459 Repeated Substring Pattern

2018年4月12日 10:05

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.)

来自 <https://leetcode.com/problems/repeated-substring-pattern/description/>
给定一个非空的字符串，判断它是否可以由它的一个子串重复多次构成。给定的字符串只含有小写英文字母，并且长度不超过10000。

Solution for Python3:

```python
class Solution1:
    def repeatedSubstringPattern(self, s):
        """
        :type s: str
        :rtype: bool
        """
        L, step = len(s), len(s) // 2
        while step > 0:
           if L % step == 0:
               m = L // step
               Fstr = s[:step]
               i = 1
               while i < m:
                   if Fstr != s[i * step : i * step + step]:
                       break
                   i += 1
               if i == m:
                   return True
           step -= 1
        return False

class Solution2:
    def repeatedSubstringPattern(self, s):
        """
        :type s: str
        :rtype: bool
        """
        return any(s[:i] * (len(s) / i) == s for i in range(1, len(s) // 2 + 1) if len(s) % i == 0)

class Solution3:
    def repeatedSubstringPattern(self, s):
        """
        :type s: str
        :rtype: bool
        """
        if not str:
           return False
        ss = (s + s)[1:-1]
        return ss.find(s) != -1

class Solution4:
    def repeatedSubstringPattern(self, s):
        """
        :type s: str
        :rtype: bool
        """
        return s in (s * 2)[1:-1]
```

Solution for C++:

```cpp
class Solution1 {
public:
    bool repeatedSubstringPattern(string s) {
        int len = s.length();
        for (int step = len / 2; step > 0; step--) {
            //step步长，最长为一半长度
            if (len % step == 0) {
                int m = len / step;
                string substr = s.substr(0, step);
                int j;
                for (j = 1; j < m; j++) {
                    if (s.compare(j * step, step, substr) != 0)
                        break;
                }
                if (j == m)
                    return true;
            }
        }
        return false;
    }
};

class Solution2 {
public:
    bool repeatedSubstringPattern(string s) {
        return (s+s).substr(1, s.length() * 2 - 2).find(s) != string::npos;
    }
};
```

Appendix:

Python any() 和 all():

1) any(x)判断x对象是否为空对象，如果都为空、0、False则返回False，只要有一个不为空、0、False则返回True。
2) alll(x)如果参数x对象的所有元素都不为空、0、''、False则返回True,否则返回False。

重复的子字符串问题:

1) 字符串s='abcabcabc'为含有重复子字符串的字符串。
2) 这种字符串可以通过旋转(即前或后移动子字符串长度)而保持不变。
3) 如s前移或者后移3个字符串不变。
4) 判断一个字符串是否含有重复的子字符串，即判断其能否通过前后一定一定长度而保持不变。这种字符串至少含有两部分重复部分。且重复部分的开头和结尾分别为为字符串开头和结尾。
5) 串接两个字符串s得到ss = s + s='abc abc abc||abc abc abc'。即相当于把后一个s的头部接到前一个s的尾部。这样相当于实现了s的旋转。
6) 去掉ss的前后元素，相当于破坏了ss的前后重复部分，即把s的第一个重复部分移动到s的尾部，如果此时能在ss中找到s字符串，说明s通过旋转第一个重复部分到s的尾后还是等于s，这个旋转是在串接两个s完成的。