El Problema de la Mayor Subsecuencia Común

Daniel Abad Fundora
Sherlyn Ballestero Cruz

November 10, 2023

Resumen

En este trabajo abordamos el problema de la mayor subsecuencia común más larga entre dos o más secuencias consiste en encontrar la subsecuencia más larga que es común a todas ellas. Este problema posee aplicaciones prácticas en la comparación de datos, en la bioinformática, el procesamiento de lenguaje natural y la detección de plagio. Este problema puede ser resuelto para dos cadenas en una complejidad temporal O(m x n) donde m y n son las longitudes de las secuencia utilizando el método de programación dinámica. Algunas de las variantes de este algoritmo son el problema de la supersecuencia común más corta y el problema

Descripción del trabajo realizado

En el trabajo realizado comenzamos por definir el problema de la mayor subsecuencia común, luego explicaremos algunas de sus aplicaciones prácticas. A continuación explicaremos el algoritmo en cuestión, así como la complejidad temporal del mismo. Por último analizaremos algunas variantes del algoritmo.

Además, hemos creado un programa en Python con una sencilla interfaz gráfica usando la biblioteca tkinter, en la cual se puede hallar la subsecuencia común mayor entre dos cadenas, además, posee una sección para dados dos documentos determinar si puede haber ocurrido plagio (para ello hallamos la mayor subsecuencia común y comprobamos si es mayor que el 50%). Además tenemos una seccion en la que hallamos la mayor subsecuencia palindromica de una cadena y otra para hallar la mayor subsecuencia comun entre k cadenas. Este proyecto consta de dos partes: gui, que es donde está implementada la interfaz gráfica y lcs, que es donde están implementados los algoritmos que usaremos (la lógica del proyecto)

Definición del Problema

El problema de la mayor subsecuencia común consiste en encontrar la subsecuencia más larga que es común en un conjunto de secuencias. Aunque en la mayoría de los casos, solo se toman dos secuencias.

Definición de Subsecuencia

Una subsecuencia es una secuencia que se puede obtener de otra secuencia eliminando algunos elementos sin cambiar el orden de los elementos restantes. Por ejemplo, si tenemos la secuencia "ABCBDAB", algunas de sus subsecuencias son "AB", "BCD", "BDAB", etc.

Es importante destacar que este problema es diferente del problema del substring común más largo. A diferencia de los substrings, las subsecuencias no necesitan tener posiciones consecutivas en la secuencia original.

Aplicaciones Prácticas

El problema de LCS tiene muchas aplicaciones prácticas en diversos campos y contextos. Algunos ejemplos son:

- **Comparación de datos**: El problema de LCS forma la base de programas que comparan datos, como la utilidad diff, que se usa para encontrar las diferencias entre dos archivos de texto. También se utiliza en sistemas de control de revisión como Git para reconciliar múltiples cambios sobre archivos controlados de revisión.

- **Bioinformática**: El problema de LCS también tiene usos en bioinformática, donde se utiliza para comparar secuencias de ADN, ARN o proteínas y encontrar regiones similares o conservadas entre ellas. Estas regiones pueden tener implicaciones funcionales o evolutivas y ayudar a entender la relación entre diferentes organismos o especies.

- **Procesamiento del lenguaje natural**: El problema de LCS también se puede aplicar al procesamiento del lenguaje natural, donde se puede usar para medir la similitud entre dos textos o documentos. Por ejemplo, se puede usar para detectar plagio, resumir textos o evaluar la calidad de traducciones automáticas.

- **Detección de plagio**: La subsecuencia común más larga también se utliza para detectar plagio en documentos. Si tienes dos documentos diferentes, puedes utilizar LCS para encontrar la subsecuencia común más larga entre ellos y determinar si hay plagio

Algoritmo para dos cadenas

El problema de la subsecuencia común más larga (LCS, por sus siglas en inglés) se trata de encontrar una subsecuencia más larga que es común en dos se-

cuencias. Este problema puede ser resuelto mediante programación dinámica siguiendo los siguientes pasos:

1. **Definir una tabla**: El algoritmo utiliza una tabla bidimensional de tamaño (n+1) x (m+1), donde n y m son las longitudes de las dos secuencias. Cada celda de la tabla almacena la longitud de la subsecuencia común más larga entre los prefijos de las secuencias hasta esa posición.

2. **Inicializar la tabla**: El algoritmo inicializa la primera fila y la primera columna de la tabla con ceros, ya que no hay subsecuencia común entre una secuencia vacía y cualquier otra secuencia.

3. **Rellenar la tabla**: El algoritmo rellena el resto de la tabla siguiendo una regla recursiva: si los últimos caracteres de las secuencias son iguales, entonces la longitud de la subsecuencia común más larga es igual a la longitud de la subsecuencia común más larga sin esos caracteres más uno. Si los últimos caracteres son diferentes, entonces la longitud de la subsecuencia común más larga es igual al máximo entre la longitud de la subsecuencia común más larga sin el último carácter de la primera secuencia y la longitud de la subsecuencia común más larga sin el último carácter de la segunda secuencia. Es decir, para cada celda (i,j), se tiene que:

$$L[i,j] = \begin{cases} 0 & \text{si } i = 0 \text{ o } j = 0 \\ L[i-1,j-1] + 1 & \text{si } X[i] = Y[j] \\ \max(L[i-1,j], L[i,j-1]) & \text{si } X[i] \neq Y[j] \end{cases}$$

4. **Obtener el resultado**: El algoritmo termina cuando ha rellenado toda la tabla. La celda (n,m) contiene la longitud de la subsecuencia común más larga entre las dos secuencias completas. Para obtener la subsecuencia en sí, se puede hacer un recorrido inverso por la tabla, partiendo desde la celda (n,m) y siguiendo los pasos que se hicieron para rellenarla.

Correctitud del algoritmo

El algoritmo de LCS tiene una estructura óptima: el problema puede ser dividido en subproblemas más pequeños y simples, los cuales pueden ser divididos en subproblemas aún más simples, y así sucesivamente, hasta que la solución llega a ser trivial. Además, el problema de LCS también tiene subproblemas que se traslapan: las soluciones de subproblemas de nivel mayor utilizan soluciones de subproblemas menores.

Complejidad temporal del algoritmo

En términos de complejidad computacional, para el caso de dos secuencias de n y m elementos, el tiempo de ejecución para la programación dinámica es de O(n × m).

Variantes del algoritmo

Existen varias variantes del problema LCS que se han estudiado y para las cuales se han desarrollado algoritmos específicos. Algunas de estas variantes incluyen:

1. **Hallar la mayor subsecuencia común de un conjunto de k cadenas**: Para resolver el problema de la subsecuencia común más larga para un número constante de cadenas mayor que 2, se puede utilizar una generalización del algoritmo de programación dinámica para dos cadenas. La idea es definir una tabla multidimensional que almacene la longitud de la subsecuencia común más larga para cada subconjunto de las cadenas de entrada. La tabla se rellena siguiendo una regla recursiva similar a la del caso de dos cadenas, pero considerando todas las posibles combinaciones de los últimos caracteres de las cadenas. El algoritmo tiene una complejidad temporal de $O(n^k)$, donde n es la longitud máxima de las cadenas y k es el número de cadenas.

2. **Problema de la subsecuencia común más larga restringida**: Esta variante del problema LCS implica encontrar la subsecuencia común más larga sujeta a ciertas restricciones adicionales. Por ejemplo, podríamos querer encontrar la subsecuencia común más larga que también sea una secuencia palindrómica, o que cumpla con alguna otra propiedad. Este problema es más complejo que el problema LCS estándar, ya que debemos tener en cuenta las restricciones adicionales al construir la subsecuencia.

3. **Problema de la supersecuencia común más corta**: En esta variante, el objetivo es encontrar la supersecuencia más corta que contenga a ambas secuencias como subsecuencias. Una supersecuencia de una secuencia S es una secuencia que contiene a S como subsecuencia. Por ejemplo, si nuestras secuencias son "ABC" y "ACD", una posible supersecuencia común más corta es "ABCD". Este problema puede ser resuelto mediante técnicas similares a las utilizadas para el problema LCS, pero requiere un enfoque ligeramente diferente

4. **Problema de la subsecuencia creciente más larga**: Esta variante implica encontrar la subsecuencia más larga que está en orden ascendente. Por ejemplo, si nuestra secuencia es "ABBAC", la subsecuencia creciente más larga es "ABC". Este problema tiene aplicaciones en áreas como el análisis de secuencias genéticas y el reconocimiento de patrones. Existen algoritmos eficientes para resolver este problema, algunos de los cuales utilizan técnicas de programación dinámica similares a las utilizadas para el problema LCS

References

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press. Capítulo 15: Problemas de programación dinámica.

[2] Geeks for Geeks. Longest Common Subsequence (LCS) - GeeksforGeeks.

[3] Masek, W. J., & Paterson, M. S. (1980). A faster algorithm computing string edit distances. Journal of Computer and System Sciences, 20(1), 18-31.

[4] Wikipedia. Longest Common Subsequence