Teoría de Números y Criptografía

F. J. Lobillo

2021/2022

Parte II

Criptografía y Curvas Elípticas

Índice general

Capítulo 1

Complejidad algorítmica

1.1 Introducción

Notación big-$\mathcal{O}$. Sean $f, g : \mathbb{N}^r \to \mathbb{R}$. Decimos que $f = \mathcal{O}(g)$ si existe una constante $c \in \mathbb{R}^+$ tal que $f(n_1, \ldots, n_r) \leq cg(n_1, \ldots, n_r)$ para cualquier $(n_1, \ldots, n_r) \in \mathbb{N}^r$.

Esta notación se emplea para analizar el tiempo esperado que va a tardar un algoritmo en realizar una tarea.

Hay una primera propiedad que es inmediata: capa cualquier $\lambda \in \mathbb{R}^+$ y $g : \mathbb{N}^r \to \mathbb{R}$, tenemos que

$$\mathcal{O}(g) = \mathcal{O}(\lambda g).$$

Número de dígitos. Todo entero $b^{k-1} \leq n < b^k$ tiene exactamente k dígitos en base b. Por tanto, el número de dígitos en base b de un entero positivo n viene dado por la fórmula

$$\lfloor \log_b n \rfloor + 1 = \left\lfloor \frac{\log n}{\log b} \right\rfloor + 1.$$

Este valor indica el tamaño de un entero.

Complejidad de la aritmética. Sean $n, m \in \mathbb{N}$. Para calcular $n + m$ empleamos la representación en alguna base de numeración de ambos números. El algoritmo escolar va calculando los dígitos de la suma uno a uno. Para cada dígito hay que emplear un máximo de dos sumas, los dígitos correspondientes más el posible acarreo de la suma anterior. El caso de la resta es análogo.

Proposición 1.1. *Si* $t(n, m)$ *es la función que mide el tiempo necesario para calcular* $n \pm m$*, entonces*

$$t = \mathcal{O}(\max(\log n, \log m)).$$

La complejidad del producto es también sencilla de calcular.

Proposición 1.2. *Si* $t(n, m)$ *es la función que mide el tiempo necesario para calcular* nm*, entonces*

$$t = \mathcal{O}((\log n)(\log m))).$$

Demostración. Para multiplicar n por un dígito en base b, hay que realizar $\lfloor \log_b n \rfloor + 1$ multiplicaciones de un dígito más $\lfloor \log_b n \rfloor$ sumas correspondientes al acarreo. Para multiplicar por un m cualquiera hay que realizar la tarea anterior $\lfloor \log_b m \rfloor + 1$ veces, más $\lfloor \log_b m \rfloor$ sumas de números con $\lfloor \log_b n \rfloor + \lfloor \log_b m \rfloor + 1$ dígitos. Por tanto

$$\begin{aligned} t = \mathcal{O}(2(\lfloor \log_b n \rfloor + 1)(\lfloor \log_b m \rfloor + 1) \\ + \lfloor \log_b n \rfloor + \lfloor \log_b m \rfloor + 1) = \mathcal{O}((\log n)(\log m)). \end{aligned}$$

□

Proposición 1.3. *Sean* $n \geq m \in \mathbb{N}$. *Si* $t(n, m)$ *es la función que mide el tiempo necesario para calcular cociente y resto de la división de* n *entre* m *escritos en binario, entonces*

$$t = \mathcal{O}\left((\log n)(\log m)\right).$$

Demostración. Durante $\lfloor \log_2 n \rfloor - \lfloor \log_2 m \rfloor$ veces debemos hacer un máximo de $\lfloor \log_2 m \rfloor$ comparaciones de bits y la resta de un número de $\lfloor \log_2 m \rfloor + 1$ dígitos y otro de $\lfloor \log_2 m \rfloor$. Por tanto

$$t = \mathcal{O}\left(\left(\lfloor \log_2 n \rfloor - \lfloor \log_2 m \rfloor\right) \lfloor \log_2 m \rfloor\right),$$

lo que implica $t = \mathcal{O}\left((\log n)(\log m)\right)$

Corolario 1.4. *El tiempo empleado en representar un número* n *en binario a una base* b *es* $\mathcal{O}\left((\log n)^2\right)$.

Demostración. Debemos realizar

$$\lfloor \log_b n \rfloor + 1 = \left\lfloor \frac{\log n}{\log b} \right\rfloor + 1$$

divisiones de números menores o iguales que n entre b. Por la Proposición 1.3, cada una de estas divisiones tiene, por la Proposición 1.3, un coste $\mathcal{O}\left((\log n)(\log b)\right)$, de donde se obtiene el resultado ya que $\log b$ es constante. □

Corolario 1.5. *El tiempo empleado en pasar a binario un entero* n *escrito en base* b *es* $\mathcal{O}\left((\log n)^2\right)$.

Demostración. Calcular cociente y resto obtenidos al dividir n en base b entre 2 requiere $\lfloor \log_b n \rfloor + 1$ multiplicaciones y restas. Esta tarea hay que repetirla con los cocientes, cuyo tamaño siempre es menor que el de n, y hay a lo sumo $\lfloor \log_b n \rfloor$ divisiones. □

Corolario 1.6. *El tiempo empleado sumar dos elementos de* $\mathbb{Z}_n$ *es* $\mathcal{O}(\log n)$. *El producto de dos elementos en* $\mathbb{Z}_n$ *puede hacerse en tiempo* $\mathcal{O}\left((\log n)^2\right)$.

Proposición 1.7. *Sean* $n \geq m \in \mathbb{N}$. *Si* $t(n, m)$ *es la función que mide el tiempo necesario para calcular* d, u, v *tales que* $d = (n, m)$ *y* $d = un + vm$ *mediante el algoritmo extendido de Euclides, entonces*

$$t = \mathcal{O}\left((\log n)^3\right).$$

Demostración. Recordemos que el algoritmo extendido de Euclides puede presentarse de la siguiente forma:

Input: $n \geq m$.
Output: d, u, v tales que $un + vm = d = (n, m)$.

1: $r_0 \leftarrow n, r_1 \leftarrow m$
2: $u_0 \leftarrow 1, u_1 \leftarrow 0$
3: $v_0 \leftarrow 0, v_1 \leftarrow 1$
4: **repeat**
5: $q \leftarrow r_0 \div r_1, r \leftarrow r_0 \bmod r_1$ {Cociente y resto de la división euclídea}
6: $u \leftarrow u_0 - qu_1, v \leftarrow v_0 - qv_1$
7: $r_0 \leftarrow r_1, r_1 \leftarrow r$
8: $u_0 \leftarrow u_1, u_1 \leftarrow u$
9: $v_0 \leftarrow v_1, v_1 \leftarrow v$

10: **until** $r = 0$
11: **return** r_0, u_0, v_0.

Cada iteración realiza una división euclídea que, por la Proposición 1.3, tiene un coste $\mathcal{O}\left((\log n)^2\right)$. Nos queda acotar el número de repeticiones que realiza el algoritmo. Observemos que $r < \frac{r_0}{2}$: si $r_1 \leq \frac{r_0}{2}$, obviamente $r < r_1 \leq \frac{r_0}{2}$; si $r_1 > \frac{r_0}{2}$, tenemos que $q = 1$ y $r = r_0 - r_1 < r_0 - \frac{r_0}{2} = \frac{r_0}{2}$. Por tanto, el número de repeticiones viene acotado por $2\log_2 n$. En consecuencia, el algoritmo extendido de Euclides es $\mathcal{O}\left((\log n)^3\right)$. □

Proposición 1.8. *El tiempo empleado en calcular* $b^m \bmod n$ *está en* $\mathcal{O}\left(\max\{(\log n)^2(\log m), (\log m)^2\}\right)$.

Demostración. Describamos el método de los cuadrados iterados para calcular $b^m \bmod n$. Para ello, si $m = m_t m_{t-1} \dots m_1 m_0)_2$ es su escritura binaria, que puede ser calculada en $\mathcal{O}((\log m)^2)$ por el Corolario 1.5, tenemos que

$$b^m = ((\cdots((b^{m_t})^2 b^{m_{t-1}})^2 b^{m_{t-2}} \cdots)^2 b^{m_1})^2 b^{m_0},$$

luego hay que realizar un máximo de $3\log_2 m$ multiplicaciones en $\mathbb{Z}_n$, cada una de ellas con un coste $\mathcal{O}((\log n)^2))$ por el Corolario 1.6. □

Ejercicios de Complejidad algorítmica

Ejercicio 1.1. Supongamos que $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lambda \in \mathbb{R}$. Demuestra que $f = \mathcal{O}(g)$. Como consecuencia demuestra que $\log n = \mathcal{O}(n^\epsilon)$ para cualquier $\epsilon > 0$.

Ejercicio 1.2. Sean $f_1 \in \mathcal{O}(g_1)$ y $f_2 \in \mathcal{O}(g_2)$. Calcula una cota para la complejidad de $f_1 + f_2$ y $f_1 f_2$.

Ejercicio 1.3. Estudia la complejidad de la suma y el producto de dos polinomios de grados n_1 y n_2 con coeficientes en $\mathbb{Z}_m$.

Ejercicio 1.4. Sabemos que

$$\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}.$$

Estudia la complejidad del cálculo de cada uno de los términos de la fórmula anterior.

Ejercicio 1.5. Supongamos que n es un primo escrito en binario. Comprueba que el algoritmo escolar calcula $\lfloor \sqrt{n} \rfloor$ en $\mathcal{O}\left((\log n)^2\right)$.

Ejercicio 1.6 (Algoritmo de Karatsuba)**.** Sea $x = x_0 + x_1\beta + \cdots + x_{n-1}\beta^{n-1}$ con n par. En base $\beta^{\frac{n}{2}}$ se representa:

$$x = x_{(0)} + x_{(1)}\beta^{\frac{n}{2}}$$

Sean a, b enteros de n dígitos en base β. El producto clásico se describe como

$$a \cdot b = a_{(0)} \cdot b_{(0)} + (a_{(1)} \cdot b_{(0)} + a_{(0)} \cdot b_{(1)}) \cdot \beta^{\frac{n}{2}} + a_{(1)} \cdot b_{(1)} \cdot \beta^{n}$$

La idea de Karatsuba:

$$\begin{aligned} a \cdot b = {} & a_{(0)} \cdot b_{(0)} \\ & + (a_{(0)} \cdot b_{(0)} + a_{(1)} \cdot b_{(1)} + (a_{(1)} - a_{(0)}) \cdot (b_{(0)} - b_{(1)})) \cdot \beta^{\frac{n}{2}} \\ & + a_{(1)} \cdot b_{(1)} \cdot \beta^{n} \end{aligned}$$

Útil para $n = 2^m$ dígitos. El algoritmo podemos escribirlo como sigue.

KARATSUBA(a, b, m, β) :
Input: a, b, m, β
Output: ab
if $m = 0$ **then**
return $a \cdot b$
else
$a1 \leftarrow 2^{m-1}$ dígitos más significativos de a
$a0 \leftarrow 2^{m-1}$ dígitos menos significativos de a
$b1 \leftarrow 2^{m-1}$ dígitos más significativos de b
$b0 \leftarrow 2^{m-1}$ dígitos menos significativos de b
$t1 \leftarrow$ KARATSUBA$(a1, b1, m-1, \beta)$
$t2 \leftarrow$ KARATSUBA$(a1 - a0, b0 - b1, m-1, \beta)$
$t3 \leftarrow$ KARATSUBA$(a0, b0, m-1, \beta)$
return $t1 \cdot \beta^{2^m} + (t1 + t2 + t3) \cdot \beta^{2^{m-1}} + t3$

Calcula la complejidad del mismo.

Bibliografía

[1] Gregory V. Bard. *Algebraic Cryptanalysis*. Springer Science and Business Media, 2009.

[2] Hans Delfs and Helmut Knebl. *Introduction to Cryptography*. Information Security and Cryptography. Springer-Verlag Berlin Heidelberg, 2015.

[3] Andreas Enge. *Elliptic curves and their applications to cryptography. An Introduction.* Kluwer Academic Publishers, 1999.

[4] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, fourth edition, 1960.

[5] Nathan Jacobson. *Basic Algebra: I*. W.H. Freeman & Company, second edition, 1985.

[6] Neal Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 2 edition, 1994.

[7] National Institute of Standards and Technology (NIST). *Digital Signature Standard (DSS)*, July 2013.

[8] Harald Niederreiter and Arne Winterhof. *Applied Number Theory*. Springer International Publishing, 2015.

[9] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer International Publishing, 2016.

[10] Joachim von zur Gathen. *CryptoSchool*. Springer-Verlag Berlin Heidelberg, 2015.