Repeated Substring Pattern

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.)
```

Solution 1

```
public boolean repeatedSubstringPattern(String str) {
 int l = str.length();
 for(int i=l/2;i>=1;i--) {
  if(l%i==0) {
   int m = l/i;
   String subS = str.substring(0,i);
   StringBuilder sb = new StringBuilder();
   for(int j=0;j<m;j++) {
    sb.append(subS);
   }
   if(sb.toString().equals(str)) return true;
  }
 }
 return false;
}
```

1. The length of the repeating substring must be a divisor of the length of the input string
2. Search for all possible divisor of `str.length`, starting for `length/2`
3. If `i` is a divisor of `length`, repeat the substring from `0` to `i` the number of times `i` is contained in `s.length`
4. If the repeated substring is equals to the input `str` return `true`

written by fabrizio3 original link here

Solution 2

```
public boolean repeatedSubstringPattern(String str) {
        //This is the kmp issue
        int[] prefix = kmp(str);
        int len = prefix[str.length()-1];
        int n = str.length();
        return (len > 0 && n%(n-len) == 0);
    }
    private int[] kmp(String s){
        int len = s.length();
        int[] res = new int[len];
        char[] ch = s.toCharArray();
        int i = 0, j = 1;
        res[0] = 0;
        while(i < ch.length && j < ch.length){
            if(ch[j] == ch[i]){
                res[j] = i+1;
                i++;
                j++;
            }else{
                if(i == 0){
                    res[j] = 0;
                    j++;
                }else{
                    i = res[i-1];
                }
            }
        }
        return res;
    }
```

written by chnsht original link here

Solution 3

Basic idea:

1. First char of input string is first char of repeated substring
2. Last char of input string is last char of repeated substring
3. Let S1 = S + S (where S in input string)
4. Remove 1 and last char of S1. Let this be S2
5. If S exists in S2 then return true else false
6. Let i be index in S2 where S starts then repeated substring length i + 1 and repeated substring S[0: i+1]

```
def repeatedSubstringPattern(self, str):

        """
        :type str: str
        :rtype: bool
        """
        if not str:
            return False

        ss = (str + str)[1:-1]
        return ss.find(str) != -1
```

written by rsrs3 original link here