Algoritmos, Azar y Autómatas
Resolución Ejercicios 9 a 13 (Aleatoriedad)

Manuel Panichelli

December 8, 2021

Resultados previos

Def. 1 (Programa elegante)**.** Un programa es elegante si es el más corto que computa una secuencia. Es decir, $p \in \{0,1\}^*$ es un programa elegante si existe $s \in \{0,1\}^*$, $U(p) = s$ y para todo otro programa $p \in \{0,1\}^*$, si $U(p') = s$ entonces $|p| \leq |p'|$.

Llamamos s^* al programa elegante que computa s, $U(s^*) = s$ y $K(s) = |s^*|$.

Def. 2 (Abiertos básicos)**.** B_s es el conjunto de palabras infinitas que comienzan por s.

$$B_s = \{s\alpha \mid \alpha \in \{0,1\}^*\}$$

Prop. 1 (Medida de un abierto)**.** $\mu(B_s) = 2^{-|s|}$

Def. 3 (Test de Martin-Löf)**.** Una sucesión $(V_i)_{i>0}$ es un test de Martin-Löf si

- $V_i = \bigcup_{s \in S_i} B_s$ con $S_1 \subseteq 0,1^*$ computable
- $\mu(V_i) \leq 2^{-i}$ *(se va achicando)*

Si existe un test tal que $x \in \bigcap_{i \geq 1} V_i$, entonces x no es aleatorio.

Ejercicio 9

Demostrar que los números Martin-Löf aleatorios son normales.

Voy a demostrarlo por el contrarecíproco:

$$x \text{ no es normal } \Rightarrow x \text{ no es Martin-Löf aleatorio}$$

Primero listo algunas definiciones y propiedades que voy a necesitar solo para este ejercicio,

Def. 4 (Simple normalidad en una base)**.** $x \in A^*$ es simplemente normal en base b si para todo $w \in A^\ell$

$$\lim_{n\to\infty} \frac{|x[1\ldots n]||_w}{n} = \frac{1}{b^\ell}$$

Def. 5 (Malas palabras)**.** El conjunto de malas palabras para un alfabeto A es

$$\mathrm{Bad}(A,k,w,\varepsilon) = \left\{ v \in A^k : \left| |v|_w - \frac{k}{b^{|w|}} \right| > \varepsilon k \right\}$$

Prop. 2 (Suma geométrica infinita)**.** Sea $-1 < r < 1$, $r \neq 0$,

$$\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}$$

Dem. Sea x no normal. Para ver que no es Martin-Löf aleatorio tenemos que dar un Test de Martin-Löf $(V_i)_{i>0}$ tal que $x \in \bigcap_{i\geq 1} V_i$.

Como x no es normal, existe una base b tal que x no es normal en esa base.

$$\begin{aligned}
&\Rightarrow \exists\ w \text{ palabra tal que } \lim_{n\to\infty} \frac{|x[1\ldots n]|_w}{n} \neq b^{-|w|} \\
&\Rightarrow \exists\ w, \delta, k_0 \text{ tal que } \forall k > k_0,\ ||x[1\ldots k]|_w - b^{-|w|}| > \delta \\
&\Rightarrow \exists\ w, \delta, k_0 \text{ tal que } \forall k > k_0,\ x[1\ldots k] \in \mathrm{Bad}(A,k,w,\delta)
\end{aligned}$$

y observo que con δ fijo, $\forall \varepsilon \leq \delta$

$$\mathrm{Bad}(A,k,w,\delta) \subseteq \mathrm{Bad}(A,k,w,\varepsilon)$$

ya que si con una cota mas laxa una palabra resulta mala, con una condición más estricta también lo será.

Definimos el conjunto $S(i)$

$$\begin{aligned}
S(i) &= \bigcup_{k>f(i)} \bigcup_{u:|u|<g(i)} \mathrm{Bad}(A,k,u,\varepsilon(k)) \\
&= \text{palabras de longitud} > f(i) \text{ con mala frecuencia} \\
&\quad\ \text{con respecto a palabras de longitud} < g(i),
\end{aligned}$$

con f, g crecientes y ε decreciente. Definimos también el conjunto V_i a partir de él,

$$V_i = \bigcup_{s\in S(i)} B_s$$

que es casi el test pero podría suceder que x no pertenezca. Para solucionarlo, vamos a *shiftearlo*. Como x es mala, sabemos que $x[1\ldots k] \in \mathrm{Bad}(A,k,w,\delta)$. Luego, como f y g son crecientes, existe un i_0 a partir del cual

- $f(i_0) > k_0$ (estamos considerando palabras lo suficientemente grandes), y
- $g(i_0) > |w|$ (estamos considerando longitudes lo suficientemente grandes).

y por lo tanto, $\forall i > i_0$, $x \in V_{i+i_0}$. Definimos entonces el test a partir de ese i_0,

$$W_i = V_{i_0+i}.$$

Tiene que cumplir dos cosas,

1. $x \in \bigcap_{i>0} W_i$.
2. $\mu\left(W_i\right) = \mu\left(V_i\right) \leq 2^{-i}$

Como sabemos que 1. se cumple por construcción de W_i y elección de i_0, solo nos queda elegir f, g y ε tales que se cumpla 2. Veamos la medida de cada conjunto,

$$\begin{aligned}
\mu\left(W_i\right) &= \mu\left(V_i\right) \\
&= \mu\left(\bigcup_{s \in S(i)} B_s\right) \\
&\leq \sum_{s \in S(i)} \mu\left(B_s\right) \\
&= \sum_{s \in S(i)} 2^{-|s|} \\
&< \sum_{s \in S(i)} 2^{-f(i)} \\
&= |S(i)|\, 2^{-f(i)} < 2^{-i} \iff |S(i)| < 2^{f(i)-i}
\end{aligned}$$

Para k suficientemente grande,

$$|\mathrm{Bad}(A, k, w, \varepsilon(k))| < \varepsilon(k) b^k \tag{1}$$

Y tengo que

$$
\begin{aligned}
|S(i)| &= \left| \bigcup_{k>f(i)} \bigcup_{u:|u|<g(i)} \mathrm{Bad}(A,k,u,\varepsilon(k)) \right| \\
&\leq \sum_{k>f(i)} \sum_{u:|u|<g(i)} |\mathrm{Bad}(A,k,u,\varepsilon(k))| \\
&< \sum_{k>f(i)} \sum_{u:|u|<g(i)} \varepsilon(k)2^k && (1) \\
&< \sum_{k>f(i)} g(i)\varepsilon(k)2^k \\
&< \sum_{k>0} g(i)\varepsilon(k)2^k \\
&= \sum_{k>0} g(i)(1/2)^k \cancel{2^{-k}}\, \cancel{2^k} \\
& && (\varepsilon(k) = (1/2)^k 2^{-k}) \\
&= \frac{g(i)}{1-1/2} = 2g(i) && (\text{Prop. } 2)
\end{aligned}
$$

Juntando, quiero encontrar g y f tales que $2g(i) < 2^{f(i)-i}$. Tomando $f(i) = 2i$ y $g(i) = i/2$, tengo que

$$
\begin{aligned}
2g(i) < 2^{f(i)-i} &\iff 2i/2 < 2^{2i-i} \\
&\iff i < 2^i,
\end{aligned}
$$

lo cual es cierto. Por lo tanto, estas elecciones de funciones hacen que el test W_i cumpla con la restricción 2,

- $f(n) = 2n$ que es creciente
- $g(n) = n/2$ que es creciente
- $\varepsilon(n) = (1/2)^n 2^{-n} = (1/4)^n$ que es decreciente.

Finalmente, como existe un test de Martin-Löf que contiene a x, no es aleatoria. Por lo tanto, como acabo de demostrar que cuando x no es normal tampoco es aleatorio, concluyo que los números Martin-Löf aleatorios son normales. $\square$

Ejercicio 10

Dar un algoritmo que permite computar Ω con un oráculo para el problema de la detención.

Recuerdo la definición de Ω, la suma de las potencias de 2 de las longitudes de todos los programas que terminan. Si U es una máquina de Turing universal,

$$\Omega = \sum_{U(p)\downarrow} 2^{-|p|}$$

Para computar Ω voy a dar una función que computa los primeros n dígitos, $\Omega[1 \dots n]$, y luego voy incrementando n e imprimiendo los resultados.

Sea g una enumeración de todos los programas que terminan (se podría obtener computablemente mediante un método como *dovetailing*), defino una aproximación de Ω hasta el m-ésimo programa,

$$\alpha_m = \sum_{j=1}^{m} 2^{-|g(j)|}$$

Lo que me gustaría saber es para qué m α_m tiene los primeros n dígitos *definitivos*, es decir $\Omega[1 \dots n] = \alpha_m[1 \dots n]$. Para ello, debo verificar que

$$\alpha_m[1 \dots n] \stackrel{?}{=} \alpha'_m[1 \dots n] \ \forall m' > m$$

Es decir, no importa que sigamos considerando más programas, los primeros n dígitos no van a cambiar. Como no es algo finito, no lo podemos computar con un algoritmo, pero acá es donde nos salva el oráculo de Halt. La siguiente función logra lo buscado

function Q(m, n)
  $m' \leftarrow m + 1$
  while true **do**
    if $\alpha_m[1 \dots n] \neq \alpha'_m[1 \dots n]$ **then**
      break
    $m' \leftarrow m' + 1$

- Si `OraculoHalt(Q(m, n)) = true` (es decir, Q termina) es porque existía m tal que cambiaban los primeros n dígitos, y por lo tanto no eran definitivos.
- Si `OraculoHalt(Q(m, n)) = false` (es decir, Q *no* termina), entonces no existe m tal que cambien los primeros n dígitos, y por lo tanto son definitivos.

El algoritmo final es el siguiente, donde la función sin argumentos Print Ω imprime Ω segmento inicial por segmento inicial.

function PRINT Ω
  for $n = 1, 2 \dots$ **do**
    print Ω(n)
function Ω(n)
  for $m = 1, 2 \dots$ **do**
    if $\neg$ OraculoHalt(Q(m, n)) **then**
      return $\alpha_m[1 \dots n]$
function Q(m, n)

$m' \leftarrow m + 1$
while true **do**
  if $\alpha_m[1 \ldots n] \neq \alpha'_m[1 \ldots n]$ **then**
    break
  $m' \leftarrow m' + 1$

Ejercicio 11

Salteado porque no era necesario resolverlo.

Ejercicio 12

12.1

Demostrar que el número $(1 - \Omega)$ *es aleatorio*

Lema 1. Con los bits de x puedo obtener los de $\bar{x} = 1 - x$ realizando un xor con todos 1s, $x \oplus 1s = \bar{x}$.

Dem. Sea $g : \mathbb{N} \to \Sigma^*$ una enumeración de los programas que se detienen, $\alpha_m = \sum_{j=1}^{m} 2^{-|g(j)|}$ una aproximación de Ω de m pasos. Defino el programa p,

$$p = b_1 b_2 \ldots b_c \bar{\Omega}[1 \ldots i]^*$$

donde $\bar{\Omega}[1 \ldots i]^*$ es una subrutina que es el programa elegante (el más corto) que computa $\bar{\Omega}[1 \ldots i]$ y $b_1 b_2 \ldots b_c$ realiza los siguientes pasos,

0. Computamos $\bar{\Omega}[1 \ldots i]$ con $\bar{\Omega}[1 \ldots i]^*$
1. Computamos $\Omega[1 \ldots i]$ en base a $\bar{\Omega}[1 \ldots i]$ mediante un $\oplus$ con todos 1s.
2. $m = 1$. Mientras $(\alpha_m \leq \Omega[1 \ldots i])$, $m = m + 1$.
3. Sea $O = \{U(g(j)) \mid 1 \leq j \leq m\}$ los outputs de los programas que se detienen que aportan a la aproximación de Ω
4. Sea s la cadena más chica lexicográficamente tal que $s \notin O$.
5. Return s.

Veamos que $|s^*| > i$, la longitud del programa elegante que computa s (su complejidad) es más chica que i. Para demostrarlo supongamos lo contrario, que $|s^*| \leq i$

$$\begin{aligned}
\Omega &> 2^{-|s^*|} + \alpha_m && (\Omega \text{ tiene infinitos aportes}) \\
&> 2^{-|s^*|} + \Omega[1 \ldots i] && (\text{porque } \alpha_m > \Omega[1 \ldots i]) \\
&\geq 2^{-i} + \Omega[1 \ldots i] && (\text{sup. } |s^*| \leq i) \\
&\geq \Omega && (\Omega[1 \ldots i] \text{ contiene exactamente los primeros } i \text{ bits de } \Omega)
\end{aligned}$$

Y llegamos a $\Omega > \Omega$ que es un absurdo. Por lo tanto, $|s^*| > i$.

Por otro lado, como p es un programa (no muy bueno) que computa s, se que

$$K(s) \leq c + |\bar{\Omega}[1 \dots i]^*| \tag{2}$$

donde $c = |b_1 \dots b_c|$.

Juntando,

$$\begin{aligned} i &< |s^*| \\ &= K(s) \\ &\leq c + |\bar{\Omega}[1 \dots i]^*| && \text{por (4)} \\ &= c + K(\bar{\Omega}[1 \dots i]) \end{aligned}$$

$\iff K(\bar{\Omega}[1 \dots i]) > i - c$, que es la definición de aleatoriedad de Chaitin.

□

12.2

Para todo conjunto X infinito y c.e pero no computable, $\alpha = \sum_{x \in X} 2^{-x}$ no es aleatorio.

Def. 6 (c.e)**.** X es c.e (computablemente enumerable) si $\exists f : \mathbb{N} \to \mathbb{N}$ tal que $Im(f) = X$. Es decir, f enumera los elementos de X en algún orden que no se puede elegir.

Ejemplo 1. $X = \{2, 5\}$. Puedo tomar $f(1) = 5, f(2) = 2$ o $f'(1) = 2, f'(2) = 5$.

Ejemplo 2. $X = \{n \mid n \text{ es impar}\} = \{1, 3, 5, 7, \dots\}$

Puedo enumerarlos en cualquier orden,

$$f(1) = 7, f(2) = 17, f(3) = 37 \dots,$$

y la función característica es

$$X = 101010 \dots$$

Def. 7 (Función característica)**.** Puedo codificar un conjunto con una sucesión de bits correspondientes a su función característica.

$$X = b_1 b_2 b_3 b_4 b_5 \dots$$

con

$$b_i = \begin{cases} 1 & \text{si } i \in X \\ 0 & \text{sino.} \end{cases}$$

Por ejemplo

$$X = \{1, 4, 10\} = \underset{1}{1}\,\underset{2}{0}\,\underset{3}{0}\,\underset{4}{1}\,\underset{5}{0}\,\underset{6}{0}\,\underset{7}{0}\,\underset{8}{0}\,\underset{9}{0}\,\underset{10}{1}$$

Obs. La expansión en base 2 de α es la codificación en bits de X. Por ejemplo, con $X = \{1, 4, 10\}$,

$$\alpha = 2^{-1} + 2^{-4} + 2^{-10} = 0.5634765625 = (0.1001000001)_2.$$

Dem. Para ver que α no es aleatorio voy a dar un test de Martin-Löf $(V_i)_{i>0}$ tal que $\alpha \in \bigcap_{i>0} V_i$.

- Supongo que $f(1) = 3$, sabemos que $X = b_1\ b_2\ 1\ b_3\ b_4\ b_5 \ldots$, y defino V_1 tal que los primeros 3 dígitos de la expansión de α aparezcan.

  $$V_1 = \{B_{b_1 b_2 1} \mid b_1, b_2 \in \{0, 1\}\},$$

  con B_s las palabras infinitas que comienzan por s. Y su medida es correcta,

  $$\mu(V_1) = 2^2 \times \mu(B_{111}) = 2^2 \times 2^{-3} = 1/2 \le 2^{-1}$$

- Supongo que $f(2) = 10$, sabemos que $X = b_1\ b_2\ 1\ b_3\ \ldots b_9\ 1\ b_{11}\ \ldots$. Defino

  $$V_2 = \{B_{b_1 \ldots b_{10}} \mid b_3 = b_{10} = 1 \text{ y los demás } b_j \in \{0, 1\}\},$$

  Su medida también es correcta,

  $$\mu(V_2) = 2^{10-2} \times \mu\left(B_{s \text{ con } |s|=10}\right) = 2^{10-2} \times 2^{-10} = 2^{-2} \le 2^{-2}$$

En general, si $f(i) = x$

$$V_i = \{B_{b_1 \ldots b_x} \mid b_{f(k)} = 1 \text{ con } k \le i \text{ y el resto } b_j \in \{0, 1\}\},$$

luego

$$\begin{aligned}
\mu(V_i) &= 2^{x-i} \times \mu\left(B_{s \text{ con } |s|=x}\right) \\
&= 2^{x-i} \times 2^{-x} \\
&= 2^{\cancel{x}-i-\cancel{x}} \\
&= 2^{-i} \le 2^{-i}
\end{aligned}$$

y se que $\alpha \in \bigcap_{i\ge1} V_i$ pues cada V_i va refinando segmentos iniciales más grandes de α. Por lo tanto, como existe un test de Martin-Löf que contiene a α, α no es aleatorio. □

12.3

El número Ω_0 que resulta de anteponer mil 0s delante de Ω es aleatorio.

Lema 2. Como $\Omega_0 = 0^{1000}\Omega$, entonces

$$\begin{aligned}\Omega_0[1\dots j] &= (0^{1000}\Omega)[1\dots j] \\ &= 0^{1000}\Omega[1\dots j-1000] \\ &= 0^{1000}\Omega[1\dots i] && (i = j - 1000)\end{aligned}$$

A partir de $\Omega_0[1\dots j]$ puedo calcular $\Omega[1\dots i]$ trivialmente.

Dem. Sea $g : \mathbb{N} \to \Sigma^*$ una enumeración de los programas que se detienen, $\alpha_m = \sum_{j=1}^{m} 2^{-|g(j)|}$ una aproximación de Ω de m pasos. Defino el programa p,

$$p = b_1 b_2 \dots b_c \Omega_0[1\dots j]^*$$

donde $\Omega_0[1\dots j]^*$ es una subrutina que es el programa elegante (el más corto) que computa $\Omega_0[1\dots j]$ y $b_1 b_2 \dots b_c$ realiza los siguientes pasos,

0. Computamos $\Omega_0[1\dots j]$ con $\Omega_0[1\dots j]^*$
1. Computamos $\Omega[1\dots i]$ en base a $\Omega_0[1\dots j]$ (Lema 2).
2. $m = 1$. Mientras $(\alpha_m \leq \Omega[1\dots i])$, $m = m + 1$.
3. Sea $O = \{U(g(j)) \mid 1 \leq j \leq m\}$ los outputs de los programas que se detienen que aportan a la aproximación de Ω
4. Sea s la cadena más chica lexicográficamente tal que $s \notin O$.
5. Return s.

Veamos que $|s^*| > i$, la longitud del programa elegante que computa s (su complejidad) es más chica que i. Para demostrarlo supongamos lo contrario, que $|s^*| \leq i$

$$\begin{aligned}\Omega &> 2^{-|s^*|} + \alpha_m && (\Omega \text{ tiene infinitos aportes}) \\ &> 2^{-|s^*|} + \Omega[1\dots i] && (\text{porque } \alpha_m > \Omega[1\dots i]) \\ &\geq 2^{-i} + \Omega[1\dots i] && (\text{sup. } |s^*| \leq i) \\ &\geq \Omega && (\Omega[1\dots i] \text{ contiene exactamente los primeros } i \text{ bits de } \Omega)\end{aligned}$$

Y llegamos a $\Omega > \Omega$ que es un absurdo. Por lo tanto, $|s^*| > i$.

Por otro lado, como p es un programa (no muy bueno) que computa s, se que

$$K(s) \leq c + |\Omega_0[1\dots j]^*| \tag{3}$$

donde $c = |b_1 \dots b_c|$.
Juntando,

$$
\begin{aligned}
i &< |s^*| \\
&= K(s) \\
&\leq c + |\Omega_0[1 \dots j]^*| && \text{por (3)} \\
&= c + K(\Omega_0[1 \dots j]) \\
&\iff K(\Omega_0[1 \dots j]) > i - c = j - 1000 - c
\end{aligned}
$$

y tomando $c' = 1000 - c$, llegamos a

$$K(\Omega_0[1 \dots j]) > j - c',$$

que es la definición de aleatoriedad de Chaitin.

$\square$

12.4

Demostrar que $\alpha = \sum_{palabra\ s} 2^{-K(s)}$ es computablemente aproximable desde abajo y aleatorio.

Prop (α es aproximable desde abajo)**.** Primero, observo que α es la probabilidad de que un programa sea elegante. Defino el conjunto

$$S(t) = \begin{Bmatrix} \text{programas que terminan en menos de } t \\ \text{pasos y son candidatos a ser elegantes} \end{Bmatrix}.$$

Si un programa p está en $S(t)$ y no está en $S(t+1)$, es porque hay otro programa p' tal que $|p| < |p'|$ (que tarda más pasos en terminar, pero es más corto). Con esto puedo definir

$$\alpha_t = \sum_{s \in S(t)} 2^{-|s|}.$$

Como en cada paso t los programas se hacen más cortos, $\alpha_t < \alpha_{t+1} < \alpha$ (pues son potencias negativas). Por lo tanto, alpha se puede aproximar computacionalmente de forma *estrictamente* creciente desde abajo.

Dem. (12.4) Sea α_t una aproximación por abajo estrictamente creciente de t pasos de α. Defino el programa p,

$$p = b_1 b_2 \dots b_c \alpha[1 \dots i]^*,$$

donde $\alpha[1 \dots i]^*$ es una subrutina que contiene el programa elegante (el más corto) que computa $\alpha[1 \dots i]$ y $b_1 b_2 \dots b_c$ realiza los siguientes pasos,

1. Computa $\alpha[1 \dots i]$ en base a $\alpha[1 \dots i]^*$

2. $t = 1$. Mientras $(\alpha_t \leq \alpha[1 \dots i])$, $t = t + 1$.
3. Sea $O = \{U(s) \mid s \in S(t)\}$ los outputs de los programas candidatos a elegantes que aportan a la aproximación de α
4. Sea s la cadena más chica lexicográficamente tal que $s \notin O$.
5. Return s.

Veamos que $|s^*| > i$, la longitud del programa elegante que computa s (su complejidad) es más chica que i. Para demostrarlo supongamos lo contrario, que $|s^*| \leq i$

$$\begin{aligned}
\alpha &> 2^{-|s^*|} + \alpha_t && (\alpha \text{ tiene infinitos aportes}) \\
&> 2^{-|s^*|} + \alpha[1 \dots i] && (\text{porque } \alpha_t > \alpha[1 \dots i]) \\
&\geq 2^{-i} + \alpha[1 \dots i] && (\text{sup. } |s^*| \leq i) \\
&\geq \alpha && (\alpha[1 \dots i] \text{ contiene exactamente los primeros } i \text{ bits de } \alpha)
\end{aligned}$$

Y llegamos a $\alpha > \alpha$ que es un absurdo. Por lo tanto, $|s^*| > i$.

Por otro lado, como p es un programa (no muy bueno) que computa s, se que

$$K(s) \leq c + |\alpha[1 \dots i]^*| \tag{4}$$

donde $c = |b_1 \dots b_c|$.

Juntando,

$$\begin{aligned}
i &< |s^*| \\
&= K(s) \\
&\leq c + |\alpha[1 \dots i]^*| && \text{por (4)} \\
&= c + K(\alpha[1 \dots i])
\end{aligned}$$

$\iff K(\alpha[1 \dots i]) > i - c$, que es la definición de aleatoriedad de Chaitin.

$\therefore \alpha$ es aleatorio. $\square$

Ejercicio 13

Dar un test de Martin Löf que contenga a los números decimales cuya expansión decimal no contiene el 7.

Dem.

Lema 3 (Contrarecíproco del ej. 9)**.**

$$\underbrace{\left(\begin{matrix}x \text{ es Martin-Löf}\\ \text{aleatorio}\end{matrix} \Rightarrow x \text{ es normal.}\right)}_{\text{Ejercicio 9}} \Longrightarrow \left(x \text{ no es normal.} \Rightarrow \begin{matrix}x \text{ no es}\\ \text{Martin-Löf}\\ \text{aleatorio}\end{matrix}\right)$$

Sea x un número cuya expansión decimal no contiene el 7. Como la frecuencia de todos los dígitos no es la misma, no es normal. Por lo tanto, por el Lema 3 tampoco es Martin-Löf aleatorio y por definición existirá un test en el que esté contenido. $\square$

Perdón Vero por resolverlo de esta manera, pero ya me estaba extendiendo mucho con el plazo de entrega!