Sliding Window

- Priyansh Agarwal

$k = 3$

10	2	9	1	3	4	5	6

12 13

21

size of array = n

window = subarray

3 5

Sliding Window

- Useful for array based problems - subarray
- When to use?
- Optimization Technique
- Use of 2 pointers.
- Super useful for interviews too

sliding windows

fixed size

variable size

Given an array, what is the maximum sum of a subarray of size k

2	3	-1	2	-3	6

$n = 6$

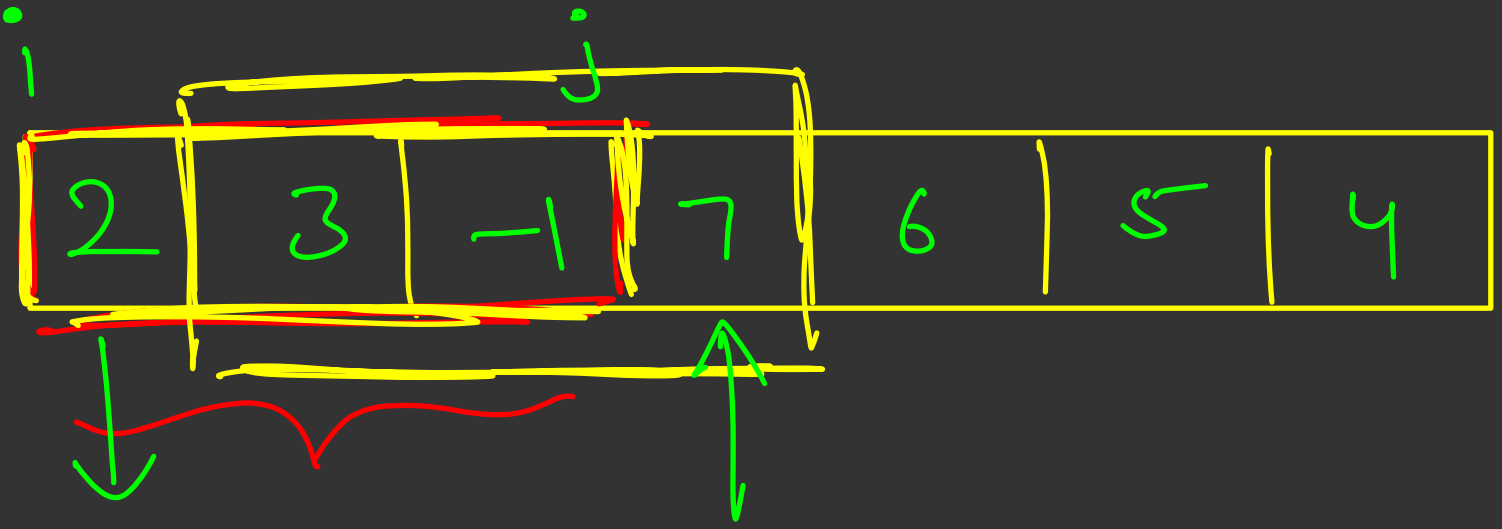
$k = 3$

$n - k + 1$ = no. of subarrays of size k in an array of size n

current window

1. what info to store for the current window to get its answer

2. how to modify this information when moving to the next window



$$\boxed{\text{Sum} = 2 + 3 + -1 = 4}$$

$$\{\text{sum} += \text{arr}[i],\ \text{sum} -= \text{arr}[i-k]\}$$

Given an array, find the first negative number in every subarray of size k

$k = 3$

-1	-2	3	4	-6	0	1	-3

-1 -2 -6 -6 -6 -3

-1	-2	3	4	-6	0	1	-3

$k = 3$

-1

$O(k)$

i-k	0		i				
-1	-2	3	4	-6	0	1	-3

set < int > negative_indices

{0, 1}

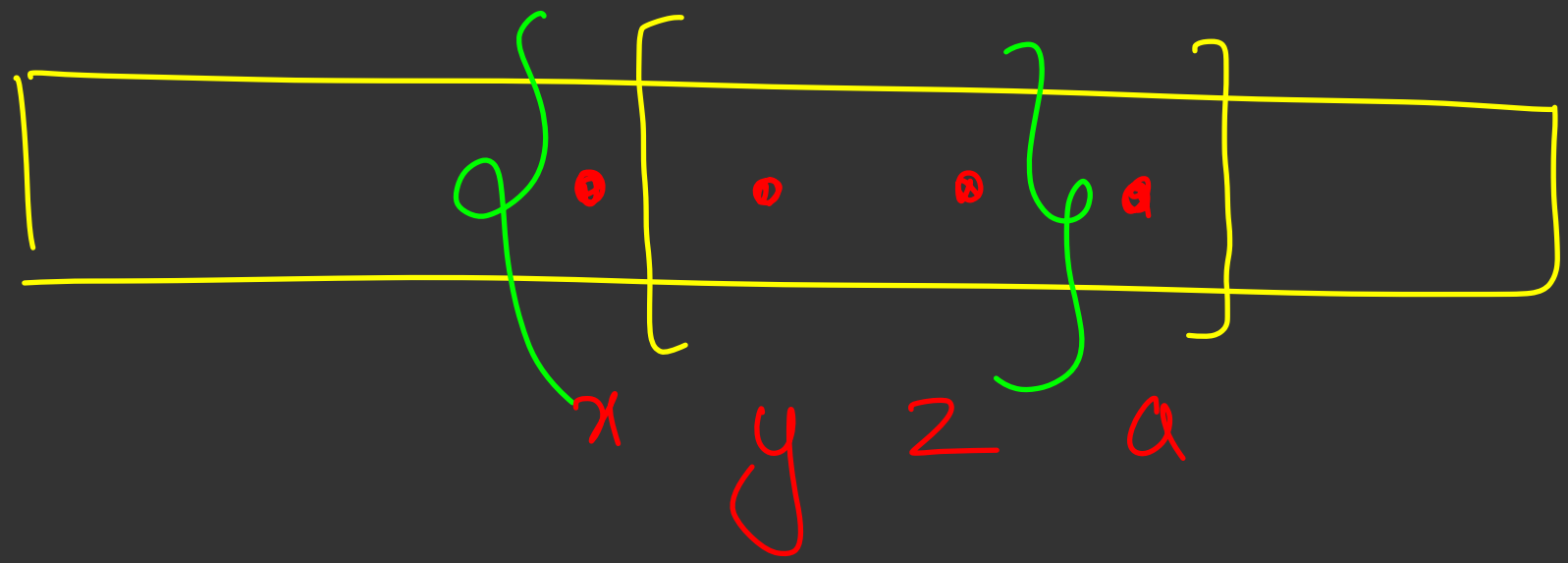
```
if( arr[i] < 0 )
    X.insert(i)
if( arr[i-k] < 0 )
    X.erase(i-k)
```

-1	-2	3	-1	-3	-4	-5

0

vector<int> negatives

{0, 1}

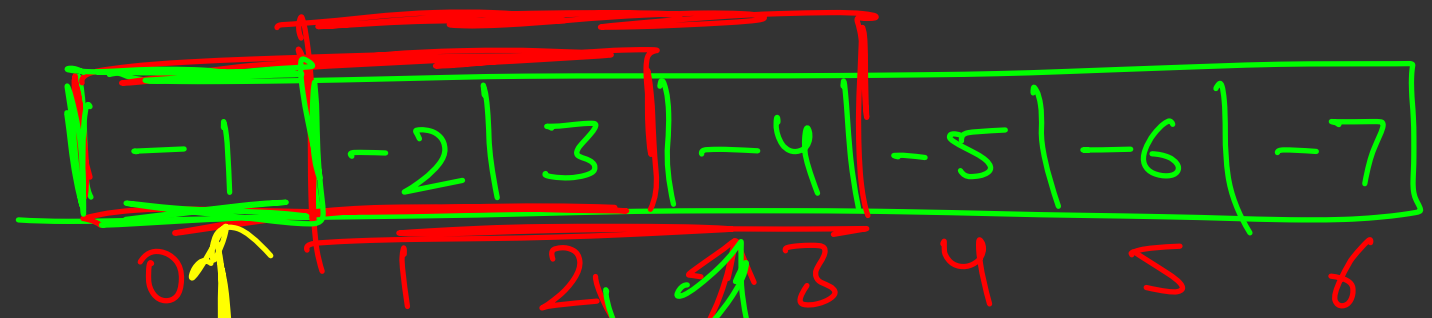


$\leftarrow [x, y, z] \leftarrow a$

vector<int> ng

$O(n)$

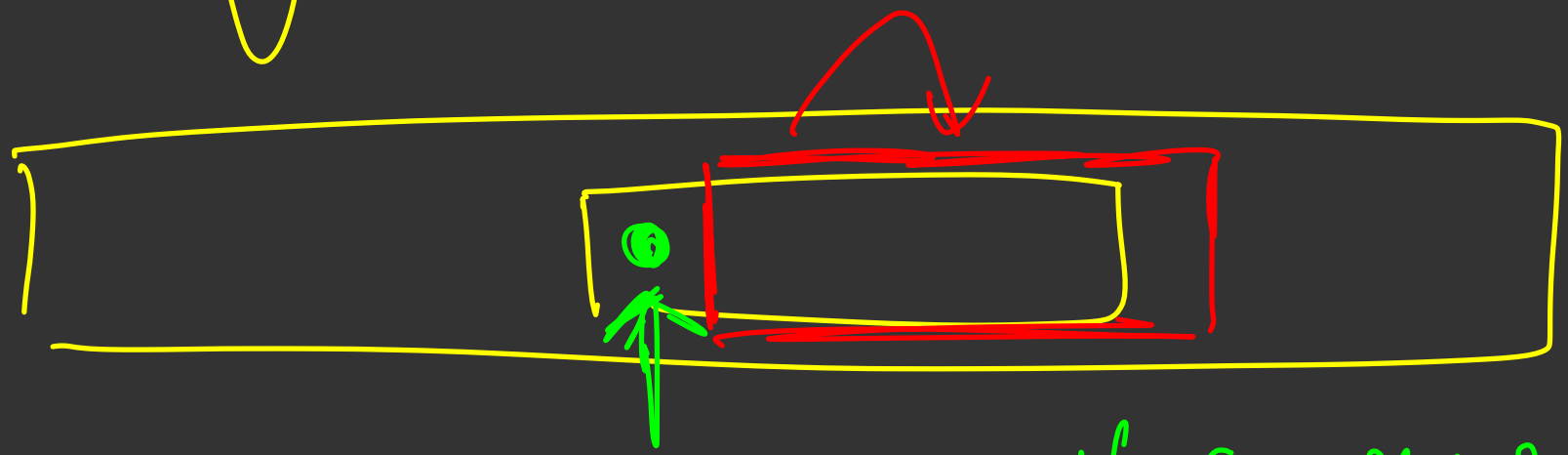
$O(1)$



queue — {0, 1, 3}

i

qu.pop()



if(arr[i] < 0)

qu.push(i)

O(1)

if(arr[i-k] < 0)

qu.pop();

O(1)

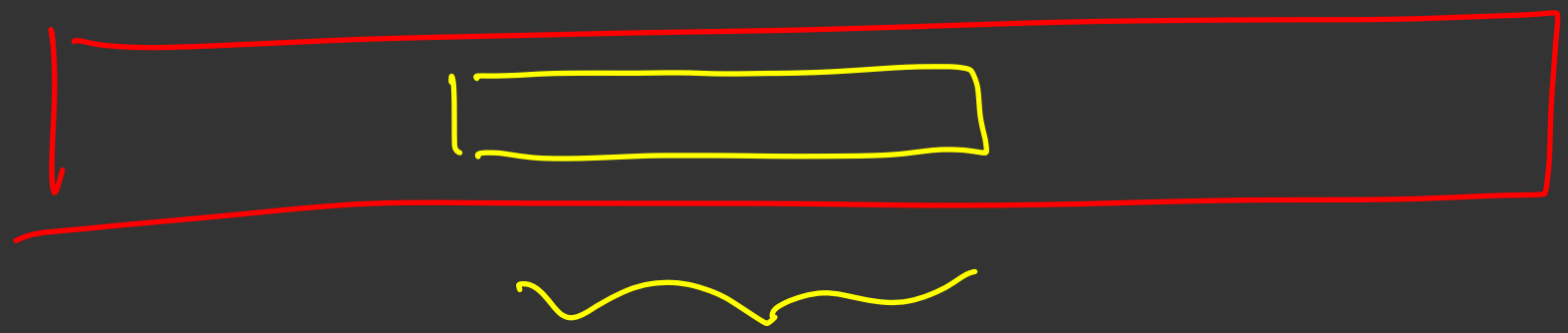
Given an array, find the median of each subarray of size k

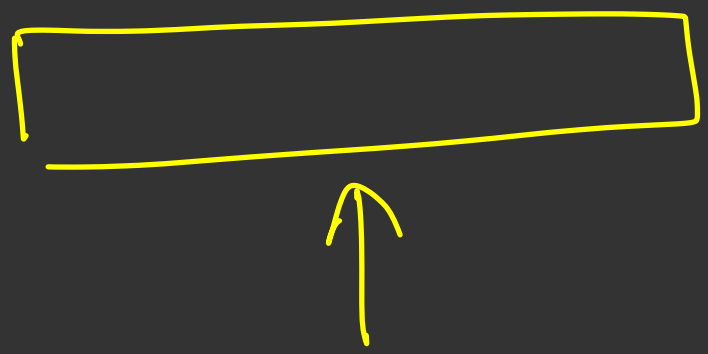
$k = 5$

k is odd

1	2	3	4	5	2	1	4

3 3 3 4





vector

$O(k \log k)$

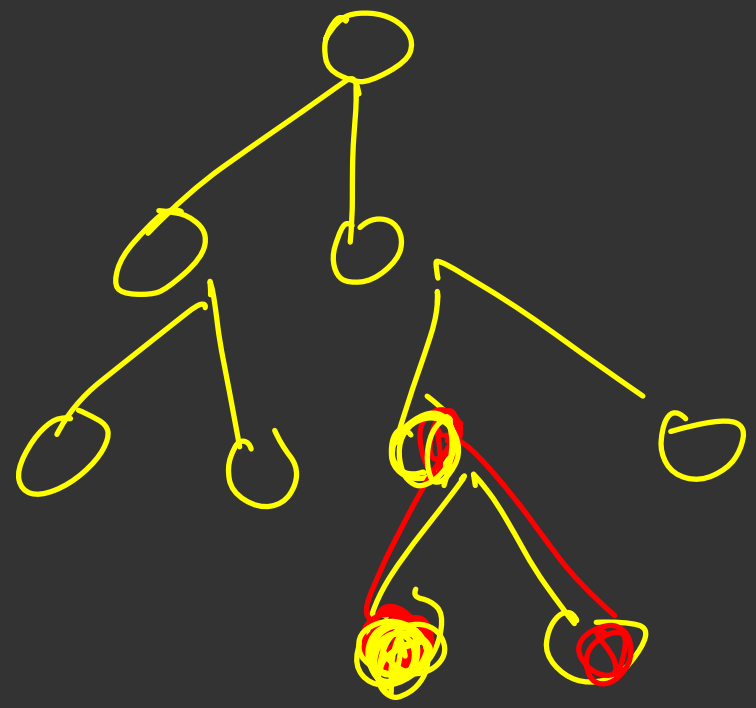
| 0 | 1 | 2 | 3 | 4 | 5 |

multiset

$O(\log x)$ ← insert, remove

multiset.begin() + (multi-set.size()/2)

multiset

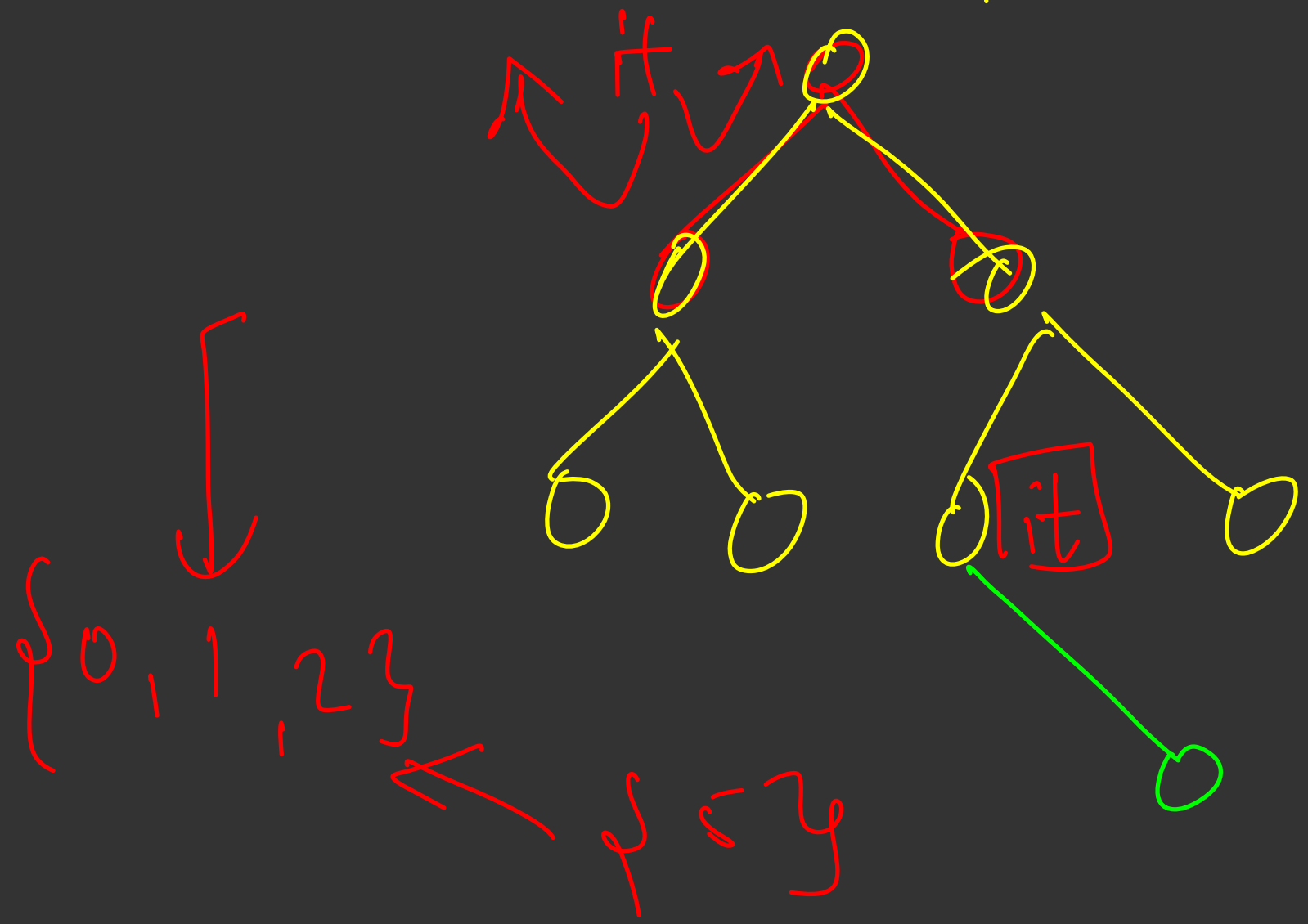


$O(\log n)$

ind ++

5 places

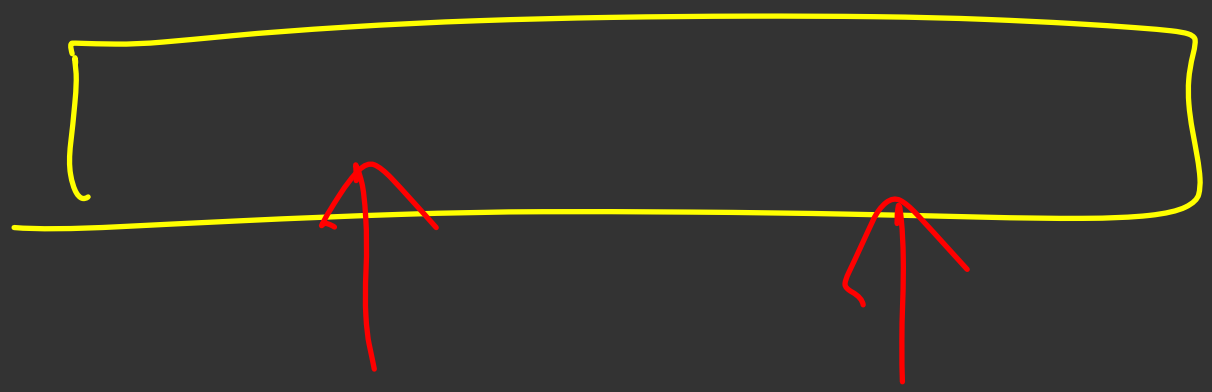
0	1	2	4	6	9	10



A datastructure that can store all elements in sorted order

And provide the

nth element quickly



policy based data structure

ordered set in C++

$\boxed{\text{set}}$ but it provides two additional functionalities

1. find the kth element in $O(\log n)$
2. find index of an element in $O(\log n)$

pdds A

1	2	3	9	10

3rd element

$O(\log n)$

4

$O(\log n)$

1	2	2	2	3

2 3 4

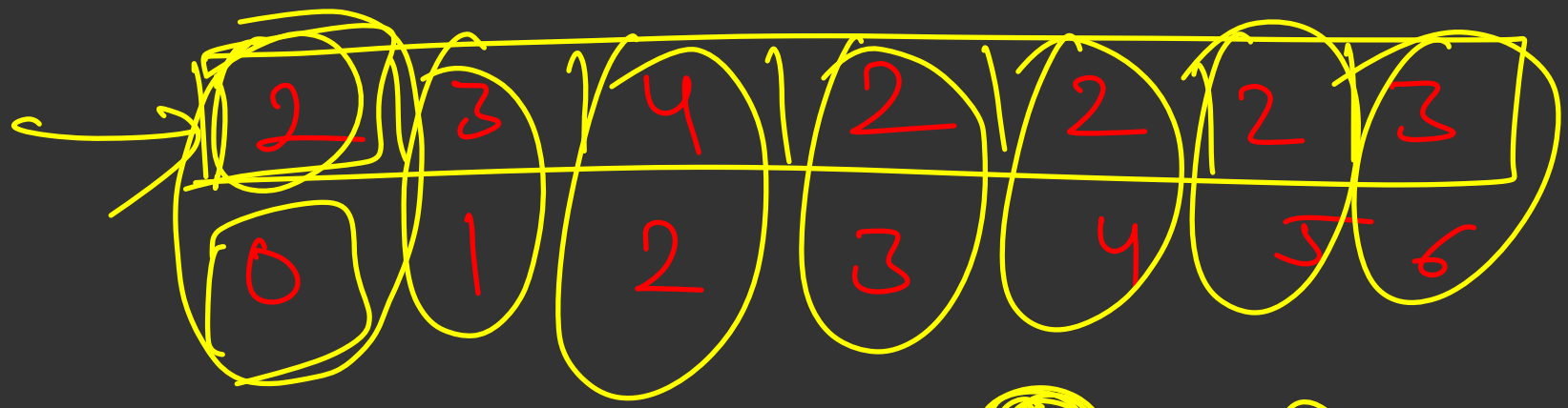
f dds A → set

does not contain duplicate elements

n 0 1 2 3

*A.find_by_order(k/2)

Set<int> s



1 2

Set < pair <int, int>> s

Given an array, find the maximum number in each subarray of size k

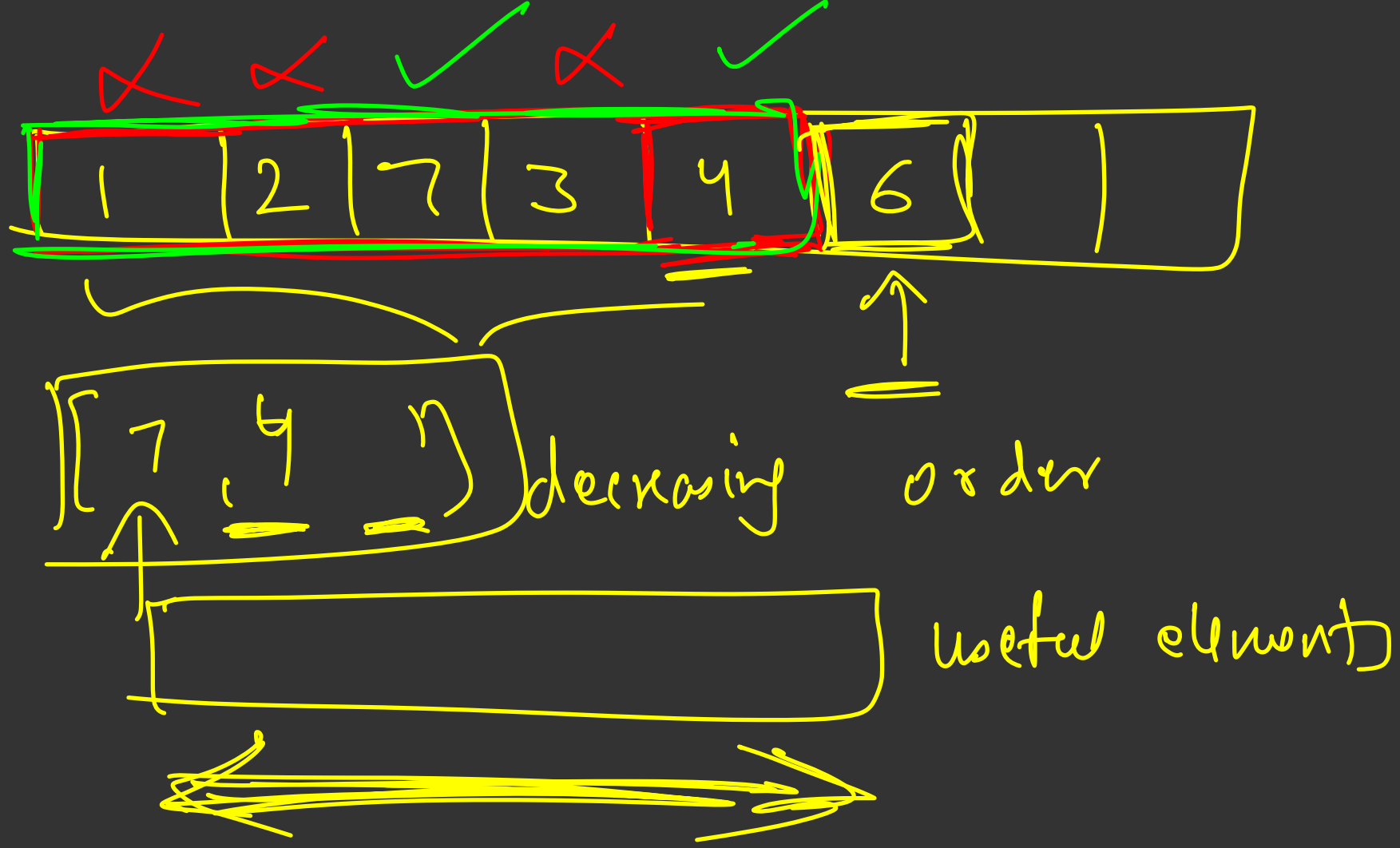
$\boxed{O(n) \text{ time}}$

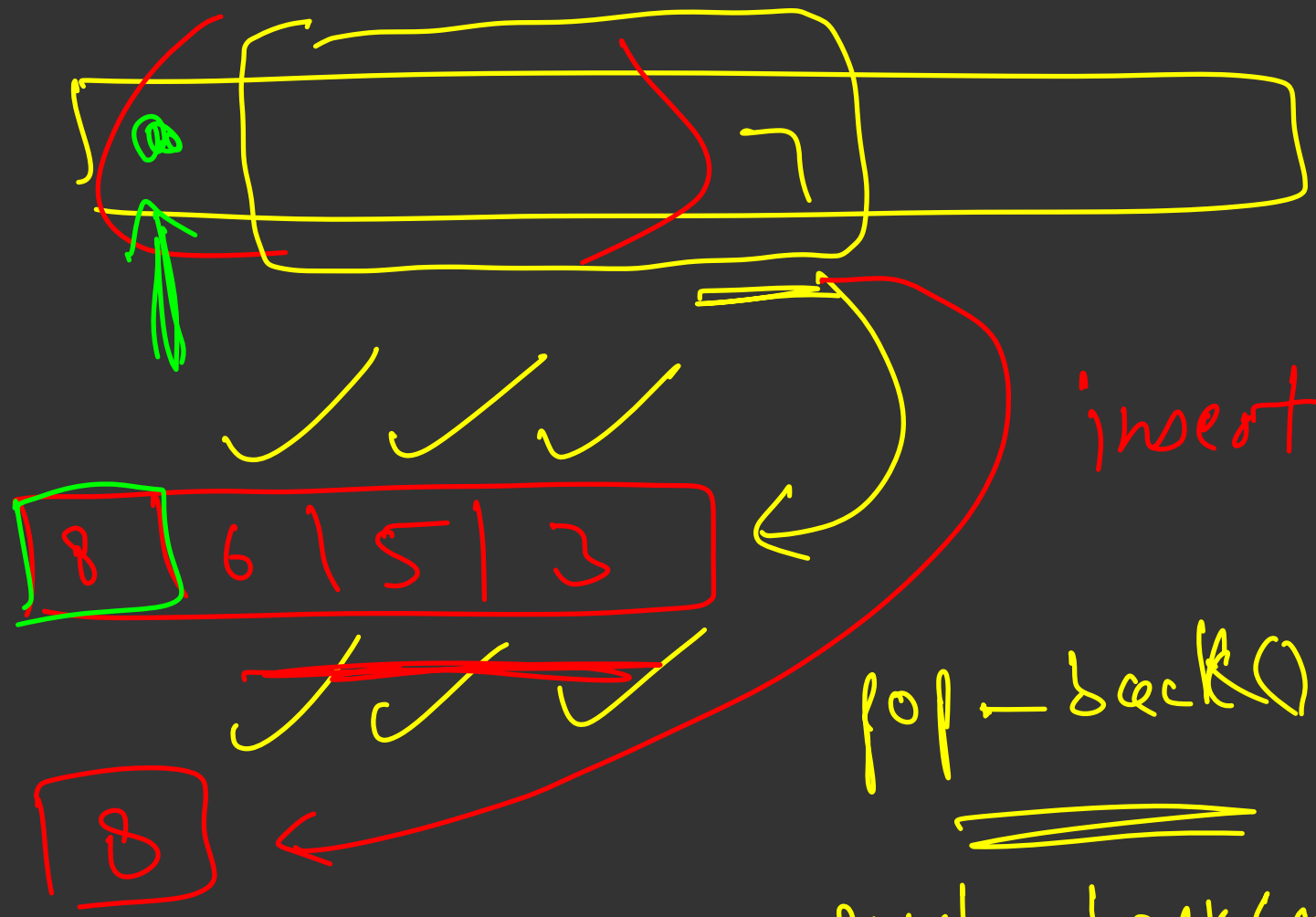
2	3	2	1	0	-1	2

$k = 3$

2	3	4	5	4	2	1	0	3		

All elements to the left of
max are useless


1
2
7
3
4
6
7 4 1
decreasing order
useful elements



insertion

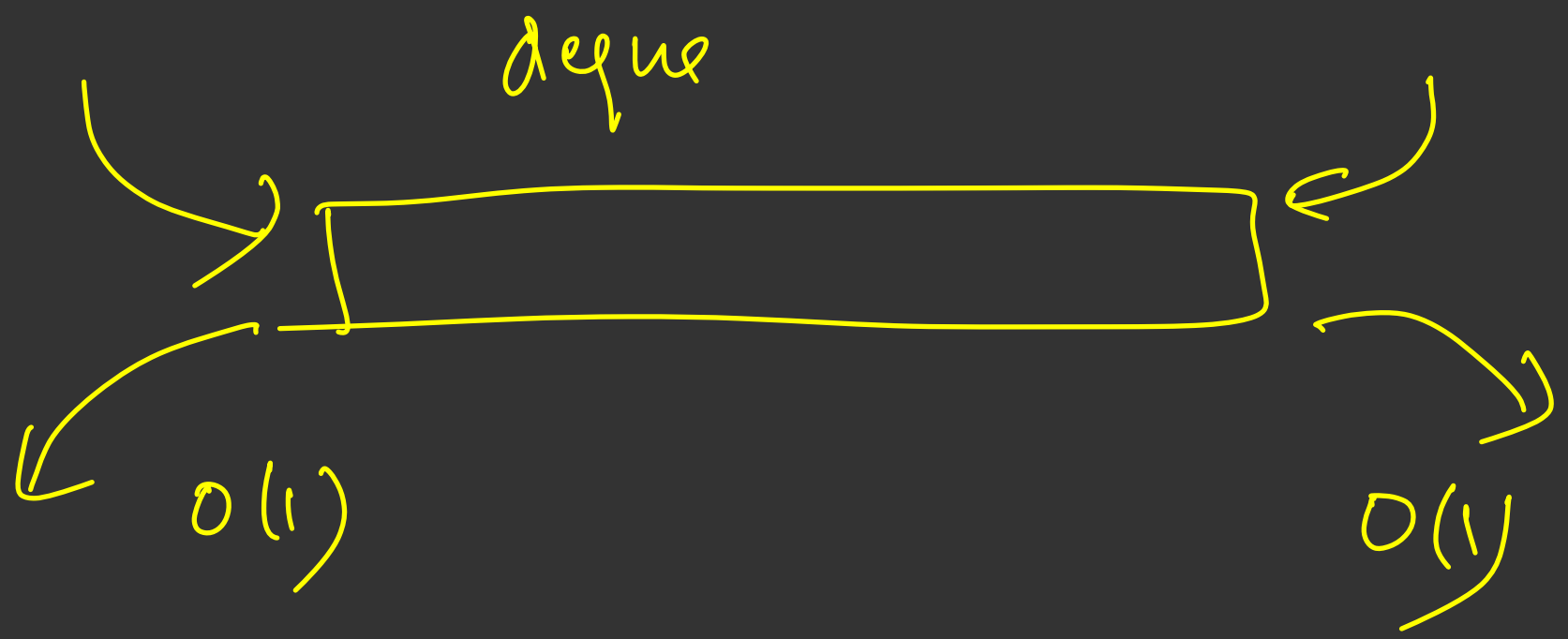
pop_back();

push_back(arr[right]);

{4, 3, 2}

pop_back() ✗ | push_back() O(1)

pop_front() ✗


deque
O(1)
O(1)

Solution:

- Sliding window
- Use of deque

```
vector<int> maxSlidingWindow(vector<int>& nums, int k) {
    deque<int> d;
    vector<int> ret;
    for(int i = 0; i < k; i++){
        while(!d.empty() && nums[i] <= nums[d.back()]){
            d.pop_back();
        }
        d.push_back(i);
    }
    for(int i = k; i < nums.size(); i++){
        ret.push_back(nums[d.front()]);
        if(!d.empty() && d.front() <= i-k){
            d.pop_front();
        }
        while(!d.empty() && nums[i] >= nums[d.back()]){
            d.pop_back();
        }
        d.push_back(i);
    }
    ret.push_back(nums[d.front()]);
    return ret;
}
```

$O(1)$

$\Sigma = O(n)$

deque ①

element

①

$O(n)$

element

Bonus Problem: Link

Bonus Problem: Link