2419. Longest Subarray With Maximum Bitwise AND

You are given an integer array `nums` of size `n`.

Consider a **non-empty** subarray from `nums` that has the **maximum** possible **bitwise AND**.

- In other words, let `k` be the maximum value of the bitwise AND of **any** subarray of `nums`. Then, only subarrays with a bitwise AND equal to `k` should be considered.

Return *the length of the* ***longest*** *such subarray*.

The bitwise AND of an array is the bitwise AND of all the numbers in it.

A **subarray** is a contiguous sequence of elements within an array.

Example 1:

```
Input: nums = [1,2,3,3,2,2]
Output: 2
Explanation:
The maximum possible bitwise AND of a subarray is 3.
The longest subarray with that value is [3,3], so we return 2.
```

Example 2:

```
Input: nums = [1,2,3,4]
Output: 1
Explanation:
The maximum possible bitwise AND of a subarray is 4.
The longest subarray with that value is [4], so we return 1.
```

Constraints:

- $1 <= \text{nums.length} <= 10^5$
- $1 <= \text{nums[i]} <= 10^6$

2419. Longest Subarray With Maximum Bitwise AND

```
/*
    **Naive approach**
    Time complexity: O(n^2)
    Space complexity: O(1)
*/
class Solution {
    public:
        int longestSubarray(vector<int>& nums) {
            int n=nums.size();
            int max_and=0;
            int ans=1;
            for(int i=0;i<n;++i){
                int cur_max_and=nums[i];
                for(int j=i;j<n;++j){
                    cur_max_and &= nums[j];
                    if(cur_max_and!=0){
                        if(cur_max_and>max_and){
                            max_and=cur_max_and;
                            ans=j-i+1;
                        }
                        else if(cur_max_and==max_and){
                            ans=std::max(ans,j-i+1);
                        }
                    }
                }
            }
            return ans;
        }
};
```

2419. Longest Subarray With Maximum Bitwise AND

```
/*
    Modified Kadane's algorithm-version 1
    Time complexity: O(n+n)=O(n)
    Space complexity: O(1)
*/
class Solution {
    public:
        int longestSubarray(std::vector<int>& nums) {
            int n=nums.size();
            int mx=*std::max_element(nums.begin(),nums.end());
            int tmp_ans=0,ans=0;
            for(auto& e: nums){
                if(e==mx) tmp_ans++;
                else tmp_ans=0;
                ans=std::max(ans,tmp_ans);
            }
            return ans;
        }
};
```

2419. Longest Subarray With Maximum Bitwise AND

```
/*
   Modified Kadane's algorithm-version 2
   Time complexity: O(n)=O(n)
   Space complexity: O(1)
*/
class Solution {
   public:
      int longestSubarray(vector<int>& nums) {
         int n=nums.size();
         int cur_max_and=0,tmp_ans=0,ans=0;
         for(auto& e: nums){
            if(e>cur_max_and){
               cur_max_and=e;
               tmp_ans=1;
               ans=0;

            }
            else if(e==cur_max_and) tmp_ans+=1;
            else {
               tmp_ans=0;
            }
            ans=std::max(ans,tmp_ans);
         }
         return ans;
      }
};
```