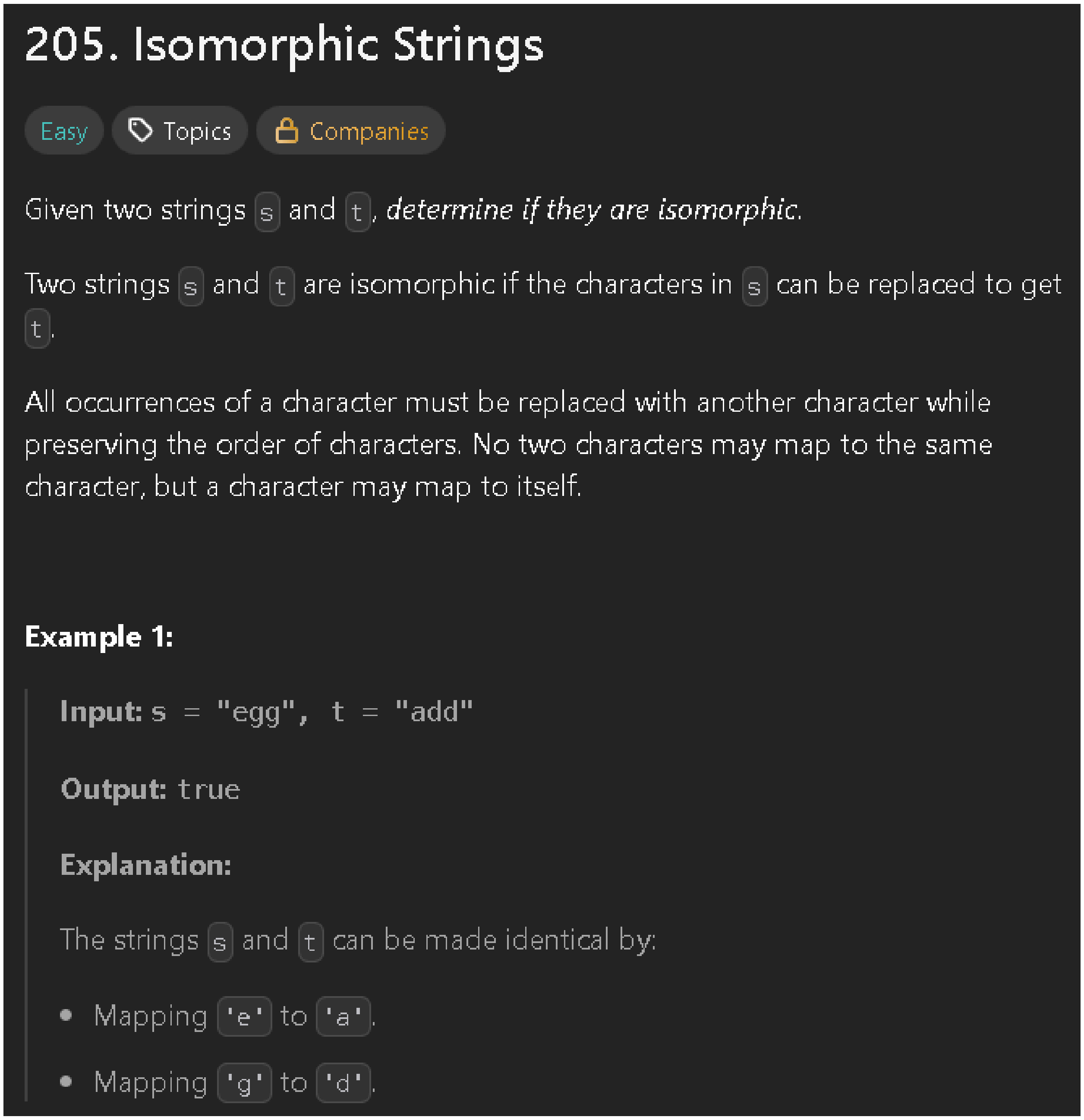

205. Isomorphic Strings

Easy Topics Companies

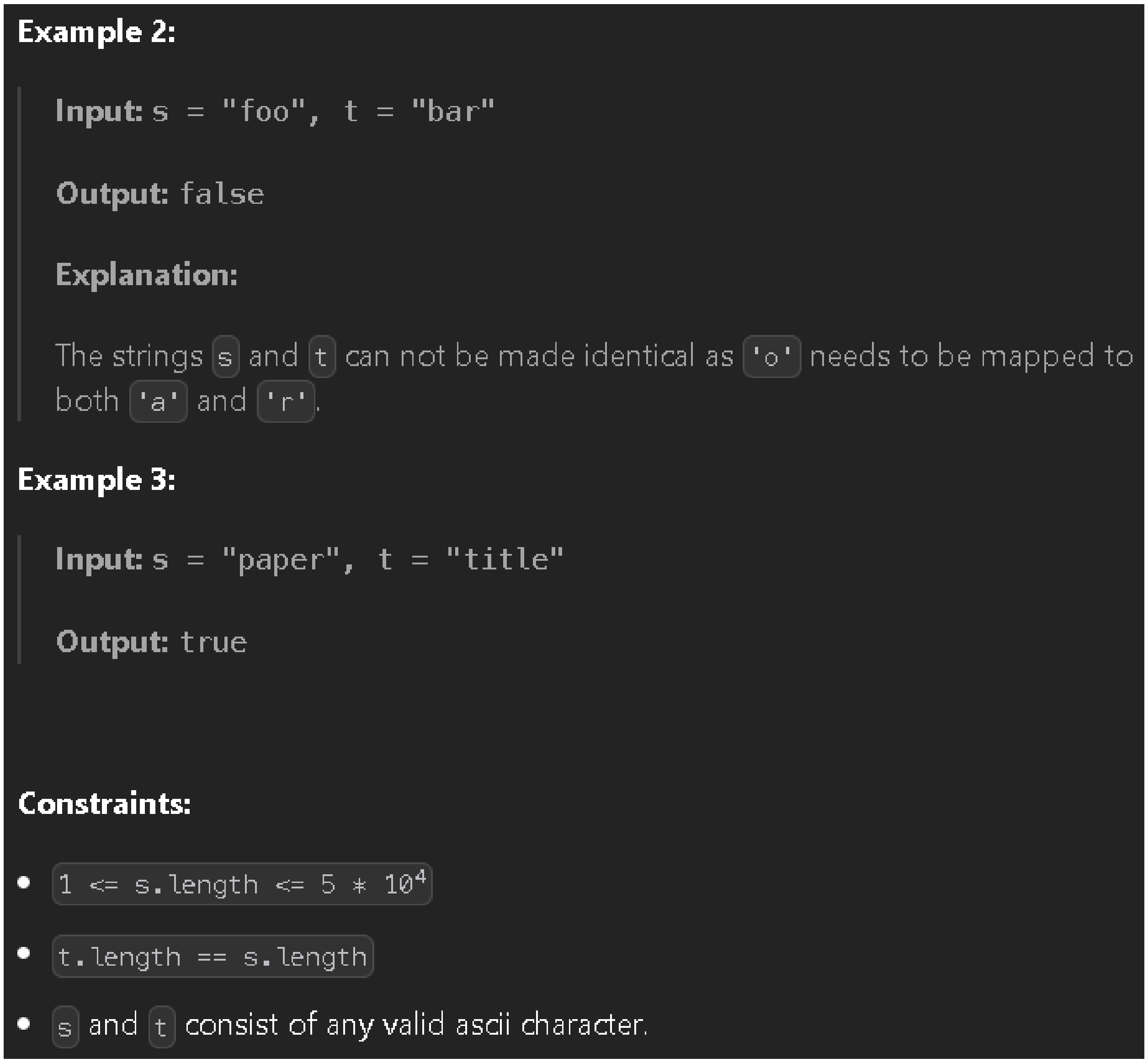
Given two strings `s` and `t`, *determine if they are isomorphic*.

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`
>
> **Explanation:**
>
> The strings `s` and `t` can be made identical by:
>
> - Mapping `'e'` to `'a'`.
> - Mapping `'g'` to `'d'`.



Example 2:

> **Input:** `s = "foo", t = "bar"`
>
> **Output:** `false`
>
> **Explanation:**
>
> The strings `s` and `t` can not be made identical as `'o'` needs to be mapped to both `'a'` and `'r'`.

Example 3:

> **Input:** `s = "paper", t = "title"`
>
> **Output:** `true`

Constraints:

- `1 <= s.length <= 5 * 10^4`
- `t.length == s.length`
- `s` and `t` consist of any valid ascii character.

Python:

```
class Solution:
    def isIsomorphic(self, s: str, t: str) -> bool:
        # If lengths are not equal, they can't be isomorphic
        if len(s) != len(t):
            return False

        # Dictionaries to store mappings
        s_to_t = {}
        t_to_s = {}

        for char_s, char_t in zip(s, t):
            # Check mapping from s to t
            if char_s in s_to_t:
                if s_to_t[char_s] != char_t:
                    return False
            else:
                s_to_t[char_s] = char_t

            # Check mapping from t to s
            if char_t in t_to_s:
                if t_to_s[char_t] != char_s:
                    return False
            else:
                t_to_s[char_t] = char_s

        return True
```

JavaScript:

```
/**
 * @param {string} s
 * @param {string} t
 * @return {boolean}
 */
var isIsomorphic = function(s, t) {
    if (s.length !== t.length) return false;

    const mapST = {};
    const mapTS = {};

    for (let i = 0; i < s.length; i++) {
        const charS = s[i];
        const charT = t[i];

        // Check s -> t mapping
        if (mapST[charS] && mapST[charS] !== charT) {
            return false;
        }

        // Check t -> s mapping
        if (mapTS[charT] && mapTS[charT] !== charS) {
            return false;
        }

        // Establish the mapping
        mapST[charS] = charT;
        mapTS[charT] = charS;
    }

    return true;
};
```

Java:

```
import java.util.HashMap;

class Solution {
    public boolean isIsomorphic(String s, String t) {
        if (s.length() != t.length()) return false;

        HashMap<Character, Character> mapST = new HashMap<>();
        HashMap<Character, Character> mapTS = new HashMap<>();

        for (int i = 0; i < s.length(); i++) {
            char c1 = s.charAt(i);
            char c2 = t.charAt(i);

            // Check mapping from s -> t
            if (mapST.containsKey(c1)) {
                if (mapST.get(c1) != c2) return false; // conflict
            } else {
                mapST.put(c1, c2);
            }

            // Check mapping from t -> s
            if (mapTS.containsKey(c2)) {
                if (mapTS.get(c2) != c1) return false; // conflict
            } else {
                mapTS.put(c2, c1);
            }
        }
        return true;
    }
}
```