438. Find All Anagrams in a String(need to review) 滑动窗口

link

subString应该用滑动窗口法, 写了个O(n^2)的, 每一次有重复的比较.

别人的总结

```
public List<Integer> findAnagrams(String s, String p) {
    List<Integer> res = new ArrayList<>();
    if(s.length() < p.length())
        return res;
    int len = p.length();
    StringBuilder sb = new StringBuilder();
    for(int i = 0; i < len; ++i)
        sb.append(s.charAt(i));
    if(isAnagrams(sb.toString(), p))
        res.add(0);
    for(int i = 0; i + len < s.length(); ++i){
        sb.deleteCharAt(0);
        sb.append(s.charAt(i + len));
        if(isAnagrams(sb.toString(), p))
            res.add(i + 1);
    }
    return res;
}
public boolean isAnagrams(String s, String p){
    int[] map = new int[26];
    for(int i = 0; i < s.length(); ++i){
        map[s.charAt(i) - 'a']++;
        map[p.charAt(i) - 'a']--;
    }
    for(int i = 0; i < 26; ++i){
        if(map[i] != 0)
            return false;
    }
    return true;
}
```

自己写了一个O(n)的, 是每次跟之前状态比, 如果之前的是, 就之比较删除的和新增的两个, 不是的话, 就跳到下一个.

```
public List<Integer> findAnagrams(String s, String p) {
```

```
        int len = s.length();
        int[] bucket = new int[26];
        List<Integer> res = new ArrayList<>();
        for(int i = 0; i < p.length(); ++i){
            bucket[p.charAt(i) - 'a']++;
        }
        int i = 0;
        int[] copy = new int[26];
        System.arraycopy(bucket, 0, copy, 0, 26);
        boolean isValid = false;
        while(i <= len - p.length()){
            int j = i + p.length();
            if(isValid && s.charAt(i - 1) == (s.charAt(j - 1))){
                res.add(i);
                i++;
                continue;
            }
            for(int n = i; n < j; ++n){
                if(bucket[s.charAt(n) - 'a'] <= 0){
                    i = n + 1;
                    isValid = false;
                    break;
                }else if(copy[s.charAt(n) - 'a'] <= 0){
                    isValid = false;
                    i++;
                    break;
                }
                copy[s.charAt(n) - 'a']--;
                isValid = true;
            }
            if(isValid){
                res.add(i);
                ++i;
            }
            System.arraycopy(bucket, 0, copy, 0, 26);
        }
        return res;
    }
```

通用的题目: 建立一个map, 然后通过map.get()—end先找到末尾 如果还是counter=0, start开始加, 如果长度刚好等于size就加入结果集合, start再加counter就不是0了, 这时候end继续加.

```
public List<Integer>findAnagrams(String s, String t) {
    List<Integer> result = new LinkedList<>();
    if(t.length()> s.length()) return result;
    Map<Character, Integer> map = new HashMap<>();
    for(char c : t.toCharArray()){
        map.put(c, map.getOrDefault(c, 0) + 1);
    }
    int counter = map.size();
    int begin = 0, end = 0;
    int len = Integer.MAX_VALUE;
    while(end < s.length()){
        char c = s.charAt(end);
        if( map.containsKey(c) ){
            map.put(c, map.get(c)-1);
            if(map.get(c) == 0) counter--;
```

```
            }
            end++;
            while(counter == 0){
                char tempc = s.charAt(begin);
                if(map.containsKey(tempc)){
                    map.put(tempc, map.get(tempc) + 1);
                    if(map.get(tempc) > 0) counter++;
                }
                begin++;
            }
        }
        return result;
}
```

159. Longest Substring with At Most Two Distinct Characters

滑动窗口, 看了答案写出来的

link

总结link

```
    public int lengthOfLongestSubstringTwoDistinct(String s) {
        int left = 0, right = 0, count = 0, max = 0;
        Map<Character, Integer> map = new HashMap<>();
        while(right < s.length()){
            char c = s.charAt(right);
            map.put(c, map.getOrDefault(c, 0) + 1);
            if(map.get(c) == 1) count++;
            right++;
            while(count > 2){
                char leftChar = s.charAt(left);
                map.put(leftChar, map.get(leftChar) - 1);
                if(map.get(leftChar) == 0)
                    --count;
                ++left;
            }
            max = Math.max(right - left, max);
        }
        return max;
    }
```

395. Longest Substring with At Least K Repeating Characters

link

这个题用普通的办法很难找到start和end的条件, 我们可以找 有一个字母大于三次, 有两个字母大于三次, 有三个字母大于3次...有26个字母大于三次.循环26次.