nums → [19, 17, 1, 3, 18, 2, 5, 6, 16, 15, 14]

ans - 6

1, 2, 3

5, 6

14, 15, 16, 17, 18, 19 ✓ → 6

Sequence → We don't care about order of the elements in which they are arranged.

↓ sort

[19, 17, 1, 3, 18, 2, 5, 6, 16, 15, 14]

A consecutive sequence can be visualized as a sorted sequence.

Sorted form of given array

→ [1, 2, 3, 5, 6, 14, 15, 16, 17, 18, 19]

↑ i

```
if(arr[i] == 1+arr[i-1])
```

curr_len = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

ans = ~~1~~ ~~3~~ 6

$$\underline{\underline{[10]}} \longrightarrow \underline{\underline{1}}$$

$$[\underbrace{1, 2, 3}_{S_1}, \underbrace{5, 6}_{S_2}, \underbrace{14, 15, 16, 17, 18, 19}_{S_3}]$$

for any consecutive sequence, how can we uniquely identify it ??

★ → start of the sequence

→ end of the sequence

★ → length of the sequence.

} any 2 properties

Start = 10

length = 7 → 10, 11, 12, 13, 14, 15, 16

OR

end = 7 → 5, 6, 7

length = 3

OR

Start = 1 1, 2, 3, 4

end = 4

[19, 17, 1, 3, 18, 2, 5, 6, 16, 15, 14]

We will try to check if the current element is a starting point of a consecutive sequence or not ??

x → if we donot have $x-1$ in the array

current element

↓

then x can become starting element

[19, 17, 1, 3, 18, 2, 5, 6, 16, 15, 14]

↑i (at 1) ↑i (at 14)

time → O(n)

possible start = ~~1~~ ~~3~~ 14 ← st

st = ~~14~~ ~~15~~ ~~16~~ ~~17~~ ~~18~~ 19

curr_len = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

ans = ~~3~~ 6

we will try to generate a consecutive sequence from st, by checking again & again whether st+1 is present or not ??

How to figure out whether any element exists in the array efficiently ?

JS → objects

key value

↳ < elements index >

sets

Complexity will be O(n) because we are touching every element at max **twice.**

$$\rightarrow 2n \rightarrow \underline{\underline{O(n)}} \quad //$$

Time **Space**

$n \log n$

$10^5 \qquad \approx 18 \approx \underline{\underline{20}}$

$n \to 10^5$

$n^2 \to \underline{\underline{10^{10}}}$

20×10^5

$2 \times 10^6 \le 10^8$

$10^{10} > 10^8$

Brute force

$n \times (n-1)$

$\downarrow$

$O(n^2)$

love leetcode (n)

$\uparrow$

$(n-1)$

for every char check the remaining String.
if in the remaining String we found the char then
this char won't be the ans.

Observation → for all those characters which are repeating, we will be having a frequency more than 1.

How about creating a frequency map ??

love leetcode

key	value
char	frequency
l	2
o	2
v	1
e	4
t	1
c	1
d	1

→ **object**

O(n)

Space → O(1)

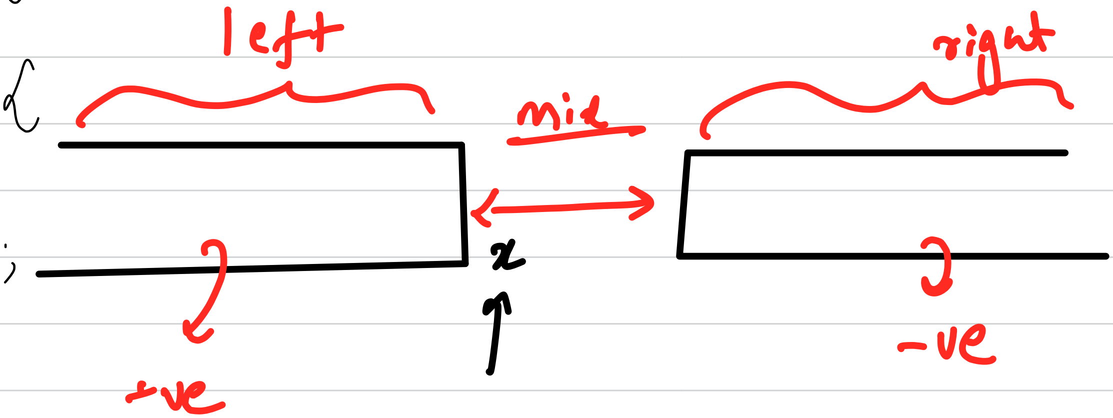
↓

const

[-3, 1, 6, 5, -8, -1, 9, -5, 13]

```
while (i<j) {
    if(arr[i]<0){
        // move to right
        swap(arr, i, j);
        j--;
    } else {
        i++
    }
}
```

$O(n)$



[1, 6, 5, 9, -2, 5, -3, -6, -7]

i j

before i we have left region

→ after j, we have right region

$1, 3, 9, \underset{\uparrow i}{-5}, \underset{\uparrow j}{10}, -8 -6, -7, -1$