题目： 438. 找到字符串中所有字母异位词

固定窗口大小题目，用两个哈希表判断。

```cpp
class Solution {
public:
    vector<int> findAnagrams(string s, string p) {
        vector<int> ret;
        int hash1[26] = {0};
        int hash2[26] = {0};
        for(auto& e : p) //统计字符串p中出现的字符
        {
            hash2[e - 'a']++;
        }
        int n = p.size();
        int left = 0;
        int count = 0;
        for(int right = 0;right < s.size();right++)
        {
            hash1[s[right]- 'a']++; //入窗口
            if(hash1[s[right]- 'a'] <= hash2[s[right] - 'a']) count++; //维护count
            if(right-left +1 == n) //判断窗口大小
            {
                if(count == n)
                    ret.push_back(left);
                if(hash1[s[left]- 'a'] <= hash2[s[left] - 'a']) count--; //出窗口，维护count
                hash1[s[left] - 'a']--;
                left++;
            }
        }
        return ret;
    }
};
```

题目： 30. 串联所有单词的子串

```cpp
class Solution {
public:
    vector<int> findSubstring(string s, vector<string>& words) {
    unordered_map<string, int> hash1; // 保存 words 里面所有单词的频次
    for (auto& word : words) hash1[word]++;

    int len = words[0].size(), m = words.size(); // 单词长度 len 和单词个数 m
    vector<int> ret;

    for (int i = 0; i < len; i++) { // 执行 len 次
        unordered_map<string, int> hash2; // 维护窗口内单词的频次
        for (int left = i, right = i, count = 0; right + len <= s.size(); right += len) {
            // 进入窗口 + 维护 count
            string in = s.substr(right, len);
            hash2[in]++;
            if (hash2[in] <= hash1[in]) count++;

            // 判断窗口大小是否超过 len * m
            if (right - left + 1 > len * m) {
                // 出窗口 + 维护 count
                string out = s.substr(left, len);
                if (hash2[out] <= hash1[out]) count--;
                hash2[out]--;
                left += len;
            }

            // 更新结果
            if (count == m) ret.push_back(left);
        }
    }

    return ret;
}
};
```