91. Decode Ways

Created	@June 10, 2021 1:14 PM
Tags	Medium
link	https://leetcode.com/problems/decode-ways/

Description

A message containing letters from `A-Z` can be **encoded** into numbers using the following mapping:

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

To **decode** an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, `"11106"` can be mapped into:

- `"AAJF"` with the grouping `(1 1 10 6)`
- `"KJF"` with the grouping `(11 10 6)`

Note that the grouping `(1 11 06)` is invalid because `"06"` cannot be mapped into `'F'` since `"6"` is different from `"06"`.

Given a string `s` containing only digits, return *the* ***number*** *of ways to* ***decode*** *it*.

The answer is guaranteed to fit in a **32-bit** integer.

Approach

- First formulate the problem as a recursive problem

$$F(X[0:L]) = F(X[1:L]) + F(X[2:L]|X[0:1]\ is\ digit)$$

- Observe that the problem has `optimal substructure` i.e. if we store count for substrings they can be reused.
- Implement a cache on top of the solution to implement top-down dp solution.

```
class Solution {
public:
    map<string, int> count_mapping;

    int findNumDecodings(string s, int index) {
        int doubleDigit;
        int countDouble = 0;
        int countSingle = 0;

        if (index >= s.length()) return 1;

        string substring = s.substr(index, s.length() - index);

        if (
            count_mapping.find(substring) != count_mapping.end()) {
                return count_mapping[substring];
            }

        if (index < s.length() - 1) {
            doubleDigit = (s[index] - '0')*10 + (s[index+1] - '0');
            if (doubleDigit <= 26 && doubleDigit > 0 && s[index]-'0' > 0) {
                countDouble = findNumDecodings(s, index+2);
            }
        }

        if (s[index]-'0' > 0)
            countSingle = findNumDecodings(s, index+1);

        count_mapping[substring] = countDouble + countSingle;
        return count_mapping[substring];
    }
    int numDecodings(string s) {
        return findNumDecodings(s, 0);
    }
};
```