438 Find All Anagrams in a String

2018年4月11日 13:58

Given a string **s** and a **non-empty** string **p**, find all the start indices of **p**'s anagrams in **s**.

Strings consists of lowercase English letters only and the length of both strings **s** and **p** will not be larger than 20,100.

The order of output does not matter.

Example 1:

Input:

s: "cbaebabacd" p: "abc"

Output:

[0, 6]

Explanation:

The substring with start index = 0 is "cba", which is an anagram of "abc".

The substring with start index = 6 is "bac", which is an anagram of "abc".

Example 2:

Input:

s: "abab" p: "ab"

Output:

[0, 1, 2]

Explanation:

The substring with start index = 0 is "ab", which is an anagram of "ab".

The substring with start index = 1 is "ba", which is an anagram of "ab".

The substring with start index = 2 is "ab", which is an anagram of "ab".

来自 <https://leetcode.com/problems/find-all-anagrams-in-a-string/description/>

Solution for Python3:

```
class Solution1:
    def findAnagrams(self, s, p):
        """
        :type s: str
        :type p: str
        :rtype: List[int]
        """
        sw, pw, res = [0] * 26, [0] * 26, []
```

```
            if len(s) < len(p):
                return res
            for i in range(len(p)):
                sw[ord(s[i]) - ord('a')] += 1
                pw[ord(p[i]) - ord('a')] += 1
            if sw == pw:
                res.append(0)
            for i in range(len(p), len(s)):
                sw[ord(s[i]) - ord('a')] += 1
                sw[ord(s[i - len(p)]) - ord('a')] -= 1
                if sw == pw:
                    res.append(i - len(p) + 1)
            return res

    class Solution2:
        def findAnagrams(self, s, p):
            """
            :type s: str
            :type p: str
            :rtype: List[int]
            """
            from collections import Counter
            res = []
            pCounter = Counter(p)
            sCounter = Counter(s[:len(p) - 1])
            for i in range(len(p) - 1, len(s)):
                sCounter[s[i]] += 1
                if sCounter == pCounter:
                    res.append(i - len(p) + 1)
                sCounter[s[i - len(p) + 1]] -= 1
                if sCounter[s[i - len(p) + 1]] == 0:
                    del sCounter[s[i - len(p) + 1]]
            return res

```

Solution for C++:

```
class Solution {
public:
```

```
    vector<int> findAnagrams(string s, string p) {
        vector<int> sw(26, 0), pw(26, 0), res;
        if (s.size() < p.size())
            return res;
        //存储初始滑动窗口sw和匹配窗口pw
        for (int i = 0; i < p.size(); i++) {
            ++sw[s[i] - 'a'];
            ++pw[p[i] - 'a'];
        }
        if (sw == pw)
            res.push_back(0);
        for (int i = p.size(); i < s.size(); i++) {
            //加入新进入sw的元素
            ++sw[s[i] - 'a'];
            //减去出去sw的元素
            --sw[s[i - p.size()] - 'a'];
            if (sw == pw)
                res.push_back(i - p.size() + 1);
        }
        return res;
    }
};
```