Date: 9th January 2022

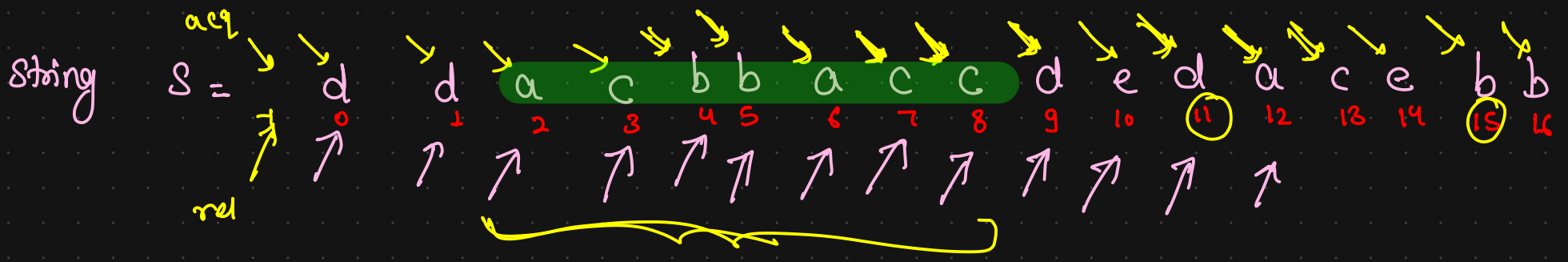
Agenda

- Longest Substring With At Most K Unique Characters
- Count Of Substrings Having At Most K Unique Characters
- Valid Anagram
- Find Anagram Mappings
- Find All Anagrams In A String

Longest Substring With At Most K Unique Characters

map.size() ≤ k

k = 3



HashMap < character, Integer >

~~d → 1 2 1 0~~

~~a → 1 2 1 0~~

~~c → 1 2 2 2 1 0~~

~~b → 1 2 1 0~~

~~d → 1 2 1 0~~

c → 1

e → 1

b → 1 2

~~e → 1 0~~

~~a → 1 0~~

len = ~~0 1 2 3 4 5~~ 6 7

acq → ① acquire until unique characters are less than equal to k.

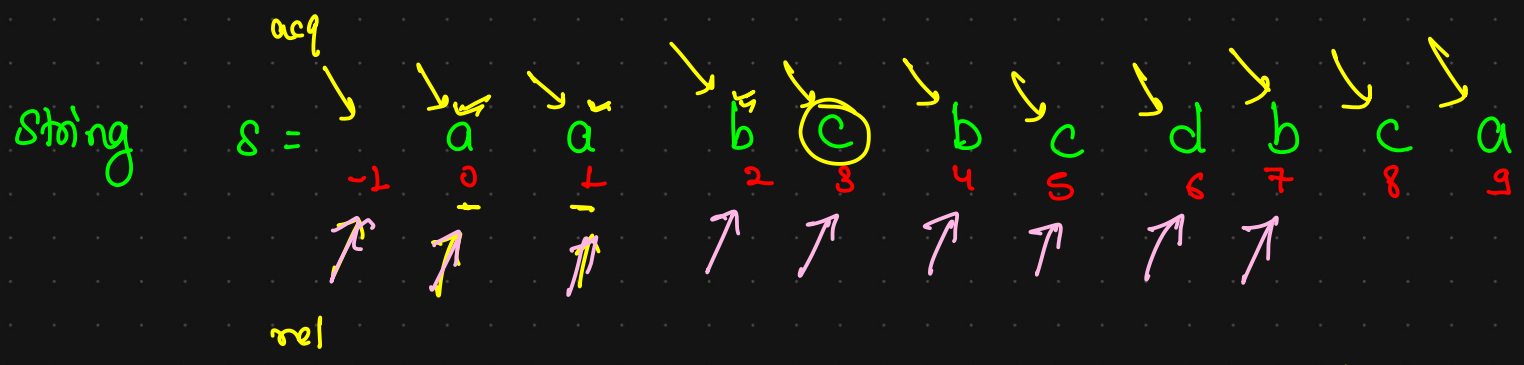
② increase length of result by acq;

rel → ① if unique characters will be more than k, rel. until character become ≤ k again,

Count Of Substrings Having At Most K Unique Characters

character → 1, 2, ..., k } unique

k = 2



HashMap <character, Integer>

~~a → 1 2 1 0~~

~~b → 1 2 1 0~~

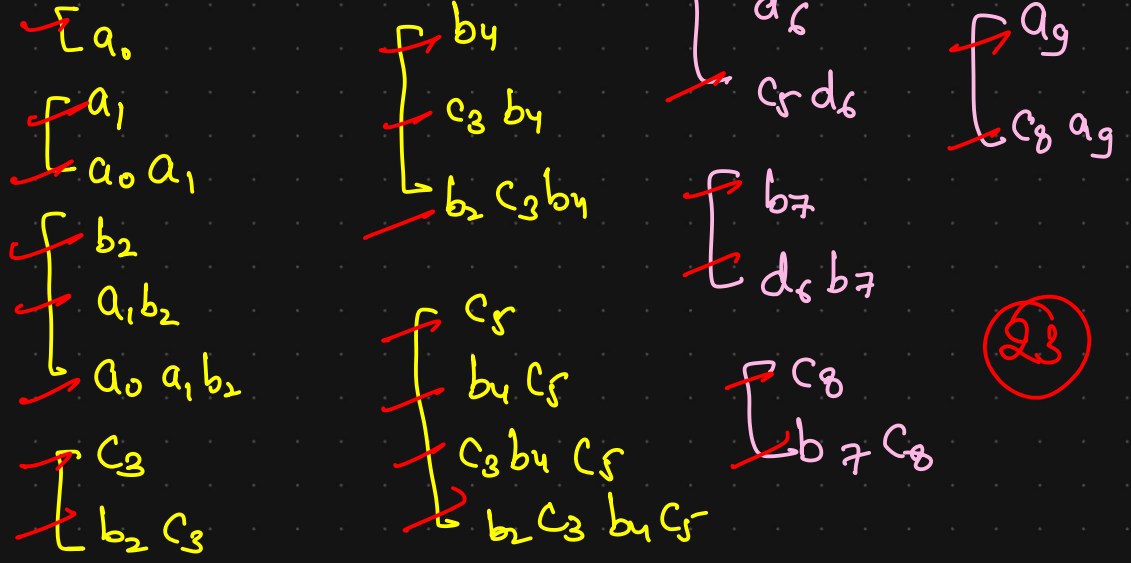
~~c → 1 2 1 0~~

~~d → 1 0~~

~~b → 1 0~~

c → 1

a → 1

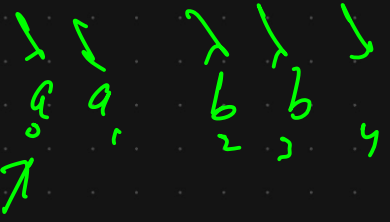


23

$k=2$

$a \to X\ X\ b\ 2$

$b \to 1$



$a \to X\ X\ 1$

$b \to 1$

a_0

a_1

$a_0\ a_1$

a_2

$a_1\ a_2$

$a_0\ a_1\ a_2$

b_3

$a_2\ b_3$

$a_1\ a_2\ b_3$

$a_0\ a_1\ a_2\ b_3$

b_4

$b_3\ b_4$

$a_2\ b_3\ b_4$

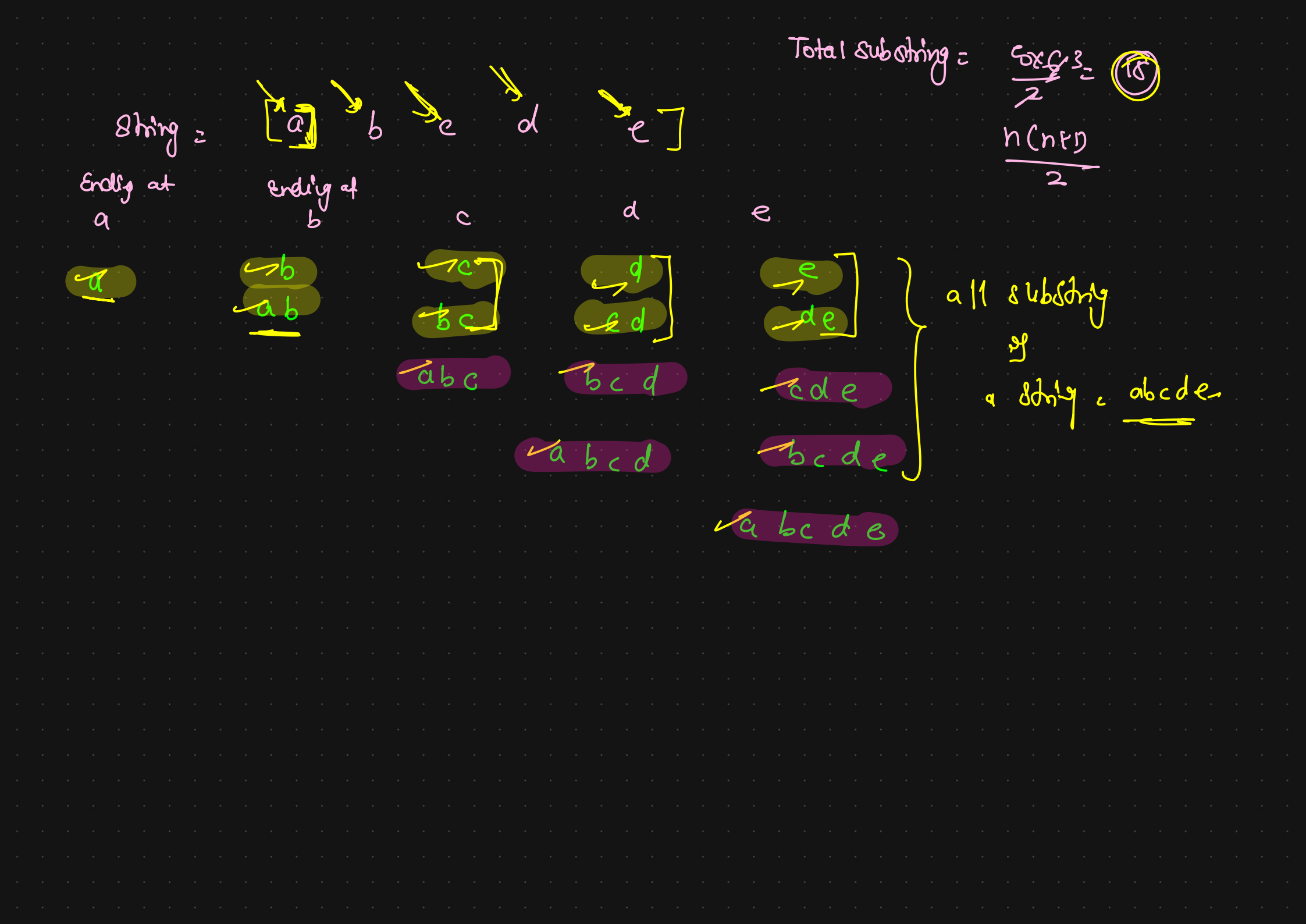
$a_1\ a_2\ b_3\ b_4$

$a_0\ a_1\ a_2\ b_3\ b_4$

$\begin{cases} a_0 \\ \begin{cases} a_1 \\ a_0\ a_1 \end{cases} \end{cases}$

$\begin{cases} b_2 \\ a_1\ b_2 \\ a_0\ a_1\ b_2 \end{cases}$

$\begin{cases} b_3 \\ b_2\ b_3 \\ a_1\ b_2\ b_3 \\ a_0\ a_1\ b_2\ b_3 \end{cases}$

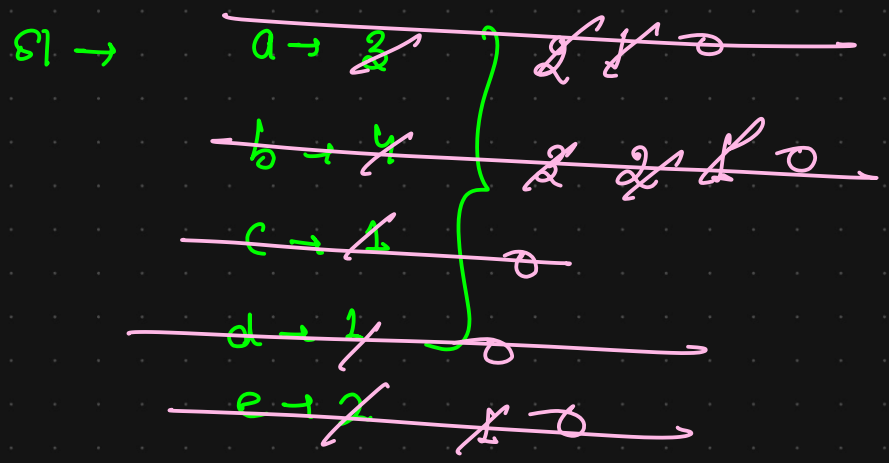

Total Substring = 5x6/2 = 15
n(n+1)/2
String = a b c d e
Ending at a
Ending at b
c
d
e
a
b
ab
c
bc
abc
d
cd
bcd
abcd
e
de
cde
bcde
abcde
all substring of string : abcde

Valid Anagram

String s1 → a a b b a e c d b e b

String s2 → a b b c b e b c a a d

Anagram : freqwy map of String s1 should be same for freq. map of s2.

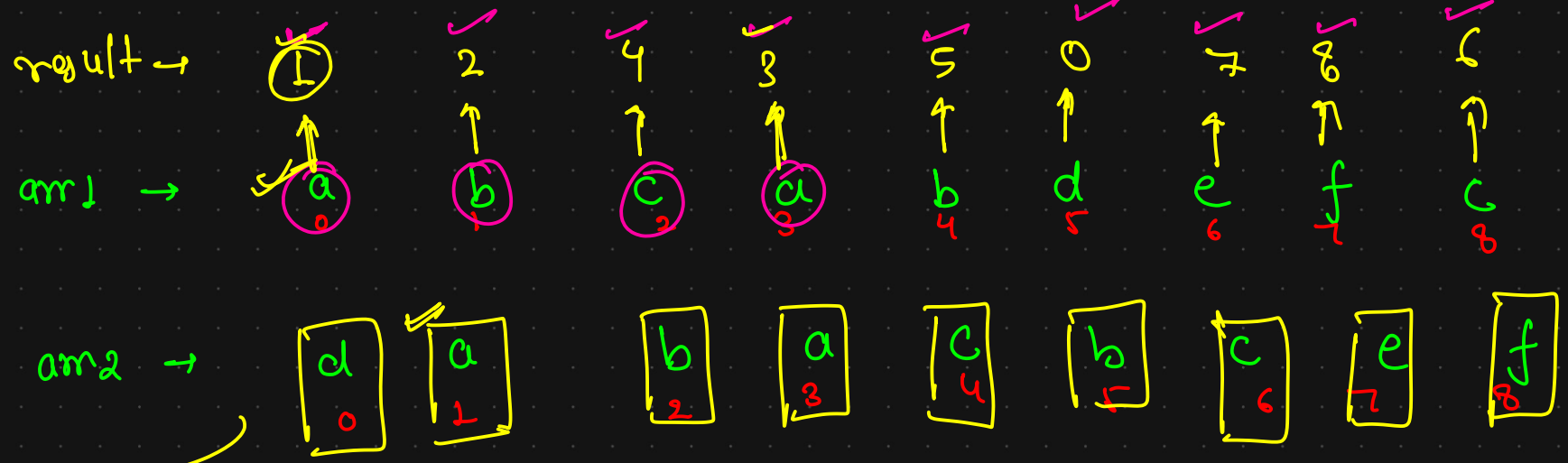


ans = map.size() == 0

if true = anagram

false = no anagram

Find Anagram Mappings



Integer vs Linked of Integers

value

d → ~~0~~

a → ~~1~~ → ~~3~~

b → ~~2~~ → ~~5~~

c → ~~4~~ → ~~6~~

e → ~~7~~

f → ~~8~~

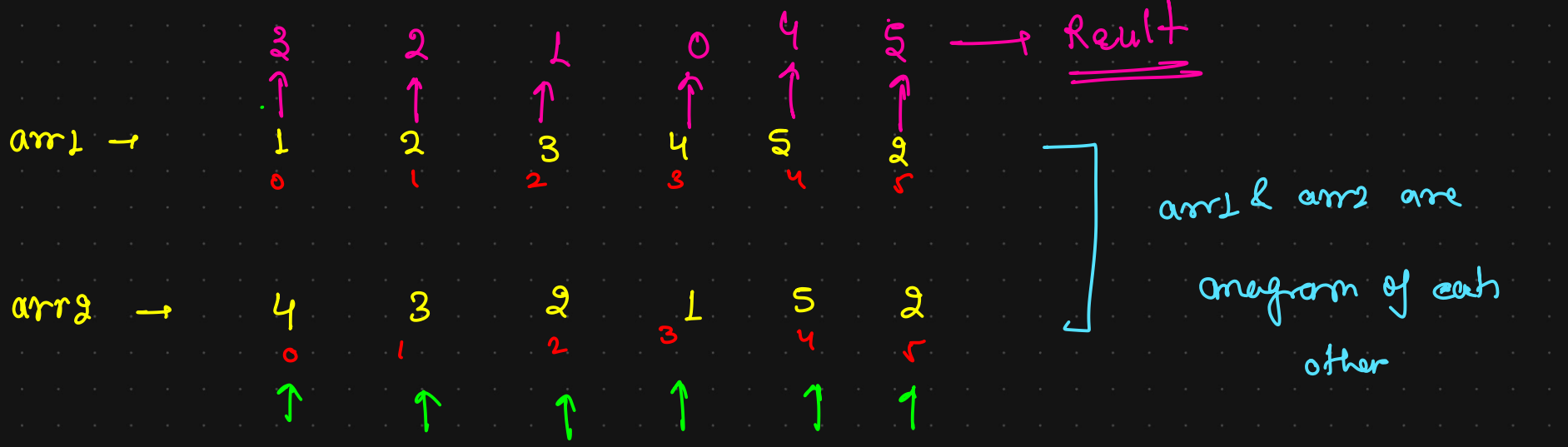
Integer → key → hold the element of array

2-

we know that element may be repeat so, we hold linked list for multiple index of same element

benefit of LinkedList → remove last / addLast

↳ O(1) =



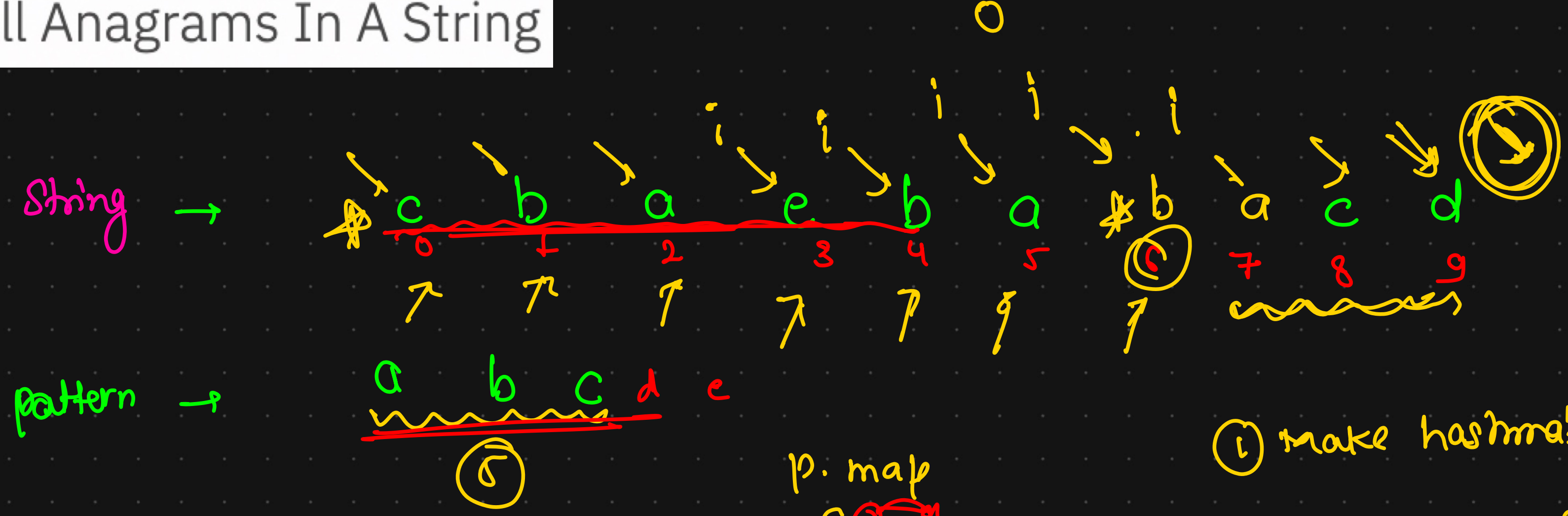
HashMap<Integer, LinkedList<Int.>> map

index of arr1[i] in arr2 if there are multiple occurences of arr1[i] in arr2, then return in specific order.

4 → 0
3 → 1
2 → 2, 5
1 → 3
5 → 4

```java
public static int[] anagramMappings(int[] arr1, int[] arr2) {
    HashMap<Integer, LinkedList<Integer>> map = new HashMap<>();
    for(int i = 0; i < arr2.length; i++) {
        int val = arr2[i];
        if(map.containsKey(val) == true) {
            map.get(val).addLast(i);
        } else {
            LinkedList<Integer> list = new LinkedList<>();
            list.add(i);
            map.put(val, list);
        }
    }
    int[] res = new int[arr1.length];
    for(int i = 0; i < arr1.length; i++) {
        int val = arr1[i];
        int indx = map.get(val).removeFirst();
        res[i] = indx;
    }
    return res;
}
```

Find All Anagrams In A String



c → 1 0
b → 1 0
a → 1
e → 1 0
c → 1
d → 1

p. map
a → 1
b → 1
c → 1

① make hashmap from pat.
② acq. patt. length from str.
③ match s.map & p.map
acq & rel

loop
check → match → store starting index
↳ s. index i − patt. length
acq → i
rel → i − patt. length

match first window