763. Partition Labels

21 March 2022 03:19 PM

You are given a string `s`. We want to partition the string into as many parts as possible so that each letter appears in at most one part.

Note that the partition is done so that after concatenating all the parts in order, the resultant string should be `s`.

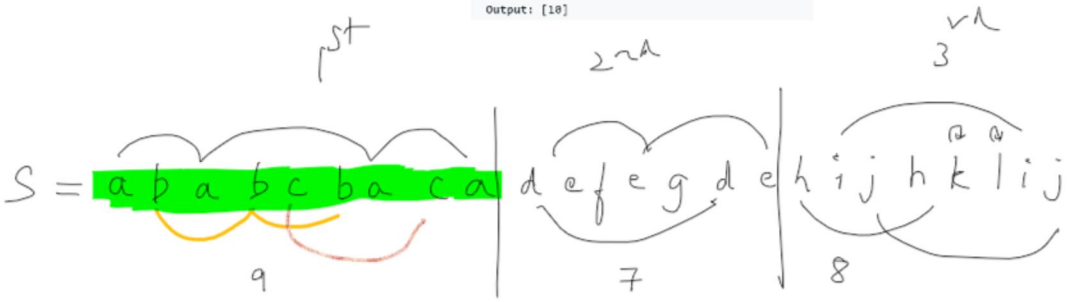
Return *a list of integers representing the size of these parts.*

Example 1:

```
Input: s = "ababcbacadefegdehijhklij"
Output: [9,7,8]
Explanation:
The partition is "ababcbaca", "defegde",
"hijhklij".
This is a partition so that each letter
appears in at most one part.
A partition like "ababcbacadefegde",
"hijhklij" is incorrect, because it splits s
into less parts.
```

Example 2:

```
Input: s = "eccbbbbdec"
Output: [10]
```



No of partition ⇒ 3

single partition ⟶ char are not present beyond its partition

prev = -1

max = 0 8

↳ index of the char

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	b	c	b	a	c	a	d	e	f	e	g	d	e	h	i	j	h	k	l	i	j

Character Maximum Impact (Index) ⟶

Hashmap — (Character, Integer) ← Index

Step1: Make Hashmap and maintain the index of last occurance of character.

a → 0 2 6 8

b → 1 3 5

c → 4 7

d - 9 13 14

e - 10 12 15

f - 11 16

i = 17 22

k = 20

j = 18 23

l = 21

h = 19

Step2 : * find the max impact here for a i.e. 8.

and keep moving the character if the max is more than 8 then

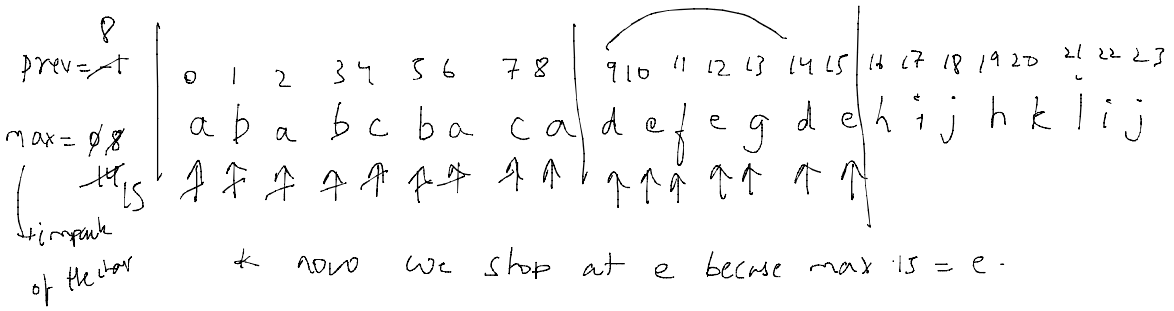
update it else keep moving, if the max reach at char of 8 then stop it's a one partition. So merge the partition.

(max − prev = res.add) as we need length of the partition we did prev = −1

8 − (−1) = res.add

q = res.add → one ans

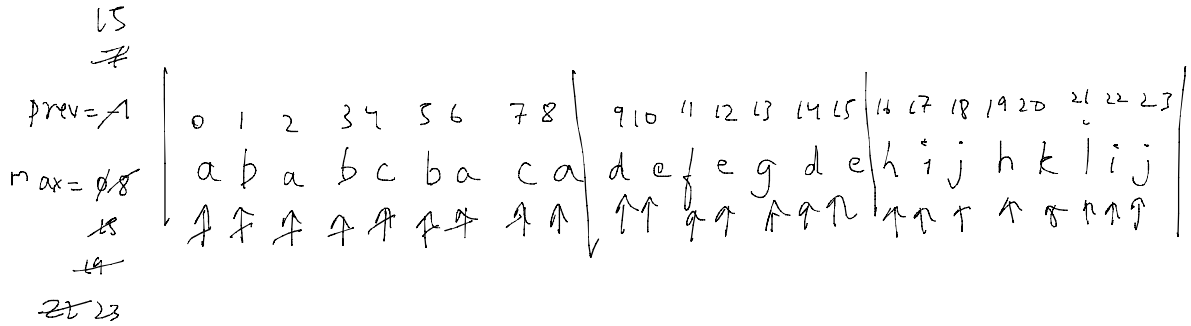
now update the prev to 8 -



* now we stop at e because max is = e.

max − prev

15 − 8 = 7 → 2nd ans



* Update the prev to 15.

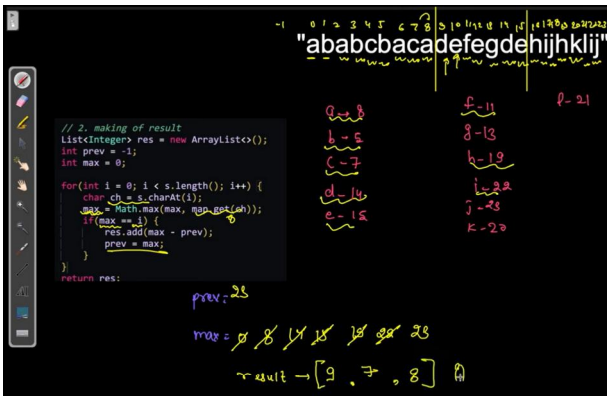
max − prev = 25 − 15 = 7 → 3rd ans. add

```
class Solution {
    public List<Integer> partitionLabels(String s) {
        HashMap<Character, Integer> map = new HashMap<>();

        //step 1. filling of impact of character
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            map.put(ch, i); //if its available, we update it  else it create new  one and add it.
        }
```

```
        // step 2. making of result
        List<Integer> res = new ArrayList<>();
        int prev = -1;
        int max = 0;

        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            max = Math.max(max, map.get(ch));
            if(max == i) { // at this moment we make partition
                res.add(max - prev);
                prev = max;     // update the prev
            }
        }
    return res;
    }
}
```



Dry Run

Watch 769, 768 leetcode.