1.Easy\05.Isomorphic_string.cpp

```cpp
/*
Question:
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.

Approach:
1. Initialize two maps to store the mapping of characters from s to t and from t to s.
2. Iterate through each character in s and t simultaneously.
3. If the current characters in s and t are already mapped differently, return false.
4. If the current characters in s and t are not mapped yet, add them to the maps.
5. If the current characters in s and t are already mapped to each other, continue to the next characters.
6. If all characters have been iterated and no inconsistencies are found, return true.

Code:
*/

bool isIsomorphic(string s, string t) {
    unordered_map<char, char> mps;
    unordered_map<char, char> mpt;

    for (int i = 0; i < s.size(); i++) {
        if (mps.find(s[i]) == mps.end() && mpt.find(t[i]) == mpt.end()) {
            mps[s[i]] = t[i];
            mpt[t[i]] = s[i];
        } else if (mps[s[i]] != t[i] || mpt[t[i]] != s[i]) {
            return false;
        }
    }

    return true;
}

/*
Time Complexity: O(n), where n is the length of the input strings s and t.
- We iterate through each character of s and t once.

Space Complexity: O(m), where m is the number of unique characters in the input strings s and t.
- In the worst case, all characters in s and t are unique, and we need to store mappings for all of them.
- The space complexity can also be considered as O(1) since the maximum number of unique characters is limited (26 English alphabets).
*/


```