

# Algoritmos, Azar y Autómatas

## Resolución Ejercicios 9 a 13 (Aleatoriedad)

Manuel Panichelli

December 5, 2021

### Resultados previos

**Def. 1** (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. 2** (Test de Martin-Löf). Una sucesión  $(V_i)_{i \geq 0}$  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.

TODO: programa elegante, complejidad de kormogolov, definiciones de martin lof y etc.

### Ejercicio 9

*Demostrar que los números Martin-Löf aleatorios son normales.*

### Ejercicio 10

*Dar un algoritmo que permite computar  $\Omega$  con un oráculo para el problema de la detención.*

Recuerdo la definición de  $\Omega$ , 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  $\Omega$  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  $\Omega$  hasta el  $m$ -ésimo programa,

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

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

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

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' ← m + 1
  while true do
    if  $\alpha_m[1 \dots n] \neq \alpha_{m'}[1 \dots n]$  then
      break
    m' ← m' + 1
```

- Si  $\text{OraculoHalt}(Q(m, n)) = \text{true}$  (es decir, Q termina) es porque existía  $m$  tal que cambiaban los primeros  $n$  dígitos, y por lo tanto no eran definitivos.
- Si  $\text{OraculoHalt}(Q(m, n)) = \text{false}$  (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  $\Omega$  imprime  $\Omega$  segmento inicial por segmento inicial.

```
function PRINT  $\Omega$ 
  for  $n = 1, 2 \dots$  do
    print  $\Omega(n)$ 
function  $\Omega(n)$ 
  for  $m = 1, 2 \dots$  do
    if  $\neg \text{OraculoHalt}(Q(m, n))$  then
      return  $\alpha_m[1 \dots n]$ 
function Q(m, n)
  m' ← m + 1
  while true do
    if  $\alpha_m[1 \dots n] \neq \alpha_{m'}[1 \dots n]$  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,  $x \oplus 1s = \bar{x}$ .

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

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

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

0. Computamos  $\bar{\Omega}[1 \dots i]$  con  $\bar{\Omega}[1 \dots i]^*$
1. Computamos  $\Omega[1 \dots i]$  en base a  $\bar{\Omega}[1 \dots i]$  mediante un  $\oplus$  con todos 1s.
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  $\Omega$
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

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

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

Juntando,

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

$\iff K(\bar{\Omega}[1 \dots i]) > i - c$ , 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. 3** (c.e).  $X$  es c.e (computablemente enumerable) si  $\exists f : \mathbb{N} \rightarrow \mathbb{N}$  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. 4** (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

$$X = \{1, 4, 10\} = \begin{matrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{matrix}$$

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

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

*Proof.* Para ver que  $\alpha$  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  $X = b_1 b_2 1 b_3 b_4 b_5 \dots$ , y defino  $V_1$  tal que los primeros 3 dígitos de la expansión de  $\alpha$  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,

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

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

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

Su medida también es correcta,

$$\mu(V_2) = 2^{10-2} \times \mu(B_s \text{ con } |s|=10) = 2^{10-2} \times 2^{-10} = 2^{-2} \leq 2^{-2}$$

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

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

luego

$$\begin{aligned} \mu(V_i) &= 2^{x-i} \times \mu(B_s \text{ con } |s|=x) \\ &= 2^{x-i} \times 2^{-x} \\ &= 2^{-i} \\ &= 2^{-i} \leq 2^{-i} \end{aligned}$$

y se que  $\alpha \in \bigcap_{i>1} V_i$  pues cada  $V_i$  va refinando segmentos iniciales más grandes de  $\alpha$ . Por lo tanto, como existe un test de Martin-Löf que contiene a  $\alpha$ ,  $\alpha$  no es aleatorio.  $\square$

### 12.3

El número  $\Omega_0$  que resulta de anteponer mil 0s delante de  $\Omega$  es aleatorio.

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

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

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

*Proof.* Sea  $g : \mathbb{N} \rightarrow \Sigma^*$  una enumeración de los programas que se detienen,  $\alpha_m = \sum_{j=1}^m 2^{-|g(j)|}$  una aproximación de  $\Omega$  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  $\Omega$
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

$$K(s) \leq c + |\Omega_0[1 \dots j]^*| \quad (2)$$

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

$$\begin{aligned}
i &< |s^*| \\
&= K(s) \\
&\leq c + |\Omega_0[1 \dots j]^*| && \text{por (2)} \\
&= c + K(\Omega_0[1 \dots j]) \\
&\iff K(\Omega_0[1 \dots j]) > i - c = j - 1000 - c
\end{aligned}$$

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

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

que es la definición de aleatoriedad de Chaitin. □

## 12.4

*Demostrar que  $\sum_{\text{palabra } s} 2^{K(s)}$  es computablemente aproximable desde abajo y Martin Löf aleatorio.*

## Ejercicio 13

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

*Proof.*

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

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

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. □

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