

Tutorial: Distância de Edição

1 Solução do Problema

O problema de encontrar o menor número de operações para transformar uma string em outra, classicamente conhecido como Distância de Edição (*Edit Distance*), pode ser resolvido de forma eficiente por meio de **programação dinâmica**.

1.1 Definição do Subproblema

Sejam w_1 e w_2 as duas strings de tamanhos N e M , respectivamente. Definimos o nosso estado da programação dinâmica como:

$dp[i][j]$ = número mínimo de operações para transformar o sufixo $w_1[i \dots N - 1]$ no sufixo $w_2[j \dots M - 1]$.

Assim, $dp[0][0]$ representará a resposta final para as duas strings inteiras.

1.2 Função de Transição

Para determinar $dp[i][j]$, olhamos para os caracteres atuais $w_1[i]$ e $w_2[j]$ e tomamos uma decisão.

Se os caracteres forem iguais ($w_1[i] == w_2[j]$), não precisamos realizar nenhuma operação. Apenas avançamos na análise de ambas as strings:

$$dp[i][j] = dp[i + 1][j + 1]$$

Se os caracteres forem diferentes ($w_1[i] \neq w_2[j]$), temos três opções de edição. Avaliamos o resultado de aplicar cada uma delas e escolhemos a que produz o menor custo final. Cada operação tem um custo de 1:

$$dp[i][j] = 1 + \min \begin{cases} dp[i + 1][j] & \text{(Remoção do caractere } w_1[i]) \\ dp[i + 1][j + 1] & \text{(Substituição de } w_1[i] \text{ por } w_2[j]) \\ dp[i][j + 1] & \text{(Inserção do caractere } w_2[j] \text{ em } w_1) \end{cases}$$

1.3 Casos Base

Os casos base ocorrem quando esgotamos os caracteres de uma das strings (ou seja, quando i ou j ultrapassam os limites):

Se consumirmos toda a string w_1 ($i \geq N$), mas ainda restarem caracteres em w_2 , a única forma de igualá-las é **inserir** todos os $M - j$ caracteres restantes:

$$dp[N][j] = M - j$$

Da mesma forma, se consumirmos toda a string w_2 ($j \geq M$), mas ainda restarem caracteres em w_1 , a única forma de igualá-las é **remover** todos os $N - i$ caracteres que sobraram:

$$dp[i][M] = N - i$$