

# Algoritmos, Azar y Autómatas

## Resolución Ejercicios 9 a 13 (Aleatoriedad)

Manuel Panichelli

December 4, 2021

### Resultados previos

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

- Si  $\text{OraculoHalt}(Q(m, n)) = \text{true}$  (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)
   $m' \leftarrow m + 1$ 
  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*

**Prop. 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  $\Omega[1 \dots i]^*$  es una subrutina que es el programa elegante (el más corto) que computa  $\Omega[1 \dots i]$  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

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

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.  $\square$

## 12.2

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

## 12.3

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

## 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 1** (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 1 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!*