Encontre o Tamanho da Menor String com Subsequências

Marcos Vinicius Silva

28 de Setembro de 2023

1 Introdução

Este documento descreve um código JavaScript que calcula o tamanho da menor string que contém duas strings, A e B, como subsequências. O código usa uma abordagem de programação dinâmica para encontrar o tamanho da menor string.

2 Código JavaScript

A seguir está o código JavaScript que realiza a tarefa:

```
function menorStringComoSubsequencia(A, B) {
  const m = A.length;
  const n = B.length;

  // Crie uma matriz para armazenar os resultados dos
    subproblemas
  const dp = new Array(m + 1).fill(0).map(() => new Array(n
    + 1).fill(0));

  // Preencha a matriz usando a abordagem de programa  o
    din mica
  for (let i = 1; i <= m; i++) {
    for (let j = 1; j <= n; j++) {
      if (A[i - 1] === B[j - 1]) {
        dp[i][j] = 1 + dp[i - 1][j - 1];
      } else {
        dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
      }
    }
  }

  // O tamanho da menor string    a soma dos tamanhos das
    duas strings menos o tamanho da LCS
```

```
  const menorTamanho = m + n - dp[m][n];
  return menorTamanho;
}
```

Listing 1: Código JavaScript

3 Aplicação de Programação Dinâmica

A programação dinâmica é uma técnica utilizada para resolver problemas que podem ser divididos em subproblemas menores e que apresentam sobreposição de subproblemas. Neste caso, estamos interessados em calcular o tamanho da menor string que contém as strings A e B como subsequências.

3.1 Matriz DP

O primeiro passo da abordagem de programação dinâmica é criar uma matriz DP (Programação Dinâmica) para armazenar os resultados de subproblemas menores. A matriz DP é uma estrutura bidimensional de tamanho $(m + 1) \times (n + 1)$, onde m é o comprimento da string A e n é o comprimento da string B. Cada célula da matriz DP, representada por $dp[i][j]$, contém o tamanho da subsequência comum mais longa (LCS) entre as primeiras i letras de A e as primeiras j letras de B.

3.2 Preenchimento da Matriz DP

O próximo passo é preencher a matriz DP. Isso é feito por meio de um loop duplo que itera sobre os índices i e j. A ideia principal é verificar se as letras correspondentes em A e B fazem parte da subsequência comum.

```
if (A[i - 1] === B[j - 1]) {
  dp[i][j] = 1 + dp[i - 1][j - 1];
} else {
  dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
}
```

Se a letra em A na posição $i-1$ for igual à letra em B na posição $j-1$, então incrementamos o valor em $dp[i][j]$ em 1 em relação ao valor em $dp[i-1][j-1]$.

Caso contrário, calculamos o máximo entre $dp[i-1][j]$ (ignorando a última letra de A) e $dp[i][j-1]$ (ignorando a última letra de B). Isso representa a escolha de ignorar uma das letras, de forma a encontrar a subsequência comum mais longa.

3.3 Cálculo do Tamanho da Menor String

Uma vez que a matriz DP está completamente preenchida, podemos calcular o tamanho da menor string que contém A e B como subsequências. Isso é feito

subtraindo o valor na última célula da matriz DP do somatório dos comprimentos de A e B.

```
const menorTamanho = m + n - dp[m][n];
```

O valor $dp[m][n]$ representa o tamanho da LCS entre A e B. Portanto, subtrair esse valor da soma dos comprimentos de A e B nos dá o tamanho da menor string que contém as duas como subsequências.

4 Utilização

Para usar o código, siga as etapas abaixo:

1. Forneça as strings A e B como entrada.
2. Chame a função `menorStringComoSubsequencia(A, B)` para calcular o tamanho da menor string.
3. O resultado será o tamanho da menor string que contém A e B como subsequências.

5 Conclusão

O código JavaScript fornece uma solução eficiente para encontrar o tamanho da menor string que possui as strings A e B como subsequências. Ele utiliza uma matriz DP para armazenar resultados intermediários e segue a abordagem de programação dinâmica para resolver o problema.