Day 15

26 November 2022 11:08

Identification

① Array | string ✓

window size (k)

subarray | substring ✓

① window size given

② window size find

Given an array of integers Arr of size **N** and a number **K**. Return the maximum sum of a subarray of size K.

Example 1:

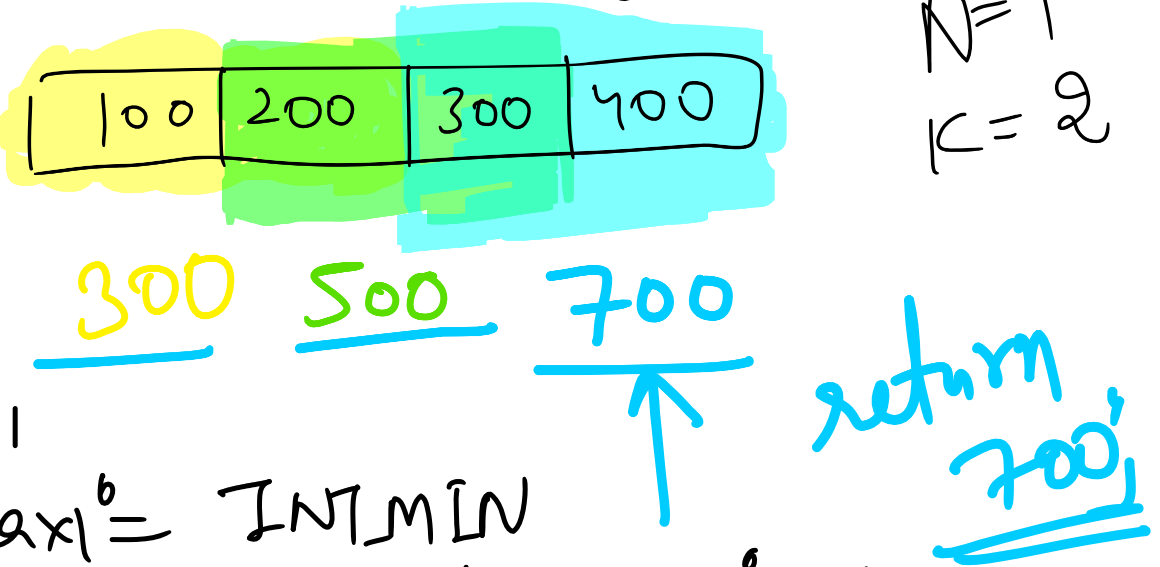
Input:
N = 4, K = 2
Arr = [100, 200, 300, 400]
Output:
700
Explanation:
$Arr_3 + Arr_4 = 700$,
which is maximum.

① Array ✓

Arr3 + Arr4 =700,
which is maximum.

1. Array ✓
2. SubArray ✓
3. window size k given ✓

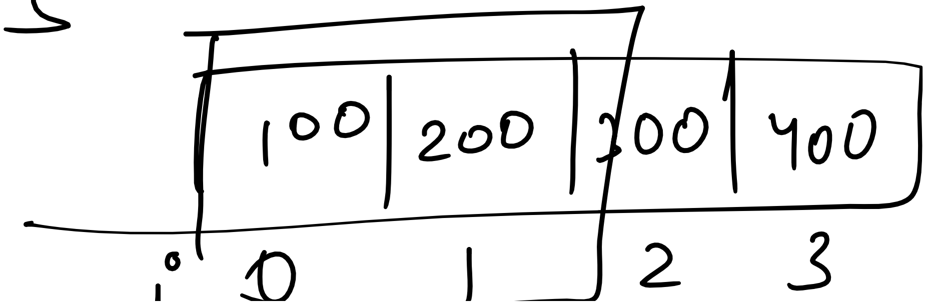
N=4
K=2

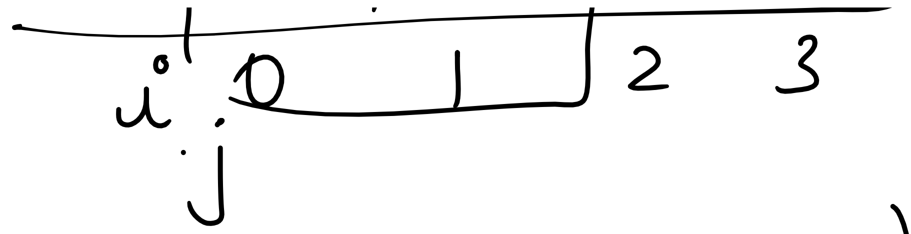


```
0       1
maxi = INT_MIN
for(int i=0; i<n; i++){
    sum=0
    for(int j=i; j<i+2; j++){
        sum+=arr[j]
    }
    maxi = max(maxi, sum);
}
```

K

N





T.C – $O(K \times N)$

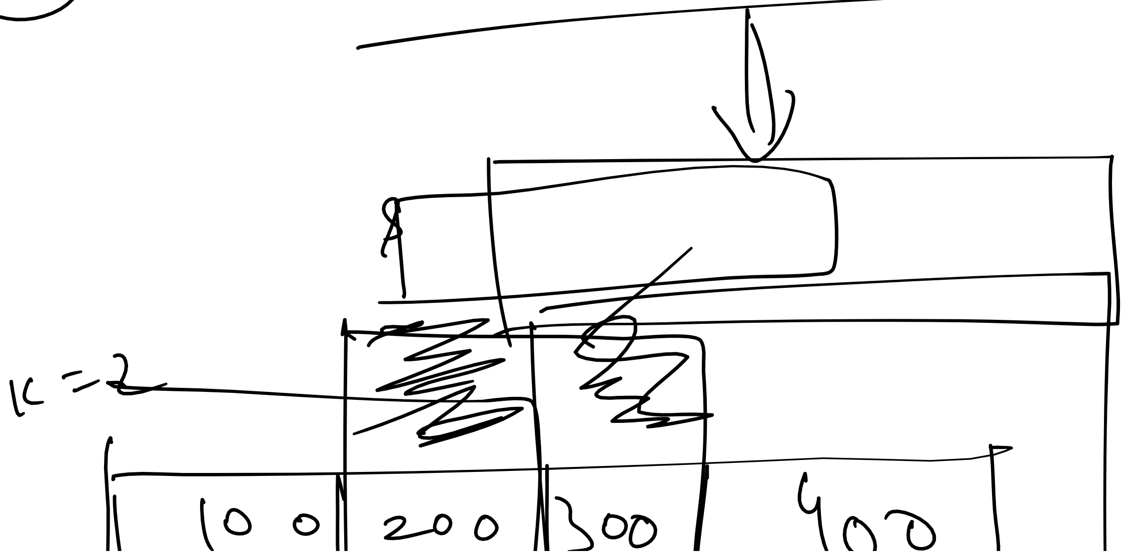
T.C – $O(N^2)$

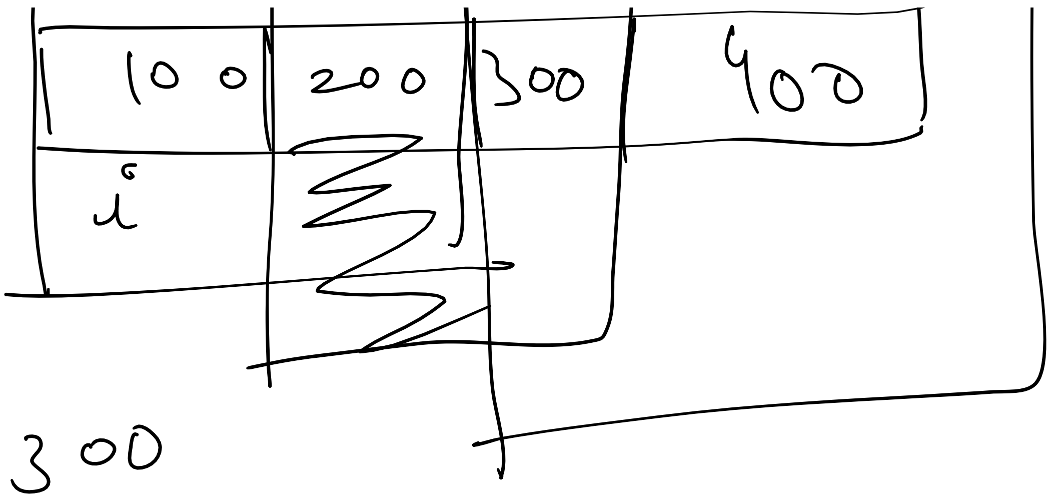
Sliding Window

1st point → window_start

2nd pointer → window_end

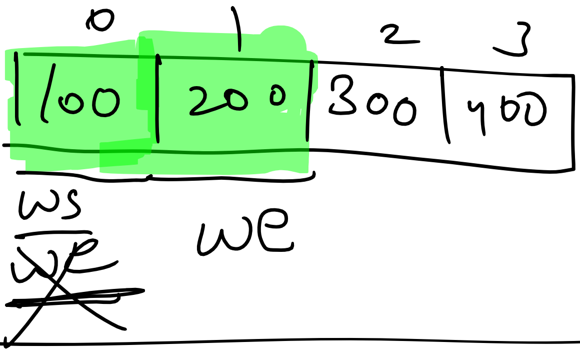
$\boxed{K}$ ⟹ K size hit हो जाए





300

ws = 0
we = 0



k = 2

maxi

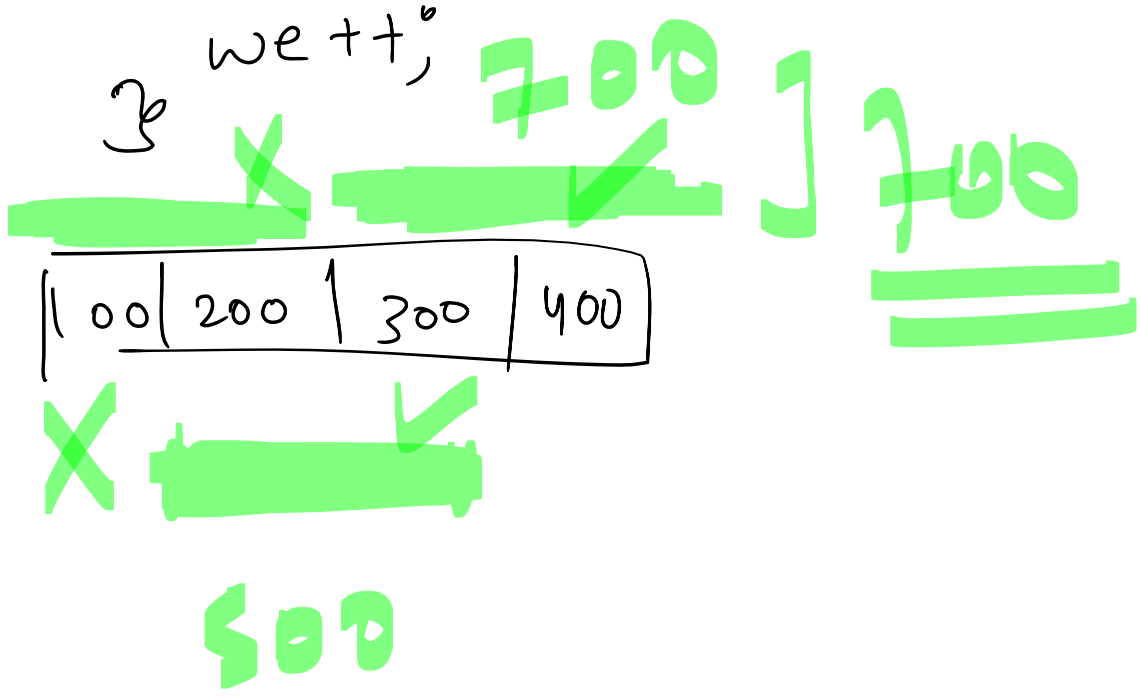
$$\boxed{\begin{aligned} wsize &= we - ws + 1 \\ & \quad 0 - 0 + 1 = \boxed{1} \end{aligned}}$$

$$\boxed{k = 2} \qquad 1 - 0 + 1 = \boxed{2}$$

maxi = INT_MIN

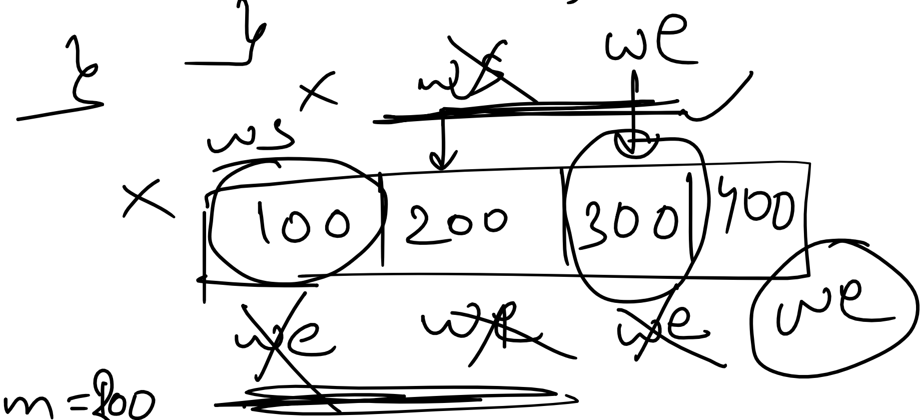
if (we − ws + 1 == k) {

```
if (we-ws+1 == k) {
    maxi = max(maxi, sum);
     ws++;
     we++;
}
```



```
we=0
ws=0     sum=0;    maxi=INT_MIN
                                   Ek element
while( we < n){
  sum += arr[we];          // Incluan
    if(we-ws+1 < k){
         we++;
         continue;
    }                        //sliding
    else if(we-ws+1 == k){
```

```
maxi = max(maxi, sum);
sum = sum - arr[ws];
ws++;
we++;
}
}
```



k=2

maxi = ~~300~~ ~~500~~ 700

sum = 300

sum = 200 + 300 = 500 - 200

= 300 + 400 = 700

T.C - O(N)

S.C - O(1)

Due to the rise of covid-19 cases in India, this year BCCI decided to organize knock-out matches in IPL rather than a league.

Today is matchday 2 and it is between the most loved team Chennai Super Kings and the most underrated team - Punjab Kings. Stephen Fleming, the head coach of CSK, analyzing the batting stats of Punjab. He has stats of runs scored by all N players in the previous season and he wants to find the maximum score for each and every contiguous sub-list of size K to strategize for the game.

subarray of size K

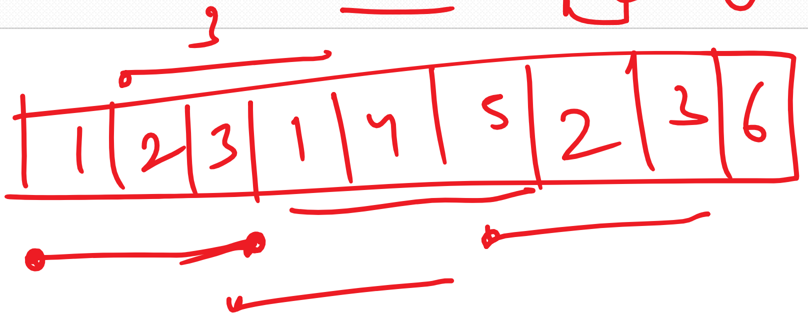
Example 1:

Input:
N = 9, K = 3
arr[] = 1 2 3 1 4 5 2 3 6

array

Output:
3 3 4 5 5 5 6

Explanation:
1st contiguous subarray = {1 2 3} Max = 3
2nd contiguous subarray = {2 3 1} Max = 3
3rd contiguous subarray = {3 1 4} Max = 4
4th contiguous subarray = {1 4 5} Max = 5
5th contiguous subarray = {4 5 2} Max = 5
6th contiguous subarray = {5 2 3} Max = 5
7th contiguous subarray = {2 3 6} Max = 6

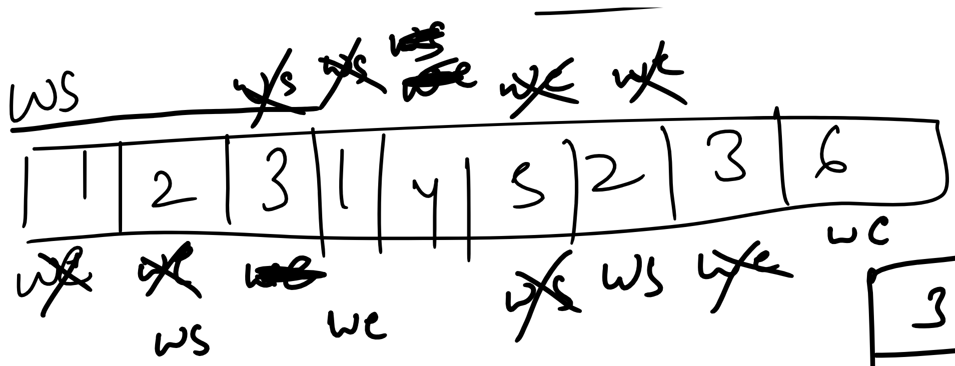


K=3

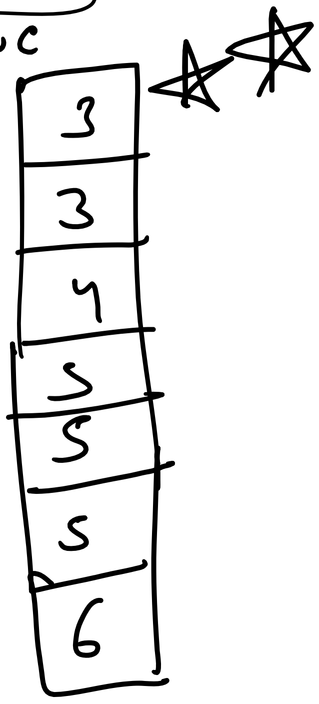
3 4

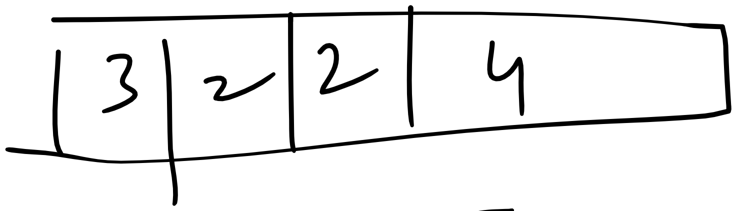
Sliding window of fixed size

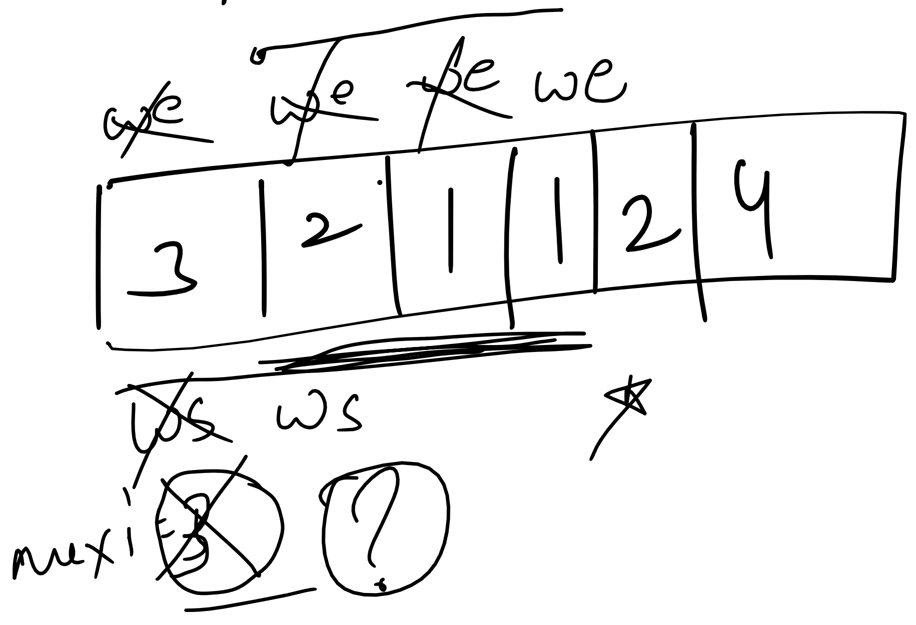
ws [illegible]

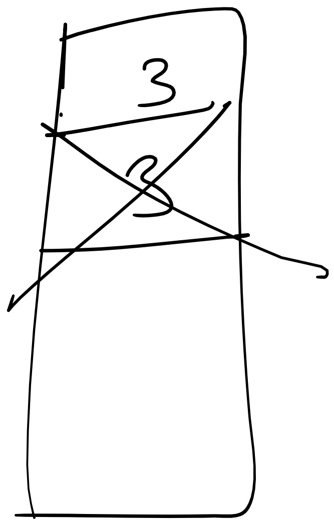

ws
~~ws~~ ~~ws~~ ~~ws~~ ~~we~~ ~~we~~ ~~we~~
1 2 3 1 4 5 2 3 6
~~we~~ ~~we~~ ~~we~~ ~~ws~~ ws ~~we~~ we
ws we

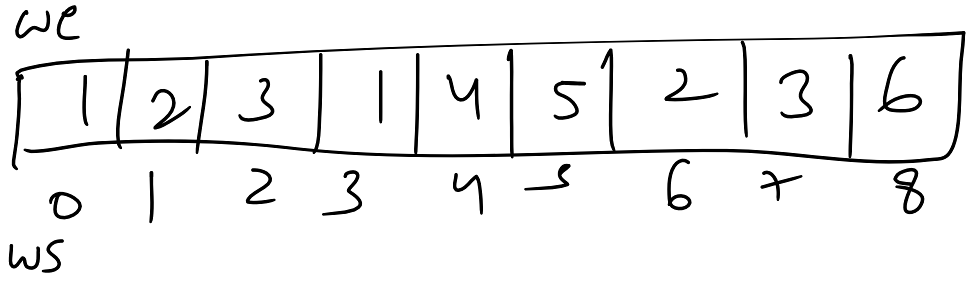
maxi = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

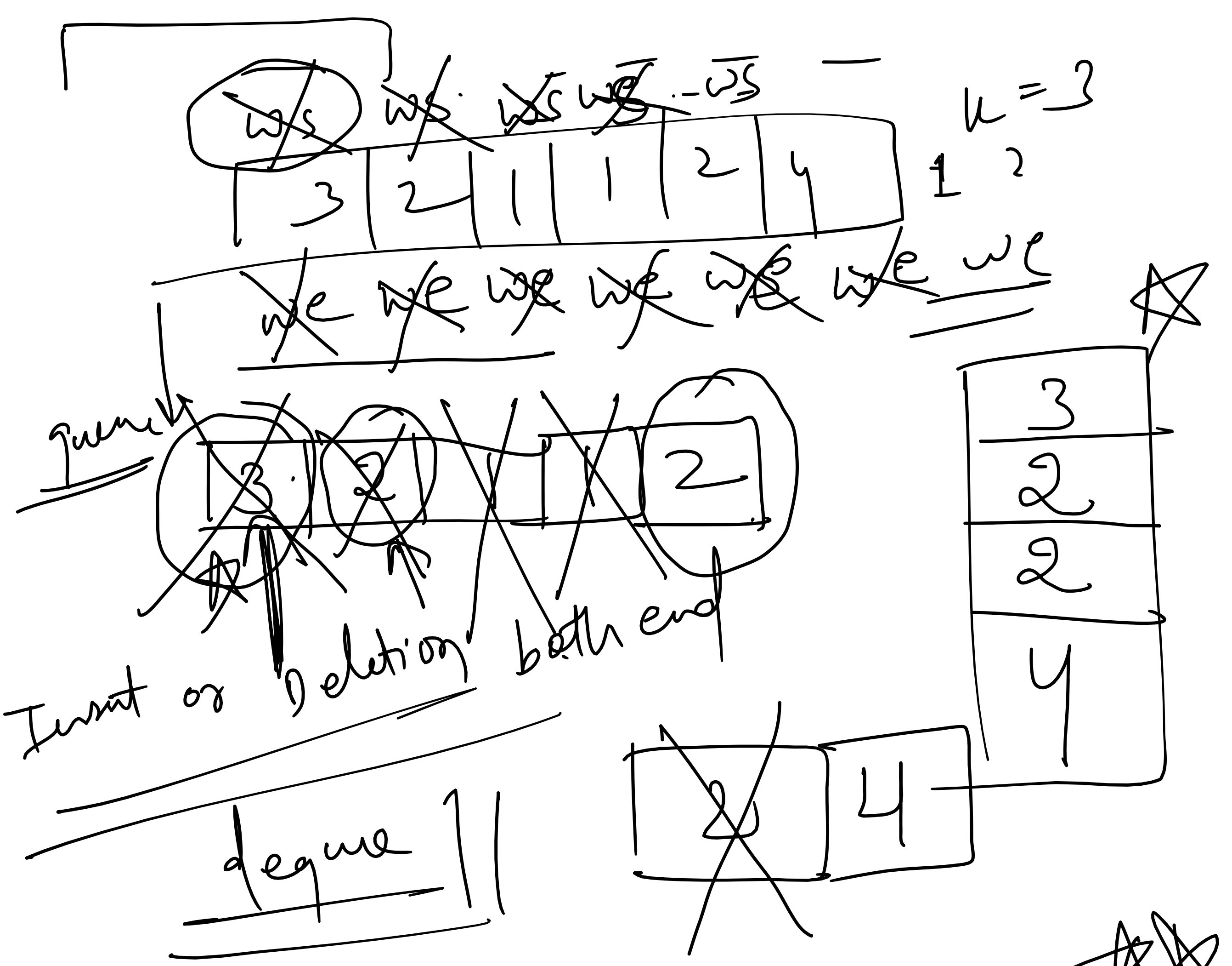

3
3
4
5
5
5
6


3 2 2 4


~~we~~ ~~we~~ ~~we~~ we
3 2 1 1 2 4
~~ws~~ ws
maxi
3
?

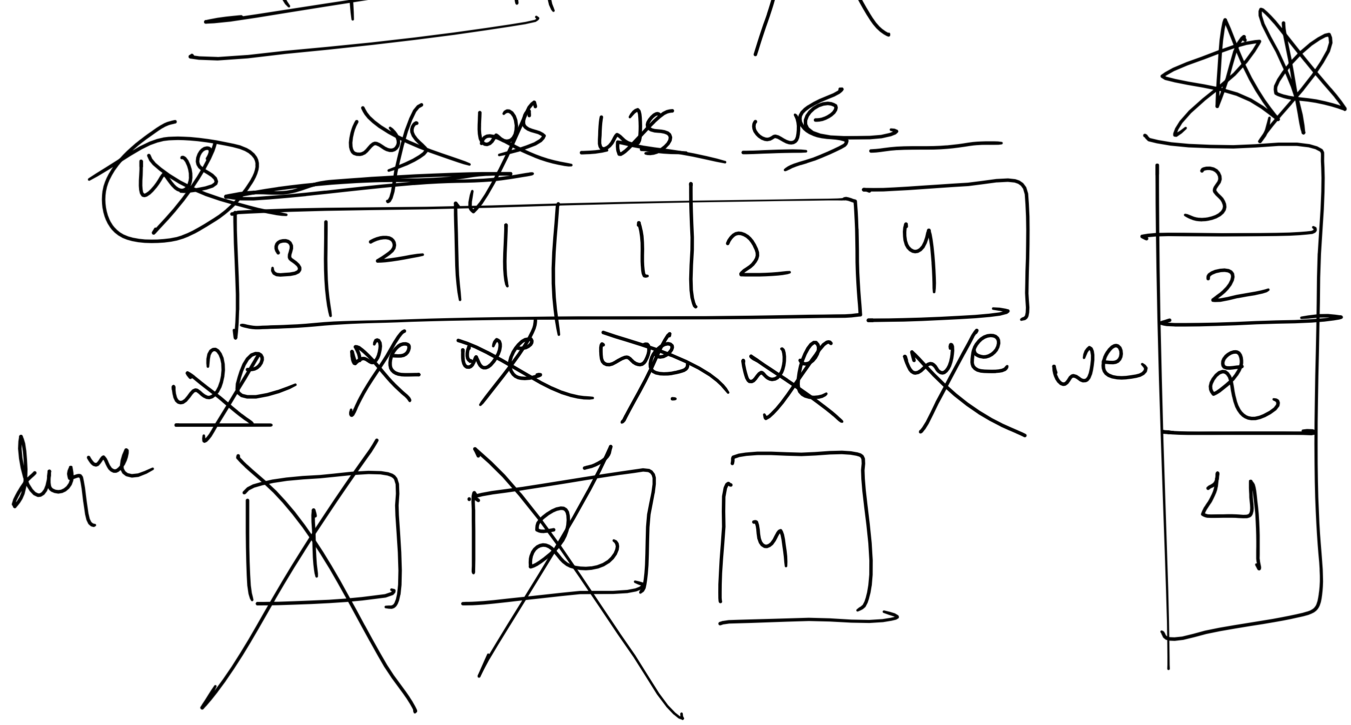

3
~~3~~


we
1 2 3 1 4 5 2 3 6
0 1 2 3 4 5 6 7 8
ws



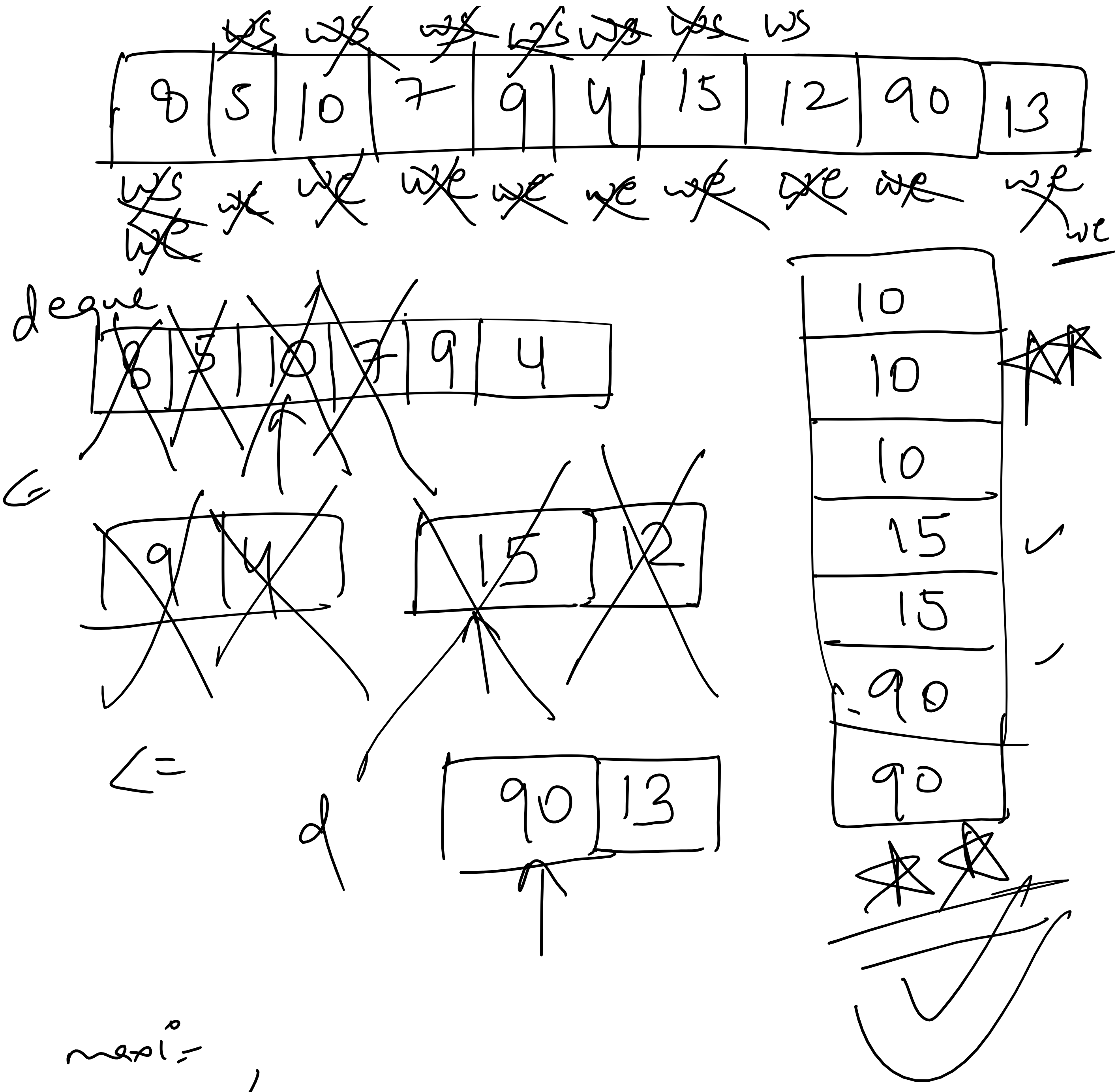
Insert or Deletion both end

deque

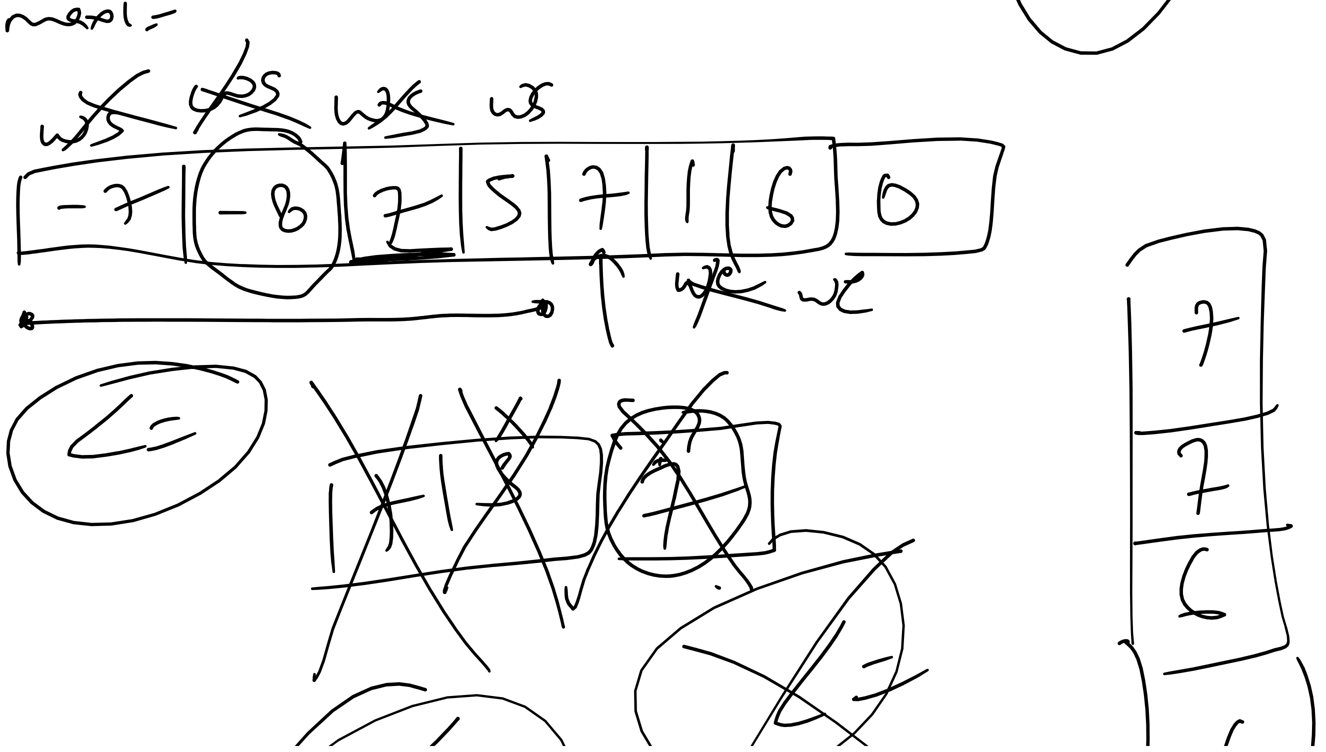


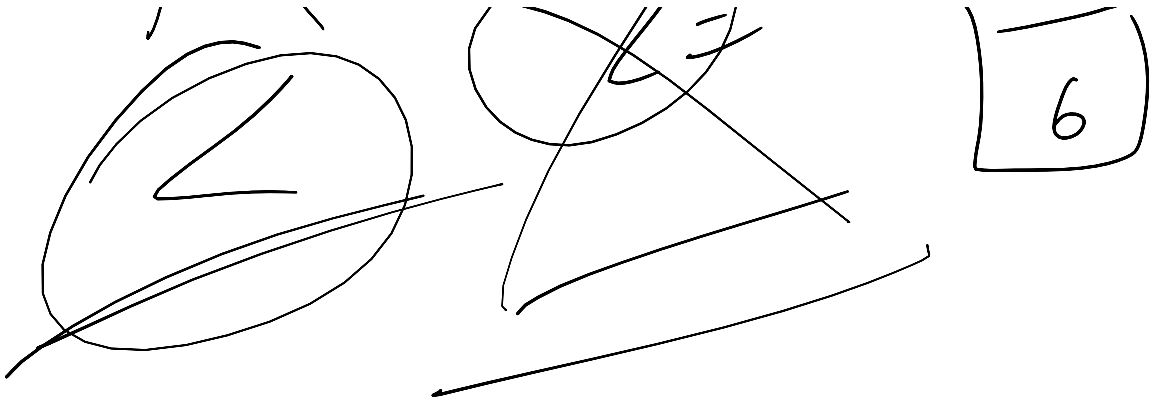
C= deque remove

K=4 N=10


~~ws~~ ~~ws~~ ~~ws~~ ~~ws~~ ~~ws~~ ~~ws~~ ws
8 5 10 7 9 4 15 12 90 13
~~ws~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ ~~we~~ we
deque
8 5 10 7 9 4
<=
9 4
15 12
<=
d
90 13
10
10
10
15
15
90
90

maxi:-


~~ws~~ ~~ws~~ ~~ws~~ ws
-7 -8 7 5 7 1 6 0
~~we~~ we
<=
1 13
7
7
7
6
<=



```
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        deque<int> d;
        vector<int> ans_array;
        int ws = 0, we = 0;
        while(we < nums.size()){
            if(d.size() == 0){
                d.push_back(nums[we]);
            }else{
                while(d.size() > 0 && d.back() < nums[we]){
                    d.pop_back();
                }
                d.push_back(nums[we]);
            }
            if(we - ws + 1 < k){
                we++;
            }else if(we - ws + 1 == k){
                // sliding
                ans_array.push_back(d.front());
                if(d.front() == nums[ws]){
                    d.pop_front();
                }
                ws++;
                we++;
            }
        }
        return ans_array;
    }
};
```

TC - O(N)

S. C — $O(k)$