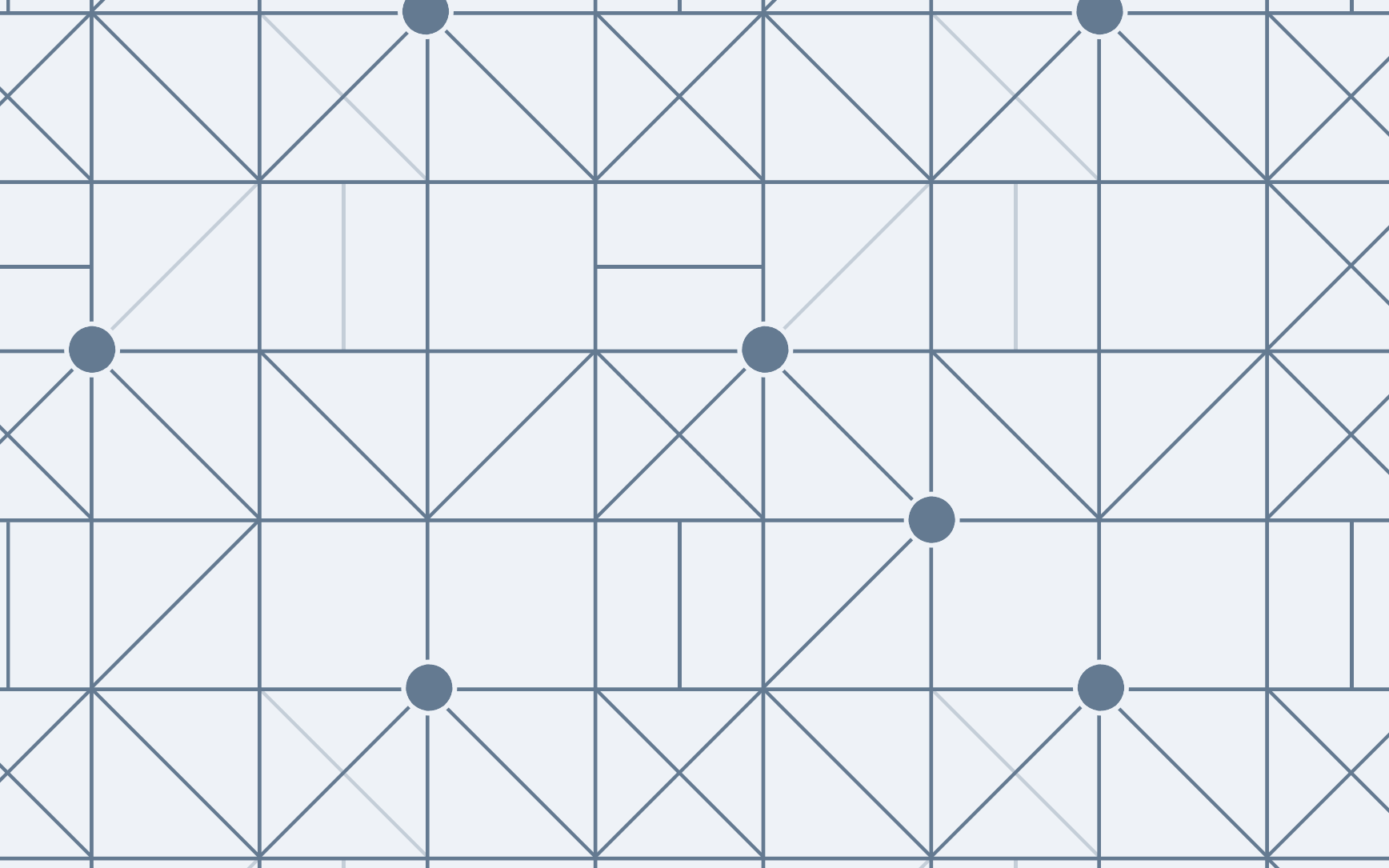


Group Anagrams

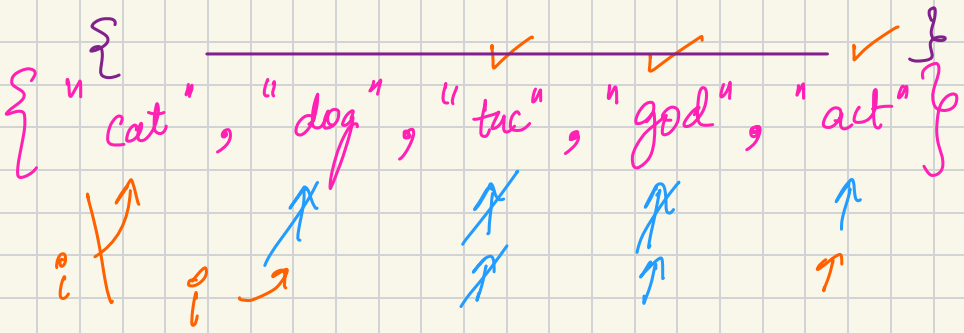
String[] words = {"cat", "dog", "tac", "god", "act"}

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

↳ ans!

str1 & str2

↳ which has equal no. of occ. of each character!



cat = { cat, tac, act }

dog = { dog, god }

size of the array { traversal }

TC: O(N²) × O(M) ≈ TC: O(N²M)

size of the word { anagramic }

SC: O(26) + O(N) ≈ O(N) ✓

hashMap (26 alpha)

Array { is used }

{"cat", "dog", "tac", "god", "act"}

HashMap

(key) Group Name

(value) Group

{c→1, a→1, t→1} cat, tac, act

{d→1, o→1, g→1} dog, god

TC: O(N × MlogM)

O(M)

can't have this kind of keys!

Encoding

c → 1
a → 1
t → 1

"c1a1t1"

char freq → 0

a → 1
c → 1
t → 1

"a1c1t1"

cat

arr[2] += 1
arr[0] += 1
arr[-] += 1

26 – Alphabets

```
int[] arr = new int[26]
```

0th → a
1st → b
|
|
|
.
25th → z

str = "abade"

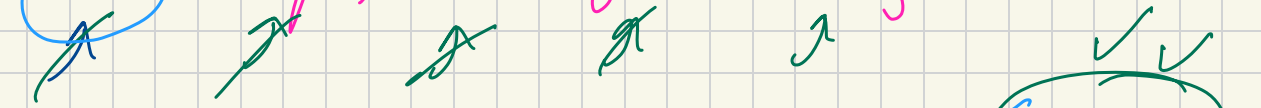
```
arr[ch - 'a'] += 1
```

arr[0] += 1
arr[1] += 1
arr[0] += 1
arr[3] += 1
arr[4] += 1

[2, 1, 0, 1, 1, ...]

→ character in sorted order

{"cat", "dog", "tac", "god", "act"}



<string> grpName

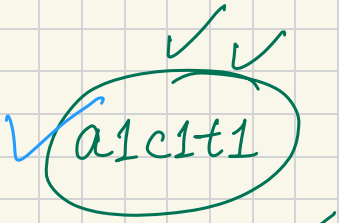
AL<string> grp

a1c1t1

d101g1

{cat, tac, act}

{dog, god}



TC: O(N) * O(M) * O(26) ≈ O(N*M)

Minimum Window Substring.

size : N

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

size : M

str2 : a b b c d c

Brute force

↳ generate all the substrings of str1

and try to find all characters of string 2 :

{ TC : $O(N^2 * M)$ SC : $O(N) + O(M) \sim O(N+M)$ ✓

size : N

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

exc ↓

inc ↑

size : M

str2 : a b b c d c → mcnt = 6

len = inc - exc

= 7

freqMap2

a	1
b	2
c	2
d	1

freqMap1

d	~~1~~ 2
b	1
a	~~1~~ 2
c	~~1~~ 2
f	1