LeetCode 340

https://leetcode.com/problems/longest-substring-with-at-most-k-distinct-characters/description/

Yifeng Zeng

Description

340. 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.

Idea Report

We can have a pair of slow (i) and fast(j) pointers. And a HashMap to save the occurrence of the characters. We need to keep the map.size() as k. If map.size() < k, we move j to right and add the occurrence of character[j]. If the map.size() > k, we move i to right and reduce the occurrence of character[i], if the occurrence of character[i] is 0, we remove it from HashMap. And we also have a global maximum to compare the distance between i and j and return this global maximum length.

Code

```
class Solution {
    public int lengthOfLongestSubstringKDistinct(String s, int k) {
        Map<Character, Integer> map = new HashMap<>();
        int i = 0;
        int j = 0;
        int len = 0;
        while (j < s.length() && map.size() < k) {
            map.put(s.charAt(j), map.getOrDefault(s.charAt(j), 0) + 1);
            j++;
```

```
            len = Math.max(len, j - i);
        }

        while (j < s.length()) {
            char ch = s.charAt(j);
            if (map.containsKey(ch)) {
                map.put(ch, map.get(ch) + 1);
                len = Math.max(len, j - i + 1);
            } else {
                map.put(ch, 1);
            }
            j++;
            while (map.size() > k) {
                char remove = s.charAt(i);
                map.put(remove, map.get(remove) - 1);
                if (map.get(remove) == 0) {
                    map.remove(remove);
                }
                i++;
            }
        }
        return len;
    }
}
```

And we can use a int[] hash of size 256 and a counter to replace the map.

Code

```
class Solution {
    public int lengthOfLongestSubstringKDistinct(String s, int k) {
        int[] hash = new int[256];
        int i = 0;
        int j = 0;
        int count = 0;
        int len = 0;
        while (j < s.length()) {
            char ch = s.charAt(j);
            if (hash[ch] == 0) {
                count++;
            }
            hash[ch]++;
            while (count > k) {
                char remove = s.charAt(i);
                hash[remove]--;
                if (hash[remove] == 0) {
                    count--;
                }
```

```
                i++;
            }
            len = Math.max(len, j - i + 1);
            j++;
        }

        return len;
    }
}
```

Summary

- Two pointers chasing one another (追击型双指针)