87. Scramble String

题目描述：https://leetcode.com/problems/scramble-string/

> Given a string s1, we may represent it as a binary tree by partitioning it to two non-empty substrings recursively.

```
Below is one possible representation of s1 = "great":
    great
   /    \
  gr    eat
 / \    /  \
g   r  e   at
           / \
          a   t
To scramble the string, we may choose any non-leaf node and swap its two child
ren.
For example, if we choose the node "gr" and swap its two children, it produces
 a scrambled string "rgeat".
    rgeat
   /    \
  rg    eat
 / \    /  \
r   g  e   at
           / \
          a   t
We say that "rgeat" is a scrambled string of "great".
Similarly, if we continue to swap the children of nodes "eat" and "at", it pro
duces a scrambled string "rgtae".
    rgtae
   /    \
  rg    tae
 / \    /  \
r   g  ta  e
       / \
      t   a
We say that "rgtae" is a scrambled string of "great".
Given two strings s1 and s2 of the same length, determine if s2 is a scrambled
 string of s1.
```

解题思路：

递归

代码：

```
class Solution {
public:
    bool isScramble(string s1, string s2) {
        if(s1 == s2) return true;
        if(s1.size() != s2.size()) return false;
        int len = s1.size();
        vector<int> count(26, 0);
        for(int i = 0; i < len; i++) {
            count[s1[i]-'a']++;
            count[s2[i]-'a']--;
        }
        for(int item: count) {
            if(item != 0) return false;
        }
        for(int i = 1; i < len; i++) {
            if(isScramble(s1.substr(0, i), s2.substr(0, i)) && isScramble(s1.subst
r(i), s2.substr(i))) {
                return true;
            }
            if(isScramble(s1.substr(0, i), s2.substr(len-i)) && isScramble(s1.subs
tr(i), s2.substr(0, len-i))) {
                return true;
            }
        }
        return false;

    }
};
```