639. Decode Ways II

- Difficulty:Hard
- Total Accepted:1.7K
- Total Submissions:7.5K
- Contributor:love_Fawn

A message containing letters from `A-Z` is being encoded to numbers using the following mapping way:

```
'A' -> 1
'B' -> 2
...
'Z' -> 26
```

Beyond that, now the encoded string can also contain the character '*', which can be treated as one of the numbers from 1 to 9.

Given the encoded message containing digits and the character '*', return the total number of ways to decode it.

Also, since the answer may be very large, you should return the output mod 109 + 7.

Example 1:

Input: `"*"` **Output:** `9` **Explanation:** `The encoded message can be decoded to the string: "A", "B", "C", "D", "E", "F", "G", "H", "I".`

Example 2:

Input: `"1*"` **Output:** `9 + 9 = 18`

Note:

1. The length of the input string will fit in range [1, 105].
2. The input string will only contain the character '*' and digits '0' - '9'.

The idea is DP. One of the hints is that you need mod the answer with a huge prime number.

For any string s longer than 2, we can decode either the last 2 characters as a whole or the last 1 character. So dp[i] = dp[i-1]* f(s.substr(i,1)) + dp[i-2]* f(s.substr(i-1, 2)). f() is the number of ways to decode a string of length 1 or 2. f() could be 0, for example f("67").

There is a lot of cases and corner cases for f(string s). For example, * cannot be '0', so ** has 15 instead of 16 possibilities, because "20" is excluded. But the time complexity is still O(n).

```
class Solution {
public:
    int numDecodings(string s) {
        int n = s.size(), p = 1000000007;
        // f2 is the answer to sub string ending at position i; Initially i = 0.
        long f1 = 1, f2 = helper(s.substr(0,1));
        // DP to get f2 for sub string ending at position n-1;
        for (int i = 1; i < n; i++) {
            long f3 = (f2*helper(s.substr(i, 1)))+(f1*helper(s.substr(i-1, 2)));
            f1 = f2;
            f2 = f3%p;
        }
        return f2;
    }
private:
    int helper(string s) {
        if (s.size() == 1) {
            if (s[0] == '*') return 9;
            return s[0] == '0'? 0:1;
        }
        // 11-26, except 20 because '*' is 1-9
```

```
            if (s == "**")
                return 15;
            else if (s[1] =='*') {
                if (s[0] =='1') return 9;
                return s[0] == '2'? 6:0;
            }
            else if (s[0] == '*')
                return s[1] <= '6'? 2:1;
            else
                // if two digits, it has to be in [10 26]; no leading 0
                return stoi(s) >= 10 && stoi(s) <= 26? 1:0;
        }
};
```