Group Anagrams.

Anagrams?

(S1) and (S2)

↳ when no. of occ. of all character in $S_1 == S_2$
then they are anagramic to each other.

S1 : "accio" S2 : "ciaco"

① sort(S1) sort(S2)

"accio" "aciio"

equal ↳ when they are anagrams

S1.equals(S2)

↳ O(N)

TC : O(N log N) &

SC : O(1)

(2)

S_1 → freq Map of each char

"accio"

key (char)	value (int)
a	1
c	2
i	1
o	1

Map1

S_2 → freq Map of each char

"cicada"

a	1
c	2
i	1
o	1

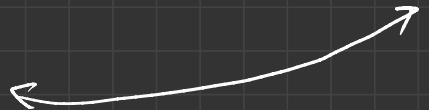
Map2

TC : O(N)

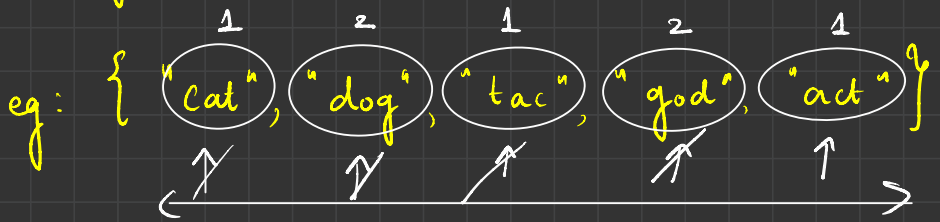
SC : O(26) ≈ O(1)

0 → 26

equals ~ O(26)



group anagrams.



ans → { { cat, tac, act }, { dog, god } }

group1 group2

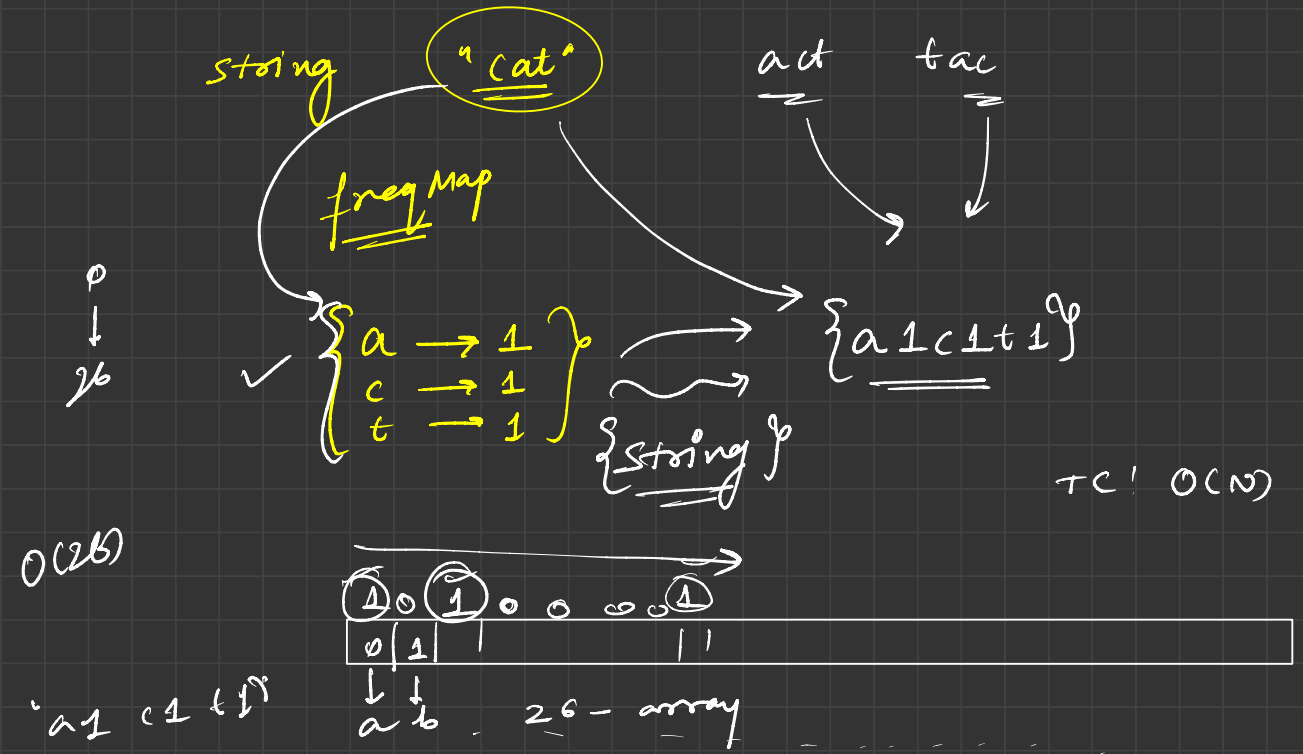
sort (cat) ⇝ act

Map < string, ArrayList<String> >

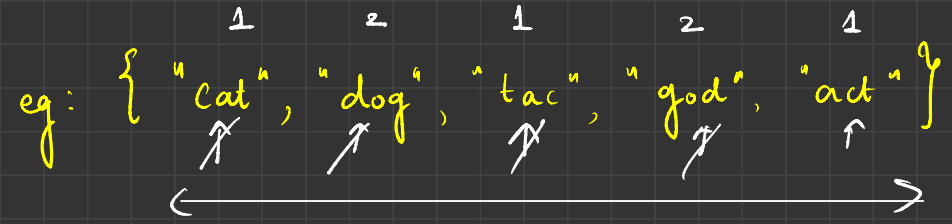
act → { cat, tac, act }
dgo → { dog, god }

TC: M*NlogN ✓

M → No. of strings
N → Length of longest string


string
"cat"
act
tac
freq Map
a → 1
c → 1
t → 1
{a1c1t1}
{string}
TC: O(N)
O(26)
'a1c1t1'
0
1
a
b
26 - array

O(26) < O(N)



ans → { { cat, tac, act }, { dog, god } }

group1 group2

TC: O(M * N)

string	AL
"a1c1t1"	→ { cat, tac, act }
"d1g1o1"	→ { dog, god }

Subarray sum divisible by k.

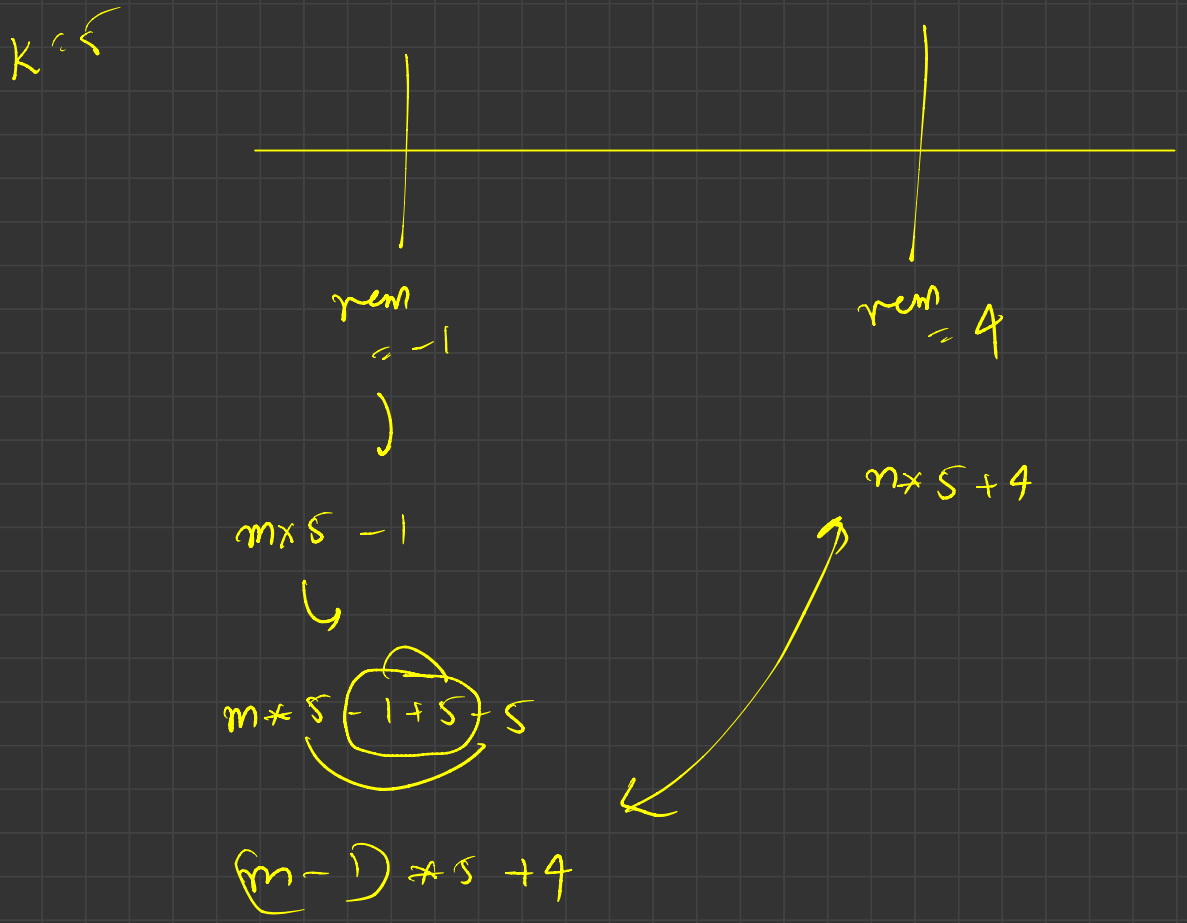
arr[] : {5, 4, 5, 0, -2, -3, 1} k = 5

s=0 s=4 s=9 s=9 s=7 s=4 s=5

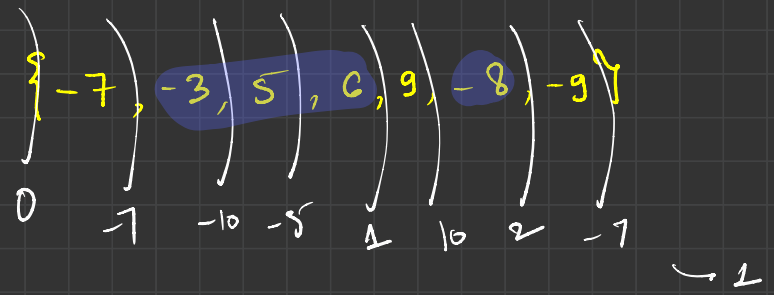
rem	freq
0	~~2~~
4	4
3	1

ans = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 7

{5}
{0}
{5,0}
{0,-2,-3}
{5,0,-2,-3}
{-2,-3}
{4,5,0,-2,-3,1}
=


K = 5
rem = -1
rem = 4
m x 5 - 1
m x 5 + 4
m * 5 - 1 + 5 - 5
(m - 1) * 5 + 4

```java
public static int subarrayDivisbleByK(int arr[], int n, int k){
    // Write code here
    HashMap<Integer, Integer> map = new HashMap<>();
    int runningSum = 0;
    map.put(0, 1);

    int ans = 0;
    for (int i = 0; i < arr.length; i++) {
        runningSum += arr[i];

        int rem = runningSum % k;

        if (rem < 0) {
            rem += k;
        }

        if (map.containsKey(rem)) {
            ans += map.get(rem);
        }

        map.put(rem, map.getOrDefault(rem, 0) + 1);
    }

    return ans;
}
```



K=8

rem

0 → 1

1 → ~~1~~ 2

6 → 1

3 → 1

4 → 1

TC: O(N)

SC: O(K)