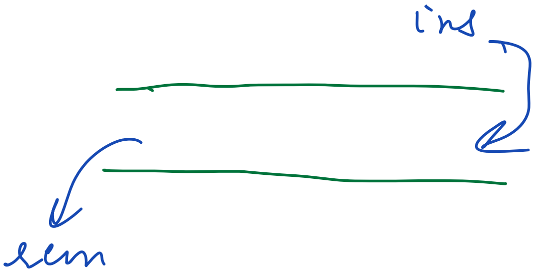
Deque ⇒ Doubly ended queue

Add ⇒ front or rear

Remove ⇒ front or rear



push_front()

push_rear ()

pop_front ()

pop_rear ()

Adobe GS
Microsoft
CRED

Sliding window maximum

Array of size N. Find max elem of every subarray of size K.

10 1 4 9 7 6 5 11 8

k=3 10 9 9 9 7 11 11

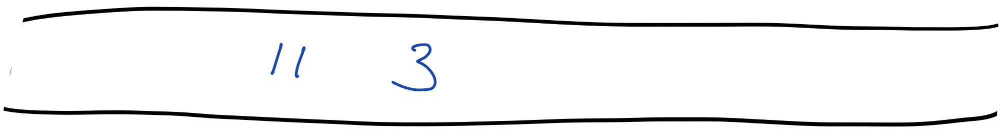
Brute: Check all subarrays of size K & find max. TC: $O(N^2)$

SC: $O(1)$

Idea k=3

10 1 8 9 7 6 5 11 3

↑ ↑



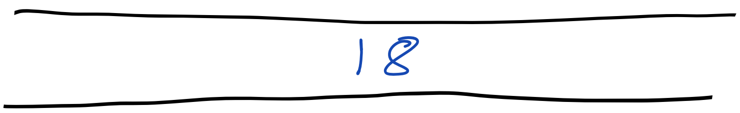
10 9 9 9 7 11 11

access last elem   stack
access first elem   queue

Deque (Doubly ended queue)

k=4

$3 \quad 15 \quad 6 \quad 15 \quad 12 \quad 4 \quad \underbrace{2 \quad 10 \quad 9 \quad 18}$



15 15 15 15 12 10 18

First prepare the deque for first window

new_elem

rear < new_elem
remove rear

rear ≥ new_elem
insert at rear

remove elem

check with front

if equal

remove from front

if not

do nothing

Max ⇒ front element

Code

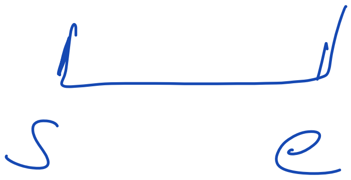
```
List <int> ans                                   0 --- k-1
  Deque <int> dq
  for (i=0; i<k; i++){
      while (!dq.empty() && dq.rear() < arr(i])
              dq.pop_rear()

      dq.push_rear(arr(i])
  }
  ans.insert(dq.front())
  int s= 1           e= k
  while (e<n){
     while (!dq.empty() &&
               dq.rear() < arr[e])
           dq.pop_rear()
     dq.push_rear(arr[e])

     if (dq.front == arr[s-1])
                dq.pop_front()
     ans.insert(dq.front())              S++   e++
}
```

TC: O(N)
SC: O(N)

return ans



1 elem → 1 insert
N elem → N insert

1 elem → 1 removal
N elem → N removal

Total ops = $N + N = 2N$
$\Rightarrow O(N)$

some elem of a window
↓ k

Q First non repeating character

first occurence in string

Eg1 a a b k   a c a b h c

a -1 b b   a a c c c b

Eg2 a a c   a c c a b h

a -1 c   a a a -1 b b

Idea ⇒ If count of char > 1, then cant be answer

⇒ How to use queue

a a b c c b b

a -1 b b b -1 -1

a b c

a a a b a c c b

Code

```
void non_repeat (string s) {
    int freq [26] = {0}
    queue <char> q
    for (i=0 ; i<n ; i++) {
        char ch = s[i]
        freq [ch-'a'] ++
       q. add (ch)
       while (!q. empty ()) {
          if ( freq [q. front() - 'a'] > 1 )
                        q.pop ()
          else {
                break

       }
    }
    if (q. empty () {
              print ( -1 )
    else
```

'a' ⇒ 97

'e' ⇒ 101

a b c d e

0 1 2 3 4

}

}

print (q.front())

TC: O(N)

SC: O(N)

b a a a a a a - - —

b a a a a a a

{done}

a a b c

a	b	c
2	1	1

$\overline{\underline{b \quad c}}$

a -1 b b