Que. Given a string, find first non-repeating character.

$\underline{\underline{a-z}} \rightarrow$ small chars.

Eg: e c h e d f c

$\uparrow$ $\uparrow$ $\uparrow$

x x ✓

h is your ans.

Eg: a b c d a c b

$\downarrow$ ans

Eg: a e a c

$\downarrow$ ans

Eg: a x a x e e

ans = '#'

$\downarrow$ No ans.

$\downarrow$ $\downarrow$ $\downarrow$

Eg: e c h e d f c

a	b	c	d	e	f	g	h	i	j	k	..	z
0	1	2	3	4	5	6	7	8	9	10		25
0	0	2	1	2	1	0	1	0	0	0		0

Steps ① Store freq of chars in freq array

② Iterate on string & whichever char has freq 1, return it.

If none, return '#'

TC : O(N) + O(N)

SC : O(range of chars)

Approach 2

Take a map

0	1	2	3	4	5	6
e	c	h	e	d	f	c

char	freq
e	~~1~~ 2
c	~~1~~ 2
h	1
d	1
f	1

Approach 3

0	1	2	3	4	5	6
e	c	h	e	d	f	c

a	b	c	d	e	f	g	h	i	j	k		z
0	1	2	3	4	5	6	7	8	9	10	..	25
8	8	9	4	9	5	8	2	8	8	8		8

(len+1) 8 $\rightarrow$ char not present till not

(len+2) 9 $\rightarrow$ Invalid .

$$\boxed{TC: O(N) + O(\text{range of chars})}$$

SC: O(range of chars)

Q. Given stream of characters, after adding every character, find first non-repeating character. If none, add # to answer.

Note: string contains only lowercase characters.

FIRST: Non repeating from L to R.

Stream	a	b	c	a	d	e	d	a	b	e	c	g
ans:	a	a	a	b	b	b	b	b	c	c	#	g

stream	x	y	z	y	x	z	m	n	n	h
ans:	x	x	x	x	z	#	m	m	m	m

Brute Force

For every incoming character,

① Add its frequency to freq array $\Rightarrow O(1)$

Approach 1 ↓

Iterate on the given string to get the ans.

$O(N)$ ↓

Approach 2 ↓

Iterate on the array to get the answer.

$O(\text{range of chars})$ ↓

N chars

$O(N^2)$

$O(N \times \text{range of chars}) \simeq O(N)$

stream:	a	b	c	a	d	e	b	d	a	c	b	e	g
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
	a	a	a	b	b	b	c	c	c	e	e	#	g

~~a~~ ~~b~~ ~~c~~ ~~d~~ ~~e~~ g

Freq array

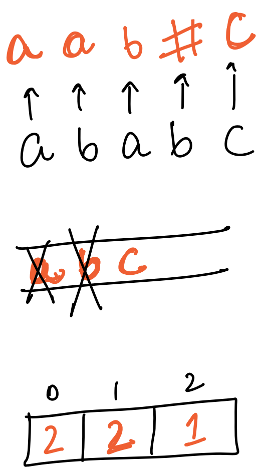
a	:	~~1~~ ~~2~~ 3
b	:	~~1~~ ~~2~~ 3
c	:	~~1~~ 2
d	:	~~1~~ 2
e	:	~~1~~ 2
g	:	1

QUEQUE

```
string stream ( string A ){
    int n = A.length()
    int arr[26] ;
    queue<char> q
    string ans = " "
    for( i=0; i<N ; i++){
        1) update the freq in the array
        arr[ A[i] - 'a' ] ++

        2) Insert in queue
        if( arr[ A[i]-'a' ] == 1) {
            q.enqueue ( A[i])
        }

        3) Print the answer
           Before' that freq of ele in front has to
           be checked
        while (q.size()>0 && A [q.front() - 'a'] > 1)
        {
            q.dequeue ( )
        }

        4) Finally print the answer
        if ( q.size() > 0){
            ans + = q.front( )
        }
        else ans += '#'
    }
    return ans .
```

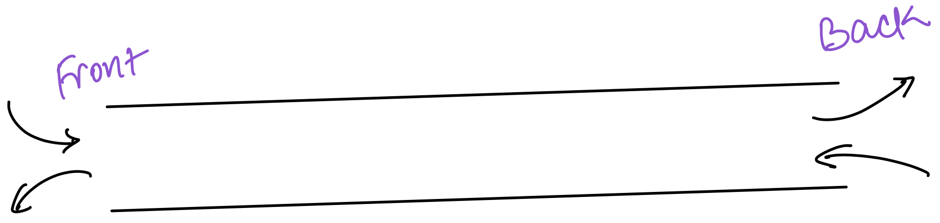
TC: O(N)

SC: O(range of chars)



DEQ

Doubly Ended Queue

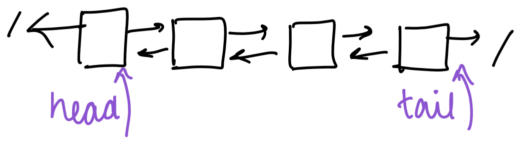


DIY

Functionalities

- push_back( ) : insert at tail
- push_front( ) : insert at head
- pop_back( ) : deleting from tail
- pop_front( ) : deleting from head
- isEmpty( ) : NULL check
- back ( ) : returns the data at tail
- front( ) : returns the data at head
- size ( ) : maintain a variable for it

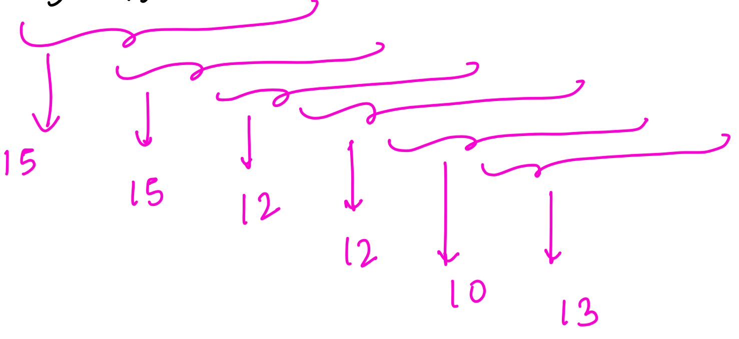
Implementation : DLL



Que. Given array of size N & integer K, find max ele in every window of size K.

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr[ ]:	3	15	6	12	4	2	10	9	13	7	2	5	3

K = 4



Brute Force

Iterate on every window of size K.

Total windows of size K → N-K+1

TC: $O(N-K+1) \times O(K)$

X $K:1$ $(N-1+1)\times(1) = N$

X $K:N$ $(N-N+1)\times(N) = N$

$K:\frac{N}{2}$ $\left(N-\frac{N}{2}+1\right)\times\left(\frac{N}{2}\right) = \left(\frac{N}{2}+1\right)\left(\frac{N}{2}\right) \approx O(N^2)$

K=4

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr[ ]:	3	15	6	12	4	2	10	9	13	7	2	5	3

~~3~~ ~~15~~ ~~6~~ ~~12~~ ~~4~~ ~~2~~ ~~10~~ ~~9~~ ~~13~~ 7 ~~2~~ 5 3

Push at back
Remove at back
Remove at Front
} DEQ (Doubly ended Queue/ DEQUE)

Ans
15 15 12 12 10 13 13 13 13 7

Note : To keep track of valid ele in que, we can keep index.

K=4

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr[ ]:	3	15	6	12	4	2	10	9	13	7	2	5	3

start =
~~0~~
~~1~~
~~2~~
~~3~~ ~~7~~
~~4~~ 8
~~5~~ 9
~~6~~

~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	~~8~~	9	~~10~~	11	12
~~(3)~~	~~(15)~~	~~(6)~~	~~(12)~~	~~(4)~~	~~(2)~~	~~(10)~~	~~(9)~~	~~(13)~~	(7)	~~(2)~~	(5)	(3)

Ans: 15 15 12 12 10 13 13 13 13 7

```
list<int> ans;
deque<int> dq;
start = 0
for(i=0; i<K ; i++){
    while( dq.size()>0  &&  A[i] > A [dq.back()]){
        dq.pop_back()
    }
    dq.push_back(i)
}
ans.add( A[ dq.front()])
start ++
for( i=K; i< N; i++){
    while( dq.size()>0  &&  A[i] > A [dq.back()]){
        dq.pop_back()
    }
    dq.push_back(i)
    if( start > dq.front()) dq.pop_front()
    ans.add( A[dq.front()])
    start ++
}
```

$$\boxed{\begin{array}{l} TC : O(N) \\ SC : O(K) \end{array}}$$

$$\begin{array}{cccccccccc}
 & & & & & & & & i & \\
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\
\{ & \boxed{1 \quad 3 \quad -1} & & & -3 & 5 & 3 & 6 & 7 & \}
\end{array}$$

$K = 3$

$$\begin{array}{cccccccc}
\cancel{0} & \cancel{1} & \cancel{2} & \cancel{3} & \cancel{4} & \cancel{5} & \cancel{6} & (7) \\
(1) & \cancel{(3)} & \cancel{(-1)} & \cancel{(-3)} & \cancel{(5)} & \cancel{(3)} & \cancel{(6)} & (7)
\end{array}$$

3 3 5 5 6 7