Tutorial: Decodificando Mensagens

Leetcode 91

O problema pode ser resolvido utilizando uma abordagem de **programação dinâmica**, na qual calculamos, passo a passo, o número de formas possíveis de decodificar a sequência até cada posição.

A ideia é que cada dígito (ou par de dígitos consecutivos) da sequência pode representar uma letra do alfabeto latino, desde que o valor correspondente esteja entre 1 e 26. Assim, precisamos contar todas as maneiras válidas de interpretar a sequência numérica.

Definimos uma estrutura de DP com dois estados para cada posição i:

- $dp[i][0]$: número de mensagens possíveis terminando no caractere i **quando o dígito atual não é concatenado** com o anterior;
- $dp[i][1]$: número de mensagens possíveis terminando no caractere i **quando o dígito atual é concatenado** com o anterior.

A transição entre os estados é definida da seguinte forma:

- Se o dígito atual $s[i]$ está entre 1 e 9, ele pode ser interpretado como uma letra individual. Logo:

$$dp[i][0] = dp[i-1][0] + dp[i-1][1]$$

- Se a combinação dos dois últimos dígitos $s[i-1]s[i]$ forma um número entre 10 e 26, então esses dois dígitos podem ser interpretados como uma única letra:

$$dp[i][1] = dp[i-2][0] + dp[i-2][1]$$

A base da recorrência é:

$$dp[0][0] = 1, \quad dp[0][1] = 0, \quad dp[1][0] = 1, \quad dp[1][1] = 0$$

pois antes de processar qualquer caractere há exatamente uma maneira "vazia" de formar uma mensagem válida, e o primeiro dígito pode, no máximo, representar uma única letra isolada.

O resultado final é dado pela soma:

$$dp[n][0] + dp[n][1]$$

onde n é o comprimento da sequência de entrada.

Essa solução possui complexidade de tempo e espaço $O(n)$, podendo ser otimizada para $O(1)$ espaço se armazenarmos apenas os últimos dois estados necessários para o cálculo.