

Tutorial: Maior Subsequência Palindrômica

1 Solução do Problema

O problema de encontrar a Maior Subsequência Palindrômica (*Longest Palindromic Subsequence* - LPS) pode ser resolvido de forma elegante utilizando **programação dinâmica em intervalos**. A ideia é analisar o problema de fora para dentro: comparamos os caracteres nas extremidades de uma substring e decidimos se eles farão parte do nosso palíndromo ou se devemos encolher o nosso intervalo de busca.

1.1 Definição do Subproblema

Seja S a nossa string original de tamanho N , indexada de 0 a $N-1$. Definimos o nosso estado da programação dinâmica como:

$dp[i][j]$ = o comprimento da maior subsequência palindrômica dentro da substring $S[i \dots j]$.

Assim, a resposta para o comprimento máximo em toda a string será encontrada em $dp[0][N-1]$, que engloba o intervalo completo do primeiro ao último caractere.

1.2 Função de Transição

Para calcular $dp[i][j]$, olhamos para os caracteres nas pontas do intervalo atual: $S[i]$ e $S[j]$. Temos dois cenários possíveis:

1. **Os caracteres são iguais** ($S[i] == S[j]$): Eles podem formar as pontas de um palíndromo. Nesse caso, ganhamos 2 de comprimento (um caractere de cada lado) e somamos ao melhor palíndromo que conseguimos formar no intervalo estritamente interno a eles.

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

2. **Os caracteres são diferentes** ($S[i] \neq S[j]$): Como eles são diferentes, não podem formar as pontas do mesmo palíndromo simultaneamente. Precisamos descartar um dos lados e ver qual escolha nos dá o melhor resultado. Calculamos ignorando o caractere da esquerda (reduzindo para $i+1$) ou ignorando o da direita (reduzindo para $j-1$).

$$dp[i][j] = \max(dp[i+1][j], dp[i][j-1])$$

Nota de implementação: Para garantir que $dp[i+1][j-1]$ ou as outras dependências já estejam calculadas quando precisarmos delas, os laços devem processar os intervalos do menor para o maior, ou preenchendo o i de trás para frente (de $N-1$ até 0) e o j de frente para trás (de $i+1$ até $N-1$).

1.3 Casos Base

Os casos base ocorrem para intervalos de tamanho 1 e intervalos inválidos (tamanho 0 ou negativo):

Todo caractere isolado é, por definição, um palíndromo de comprimento 1. Portanto, a diagonal principal da nossa matriz de estados recebe 1:

$$dp[i][i] = 1 \quad \text{para todo } 0 \leq i < N$$

Para intervalos onde o início ultrapassa o fim (o que pode ocorrer durante a transição de $dp[i+1][j-1]$ quando $j = i+1$), o comprimento é zero:

$$dp[i][j] = 0 \quad \text{para todo } i > j$$