395. Longest Substring with At Least K Repeating Characters

Description | Hints | Submissions | Discuss | Solution

Pick One

Find the length of the longest substring *T* of a given string (consists of lowercase letters only) such that every character in *T* appears no less than *k* times.

Example 1:

```
Input:
s = "aaabb", k = 3

Output:
3

The longest substring is "aaa", as 'a' is repeated 3 times.
```

Example 2:

```
Input:
s = "ababbc", k = 2

Output:
5

The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.
```

Seen this question in a real interview before? Yes No

```java
public class L395 {
    /*
     * 这里的子串是连续子串，因为在字符串中的子串必须是连续的，重点重点
     */
     public int longestSubstring(String s, int k) {

       return   logestSubstringSub(s, k, 0, s.length() - 1);

     }
     /*
      * 思想，要找s[start]~s[end]的最大子串（连续），先统计子串的每个字符的频次，
      * 找到一个一个频数小于k且大于0的字符，然后找出这个字符的位置。这个字符一定不能出现
      * 在s[start]~s[end]的任何子串中，因为s[start]~s[end]是整个的字符串，
      * 在s[start]~s[end]中频次没有达到k，那么在其任何子串中也不可能达到k，所以不能有
      * 这个字符，就在这个位置做一个分治，返回前半部分和后半部分的最大值。
      */

     private int logestSubstringSub(String s, int k, int start, int end) {
         if(start > end)
             return 0;
         int [] count = new int [26];
         for(int i = start; i <= end; i ++) {
             count[s.charAt(i) - 'a'] ++;
         }

         for(int i = 0; i < 26; i ++) {
             if(count[i] > 0 && count[i] < k) {
                 int pos = s.indexOf((char)(i + 'a'), start);
                 return Math.max(logestSubstringSub(s, k, start, pos - 1), logestSubstringSub(s, k, pos + 1,
             }
         }
         return end - start + 1;
     }

     public static void main(String [] args) {
         System.out.println(new L395().longestSubstring("ababacb", 3));
     }
}
```