6957. Count Stepping Numbers in Range

My Submissions (/contest/weekly-contest-356/problems/count-stepping-numbers-in-range/submissions/) Back to Contest (/contest/weekly-contest-356/)

User Accepted:	0
User Tried:	2
Total Accepted:	0
Total Submissions:	2
Difficulty:	Hard

Given two positive integers `low` and `high` represented as strings, find the count of **stepping numbers** in the inclusive range `[low, high]`.

A **stepping number** is an integer such that all of its adjacent digits have an absolute difference of **exactly** `1`.

Return *an integer denoting the count of stepping numbers in the inclusive range* `[low, high]`.

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

Note: A stepping number should not have a leading zero.

Example 1:

```
Input: low = "1", high = "11"
Output: 10
Explanation: The stepping numbers in the range [1,11] are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. There are a total of 10 stepping nu
```

Example 2:

```
Input: low = "90", high = "101"
Output: 2
Explanation: The stepping numbers in the range [90,101] are 98 and 101. There are a total of 2 stepping numbers in the range. H
```

Constraints:

- `1 <= int(low) <= int(high) < 10^100`
- `1 <= low.length, high.length <= 100`
- `low` and `high` consist of only digits.
- `low` and `high` don't have any leading zeros.

JavaScript

```javascript
const minus_mod = (x, y, mod) => ((x - y) % mod + mod) % mod;
const mod = 1e9 + 7, ll = BigInt;

const countSteppingNumbers = (low, high) => {
    let x = go(high), y = go((ll(low) - 1n).toString());
    return minus_mod(x, y, mod);
};

let memo;
const go = (s) => {
    memo = new Map();
    return dfs(0, 0, true, false, s);
};

const dfs = (i, mask, isLimit, isNum, s) => {
    let ke = i + " " + mask + " " + isLimit + " " + isNum;
    if (memo.has(ke)) return memo.get(ke);
    if (i == s.length) return isNum - '0';
    let res = 0;
    if (!isNum) res = dfs(i + 1, mask, false, false, s);
    let leading = isNum ? 0 : 1;
    let up = isLimit ? s[i] - '0' : 9;
    for (let digit = leading; digit <= up; digit++) {
        if (!isNum || Math.abs(digit - mask) == 1) {
            res += dfs(i + 1, digit, isLimit && digit == up, true, s);
        }
    }
    res %= mod;
```

```
    memo.set(ke, res);
    return res;
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/1007535907/) | More Details > (/submissions/detail/1007535907/)

Share your acceptance!