Isomorphic Strings

Question:

https://leetcode.com/problems/isomorphic-strings/

Given two strings s and t, determine if they are isomorphic.

Two strings s and t 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.

Example 1:

Input: s = "egg", t = "add"
Output: true

Example 3:

Input: s = "paper", t = "title"
Output: true

Approach 1:

My first approach was to use a hashmap to check occurrences,
{e:a , g:d}
but the problem I encountered was that I could only map one-way.
So inorder to map both-ways, I used 2 hashmaps.
{e:a,g:d}
{a:e,d:g}

Now we check for similar mapping in both the hashmaps simultaneously.

Solution 1:

```
if len(s)!=len(t):
        return False
    d1={}
    d2={}
    for i in range(len(s)):
        if s[i] in d1:
            if d1[s[i]]!=t[i]:
                return False
        if t[i] in d2:
            if d2[t[i]]!=s[i]:
                return False
        d1[s[i]]=t[i]
        d2[t[i]]=s[i]
    return True
```

Approach 2:

If we can map alphabets to numbers then we can find an encoding which should be same for both the strings.
Ex: PAPER will be 01023 and TITLE will also be 01023
Since both strings have the same encoded string hence they are isomorphic.
Another ex: FOO will 011 while BAR will be 012.
Since they are not equal hence they are not isomorphic.

Solution 2:

```
def isIsomorphic(self, s: str, t: str) -> bool:
    return self.encoding(s)==self.encoding(t)

  def encoding(self,s):
```

```
d={}
encoded=[]
for c in s:
    if c not in d:
        d[c]=len(d)
    encoded+=[d[c]]
return encoded
```