2266. Count Number of Texts

My Submissions (/contest/weekly-contest-292/problems/count-number-of-texts/submissions/) Back to Contest (/contest/weekly-contest-292/)

User Accepted:	1659
User Tried:	2159
Total Accepted:	1761
Total Submissions:	4102
Difficulty:	Medium

Alice is texting Bob using her phone. The **mapping** of digits to letters is shown in the figure below.

In order to **add** a letter, Alice has to **press** the key of the corresponding digit `i` times, where `i` is the position of the letter in the key.

- For example, to add the letter `'s'`, Alice has to press `'7'` four times. Similarly, to add the letter `'k'`, Alice has to press `'5'` twice.
- Note that the digits `'0'` and `'1'` do not map to any letters, so Alice **does not** use them.

However, due to an error in transmission, Bob did not receive Alice's text message but received a **string of pressed keys** instead.

- For example, when Alice sent the message `"bob"`, Bob received the string `"2266622"`.

Given a string `pressedKeys` representing the string received by Bob, return *the **total number of possible text messages** Alice could have sent*.

Since the answer may be very large, return it **modulo** $10^9 + 7$.

Example 1:

```
Input: pressedKeys = "22233"
Output: 8
Explanation:
The possible text messages Alice could have sent are:
"aaadd", "abdd", "badd", "cdd", "aaae", "abe", "bae", and "ce".
Since there are 8 possible messages, we return 8.
```

Example 2:

```
Input: pressedKeys = "222222222222222222222222222222222222"
Output: 82876089
Explanation:
There are 2082876103 possible text messages Alice could have sent.
Since we need to return the answer modulo 10^9 + 7, we return 2082876103 % (10^9 + 7) = 82876089.
```

Constraints:

- `1 <= pressedKeys.length <= 10^5`
- `pressedKeys` only consists of digits from `'2' - '9'`.

Discuss (https://leetcode.com/problems/count-number-of-texts/discuss)

JavaScript

```
const mod = 1e9 + 7;
const cal = (c) => c == '7' || c == '9' ? 4 : 3

const countTexts = (s) => {
    let n = s.length;
    let dp = Array(n + 1).fill(0);
    dp[0] = 1;
    for (let i = 0; i < n; i++) {
        for (let j = 1; j <= cal(s[i]); j++) {
            let len = i - j + 1;
            if (len < 0 || s[i] != s[len]) break;
```

```
            dp[i + 1] += dp[i + 1 - j];
        }
        dp[i + 1] %= mod;
    }
    return dp[n];
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/695402361/) | More Details > (/submissions/detail/695402361/)

Share your acceptance!

3