Majority Element - II

Tuesday, 31 August 2021 8:42 PM

$N/3$ — Element which have more than $N/3$

arr → 1 2 3 1 4 1 7 1 1 9 6 8

freq.

- Self Reading.
- problem Introduce
- Thinking
- Explanation
- TRY
- code. → Ok

$N/3 \times x = N$ → $N/3$

$x = \frac{3 \times N}{N} = 3$

3 elements possible which have freq. Exactly $n/3$

→ more $n/3$ →

Allowed

time complexity → $O(n)$

space complexity → $O(1)$ → with hash map $O(n)$

Hint → Triplet

2 counts

2 val → val1 ×, val2 ×

Triplet prepare

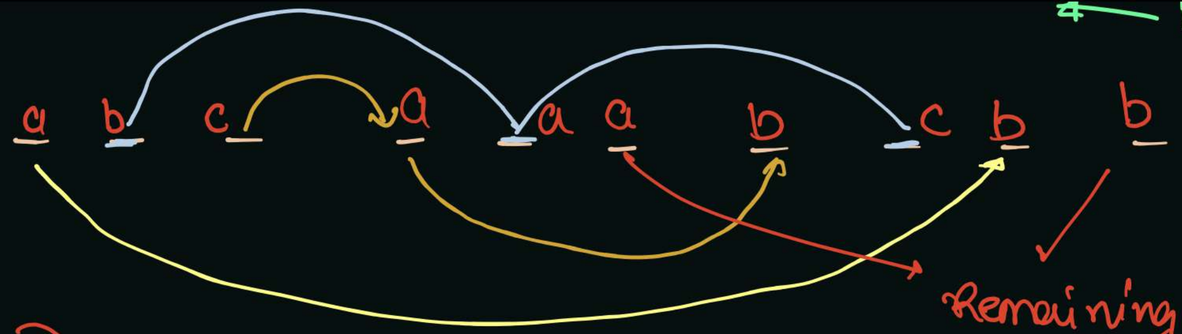
At most 2 Elements more $n/3$ freq.

Majority Element → $N/2$ < freq →

$< \left(\frac{n}{2} + \frac{n}{2}\right) = n$.

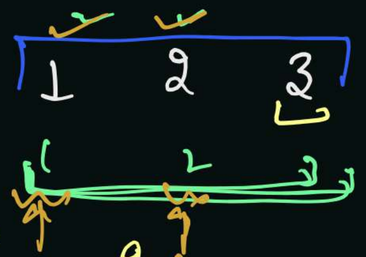
count $< n/2$ → 1 Element more $n/2$ freq

Mapping of Triplets-



val1 = a
val2 = b
→ check if freq more than N/3 or not!

arr → 1 2 3 | 1 2 3 | 3 1 2 | 2 1



val1 = arr[0] = 1 3

count1 = 1 0 1 0 1 0 1

val2 = arr[0] = 1 2 1

count2 = 0 1 0 1 0 1 0 1

val2 = 3 2

count = 0 1

Valid candidate for majority element

val1 = 2 → Check it

val2 = 1 → from more than n/3.

4 > 11/3

n/3

4 > 11/3

new val → nval == val1 → val1 count participate

→ nval == val2 → val2 count participate

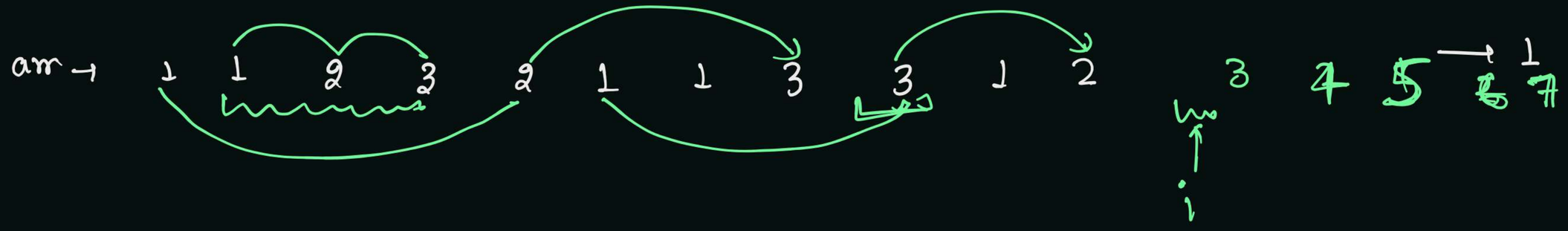
otherwise - distinct from val1 & val2 contribute in triplet

otherwise -

count1 == 0
// initialise val1
val1 = arr[i];
count1 = 1;

count2 == 0
// initialise val2 → val2 = arr[i]
count2 = 1;

otherwise →
// triplet → count1--
count2--



val1 == arr[i]
count1 ++;

val2 == arr[i]
count2++;

else{
count1 == 0

count2 == 0
val2 = arr[i]
count2 = 1
otherwise
count1 --
count2 --
}

val1 = arr[0] = 1

count1 = 1 2 1 2 3 2 2 2

val2 = arr[0] = 1 2 3 ⇒

count2 = 0 1 0 1 0 1 0

check

→ 1 2 3 1 2 3 1 2 3

Next Greater Element - 3

Thursday, 2 September 2021 9:59 PM

i/p "213" (string) → next greater

o/p → 231

set of digits → {1, 2, 3}

all permutation →

- 1 2 3
- 1 3 2
- 2 1 3
- 2 3 1
- 3 1 2
- 3 2 1

increasing order

→ 213

↳ next greater

213 → 231

input < all possible greater number than given input

min of this set is output

i/p → "132" → next greater → o/p → "213"

String n → "2 1 3"

number → _ _ _ _ _ _ _

Most significant digit

Least significant digit

→ Smallest possible number using same digit = Increasing order "1 2 3"

→ Largest possible number using digit = Decreasing order "3 2 1"

Most significant digit

smaller → smaller num

greater → greater number

Number → "2 1 4 8 7 5 3 2"

ceil of 4

digit which can Replace point ?? ceil of 4 from Right part

point on which we can make greater Element

given input

218

217

215

After Replacing with ceil ⟶

Input → 2 1 [4] 8 7 5 3 2 ⟶ input

Point

Replace with ceil of Right part

Swap.

max form

2 1 # [5] [8 7 4 3 2] ⟶ next greater?? of given input

greater ||

next greater ✗

Reverse (Right part)

2 1 5 2 3 4 7 8 ⟶ output

2 1 4 [max form]

2 1 5 [max fm]

2 1 4 [max] < 2 1 5 [min fm]

2 1 4 [9 8 7] < 2 1 5 [2 8 9]

max min

Steps of Algorithm.

(1) find point from which we can swap digit to make next greater element. → if no. is already in max form return -1

O(n)

(2) Swap pointed digit from its ceil from Right part.

O(1)

(3) Reverse Right part to make it min. possible no.

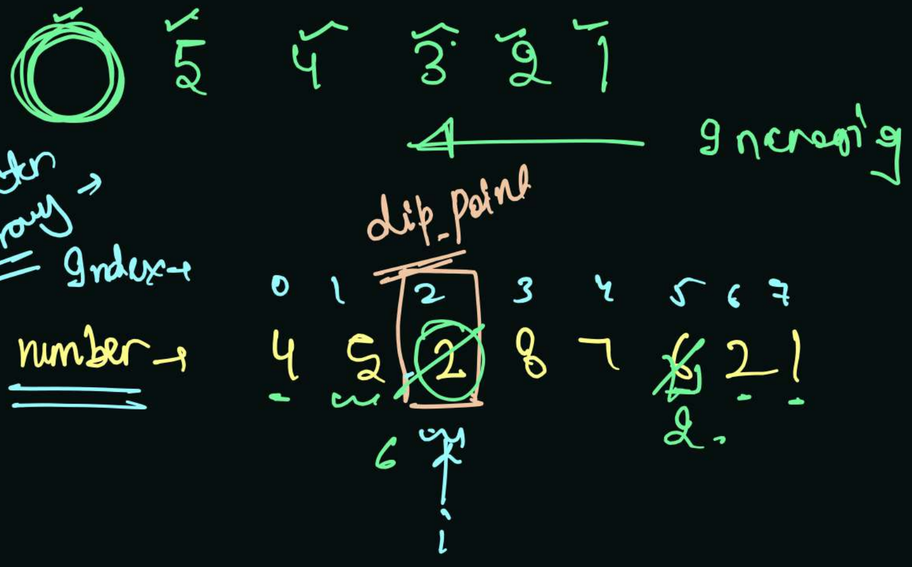
O(n)

(4) Convert this into String.

O(n)

Ceil → qualified Min.

Complexity → O(3n) ≡ O(n) (n → no. of digits)



dip → 2

dip index

ceil → 8

ceil index

Swap (2, 5)

number → 4 5 6 8 7 2 2 1

Reverse from dip index +1 to end

→ 4 5 6 1 2 2 7 8

2 1 2 2 2 1

dip

2 4 2 2 2 1

2 1 2 2 2 3

factor greater

Min. from all greater

Max Product Of 3 Numbers

Thursday, 2 September 2021 11:20 PM

arr → 3, 2, -12, -6, -11, -9, 1, 7, 10, 4,

max1 =
max2 =
max3 =

max1 * max2 * max3 → result

Elements combe -ve.

min1 = -12
min2 = -11
max

min1 * min2 → +ve

max1 = 10
max2 = 7
max3 = 6

task → find
min1, min2,
max1, max2, max3
in single Enteration

may be it can overcome max2 * max3

figure for max product

result = max [max1 * max2 * max3 , 10 * 7 * 6]

VS

max

min1 * min2 * max1

-12 * -11 * 10

if single value is -ve → Longer

How to find max and second max in single iteration →

max = ~~-∞~~ ~~4~~ ~~6~~ ~~7~~ 10

Smax = ~~-∞~~ ~~3~~ ~~4~~ ~~6~~ ~~7~~ 9

```
if(arr[i] > max){
    smax = max;
    max = arr[i];
} else if( arr[i] > smax){
    smax = arr[i];
}
```

End of ~~iteration~~

max → 10

smax → 9

] in single iteration

Similarly find max and min

↓

[max1
  max2
  max3

↓

[min1
  min2
  min3