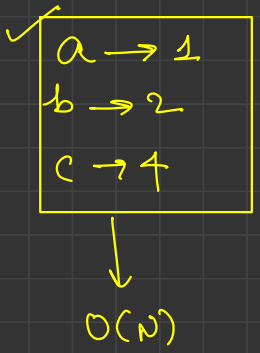
Group Anagrams

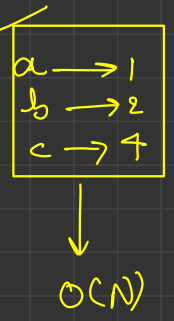
eg: ["cat", "dog", "tac", "god", "act"]

Anagrams?



s1 → freq of each character ==

s2 → freq of each character



TC: O(N)

sort(s1) == sort(s2) TC: O(NlogN)

eg: ["cat", "dog", "tac", "god", "act"]

dgo

Key → value

{a→1, c→1, t→1} ⟶ ① cat, tac, act

{d→1, g→1, o→1} ⟶ ② dog, god, dgo

HashMap<String, ArrayList<Strings>> map

code → {a1c1t1} → group 1

{d1g1o1} → group 2

StringBuilder

class in Java,

allows to create mutable strings.

```
StringBuilder sb = new StringBuilder();
```

sb.append('') → O(1)

sb.toString() → String

```java
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        HashMap<String, ArrayList<String>> map = new HashMap<>();

        for (int i = 0; i < strs.length; i++) {
            String str = strs[i];

            // Step 1 -> generate freq array
            int[] freq = new int[26];
            for (int j = 0; j < str.length(); j++) {
                char ch = str.charAt(j);
                freq[ch - 'a'] += 1;
            }

            // Step 2 -> generate code from freq array
            StringBuilder code = new StringBuilder("");
            for (int j = 0; j < 26; j++) {
                if (freq[j] > 0) {
                    code.append((char)('a' + j));
                    code.append(freq[j]);
                }
            }

            // Step 3 -> if code is seen first time, initialize a new group
            if (map.containsKey(code.toString()) == false) {
                ArrayList<String> t = new ArrayList<>();
                map.put(code.toString(), t);
            }

            ArrayList<String> list = map.get(code.toString());
            list.add(str);
            map.put(code.toString(), list);
        }

        List<List<String>> ans = new ArrayList<>();
        for (String key : map.keySet()) {
            ArrayList<String> grp = map.get(key);
            ans.add(grp);
        }

        return ans;
    }
}
```

O(N) → size of String Array

O(M) → Max size of string

O(1)

2 char('a'+2) → 'c'1

O(1)

ArrayList<ArrayList

["cat", "dog", "tac", "god", "act"]

a → 1, c → 1, t → 1

a1c1t1

code	group
a1c1t1	{cat, tac}
d1g1o1	{dog}
—	{-}

TC: O(N x M)

Subarray Sum divisible by K

arr[] : [4, 5, 0, -2, -3, 1] K = 5

sum	4	9	9	7	4	5
rem	0_1	4_1	4_2	4_3	2_1	4_4	0_2

rem = (-)ve

rem += K

ans = 1 + 2 + 3 + 1 = 7

Number = (5n - 3) % 5

= -3

5n - 3 + 5 - 5

K + rem

5(n-1) + (5 - 3)

→ (5(n') + 2) % 5 = 2

Minimum Window Substring

d 2 b 3 a 4
f 1 c 1 g 1

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

exc

inc

mcnt

str2: a b b c d c

a	1
b	2
c	2
d	1

→ 6 → dmant

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

c b b a b d c

```java
HashMap<Character, Integer> freq = new HashMap<>();
for (int i = 0; i < t.length(); i++) {
    char ch = t.charAt(i);
    freq.put(ch, freq.getOrDefault(ch, defaultValue: 0) + 1);
}

String ans = "";
int dmcnt = t.length();
int mcnt = 0;
HashMap<Character, Integer> map = new HashMap<>();
int inc = -1;
int exc = -1;

while (true) {
    boolean f1 = false;
    boolean f2 = false;

    // inc
    while (inc < s.length() - 1 && mcnt < dmcnt) {
        inc++;
        char ch = s.charAt(inc);
        map.put(ch, map.getOrDefault(ch, defaultValue: 0) + 1);

        if (map.get(ch) <= freq.getOrDefault(ch, defaultValue: 0)) {
            mcnt++;
        }

        f1 = true;
    }

    // exc
    while (exc < inc && mcnt == dmcnt) {
        String currAns = s.substring(exc + 1, inc + 1);
        if (ans.length() == 0 || ans.length() > currAns.length()) {
            ans = currAns;
        }

        exc++;
        char ch = s.charAt(exc);
        map.put(ch, map.getOrDefault(ch, defaultValue: 0) - 1);

        if (freq.containsKey(ch) && freq.get(ch) > map.get(ch)) {
            mcnt--;
        }

        if (map.get(ch) == 0) {
            map.remove(ch);
        }

        f2 = true;
    }

    if (f1 == false && f2 == false) {
        break;
    }
}
```

s t

SC : O(N+M)

TC : O(N+M)

exc

s= d b a e c b b a b d c a a f b d d c a b g b a

inc

t= a b b c d c

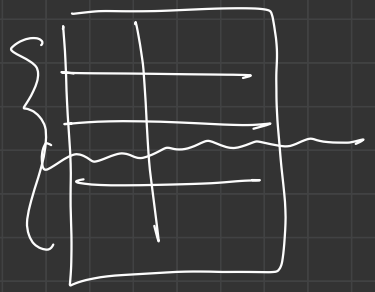
a	1
b	2
c	2
d	1

dmcnt = 6

mcnt = 0 1 2 3 4 5 6 5 6

d	3
b	3
a	3
f	1
c	2

a b a cbcdbca



abcd