题目：

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".
```

Subscribe to see which companies asked this question.

1.时间：O（N*M）；空间：O（1）

```
class Solution {
public:
    vector<int> findAnagrams(string s, string p) {
        if (p.empty() || s.size() < p.size()) return std::vector<int>();
        std::vector<int> hashTable(26, 0);
        for (int i = 0; i < p.size(); ++i){
            hashTable[p[i] - 'a']++;
        }
        std::vector<int> result;
        const int Upper = s.size() - p.size();
        for (int i = 0; i <= Upper; ++i){    /* Upper ~ s.size() - 1 也可能匹配 */
            if (isAnagrams(s.substr(i, p.size()), hashTable))
                result.push_back(i);
        }
        return result;
    }
private:
    bool isAnagrams(const std::string& str, std::vector<int> hashTable){
        for (int i = 0; i < str.size(); ++i){
```

```
            if (--hashTable[str[i] - 'a'] < 0) return false;
        }
        return true;
    }
};
```

2.时间：O（N）；空间：O（N）

```
class Solution {
public:
    vector<int> findAnagrams(string s, string p) {
        if (p.empty() || s.size() < p.size()) return std::vector<int>();
        std::vector<int> hashTable(26, 0);
        for (int i = 0; i < p.size(); ++i){
            hashTable[p[i] - 'a']++;
        }
        std::vector<int> result;
        const int Upper = s.size() - p.size();
        int need = p.size();
        int start_index = 0;
        for (int i = 0; i < s.size(); ++i){
            const int index = s[i] - 'a';
            if (--hashTable[index] >= 0) need--;
            if (need == 0) result.push_back(start_index);
```

```
            if (i - start_index == p.size() - 1){
                hashTable[s[start_index] - 'a']++;
                if (hashTable[s[start_index] - 'a'] > 0){
                    need++;
                }
                start_index++;
            }
        }
        if (need == 0) result.push_back(start_index);
        return result;
    }
};
```