

## 2958. Length of Longest Subarray w/ at Most K Frequency


Given: integer array 'nums' and integer 'k'

The frequency of a given number in 'nums' is denoted w/ 'x'


An array is good if all numbers have an  $x \leq k$

Return length of longest good subarray of 'nums'.

```
int maxSubarrayLength (std::vector<int> nums, int k) {
```


Input:  [1, 2, 1, 2] k=1

Output: 2

Input:  [1, 3, 3, 3, 4, 5, 6, 7] k=1

Output: 5

if  $k < 1 \Rightarrow 0$  ✗

 1 3 3 3 4 5 6 7

m < int, int> result = 2

3 : 1

4 :

Yay! I was close w/ idea. I was right that we could use a sliding window.

Have two iterators. left and right

- 1) Create a map<int, int> where Key = num in nums and Value = count
- 2) Assign a left and right integer to zero.
- 3) For each value increment map counter using right iterator  
If count > K, increment left, other wise try to update result w/ new len. Repeat until right is at end.

```
int maxSubarrayLength( std::vector<int> nums, int K ) {  
    int result {0};  
    auto left { nums.cbegin() };  
    auto right { nums.cbegin() };  
    std::unordered_map<int, int> m;  
    while ( right < nums.cend() ) {  
        m[*right] += 1;  
        if ( m.at(*right) > K ) {  
            m.at(*left) -= 1;  
            ++left;  
        }  
        else {  
            ++right;  
        }  
        result = std::max( result, static_cast<int>(right - left) );  
    }  
    return result;  
}
```