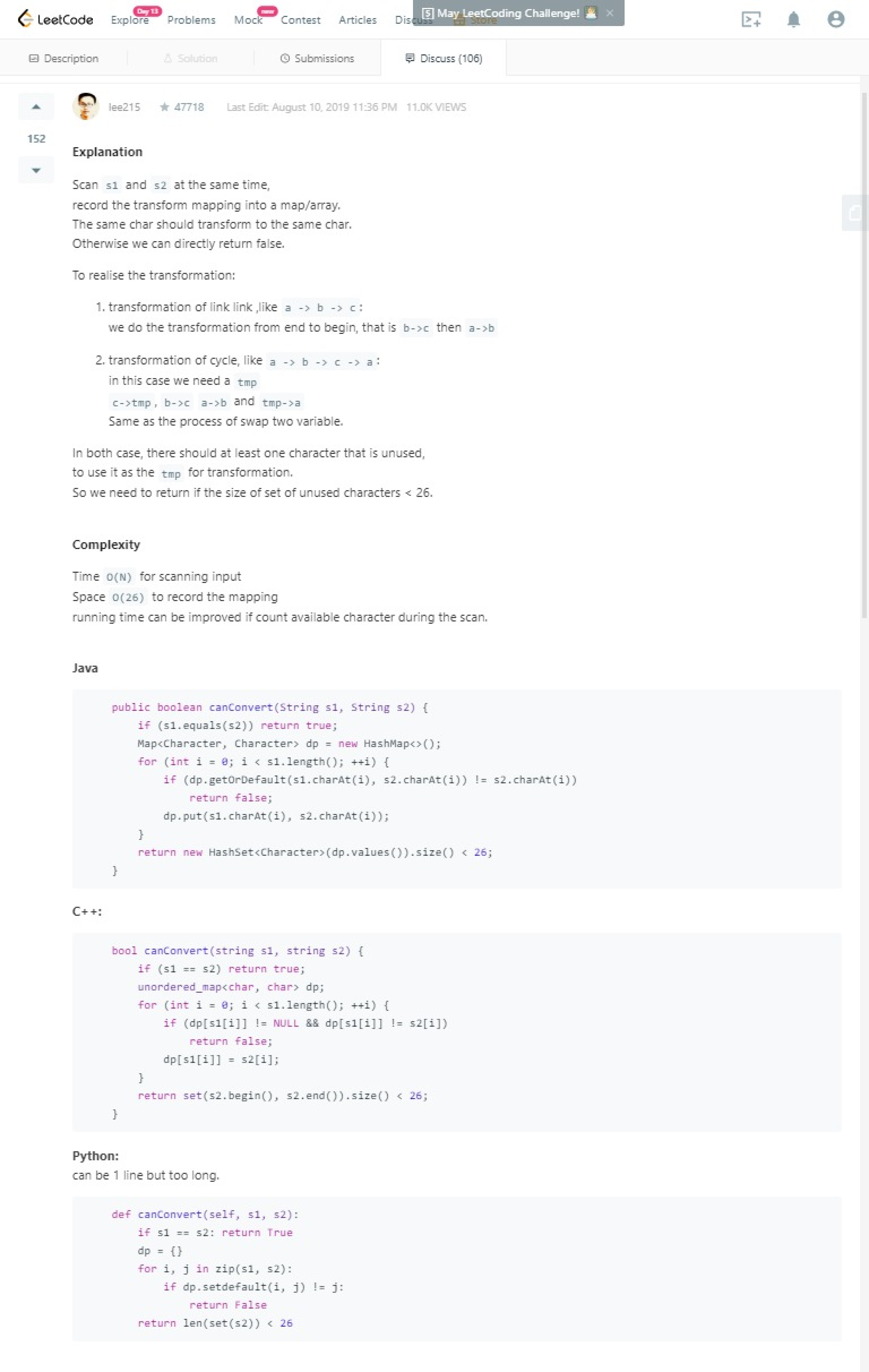

Explanation

Scan `s1` and `s2` at the same time,
record the transform mapping into a map/array.
The same char should transform to the same char.
Otherwise we can directly return false.

To realise the transformation:

1. transformation of link link ,like `a -> b -> c`:
   we do the transformation from end to begin, that is `b->c` then `a->b`

2. transformation of cycle, like `a -> b -> c -> a`:
   in this case we need a `tmp`
   `c->tmp`, `b->c` `a->b` and `tmp->a`
   Same as the process of swap two variable.

In both case, there should at least one character that is unused,
to use it as the `tmp` for transformation.
So we need to return if the size of set of unused characters < 26.

Complexity

Time `O(N)` for scanning input
Space `O(26)` to record the mapping
running time can be improved if count available character during the scan.

Java

```java
public boolean canConvert(String s1, String s2) {
    if (s1.equals(s2)) return true;
    Map<Character, Character> dp = new HashMap<>();
    for (int i = 0; i < s1.length(); ++i) {
        if (dp.getOrDefault(s1.charAt(i), s2.charAt(i)) != s2.charAt(i))
            return false;
        dp.put(s1.charAt(i), s2.charAt(i));
    }
    return new HashSet<Character>(dp.values()).size() < 26;
}
```

C++:

```cpp
bool canConvert(string s1, string s2) {
    if (s1 == s2) return true;
    unordered_map<char, char> dp;
    for (int i = 0; i < s1.length(); ++i) {
        if (dp[s1[i]] != NULL && dp[s1[i]] != s2[i])
            return false;
        dp[s1[i]] = s2[i];
    }
    return set(s2.begin(), s2.end()).size() < 26;
}
```

Python:
can be 1 line but too long.

```python
def canConvert(self, s1, s2):
    if s1 == s2: return True
    dp = {}
    for i, j in zip(s1, s2):
        if dp.setdefault(i, j) != j:
            return False
    return len(set(s2)) < 26
```