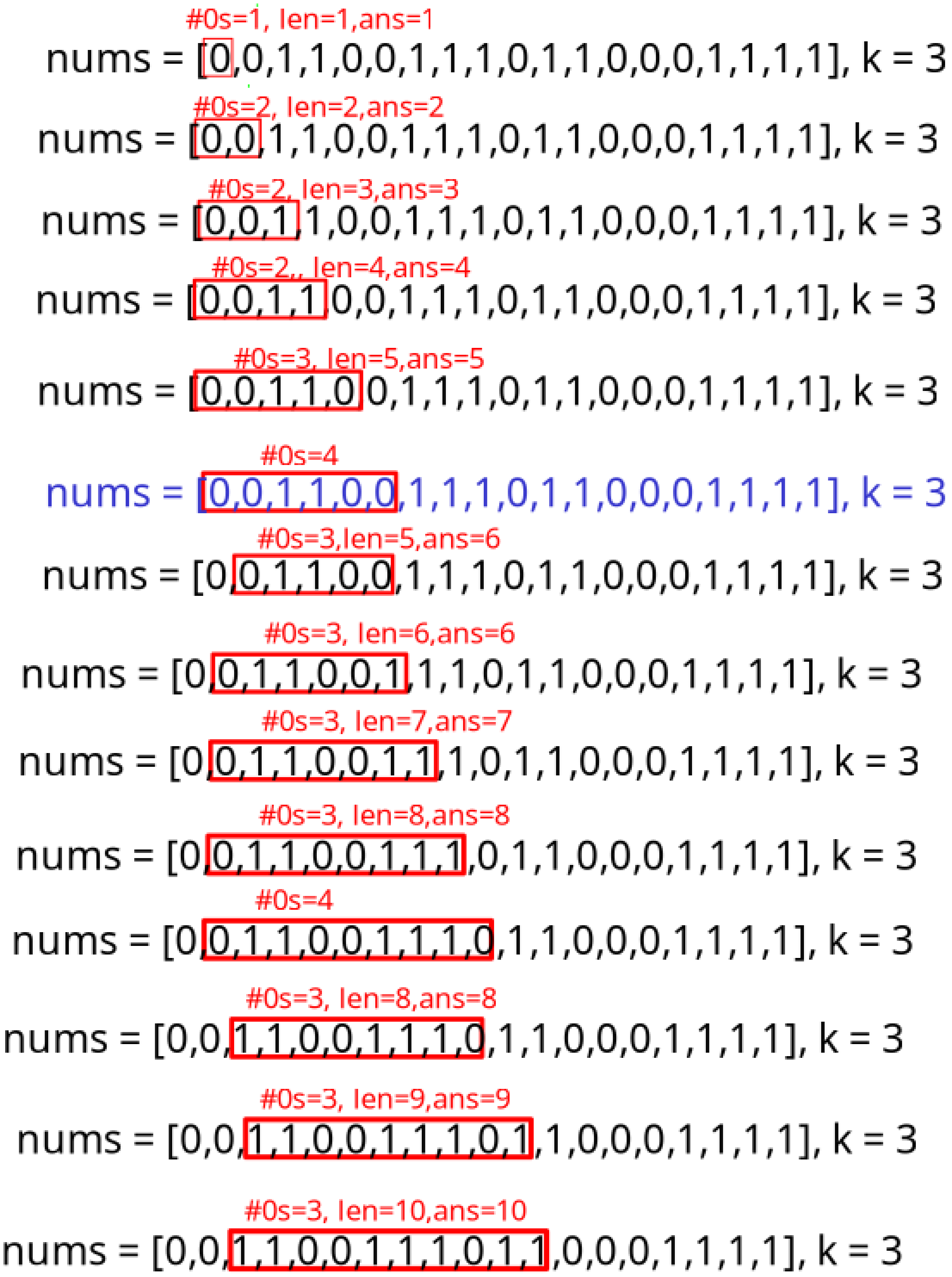
1004. Max Consecutive Ones III

```
/*
   Brute Force: try all possibilities
   Time complexity: $O(n^2)$ (TLE)
   Space complexity: O(1)
*/
class Solution {
public:
   int longestOnes(vector<int>& nums, int k) {
      int n=nums.size();

      int ans=INT_MIN;

      // For each window
      for(int i=0;i<n;++i){
         int j=i; // starting from i

         // Expand it to the right, while counting the numbers of zeros

         int cnt_zero=0; // Initialize the number of zeros to 0
         // While the end of the array is not reached and the number of zeros is
         // not out of the limit k
         // Otherwise, the actual window can not be expanded any more
         while(j<n && cnt_zero<=k){
            cnt_zero+=(nums[j]==0); // Update the number of zeros
            j++; // Expand it to the right
         }

         // If number of zeros is greater than k, means
         // we have an extra 0 in the length of the sequence
         // If number of zeros is less or equal to k, means
         // no xtra 0 in the length of the sequence is len=j-i
         int len=cnt_zero>k?j-i-1:j-i;

         ans=std::max(ans,len); // Maximize the answer
      }

      return ans;
   }
};
```

1004. Max Consecutive Ones III

Dynamic sliding window

Expand the window while number of zeros is less or equal to k, shrink it otherwise.



#0s=1, len=1,ans=1

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=2, len=2,ans=2

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=2, len=3,ans=3

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=2,, len=4,ans=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=5,ans=5

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=5,ans=6

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=6,ans=6

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=7,ans=7

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=8,ans=8

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=8,ans=8

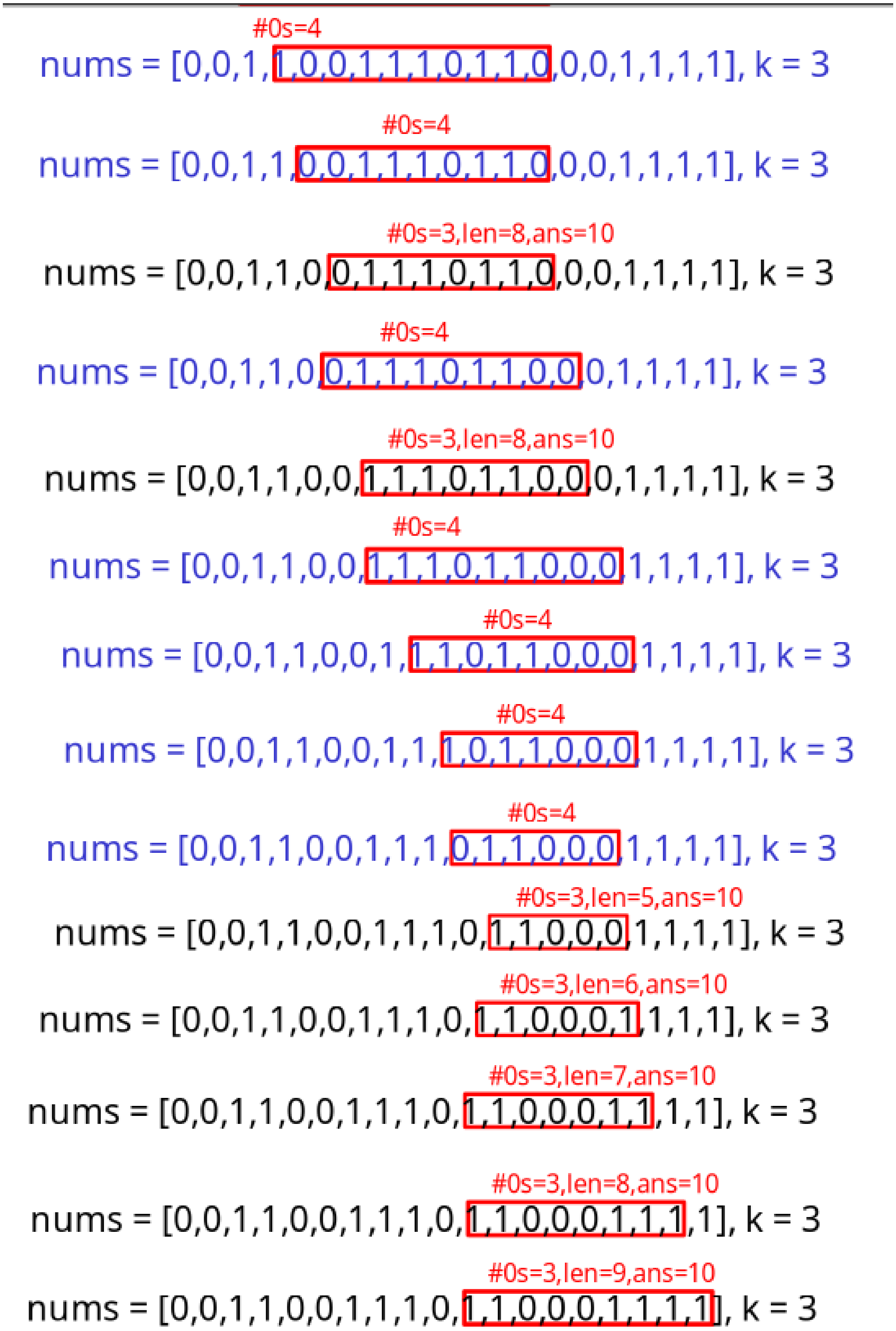
nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=9,ans=9

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3, len=10,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3



#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=8,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=8,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=4

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=5,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=6,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=7,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=8,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

#0s=3,len=9,ans=10

nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3

1004. Max Consecutive Ones III

```
/*
    Dynamic sliding window
    Time complexity: O(n) (AC)
    Space complexity: O(1)
*/
class Solution {
public:
    int longestOnes(vector<int>& nums, int k) {
        int n=nums.size();
        int ans=INT_MIN;
        int cnt_zeros=0;

        // Expand the window to the right
        int l=0;
        for(int r=0;r<n;++r){
            cnt_zeros+=(nums[r]==0); // Add 1, if we encounter a 0

            // While number of zeros is greater than k
            while(cnt_zeros>k){
                cnt_zeros-=(nums[l]==0); // Subtract 1, if we encounter a 0, while ...
                l++; // ... shrinking the window from the left
            }

            ans=std::max(ans,r-l+1); // Maximize the answer
        }
        return ans;
    }
};
```