Group Anagrams

What?

str1 = "accio" str2 = "oiacc"

if all the characters of a string1 can be rearranged to String2.

How will find anagrams?

sort (str1) → lexographical

same solⁿ → Anagramic

sort (str2)

✓ TC: O(N log N)

SC: O(1)

str1 = " accio "

str2 = " ciaoc "

Map1

charfreqMap

a	$\longrightarrow$ 1
c	$\longrightarrow$ 2
i	$\longrightarrow$ 1
o	$\longrightarrow$ 1

Map2

CharFreqMap

a	$\longrightarrow$ 1
i	$\longrightarrow$ 1
o	$\longrightarrow$ 1
c	$\longrightarrow$ 2

Create Hash Map

Same hashmap or not

TC : $O(N) + O(26) = O(N)$

SC : $O(26) + O(26) = O(1)$

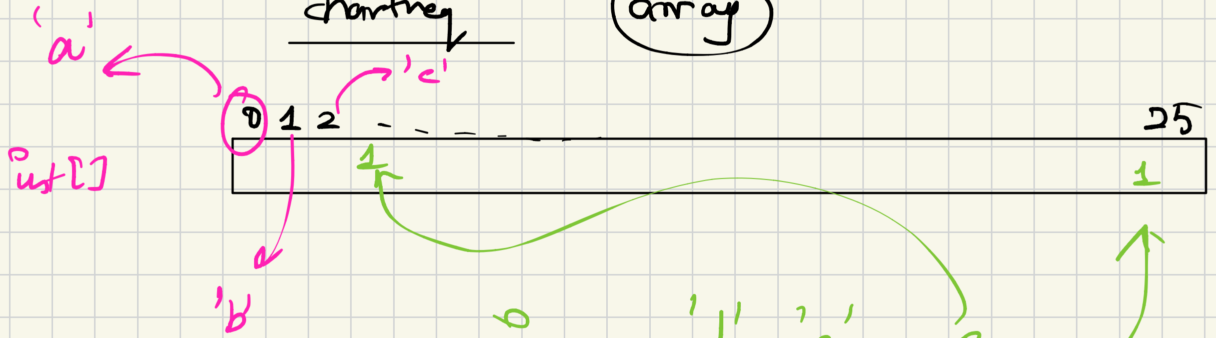
X $\boxed{\text{Map1} == \text{Map2}}$

for (a $\rightarrow$ z)

is freq of Each Char is same in Both

charfreq

array



int pos = 'd' - 'a' = 3

'y' - 'a' = 24

$\boxed{\text{int pos} = (ch - 'a')}$ ✓

TC : O(N)
SC : O(1)

Group Anagrams

string[] words = { "cat", "dog", "tac", "act", "god" }

o/p = { { "cat", "tac", "act" }, { "dog", "god" } }

Brute Force

- TC : $O(N^2 * M)$

length of word array

avg length of word.

SC : $O(N)$

→ keep track of words used.

string[] words = { "cat", "dog", "act", "tac", "god" }

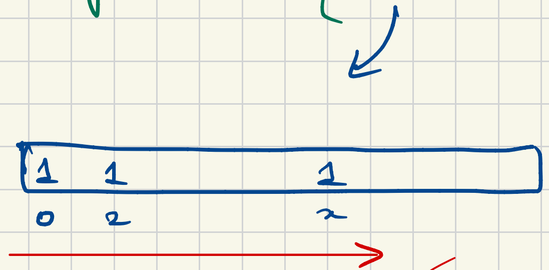
$\boxed{dgo}$

Hashmap

string	ArrayList<string>
common	Group
act	{ "cat", "act", "tac" }
dgo	{ "dog", "god" }

TC : O(N × M logM)
SC : O(N)

string[] word = { "cat", "dog", "tac", "act", "god" }



$O(M) + O(M) = O(M)$

encode = a1c1t1

$TC: O(N*M)$
$SC: O(N)$

Map

encoded	group
a1c1t1	{"cat", "tac", "act"}
d1g1o1	{"dog", "god"}