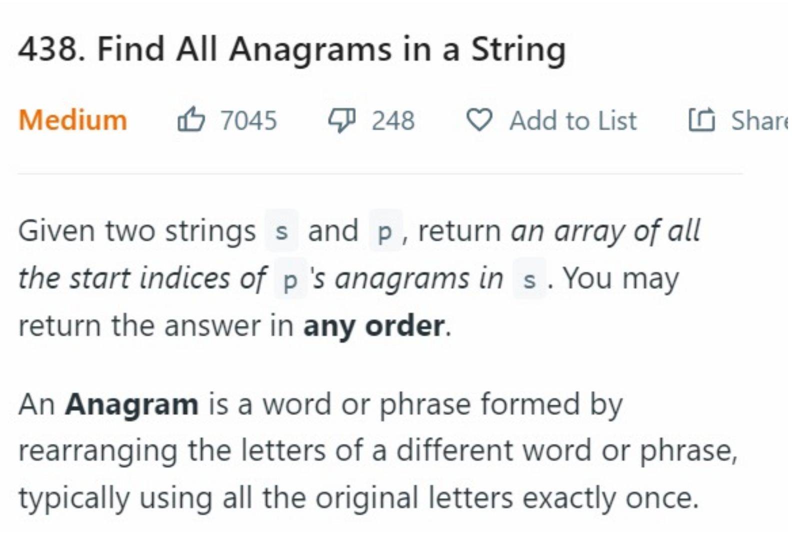


438. Find All Anagrams in a String

Medium 7045 248 Add to List Share

Given two strings `s` and `p`, return *an array of all the start indices of* `p`*'s anagrams in* `s`. You may return the answer in **any order**.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

s = a b c a d b a c

0 1 2 3 4 5 6 7

p = abc

ans : [0, 1, 5]

a b c a d b a c

0, 1, 5

fixed window size

j

s = a_0 b_1 c_2 a_3 d_4 b_5 a_6 c_7

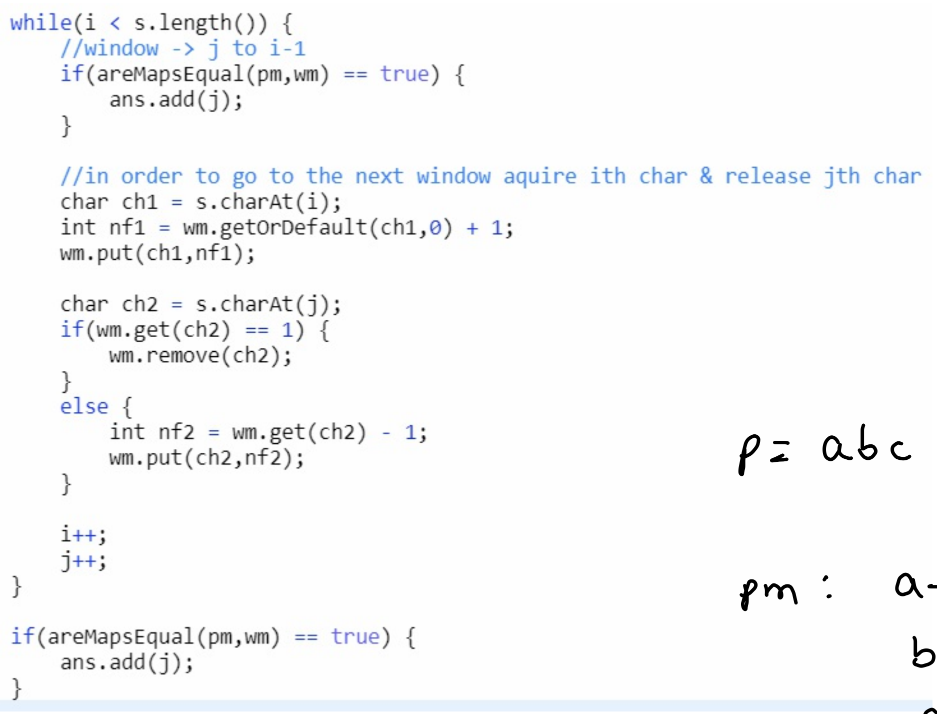
i

p = abc

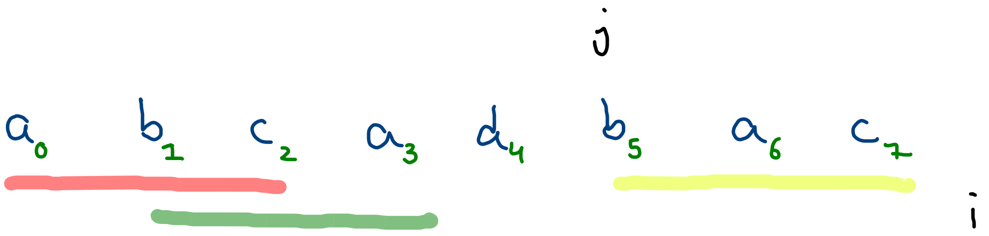
pm : a - 1
b - 1
c - 1

wm :
c - 1
b - 1
a - 1

ans = [0, 1, 5]



```
while(i < s.length()) {
    //window -> j to i-1
    if(areMapsEqual(pm,wm) == true) {
        ans.add(j);
    }

    //in order to go to the next window aquire ith char & release jth char
    char ch1 = s.charAt(i);
    int nf1 = wm.getOrDefault(ch1,0) + 1;
    wm.put(ch1,nf1);

    char ch2 = s.charAt(j);
    if(wm.get(ch2) == 1) {
        wm.remove(ch2);
    }
    else {
        int nf2 = wm.get(ch2) - 1;
        wm.put(ch2,nf2);
    }

    i++;
    j++;
}

if(areMapsEqual(pm,wm) == true) {
    ans.add(j);
}
```



p = abc

pm : a-1
b-1
c-1

wm : a-1
b-1
c-1

ans = [0, 1, 5]

longest substring with exactly K distinct char

a b a c d e d e a c   K = 3

① · aquire : while (map· size () <= K)

② · release : while (map· size() > K)

j

a_0 b_1 a_2 c_3 d_4 e_5 d_6 e_7 a_8 c_9

i

map

a - 1
e - 1
c - 1

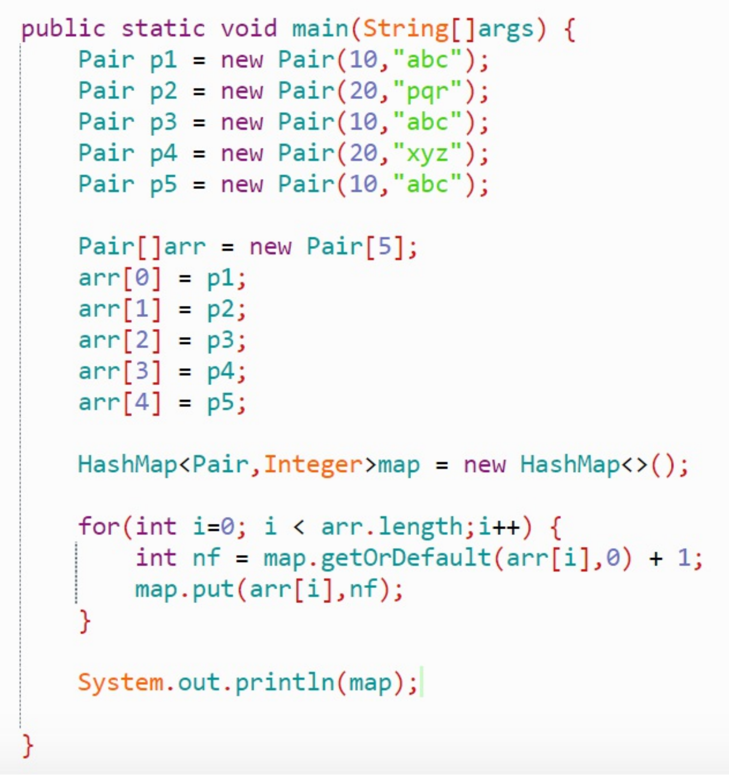
olen = ~~4~~ 5

Count Of Substrings With Exactly K Unique Characters

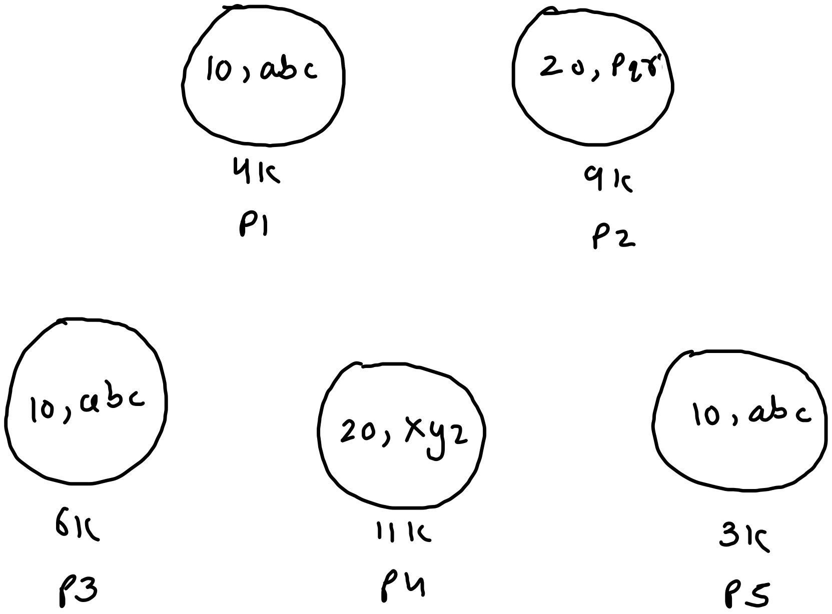
$K = 3$

$$a_0 \quad b_1 \quad a_2 \quad c_3 \quad d_4 \quad e_5 \quad d_6 \quad e_7 \quad a_8 \quad c_9$$

count of ss with exactly k distinct chars = count of ss with atmost k distinct chars − count of ss with atmost (k-1) distinct chars



```
public static void main(String[]args) {
    Pair p1 = new Pair(10,"abc");
    Pair p2 = new Pair(20,"pqr");
    Pair p3 = new Pair(10,"abc");
    Pair p4 = new Pair(20,"xyz");
    Pair p5 = new Pair(10,"abc");

    Pair[]arr = new Pair[5];
    arr[0] = p1;
    arr[1] = p2;
    arr[2] = p3;
    arr[3] = p4;
    arr[4] = p5;

    HashMap<Pair,Integer>map = new HashMap<>();

    for(int i=0; i < arr.length;i++) {
        int nf = map.getOrDefault(arr[i],0) + 1;
        map.put(arr[i],nf);
    }

    System.out.println(map);

}
```



4k -> 1

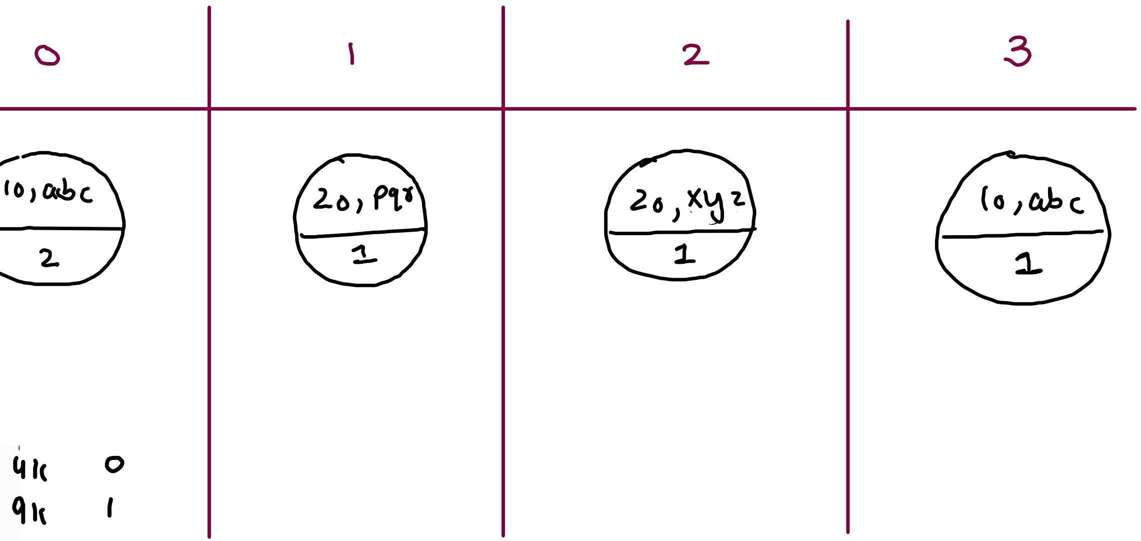
9k -> 1

6k -> 1

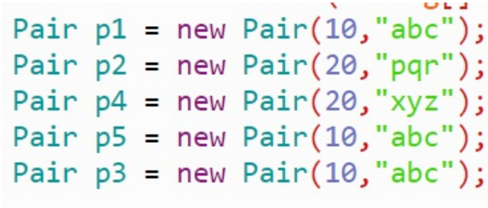
11k -> 1

3k -> 1

Hashmap :   LinkedList < HMNode > [ ] buckets;



```
HMNode {
  K key;
  V value;
}
```



```
Pair p1 = new Pair(10,"abc");
Pair p2 = new Pair(20,"pqr");
Pair p4 = new Pair(20,"xyz");
Pair p5 = new Pair(10,"abc");
Pair p3 = new Pair(10,"abc");
```

4k 0
9k 1
6k 2
11k 3
3k 0

bi