Leetcode - 205

🧩 Problem Statement

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

🟡 Two strings are *isomorphic* if you can **replace characters** in `s` to get `t`, with:

- Each character mapping to **exactly one** character (one-to-one)
- The **order preserved**
- **No two characters** in `s` mapping to the **same** character in `t`

🧠 Example

```
s = "egg", t = "add" ✅
s = "foo", t = "bar" ❌
s = "paper", t = "title" ✅
```

🔍 Code Walkthrough

```
class Solution {
public:
    bool isIsomorphic(string s, string t) {
```

Defines the function `isIsomorphic`, which returns `true` if the strings `s` and `t` are isomorphic.

🛡️ Step 1: Base Check

```
        if(s.length()!=t.length()) return false;
```

If lengths differ, they're clearly **not isomorphic**.

🧮 Step 2: One-way Mapping (s → t)

```
vector<int>v(150,1000);
```

- Initializes a vector of size 150 (ASCII cover) to store differences.
- All values start at `1000` (our "unassigned" marker).

```
for(int i=0;i<s.length();i++){
    int idx=(int)s[i];
    if(v[idx]==1000)
        v[idx]=s[i]-t[i]; // Save the char difference
    else if(v[idx]!=(s[i]-t[i]))
        return false; // Mismatch? Not isomorphic
}
```

🧠 This loop:

- Checks if the current character `s[i]` was **mapped before**.
- If not → store the difference `s[i]-t[i]` (which remains consistent if mapping is valid).
- If already mapped → ensures the **same difference** exists. If not → invalid mapping.

📌 Example:

`egg` vs `add`

`e - a = 4`,

`g - d = 3`, and again `g - d = 3`

Works fine.

🔁 Step 3: Reverse Mapping (t → s)

```
for(int i=0;i<150;i++) v[i]=1000;
```

Reset the vector for reverse mapping.

```
for(int i=0;i<s.length();i++){
    int idx=(int)t[i];
    if(v[idx]==1000)
        v[idx]=t[i]-s[i]; // Reverse diff
    else if(v[idx]!=(t[i]-s[i]))
        return false;
}
```

Just like before, now we check that `t → s` mapping is also consistent.

👉 **Why both directions?**

To avoid cases like:

```
s = "ab", t = "aa"
```

Here:

- `a → a` is fine
- `b → a` is invalid (two chars → one char).

  But `s[i] - t[i]` for both is `0` → only checking one way would **miss** this problem.

✅ Final Return

```
return true;
```

If all checks pass, the strings are isomorphic.

🧮 Time & Space Complexity

- **Time**: `O(n)` for both mappings
- **Space**: `O(1)` (vector of size 150, constant size)

🧙 TL;DR Summary

Step	Purpose
`length check`	Bail out early if sizes don't match
`s → t mapping`	Ensure each character in `s` maps consistently to a character in `t`
`t → s mapping`	Ensure no two characters in `s` map to the same in `t`
`v[idx] = s[i] - t[i]`	Stores a difference to keep track of the unique mapping rule