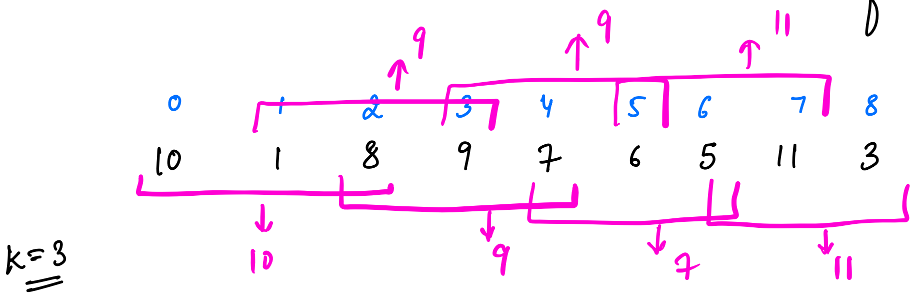
# • Sliding Window Maximum

given an integer array. Find max element ∀ subarrays of size k.



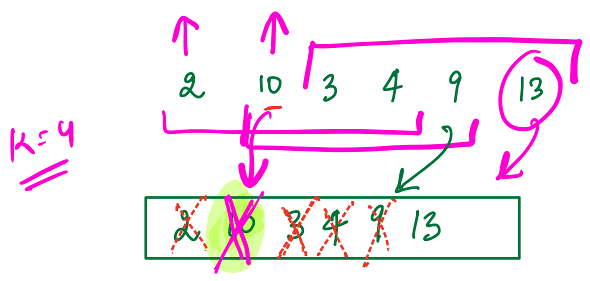
B.F:- 1) check for all windows $\overset{\boxed{k}}{n-k+1}$ — n ≈ O(n)

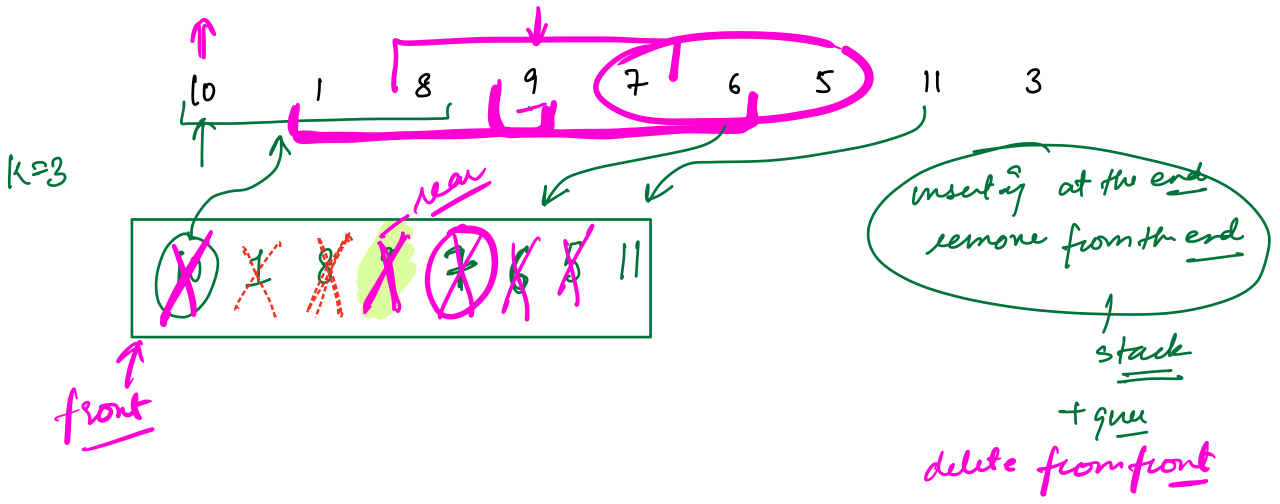
↓

getting the max by iterating

O(n*k)

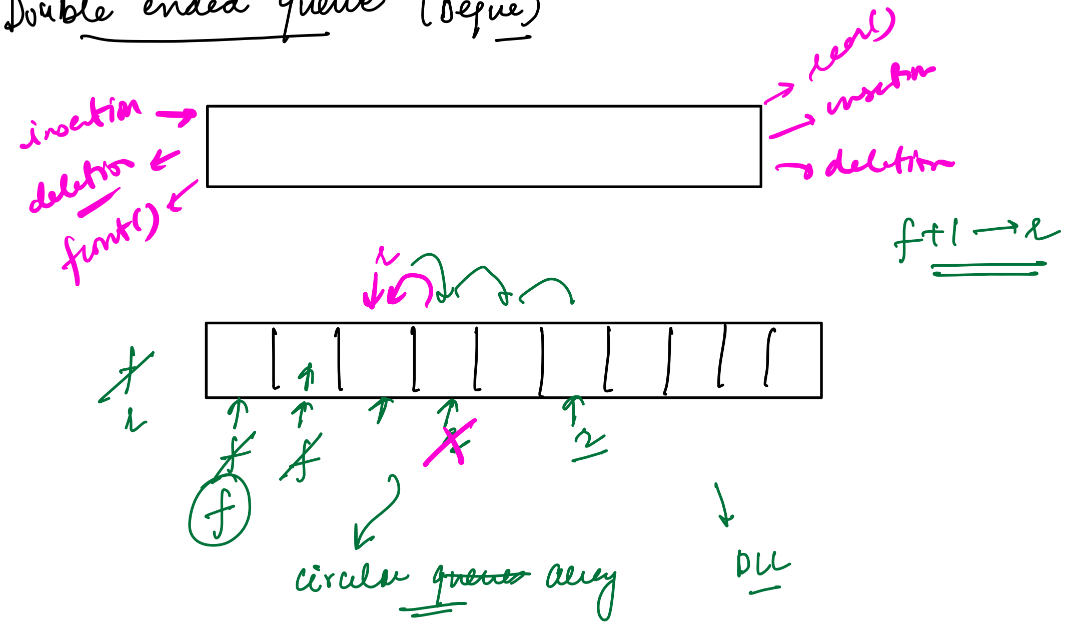
2) treemap/ordered.map ≈ O(N log k)

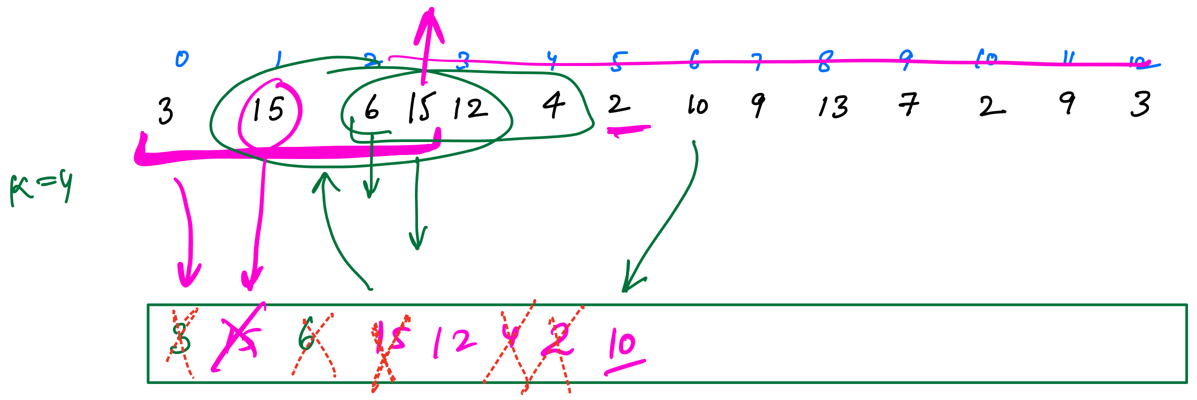




# insert at rear
# delete from rear
# access front
# access rear
# delete from front

## Double ended queue (Deque)



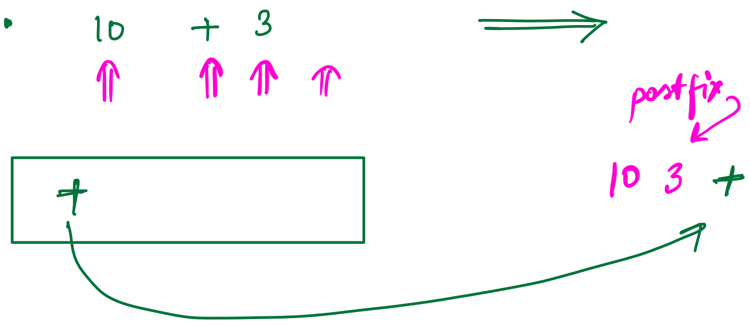


```
deque   dq;

for(  i=0  ⟶  k)
{
        while( dq.size()>0 && dq.rear() < arr[i])
                dq.pop-rear();

        dq.push-rear( arr[i]);
 }

 ans.insert( dq.front());

for(  i=k;  i<n; i++)
{         //   ith element will come
          //   i-kth  element will go out
        while( dq.size()>0 && dq.rear() < arr[i])
                dq.pop-rear();

        dq.push-rear( arr[i]);
        if( dq.front() == arr[i-k])
                dq.pop-front();
        ans.insert( dq.front());
}
```

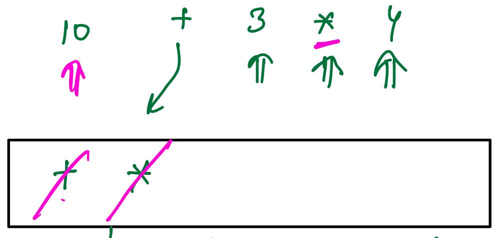
T.C : O(n)

- postfix  infix $\to$ order of execution of operators

- 10 + 3 $\Longrightarrow$ 10 3 +



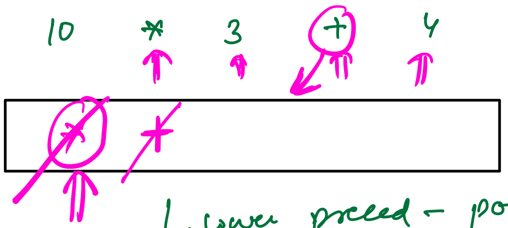
- 10 + 3 * 4



10 3 4 * +

↳ larger precedence come, wait & push

- 10 * 3 + 4

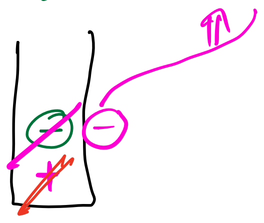


10 3 * 4 +

Lower preced - pop the higher ones

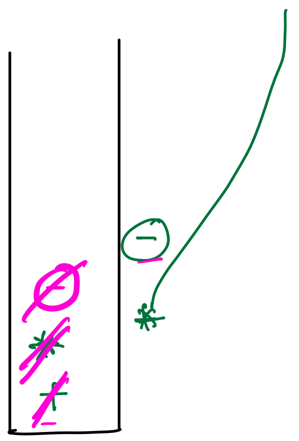
Stack

- 5 + 6 − 10 $\Rightarrow$ equal preced - remove



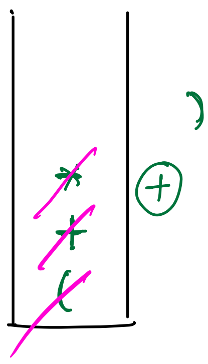
5 6 + 10 −

- $5 + 6 * 10 - 4$



$5\ 6\ 10\ *\ +\ 4\ -$

- $(10 + 3) * 5 \Rightarrow \underline{10\ 3\ +\ 5\ *}$

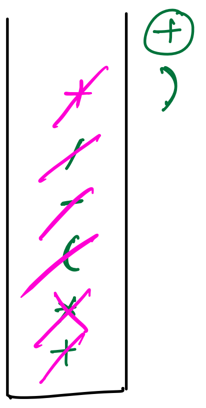


) — closing ) look for the first open(

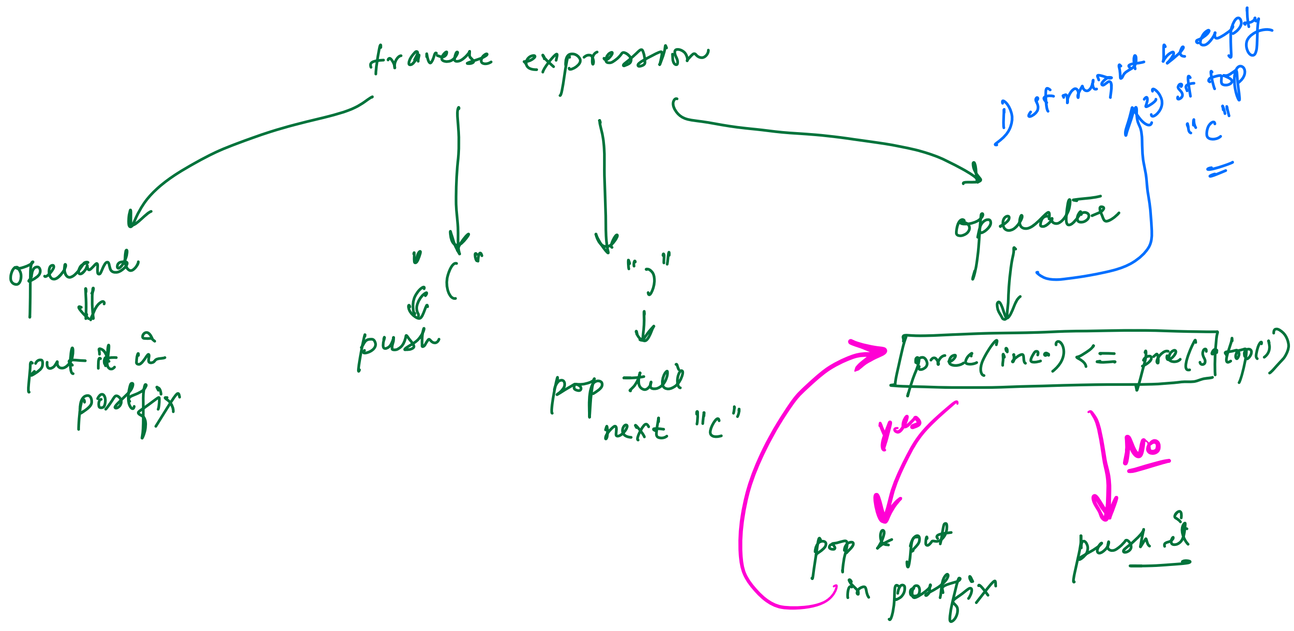
$( ) > \underline{\underline{*, / > +, -}}$

$10\ 3\ +\ 5\ *$

- $3 + 10 * (3 - 4/2) + 3$

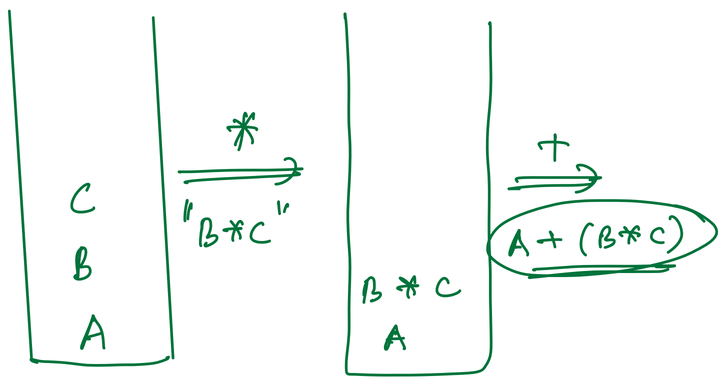


$3\ 10\ 3\ 4\ 2\ /\ -\ *\ +\ 3\ +$



$$\text{Infix} \longrightarrow \text{postfix} \longrightarrow \text{evaluation}$$

$$A + B * C \Rightarrow ABC * +$$



7 + ( 5 * 3 + 3 )

7 5 3 * 3