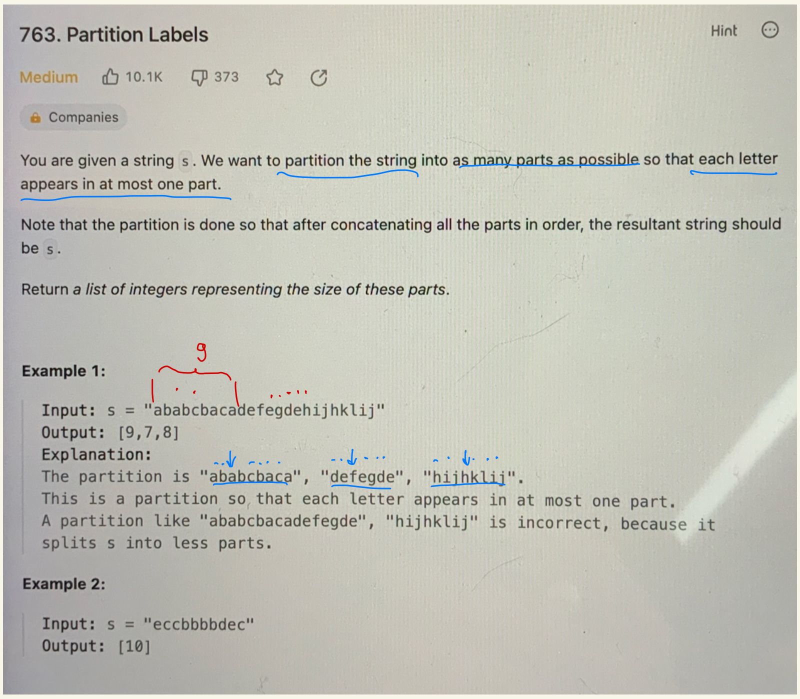
2.7



763. Partition Labels

Hint

Medium 10.1K 373

Companies

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]
```

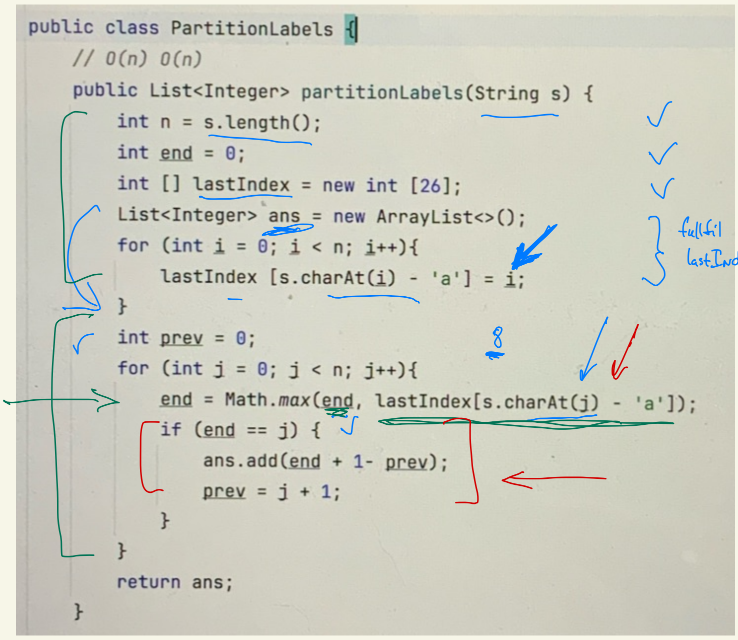
'a' - 97
b 98
c 99
d 100
e 101
f 102
g 103
h

'A' - 65

7

9 104
6
c
d
e
f
g

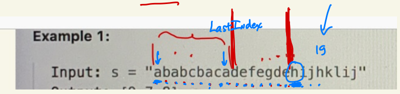
h
i
j
k
l



```java
public class PartitionLabels {
    // O(n) O(n)
    public List<Integer> partitionLabels(String s) {
        int n = s.length();
        int end = 0;
        int [] lastIndex = new int [26];
        List<Integer> ans = new ArrayList<>();
        for (int i = 0; i < n; i++){
            lastIndex [s.charAt(i) - 'a'] = i;
        }
        int prev = 0;
        for (int j = 0; j < n; j++){
            end = Math.max(end, lastIndex[s.charAt(j) - 'a']);
            if (end == j) {
                ans.add(end + 1- prev);
                prev = j + 1;
            }
        }
        return ans;
    }
```

fullfil lastIndex

8



Example 1:

```
Input: s = "ababcbacadefegdehijhklij"
```

lastIndex[] j a 0 1 2 3 4 5 6 7
8, 5, 7, 14, 15, 11, 13, 19, 22, 23, 20, 21

end: ∅ 8 ~~14~~ 15
prev: ∅ ~~9~~ 16

ans. add (end + 1 - prev)
9 7 24 - 16
8

prev = j + 1 ;

a b c d e f g h i j k l (12 letters)

for (i++) lastIndex

for (j++)

end = Math. Max (end, last [j - 'a'];

j = 8

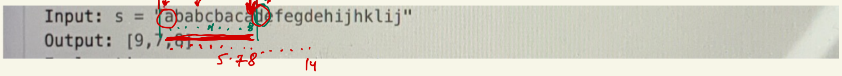
end = ∅, 8,

prev = ∅;

ans . add(end + 1 - prev);

prev = j + 1;

return ans.



```
Input: s = "ababcbacadefegdehijhklij"
Output: [9,7,8]
```

0 1 2 3 4 5 6 7 8 9 10 11
8, 5, 7, 14, 15, 11, 13, 19, 22, 23, 20, 21

1. create last Index array for chars.

2. loop through s

p
e == j

j == e
8
ans.add (end+1 - prev);
prev = j+1;