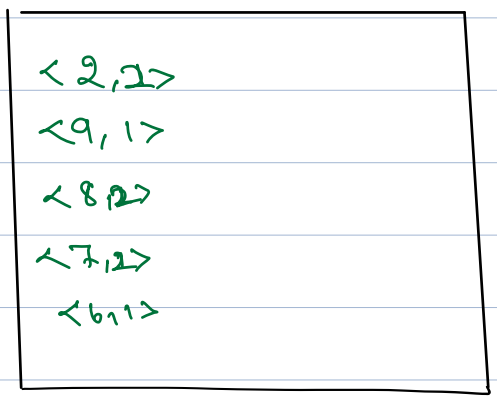
Ques) Given N array elements, find no. of distinct elements.

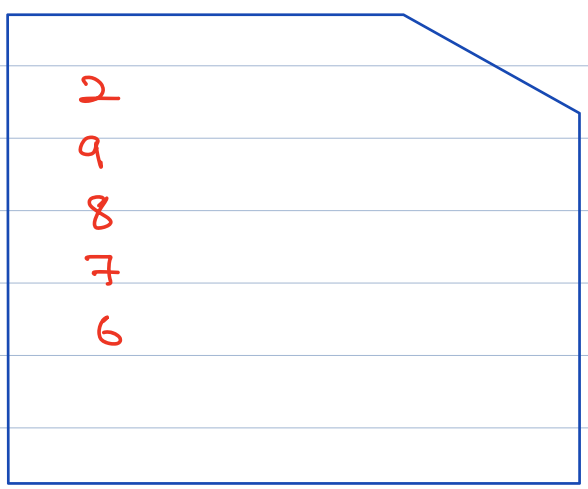
arr[7] = {6, 3, 7, 3, 8, 6, 9} → 5.

arr[8] = {2, 9, 8, 7, 6, 8, 2, 7} → 5.



→ hashmap size.

T.C → O(N)



T.C → O(N).

→ set.size().

```
HashSet<Integer> hset;
for (i=0; i<n; i++){
        hs.insert(arr[i]);
}
print(hs.size());
```

(If we store same keys again in hashset, it will only store a single occurence).

Ques) :- Given N array elements, check if there exists a subarray with sum=0.

	0	1	2	3	4	5	6	7	8	9
arr :-	2	2	1	-3	4	3	1	-2	-3	2
pf :-	2	4	5	2	6	9	10	8	5	7.

5

5

pf[2] = sum[0-2] = 5

pf[8] = sum[0-8] = 5.

pf[8] = sum[0-2] + sum[3-8]. → pf[8] = pf[2] + sum[3-8].

These elements sum's 0.

Brute force :-

1) Generate all subarray sums → carry forward
   - $O(N^3)$
   - $O(N^2)$

$sum[3-5] = pf[5] - pf[2]$.

$sum[3-5] = 0$.

Obs :- ① If your pf[], if an element repeats, subarray sum = 0.

② check if 0 is in pf[].

	0	1	2	3
Ex :-	3	-1	-2	4
pf :-	3	2	0	4

Pseudocode :-

Steps :-

1) calculate pf[].
2) check if 0 is present in pf[].
3) check if element repeats in pf[].

↳ do it with prev ques.

↓

In the last ques end check :-

(hset.size() == N) or not.

Ques) find the length of longest subarray with sum = 0.

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr[] =	3	3	4	-5	-2	2	1	-3	3	-1	5	-4	-1
pf[] =	3	6	10	5	3	5	6	3	6	5	10	6	5

Obs :- for every element in pf(), store its 1st and last occurrence.

2 hashMaps :-

1 hashMap → for every elem 3'll store first occurr

2 hashmap → for every elem 3'll store last occurrence.

	0	1	2	3	4	5	6	7	8	9	10	11	12
pf[] =	3	6	10	5	3	5	6	3	6	5	10	6	5

<3,0>
<6,1>
<10,2>
<5,3>

first index	len	ans
0	4	4
3	2	4
1	6	5
0	7	7
1	7	7
3	6	7
2	8	8
1	10	10
3	9	10

hm <int, int>.

hm.insert<0,-1> → Edge case.

```
for(i=0; i<n; i++){
    if(pf[i] is in hm){
        int len = i - hm.get(pf[i])
        ans = max(ans, len);
    }
    else{
        hm.insert(pf[i], i);
    }
}
```

Edge Case :-

	0	1	2	3	4	5
arr →	4	-3	-1	2	3	-5
pf →	4	1	0	2	5	0

<0, -1>

Ques) Given array only contains 0's and 1's, find max length subarray which contains equal 0's and 1's.

Soln:- replace all 0 → with -1.

→ 0 0 1 1 0 0 1 1

↓

-1 -1 1 1 -1 -1 1 1

Ques) Count the subarrays with sum = 0.

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr[] =	3	3	4	-5	-2	2	1	-3	3	-1	5	-4	-1
pf[] =	3	6	10	5	3	5	6	3	6	5	10	6	5

$5 \rightarrow 4 \rightarrow {}^4C_2 \rightarrow 6$

$6 \rightarrow 3 \rightarrow {}^4C_2 \rightarrow 6$..

(*) Count no. of times a PfSum[] is repeating.

PfSum freq.

3 → 3 → $^3C_2$ +

6 → 4 → $^4C_2$ + => 16

10 → 2 → $^2C_2$ +

5 → 4 → $^4C_2$ +

9 → 1 → $^1C_2$ → $1(\frac{1-1}{2})$ no.

$^nC_2$ => $\frac{n(n-1)}{2}$

$\frac{n}{2!(n-2)!}$ => $\frac{n(n-1)(n-2)!}{2 \times (n-2)!}$

Pseudo code :-

① HashMap<Int, Int> fmap;

② Create a Pf array.

③ Store freq of Pf array in fmap.

add (0,1)?

int ans = 0;

for each loop

for (int val : fmap.keySet()) {

int freq = fmap.get(val);

ans = ans + $\left(freq * \frac{freq-1}{2}\right)$

3

return ans;

Edge Case :-

arr →	4	-3	-1	2	3	-5
pf →	4	1	0	2	5	0

0 → 2 → $^2C_2$ → 1.

0 → 3 → $^3C_2$ → 3.

<0,1>

↳ add it to start.

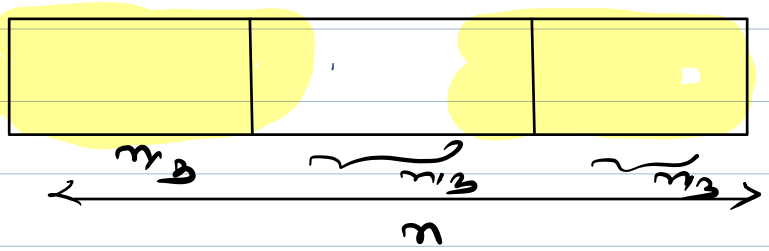
There is a sum 0 with freq 1.

⊛ Arrays and Maths

* Majority Element ($N/3$) → freq should be greater than $\frac{N}{3}$.

(10, 10, 20, 30, 10, 10) → $\frac{6}{3}$ → 2 → 3

→ How many majority elements an array can have ?.



At most two majority element can be there.

* Here, if you remove 3 distinct elements majority will not change.

10, 10, 20, 30, 10, 10, 80

maj1 = 10 maj2 = ~~20~~ 20

freq1 = ~~0~~ ~~1~~ ~~2~~ ~~1~~ 3 freq2 = ~~0~~ 1 0

Pseudo Code :-

```
int maj1 = -∞
int maj2 = ∞
for (int val : arr) {          // val : arr[i];
    if (maj1 == val) {
        freq1++;
    } else if (maj2 == val) {
        freq2++;
    } else if (freq1 == 0) {
        freq1 = 1;
        maj1 = val;
    } else if (freq2 == 0) {
        freq2 = 1;
        maj2 = val;
    } else {
        freq1--;
        freq2--;
    }
}
```

for(i=0; i<n; i++)
{ val = arr[i]

for each loop.

$\frac{12}{3}^4 \rightarrow (5)$

N = 12 → min freq ⇒ $\frac{12}{3}^4 \rightarrow 5$

4 5 6 6 5 5 6 5 6 4 5 6

may1 = 4 6 may2 = ~~6~~ 5

freq1 = ~~0~~ X 0 X 2 ~~3~~ 2 3 freq2 = ~~0~~ X 0 X 2 X 2

$\frac{N}{2}$ majority → Once you realise that.

$\frac{N}{2}$ → 1 Max → 2

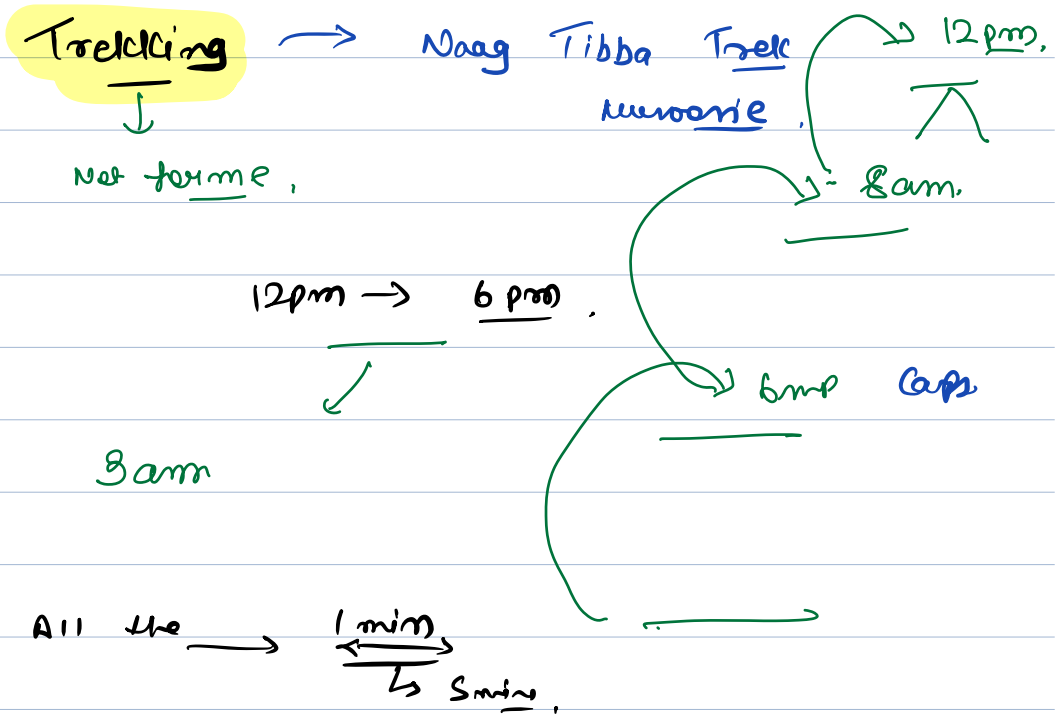
$\frac{N}{3}$ → 2 → 3

$\frac{N}{4}$ → 3 → 4

$\frac{N}{K}$ → K-1 → K.

2:04 - 2:14 pm.

Top



It will happen eventually.

Trekking → Once.

← Comparator →

Ques) Given n array elements, sort them in increasing order of their no' of factors.

If two elements have same factors, ele with less value should come first.

	9	3	4	8	16	32	6	13	15
# factors.	3	2	3	4	5	2	4	2	4

→ 3, 13, 37, 4, 9, 6, 8, 15, 16.

Eg:- → {1, 21, 6, 23, 10, 14, 25}

factors. 1 4 4 2 4 4 3

→ {1, 23, 25, 6, 10, 14, 21}

→ Arrays.sort(arr) *

int arr[] = {1, 2, 3, 4, 5, -1}.

```
Arrays.sort(arr, new Comparator<Integer>(){
// for descending.
    @Override
    public int compare(Integer a, Integer b){
                                  100   20
        if (a > b) {
// you want a to come first
            return -1
        } else {
// you want b to come first.
            return 1
        }
    });

Arrays.sort(arr, new Comparator<Integer>(){
```

```
@Override
public int compare(Integer a, Integer b){
    if (a should b come before b
        return -1
    if a = b return 0
    else return 1.
}
});
```

Sorting on factors

```
Arrays.sort(arr, new Comparator<Integer>(){

@Override
public int compare(Integer a, Integer b){
    int f1 = factor(a)
    int f2 = factor(b)
    if (f1 < f2) {
        return -1
    } else if (f1 == f2 and a<b){
        return -1
    } else {
        return 1
    }
}
```

3);

```
Arrays.sort(arr,new Comparator<Integer>(){

    @Override
    public int compare(Integer o1, Integer o2) {
        return o1.compareTo(o2);
    }
});

for(int str:list){
    System.out.println(str);
}
```

write your logic

above logic will not work on,
int[] arr,
Integer[] arr.

```java
Integer[] arr = {1,2,3,4,5};
ArrayList<Integer> list = new ArrayList<>();
list.add(1);
list.add(2);
list.add(3);
list.add(4);

Collections.sort(list,new Comparator<Integer>(){

    @Override
    public int compare(Integer a, Integer b) {
        if(a<b){
            return -1;
        }else{
            return 1;
        }
    }
});

Arrays.sort(arr,new Comparator<Integer>(){

    @Override
    public int compare(Integer o1, Integer o2) {
        return o1.compareTo(o2);
    }
});
```

(Ques) :- Given N array ele, sort in increasing order of no. of digits, if two ele have same no. of digits, ele with more value should come first.

93, 2, 97, 639, 8, 100, 345, 49

2 1 2 3 1 3 3 2

→ 8 2 93 49 37 639 345 100.

```
Arrays.sort (arr, new Comparator<Integer>(){

    @Override
    public int compare(Integer a, Integer b){
        int d1 = digits(a)
        int d2 = digits(b)
        if(d1 < d2){
            return -1;
        }
        if (d1 == d2 and a > b){
            return -1
        } else {
            return 1
        }
    });
```

Tuesday → Last class.

→ Pending Problem

Problem solving session

↳ Timing.

7 days gap you'll get.

Advance → ?

Arrays
Bits
recursion
HashMap
Strings
} → Confident

H.W – 50-60%.

7 days break

↳ ×
10LPA ⟶ 40LPA.
20years.
1years
↓
Time is the currency.

10years

α → β → γ → δ

DSA 50 people Slack / whatsapp
TA.
me

Begin 5%
Int 15 -25%.
Adv rest.

* Rest in the end not in the middle.

Advances → 4-5 months

kumar.aman@scaler.com.