Proyecto final de la asignatura Diseño y Análisis de Algoritmos

Autor :
Amanda Noris Hernández

4to Año Ciencias de la Computación
Universidad de La Habana

September 22, 2024

1 Superstring Problem (SCS)

Una supercadena de un conjunto de cadenas $\{s_1, \ldots, s_n\}$ es una cadena que contiene cada s_i para $1 \leq i \leq n$ como una subcadena. El problema de las supercadenas es: Dado un conjunto S de cadenas y un entero positivo K, ¿tiene S una supercadena de longitud K? No se pierde ninguna generalidad definiendo S como un conjunto, porque si S es una colección de cadenas donde aparecen algunas cadenas más de una vez, entonces S tiene exactamente el mismo conjunto de supercadenas que el conjunto $S' = \{s : s \text{ está en } S\}$.

2 Demostración de pertenencia a NP

Para mostrar que el problema está en NP, debemos demostrar que, dada una posible solución T para el conjunto S, podemos verificar en tiempo polinomial si:

- T es una supercadena que contiene todas las cadenas $s_1, s_2, \ldots, s_n$ como subcadenas.
- La longitud de T es la mínima posible.

Verificación en tiempo polinomial:

Dada una solución candidata T, podemos verificar si T contiene a todas las cadenas $s_1, s_2, \ldots, s_n$ como subcadenas en tiempo polinomial. Para cada cadena s_i, verificamos si s_i aparece como subcadena en T. Esta verificación puede realizarse en tiempo $O(|T| \cdot |s_i|)$, utilizando un algoritmo como el de Knuth-Morris-Pratt (KMP).

Como hay n cadenas, la verificación de todas las cadenas en T toma tiempo polinomial. Por lo tanto, podemos verificar si una cadena T es una solución válida del problema SCS en tiempo polinomial, lo que implica que el problema está en NP.

3 Demostración de pertenencia a NP-Complete

Teorema El problema de las supercadenas es NP-completo. Además, este problema es NP-completo para cualquier entero $H \geq 3$, si se hace la restricción de que todas las cadenas en el conjunto sean primitivas y de longitud H.

3.1 Cadenas no primitivas de longitud 3

Sea $G = (V, E)$ una instancia del problema del camino de Hamilton dirigido restringido, donde V es el conjunto de números enteros del 1 al n (1 es el nodo inicial y n el nodo final) y $|E| = m$. Construimos cadenas para G sobre el alfabeto $Z = V \cup B \cup S$, donde $B = \{\overline{v} \mid v \in V - \{n\}\}$ y $S = \{e, \#, \$\}$ es el conjunto de símbolos especiales. En la reducción, los símbolos pertenecientes a B pueden considerarse como locales a un nodo, mientras que los símbolos de V se consideran globales para el conjunto gráfico G.

Para cada nodo $v \in V - \{n\}$ creamos un conjunto A_v de $2 \cdot \text{OUT}(v)$ cadenas. Sea $R_v = \{w_1, w_2, \ldots, w_{\text{OUT}(v)}\}$ el conjunto de nodos adyacentes a v. Entonces, $A_v = \{\overline{v} w_i \overline{v} \mid w_i \in R_v\} \cup \{w_i \overline{v} w_{i@1} \mid w_i \in R_v\}$, donde @ denota el módulo de suma OUT(v). Para cada nodo $v \in V - \{1, n\}$ creamos un conjunto unitario C_v que contiene una cadena de la forma $v\#\overline{v}$ llamada conector. Finalmente, creamos un conjunto T que contenga elementos llamados cadenas terminales: $T = \{e\#\overline{1}, n\#\$\}$.

Sea S la unión de A_j, $1 \leq j < n$, C_i, $1 \leq i < n$, y T. Demostremos que G tiene un camino de Hamilton dirigido si y solo si S tiene una supercadena de longitud $2m + 3n$.

Resultado: $2m + 3n$ es un límite inferior sobre el tamaño de una supercadena para S.

Demostración: Hay un total de $2m + n$ cadenas, con una longitud total de $3(2m + n)$. La mejor compresión resultaría de un orden en el cual cada cadena, excepto la primera y la última, tuviera un solapamiento de longitud 2 en ambos lados. Este orden daría una supercadena de longitud $3(2m + n) - 2(2m + n - 1) = 2m + n + 2$. Sin embargo, los $n - 2$ conectores solo pueden tener solapamientos de longitud 1 en cada lado, ya que ninguna cadena comienza o termina con #. Además, las cadenas terminales pueden solaparse como máximo en un símbolo en solo un lado. Debido a esto, obtenemos un límite inferior de $(2m + n + 2) + 2(n - 2) + 2 = 2m + 3n$ sobre la longitud de una supercadena para S. Esta supercadena debe comenzar con $e\#i$ y terminar con $n\#\$$.

Demostración hacia la derecha

Supongamos que G tiene un camino de Hamilton dirigido. Sea (v, w_i) una arista en el camino. Entonces primero creamos una supercadena de longitud $2 \cdot \text{OUT}(v) + 2$ para A_v, de la forma :

$$\overline{v}w_i\overline{v}w_{i@1}\overline{v}\ldots\overline{v}w_i,$$

llamada la w_i-supercadena estándar w_i para A_v. Esta supercadena se forma superponiendo las cadenas de A_v en el orden

$$\overline{v}w_i\overline{v}, w_i\overline{v}w_{i@1}, \overline{v}w_{i@1}\overline{v}, \ldots, \overline{v}w_{i@OUT(v)}\overline{v}, w_{i@OUT(v)}\overline{v}w_i$$

donde cada par sucesivo tiene una superposición de longitud 2. El conjunto de supercadenas estándar w_i para A_v está en correspondencia uno a uno con las permutaciones cíclicas de los enteros del 0 hasta $\text{OUT}(v) - 1$ ya que al superponer estas cadenas, se puede comenzar desde cualquier cadena y seguir el orden de los nodos adyacentes. Esto crea un ciclo, ya que al llegar al último nodo, puedes regresar al primero. Las supercadenas estándar para A_v son las únicas que tienen esta longitud tan corta porque al superponer cada cadena de manera óptima (con una superposición de longitud 2), se minimiza el tamaño de la cadena resultante. Cualquier otra combinación que no respete este tipo de superposición no podrá alcanzar la longitud mínima, ya que agregaría caracteres adicionales.

Sea $(u_1, u_2, \ldots, u_n)$ el camino de Hamilton dirigido, donde $u_1 = 1$ y $u_n = n$. Abreviaremos la u_j-supercadena estándar para A_{u_i} como $\text{STD}(\overline{u_i}, u_j)$. Podemos formar una supercadena para S superponiendo las supercadenas estándar y las cadenas en S que no están en ningún A_v en el orden:

$$e\#\overline{1}, \text{STD}(\overline{1}, u_2), u_2\#\overline{u_2}, \text{STD}(\overline{u_2}, u_3), u_3\#\overline{u_3}, \ldots, u_{n-1}\#\overline{u_{n-1}}, \text{STD}(\overline{u_{n-1}}, n), n\#\$.$$

Esta supercadena tiene longitud

$$\sum_{i=1}^{n-1}(2 \cdot \text{OUT}(i) + 2) + (n-2) + 4 = 2m + 3n.$$

donde la suma principal captura la longitud de todas las supercadenas estándar derivadas de los nodos intermedios, el término (n - 2) se refiere a la longitud de los conectores que conectan los nodos intermedios y el término 4 representa los símbolos especiales de inicio $e\#\overline{1}$ y final $n\#\$$ de la supercadena.

Demostración hacia la izquierda

Sea S una supercadena de longitud $2m+3n$ que comienza con $e\#\overline{1}$ y termina con $n\#\$$. Consideremos dos ocurrencias consecutivas de # en tal supercadena. Sea x la cadena entre los dos #. El primer símbolo de x debe estar pertenecer al conjunto B , y el último no, ya que son subcadenas de conectores. Dado que no hay conectores en x, todas las subcadenas de x excepto la primera y la última deben tener solapamientos de longitud dos en ambos lados. La primera cadena debe ser $\overline{v}u_j\overline{v}$, la siguiente $u_j\overline{v}u_{j@1}$, y así sucesivamente. Además, todas las cadenas en A_v, excepto dos, deben tener solapamientos de longitud 2 en ambos lados, por lo que cada cadena en A_v, salvo una, debe ser sucedida en orden por la cadena única que la solapa en 2. Así, todas las cadenas en A_v deben aparecer contiguamente en orden, y dado que x contiene una cadena de A_v, debe contenerlas todas. Por lo tanto, x es la supercadena w_j-estándar para A_v.

Al aplicar el análisis anterior a todos los pares secuenciales de # obtenemos $n-1$ cadenas estándar diferentes. Podemos entonces encontrar un camino de Hamilton dirigido observando los símbolos al lado de cada #, ya que el símbolo que pertenece a B y el que no de cada conector corresponden al mismo nodo en G. Por la ubicación de $e\#\overline{1}$ y $n\#\$$, el camino va del nodo 1 al nodo n.

3.2 Cadenas primitivas y de longitud H, para $H \geq 3$

Modificaciones para adecuar la demotración para este caso:

- El alfabeto Z se amplía para incluir $\{\hat{a} \mid a \in V\}$.
- Para $H = 3$, solo necesitamos cambiar A_v reemplazando las cadenas de la forma $\overline{v}a\overline{v}$ por las cadenas $\overline{v}a\hat{v}$, $a\hat{v}\hat{a}$ y $\hat{v}\hat{a}\overline{v}$. Además, hay que reemplazar las cadenas de la forma $a\overline{v}b$ por $\hat{a}\overline{v}b$. Cada una de estas nuevas cadenas tiene longitud 3, y al introducir los nuevos símbolos de Z, las cadenas se vuelven primitivas. Esto evita que sean repeticiones de una cadena más corta.
- Para $H \geq 4$, sean y y y' cadenas primitivas sobre un alfabeto disjunto de Z de longitud $H-4$ y $H-2$, respectivamente. Reemplace el símbolo # en todos los conectores y terminales por y'. Antes de esta modificación, los conectores con el símbolo # no cumplían con la longitud mínima

H, ya que # es solo un carácter. Al reemplazarlo por una cadena y' de longitud $H-2$, logramos que los conectores y terminales alcancen la longitud adecuada sin perder la estructura necesaria para la reducción. Además, y' es primitiva, por lo que la cadena completa también es primitiva. Para A_v, reemplace las cadenas de la forma $a\overline{v}b$ por $\hat{a}\hat{v}y\overline{v}b$. La cadena original $a\overline{v}b$ no cumplía con la longitud mínima de $H \geq 4$, ya que tenía una longitud de solo 3 caracteres. Al reemplazarla por $\hat{a}\hat{v}y\overline{v}b$, la longitud aumenta a H debido a los símbolos adicionales $\hat{a}$ y $\hat{v}$, y a la introducción de la cadena primitiva y, de longitud $H-4$. Por lo tanto, la nueva cadena cumple con la longitud mínima requerida H y mantiene las propiedades del problema.

3.3 Conclusiones

Dado que el problema del camino de Hamilton es NP-completo, y hemos demostrado que es equivalente al problema del SCS, concluimos que este último también es NP-completo.

4 Solución de fuerza bruta

Se puede pensar en un algoritmo para encontrar el shortest common superstring (SCS) utilizando un enfoque de fuerza bruta al generar todas las permutaciones posibles de una lista de cadenas. Para cada permutación, construye una supercadena combinando las cadenas en el orden dado, optimizando la combinación mediante la identificación del solapamiento máximo entre cada par de cadenas. Esto se logra evaluando cuántos caracteres de la segunda cadena coinciden con los finales de la primera, asegurando que se minimice la longitud total de la supercadena resultante. Al final, el algoritmo selecciona la supercadena más corta entre todas las permutaciones evaluadas y la devuelve como resultado.

4.1 Complejidad temporal

La generación de todas las permutaciones de las cadenas tiene una complejidad de $O(n!)$, donde n es la cantidad de cadenas, lo que vuelve esta propuesta sumamente impráctica aún sin analizar la evaluación de los solapamientos por cada permutación. Por esto consideremos otras soluciones

para este problema.

5 Solución aproximada

Dado un conjunto R de cadenas, un enfoque es desarrollar un algoritmo de aproximación greedy basado en la siguiente idea : encontrar y eliminar dos cadenas en el conjunto que tengan la mayor superposición mutua entre todos los pares posibles en R. Luego, formar la cadena superpuesta a partir de las dos cadenas eliminadas y volver a reemplazarla en R. Repetir este proceso hasta que solo quede una cadena en R o hasta que no haya dos cadenas con una superposición no vacía.

Los pasos principales se pueden resumir de la siguiente manera:

1. Calcular las superposiciones máximas por pares entre todas las cadenas en R.

2. Formar un R reducido eliminando todas las cadenas x que son subcadenas de alguna otra cadena. Esto se puede llevar a cabo junto con el paso 1. Reducir R no cambia el conjunto de las supercadenas comunes más cortas de R.

3. Usando heurísticas greedy, encontrar una aproximación al camino Hamiltoniano más largo desde el nodo inicial hasta el nodo final en el grafo para el R reducido. A partir del camino H encontrado de esta manera, construir una supercadena común para R.

5.1 Complejidad temporal

Paso 1: El problema que se debe resolver aquí es, dado dos cadenas x y x', determinar la longitud k de la superposición máxima entre x y x'. Si x' es una subcadena de x, entonces $k = |x'|$.

Este problema puede entenderse como una instancia del clásico problema de coincidencia de patrones donde x es el texto y x' es el patrón, y preguntamos si el patrón ocurre en el texto. Si el patrón no ocurre en el texto, ahora también preguntamos cuál es el sufijo más largo del texto que también es un prefijo del patrón. De esta manera, las superposiciones máximas entre cualquier par de cadenas pueden encontrarse en tiempo $O(|x| + |x'|)$ con el

algoritmo Knuth-Morris-Pratt (KMP). Se pueden encontrar todas las superposiciones por pares en un tiempo total de $O(mn)$.

Paso 2: Este paso se implementa de manera natural incorporando lo siguiente en el paso 1: cada vez que el paso 1 determina que x_i es una subcadena de x_j, se elimina x_i de R y se procede al siguiente x_i. Esto no aumenta el requisito de tiempo asintótico del paso 1.

Paso 3: Las heurísticas greedy para encontrar un camino Hamiltoniano aproximado se ejecutan encontrando repetidamente un arco con peso maximal r, que con los arcos seleccionados anteriormente, se pueda expandir a un camino Hamiltoniano desde el nodo inicial hasta el nodo final. Por lo tanto, está prohibido seleccionar un arco que ya no esté libre. Un arco (x, y) se llama libre si x no es el nodo inicial de algún arco seleccionado anteriormente y y no es el nodo final de algún arco seleccionado anteriormente, y que junto con los arcos seleccionados anteriormente no crea un ciclo dirigido.

Sea H el conjunto de arcos seleccionados para el camino Hamiltoniano. Inicialmente, está vacío. Procedemos de la siguiente manera:

1. Ordenar los arcos según el peso.

2. Escanear los arcos en orden decreciente. Para cada arco (x, y) encontrado, si (x, y)es libre, entonces

   (a) agregar (x, y) a H

   (b) marcar x como nodo inicial y y como nodo final (esto hace que todos los arcos restantes que comienzan en x o terminan en y no sean libres)

   (c) para encontrar los extremos del camino en H que contiene el nuevo arco (x, y), recorrer en H el camino desde y hasta que se encuentre el nodo final y', y recorrer el camino desde x en contra de la dirección de los arcos, hasta que se encuentre el nodo inicial x', y luego marcar el arco (y', x') como un arco que crea un ciclo.

El paso (1) se puede implementar de manera eficiente mediante bucket sort en tiempo $O(m^2)$. Utilizar bucket sort es razonable ya que los pesos a ordenar están en el rango limitado $0, \ldots, n$.

El paso (2) tiene complejidad $O(m^2)$ para escanear los arcos y para comprobar si son libres o no. Las marcas realizadas en los pasos (b) y (c) aseguran que la libertad de cada arco se pueda probar en tiempo constante. Por lo

tanto, el tiempo total para los pasos (b) y (c) es $O(m)$. Cada aplicación del paso (2.3) toma tiempo $O(l|H|)$, por lo que el tiempo total es $O(m^2)$. El tiempo total para el paso (A3) es, por lo tanto, $O(m^2) = O(mn)$. Finalmente, el string que buscamos se construye fácilmente a partir de H en tiempo $O(n)$.

Conclusión: El algoritmo en los pasos 1-3 se puede implementar de tal manera que su requisito de tiempo es $O(mn)$, donde m es el número de cadenas en R y n es la longitud total de las cadenas.

6 Implementación

Una implementación del primer algoritmo de fuerza bruta propuesto se puede encontrar en los archivos adjuntos.