



Group Anagrams .

What?

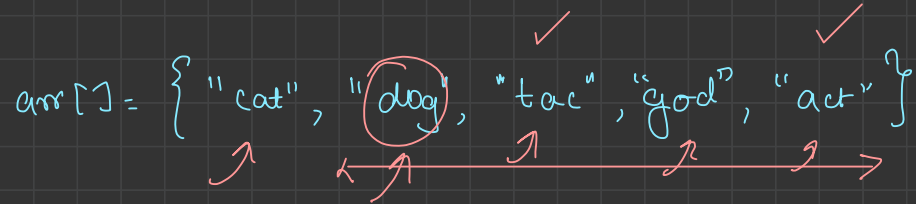
arr[] = { "cat", "dog", "tac", "god", "act" }

{ grp1: cat, tac, act
grp2: dog, god }

o/p cat, tac, act, dog, god ✓

Brute force

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



TC: $O(N^2 \times M)$

Traverse anagramic

SC: $O(26) + O(N) \approx O(N)$

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

TC: $O(N \times M)$



key
grp 1

fmap

code

c → 1
a → 1
t → 1

value

{ "cat", "tac", "act" }

grp 2
fmap

code

d → 1
o → 1
g → 1

{ "dog", "god" }

anagrams.

→ String S1 and S2 are anagrams to each other if freq of each char in both the strings are same.

code

"accio"

→ "a1c2i1o1"

"iocac"

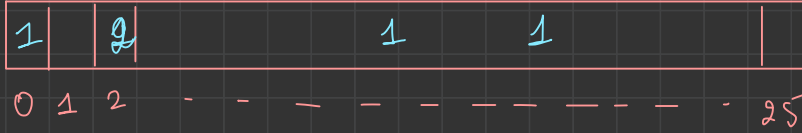
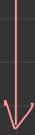
→ "a1c2i1o1"

freq

a → 1
c → 2
i → 1
o → 1

freq?

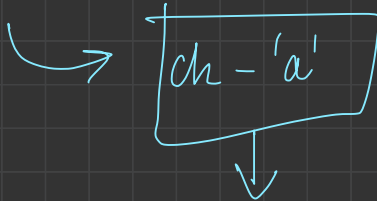
accio



→ char, are sorted
order to create cost?

TC:
 $O(26)$
 $\sim O(1)$

$\left\{ \begin{array}{l} a \longrightarrow 0 \\ b \longrightarrow 1 \end{array} \right.$



relative dist

Minimum Window Substring

{ str1: d b a e c b b a b d c a a f b d d c a b g b a
(N)
str2: a b b c d c
(M)

bruteforce

↳ creating substring of str1
 eg finding str2 all char is that }

TC: $O(N^2 \times N)$

$= O(N^3)$

SC: $O(N)$

str1: d b a e c b b a b d c a a f b d d c a b g b a

etc ↓

↑ inc

str2: a b b c d c

fmap

{ a → 1
b → 2
c → 2
d → 1

useful
char.

max = 6

fmap

d → 1
b → 3
a → 1
c → 1

delcnt = ^pinc
the delcnt when
inc. a useful char
substring(enc+1, ^pinc+1)

del. the when
enc a useful char

