Agenda

1) First non-repeating character **

2) Intro to deque (Doubly ended queue)

3) Sliding window maximum **

Q.1 Given a string A, denoting stream of lowercase alphabets. Find first non-repeating char, each time a char is coming in A string stream.

(till every index what is the 1st non-repeating char)

A = a b a b c

ans : a a b # c

Expected TC: O(n)

A = a b c a c e b

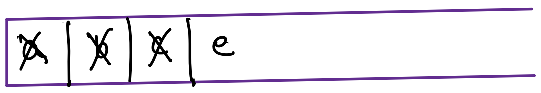
ans: a a a b b b e

A = a b c a c b d K a d

ans: a a a b b # d d d K

A = a b c a c e b

ans : a a a b b b e

queue


$a \to \cancel{1}\,2$

$b \to \cancel{1}\,2$

$c \to \cancel{1}\,2$

$e \to 1$

map
(char vs Int)

```
Queue < Character > q = new ArrayDeque < >( );
HashMap < Character, Integer > map = new HashMap < > ( );
StringBuilder ans = new StringBuilder( );

for (int i = 0; i < A.length( ); i++) {
        char ch = A.charAt(i);
        if (ch is coming first time) {
                map.put (ch, 1);
                q.add (ch);
        }
        else {
                map.put (ch, updated_freq);
        }
        while (q.size( ) > 0 && map.get (q.peek( )) > 1) {
                q.remove( );
        }
        // 1st non repeating char till now is q.peek( ), but if
           queue became empty then no first non-repeating char is
           there till ith index so use #.
}
```

queue add → n

queue removal → n

itr = 2n, TC: O(n)

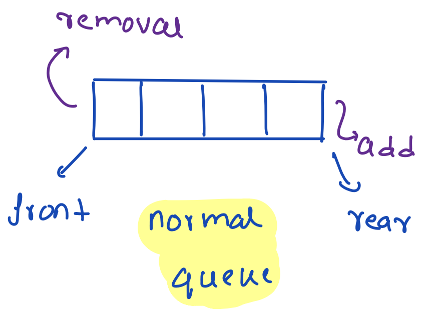
space in HM → 26

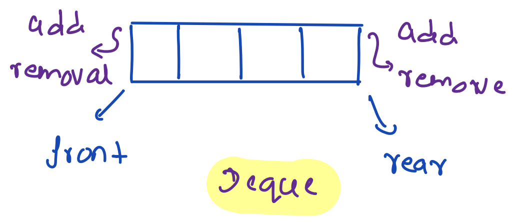
space in queue → 26

SC: O(1)

Intro to Deque

Doubly Ended queue ⇒ Deque





{ DLL is also used to create and work with Deque }

How to create Deque in Java and use it:

```
ArrayDeque <Integer> dq = new ArrayDeque<>();
```

function

```
dq.addLast();  //  or   dq.add()
dq.addFirst();
dq.removeLast();
dq.removeFirst();  //  or   dq.remove()
dq.getFirst();
dq.getLast();
```

Each function

TC: O(1)

Q.2 **Sliding window Maximum**

Given an array of int values A[] and k, find max of every subarray of length k in A[].

	0	1	2	3	4	5	6	7	8	
A =	10	2	9	3	1	6	5	11	8	k = 3
ans :	10	9	9	6	6	11	11			

	0	1	2	3	4	5	6	7	8	9	
A =	3	15	16	12	4	2	10	9	13	7	k = 4
ans :	16	16	16	12	10	13	13				

In array len = n, how many subarrays of k length

$= n-k+1$

go on every window of size k and find its max :

itr : $(n-k+1) * k$

$= (n-\frac{n}{2}+1) * n$ $\quad \{ \text{ if } k=\frac{n}{2} \}$

$\approx n^2$

TC : $O(n^2)$

Expected TC: O(n)

		i		j					
	0	1	2	3	4	5	6	7	8
A =	~~10~~	2	9	3	1	6	5	11	8

K = 3

max = 10

{single max is not enough}

								i		j
	0	1	2	3	4	5	6	7	8	
A =	10	2	9	3	1	6	5	11	8	
ans :	10	9	9	6	6	11	11			

K = 3

deque :	~~10~~	~~2~~	~~9~~	~~3~~	~~1~~	~~6~~	~~5~~	11	8

keep values (dec order) $\longrightarrow$

aquire (j)

release (i-1)

								i			j
	0	1	2	3	4	5	6	7	8	9	
A =	3	15	16	12	4	2	10	9	13	7	
ans :	16	16	16	12	10	13	13				

K = 4

deque :	~~3~~	~~15~~	~~16~~	~~12~~	~~4~~	~~2~~	~~10~~	~~9~~	13	7

aquire (j)

release (i-1)

```
void  sliding_window_max (int [ ]A, int K) {

    ArrayDeque <Integer> dq = new ArrayDeque<>();
    //calculate ans of 1st window
    for(int i=0; i<K; i++) {
            while (dq.size()>0 && dq.getLast () < arr [i]) {
                    dq.removeLast();
            }
            dq.addlast (arr[i]);
    }
    sopln( dq.getFirst ());

    // travel the rest of the windows
     int i=1 , j=K;
     while (j < arr.length ) {
            //aquire jth ele
            while (dq.size()>0 && dq.getLast () < arr [j]) {
                    dq.removeLast();
            }
            dq.addlast (arr[j]);

            // release (i-1)th ele
            if(dq.getFirst() == arr [i-1]) {
                    dq.removeFirst();
            }
            sopln( dq.getFirst());
             i++; j++;
     }

}
```

Dry run

```
ArrayDeque<Integer> dq = new ArrayDeque<>();
//calculate ans of 1st window
for(int i=0; i<k; i++) {
        while (dq.size()>0 && dq.getLast() < arr[i]) {
                dq.removeLast();
        }
        dq.addLast(arr[i]);
}
sopln( dq.getFirst());
//travel rest of the windows
int i=1, j=k;
while (j < arr.length) {
        //aquire jth ele
        while (dq.size()>0 && dq.getLast() < arr[j]) {
                dq.removeLast();
        }
        dq.addLast(arr[j]);

        // release (i-1)th ele
        if(dq.getFirst() == arr[i-1]) {
                dq.removeFirst();
        }
        sopln( dq.getFirst());
        i++; j++;
}
```

					i			j
	0	1	2	3	4	5	6	7
A =	3	15	16	12	4	2	10	19
O/P :	16	16	16	12	19			

k = 4

dq | ~~3~~ | ~~15~~ | ~~16~~ | ~~12~~ | ~~4~~ | ~~2~~ | ~~10~~ | 19 |