CC40A
Complejidad de Kolmogorov

Ricardo Baeza Yates, Rodrigo Paredes

Dept. de Ciencias de la Computación, Universidad de Chile.

1. Definiciones Básicas

La Complejidad de Kolmogorov (CK) de un string $s \in \Sigma^*$ es el largo mínimo de un programa para una Máquina de Turing Universal U con entrada $\in \{0,1\}$ que genera s y se detiene (Figura 1).

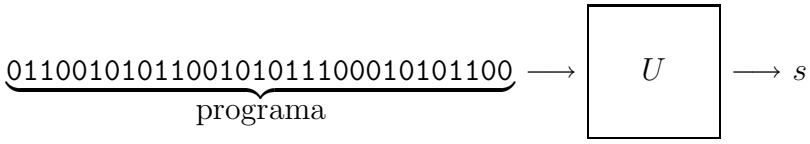


Figura 1. Máquina de Turing Universal que genera s a partir de un programa.

El largo depende de U, pero sólo en una constante aditiva.

Si la CK de un string s de largo n es cercana a n, se dice que s es aleatorio de Kolmogorov, es decir, no se puede comprimir.

Como hay a lo más 2^{n-1} programas binarios de largo $n-1$ o menos, claramente hay un string de largo n con CK al menos n. Este mismo argumento permite demostrar que para cada c y n, hay a lo más $2^{n-c+1}-1$ strings en Σ^n con CK $\leq n-c$.

Esto quiere decir que todos los strings menos una fracción de 2^i son casi aleatorios y no pueden ser comprimidos en más de c bits.

Este argumento se puede traducir a que si queremos demostrar que una propiedad $P(u)$ es cierta para un string u, tomamos un string aleatorio de Kolmogorov y suponemos que $P(u)$ es falsa, demostrando que si se puede escribir en forma concisa, lo que es una contradicción. Luego, $P(u)$ es cierta con alta probabilidad (Figura 2).

$$\text{String aleatorio } u \xrightarrow[|u| \leq |\text{descripción}(u)|]{\text{reescribir } u} \text{descripción de } u$$

Figura 2. Utilizando Kolmogorov para verificar una propiedad.

2. Ejemplo

2.1. Cálculo de *LCS* de dos strigs

La subsecuencia común más larga (LCS) de dos strings A y B, es la secuencia de letras más larga que estañ en A y B en el mismo orden en ambos strings. El LCS no es único. En la Figura 3 se muestra un ejemplo de LCS de dos strings A y B.

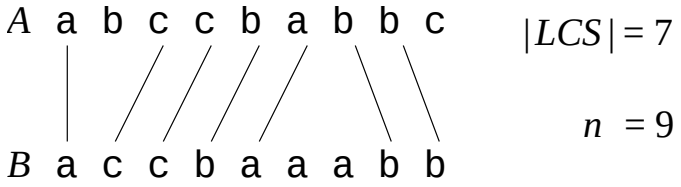


Figura 3. Ejemplo de LCS de dos strings A y B.

Como una motivación para este problema, consideremos la situación de tener dos secuencias de ADN de tamaño n: ADN_1 y ADN_2 a las que se le calcula el $LCS(ADN_1, ADN_2)$. Luego, en términos biológicos es significativo que el tamaño del $LCS(ADN_1, ADN_2)$ sea de un 75 % de n. La respuesta depende de si el largo promedio del LCS de dos secuencias de ADN aleatorias es (bastante) menor a 75 %.

Consideremos el cálculo del largo promedio del LCS de dos strings de tamaño n en un alfabeto de $k = |\Sigma|$ letras con distribución de frecuencias uniforme. Vamos a suponer que en promedio $|LCS| \approx \gamma n$.

Para describir los strings en base al LCS se necesita lo siguiente (se utiliza $\lg(\cdot) = \log_2(\cdot)$):

- valores de n y γn: $O(\lg n)$.
- la secuencia del LCS: $\gamma n \lg k$.
- las posiciones donde va el LCS: $2\lg(\binom{n}{\gamma n})$. En cada string hay γn letras en el LCS, luego hay $\binom{n}{\gamma n}$ posibles combinaciones para estas letras en cada string, luego los posibles calces entre ambos strigs, considerando calces hacia adelante y hacia atras, son $\binom{n}{\gamma n}^2$, como sólo interesan los calces hacia adelante, tenemos que las posibles posiciones son $\leq \binom{n}{\gamma n}^2$.
- las letras que sobran arriba y abajo: $2(1-\gamma)n\lg(k-1)$. Es $k-1$ (y no k) pues en cada segmento hay una letra que no puede estar (sino, sería parte del LCS).

Luego, la descripción de los strings en base al LCS suma:

$$\gamma n \lg k + 2\lg\left(\binom{n}{\gamma n}\right) + 2(1-\gamma)n\lg(k-1) \geq 2n\lg k \ , \tag{1}$$

y esta descripción la acotamos por el tamaño de los dos strings.

Para valores de n grande, y utilizando la aproximacón de Stirling ($n! \approx (\frac{n}{e})^n$), se puede resover $\lg(\binom{n}{\gamma n})$.

$$\lg\left(\binom{n}{\gamma n}\right) = \lg\left(\frac{n!}{(n-\gamma n)!(\gamma n)!)}\right) \tag{2}$$

$$\approx \lg\left(\frac{(\frac{n}{e})^n}{(\frac{n-\gamma n}{e})^{n-\gamma n}(\frac{\gamma n}{e})^{\gamma n}}\right) \tag{3}$$

$$\approx n\left(\lg n - (1-\gamma)\lg n - (1-\gamma)\lg(1-\gamma) - \gamma\lg\gamma - \gamma\lg n\right) \tag{4}$$

$$\approx n\underbrace{\left(-\gamma\lg\gamma - (1-\gamma)\lg(1-\gamma)\right)}_{\text{Entropía } H(\gamma)} \tag{5}$$

$$\lg\left(\binom{n}{\gamma n}\right) \approx nH(\gamma) \tag{6}$$

Reemplazando la Ecuación (6) en la Ecuación (1), se tiene:

$$2H(\gamma) + 2(1-\gamma)\lg(k-1) \geq (2-\gamma)\lg k\ . \tag{7}$$

Resolviendo numéricamente la Ecuación (7) para $k = 2$ (alfabeto binario) se obtiene $\gamma_2 \in [0.282, 0.867]$, y para $k = 4$ (ADN) se obtiene $\gamma_4 \in [0.102, 0.730]$. Los valores aproximados de γ para $k = 2$ y 4 son $\gamma_2 \approx 0{,}81$ y $\gamma_4 \approx 0{,}65$. En la Figura 4 se muestran las gráficas de la Ecuación (7) para $k = 2$ y 4.

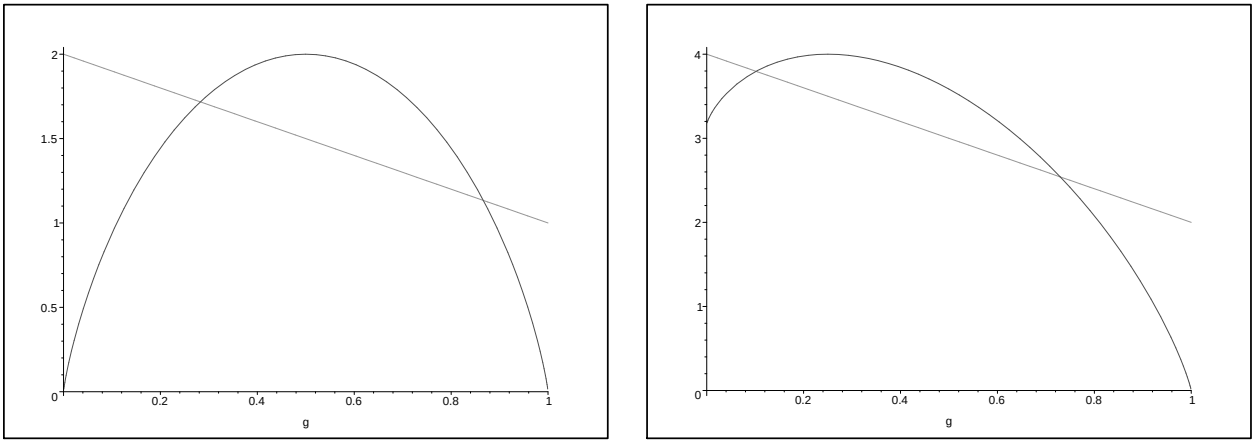


Figura 4. Resolución numérica de la Ecuación (7). A la izquierda, $k = 2$. A la derecha, $k = 4$.

2.2. Mejorando la cota inferior de γ

La cota inferior se puede mejorar con un programa que trata de encontrar una subsecuencia común larga. Este programa se puede codificar en un autómata, por lo que a medida que el programa considera más posibilidades el autómata resulta cada vez más complejo, pero a la vez, mejora la cota inferior de γ.

Para el caso binario, el programa más sencillo (que sólo verifica si hay calce en las letras actuales) se puede codificar en el autómata de la Figura 5, utilizando este autómata se obtiene una cota inferior de $\gamma \geq \frac{1}{2}$.

Si consideramos que el programa puede recordar la letra recién vista en ambos strings, podemos mejorar aún más la cota inferior de γ. Consideremos que tenemos dos strings,

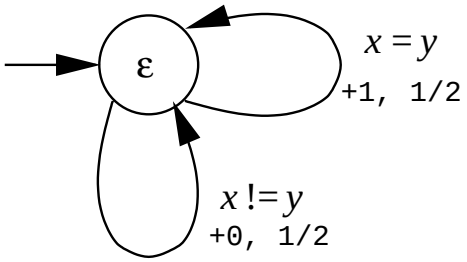
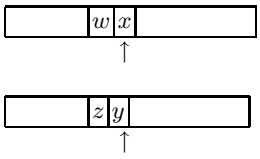


Figura 5. Autómata de 1 estado para encontar una subsecuencia común larga entre dos strings.

dos punteros que recorren cada uno de los strings, x e y son las letras actuales de cada strings respectivamente, w y z son las letras que se vieron anteriormente en cada string respectivamente y l el largo de la secuencia común.




```
while(quede string por leer)
  if x = y then
    l = l + 1, w ← NULL, z ← NULL
  else if x ≠ y then
    w ← x, z ← y
    avanza los punteros
    if w = y then l = l + 1, w ← x, z ← NULL
    else if z = x then l = l + 1, w ← NULL, z ← y
    else w ← x, z ← y
  avanza los punteros
```

Figura 6. Programa que calcula una subsecuencia común larga con un caracter de memoria.

Llevando el programa de la Figura 6 al autómata de la Figura 7, se obtiene $\gamma \geq \frac{9}{13} \approx 0{,}69$.

Aislando la información de las probabilidades se construye una cadena de Markov, cuya matriz de transición es T:

$$T = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/4 & 3/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}$$

Sea $\overrightarrow{p}_n$ el vector de la probabilidad que el autómata esté en cada estado en el paso n. Luego tenemos que $\overrightarrow{p}_0 = [1 \;\; 0 \;\; 0]$, y $\overrightarrow{p}_{n+1} = \overrightarrow{p}_n \cdot T$.

Cuando $n \to \infty$, tenemos la probabilidad estacionaria $\overrightarrow{p}_\infty = \overrightarrow{p}_\infty \cdot T$, luego:

$$\overrightarrow{p}_\infty (T - I) = 0 \tag{8}$$

$$\sum_i \overrightarrow{p}_{\infty i} = 1 \tag{9}$$

Del sistema de ecuaciones de (8) se obtienen dos ecuaciones independientes, y si consideramos la Ecuacion (9), tenemos un sistema de tres ecuaciones y tres incognitas ($\overrightarrow{p}_{\infty 1}$, $\overrightarrow{p}_{\infty 2}$

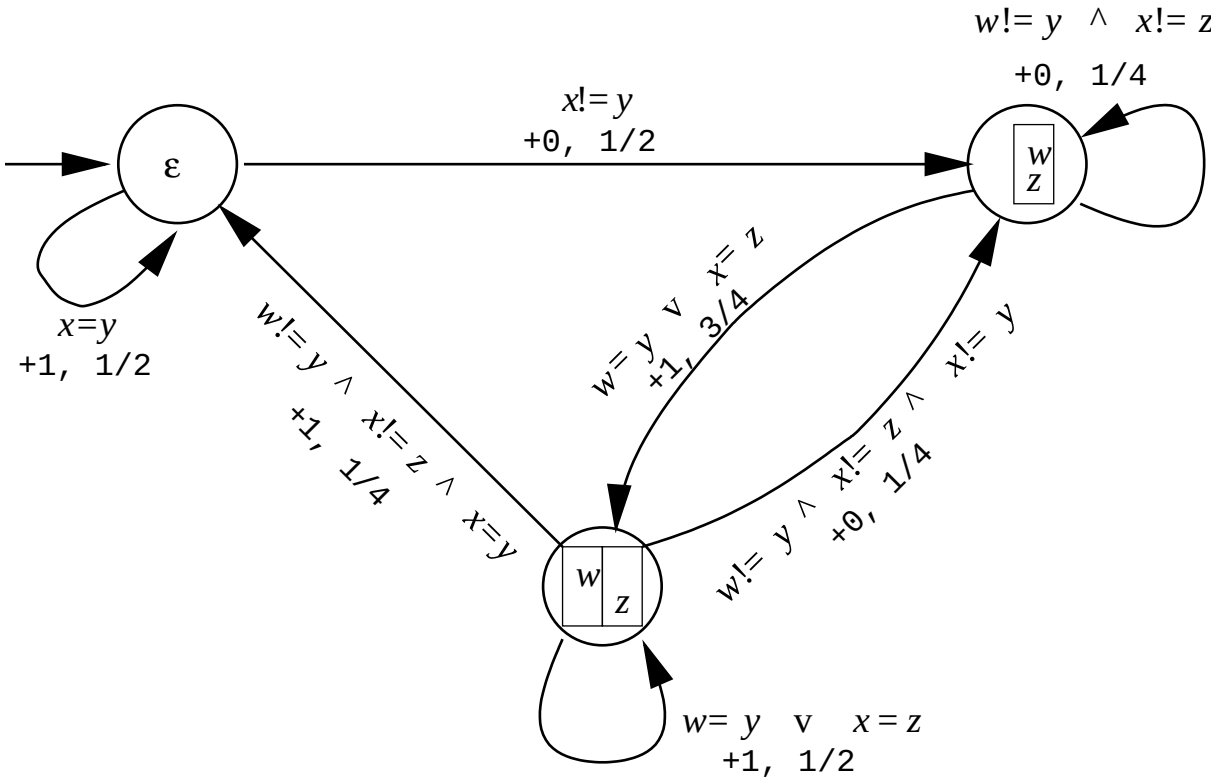


Figura 7. Autómata de 3 estados para encontar una subsecuencia común larga entre dos strings.

y $\overrightarrow{p}_{\infty 3}$). Con esto podemos escribir el siguiente sistema de ecuaciones:

$$\begin{bmatrix} -1/2 & 0 & 1/4 \\ 1/2 & -3/4 & 1/4 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \overrightarrow{p}_{\infty 1} \\ \overrightarrow{p}_{\infty 2} \\ \overrightarrow{p}_{\infty 3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Resolviendo este sistema se obtiene

$$\overrightarrow{p}_{\infty} = \begin{bmatrix} 3/13 & 4/13 & 6/13 \end{bmatrix}$$

Luego, el largo promedio aumenta en $3/13 . 1/2 + 4/13 . 3/4 + 6/13 . 3/4 = 9/13$.
Por lo tanto $\gamma_2 \geq 9/13 \approx 0.69$.