题目：

Given a non-empty string check if it can be constructed by taking a substring of it and appending multiple copies of the substring together. You may assume the given string consists of lowercase English letters only and its length will not exceed 10000.

Example 1:

```
Input: "abab"


Output: True


Explanation: It's the substring "ab" twice.
```

Example 2:

```
Input: "aba"


Output: False
```

Example 3:

```
Input: "abcabcabcabc"


Output: True


Explanation: It's the substring "abc" four times. (And the substring "abcabc
" twice.)
```

Subscribe to see which companies asked this question.

1.时间：O（N^2）;空间：O（N） ->暴力法

class Solution {

```
public:
    bool repeatedSubstringPattern(string s) {
        if (s.empty()) return true;
        if (s.size() == 1) return false;
        const int len = s.size();
        for (int i = 1; i <= len / 2; ++i){
            if (len % i == 0){
                std::string substr = s.substr(0, i);
                int k = 1;
                for (; k < len / i; ++k){
                    if (s.substr(k*i, i) != substr) break;
                }
                if (k == len / i) return true;
            }
        }
        return false;
    }
};
```

2.时间：O（N）; 空间：O（N）

```
class Solution {
public:
    bool repeatedSubstringPattern(string s) {
        if (s.empty()) return true;

        if (s.size() == 1) return false;

        const int len = s.size();

        std::vector<int> dp(len + 1, 0);

        for (int i = 1, k = 0; i < len;){

            if (s[i] == s[k]) dp[++i] = ++k;

            else if (k == 0) ++i;

            else k = dp[k];

        }

        return dp[len] && (dp[len] % (len - dp[len]) == 0);

    }

};
```

3.时间：O（N）; 空间：O（N）

```
class Solution {

public:

    bool repeatedSubstringPattern(string s) {

        if (s.empty()) return true;

        if (s.size() == 1) return false;

        const int len = s.size();

        std::string ss = (s + s).substr(1, 2 * len - 2);

        return ss.find(s) != -1;

    }
};
```