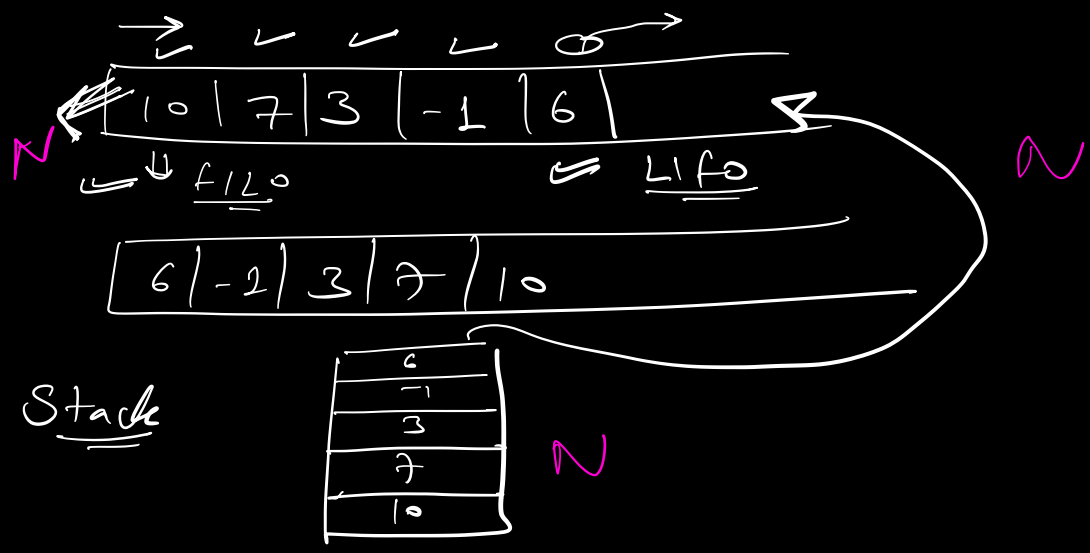
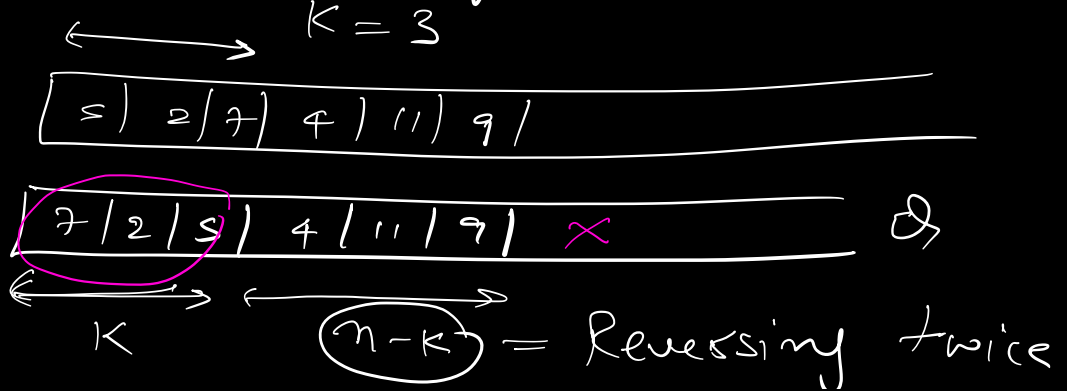
Q ⇒ Given a queue (Stacks)
Reverse the queue.



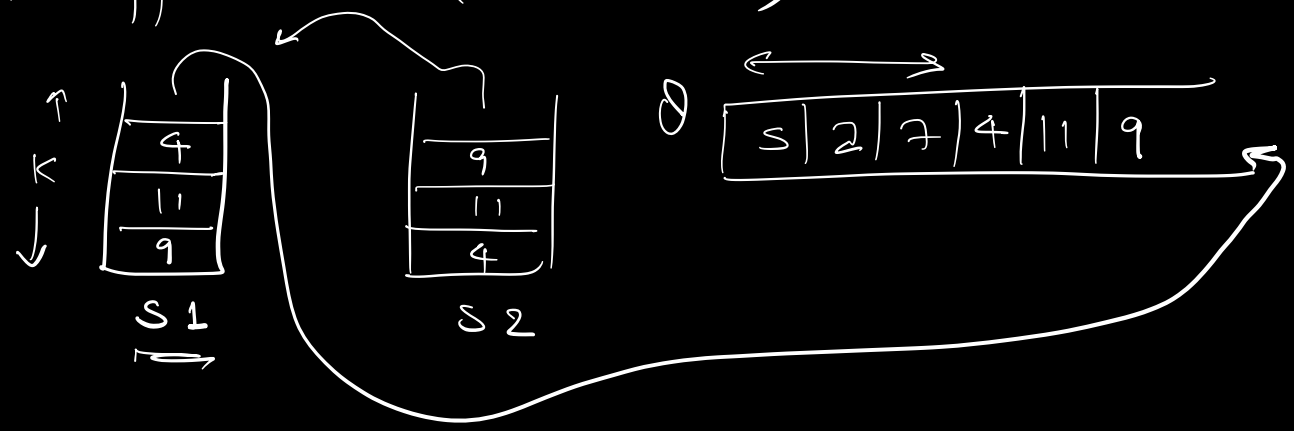
T.C. = O(N)

S.C. = O(N)

Q ⇒ Given a queue, K
Reverse the first K nos of the queue



1) Approach 1 (2 Stacks)

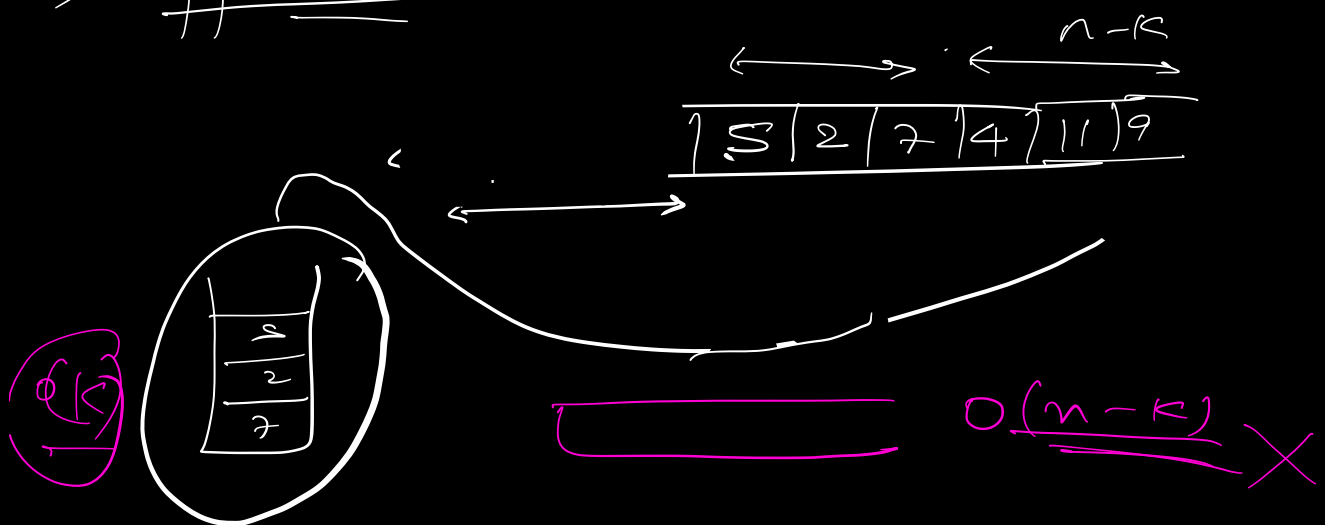


T.C. = O(N)

S.C. = O(N)

---

2) Approach 2



T.C. = O(N)

S.C

Using S2 → O(n)

w/o using S2 → O(K)

Q Stream of characters

(After adding a single char, find the first non repeating character]

If no non repeating chars print "#"

Stream = a b c a d b d c

O/P = a a a b b c c #

Sub Q1 Given a string find the first non repeating char

→ [c c] e d c] :]

Soln : Construct a freq array

a : 0
b : 0
c : 2
d : 1
e : 1

$\Rightarrow$ Add a character to the string

(i) Update the freq array $\Rightarrow$ $O(1)$

(ii) Iterate over the string

$1+2+3+4+\ldots+N = \frac{N(N+1)}{2} = O(N^2)$

T.C. $= O(N^2)$

Optimization

Stream : a b c a

O/P : a a a

Key	Value
a	~~1~~ 2
b	1 (2)
c	1

(i) If a char is coming for the 2nd time, we don't store it.

(ii) When the first char is coming again as an I/P, we remove char from the front. till we find a char with freq 1

⇒ Queue

T.C. = O(N)

S.C. = O(N) ⇒ O(26)

Stream a b c ~~b~~ ~~c~~ d a d

O/P a a a a a a d #

Queue

freq

a	~~1~~	2
b	~~1~~	2
c	~~1~~	2
d	~~1~~	2

Sliding Window Maximum

Given an integer array A.
There is a sliding window of size K
You have to find the max of all windows of size K

$K = 3$

I/P = [1, 3, -1, 3, 5, 3, 6, 7]

O/P = [3, 3, 5, 5, 6, 7]

$\overline{L}$, $\overline{R}$
(i, j)

$K^2 << n$

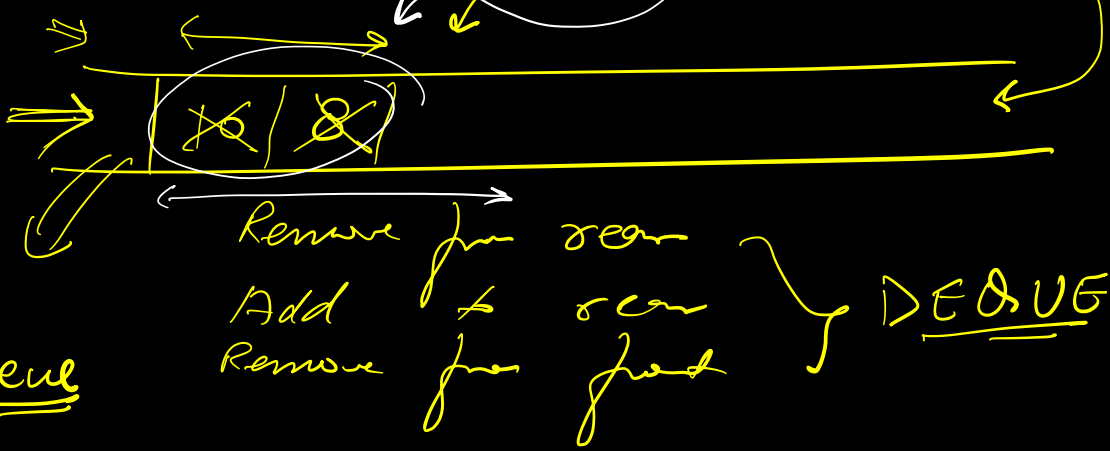
1) Brute force approach
⇒ Calculate the max for every sliding window.

$O(K) * O(n-K)$

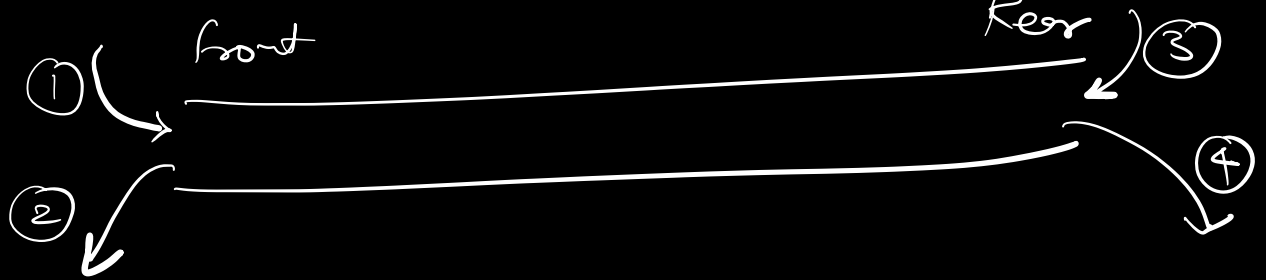
$O(nK - K^2) = O(nK)$

$K = 4$

$A = [1, 10, 3, 2, 8, 11, 6, 7]$

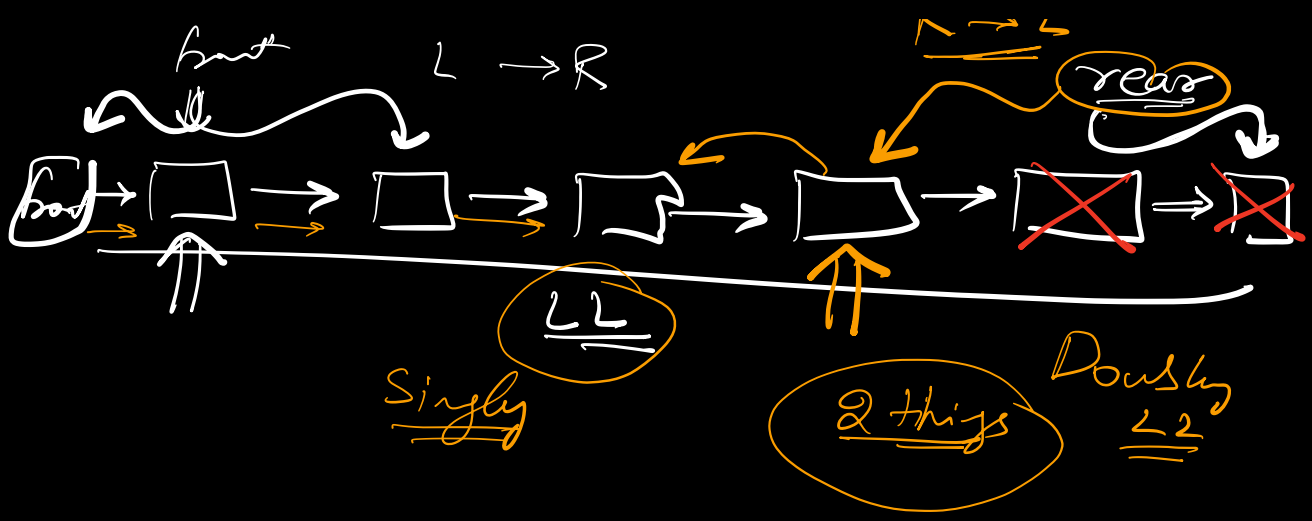


Queue — Remove from rear, Add & Remove from front → DEQUE



Stacks : 3, 4

Queues : 2, 3



K = 4

A = [1, 10, 3, 2, 8, 11, 6, 7]

max = 10

v2 = 3

K = 45