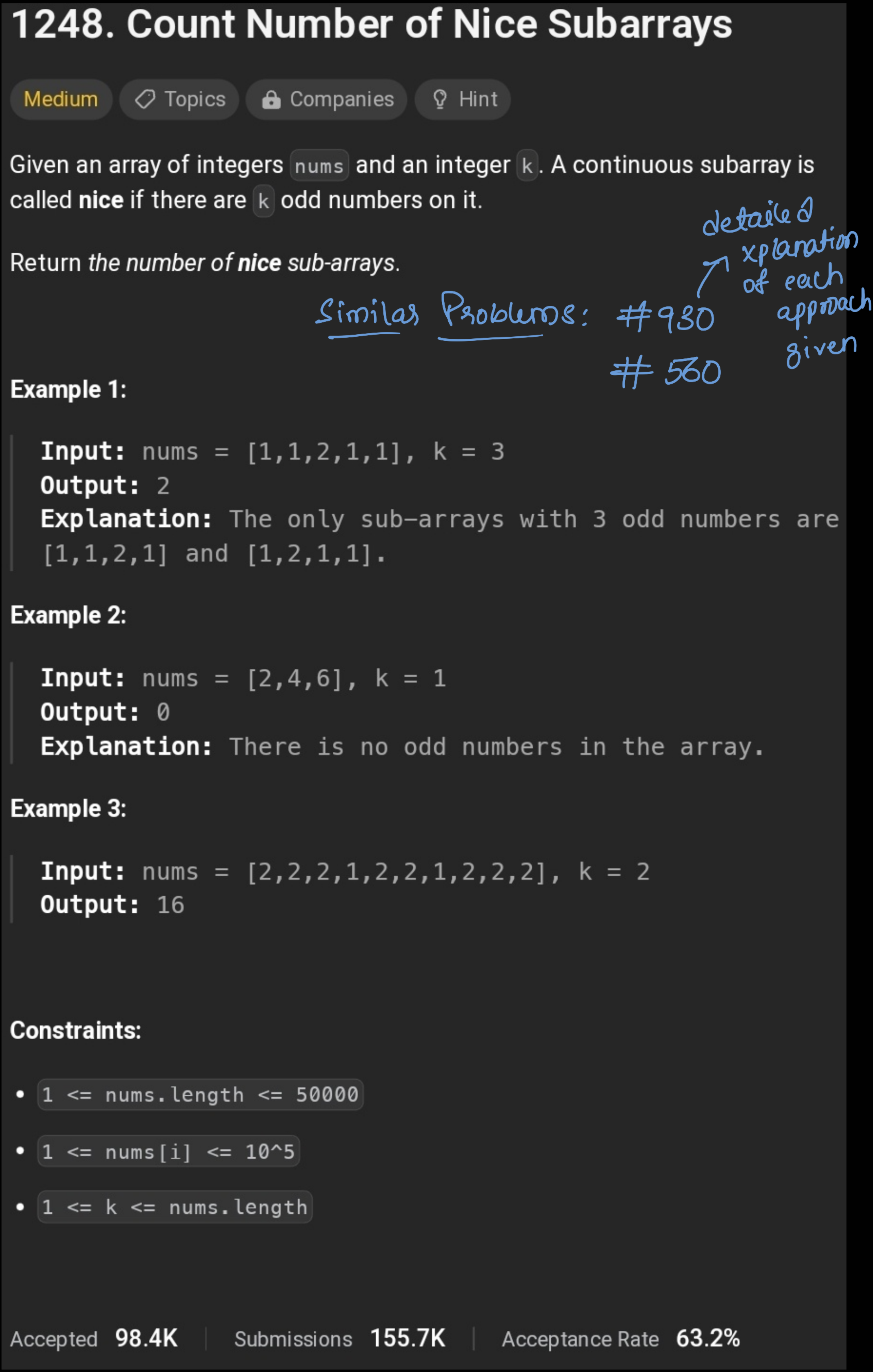

1248. Count Number of Nice Subarrays

Given an array of integers `nums` and an integer `k`. A continuous subarray is called **nice** if there are `k` odd numbers on it.

Return *the number of **nice** sub-arrays*.

Similar Problems: #930 $\nearrow$ detailed xplanation of each approach given

#560

Example 1:

```
Input: nums = [1,1,2,1,1], k = 3
Output: 2
Explanation: The only sub-arrays with 3 odd numbers are
[1,1,2,1] and [1,2,1,1].
```

Example 2:

```
Input: nums = [2,4,6], k = 1
Output: 0
Explanation: There is no odd numbers in the array.
```

Example 3:

```
Input: nums = [2,2,2,1,2,2,1,2,2,2], k = 2
Output: 16
```

Constraints:

- `1 <= nums.length <= 50000`
- `1 <= nums[i] <= 10^5`
- `1 <= k <= nums.length`

Accepted 98.4K | Submissions 155.7K | Acceptance Rate 63.2%

Approach 1: Brute force

$\underline{1 \quad 1 \quad 2 \quad 1} \quad 1$

$T(n): O(n^2)$
$S(n): O(1)$

Approach 2: Prefix Sum + Hashmap

$1 \quad 1 \quad 2 \quad 1 \quad 1 \qquad k=3$

Prefix array: $1 \quad 2 \quad 2 \quad 3 \quad 4$

$\downarrow$

stores the no. of odd numbers upto that index

$0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4$

$4 \rightarrow 4-3=1$ exists

$3 \rightarrow 3-3=0$ exists

So answer is two subarrays

$T(n): O(n)+O(n)$
$S(n): O(n)+O(n)$

we can remove prefix array if

want to
as in #930

Approach 3: Sliding window

no. of Sub arrays that has exactly K odd numbers = no. of Sub arrays that has atmost K odd numbers − no. of sub arrays that has atmost K-1 odd numbers

Algo:

```
int Count( nums , k )
{
        l = 0 , r = 0 , ans = 0 , count = 0

        while( r < n )
        {
              if ( nums [r] is odd )
                      count ++

              if (count > K)
              {
                   while(l<n && count > K )
                   {
                        if (nums [l] is odd)
                          count - -
```

```
            l++
         }
      }

      r++

   }  ans = ans + r - l
 }
int main( )
{

    return Count(nums, K ) -
              Count(nums, K -1)
}
```

$T(n) : O(n) +$
$O(n)$
$S(n) : O(1)$

Takeway:
When we are asked to find subarrays that has exactly constrains we can think of using atmost k - atmost k-1.

note: here exactly refers to **counting** only. This technique cannot be used on Subarrays with Sums.