5302. Encrypt and Decrypt Strings

My Submissions (/contest/weekly-contest-287/problems/encrypt-and-decrypt-strings/submissions/) Back to Contest (/contest/weekly-contest-287/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

You are given a character array `keys` containing **unique** characters and a string array `values` containing strings of length 2. You are also given another string array `dictionary` that contains all permitted original strings after decryption. You should implement a data structure that can encrypt or decrypt a **0-indexed** string.

A string is **encrypted** with the following process:

1. For each character `c` in the string, we find the index `i` satisfying `keys[i] == c` in `keys`.
2. Replace `c` with `values[i]` in the string.

A string is **decrypted** with the following process:

1. For each substring `s` of length 2 occurring at an even index in the string, we find an `i` such that `values[i] == s`. If there are multiple valid `i`, we choose **any** one of them. This means a string could have multiple possible strings it can decrypt to.
2. Replace `s` with `keys[i]` in the string.

Implement the `Encrypter` class:

- `Encrypter(char[] keys, String[] values, String[] dictionary)` Initializes the `Encrypter` class with `keys, values`, and `dictionary`.
- `String encrypt(String word1)` Encrypts `word1` with the encryption process described above and returns the encrypted string.
- `int decrypt(String word2)` Returns the number of possible strings `word2` could decrypt to that also appear in `dictionary`.

Example 1:

```
Input
["Encrypter", "encrypt", "decrypt"]
[[['a', 'b', 'c', 'd'], ["ei", "zf", "ei", "am"], ["abcd", "acbd", "adbc", "badc", "dacb", "cadb", "cbda", "abad"]], ["abcd"],
Output
[null, "eizfeiam", 2]

Explanation
Encrypter encrypter = new Encrypter([['a', 'b', 'c', 'd'], ["ei", "zf", "ei", "am"], ["abcd", "acbd", "adbc", "badc", "dacb", "
encrypter.encrypt("abcd"); // return "eizfeiam".
                           // 'a' maps to "ei", 'b' maps to "zf", 'c' maps to "ei", and 'd' maps to "am".
encrypter.decrypt("eizfeiam"); // return 2.
                              // "ei" can map to 'a' or 'c', "zf" maps to 'b', and "am" maps to 'd'.
                              // Thus, the possible strings after decryption are "abad", "cbad", "abcd", and "cbcd".
                              // 2 of those strings, "abad" and "abcd", appear in dictionary, so the answer is 2.
```

Constraints:

- `1 <= keys.length == values.length <= 26`
- `values[i].length == 2`
- `1 <= dictionary.length <= 100`
- `1 <= dictionary[i].length <= 100`
- All `keys[i]` and `dictionary[i]` are **unique**.
- `1 <= word1.length <= 2000`
- `1 <= word2.length <= 200`
- All `word1[i]` appear in `keys`.
- `word2.length` is even.
- `keys`, `values[i]`, `dictionary[i]`, `word1`, and `word2` only contain lowercase English letters.
- At most `200` calls will be made to `encrypt` and `decrypt` **in total**.

JavaScript

```
const counter_value_in_indexA_in = (a_or_s) => { let m = new Map(); let n = a_or_s.length; for (let i = 0; i < n; i++) {
if (!m.has(a_or_s[i])) m.set(a_or_s[i], []); m.get(a_or_s[i]).push(i); } return m; };

function Encrypter(k, v, dict) {
    dict = new Set(dict);
    let mk = counter_value_in_indexA_in(k), mv = counter_value_in_indexA_in(v), memo = new Map();
    return { encrypt, decrypt }
    function encrypt(s) {
        let res = '';
        for (const c of s) {
            let i = mk.get(c)[0];
            res += v[i];
        }
        return res;
    }
    function decrypt(s) {
        if (memo.has(s)) return memo.get(s);
        let res = 0;
        for (const dic of dict) {
            if (encrypt(dic) == s) res++;
        }
        memo.set(s, res);
        return res;
    }
}
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/672749036/) More Details (/submissions/detail/672749036/)

Share your acceptance!

1