Q) Rolling Median (Abhishek Saini) :-

Find the median of all subarrays of size len (k)

Ans)

Intuition :-

→ Since we're going to iterate to every window of size 'k'

→ So to a window we'll add elem, remove elem, get median of the window

(sort all the number + get the middle no)

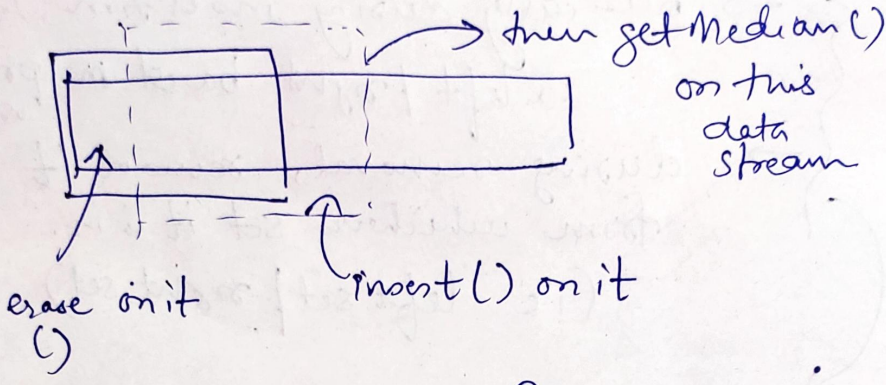
→ we cannot just sort each window manually & then get get median bcz then T.C 'll be

#windows (n-k+1) × (T.C to get median of a window (k log k))

$= O(n^2 \times \log n)$

→ So design a data Structure

on which we'll easily be able to perform | insert(x), erase(x), getMedian() |



* How to get the median?

10, 5, 2, 3, 6, 12, 9

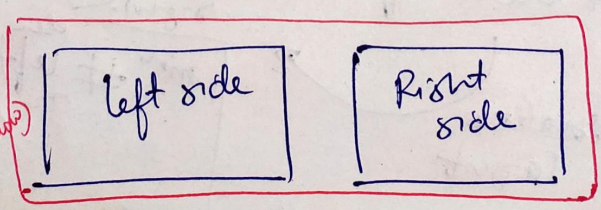
↓ sorted order

2 3 5 [6] 9 10 12

(If we can maintain these 2 parts then largest elem in the left side is the median)

we can use 2 ordered sets.
(to maintain sorted order in each half)

(this is the data Struct. to maintain for every window)



If we can maintain half sorted elems in left & remaining in right set.

Then largest elem in left set is median

how??

→ basically during insertion, insert in left/right based on prev median

→ during removal, remove it from whichever set it is in. (i.e left set/ right set)

So size of left/right might not be what it should be & elems in it might not in sorted from left → right.

So after every insert/remove

↓

these should be some Rebalance

ideally (# elem in left == # elem in right)

unless k = odd, then left has 1 more

So # elem left > # elem right

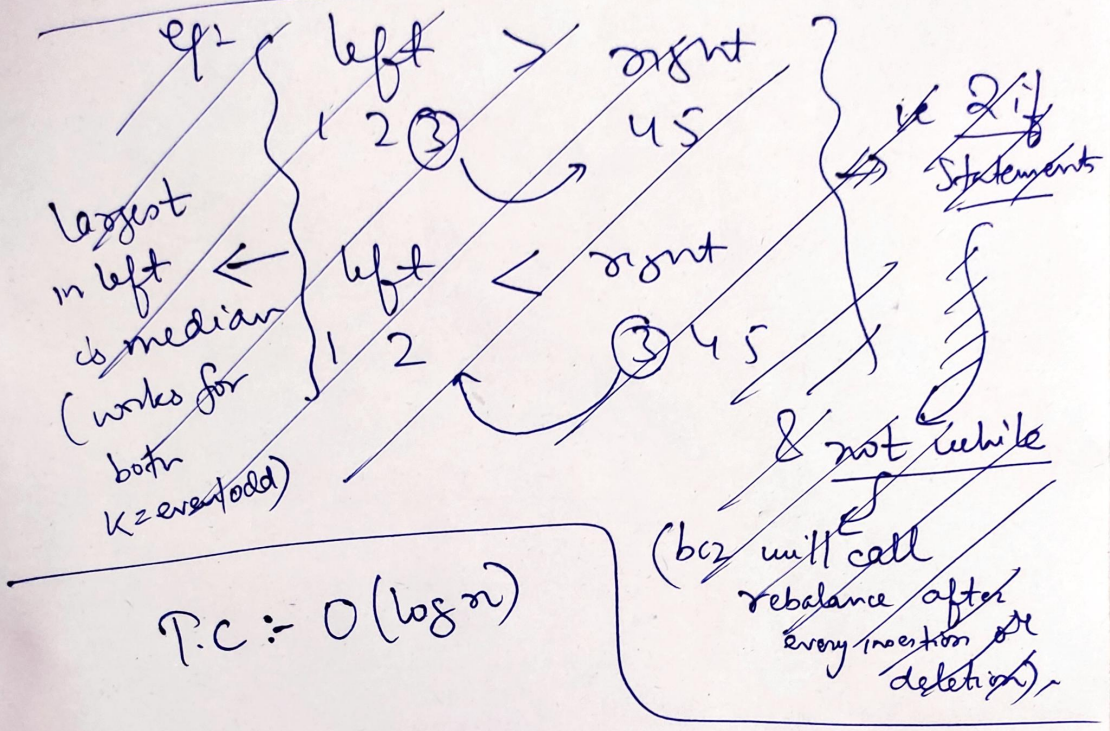
largest

elem left < # elem right

smallest

→ we'll call getMedian() when left & right have equal no of elem

→ (then we'll return largest from the left)



T.C :- O(log n)