205 Isomorphic Strings

2018年4月4日 19:51

Question:

Given two strings ***s*** and ***t***, determine if they are isomorphic.
Two strings are isomorphic if the characters in ***s*** can be replaced to get ***t***.
All occurrences of a character must be replaced with another character while preserving the order of characters.
No two characters may map to the same character but a character may map to itself.
For example,
Given "egg", "add", return true.
Given "foo", "bar", return false.
Given "paper", "title", return true.
Note:
You may assume both ***s*** and ***t*** have the same length.

来自 <https://leetcode.com/problems/isomorphic-strings/description/>

给定两个字符串 *s* 和 *t*，判断它们是否是同构的。
如果 *s* 中的字符可以被替换最终变成 *t* ，则两个字符串是同构的。
所有出现的字符都必须用另一个字符替换，同时保留字符的顺序。两个字符不能映射到同一个字符上，但字符可以映射自己本身。
例如，
给定 "egg", "add", 返回 true.
给定 "foo", "bar", 返回 false.
给定 "paper", "title", 返回 true.
注意：
你可以假设 *s* 和 *t* 具有相同的长度。

Solution for Python3:

```python
class Solution1:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        return self.getStructure(s) == self.getStructure(t)

    def getStructure(self, str):
        d = {}
        ss = []
        cnt = 0
        for i in str:
            t = d.get(i)
            if not t:
                cnt += 1
                d[i] = cnt
                ss.append(cnt)
            else:
                ss.append(t)
        return ss

```

```python

class Solution2:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        ms, mt = [0] * 128, [0] * 128
        for i in range(len(s)):
            if ms[ord(s[i])] != mt[ord(t[i])]:
                return False
            elif not ms[ord(s[i])]:
                ms[ord(s[i])] = i + 1
                mt[ord(t[i])] = i + 1
        return True

class Solution3:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        d1, d2 = {}, {}
        for i, val in enumerate(s):
            d1[val] = d1.get(val, []) + [i]
        for i, val in enumerate(t):
            d2[val] = d2.get(val, []) + [i]
            # sort(d.values())只是把字典结构的值取出来按照list有序存放
            # 进而对list进行比较
        return sorted(d1.values()) == sorted(d2.values())

class Solution4:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        d1, d2 = [[] for _ in range(128)], [[] for _ in range(128)]
        for i, val in enumerate(s):
            d1[ord(val)].append(i)
        for i, val in enumerate(t):
            d2[ord(val)].append(i)
        return sorted(d1) == sorted(d2)

class Solution5:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
```

```
        # zip() 函数用于将可迭代的对象作为参数，
        # 将对象中对应的元素打包成一个个元组，
        # 然后返回由这些元组组成的列表。
        return len(set(zip(s, t))) == len(set(s)) == len(set(t))

class Solution6:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        s.find(i)找到元素i在s中出现的首次位置
        return [s.find(i) for i in s] == [t.find(j) for j in t]

class Solution7:
    def isIsomorphic(self, s, t):
        """
        :type s: str
        :type t: str
        :rtype: bool
        """
        return map(s.find, s) == map(t.find, t)
```

Solution for C++:

```
class Solution {
public:
    bool isIsomorphic(string s, string t) {
        int ms[128] = {0}, mt[128] = {0};
        for (int i = 0; i < s.length(); i++) {
            if (ms[s[i]] != mt[t[i]]) {
                return false;
            } else if (ms[s[i]] != 0) {
                ms[s[i]] = mt[t[i]] = i + 1;
            }
        }
        return true;
    }
};
```

Appendix:

Python 内置函数 zip([iterable,...]): iterabl -- 一个或多个迭代器;

1) 用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。
2) 如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 * 号操作符，可以将元组解压为列表。
   a. >>>a = [1,2,3]
   b. >>> b = [4,5,6]

c. >>> c = [4,5,6,7,8]

d. >>> zipped = zip(a,b) # 打包为元组的列表 [(1, 4), (2, 5), (3, 6)]

e. >>> zip(a,c) # 元素个数与最短的列表一致 [(1, 4), (2, 5), (3, 6)]

f. >>> zip(*zipped) # 与 zip 相反，可理解为解压，返回二维矩阵式 [(1, 2, 3), (4, 5, 6)]