Longest Substring with At Most K Distinct Characters

Given a string, find the length of the longest substring T that contains at most *k* distinct characters.

For example, Given s = `“eceba”` and k = 2,

T is "ece" which its length is 3.

Solution 1

feel it is not a new question, just use num to track the number of distinct characters within the slide window

```
public class Solution {
    public int lengthOfLongestSubstringKDistinct(String s, int k) {
        int[] count = new int[256];
        int num = 0, i = 0, res = 0;
        for (int j = 0; j < s.length(); j++) {
            if (count[s.charAt(j)]++ == 0) num++;
            if (num > k) {
                while (--count[s.charAt(i++)] > 0);
                num--;
            }
            res = Math.max(res, j - i + 1);
        }
        return res;
    }
}
```

written by jiangbowei2010 original link here

Solution 2

da jia zhao gong zuo jia you!!!

two cases: 1. when there are less than k chars in the hash table we put the char in the hash table

1. when there are exactly distinct k chars in the hash table and we need to add a new char, we delete the left most char

#

```
class Solution(object):
    def lengthOfLongestSubstringKDistinct(self, s, k):
        """
        :type s: str
        :type k: int
        :rtype: int
        """
        if k == 0:
            return 0
        start = 0
        char_hash = {}
        max_len = 0
        for char in s:
            if len(char_hash) < k:
                if char not in char_hash:
                    char_hash[char] = 1
                else:
                    char_hash[char] += 1
            elif len(char_hash) == k:
                if char not in char_hash:
                    max_len = max(max_len, sum(char_hash.values()))
                    while len(char_hash) == k:
                        char_hash[s[start]] -= 1
                        if char_hash[s[start]] == 0:
                            del char_hash[s[start]]
                        start += 1
                    char_hash[char] = 1
                else:
                    char_hash[char] += 1
        max_len = max(max_len, sum(char_hash.values()))
        return max_len
```

written by liangwang original link here

Solution 3

This problem can be solved using two pointers. The important part is `while (map.size() > k)`, we move left pointer to make sure the map size is less or equal to `k`. This can be easily extended to any number of unique characters.

```
public int lengthOfLongestSubstringKDistinct(String s, int k) {
    Map<Character, Integer> map = new HashMap<>();
    int left = 0;
    int best = 0;
    for(int i = 0; i < s.length(); i++) {
        // character at the right pointer
        char c = s.charAt(i);
        map.put(c, map.getOrDefault(c, 0) + 1);
        // make sure map size is valid, no need to check left pointer less than s
.length()
        while (map.size() > k) {
            char leftChar = s.charAt(left);
            map.put(leftChar, map.get(leftChar) - 1);
            if (map.get(leftChar) == 0) {
                map.remove(leftChar);
            }
            left++;
        }
        best = Math.max(best, i - left + 1);
    }
    return best;
}
```

written by yfcheng original link here