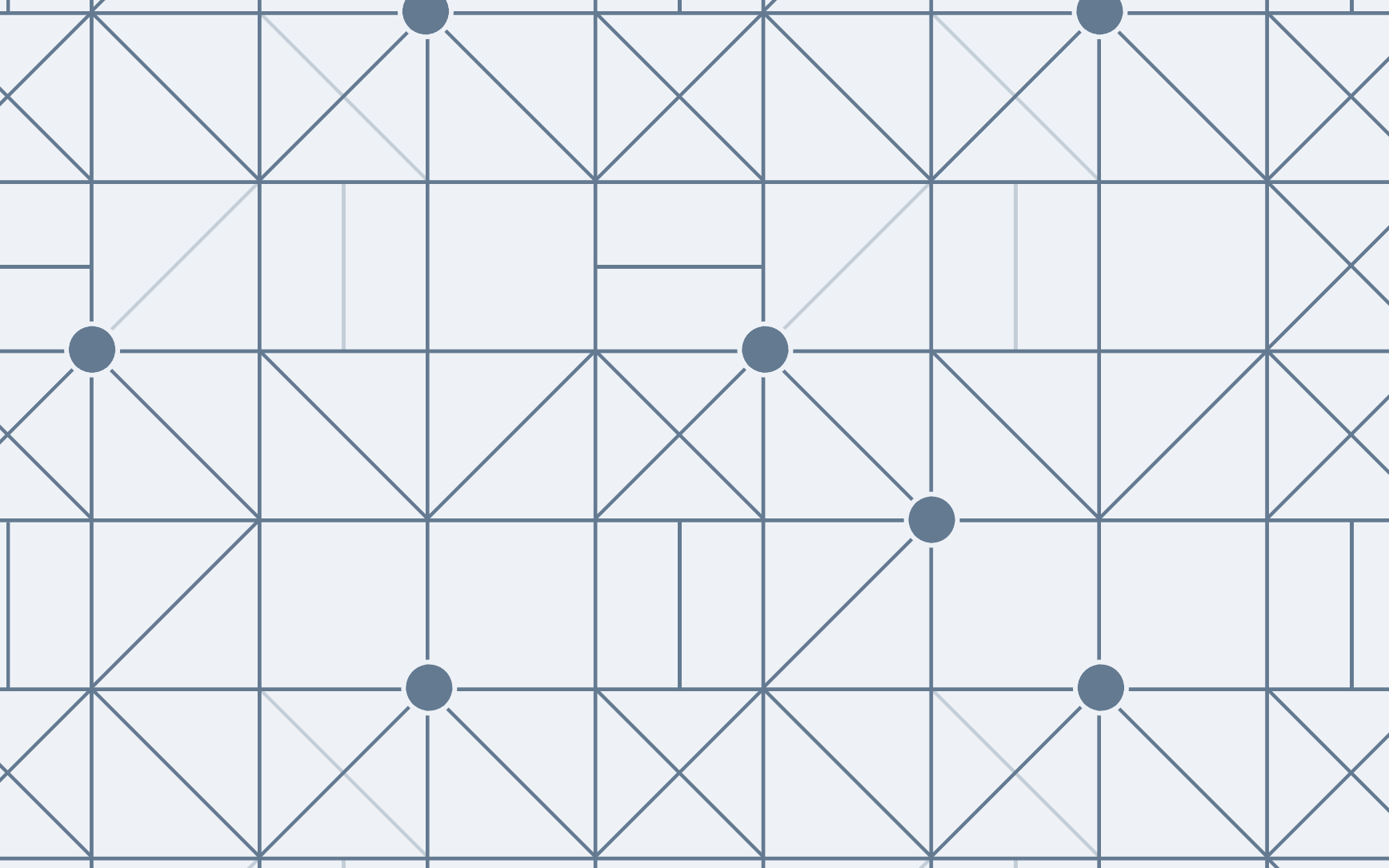


Count Number of Pairs with absolute diff. k

int[] arr = {5, 10, 25, 15, 20, 25} k = 5

(x, y)

y = 5
x = y + k = 10
  = y - k = 0

y = 10
x = y + k = 15
  = y - k = 5

y = 25
x = 30
x = 20

y = 25
x = 30
x = 20

y = 15
x = 20
x = 10

y = 20
x = 25
x = 15

(5, 10) (10, 15) (25, 20) (15, 20) (20, 25)

num	freq
20	1
5	1
10	1
25	1
15	1

count = 0 1 2 3 5

int[] arr = { 1, 2, 1, 2, 2, 2, 3 } k = 0

{ (1,1) (2,2) }

freq

k = 0

→ 1 ⟶ 2

→ 2 ⟶ 4

3 ⟶ 1

y = 1 }
x = 1

y = 2 }
x = 2

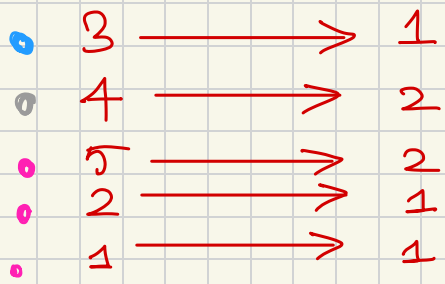
$\{3, 4, 5, 2, 1, 4, 5\}$ $k = 1$

(3, 4)

(4, 5)

(2, 3)

(1, 2)



(x, y)

$y = 1$

$x = 2$

$y = 2$

$x = 3$

$y = 3$

$x = y + k = 4$

count = ~~0 1 2 3~~ 4

$y = 4$

$x = 5$

$y = 5$

$x = 6$

Roll Number Problem

Agenda

1. group anagrams
2. Minimum window substring
3. Distinct windows
4. Substring with K unique characters.

Group Anagrams.

String[] = { act, tac, god, dog, cat }
0 1 2 3 4

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

Brute force

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

SC: $O(1)$

M → length of string.

N → length of array.

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

0 1 2 3 4

Hash Map

TC : O(N x N x M)

SC : O(N)

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

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

0 1 2 3 4

Hash Map

sort(str)

Key	Value
grp_name	list of group
act	{act, tac, cat}
dgo	{god, dog}

dgo act

TC: O(N x M x $\log_2$ M)

SC: O(N)

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

0 1 2 3 4

10K 12K 13K

a → 1
c → 1
t → 1

g → 1
o → 1
d → 1

Hash Map

act, tac, cat

fmap

a → 1
c → 1
t → 1

Key Value

10K {act}

{act, tac, cat}

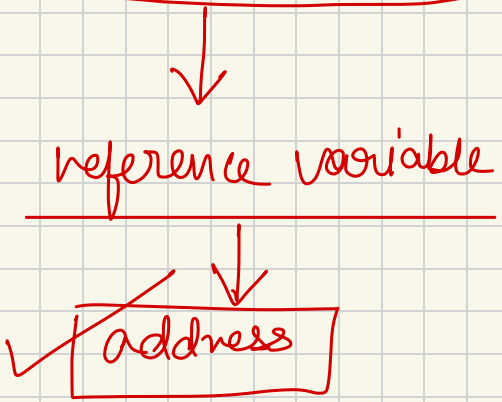
{tac}

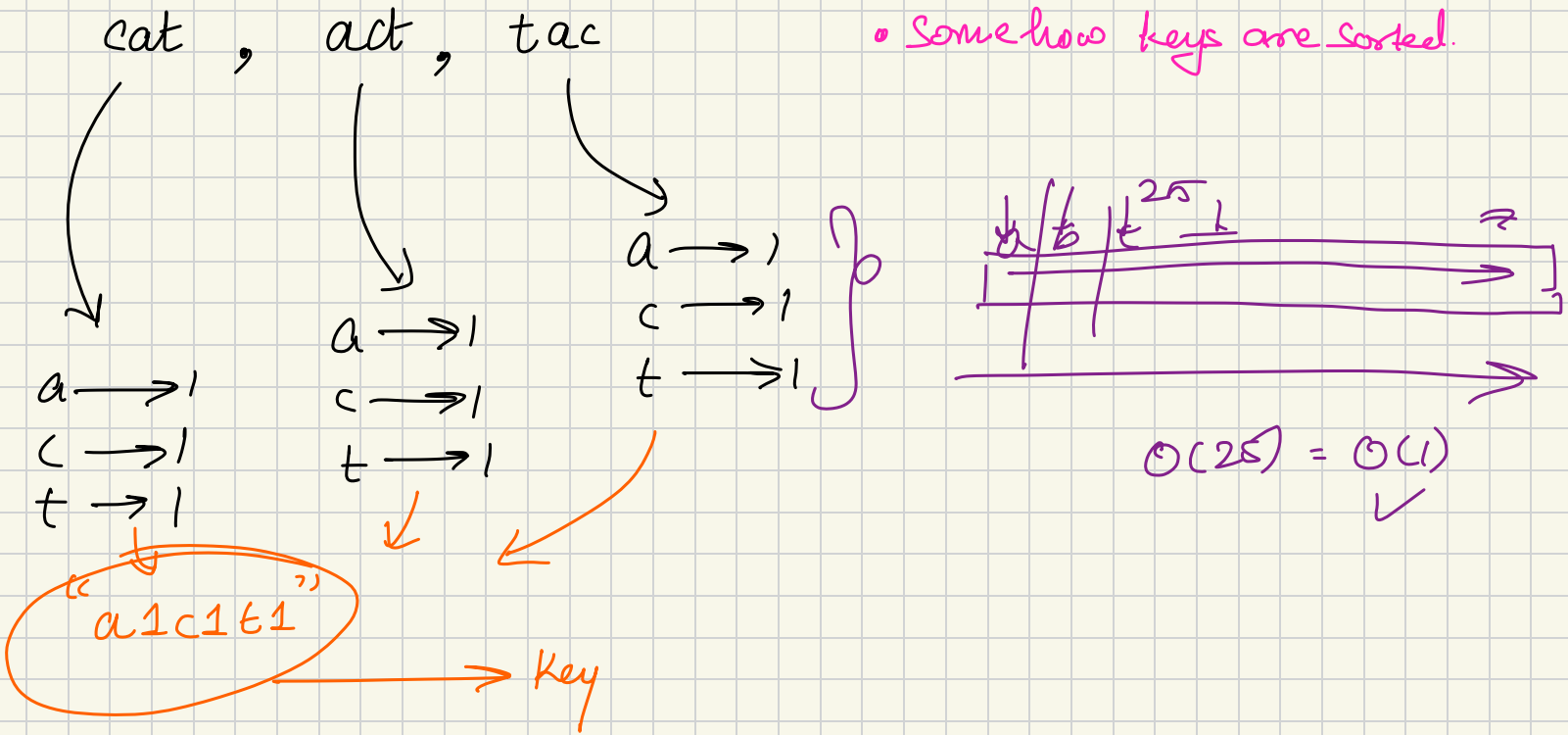
12K

{god, dog}

TC: O(N×M) SC: O(N)

HashMap < HashMap < Character, Integer > , ArrayList < String >>





- Somehow keys are sorted.

$O(25) = O(1)$

Minimum window substring

str1 = " d b e a c d b c c a b "

str2 = " a b b c d c "

a → 1
b → 2
c → 2
d → 1

Brute force

↳ generate map of all substring of str1 & try to accumulate str2 in it.

TC : O(N² × M)
SC : O(1)

N → len of str1
M → len of str2

str1 = " d b e a c d b c c a b "

exc ↓ (d)

inc ↑ (b)

str1.substring(exc+1, inc+1)

str2 = " a b b c d c "

TC : O(N+M)

SC : O(1)

ans = ~~b e a c d b c~~

d b c c a b

fmap2

a → 1
b → 2
c → 2
d → 1

Contained ?

fmap1

b → ~~2~~ 1
a → 1
c → ~~2~~ 1

str1 = " d b e a c d b c c a b "

inc ↓

exc ↑

str2 = " a b b c d c "

{ TC: O(N+M)
SC: O(1)

dmct = 6

mct = ~~0 1 2 3 4~~ 5

fmap 2

a → 1
c → 2
b → 2
d → 1

fmap 1

b → ~~1~~ 2

a → 1
c → ~~1~~ 2

H/W

- Smallest Substring with all characters
- Maximum Ones after modification