ASSUMPTION UNIVERSITY
Vincent Mary School of Science and Technology
Bachelor of Science Program in Information Technology

CS 3201 Algorithm Design 2/2021

Report

Term Project

Partition Labels

By

Phumin Abdul Hameed 6135117

Thanadon Na Lampang 6213930

Semester 2/2022

Content

1. Problem Statement

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.

Constraints:

1 <= s.length <= 500

s consists of lowercase English letters.

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"

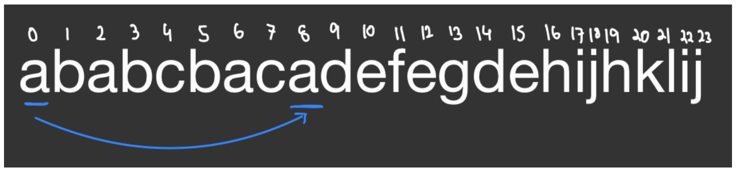
Output: [10]

Full Problem can be found here:

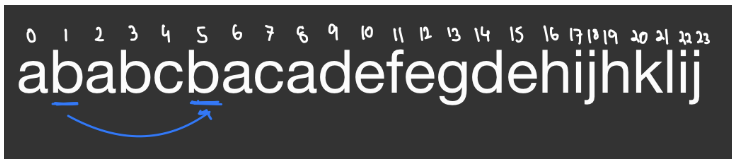
https://leetcode.com/problems/partition-labels/

2. Problem Analysis

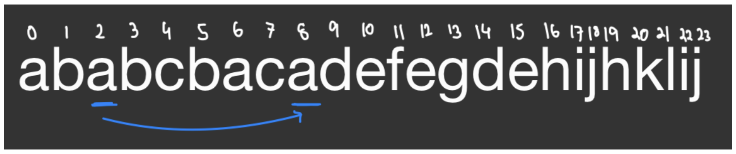
As you can see to ana;yse the problem we have gone with the first example to see how they drive at that output.



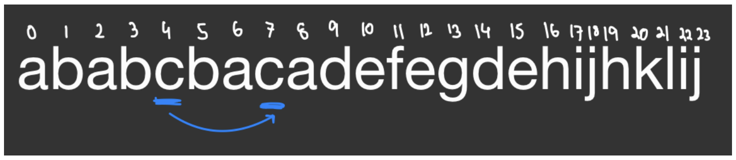
We started with letter “a” and the last letter “a” in the list is in index 8 we put in to the hashmap, same with every new letter.



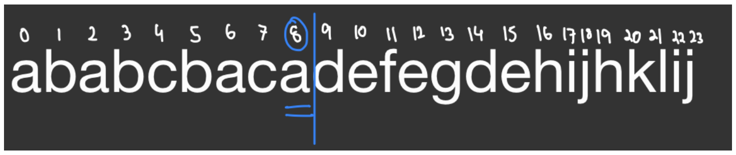
Next er increment by one and letter “b” has the last letter at index 5, we record in hashmap, it doesn't replace cur as its lower than 15.



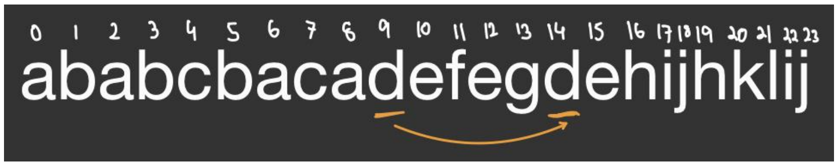
We check again with all the letters and also incrementing pre every step of the way by 1.



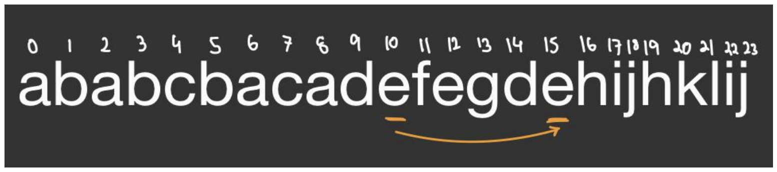
Letter “c” has also dont passed letter “a” in index meaning cur is still at 15.



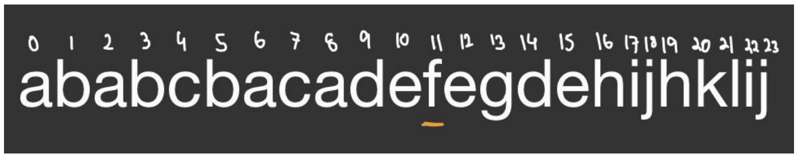
As you can see we are at the last letter in “a” and the cur still remains 15 and we have got the result for the first partition which is 9 because thats how much it incremented.

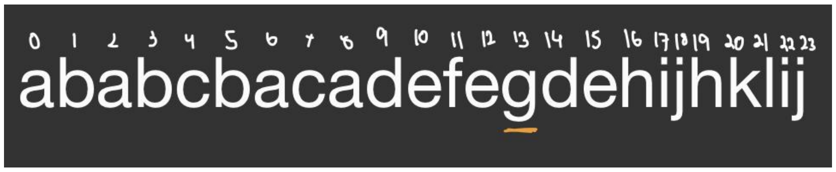


Now for the new partition we start again with 0 and move up and as you can see letter “d” will have the cur value of 14.

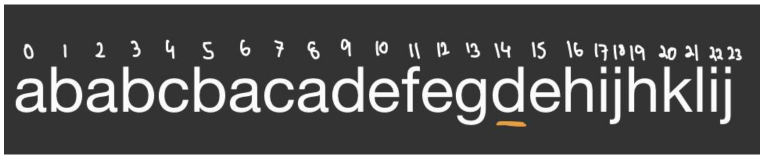


However when incremented to 2 and letter “e” the cur changes to 15 because higher cur value replaces it.

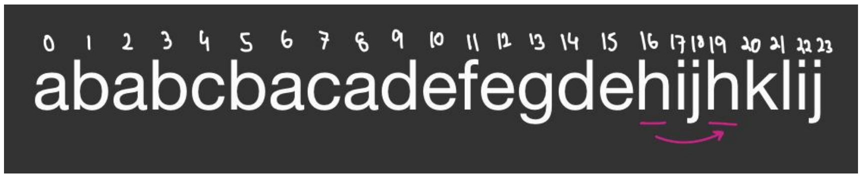




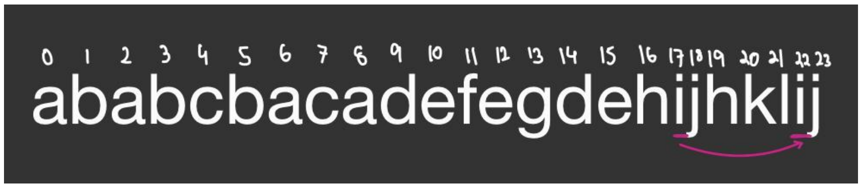
Letter “f” & “g” are standalone with no repeating so the cur doesnt get replace or anything, however the pre still increments.



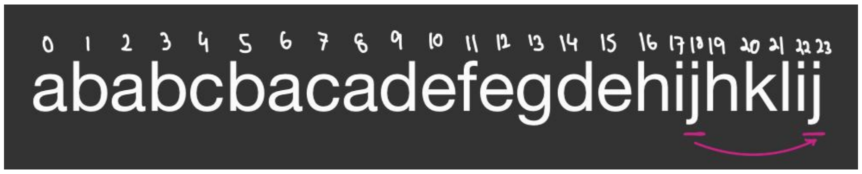
Letter “d” has already been replaced with letter “e” so only pre increment takes place here. Anf finally we landed on cur 15 and letter “e” this is the end of 2nd partition with 7 increments.



Now we move onto the next partition where the letter “h” has duplicate in index 19 meaning the cur as well.



Letter “i” quickly replace letter “h” with with the cur of 22.

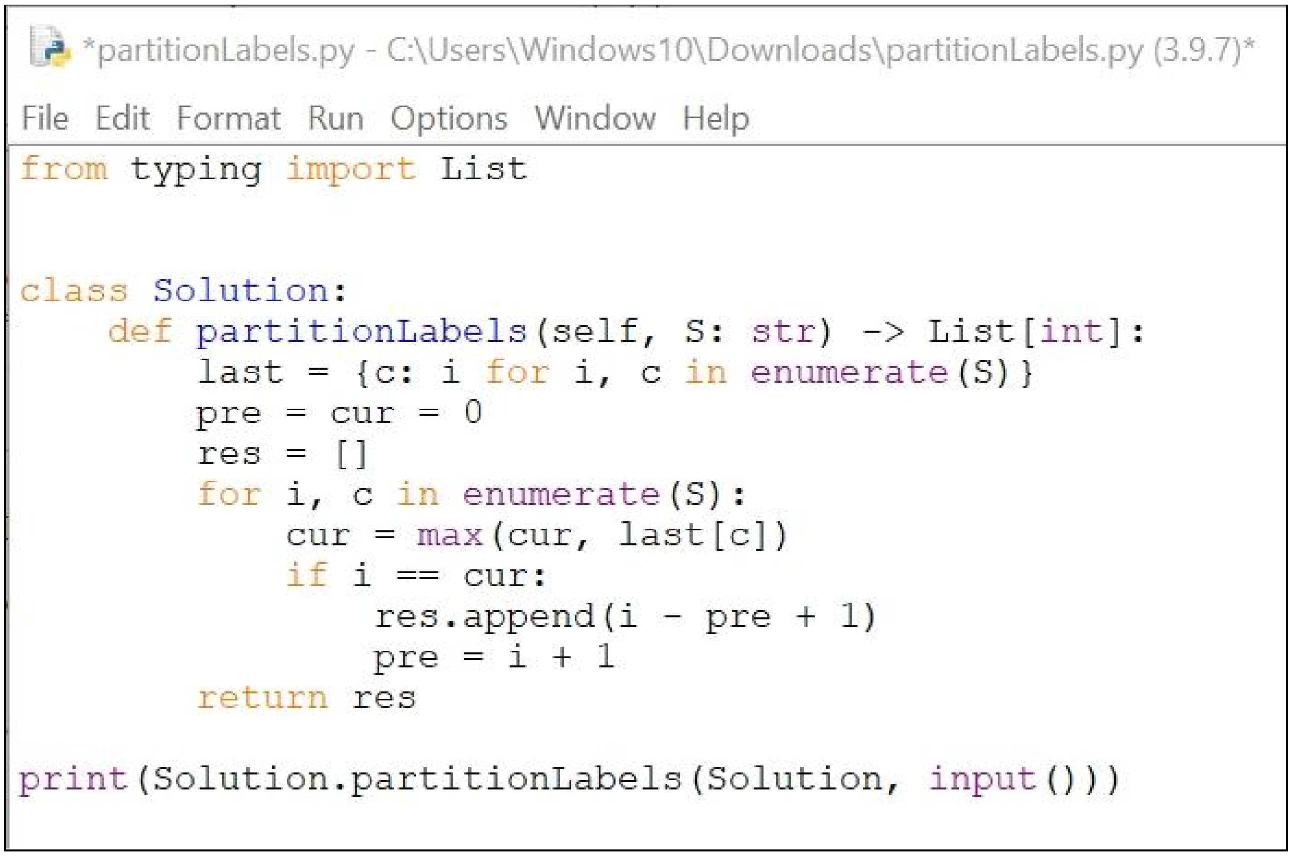


Lastly letter “j” had replaced everyone by being the last index on the list meaning all the pre are included from letter “h” which would be 9 and this is the partition. All these together give you the output = [9, 7, 8]

3. Solution & Code

The solution would firstly, if the letter appears in more than one segment of the string, we wish to separate it. It isn't possible for it to appear in the other section where we divide. We'd want to get a list of numbers that indicate the sizes of these components back. So, if we split "abcdefgabcde," one of the parts will be "a." As a result, we must ensure that it is divided at any point after the final index of a. As a result, we can only have one letter at a time. To be broken into two halves. We see the letters whenever we view them. We must ensure that we separate it to the final currents.

Below is the code that has been used to solve the problem.


partitionLabels.py - C:\Users\Windows10\Downloads\partitionLabels.py (3.9.7)
File Edit Format Run Options Window Help

```
from typing import List


class Solution:
    def partitionLabels(self, S: str) -> List[int]:
        last = {c: i for i, c in enumerate(S)}
        pre = cur = 0
        res = []
        for i, c in enumerate(S):
            cur = max(cur, last[c])
            if i == cur:
                res.append(i - pre + 1)
                pre = i + 1
        return res

print(Solution.partitionLabels(Solution, input()))
```

The time complexity of the program is big O(n) as it was scanning through the input and The memory complexity of the memory is big O(1) because the hashmap is limited.

5. Conclusion

One of the most popular interview questions is about partition labels. As a result, I picked this question, which is both intriguing to me and to my friends. Furthermore, we gain a lot of knowledge from it, such as how to use Looping to find and calculate the check value based on the number of indexes obtained from the message input. Furthermore, we believed this question would be difficult to code at first, but certainly, we can figure it out and complete it on time.

References

https://leetcode.com/problems/partition-labels/

Partition Labels - Leetcode 763 - Python