0091. 解码方法

ITCharge 大约 3 分钟

- 标签：字符串、动态规划
- 难度：中等

题目链接

- 0091. 解码方法 - 力扣

题目大意

描述：给定一个数字字符串 s。该字符串已经按照下面的映射关系进行了编码：

- `A` 映射为 1。
- `B` 映射为 2。
- ...
- `Z` 映射为 26。

基于上述映射的方法，现在对字符串 s 进行「解码」。即从数字到字母进行反向映射。比如 `"11106"` 可以映射为：

- `"AAJF"` ，将消息分组为 (11106)。
- `"KJF"` ，将消息分组为 (11106)。

要求：计算出共有多少种可能的解码方案。

说明：

- $1 \leq s.length \leq 100$。
- s 只包含数字，并且可能包含前导零。
- 题目数据保证答案肯定是一个 32 位的整数。

示例：

- 示例 1：

```py
输入：s = "226"
输出：3
解释：它可以解码为 "BZ" (2 26), "VF" (22 6), 或者 "BBF" (2 2 6) 。
```

解题思路

思路 1：动态规划

1. 划分阶段

按照字符串的结尾位置进行阶段划分。

2. 定义状态

定义状态 $dp[i]$ 表示为：字符串 s 前 i 个字符构成的字符串可能构成的翻译方案数。

3. 状态转移方程

$dp[i]$ 的来源有两种情况：

1. 使用了一个字符，对 $s[i]$ 进行翻译。只要 $s[i]! = 0$，就可以被翻译为 `A ~ I` 的某个字母，此时方案数为 $dp[i] = dp[i - 1]$。
2. 使用了两个字符，对 $s[i - 1]$ 和 $s[i]$ 进行翻译，只有 $s[i - 1]! = 0$，且 $s[i - 1]$ 和 $s[i]$ 组成的整数必须小于等于 26 才能翻译，可以翻译为 `J ~ Z` 中的某字母，此时方案数为 $dp[i] = dp[i - 2]$。

这两种情况有可能是同时存在的，也有可能都不存在。在进行转移的时候，将符合要求的方案数累加起来即可。

状态转移方程可以写为：

$$dp[i] += \begin{cases} dp[i - 1] & s[i] \neq 0 \\ dp[i - 2] & s[i - 1] \neq 0, s[i - 1 : i] \leq 26 \end{cases}$$

4. 初始条件

- 字符串为空时，只有一个翻译方案，翻译为空字符串，即 $dp[0] = 1$。

- 字符串只有一个字符时，需要考虑该字符是否为 0，不为 0 的话，$dp[1] = 1$，为 0 的话，$dp[0] = 0$。

5. 最终结果

根据我们之前定义的状态，$dp[i]$ 表示为：字符串 s 前 i 个字符构成的字符串可能构成的翻译方案数。则最终结果为 $dp[size]$，$size$ 为字符串长度。

思路 1：动态规划代码

```py
class Solution:
    def numDecodings(self, s: str) -> int:
        size = len(s)
        dp = [0 for _ in range(size + 1)]
        dp[0] = 1
        for i in range(1, size + 1):
            if s[i - 1] != '0':
                dp[i] += dp[i - 1]
            if i > 1 and s[i - 2] != '0' and int(s[i - 2: i]) <= 26:
                dp[i] += dp[i - 2]
        return dp[size]
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。一重循环遍历的时间复杂度是 $O(n)$。
- **空间复杂度**：$O(n)$。用到了一维数组保存状态，所以总体空间复杂度为 $O(n)$。