Universidad de La Laguna

SECCIÓN DE MATEMÁTICAS | FACULTAD DE CIENCIAS

Sandra Beatriz Jiménez Carballo

Test de primalidad: el algoritmo AKS

Primality tests: the AKS algorithm

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Septiembre de 2018

Dirigido por

Ignacio García Marco
María Victoria Reyes Sánchez

Ignacio García Marco
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

María Victoria Reyes Sánchez
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

A mis tutores,
Ignacio García Marco y María Victoria Reyes Sánchez,
por su gran ayuda durante la elaboración de este trabajo.
A todos mis profesores,
por haberme enseñado tanto y contagiarme por completo
el amor por las matemáticas.
A mi familia y amigos,
por apoyarme siempre.
En especial,
a mis padres, mi hermana y mi abuela,
porque sin ellos nada habría sido posible.
A Bianca y Yoselin,
por estar a mi lado incluso antes de emprender este camino juntas.
A Kirti,
por ser una fuente de inspiración en mi vida.

Resumen · Abstract

Resumen

El estudio comienza con un capítulo introductorio sobre números primos, algunas familias de números especiales, test de primalidad y resultados importantes. Esto constituye una base para el verdadero objetivo de nuestra memoria: estudiar en detalle el Algoritmo AKS. Este es el primer test de primalidad determinista y polinomial, y se construye a partir de una generalización para polinomios del Pequeño Teorema de Fermat. Tras una serie de resultados técnicos, se enuncia y demuestra el Teorema de Agrawal, Kayal y Saxena, del que se deduce directamente el citado algoritmo. Tras esto, demostramos que el problema de primalidad está en la clase de complejidad P (de problemas polinomiales), y por último se comentan mejoras proporcionadas por otros investigadores de esta área.

Palabras clave: *Números primos – Test de primalidad – Pequeño Teorema de Fermat – Algoritmo AKS.*

Abstract

This work begins with an introductory chapter about prime numbers, some families of special numbers, primality tests and some important results about these topics. This constitutes a basis for the main objective of our memory: to study in detail the AKS Algorithm. This is the first deterministic and polynomial-time primality test. This test is built as a generalization for polynomials of Fermat's Little Theorem. After a series of technical results, we state and prove the Theorem of Agrawal, Kayal and Saxena; the key result for obtaining the algorithm. As a consequence, we get that the problem of testing primality of an integer is in the complexity class P (of polynomial-time solvable problems). Finally, we comment on some improvements of the algorithm provided by other researchers.

Keywords: *Prime numbers – Primality test – Fermat's Little Theorem – AKS Algorithm.*

Contenido

Introducción

Un problema siempre presente a lo largo de la historia de las matemáticas es el de determinar si un número es o no primo. Para resolverlo se desarrollan los denominados tests de primalidad. Estos son o deterministas o eficientes, no ambas cosas. Surge entonces uno de los grandes dilemas de la teoría de números y teoría de la complejidad computacional: ¿el problema de primalidad está en P? No es hasta el año 2002 cuando los investigadores de la computación Agrawal, Kayal y Saxena proporcionan una respuesta a dicha cuestión publicando su artículo *"PRIMES is in P"* [1], en el que presentan un algoritmo determinista y eficiente, el denominado algoritmo AKS.

El primer capítulo comienza definiendo formalmente los números primos y continúa describiendo la primera prueba de primalidad conocida, la famosa Criba de Eratóstenes (datada en el siglo II a.C.) que, a pesar de funcionar bien, es ineficiente. Acto seguido, se estudia una familia particular de números: los primos de Mersenne. Tras ello, se enuncia y demuestra el Pequeño Teorema de Fermat, de especial importancia en nuestro estudio. Este Teorema aporta una condición necesaria para que un entero sea primo. No obstante, de él no se deriva directamente un test de primalidad determinista puesto que hay enteros (los llamados números de Carmichael) que no son primos y satisfacen la condición del Teorema. El capítulo finaliza con un test probabilista y eficiente, el Test de Miller-Rabin.

El segundo capítulo se centra en introducir el algoritmo AKS y probar que efectivamente resuelve el problema de primalidad de forma determinista. El punto de partida es una extensión para polinomios del Pequeño Teorema de Fermat. Se desarrolla una estrategia para convertir esta propiedad en un test adecuado. La idea se basa en hacer varias comprobaciones de la igualdad $(x+a)^n = x^n + a$ módulo n y $x^r - 1$, para un valor convenientemente elegido de r y varios valores

de a, y así poder garantizar la primalidad de un número determinado y a la vez hacerlo en un tiempo aceptablemente rápido. Tras varios resultados finalmente se enuncia y demuestra el Teorema de Agrawal, Kayal y Saxena, y se desarrolla el algoritmo.

En el tercer y último capítulo se analiza la complejidad del Test AKS. Aún sin ser nuestro objetivo realizar un estudio profundo al respecto, ni tratar de conseguir el mejor tiempo de ejecución, se demuestra que el algoritmo es polinomial y que por lo tanto, el problema de primalidad está en P. Para poder hacerlo, previamente se fijan algunas bases de la Teoría de la Complejidad. Nuestra memoria finaliza explicando algunas mejoras del AKS y comentando que, a pesar de tener por fin un algoritmo de primalidad determinista y polinomial, los test utilizados en los software comerciales son los probabilistas, dado que sus tiempos de ejecución son mejores y la probabilidad de error baja. En la mayoría de ocasiones es el test de Miller-Rabin, introducido en el primer capítulo, el que se encuentra implementado.

Como hemos comentado, la importancia teórica del Algoritmo AKS es esencialmente resolver un importante problema matemático que llevaba abierto más de 2000 años. En cuanto a la cuestión práctica, la mayor aplicación es la de generar claves criptográficas, puesto que para ello es necesario computar números primos exageradamente grandes.

Para llevar a cabo esta memoria, se han consultado varias fuentes bibliográficas. La que más se ha utilizado es [14]. La principal aportación de este trabajo es la de reescribir de forma directamente accesible a un alumno del último curso del Grado en Matemáticas la demostración de la corrección y eficiencia del algoritmo AKS. A diferencia del artículo original [1] y del libro [14], en este trabajo se explotan las estructuras algebraicas de grupo, anillo y cuerpo que surgen de forma natural en el estudio de este problema. Asimismo se incluye una introducción a la teoría de la complejidad amigable y adaptada a no expertos.

1

Números primos

En este capítulo vamos a hablar de los números primos, algunas familias de primos especiales y sus respectivas propiedades. Además veremos algunos algoritmos que sirven para determinar si un número es primo o no.

1.1. Definición y resultados básicos

La primera pregunta que surge es cómo detectar si un número es primo o no, lo que lleva a plantearnos: ¿hay un número finito de primos? Si esto fuera así, para cualquier entero que nos dieran sería "fácil" comprobar si es primo (al menos de forma teórica), simplemente buscando en esa lista hipotética y finita de primos. Desafortunadamente esto no se verifica. No hay límite para la cantidad de números primos (un hecho descubierto por Euclides), y por lo tanto no es tan sencillo averiguar si un entero es primo o no. Comencemos con una definición formal de qué es un número primo y algunos resultados acerca de su naturaleza y distribución.

Definición 1.1. *Sea $p \in \mathbb{Z}$, con $p \geq 2$. Diremos que p es un número primo si sus únicos divisores son ± 1 y $\pm p$.*

Los números primos son una especie de "átomos" de la aritmética, ya que cualquier entero mayor que 1 se escribe de forma única como producto de primos.

Teorema 1.2 (Teorema fundamental de la aritmética). *Sea $n \in \mathbb{Z}$, con $n > 1$. Entonces n puede escribirse como $n = p_1 \cdot p_2 \cdot \ldots \cdot p_r$, siendo r_i primo, $1 \leq i \leq r$. Además, dicha escritura es única, salvo el orden en el que aparecen los primos.*

Otro resultado importante acerca de los números primos es el siguiente.

Teorema 1.3 (Teorema de Euclides). *Existen infinitos números primos.*

La siguiente pregunta que surge naturalmente es cuál es la densidad de los números primos en el conjunto de los enteros. Para responder a esta cuestión, primero debemos introducir la siguiente notación.

Definición 1.4. *Sean $f, g : \mathbb{N} \longrightarrow \mathbb{R}$ aplicaciones. Se dice que f y g son equivalentes asintóticamente (o que f es asintóticamente equivalente a g, o viceversa) cuando $n \to \infty$ si:*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$$

Definición 1.5. *Para todo $n \in \mathbb{Z}^+$ se define $\pi(n)$ como el número de primos menores o iguales que n, esto es $\pi(n) = |\{p \text{ primo } / p \leq n\}| \in \mathbb{Z}^+$. π se denomina función recuento de primos.*

El siguiente teorema es un famoso resultado conjeturado por primera vez por Gauss a finales del siglo XVIII y no probado hasta finales del siglo XIX por los matemáticos Jacques Hadamard y Charles-Jean de la Vallée Poussin.

Teorema 1.6 (Teorema de los números primos). *Sea $n \in \mathbb{Z}^+$. Entonces $\pi(n)$ es asintóticamente equivalente a $n/\ln n$, es decir:*

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1$$

Ejemplo 1.7. La precisión de este teorema para tres casos particulares:

n	$\approx n/\ln n$	$\pi(n)$ = número de primos menores o iguales que n
10^3	144,76	168
10^6	72.382,41	78.498
10^9	48.254.942,43	50.847.478

Este resultado nos dice que hay muchos primos, pero, ¿cómo los encontramos? Aquí entran en juego los tests de primalidad.

1.2. Criba de Eratóstenes

Uno de los test de primalidad más antiguos es la Criba de Eratóstenes. Se le atribuye a este matemático griego (276-194 aC.) y nos permite, no sólo determinar si un número n es primo, sino que de hecho encuentra todos los primos menores o iguales que n. Funciona de la siguiente manera:

Se listan todos enteros entre 2 y n. El primer número (a saber 2) debe ser primo. Tachamos todos los múltiplos de 2 que son mayores que 2. El primer número entero después de 2 que no ha sido tachado (es decir, 3) debe ser primo. Tachamos todos los múltiplos de 3 que son mayores que 3. El primer número entero después de 3 que no ha sido tachado (en este caso 5, pues el 4 ya ha sido tachado al ser múltiplo de 2) debe ser primo. Tachamos todos los múltiplos de 5 que son mayores que 5. Y así sucesivamente. Cuando tenemos un primo nuevo, tachamos todos los múltiplos de ese nuevo primo que son más grandes que él, y luego pasamos al siguiente entero que no ha sido tachado y que nuevamente debe ser primo.

Nótese que no es necesario llegar hasta n. Una vez que hayamos encontrado un primo más grande que $\sqrt{n}$, todos los enteros restantes que no se han tachado, deben ser primos. Para probar esto, supongamos por reducción al absurdo que hemos encontrado un número m no tachado y compuesto. Entonces se tiene que:

$$m \leq n \text{ y } m = ab$$

Necesariamente $a \leq \sqrt{m}$ o $b \leq \sqrt{m}$, y por lo tanto, $a \leq \sqrt{n}$ o $b \leq \sqrt{n}$. Al ser m un múltiplo de a y de b, ya se habría tachado porque el algoritmo llegó hasta $\sqrt{n}$.

Algoritmo 1 Criba de Eratóstenes

Require: Un número natural $n \geq 2$.
Ensure: Todos los números primos menores o iguales que n.

```
for i = 2 to n do
    a_i ← i
end for
j ← 2
while j^2 ≤ n do
    if a_j ≠ 0 then
        for i = 2 to ⌊n/j⌋ do
            a_{i·j} ← 0
        end for
    end if
    j ← j + 1
end while
for i = 2 to n do
    if a_i ≠ 0 then
        escribir i
    end if
end for
```

A pesar de la efectividad de este algoritmo, tiene un problema importante, y es que si n es demasiado grande, el número de pasos es tan alto que no se puede llevar a cabo en la práctica.

En la siguiente sección estudiaremos una familia de primos especiales, que ha sido objeto de estudio a lo largo del tiempo, y que ha llamado nuestra atención dado que hay resultados recientes al respecto.

1.3. Números perfectos y primos de Mersenne

Definición 1.8. *Sea $n \in \mathbb{Z}^+$. Se dice que n es un número perfecto si es suma de sus divisores propios.*

Ejemplo 1.9. Los cuatro primeros números perfectos son:

$$\begin{aligned} 6 &= 1+2+3 \\ 28 &= 1+2+4+7+14 \\ 496 &= 1+2+4+8+16+31+62+124+248 \\ 8128 &= 1+2+4+8+16+32+64+127+254+508+1016+2032+4064 \end{aligned}$$

Todavía en la actualidad existen dos problemas famosos abiertos sobre los números perfectos: ¿hay infinitos? ¿Hay alguno impar? Se cree que la respuesta a la primera pregunta es "sí", y a la segunda "no". No obstante, existen algunos resultados parciales al respecto. Lo que sí se ha demostrado es que si existe un número perfecto impar, este debe ser mayor que 10^{1500} y tiene que tener al menos 10 factores primos distintos [13] (al menos 12 si no es divisible por 3 [12]). Uno de esos factores debe ser mayor que 10^8 [7], dos deben ser mayores que 10.000 [8] y tres deben ser mayores que 100 [9].

Otras preguntas que nos pueden surgir al ver la pequeña lista de números perfectos del ejemplo 1.9 son las siguientes: ¿Hay alguna forma simple de generarlos? ¿Todos tienen que terminar en 6 u 8? Si nos restringimos a los números perfectos pares, entonces sí que hay respuesta para estas dos preguntas. Responderemos a la segunda más adelante en esta sección, y para la primera es necesario definir previamente los números de Mersenne.

Definición 1.10. *Sea $m \in \mathbb{Z}^+$. Se dice que m es un número de Mersenne si es de la forma $m = 2^n - 1$ para algún entero positivo n; y se denota por $M(n)$. Si además se tiene que m es primo, entonces m es un primo de Mersenne.*

Ejemplo 1.11. Los cuatro primeros primos de Mersenne son 3, 7, 31 y 127.

Consideremos la factorización de los cuatro primeros números perfectos.

$$\begin{aligned} 6 &= 2 \cdot 3 \\ 28 &= 2^2 \cdot 7 = 4 \cdot 7 \\ 496 &= 2^4 \cdot 31 = 16 \cdot 31 \\ 8128 &= 2^6 \cdot 127 = 64 \cdot 127 \end{aligned}$$

Observamos que en nuestra lista, todos los números perfectos se expresan como una potencia de 2 por un primo de Mersenne. ¿Ocurrirá esto siempre? Debemos estudiar una serie de resultados para poder responder a esta pregunta.

Teorema 1.12. *Sea $n \in \mathbb{Z}^+$. Si $M(n)$ es un primo de Mersenne, entonces:*

$$m = 2^{n-1}M(n) = 2^{n-1}(2^n - 1)$$

es un número perfecto.

Demostración. Como por hipótesis $M(n)$ es primo, entonces los divisores propios de $m = 2^{n-1}M(n)$ son:

$$1, 2, 2^2, \ldots, 2^{n-1}, M(n), 2M(n), 2^2M(n), \ldots, 2^{n-2}M(n).$$

Y por lo tanto, la suma de los divisores propios de m es:

$$(1 + 2 + 2^2 + \ldots + 2^{n-1}) + M(n)\left(1 + 2 + 2^2 + \ldots + 2^{n-2}\right).$$

Como conocemos la suma de una progresión geométrica, y por la propia definición de $M(n)$, esto igual a:

$$(2^n - 1) + (2^{n-1} - 1)(2^n - 1) = 2^{n-1}(2^n - 1) = m.$$

Es decir que m es un número perfecto, como queríamos ver. □

Analizando de cerca este teorema, vemos que por cada primo de Mersenne, hay un número perfecto. El siguiente teorema establece que, de hecho, no existen otros números perfectos pares.

Teorema 1.13. *Sea m un número perfecto par. Entonces existe un entero n tal que $m = 2^{n-1}M(n) = 2^{n-1}(2^n - 1)$, donde $M(n)$ es un primo de Mersenne.*

Demostración. Por hipótesis m es un número par, luego, podemos expresarlo de la forma $m = 2^a t$, donde t es impar y $a \in \mathbb{Z}^+$. Denominamos S a la suma de todos los divisores de t, es decir, la suma de los divisores impares de m. Entonces, la suma de todos los divisores de m será de la forma:

$$S + 2S + 2^2S + \ldots + 2^aS$$

Nótese que esto incluye a m. Si queremos obtener la suma de los divisores propios, debemos restarlo. Además, como m es un número perfecto, sabemos que esta suma será exactamente él mismo. Así:

$$m = (1 + 2 + 2^2 + \ldots + 2^a)S - m = (2^{a+1} - 1)S - m$$

Despejando S y operando:

$$S = \frac{2m}{2^{a+1} - 1} = \frac{2^{a+1}t}{2^{a+1} - 1} = \frac{(2^{a+1} - 1)t + t}{2^{a+1} - 1} = t + \frac{t}{2^{a+1} - 1} = t + \frac{t}{M(a+1)}$$

Como s y t son números enteros, $t/M(a+1)$ debe ser también entero, y por lo tanto, un divisor de t. Habíamos definido S como la suma de todos los divisores de t, y ahora lo tenemos escrito como una suma de exactamente dos divisores de t. Esto significa que t tiene exactamente dos divisores, necesariamente él mismo y la unidad, es decir que t es primo. Además:

$$\frac{t}{M(a+1)} = 1 \Rightarrow t = M(a+1) \text{ primo } \Rightarrow m = 2^a t = 2^a M(a+1)$$

Llamando $n = a + 1 \in \mathbb{Z}^+$, se tiene que

$$m = 2^{n-1} M(n), \text{ siendo } M(n) \text{ primo.}$$

como queríamos ver. □

De los teoremas 1.12 y 1.13 se deduce que, si ignoramos la posibilidad de que existan números perfectos impares, entonces podemos caracterizar los números perfectos encontrando los números primos de Mersenne. Pero, ¿cuándo un número de Mersenne $M(n)$ es primo? Veremos un resultado rápido que al menos nos dice cuando no es primo, pero antes estudiemos el siguiente lema:

Lema 1.14. *Sean $a, b \in \mathbb{Z}$ tales que $a \mid b$. Entonces, $c^a - 1 \mid c^b - 1$, $\forall c \in \mathbb{Z}$.*

Demostración. Por hipótesis $a \mid b$, esto es, $b = aq$, con $q \in \mathbb{Z}$. Entonces:

$$c^b - 1 = c^{aq} - 1 = (c^a)^q - 1 = (c^a - 1) \cdot \left((c^a)^{q-1} + (c^a)^{q-2} + \ldots + c^a + 1\right)$$

Es decir que $c^a - 1 \mid c^b - 1$.

Teorema 1.15. *Sea $n \in \mathbb{Z}^+$. Si n es compuesto, entonces $M(n)$ es compuesto.*

Demostración. Por hipótesis suponemos que n es compuesto, luego, podemos expresarlo de la forma $n = ab$, donde a y b son enteros mayores que 1. Usando el Lema 1.14 tenemos que:

$$\begin{aligned} M(n) = M(ab) = 2^{ab} - 1 &= (2^a - 1)\frac{1 - 2^{ab}}{1 - 2^a} = (2^a - 1)\frac{1 - 2^{(b-1)a} \cdot 2^a}{1 - 2^a} \\ &= (2^a - 1)\left(1 + 2^a + 2^{2a} + \ldots + 2^{(b-1)a}\right) \end{aligned}$$

Como cada uno de los factores es mayor que 1, $M(n)$ es compuesto. □

Observación 1.16. Nótese que el Teorema 1.15 nos dice:

$$M(n) \text{ primo} \implies n \text{ primo}$$

y por lo tanto, en los Teoremas 1.12 y 1.13, necesariamente $n = p$ primo. Es decir:

Corolario 1.17. *Sea $n \in \mathbb{Z}$. Entonces:*

$$n \text{ perfecto par} \Leftrightarrow n = 2^{p-1}M(p) = 2^{p-1}(2^p - 1) \text{ para algún } p \text{ primo.}$$

Ya estamos listos para responder a la pregunta que habíamos formulado anteriormente: ¿todos los números perfectos acaban en 6 u 8?

Lema 1.18. *Sean $a, n \in \mathbb{Z}$. Si $n \equiv a$ (mód 5), entonces $n \equiv a$ (mód 10) o $n \equiv a + 5$ (mód 10).*

Demostración. Sean $a, n \in \mathbb{Z}$. Suponemos por hipótesis que $n \equiv a$ (mód 5), esto es $n - a = 5q$, con $q \in \mathbb{Z}$. Distinguimos dos posibles casos:

- Si q es par, será de la forma $q = 2h$, con $h \in \mathbb{Z}$. Luego, $n - a = 5 \cdot 2h = 10h$ con $h \in \mathbb{Z}$, es decir que $n \equiv a$ (mód 10).
- Si q es impar, será de la forma $q = 2h + 1$, con $h \in \mathbb{Z}$. Luego, $n - a = 5 \cdot (2h + 1) = 10h + 5$, con $h \in \mathbb{Z}$. Es decir que $n - (a + 5) = 10h$, con $h \in \mathbb{Z}$, y por lo tanto, $n \equiv a + 5$ (mód 10).

□

Teorema 1.19. *Si n es un número perfecto par, entonces su último dígito es* 6 *u* 8.

Demostración. Para probar que el último dígito de n es 6 u 8, lo que tenemos que ver es que 10 divide a $n - 6$ o a $n - 8$, es decir que $n \equiv 6$ (mód 10) o $n \equiv 8$ (mód 10). Antes de comenzar, nótese que:

$$2^4 \equiv 1 \text{ (mód 5)} \tag{1.1}$$

Como n es un número perfecto par, sabemos por el corolario 1.17 que es de la forma:

$$n = 2^{p-1}M(p) = 2^{p-1}(2^p - 1)$$

para algún número primo p. Si $p = 2$, el teorema se verifica pues $n = 6$. Suponemos ahora que $p > 2$. Entonces, p es un número impar y por lo tanto, $p \equiv 1$ (mód 4) o $p \equiv 3$ (mód 4). Luego, podemos expresar $p - 1$ de dos formas posibles: $p - 1 = 4m$ o $p - 1 = 4m + 2$ para algún entero m.

- Si $p-1=4m$: Entonces por (1.1) se tiene que:

$$2^{p-1}=2^{4m}=(2^4)^m\equiv 1^m=1 \text{ (mód 5) y}$$
$$2^p-1=2^{p-1}\cdot 2-1\equiv 1\cdot 2-1=1 \text{ (mód 5)}$$

  Luego, $n=2^{p-1}(2^p-1)\equiv 1\cdot 1=1$ (mód 5), y por el Lema 1.18, $n\equiv 1$ (mód 10) o $n\equiv 6$ (mód 10), pero como n es par, necesariamente $n\equiv 6$ (mód 10).
- Si $p-1=4m+2$: Entonces por (1.1) se tiene que:

$$2^{p-1}=2^{4m+2}=(2^4)^m\cdot 2^2\equiv 1^m\cdot 2^2=4 \text{ (mód 5) y}$$
$$2^p-1=2^{p-1}\cdot 2-1\equiv 4\cdot 2-1=7\equiv 2 \text{ (mód 5)}$$

  Luego, $n=2^{p-1}(2^p-1)\equiv 4\cdot 2=8\equiv 3$ (mód 5), y por el Lema 1.18, $n\equiv 3$ (mód 10) o $n\equiv 8$ (mód 10). De nuevo n es par, así que $n\equiv 8$ (mód 10).

□

Nuestro problema de encontrar números perfectos se ha reducido a saber cuando $M(p)$ es primo (siendo p primo). En un principio podríamos pensar que esto siempre es así:

$$M(2)=3 \text{ primo}$$
$$M(3)=7 \text{ primo}$$
$$M(5)=31 \text{ primo}$$
$$M(7)=127 \text{ primo}$$

Pero rápidamente comprobamos que no:

$$M(11)=2047=23\cdot 89$$
$$M(13)=8191 \text{ primo}$$
$$M(17)=131071 \text{ primo}$$
$$M(19)=524287 \text{ primo}$$
$$M(23)=8388607=47\cdot 178481$$

De hecho, los números primos p para los cuales $M(p)$ es primo, comienzan a desaparecer en este punto. Los siguientes 13 primos de Mersenne, tienen los siguientes valores para p:

$$31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423.$$

Estamos hablando de números realmente grandes, por ejemplo, $M(4423)$ tiene 1332 dígitos. ¿Cómo se sabe entonces si es primo? Para poder responder esta pregunta, entra en juego el Test de Lucas-Lehemer. Se trata de un algoritmo extremadamente rápido y simple que sirve para determinar si un número de Mersenne es primo o no. La teoría detrás de esto fue desarrollada por Edouard Lucas en 1878 y posteriormente mejorada por Derrick Henry Lehmer en la los años 30 del siglo pasado. Solo funciona con los números de Mersenne, y su

justificación requiere conocimientos más profundos en teoría de números que los presentados en este trabajo (ver [4]).

Algoritmo 2 Test de Lucas-Lehmer

Require: Un número impar $n \geq 3$.
Ensure: $M(n)$ si es primo.
1: $M \leftarrow 2^n - 1$
2: $S \leftarrow 4$
3: **for** $i = 2$ **to** $n - 1$ **do**
4: $\quad S \leftarrow S^2 - 2$ mód M
5: **end for**
6: **if** $S = 0$ **then**
7: $\quad$ escribir M
8: **end if**

A día de hoy, se conocen 50 primos de Mersenne, y el último encontrado es $2^{77232917}$. Tiene 23249425 cifras y fue descubierto hace tan sólo unos meses (el 26 de diciembre de 2017) por Jonathan Pace, a través del proyecto GIMPS (Great Internet Mersenne Prime Search) que utiliza principalmente el Test de Lucas-Lehmer 1.3.

1.4. Pequeño Teorema de Fermat

Ahora estudiaremos uno de los teoremas clásicos en teoría de números: el pequeño teorema de Fermat. Este resultado sirve para descartar, de una forma sencilla, aquellos enteros que no son primos, y nos será de utilidad en capítulos posteriores. Antes de enunciarlo, veremos unos resultados previos que nos ayudarán en su demostración.

Lema 1.20. *Sea $n \geq 2$ un número natural y sea p un número primo divisor de n. Entonces $\binom{n}{p}$ no es divisible por n.*

Demostración. Tenemos por hipótesis que $n \in \mathbb{Z}^+$, $n \geq 2$, p primo y $p \mid n$. Luego, podemos expresar n de la siguiente forma:

$$n = p^\alpha q, \text{ donde } \alpha, q \in \mathbb{Z}^+, \text{ con } p, q \text{ coprimos.}$$

Supongamos por reducción al absurdo que $\binom{n}{p}$ es divisible por n. Entonces, en particular, $\binom{n}{p}$ es divisible por p^α, es decir que $\binom{n}{p} = p^\alpha r$, donde $r \in \mathbb{Z}$.

Así, por definición de número combinatorio, $\frac{n!}{p!(n-p)!} = p^\alpha r$, y operando se tiene que $n \cdot (n-1) \cdot \ldots \cdot (n-(p-1)) = p^\alpha \cdot r \cdot p!$. Como $n = p^\alpha q$, simplificando nos queda $q \cdot (n-1) \cdot \ldots \cdot (n-(p-1)) = r \cdot p \cdot (p-1)!$. Entonces p divide a

$q\cdot(n-1)\cdot\ldots\cdot(n-(p-1))$. Por lo tanto p divide a $(n-1)\cdot\ldots\cdot(n-(p-1))$ ya que m.c.d.$(p,q)=1$. Como p es primo, esto implica que existe $b\in\{1,2,\ldots,p-1\}$ tal que $p \mid (n-b)$. Además por hipótesis $p \mid n$, luego $p \mid b$. Llegamos a una contradicción pues $b\in\{1,2,\ldots,p-1\}$. □

El Lema 1.20 nos sirve para probar la siguiente caracterización de los números primos, que a su vez utilizaremos en la demostración del Pequeño Teorema de Fermat.

Proposición 1.21. *Sea n un número natural, $n\geq 2$. Entonces:*

$$n \text{ divide a } \binom{n}{i}, \text{ para cualquier } i\in\{1,2,\ldots,n-1\} \Leftrightarrow n \text{ es primo.}$$

Demostración. Para la implicación directa suponemos por contrarrecíproco que n no es primo. Entonces existe p un número primo divisor de n. Luego, por el Lema 1.20, n no divide a $\binom{n}{p}$.
Para la otra implicación tomamos un $i\in\{1,2,\ldots,n-1\}$. Tenemos que:

$$\binom{n}{i}=\frac{n!}{i!(n-i)!}=m\in\mathbb{N}.$$

Operando, $n\cdot(n-1)!=n!=m\cdot i!\cdot(n-i)!$. Luego, n divide a $m\cdot i!\cdot(n-i)!$ y como tenemos que n es primo y además $i, n-i<n$, concluímos que $n \mid m$, es decir $n \mid \binom{n}{i}$, como queríamos ver. □

Tras esta caracterización de los números primos, podríamos construir un algoritmo de primalidad comprobando $n \mid \binom{n}{i}$ para los $i\in\{1,2,\ldots,n-1\}$. Si alguno no se verifica, entonces n es compuesto. Si por el contrario se verifica para todos, entonces n es primo. No obstante, igual que ocurría con la Criba de Eratóstenes, el número de pasos es demasiado grande para ser un test viable en la práctica.

Teorema 1.22 (Pequeño Teorema de Fermat).
Sean p un número primo y $a\in\mathbb{Z}$. Entonces $a^p\equiv a$ (mód p).

Demostración. En primer lugar observamos que es suficiente con probar el Teorema para los números a no negativos, ya que en otro caso basta con sustituir a por un número no negativo congruente con él (por ejemplo por el resto de dividirlo por p). Procedamos por inducción sobre a. Para $a=0$ tenemos:

$$a^p=0^p=0=a\equiv a \text{ (mód } p\text{) (Base de la inducción)}$$

Para el paso inductivo suponemos que se verifica para un $a\in\mathbb{N}_0$. Esto es:

$$a^p\equiv a \text{ (mód } p\text{) (Hipótesis de inducción)}$$

Y tenemos que demostrar que entonces también se verifica para $a+1$. Por el Binomio de Newton sabemos que: $(a+1)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i}$, y con la Proposición 1.21 podemos afirmar que los coeficientes binomiales $\binom{p}{1}, \binom{p}{2}, \ldots, \binom{p}{p-1}$ son todos divisibles por p. Así que si reducimos la ecuación anterior módulo p, sólo quedarán los términos para $i = 0$ y para $i = p$. Luego, con esto y con la hipótesis de inducción concluimos que:

$$(a+1)^p \equiv a^p + 1 \equiv a + 1 \ (\text{mód } p)$$

lo que completa la inducción y prueba el teorema. □

1.5. Pseudoprimos y números de Carmichael

Tras estudiar el Pequeño Teorema de Fermat, en un principio podríamos pensar que proporciona directamente un test de primalidad de la siguiente forma:
Sea n un entero, que queremos saber si es o no primo. Tomamos un $a \in \mathbb{Z}^+$ cualquiera y comprobamos si $a^n \equiv a$ (mód n). Si no se verifica, tenemos que n no es primo, por el Pequeño Teorema de Fermat (Teorema 1.22). Si se verifica, de momento no podemos afirmar nada, pero tomando una cantidad lo suficientemente grande de enteros $a \in \mathbb{Z}^+$ distintos y todos verificando la congruencia, sería lógico pensar que n es primo casi seguro, con una baja probabilidad de error. Pero hay un problema: el recíproco del teorema de Fermat no es cierto:

$$a^p \equiv a \ (\text{mód } p), \ \forall a \in \mathbb{Z} \ \not\Rightarrow \ p \text{ primo.}$$

El contraejemplo es la existencia de los denominados números de Carmichael. Antes de definirlos, vamos a estudiar los pseudoprimos [5].

Definición 1.23. *Sean $a, n \in \mathbb{Z}^+$. Se dice que n es un pseudoprimo en base a, si n es compuesto verificando la congruencia $a^n \equiv a$ (mód n). Para el caso particular $a = 2$, se dice directamente que n es un pseudoprimo.*

Ejemplo 1.24. Nótese que 91 es pseudoprimo en base 3 ya que $91 = 7 \cdot 13$ es compuesto y $3^{91} \equiv 3$ (mód 91). De forma similar, 341 es pseudoprimo (es decir, pseudoprimo en base 2) pues $341 = 11 \cdot 31$ es compuesto y $2^{341} \equiv 2$ (mód 341).

Los pseudoprimos son bastante escasos en el conjunto de los números enteros; hay muchos menos que números primos. Para hacernos una idea, sólo hay 3 pseudoprimos menores que mil (341, 561 y 645) y 245 menores que un millón. No obstante, son infinitos.

Teorema 1.25. *Para cada entero $a \geq 2$, existen infinitos pseudoprimos base a.*

Demostración. Sea $a \in \mathbb{Z}$, con $a \geq 2$. Por el Teorema de Euclides (Teorema 1.3) y el Teorema fundamental de la aritmética 1.2, es obvio que existirán infinitos primos que no dividen a $a^2 - 1$. Luego, si probamos que para cualquier primo impar p que no divide a a^2-1, se tiene que $n = (a^{2p}-1)/(a^2-1)$ es pseudoprimo base a, ya tendríamos que existen infinitos pseudoprimos base a.
Sea entonces p un primo impar que no divide a $a^2 - 1$. En primer lugar, nótese que $n \in \mathbb{Z}$ ya que $a^2 - 1 \mid a^{2p} - 1$ por el Lema 1.14. Además:

$$n = \frac{a^p - 1}{a - 1} \cdot \frac{a^p + 1}{a + 1}$$

y por lo tanto, n es compuesto. Por el Pequeño Teorema de Fermat (Teorema 1.22), $a^p \equiv a \pmod{p}$, luego, $a^{2p} \equiv a^2 \pmod{p}$. Esto es, p divide a $a^{2p} - a^2 = (n - 1)(a^2 - 1)$, y como por hipótesis p no divide a $a^2 - 1$, necesariamente p divide a $n - 1$.
Podemos también concluir un segundo hecho acerca de $n - 1$:

$$n - 1 = \frac{a^{2p} - a^2}{a^2 - 1} = \frac{a^2 - a^{2p-2} \cdot a^2}{1 - a^2} = a^2 + a^4 + \ldots + a^{2(p-2)} + a^{2(p-1)}$$

Tenemos expresado $n - 1$ como la suma de un número par de términos de la misma paridad, así que $n-1$ debe ser par. Entonces, tanto p como 2 son divisores de $n-1$, así que $2p$ también debe serlo. Luego, por el Lema 1.14, $a^{2p}-1$ divide a $a^{n-1}-1$, y como además n divide a $a^{2p}-1$, concluimos que n divide a $a^{n-1}-1$, es decir que $a^n \equiv a \pmod{n}$, como queríamos ver. □

Hablemos ahora de un tipo especial de pseudoprimos: los números de Carmichael.

Definición 1.26. *Si $n \in \mathbb{Z}^+$ es un pseudoprimo en base a, para todo $a \in \mathbb{Z}^+$, entonces n recibe el nombre de número de Carmichael.*

Una caracterización de estos números es la siguiente:

Teorema 1.27. *Un entero n es un número de Carmichael si y sólo si es positivo, compuesto, libre de cuadrados y tal que para cada primo p que divide a n, se tiene que $p - 1$ divide a $n - 1$.*

Demostración. Para la primera implicación, por la propia definición de número de Carmichael, n es compuesto, positivo y

$$a^n \equiv a \pmod{n}, \ \forall a \in \mathbb{Z}. \tag{1.2}$$

Veamos ahora que es libre de cuadrados, es decir que para todo p divisor de n, p^2 no divide a n. Sea entonces $p \mid n$ y supongamos por reducción al absurdo que $p^2 \mid n$. Así, podemos expresar n de la forma $n = p^2 m$, con $m \in \mathbb{Z}$. En particular,

por (1.2), $p^n \equiv p \pmod{n}$, es decir que $p^n = p + nq = p + p^2mq$, con $q \in \mathbb{Z}$. Como $n - 1 \geq 1$, operando se llega a que $p(p^{n-2} - mq) = 1$. Esto es, $pt = 1$, con $t \in \mathbb{Z}$. Hemos llegado a una contradicción ya que p es primo.

Faltaría probar que si p es un primo divisor de n, entonces $p - 1$ es un divisor de $n - 1$. Nótese que, al ser p primo, $(\mathbb{Z}_p^*, \cdot)$ es un grupo cíclico, y por lo tanto existe $a \in \mathbb{Z}$ tal que $o(\overline{a}) = |\mathbb{Z}_p^*| = p - 1$. Es decir que $a^{p-1} \equiv 1 \pmod{p}$ y

$$a^r \equiv 1 \pmod{p} \Leftrightarrow \ p - 1 \mid r \tag{1.3}$$

Este generador a del grupo $(\mathbb{Z}_p^*, \cdot)$ recibe el nombre de raíz primitiva módulo p. En particular, por (1.2), $a^n \equiv a \pmod{n}$. Y como además, p divide a n, se tiene que $a^n \equiv a \pmod{p}$. Al ser a es la raíz primitiva módulo p, necesariamente $a \not\equiv 0 \pmod{p}$, y por lo tanto $a^{n-1} \equiv 1 \pmod{p}$. Así, por (1.3), $p - 1 \mid n - 1$.
Para el recíproco tenemos varias hipótesis, entre ellas que n es libre de cuadrados, de lo que se deduce que:

$$n = p_1 \cdot p_2 \cdot \ldots \cdot p_r, \text{ con } p_i \text{ primo } \forall i \in \{1, 2, \ldots, r\} \text{ y } p_i \neq p_j, \ \forall i \neq j.$$

Por lo tanto, basta con probar que $a^n \equiv a \pmod{p_i}$, $\forall i \in \{1, 2, \ldots, r\}$, pues:

$$\begin{aligned} a^n \equiv a \pmod{p_i}, \ \forall i \in \{1, 2, \ldots, r\} &\Rightarrow p_i \mid a^n - a, \ \forall i \in \{1, 2, \ldots, r\} \\ &\Rightarrow n = p_1 \cdot p_2 \ldots p_r \mid a^n - a \\ &\Rightarrow a^n \equiv a \pmod{n} \end{aligned}$$

Sea entonces p un divisor de n y $a \in \mathbb{Z}$. Distinguimos dos posibles casos:

1. Si $p \mid a$, entonces $a^n \equiv 0 \equiv a \pmod{p}$.
2. Si $p \nmid a$, entonces por el Pequeño Teorema de Fermat 1.22, $a^{p-1} \equiv 1 \pmod{p}$, y como $p \mid n$, por hipótesis, $p - 1 \mid n - 1$. Por lo tanto, $a^{n-1} \equiv a^{(p-1)m} \equiv 1^m = 1 \pmod{p}$.

□

Ejemplo 1.28. Los números de Carmichael menores que 10^4 son:

$$\begin{aligned} 561 &= 3 \cdot 11 \cdot 17 \\ 1105 &= 5 \cdot 13 \cdot 17 \\ 1729 &= 7 \cdot 13 \cdot 19 \\ 2465 &= 5 \cdot 17 \cdot 29 \\ 2821 &= 7 \cdot 13 \cdot 31 \\ 6601 &= 7 \cdot 23 \cdot 41 \\ 8911 &= 7 \cdot 19 \cdot 67 \end{aligned}$$

Aunque por restricciones de espacio no podamos mostrar la prueba, se sabe que existen infinitos números de Carmichael, (ver [2]).

1.6. Test de Miller-Rabin

En esta última sección, vamos a hablar de un test de primalidad aleatorizado (es decir, que se basa en una elección aleatoria de ciertos parámetros). Se fundamenta en la siguiente propiedad de los números primos:

Lema 1.29. *Sean p un número primo y $a \in \mathbb{Z}$ tales que $a^2 \equiv 1$ (mód p). Entonces $a \equiv 1$ (mód p) o $a \equiv -1$ (mód p).*

Demostración. Por hipótesis tenemos que:

$$a^2 - 1 = (a-1)(a+1) \equiv 0 \pmod{p}$$

Como p es primo, necesariamente $a - 1 \equiv 0 \pmod{p}$ o $a + 1 \equiv 0 \pmod{p}$. En otras palabras: $a \equiv 1 \pmod{p}$ o $a \equiv -1 \pmod{p}$. □

Ahora el Teorema de Fermat-Miller, que es una mejora del Pequeño Teorema de Fermat.

Teorema 1.30 (Teorema de Fermat-Miller). *Sea p un número primo impar. Escribimos $p - 1 = d \cdot 2^l$, donde $d, l \in \mathbb{Z}^+$ y d es un número impar. Si $a \in \mathbb{Z}$ no es un múltiplo de p, entonces se verifica una, y sólo una, de las afirmaciones siguientes:*

(a) $a^d \equiv 1$ *(mód p)*
(b) $a^{2^i d} \equiv -1$ *(mód p) para algún* $i \in \{0, \ldots, l-1\}$

Demostración. Como por hipótesis, p es un número primo, por el Pequeño Teorema de Fermat (Teorema 1.22) tenemos que:

$$(a^d)^{2^l} = a^{d \cdot 2^l} = a^{p-1} \equiv 1 \pmod{p}$$

Distinguimos dos posibles casos:

- $a^d \equiv 1 \pmod{p}$. Entonces se tendría *(a)*.
- $a^d \not\equiv 1 \pmod{p}$. Sea $i \in \{0, \ldots, l-1\}$ el primer índice tal que $(a^d)^{2^{i+1}} = (a^d)^{2^i \cdot 2} = \left((a^d)^{2^i}\right)^2 \equiv 1 \pmod{p}$. Por el lema 1.29, necesariamente $(a^d)^{2^i} \equiv 1 \pmod{p}$ o $(a^d)^{2^i} \equiv -1 \pmod{p}$, pero observamos que el primer caso es imposible por cómo hemos definido i.

Ya estamos listos para formular el algoritmo.

Algoritmo 3 Test de Miller-Rabin

Require: Un número natural $n > 1$.

1: Si n es un número par y $n > 2$, o si n es una potencia de algún otro número natural, responder "n es compuesto".
2: En otro caso, escribir $n - 1 = d \cdot 2^l$ y elegir un número al azar $a \in \{1, 2, \ldots, n-1\}$.
3: Si m.c.d.$(a, n) \neq 1$, responder "n es compuesto".
4: De lo contrario, calcular $b := a^d$ (mód n).
5: Si $b = 1$, entonces responder "n es probablemente primo".
6: En otro caso, calcular $b, b^2, b^4, \ldots, b^{2^{l-1}}$ (mód n).
7: Si ninguno de esos números es congruente con -1 módulo n, entonces responder "n es compuesto".
8: De lo contrario, responder "n es probablemente primo".

Como comentábamos desde un principio, el Test de Miller-Rabin es aleatorizado y por lo tanto, existe una posibilidad de que nos de una respuesta errónea. No obstante, la probabilidad de error es tan insignificante, que este algoritmo es utilizado en la práctica sin problema. Sin embargo, desde un punto de vista teórico, nos preguntamos si esta aleatoriedad es realmente necesaria. ¿Existe un algoritmo que resuelva el problema de primalidad con total seguridad y de forma lo suficientemente rápida? Es aquí donde entra en juego el Algoritmo AKS, en el que se basa este trabajo y que estudiaremos en profundidad durante los próximos capítulos.

2

El algoritmo AKS

El objetivo de este capítulo es presentar y demostrar la corrección y eficiencia del ya mencionado algoritmo AKS. Este algoritmo se construye a partir de un nuevo criterio de primalidad, a saber, una generalización del Pequeño Teorema de Fermat que involucra polinomios en una variable.

2.1. Una generalización del Pequeño Teorema de Fermat

El punto de partida para empezar a desarrollar el algoritmo AKS será probar una extensión para polinomios del Pequeño Teorema de Fermat. Comenzamos la sección añadiendo algo de notación elemental.

Notación 2.1 *Sean* $P(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_2 x^2 + a_1 x + a_0 \in \mathbb{Z}[x]$ *y* $p \in \mathbb{Z}$. *Denotaremos por* $\overline{P}(x)$ *al polinomio de grado menor o igual que* d *cuyos coeficientes* $\overline{a}_i \in \mathbb{Z}_p$ *son las clases módulo* p *de los coeficientes* $a_i \in \mathbb{Z}$ *de* $P(x)$, *para todo* $i \in \{1, 2, \ldots, n\}$. *Así:*

$$\overline{P}(x) = \overline{a}_d x^d + \overline{a}_{d-1}x^{d-1} + \ldots + a_2 x^2 + \overline{a}_1 x + \overline{a}_0 \in \mathbb{Z}_p[x]$$

Ya tenemos lo necesario para enunciar y demostrar el teorema deseado.

Teorema 2.2 (Fermat para polinomios). *Sea* p *un número primo. Entonces, para todos los polinomios* $P(x)$ *con coeficientes enteros, se tiene que:*

$$\overline{P}(x)^p = \overline{P}(x^p) \text{ en } \mathbb{Z}_p[x].$$

Demostración. Sea d el grado de $P(x)$. Procedemos por inducción sobre d. Trivialmente se tiene para el polinomio cero. Si $d = 0$, $P(x)$ es constante no nulo. Esto es, $P(x) = a \in \mathbb{Z}$, con $a \neq 0$. Por lo tanto, lo que habría que probar es que $\overline{a}^p = \overline{a}$ en $\mathbb{Z}_p$, es decir que $a^p \equiv a \pmod{p}$, y se tiene directamente por

el Pequeño Teorema de Fermat. Esto establece la base de la inducción. Ahora suponemos que el teorema se cumple para todos los polinomios con coeficientes enteros de grado a lo sumo d (hipótesis de inducción) y vamos a probar que se verifica para un polinomio $P(x)$ de grado $d+1$. Sea $Q(x)$ el polinomio obtenido al quitar el término líder de $P(x)$. Es decir, $Q(x)$ es un polinomio de grado a lo sumo d y existe un $a \in \mathbb{Z}$, $a \neq 0$, tal que:

$$P(x) = ax^{d+1} + Q(x)$$

Aplicando el Binomio de Newton:

$$\begin{aligned} P(x)^p &= (ax^{d+1} + Q(x))^p \\ &= \sum_{i=0}^{p} \binom{p}{i} (ax^{d+1})^i \cdot Q(x)^{p-i} \\ &= Q(x)^p + \sum_{i=1}^{p-1} \binom{p}{i} (ax^{d+1})^i \cdot Q(x)^{p-i} + (ax^{d+1})^p. \end{aligned}$$

Observamos que:

- $\overline{Q}(x)^p = \overline{Q}(x^p)$ en $\mathbb{Z}_p[x]$ por la hipótesis de inducción.
- Los coeficientes binomiales $\binom{p}{i}$ son múltiplos de p para todo $i \in \{1, 2, \ldots, n-1\}$ por la Proposición 1.21.
- $(\overline{a}x^{d+1})^p = \overline{a}^p (x^p)^{d+1} = \overline{a}(x^p)^{d+1}$ en $\mathbb{Z}_p[x]$ usando el Pequeño Teorema de Fermat (Teorema 1.22).

Así que, en efecto:

$$\overline{P}(x)^p = \overline{Q}(x^p) + \overline{0} + \overline{a}(x^p)^{d+1} = \overline{a}(x^p)^{d+1} + \overline{Q}(x^p) = \overline{P}(x^p) \text{ en } \mathbb{Z}_p[x].$$

Esto completa la inducción y prueba el teorema. □

Corolario 2.3. *Sean $a, p \in \mathbb{Z}$. Si p es primo, entonces:*

$$(x + \overline{a})^p = x^p + \overline{a} \text{ en } \mathbb{Z}_p[x].$$

Demostración. Es un caso particular del Teorema de Fermat para polinomios (Teorema 2.2) donde $P(x) = x + a$. □

Veamos que en el Corolario 2.3, si a y p son coprimos, entonces se cumple el recíproco.

Teorema 2.4 (Criterio de Primalidad). *Sean $a, n \in \mathbb{Z}^+$ coprimos. Entonces n es primo si y sólo si se verifica*

$$(x + \overline{a})^n = x^n + \overline{a} \in \mathbb{Z}_n[x]. \tag{2.1}$$

Demostración. La implicación directa es el Corolario 2.3. Para la otra implicación suponemos por contrarecíproco que n es un número compuesto. Entonces existe p primo divisor de n. Sabemos que el coeficiente de x^p en $(x + a)^n$ es $\binom{n}{p}a^{n-p}$. Como por hipótesis a es coprimo con n y por el Lema 1.20, $\binom{n}{p}$ no es divisible por n, se sigue que el p-ésimo coeficiente de $(x+a)^n$ no es divisible por n. Por lo tanto:

$$(x + \overline{a})^n \neq x^n + \overline{a} \text{ en } \mathbb{Z}_n[x]$$

□

La hipótesis de que a y n deben ser coprimos, es indispensable para tener esta caracterización. De lo contrario, n no tiene por qué ser primo.

Ejemplo 2.5. Observamos que

$$\begin{aligned}(x + \overline{4})^4 &= x^4 + \overline{16}x^3 + \overline{96}x^2 + \overline{256}x + \overline{256} \\ &= x^4 \\ &= x^4 + \overline{4} \in \mathbb{Z}_4[x]\end{aligned}$$

pero 4 no es primo.

Nótese que en este teorema ya tenemos un "si y sólo si", y por lo tanto un criterio de primalidad del que se desprende un algoritmo para determinar si un entero es primo. Este algoritmo consistiría en elegir un entero a cualquiera que sea menor que n. Si a no es primo con n, entonces n es compuesto. Si a es primo con n, entonces n es primo si y solo si $(x + \overline{a})^n = x^n + \overline{a} \in \mathbb{Z}_n[x]$. No obstante, no se conoce cómo hacer esta última comprobación de forma eficiente y, por ende, el algoritmo propuesto no es eficiente.

2.2. Fermat para polinomios módulo $x^r - 1$

La idea del algoritmo AKS será cambiar la comprobación del Teorema 2.4, por varias comprobaciones (modificando el valor de a) módulo $x^r - 1$, para un valor de r convenientemente elegido. El hecho de calcular módulo $x^r - 1$ va a ser crucial para demostrar la eficiencia del algoritmo. Como contrapartida, habrá que efectuar la comprobación (2.1) módulo $x^r - 1$ para varios valores de a, para así poder garantizar la primalidad de un número.

Notación 2.6 *Sean $P(x), G(x) \in \mathbb{Z}[x]$ y $p \in \mathbb{Z}$. Denotaremos por $[\overline{P}(x)]_{\overline{G}}$ a la clase de equivalencia de $\overline{P}(x) \in \mathbb{Z}_p[x]$ en $\mathbb{Z}_p[x]/(\overline{G})$. Además, si $\deg(\overline{G}) = d$, entonces el anillo cociente es:*

$$\mathbb{Z}_p[x]/(\overline{G}) = \{[\overline{a}_0 + \overline{a}_1 x + \ldots + \overline{a}_{d-1}x^{d-1}]_{\overline{G}} \,/\, \overline{a}_i \in \mathbb{Z}_p\}$$

Corolario 2.7. *Sean p primo, $a \in \mathbb{Z}$ y $G(x) = x^r - 1$ con $r \in \mathbb{Z}^+$. Entonces:*

$$[(x + \overline{a})^p]_{\overline{G}} = [x^p + \overline{a}]_{\overline{G}} \text{ en } \mathbb{Z}_p[x]/(G).$$

Demostración. Trivialmente por el Corolario 2.3 □

Proposición 2.8. *Sean $a, b \in A$ con A anillo, y $m \in \mathbb{N}$. Entonces*

$$a - b \text{ divide a } a^m - b^m.$$

Demostración. Procedemos por inducción sobre $m \in \mathbb{N}$. Trivialmente se verifica para $m = 0$ y $m = 1$, luego, esta será la base de nuestra inducción. Supongamos que se verifica para m, así $a - b$ divide a $a^m - b^m$, o lo que es lo mismo:

$$a^m - b^m = (a - b)c, \text{ con } c \in A \tag{2.2}$$

Y tenemos que ver que se verifica para $m + 1$. Observamos que:

$$a^{m+1} - b^{m+1} = a(a^m - b^m) + ab^m - b^{m+1}$$

Así, aplicando (2.2) se tiene:

$$a^{m+1} - b^{m+1} = a(a - b)c + ab^m - b \cdot b^m$$

Y operando llegamos a que:

$$a^{m+1} - b^{m+1} = (a - b) \cdot (ac + b^m)$$

es decir, $a - b$ divide a $a^{m+1} - b^{m+1}$. □

Corolario 2.9. *Sean $r, m \in \mathbb{Z}^+$ y $G(x) = x^r - 1$. Entonces*

$$G(x) \text{ divide a } G(x^m).$$

Demostración. Trivial tomando $A = \mathbb{Z}[x]$, $a = x^r$ y $b = 1$ en la Proposición 2.8. □

Lema 2.10. *Sean $r, m, n \in \mathbb{Z}^+$, $n \geq 2$, $G(x) = x^r - 1$ y $P(x)$ un polinomio con coeficientes enteros tal que $\overline{P}(x) \in (\overline{G}) \subseteq \mathbb{Z}_n[x]$. Entonces $\overline{P}(x^m) \in (\overline{G}) \subseteq \mathbb{Z}_n[x]$.*

Demostración. Tenemos por hipótesis que $\overline{P}(x) \in (\overline{G}) \subseteq \mathbb{Z}_n[x]$, es decir que $\overline{G}(x)$ divide a $\overline{P}(x)$ en $\mathbb{Z}_n[x]$. Entonces, tomando $y = x^m$ podemos afirmar que:

$$\overline{G}(y) = \overline{G}(x^m) \text{ divide a } \overline{P}(y) = \overline{P}(x^m) \text{ en } \mathbb{Z}_n[x] \tag{2.3}$$

Además por el Corolario 2.9 se tiene que $G(x)$ divide a $G(x^m)$ en $\mathbb{Z}[x]$. En particular:

$$\overline{G}(x) \text{ divide a } \overline{G}(x^m) \text{ en } \mathbb{Z}_n[x] \tag{2.4}$$

Luego, de (2.3) y (2.4) deducimos que $\overline{G}(x)$ divide a $\overline{P}(x^m)$ en $\mathbb{Z}_n[x]$, es decir que $\overline{P}(x^m) \in (\overline{G}) \subseteq \mathbb{Z}_n[x]$, como queríamos ver. □

A partir de ahora, en el resto del capítulo n denotará un número compuesto, p un primo divisor de n, $G(x) = x^r - 1$ con $r \in \mathbb{Z}^+$, y para todo polinomio $P(x) \in \mathbb{Z}[x]$, $\overline{P}(x) \in \mathbb{Z}_p[x]$.

A cada par de primos p y r, le asociaremos un submonoide de $(\mathbb{N}, \cdot)$, y a cada tripleta p, r, n le asociaremos un submonoide $\mathcal{P}_{p,n,r}$ de $(\mathbb{Z}_p[x], \cdot)$,. Ambos monoides serán claves para obtener el algoritmo. Comenzamos recordando brevemente la definición de monoide.

Definición 2.11. *Sea G un conjunto y $\cdot$ una operación. Diremos que $(G, \cdot)$ (o simplemente G) es un monoide si se verifican las siguientes propiedades:*

(1) G no vacío ($G \neq \phi$).
(2) $\cdot$ es ley de composición interna. $\forall a, b \in G,\ a \cdot b \in G$.
(3) Asociativa. $\forall a, b, c \in G,\ (a \cdot b) \cdot c = a \cdot (b \cdot c)$.
(4) Existencia del elemento neutro. $\exists 1 \in G$ tal que $a \cdot 1 = 1 \cdot a = a,\ \forall a \in G$.

Definición 2.12. *Sean $(G, \cdot)$ un monoide y H un subconjunto no vacío de G. Si $(H, \cdot)$ es un monoide, se dice que $(H, \cdot)$ es un submonoide de $(G, \cdot)$ (o simplemente H submonoide de G).*

Observación 2.13. Nótese que un monoide es un semigrupo que también tiene elemento neutro. Si además se satisface que todo elemento tenga inverso, entonces es grupo.

Lema 2.14. *Sea $P(x) \in \mathbb{Z}[x]$. Entonces el conjunto*

$$B = \{m \in \mathbb{Z}^+ \ / \ [\overline{P}(x)^m]_{\overline{G}} = [\overline{P}(x^m)]_{\overline{G}}\}$$

es un submonoide de $(\mathbb{N}, \cdot)$.

Demostración. Trivialmente se verifican *(1), (3)* y *(4)* de la definición 2.11 (pues $1 \in B$ es el elemento neutro, y el producto de enteros siempre es asociativo). Sólo faltaría ver que $\cdot$ es ley de composición interna. Sean $m_1, m_2 \in B$. Entonces:

$$m_1 \in \mathbb{Z}^+ \text{ y } [\overline{P}(x)^{m_1}]_{\overline{G}} = [\overline{P}(x^{m_1})]_{\overline{G}} \tag{2.5}$$

$$m_2 \in \mathbb{Z}^+ \text{ y } [\overline{P}(x)^{m_2}]_{\overline{G}} = [\overline{P}(x^{m_2})]_{\overline{G}} \tag{2.6}$$

Obviamente $m_1 m_2 \in \mathbb{Z}^+$. Además, de (2.5) se deduce que

$$[\overline{P}(x)^{m_1 m_2}]_{\overline{G}} = [(\overline{P}(x)^{m_1})^{m_2}]_{\overline{G}} = [\overline{P}(x^{m_1})^{m_2}]_{\overline{G}}$$

y por (2.6) esto es igual a $[\overline{P}(x^{m_1 \cdot m_2})]_{\overline{G}}$. Así, $m_1 \cdot m_2 \in B$, lo que concluye que $(B, \cdot)$ es submonoide de $(\mathbb{N}, \cdot)$, como queríamos ver. □

2.3. El conjunto $\mathcal{P}_{p,n,r}$

Como ya avanzamos con anterioridad, el conjunto $\mathcal{P}_{p,n,r}$ será clave para la obtención del algoritmo. En esta sección introducimos el conjunto y demostramos algunas de sus propiedades.

Notación 2.15 *Definimos:*

$$\begin{aligned}\mathcal{P}_{p,n,r} :&= \{\overline{Q}(x) \in \mathbb{Z}_p[x] \,/\, \overline{Q}(x)^n - \overline{Q}(x^n) \in (\overline{G}) \subseteq \mathbb{Z}_p[x]\} \\ &= \{\overline{Q}(x) \in \mathbb{Z}_p[x] \,/\, [\overline{Q}(x)^n]_{\overline{G}} = [\overline{Q}(x^n)]_{\overline{G}} \in \mathbb{Z}_p[x]/(\overline{G})\}\end{aligned}$$

Corolario 2.16. *Si $\overline{Q}(x) \in \mathcal{P}_{p,n,r}$, entonces $\overline{Q}(x)^m - \overline{Q}(x^m) \in (\overline{G})$ para cualquier $m = n^i \cdot p^j$ con $i, j \in \mathbb{Z}^+$.*

Demostración. Si vemos que se verifica para $m = n$ y $m = p$, ya lo tenemos para cualquier $m = n^i \cdot p^j$ con $i, j \in \mathbb{Z}^+$ por el Lema 2.14.

- Si $m = n$ es trivial por la propia definición del conjunto $\mathcal{P}_{p,n,r}$.
- Si $m = p$ se tiene por el Teorema de Fermat para polinomios (Teorema 2.2).

□

Lema 2.17. *$(\mathcal{P}_{p,n,r}, \cdot)$ es submonoide de $(\mathbb{Z}_p[x], \cdot)$, siendo $\cdot$ el producto de polinomios.*

Demostración. Trivialmente se verifican *(1), (3)* y *(4)* de la definición 2.11 (pues $\overline{1} \in \mathcal{P}_{p,n,r}$ es el elemento neutro, y el producto de polinomios siempre es asociativo). Sólo faltaría ver que $\cdot$ es ley de composición interna. Sean $Q_1(x), Q_2(x) \in \mathcal{P}_{p,n,r}$. Entonces:

$$\overline{Q}_1(x) \in \mathcal{P}_{p,n,r} \Rightarrow \overline{Q}_1(x) \in \mathbb{Z}_p[x] \text{ y } [\overline{Q}_1(x)^n]_{\overline{G}} = [\overline{Q}_1(x^n)]_{\overline{G}} \tag{2.7}$$

$$\overline{Q}_2(x) \in \mathcal{P}_{p,n,r} \Rightarrow \overline{Q}_2(x) \in \mathbb{Z}_p[x] \text{ y } [\overline{Q}_2(x)^n]_{\overline{G}} = [\overline{Q}_2(x^n)]_{\overline{G}} \tag{2.8}$$

Obviamente, $\overline{Q}_1(x) \cdot \overline{Q}_2(x) \in \mathbb{Z}_p[x]$. Además, de (2.7) se deduce que

$$[(\overline{Q}_1(x) \cdot \overline{Q}_2(x))^n]_{\overline{G}} = [\overline{Q}_1(x)^n]_{\overline{G}} \cdot [\overline{Q}_2(x)^n]_{\overline{G}} = [\overline{Q}_1(x^n)]_{\overline{G}} \cdot [\overline{Q}_2(x^n)]_{\overline{G}}$$

y por (2.8) esto es igual a $[\overline{Q}_1(x^n) \cdot \overline{Q}_2(x^n)]_{\overline{G}}$. Así, $\overline{Q}_1(x) \cdot \overline{Q}_2(x) \in \mathcal{P}_{p,n,r}$, lo que concluye que $(\mathcal{P}_{p,n,r}, \cdot)$ es submonoide de $(\mathbb{Z}_p[x], \cdot)$, como queríamos ver. □

Lema 2.18. *Sean $P_1(x), P_2(x), Q(x)$ polinomios con coeficientes enteros y sea $H(x) \in \mathbb{Z}[x]$ un divisor de $Q(x)$. Entonces:*

$$\overline{P}_1(x) - \overline{P}_2(x) \in (\overline{Q}) \subseteq \mathbb{Z}_p[x] \Rightarrow \overline{P}_1(x) - \overline{P}_2(x) \in (\overline{H}) \subseteq \mathbb{Z}_p[x]$$

Demostración. Trivial pues al ser $H(x)$ divisor de $Q(x)$, en particular $\overline{H}(x)$ es divisor de $\overline{Q}(x)$, y por lo tanto $(\overline{Q}) \subseteq (\overline{H})$. □

Tomemos $\overline{H}(x)$ un factor irreducible de $\overline{G}(x) = x^r - \overline{1}$. Al ser p primo, $\mathbb{Z}_p$ es cuerpo, luego $\mathbb{Z}_p[x]$ es dominio de ideales principales. Entonces, como $\overline{H}(x) \in \mathbb{Z}_p[x]$ es irreducible, $(\overline{H})$ es maximal y por lo tanto $\mathbb{Z}_p[x]/(\overline{H})$ es un cuerpo y $\mathbb{Z}_p \hookrightarrow \mathbb{Z}_p[x]/(\overline{H})$ es una extensión de cuerpos. Consideramos ahora el siguiente conjunto:

$$\mathcal{M} = \{x^{n^i p^j} \ / \ (i,j) \in \mathbb{N}^2\} \subset \mathbb{Z}_p[x] \tag{2.9}$$

Y la proyección canónica:

$$\pi : \ \mathbb{Z}_p[x] \longrightarrow \mathbb{Z}_p[x]/(\overline{H})$$

$$\pi(\overline{P}(x)) = [\overline{P}(x)]_{\overline{H}}$$

Nótese que $\mathbb{Z}_p[x]/(\overline{H})$ es un conjunto finito.
Denominamos t al cardinal de la imagen del conjunto $\mathcal{M}$ por la proyección canónica:

$$\begin{aligned} t :&= |\pi(\mathcal{M})| \\ &= |\{[x^{n^i p^j}]_{\overline{H}} \ / \ (i,j) \in \mathbb{N}^2\}| < \infty. \end{aligned} \tag{2.10}$$

Lema 2.19. *Sean* $P_1(x), P_2(x) \in \mathbb{Z}[x]$ *con* $\deg(P_1(x)), \deg(P_2(x)) < t$. *Si se tiene que* $\overline{P}_1(x), \overline{P}_2(x) \in \mathcal{P}_{p,n,r}$, *entonces:*

$$\overline{P}_1(x) = \overline{P}_2(x) \Leftrightarrow [\overline{P}_1(x)]_{\overline{H}} = [\overline{P}_2(x)]_{\overline{H}}.$$

Demostración. La implicación directa es trivial. Para el recíproco definimos:

$$\overline{T}(y) := \overline{P}_1(y) - \overline{P}_2(y) \in \mathbb{Z}_p[y].$$

Obviamente $\deg(\overline{T}) < t$ pues $\deg(P_1), \deg(P_2) < t$. Observamos que:

- $\overline{P}_1(x) \in \mathcal{P}_{p,n,r}$, luego, por el Corolario 2.16: $\overline{P}_1(x)^{n^i p^j} - \overline{P}_1(x^{n^i p^j}) \in (\overline{G})$. Por el Lema 2.18, esto implica que $\overline{P}_1(x)^{n^i p^j} - \overline{P}_1(x^{n^i p^j}) \in (\overline{H})$, es decir:

$$[\overline{P}_1(x)^{n^i p^j}]_{\overline{H}} = [\overline{P}_1(x^{n^i p^j})]_{\overline{H}} \tag{2.11}$$

- $\overline{P}_2(x) \in \mathcal{P}_{p,n,r}$, luego, por el Corolario 2.16: $\overline{P}_2(x)^{n^i p^j} - \overline{P}_2(x^{n^i p^j}) \in (\overline{G})$. Por el Lema 2.18, esto implica que $\overline{P}_2(x)^{n^i p^j} - \overline{P}_2(x^{n^i p^j}) \in (\overline{H})$, es decir:

$$[\overline{P}_2(x)^{n^i p^j}]_{\overline{H}} = [\overline{P}_2(x^{n^i p^j})]_{\overline{H}} \tag{2.12}$$

Ahora, de (2.11) y (2.12) deducimos:

$$\begin{aligned} [\overline{T}(x^{n^i p^j})]_{\overline{H}} &= [\overline{P}_1(x^{n^i p^j})]_{\overline{H}} - [\overline{P}_2(x^{n^i p^j})]_{\overline{H}} \\ &= [\overline{P}_1(x)^{n^i p^j}]_{\overline{H}} - [\overline{P}_2(x)^{n^i p^j}]_{\overline{H}} \end{aligned}$$

Entonces, por la Proposición 2.8:

$$[\overline{T}(x^{n^i p^j})]_{\overline{H}} = ([\overline{P}_1(x)]_{\overline{H}} - [\overline{P}_2(x)]_{\overline{H}}) \cdot [\overline{Q}(x)]_{\overline{H}} \text{ en } \mathbb{Z}_p[x]/(\overline{H}) \text{ con } Q(x) \in \mathbb{Z}[x].$$

Por último, aplicando la hipótesis $[\overline{P}_1(x)]_{\overline{H}} = [\overline{P}_2(x)]_{\overline{H}}$, se llega a que $[\overline{T}(x^{n^i p^j})]_{\overline{H}} = [\overline{0}]_{\overline{H}}$ en $\mathbb{Z}_p[x]/(\overline{H})$, es decir que $[x^{n^i p^j}]_{\overline{H}}$ es raíz de $\overline{T}(y)$, y sabemos que hay t distintas (por (2.9) y (2.10)). Luego, hemos encontrado t raíces de $\overline{T}(y)$ en una extensión de $\mathbb{Z}_p$, y como $\deg(\overline{T}) < t$ concluímos que necesariamente $\overline{T}$ es el polinomio cero en $\mathbb{Z}_p[y]$, es decir que $\overline{P}_1 = \overline{P}_2$ en $\mathbb{Z}_p[x]$. □

Consideramos el siguiente conjunto:

$$\begin{aligned} \mathcal{A} :&= \pi(\mathcal{P}_{p,n,r}) \\ &= \{[\overline{Q}(x)]_{\overline{H}} \in \mathbb{Z}_p[x]/(\overline{H}) \ / \ \overline{Q}(x) \in \mathcal{P}_{p,n,r}\} \end{aligned} \tag{2.13}$$

Y denotamos por $A := |\mathcal{A}|$ su cardinal.

Observación 2.20. Nótese que las clases del polinomio cero y del polinomio uno están en $\mathcal{A}$, luego $\mathcal{A} \neq \phi$ y $A \geq 2$.

Tomamos ahora el conjunto siguiente:

$$\mathcal{L} := \{[x + \overline{a}]_{\overline{H}} \in \mathbb{Z}_p[x]/(\overline{H}) \ / \ x + \overline{a} \in \mathcal{P}_{p,n,r}\} \subset \mathcal{A} \tag{2.14}$$

con cardinal $l := |\mathcal{L}| \leq A$.

El resto del capítulo está dedicado a demostrar una serie de cotas y resultados técnicos. Estos tienen como consecuencia principal el Corolario 2.25, en el que se concluye que si las cantidades t y l ya definidas satisfacen ciertas condiciones, entonces n tiene que ser una potencia de p.

Teorema 2.21 (Cota inferior de A en términos de t y l).
El número A satisface la desigualdad $A \geq \binom{t+l-1}{t-1}$.

Demostración. Trivialmente, de (2.13) y del Lema 2.19 se tiene que:

$$\begin{aligned} A = |\mathcal{A}| &\geq |\{[\overline{Q}(x)]_{\overline{H}} \in \mathcal{A} \ / \ \deg(\overline{Q}(x)) < t\}| \\ &= |\{\overline{Q}(x) \in \mathcal{P}_{p,n,r} \ / \ \deg(\overline{Q}(x)) < t\}| \end{aligned}$$

Y a su vez, si aplicamos (2.14) y el Lema 2.17, como $\mathbb{Z}_p[x]$ es un dominio de factorización única, concluímos que:

$$\begin{aligned} A &\geq |\{(x + \overline{a}_1) \cdot \ldots \cdot (x + \overline{a}_s) \in \mathcal{P}_{p,n,r} \ / \ s < t \text{ y } [x + \overline{a}_i]_{\overline{H}} \in \mathcal{L}, \ \forall i \in \{1, \ldots, s\}\}| \\ &= \text{Número de monomios en las variables } y_1, \ldots, y_l \text{ de grado menor que } t \\ &= \binom{t+l-1}{l-1}. \end{aligned}$$

□

Lema 2.22. *Si n no es potencia de p, entonces la siguiente aplicación es inyectiva:*

$$\begin{aligned}\varphi\colon\ \mathbb{N}^2 &\longrightarrow \mathbb{N}\\ (i,j) &\longmapsto \varphi(i,j) = n^i p^j\end{aligned}$$

Demostración. Como p es un divisor de n pero n no es potencia de p, podemos expresarlo de la siguiente manera:

$$n = p^\alpha q, \text{ siendo } \alpha, q \in \mathbb{Z}^+ \text{ y m.c.d.}(p,q) = 1. \tag{2.15}$$

Sean $(i,j),(i',j') \in \mathbb{N}^2$ tales que $\varphi(i,j) = \varphi(i',j')$. Entonces, por cómo está definido φ, se tiene que $n^i p^j = n^{i'} p^{j'}$. Por 2.15 esto es $(p^\alpha q)^i p^j = (p^\alpha q)^{i'} p^{j'}$ y operando llegamos a:

$$p^{(\alpha i + j) - (\alpha i' + j')} = q^{i' - i} \tag{2.16}$$

De esta forma, como $\mathbb{Z}$ es un dominio de factorización única, se tiene que $(\alpha i + j) - (\alpha i' + j') = 0$, entonces $1 = p^0 = q^{i-i'}$. De aquí se deduce que $i = i'$, luego $j = j'$ y por lo tanto $(i,j) = (i',j')$. Así, concluímos que φ es inyectiva. □

Teorema 2.23 (Cota superior de A en términos de t). *Si n no es potencia de p, entonces se satisface la desigualdad $A \leq \frac{n^{2\sqrt{t}}}{2}$.*

Demostración. Echemos un vistazo a los números de la forma $m = n^i p^j$ con $i,j \in \mathbb{N}$. Sabemos por el Lema 2.22 que para diferentes elecciones de (i,j) resultarán diferentes números m. Si exigimos también que $0 \leq i,j \leq \lfloor\sqrt{t}\rfloor$, entonces habrá exactamente $(\lfloor\sqrt{t}\rfloor + 1)^2 > t$ elecciones para (i,j), es decir $(\lfloor\sqrt{t}\rfloor + 1)^2 > t$ elecciones para $m = n^i p^j$. Así, por cómo hemos definido t en (2.10) podemos afirmar que existen $m_1 = n^i p^j$ y $m_2 = n^{i'} p^{j'}$ $(m_1 \neq m_2)$ con $i,i',j,j' \in \{0,\cdots,\lfloor\sqrt{t}\rfloor\}$ e $(i,j) \neq (i',j')$ tales que:

$$[x^{m_1}]_{\overline{H}} = [x^{m_2}]_{\overline{H}} \text{ en } \mathbb{Z}_p[x]/(\overline{H}) \tag{2.17}$$

Suponemos sin pérdida de generalidad que $m_1 > m_2$, luego:

$$m_2 < m_1 = n^i p^j \leq n^{\lfloor\sqrt{t}\rfloor} p^{\lfloor\sqrt{t}\rfloor} = (np)^{\lfloor\sqrt{t}\rfloor} \tag{2.18}$$

Al ser p un divisor de n pero n no potencia de p podemos escribir $n = p^\alpha q$ con $\alpha, q \in \mathbb{Z}^+$ y $q \neq 1$, es decir que $q \geq 2$. De esta forma:

$$n = p^\alpha q \geq p^\alpha \cdot 2 \Rightarrow p^\alpha \leq \tfrac{n}{2} \Rightarrow p \leq \tfrac{n}{2}$$

Así, de (2.18) se sigue:

$$m_1 \leq (np)^{\lfloor\sqrt{t}\rfloor} \leq (n \cdot \frac{n}{2})^{\lfloor\sqrt{t}\rfloor} = \frac{n^{2\lfloor\sqrt{t}\rfloor}}{2^{\lfloor\sqrt{t}\rfloor}} \leq \frac{n^{2\sqrt{t}}}{2} \tag{2.19}$$

Por otro lado, para todos los polinomios $\overline{P}(x) \in \mathcal{P}_{p,n,r}$ se deduce de la ecuación (2.17) y del Corolario 2.16 que en $\mathbb{Z}_p[x]/(\overline{H})$:

$$[\overline{P}(x)^{m_1}]_{\overline{H}} = [\overline{P}(x^{m_1})]_{\overline{H}} = [\overline{P}(x^{m_2})]_{\overline{H}} = [\overline{P}(x)^{m_2}]_{\overline{H}} \tag{2.20}$$

Por lo tanto, cada elemento $[\overline{P}(x)]_{\overline{H}} \in \mathcal{A}$ es una raíz del polinomio $\overline{R}(y) = y^{m_1} - y^{m_2} \in \mathbb{Z}_p[y]$, es decir que $\overline{R}(y)$ tiene al menos $|\mathcal{A}| = A$ raíces. Además vemos que $\overline{R}(y)$ es un polinomio distinto de cero y de grado m_1, ya que $m_1 > m_2$. De esta forma, por (2.19) concluímos que:

$$A = |\mathcal{A}| \leq \deg(\overline{R}) = m_1 \leq \frac{n^{2\sqrt{t}}}{2}.$$

□

Lema 2.24. *Para todo $n \in \mathbb{N}$ se verifica $\binom{2n}{n} \geq 2^n$.*

Demostración. Procedemos por inducción sobre $n \in \mathbb{N}$. Observamos que para $n = 1$ se verifica pues $\binom{2\cdot 1}{1} = 2 \geq 2^1 = 2$, así que esto establece la base de la inducción. Supongamos que se verifica para n, esto es $\binom{2n}{n} \geq 2^n$. Y veamos que entonces también se da para $n+1$:

$$\begin{aligned}\binom{2(n+1)}{n+1} &= \binom{2n+2}{n+1} = \frac{(2n+2)!}{(n+1)!((2n+2)-(n+1))!} = \frac{(2n+2)!}{(n+1)!(n+1)!} \\ &= \frac{(2n+2)(2n+1)(2n)!}{(n+1)^2(n!)^2} = \frac{(2n+2)(2n+1)}{(n+1)^2} \cdot \frac{(2n)!}{(n!)^2} \\ &= \frac{(2n+2)(2n+1)}{(n+1)^2} \cdot \binom{2n}{n}\end{aligned}$$

Ahora, aplicando la hipótesis de inducción tenemos que:

$$\begin{aligned}\binom{2(n+1)}{n+1} &\geq \frac{(2n+2)(2n+1)}{(n+1)^2} \cdot 2^n = \frac{2(n+1)(2n+1)}{(n+1)^2} \cdot 2^n \\ &= \frac{2n+1}{n+1} \cdot 2^{n+1} \geq 2^{n+1}\end{aligned}$$

□

Corolario 2.25. *Si $t > 4(\log_2 n)^2$ y $l \geq t-1$, entonces n es potencia de p.*

Demostración. Suponemos por reducción al absurdo que n no es potencia de p. Sabemos por el Teorema 2.21 que $A \geq \binom{t+l-1}{t-1}$. Además, como $m \geq n \geq r$, entonces $\binom{m}{r} \geq \binom{n}{r}$. Luego, aplicando la hipótesis $l \geq t-1$:

$$A \geq \binom{t+l-1}{t-1} \geq \binom{2(t+1)}{t-1}$$

Y por el Lema 2.24:

$$A \geq \binom{2(t-1)}{t-1} \geq 2^{t-1} = \frac{2^t}{2} \tag{2.21}$$

También tenemos por hipótesis que $t > 4(\log_2 n)^2$, y observamos que:

$$t > 4(\log_2 n)^2 \Rightarrow \sqrt{t} > 2\log_2 n \Rightarrow t = \sqrt{t}\sqrt{t} > 2\sqrt{t}\log_2 n \tag{2.22}$$

Por lo tanto, de (2.21), (2.22) y del teorema 2.23 se deduce que:

$$A \geq \frac{2^t}{2} > \frac{2^{2\sqrt{t}\log_2 n}}{2} = \frac{(2^{\log_2 n})^{2\sqrt{t}}}{2} = \frac{n^{2\sqrt{t}}}{2} \geq A$$

Hemos llegado a que $A > A$, y el absurdo viene de suponer que n no es potencia de p. □

2.4. Polinomios ciclotómicos

Comencemos la sección con un breve recordatorio sobre los polinomios ciclotómicos.

Definición 2.26. *Sea $r \in \mathbb{Z}^+$. Se denomina r-ésimo polinomio ciclotómico al polinomio mónico cuyas raíces son todas las raíces primitivas de orden r de la unidad.*

Observación 2.27. Los polinomios ciclotómicos se calculan recursivamente de la siguiente forma:

$$\Phi_r(x) = \frac{x^r - 1}{\prod_{\substack{d|r \\ d<r}} \Phi_d(x)}$$

En particular, si r es primo, entonces

$$\Phi_r(x) = \frac{x^r - 1}{x - 1} = x^{r-1} + x^{r-2} + \ldots + x + 1$$

$\Phi_r(x)$ es irreducible en $\mathbb{Z}[x]$, pero en general, $\overline{\Phi}_r(x)$ no es irreducible en $\mathbb{Z}_p[x]$.

Recordemos que queremos obtener factores irreducibles del polinomio $\overline{G}(x) = x^r - \overline{1} \in \mathbb{Z}_p[x]$ para poder trabajar en un cuerpo $\mathbb{Z}_p[x]/(\overline{H})$. Escribimos entonces $G(x) = x^r - 1 \in \mathbb{Z}[x]$ como:

$$G(x) = x^r - 1 = (x-1)(x^{r-1} + x^{r-2} + \ldots + x + 1)$$

Si r es primo, entonces $G(x) = (x-1)\Phi_r(x)$.

Lema 2.28. *Sean $r, p \in \mathbb{Z}^+$, con p primo, y sea $\overline{H}(x)$ un factor irreducible de $\overline{G}(x) = x^r - \overline{1}$. Si k es el menor entero positivo tal que $x^k - \overline{1} \in (\overline{H})$, entonces k divide a r.*

Demostración. Como $\overline{H}(x)$ es un polinomio irreducible, $\mathbb{Z}_p[x]/(\overline{H})$ es cuerpo, luego, $(\mathbb{Z}_p[x]/(\overline{H}))\backslash\{[\overline{0}]_{\overline{H}}\}$ con el producto es un grupo multiplicativo.
Por cómo está definido k, se tiene que

$$\text{el orden de } [x]_{\overline{H}} \text{ en este grupo es } k \tag{2.23}$$

Además, como $\overline{H}(x)$ divide a $\overline{G}(x) = x^r - \overline{1}$, en particular $\overline{G}(x) = x^r - \overline{1} \in (\overline{H})$ y por lo tanto

$$([x]_{\overline{H}})^r = [x^r]_{\overline{H}} = [1]_{\overline{H}} \tag{2.24}$$

De (2.23) y (2.24) se deduce que, necesariamente, k divide a r. □

Lema 2.29. *Sean p, r dos números primos distintos y $\overline{H}(x)$ un factor irreducible del r-ésimo polinomio ciclotómico $\overline{\Phi}_r(x)$ de $\mathbb{Z}_p[x]$. Entonces se tiene que el orden de $[x]_{\overline{H}}$ en $\left(\mathbb{Z}_p[x]/(\overline{H})\right)^*$ es r.*

Demostración. Para ver esto, tenemos que probar dos cosas:

(a) $x^r - \overline{1} \in (\overline{H}) \subseteq \mathbb{Z}_p[x]$.
Trivial pues como $\overline{H}(x)$ es un factor irreducible de $\overline{\Phi}_r(x)$, en particular es un divisor de $\overline{G}(x) = x^r - \overline{1}$, luego, $x^r - \overline{1} \in (\overline{G}) \subseteq (\overline{H})$.

(b) $x^k - \overline{1} \notin (\overline{H}) \subseteq \mathbb{Z}_p[x]$, para cualquier $i \in \{1, 2, \ldots, r-1\}$.
Tomamos $k \geq 1$ el número más pequeño que verifica que $x^k - \overline{1} \in (\overline{H})$. Luego, por el lema anterior k divide a r, y como r es primo necesariamente $k = 1$ o $k = r$.
Si $k = 1$, entonces $x - \overline{1} \in (\overline{H})$, es decir que $(x - \overline{1}) = \overline{H}(x)\overline{P}(x)$, con $\overline{P}(x) \in \mathbb{Z}_p[x]$, y por lo tanto $\deg(\overline{H}(x)) = 1$. Así, $\overline{H}(x)$ será de la forma $\overline{H}(x) = \overline{a}(x - \overline{1})$, con $\overline{a} \in \mathbb{Z}_p$, $\overline{a} \neq 0$. Luego, $\overline{H}(\overline{1}) = \overline{0}$, y como $\overline{H}(x)$ es un divisor de $\overline{\Phi}_r(x) = x^{r-1} + x^{r-2} + \ldots + x + \overline{1}$ se tiene que:

$$\overline{0} = \overline{\Phi}_r(\overline{1}) = \overline{1} + \overline{1} + \ldots + \overline{1} = \overline{r} \neq \overline{0}$$

pues p y r son primos distintos. Hemos llegado a un absurdo. Necesariamente $k = r$.

□

Notación 2.30 *Sean $r, n \in \mathbb{Z}$ tales que m.c.d.(r,n)=1. Denotamos por $ord_r(n)$ al orden de $\overline{n}$ en $(\mathbb{Z}_r)^*$. Es decir:*

$$ord_r(n) = mín\{k \in \mathbb{Z}^+ \ / \ \overline{n}^k = \overline{1} \in (\mathbb{Z}_r)^*\}$$

Corolario 2.31. *Sean p, r dos números primos distintos y $\overline{H}(x)$ un factor irreducible de $\overline{\Phi}_r(x)$. Sea también $n \in \mathbb{Z}^+$ múltiplo de p y coprimo con r. Entonces, $ord_r(n) \leq t \leq r$.*

Demostración. Recordamos que $t = |\pi(\mathcal{M})| = |\{[x^{n^i p^j}]_{\overline{H}} \,/\, (i,j) \in \mathbb{N}^2\}|$.
Dividimos la demostración en dos partes:
(1) $ord_r(n) \leq t$. Para probar esta desigualdad, veremos que la aplicación:

$$\begin{array}{rcl} \psi\colon \{1,2,\ldots,ord_r(n)\} & \longrightarrow & \{[x^{n^i p^j}]_{\overline{H}} \,/\, (i,j) \in \mathbb{N}^2\} \\ i & \longrightarrow & \psi(i) = [x^{n^i}]_{\overline{H}} \end{array}$$

es inyectiva, es decir que si $1 \leq i < j \leq ord_r(n)$, entonces $[x^{n^i}]_{\overline{H}} \neq [x^{n^j}]_{\overline{H}}$ en $\mathbb{Z}_p[x]/(\overline{H})$. Luego, sean $i, j \in \mathbb{Z}^+$ tal que $1 \leq i < j \leq ord_r(n)$, y suponemos por reducción al absurdo que $[x^{n^i}]_{\overline{H}} = [x^{n^j}]_{\overline{H}}$ en $\mathbb{Z}_p[x]/(\overline{H})$. Esto es

$$x^{n^i} - x^{n^j} \in (\overline{H}) \subseteq \mathbb{Z}_p[x].$$

Observamos que $x^{n^i} - x^{n^j} = x^{n^i}(\overline{1} - x^{n^j - n^i})$. Así, $\overline{H}(x)$ divide a $x^{n^i}(\overline{1} - x^{n^j - n^i})$, y como $\overline{H}(x)$ es irreducible, necesariamente divide a x^{n^i} o a $\overline{1} - x^{n^j - n^i}$.

- Si $\overline{H}(x) \mid x^{n^i}$, entonces $\overline{H}(x) = x$, de nuevo por ser $\overline{H}(x)$ irreducible. Esto es una contradicción pues por hipótesis $\overline{H}(x)$ divide a $\overline{\Phi}_r(x)$ pero sabemos que x no divide a $\overline{\Phi}_r(x)$ ya que el $\overline{0}$ no es raíz de $\overline{\Phi}_r(x)$.
- Si $\overline{H}(x) \mid \overline{1} - x^{n^j - n^i}$, entonces $x^{n^j - n^i} - \overline{1} \in (\overline{H}) \subseteq \mathbb{Z}_p[x]$, y por el lema 2.29, necesariamente, $r = o(x) \mid n^j - n^i$, luego, $n^j \equiv n^i \pmod{r}$. Hemos llegado a una contradicción pues n es coprimo con r.

(2) $t \leq r$. Trivialmente, se tiene la siguiente desigualdad:

$$\begin{aligned} t &= |\{[x^{n^i p^j}]_{\overline{H}} \,/\, (i,j) \in \mathbb{N}^2\}| \\ &\leq |\{[x^{n^i}]_{\overline{H}} \,/\, i \in \mathbb{N}\}| \end{aligned}$$

Y por el Lema 2.29, esto a su vez es igual a:

$$|\{[\overline{1}]_{\overline{H}}, [x]_{\overline{H}}, \ldots, [x^{r-1}]_{\overline{H}}\}| = r$$

□

2.5. Teorema de Agrawal, Kayal y Saxena

Por fin estamos listos para demostrar el siguiente teorema, que nos dice cómo encontrar un polinomio que identifique n como compuesto, siempre y cuando n no sea potencia de un primo.

Teorema 2.32 (Teorema de Agrawal, Kayal y Saxena). *Sean n un número compuesto, p un primo divisor de n y r un primo que no divide a n y tal que $ord_r(n) > 4(\log_2 n)^2$. Sea también $Q(x) = x^r - 1$. Si n no es potencia de p,*

entonces hay a lo sumo $r-1$ *polinomios de la forma* $P(x)=x+a$, *con* $a \in \{0,1,\ldots,p-1\}$, *que satisfacen:*

$$[\overline{P}(x)^n]_{\overline{H}} = [\overline{P}(x^n)]_{\overline{H}} \text{ en } \mathbb{Z}_p[x]/(\overline{H})$$

Demostración. Recordamos que $l = |\{[x+\overline{a}]_{\overline{H}} \in \mathbb{Z}_p[x]/(\overline{H}) \ / \ x+\overline{a} \in \mathcal{P}_{p,n,r}\}|$. Entonces, lo que realmente tenemos que demostrar es que si n no es potencia de p, entonces $l \leq r-1$. Suponemos por contrarecíproco que:

$$l \geq r \tag{2.25}$$

Por hipótesis tenemos:

$$4(\log_2 n)^2 < ord_r(n) \tag{2.26}$$

Además, como r es coprimo con n, es evidente que $r \neq p$. Luego, por el Corolario 2.31:

$$ord_r(n) \leq t \leq r \tag{2.27}$$

De (2.25) y (2.27) se deduce que:

$$l \geq t-1 \tag{2.28}$$

Y de (2.26) y (2.27) se sigue:

$$t > 4(\log_2 n)^2 \tag{2.29}$$

Por último, con (2.28) y (2.29) se satisfacen las hipótesis para poder aplicar el Corolario 2.25, y por lo tanto afirmamos que n es potencia de p. □

2.6. El algoritmo de Agrawal, Kayal y Saxena

El objetivo de esta sección es el de enunciar y demostrar la corrección del algoritmo.

Definición 2.33. *Sea* $n \in \mathbb{N}$. *Decimos que* n *es una potencia perfecta si existen* $a, b \in \mathbb{Z}^+$ *con* $b \geq 2$ *tales que* $n = a^b$.

Algoritmo 4 Algoritmo AKS

Require: Un número natural $n \geq 2$

1: Si n es una potencia perfecta, entonces responder "n es compuesto".
2: En otro caso, hacer lo siguiente para $r = 2, 3, 4, \cdots$:

   (a) Comprobar si r es primo (por ejemplo usando la Criba de Eratóstenes).
   (b) Si $r \mid n$ y $r < n$, responder "n es compuesto".
   (c) Si $r = n$, responder "n es primo".
   (d) Calcular $ord_r(n)$.
   (e) Si r es primo y $ord_r(n) > 4(\log_2 n)^2$, fijar $Q(x) := x^r - 1$ y continuar con el Paso 3.

3: Comprobar las igualdades:

$$[(x + \overline{a})^n]_{\overline{Q}} = [x^n + \overline{a}]_{\overline{Q}} \in \mathbb{Z}_n[x]/(\overline{Q})$$

   para todos los elementos $\overline{a} \in \mathbb{Z}_n$ de $\overline{1}$ a $\overline{r-1}$.
4: Si alguna de esas igualdades no se verifica, responder "n es compuesto", de lo contrario responder "n es primo".

Observación 2.34. Nótese que el Paso 2 del algoritmo, si no se detiene en (b) o (c), lo que hace realmente es buscar el menor número primo que satisface $ord_r(n) > 4(\log_2 n)^2$ y fija $Q(x) := x^r - 1$.

Teorema 2.35. *El algoritmo AKS resuelve el problema de primalidad de forma determinista.*

Demostración. Veamos que el algoritmo devuelve "n es primo"si y sólo si n es primo. Es fácil ver que el algoritmo siempre termina. Para la implicación directa observamos que hay dos casos en los que el algoritmo responde que "n es primo":

- Si nos paramos en el Paso 2(c). Tenemos $r = n$, entonces hemos probado todos los números $r < n$ en el Paso 2 sin encontrar ningún divisor primo de n (pues de lo contrario, el algoritmo se habría parado en el Paso 2(b) devolviendo "n es compuesto"). Por lo tanto n es primo.
- Si nos detenemos en el Paso 4. Esto es debido a que todas las igualdades del Paso 3 se satisfacen, y por el Teorema 2.32 (Teorema de Agrawal, Kayal y Saxena), entonces n es primo o potencia de un primo. Pero observamos que en este último caso, el algoritmo se habría detenido en el Paso 1, luego, necesariamente n es primo.

Para la otra implicación suponemos por contrarecíproco que el algoritmo devuelve "n es compuesto"(y tenemos que ver que entonces n es compuesto). Luego, sucedió alguno de los siguientes casos:

- En el Paso 1 se detectó que n es una potencia perfecta. Efectivamente n es compuesto.

- En el Paso 2(b), el algoritmo encontró r un divisor primo de n tal que $r < n$, luego, n es compuesto.
- Se detuvo en el Paso 4. Por lo tanto, al menos una de las igualdades del Paso 3 no se satisface. Entonces, por el Corolario 2.7 (que es consecuencia directa del Teorema 2.2 - Fermat para polinomios), necesariamente n es compuesto.

3

El problema de primalidad está en P

En el capítulo anterior enunciamos el algoritmo AKS y demostramos que resolvía el problema de primalidad. Ahora vamos a probar que lo hace de forma "rápida", y para ello debemos introducirnos en la teoría de la complejidad.

No es un objetivo de este trabajo el de adentrarse formalmente dicha teoría, sino introducir las ideas esenciales que nos llevarán a la demostración de que el algoritmo AKS pertenece a la clase de complejidad P. Para un estudio en profundidad de la teoría de complejidad por medio del concepto de máquina de Turing, ver [10].

3.1. La clase de complejidad P

Comencemos la sección con unas definiciones básicas.

Definición 3.1. *Un algoritmo determinista es aquel que es completamente predictivo si se conocen sus entradas. Dicho de otra forma, si se conocen las entradas del algoritmo siempre producirá la misma salida, y la máquina interna pasará por la misma secuencia de estados.*

El concepto de algoritmo determinista se opone al de probabilista.

Definición 3.2. *Un algoritmo probabilista es aquel que emplea un grado de aleatoriedad como parte de su lógica, es decir, que basa su resultado en la toma de algunas decisiones al azar, de forma que en promedio, obtiene una solución correcta al problema planteado.*

Al contrario que en un algoritmo determinista, en uno probabilista se pueden obtener distintas soluciones a partir de los mismos datos, e incluso en algunos casos, las soluciones pueden ser erróneas (idealmente, la probabilidad de que esto

ocurra será muy baja). Repitiendo la ejecución de un algoritmo probabilista un número suficiente de veces para el mismo dato, se puede aumentar tanto como se quiera el grado de confianza.

Definición 3.3. *Un problema de decisión es aquel donde las respuestas posibles son «sí» o «no» («verdadero» o «falso»).*

Ejemplo 3.4. El problema de primalidad, es decir, la determinación de si un entero dado es un número primo, es un problema de decisión.

En ocasiones es difícil encontrar un algoritmo que resuelva un problema matemático dado, pero la teoría de la complejidad va más allá y considera, estudia y determina la eficiencia de los distintos algoritmos que resuelven el problema.

Ejemplo 3.5. La Criba de Eratóstenes es un algoritmo determinista de primalidad, pero desde el punto de vista de la complejidad no es nada eficiente pues a valores muy grandes, el número de pasos es inviable.

3.1.1. Complejidad algorítmica

Normalmente, un mismo problema puede tener distintos algoritmos que lo resuelven, y por lo tanto necesitamos, de alguna forma, poder compararlos entre sí para saber cuál es el más adecuado. De manera intuitiva, la complejidad algorítmica es una métrica teórica que se aplica a los algoritmos.
La comparación entre algoritmos se puede afrontar de dos maneras principales: complejidad temporal (tiempo que tarda el algoritmo en resolver el problema) y complejidad espacial (cantidad de memoria que necesita el algoritmo). En nuestro estudio nos restringiremos a la complejidad temporal, luego, cuando hablemos de complejidad a secas, nos estaremos refiriendo a la temporal.
Concluimos entonces que la complejidad algorítmica representa la cantidad de recursos (temporales) que necesita un algoritmo para resolver un problema y por tanto permite determinar la eficiencia de dicho algoritmo. La complejidad de un problema se define como la complejidad del mejor algoritmo que lo resuelve, y su estudio se conoce como teoría de la complejidad. Para profundizar en estas definiciones necesitamos introducir los conceptos de tiempo de ejecución de un algoritmo y crecimiento asintótico.

3.1.2. Tiempo de ejecución de un algoritmo

Definición 3.6. *Si A es un algoritmo que resuelve un problema determinado e I es una entrada para A, entonces el tiempo de ejecución del algoritmo A en la entrada I se define como el número de operaciones elementales que se llevan a*

cabo cuando A se aplica a I. Y de forma más general, el tiempo de ejecución del algoritmo A se define como la función:

$$\begin{aligned} t_A\colon \mathbb{N} &\longrightarrow \mathbb{R} \\ s &\longmapsto t_A(s) = \textit{mayor tiempo de ejecución de } A \textit{ en una entrada de longitud } s. \\ &= \textit{mayor número de operaciones elementales que se llevan a cabo} \\ &\qquad \textit{cuando } A \textit{ se aplica a una entrada de longitud } s. \end{aligned}$$

Observación 3.7. Nótese que en nuestra memoria, la definición de «tiempo de ejecución de un algoritmo A», de todos los tiempos de ejecución para una entrada de longitud s, está cogiendo el mayor de ellos, es decir, está considerando el peor de los casos. Pero también existen otras definiciones que consideran por ejemplo el tiempo medio, el mejor caso, etc.

Definición 3.8. *Sean A y B dos algoritmos que resuelven un mismo problema. Decimos que A es más eficiente que B si su tiempo de ejecución es menor cuando s es lo suficientemente grande; es decir si*

$$\exists S \in \mathbb{N} \textit{ tal que } t_A(s) \leq t_B(s), \textit{ para todo } s \geq S$$

En nuestro caso, las entradas al algoritmo van a ser números enteros, ¿pero cuántos caracteres tiene cada uno de ellos? Cualquier entero que introduzcamos en el ordenador se codifica en el sistema binario, por lo tanto, si la entrada de nuestro algoritmo es un número natural n, entonces su longitud será $\lfloor \log_2 n \rfloor + 1$. Con el único fin de simplificar los cálculos, de ahora en adelante redondearemos esta cantidad a $\log_2 n$.

3.1.3. Crecimiento asintótico

Nuestro objetivo es comparar la eficiencia y complejidad de distintos algoritmos utilizando su tiempo de ejecución, pero donde nuestras conclusiones sean independientes de la máquina que los ejecute. Por lo tanto, una medida de eficiencia significativa más allá del estado actual de la tecnología informática, debería tener las siguientes propiedades:

- No debe distinguir entre funciones de tiempo de ejecución que difieren a lo sumo por una constante multiplicativa.
- Debe limitarse a conclusiones sobre el comportamiento del algoritmo para entradas "grandes".

Existe un concepto matemático que satisface precisamente estos requisitos:

Definición 3.9. *Sean $f, g : \mathbb{N} \longrightarrow \mathbb{R}$ aplicaciones. Decimos que f crece asintóticamente a lo sumo tan rápido como g para $s \to \infty$, y denotamos $f(s) = O(g(s))$, o simplemente $f = O(g)$, si existe una constante real $C > 0$ tal que:*

$$|f(s)| \leq C|g(s)| \tag{3.1}$$

para todo s suficientemente grande. (Es decir, que existe un $S \in \mathbb{N}$ tal que se satisface la inecuación (3.1) para todo natural $s \geq S$).

Observación 3.10. Realmente $O(g)$ es una clase de funciones:

$$O(g) = \{f : \mathbb{N} \to \mathbb{R} \ / \ \exists C, S > 0, \text{ tales que } |f(s)| \leq C \cdot |g(s)|, \ \forall s \in \mathbb{N}, \ s \geq S\}$$

Luego, en realidad nuestra notación $f = O(g)$ no indica igualdad, sino pertenencia. Si fuéramos estrictos, lo correcto sería escribir $f \in O(g)$, sin embargo la otra notación está bien establecida y no debería causar confusión.

Definición 3.11. *Decimos que un algoritmo A tiene un tiempo de ejecución asintótico a lo sumo g si su tiempo de ejecución satisface $t_A(s) = O(g(s))$*

Antes de terminar la subsección, veamos unos sencillos resultados relacionados con el crecimiento asintótico, que nos serán de utilidad más adelante.

Lema 3.12. *Sean $n \in \mathbb{N}$ y $\varepsilon > 0$ un número real. Entonces $\log_2 n = O(n^\varepsilon)$.*

Demostración. Sea $\varepsilon > 0$. Observamos que:

$$\log_2 n = \log_2 \left((n^\varepsilon)^{1/\varepsilon} \right) = \frac{1}{\varepsilon} \cdot \log_2(n^\varepsilon) < \frac{1}{\varepsilon} \cdot n^\varepsilon$$

Luego, $\log_2 n = O(n^\varepsilon)$, como queríamos ver. □

Corolario 3.13. *Sea $n \in \mathbb{N}$. Entonces, $(\log_2 n)^2 = O(n)$.*

Demostración. Aplicando el lema 3.12 para $\varepsilon = \frac{1}{2}$, se tiene que $\log_2 n = O(n^{1/2})$, luego, $(\log_2 n)^2 = O((n^{1/2})^2) = O(n)$. □

3.1.4. Clases de Complejidad

Ya estamos listos para profundizar en la definición de complejidad algorítmica, y con ella poder hablar de las clases de complejidad.

Definición 3.14. *Se dice que un algoritmo A es polinomial si el número de operaciones está acotado superiormente por un polinomio en el tamaño de la entrada, es decir, si su función tiempo de ejecución es un polinomio:*

$$t_A(s) = O(s^k) \text{ para algún número positivo } k.$$

Observación 3.15. La idea es que los avances en la tecnología pueden tener un impacto real en la ejecución de algoritmos polinomiales, mientras que esto es mucho más difícil, por ejemplo, para un algoritmo cuyo tiempo de ejecución es exponencial en el tamaño de la entrada.
Entre los tiempos polinomiales se pueden distinguir los lineales $O(s)$, los cuadráticos $O(s^2)$, los cúbicos $O(s^3)$, etc.

En teoría de la complejidad, se dice que un algoritmo A tiene complejidad $O(f(s))$ si su tiempo de ejecución es $t_A(s) = O(f(s))$, y llamamos clase de complejidad al conjunto de problemas de decisión de complejidad relacionada, es decir, aquel donde para todos los problemas existe un algoritmo que los resuelve con tiempo de ejecución en $O(g(s))$.

Definición 3.16. *La clase de complejidad P es la clase de los problemas de decisión para los cuales existe un algoritmo determinista y polinomial.*

Veremos a lo largo de este capítulo que el problema de primalidad está en P.

Definición 3.17. *La clase de complejidad NP es la clase de los problemas de decisión para los cuales, en caso de tener una respuesta afirmativa, existe un certificado que nos permite comprobar en tiempo polinomial la veracidad de esta respuesta.*

Ejemplo 3.18. El problema de determinar si un número es compuesto está en NP, pues, si un número n es compuesto, entonces sabemos que tiene al menos un divisor (el certificado) tal que, si supiéramos cuál es, podríamos comprobar en tiempo polinomial que efectivamente es divisor de n y por lo tanto verificar que n es compuesto.

Definición 3.19. *La clase de complejidad co-NP es la clase de los problemas de decisión para los cuales, en caso de tener una respuesta negativa, existe un certificado que nos permite verificar en tiempo polinomial la comprobación de este problema.*

Observación 3.20. La clase co-NP es la complementaria a la clase NP. Es decir, un problema de decisión está en co-NP si y sólo si su problema complementario está en NP. Por problema complementario se entiende aquel cuyas respuestas positiva o negativa están invertidas. Por ejemplo, determinar si un número es compuesto, es el problema complementario a determinar si un número es primo.

Ejemplo 3.21. El problema de determinar si un número es primo está en co-NP pues vimos en el ejemplo 3.18 que su problema complementario (determinar si un número es compuesto) está en NP.

Está claro que todos los problemas que están en P, están también en NP, pues si podemos encontrar una solución en un tiempo razonable, podemos comprobar si algo es o no solución en un tiempo razonable. Lo que no parece tan claro es el recíproco. Si un problema que está en NP, ¿estará también en P? Es decir, ¿el que podamos comprobar fácilmente una solución, implica que podamos encontrar fácilmente una solución? Esta cuestión fue insinuada por Godel y formulada formalmente en 1970 por Cook en lo que se denomina la conjetura de Cook. Lleva mucho tiempo abierta y es uno de los denominados “7 problemas del milenio”.

3.2. El algoritmo AKS es polinomial

El problema de primalidad es un problema de decisión y hemos encontrado un algoritmo determinista que lo resuelve: el algoritmo AKS. Si probamos que este algoritmo es también polinomial, entonces tendremos que el problema de primalidad está en P.
Nos centraremos exclusivamente en ver que AKS es polinomial. No vamos a profundizar en calcular el mejor tiempo de ejecución, sino que haremos una sobreestimación del número de pasos que se realizan. Una vez terminado, sí que hablaremos de las mejoras de AKS que hacen más rápido el algoritmo.
Para ver que el algoritmo se resuelve en tiempo polinomial, una de las cosas que debemos probar es que en el Paso 2, el número r no se hace demasiado grande. Para ello vamos a introducir la siguiente notación.

Definición 3.22. *Sean $n \geq 2$ y $k \in \mathbb{N}$. Denotamos por $r(n,k)$ al menor número primo r tal que $r \mid n$ o $ord_r(n) > k$.*

Sea p primo tal que $p < r(n,k)$. Entonces, por definición de $r(n,k)$, se tiene que $p \nmid n$ y $ord_p(n) \leq k$. Llamamos $m := ord_p(n)$, es decir que m es el orden de n en $\mathbb{Z}_p$. Así, $n^m \equiv 1$ (mód p), y por lo tanto $p \mid n^m - 1$. Como $m = ord_p(n) \leq k$, en particular:

$$p \text{ divide a } \prod_{t=1}^{k}(n^t - 1)$$

Esto se tiene para todo primo $p < r(n,k)$, luego:

$$\prod_{\substack{p \text{ primo} \\ p<r(n,k)}} p \text{ divide a } \prod_{t=1}^{k}(n^t - 1) \tag{3.2}$$

Nótese que, si denotamos $\Pi := \prod_{\substack{p \text{ primo} \\ p<r(n,k)}} p$ y $N := \prod_{t=1}^{k}(n^t - 1)$, entonces (3.2) es equivalente a $\Pi \mid N$, y por lo tanto, $\Pi \leq N$. Además observamos que el número N depende de n y de k.
Recordamos la Definición 1.5 de la función recuento de primos, vista en el Capítulo 1:

$$\begin{aligned} \pi \colon \mathbb{Z}^+ &\longrightarrow \mathbb{Z}^+ \\ n &\longmapsto \pi(n) = |\{p \text{ primo } / p \leq n\}| \end{aligned}$$

El siguiente resultado es consecuencia directa del Teorema 1.6 (Teorema de los números primos). La prueba con detalle puede encontrarse en el libro [4].

Teorema 3.23 (Versión débil del Teorema de los números primos). *Existe un número real $C > 0$ tal que para todo natural $n \geq 2$:*

$$\pi(n) \geq C\frac{n}{\log_2 n}$$

Teorema 3.24 (Tamaño de $r(n,k)$). *Sean $n \geq 2$ y $k \in \mathbb{N}$. Entonces*

$$r(n,k) = O\left(k^4(\log_2 n)^2\right)$$

Demostración. Simplificamos la notación escribiendo r en lugar de $r(n,k)$. Aplicando el Teorema 3.23 vemos que:

$$\Pi = \prod_{\substack{p \text{ primo} \\ p<r}} p \geq \prod_{\substack{p \text{ primo} \\ p<r}} 2 = 2^{\pi(r-1)} \geq 2^{C\frac{r-1}{\log_2(r-1)}} > 2^{C\frac{r-1}{\log_2 r}} \geq 2^{\frac{Cr}{2\log_2 r}} \tag{3.3}$$

Y por otro lado

$$\begin{aligned} N = \prod_{t=1}^{k}(n^t - 1) < \prod_{t=1}^{k} n^t = n^1 \cdot n^2 \cdot \ldots \cdot n^k = n^{1+2+\ldots+k} = n^{\frac{k(k+1)}{2}} < n^{\frac{(k+1)^2}{2}} \\ = \left(2^{\log_2 n}\right)^{\frac{(k+1)^2}{2}} = 2^{(k+1)^2 \cdot \frac{\log_2 n}{2}} \end{aligned} \tag{3.4}$$

Como $\Pi \leq N$, de (3.3) y (3.4) se deduce que $2^{\frac{Cr}{2\log_2 r}} < 2^{(k+1)^2 \cdot \frac{\log_2 n}{2}}$. Despejando y operando se tiene:

$$\frac{r}{\log_2 r} < \frac{(k+1)^2 \log_2 n}{C} < \frac{(2k)^2 \log_2 n}{C} = \frac{4}{C}k^2 \log_2 n.$$

Es decir que $\frac{r}{\log_2 r} = O(k^2 \log_2 n)$. Hemos obtenido una estimación para $\frac{r}{log_2 r}$, y queremos encontrar una para r. Por el corolario 3.13 sabemos que $(\log_2 r)^2 = O(r)$, luego, $r(\log_2 r)^2 = O(r^2)$ y por lo tanto

$$r = O\left(\frac{r^2}{(\log_2 r)^2}\right) = O\left(k^4(\log_2 n)^2\right).$$

□

Para la aplicación en el algoritmo AKS estamos interesados en el orden de magnitud de $r_0 = r(n, 4(\log_2 n)^2)$. Por el teorema 3.24, este número satisface:

$$r_0 = r(n, 4(\log_2 n)^2) = O((4(\log_2 n)^2)^4(\log_2 n)^2) = O((\log_2 n)^{10}) \tag{3.5}$$

Para el siguiente teorema utilizaremos operaciones aritméticas de números enteros como por ejemplo la suma, resta, multiplicación, división, exponenciación modular y parte entera de la raíz k-ésima. También usaremos operaciones con polinomios (producto, división y exponenciación modular). Todas ellas se pueden resalizar en tiempo polinomial (ver [10]).

Teorema 3.25. *El algoritmo AKS está en P.*

Demostración. Ya demostramos en el capítulo anterior que AKS es un algoritmo determinista que resuelve el problema de primalidad. Luego, para probar que está en P, sólo falta ver que es polinomial, y para ello estudiaremos uno por uno los diferentes pasos del algoritmo:

- Paso 1. Comprobar si n es una potencia perfecta.
  Queremos ver si existen $m, k \in \mathbb{Z}^+$ tales que $k > 1$ y $n = m^k$. Para ello, primero calculamos los números $m_k := \lfloor \sqrt[k]{n} \rfloor$, y luego los $(m_k)^k$. Si para alguno de estos k ocurre que $(m_k)^k = n$, entonces n es una potencia perfecta; en caso contrario n no lo es.
  Pero, ¿cuántos números debemos calcular? Observamos que:

$$\begin{aligned} m_2 &= \lfloor \sqrt{n} \rfloor > 1 \\ m_3 &= \lfloor \sqrt[3]{n} \rfloor > 1 \\ &\vdots \\ m_{d-1} &= \lfloor \sqrt[d-1]{n} \rfloor > 1 \\ m_d &= \lfloor \sqrt[d]{n} \rfloor = 1 \end{aligned}$$

  De la penúltima ecuación se tiene que $\sqrt[d-1]{n} \geq 2$ y operando llegamos a que $d \leq \log_2 n + 1$. Por lo tanto, habrá que computar un máximo de $2 \log_2 n$ números (los $m_k := \lfloor \sqrt[k]{n} \rfloor$ y los $(m_k)^k$), y cada uno de esos cálculos se hace en tiempo polinomial.
- Paso 2(a). Aplicar la Criba de Eratóstenes a r.
  Veamos una estimación del número de pasos que lleva a cabo este algoritmo para un entero positivo cualquiera M.

$$\begin{array}{cccccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 \end{array}$$
$$\vdots$$
$$M$$

  - Paso 1. Se tachan todos los múltiplos de 2, es decir, $\lfloor \frac{M}{2} \rfloor < \frac{M}{2}$ números.
  - Paso 2. Se recorren todos los múltiplos de 3, es decir, $\lfloor \frac{M}{3} \rfloor < \frac{M}{3}$ números, y se tachan aquellos que no lo estuvieran ya.
    $\vdots$
  - Paso $\frac{M}{2} - 1$. Se recorren todos los múltiplos de $M/2$, es decir, $\lfloor \frac{M}{M/2} \rfloor = \frac{M}{M/2} = 2$ números, y se tachan aquellos que no lo estuvieran ya.

Aquí se detiene el algoritmo, luego, el número de operaciones será a lo sumo:

$$\frac{M}{2}+\frac{M}{3}+\ldots+\frac{M}{M/2} < \frac{M}{2}+\frac{M}{2}+\ldots+\frac{M}{2} = \frac{M}{2}\left(\frac{M}{2}-1\right) < \frac{M}{2}\cdot\frac{M}{2} = \frac{M^2}{4}.$$

Concluimos entonces que la Criba de Eratóstenes hace a lo sumo $O(M^2) = O(2^{2s})$ operaciones, es decir, está acotado superiormente por una exponencial en el tamaño de la entrada. Esto no es polinomial, pero como en nuestro caso la criba se aplica únicamente a los $r \leq r_0 = O((\log_2 n)^{10})$, el número de pasos será a lo sumo $O(r^2) = O((\log_2 n)^{20})$ un polinomio en el tamaño de la entrada.

- Pasos 2(b) y 2(c).
  Trivialmente, ambos se realizan en tiempo polinomial pues la división de enteros es polinomial, y para verificar las desigualdades sólo hace falta restar ambas partes y ver si el resultado es menor, mayor o igual que cero, lo que también es polinomial.
- Paso 2(d). Calcular $ord_r(n)$.
  Si estamos en este paso es porque necesariamente r es un primo menor estricto que n y que no lo divide, luego m.c.d$(r, n) = 1$. Recordamos que:

$$ord_r(n) = \text{mín}\{k \in \mathbb{Z}^+ \,/\, \overline{n}^k = \overline{1} \in \mathbb{Z}_r^*\}$$

  Como r es primo, $\mathbb{Z}_r^* = \{\overline{1}, \overline{2}, \ldots \overline{r-1}\}$ tiene $r-1$ elementos, por lo tanto habrá que realizar un máximo de $r-1$ cálculos:

$$\begin{aligned} \overline{n}^2 &= \overline{n}\cdot\overline{n} \\ \overline{n}^3 &= \overline{n}\cdot\overline{n}\cdot\overline{n} \\ &\vdots \\ \overline{n}^{r-1} &= \overline{n}\cdot\overline{n}\cdot\ldots\cdot\overline{n} \end{aligned}$$

  Y sabemos que la exponenciación modular se realiza en tiempo polinomial, por lo tanto, el número de operaciones elementales es polinomial. Además, el número de dígitos de $\overline{n}^k$ es:

$$\log_2(n^k) = k\log_2 n \leq r\log_2 n = O((\log_2 n)^{11})$$

  para todo $k \in \{2, 3, \ldots r-1\}$
- Paso 3. Comprobar las igualdades $[(x+\overline{a})^n]_{\overline{Q}} = [x^n+\overline{a}]_{\overline{Q}} \in \mathbb{Z}_n[x]/(\overline{Q})$ para todo $\overline{a} \in \mathbb{Z}_n$ de $\overline{1}$ a $\overline{r-1}$.
  El número de igualdades que se prueban en este paso es como máximo $r_0 = O((\log_2 n)^{10})$. Además, la esencia de comprobar cada una de estas igualdades, radica en calcular $[(x+\overline{a})^n]_{\overline{Q}}$. Para ello hacen falta $O(\log_2 n)$ multiplicaciones de polinomios. Sabemos que se hacen en tiempo polinomial, y como los coeficientes están en $\mathbb{Z}_n$ módulo $\overline{Q}$, el tamaño de los polinomios involucrados en el cálculo también es polinomial.

Concluimos entonces que el algoritmo AKS está en P, como queríamos ver. □

3.3. Mejoras del AKS

Como comentábamos al principio de la sección anterior, hemos hecho una sobreestimación del número de pasos. La cota de complejidad que se ha dado es elevada ya que nuestro objetivo era proporcionar una demostración completa y accesible, pero pruebas más sofisticadas hacen mejoras en el tiempo de ejecución del algoritmo.

3.3.1. Artículo original

El algoritmo AKS del artículo original hace más hincapié en obtener un mejor tiempo de ejecución y por lo tanto difiere en algunos aspectos del que nosotros hemos formulado en este trabajo. Agrawal, Kayal y Saxena en su ensayo probaron una versión más fuerte del Teorema 2.32, donde en el enunciado, en lugar de "a lo sumo $r-1$", declaran "a lo sumo $2\sqrt{r}\log_2 n$". Para ver esto, también habría que hacer ciertos cambios en los resultados anteriores, por ejemplo en el Corolario 2.25, la hipótesis $l \geq t-1$ puede ser reemplazada por la condición más débil $l \geq 2\sqrt{t}\log_2 n - 1$. Todo ello desemboca en un mejor tiempo de ejecución en el algoritmo ya que podemos reemplazar la cota r del tercer paso por $2\sqrt{r}\log_2 n < r$.
Además, otra cosa que debemos destacar es que realmente no es necesario exigir la primalidad de r. Podríamos estudiar los factores irreducibles de Φ_r módulo un número primo p, con r y p coprimos, y el Lema 2.29 se seguiría verificando (aunque la demostración se vuelve considerablemente más complicada). Teniendo en cuenta que este lema es el único lugar de la demostración del Teorema 2.32 donde se aplica la primalidad de r, concluimos que esta suposición puede eliminarse del teorema y por consiguiente del algoritmo. De nuevo, esto acelera la ejecución del algoritmo por dos razones:

1. Podemos prescindir de la prueba de primalidad en r (en nuestro caso, la criba de Eratóstenes)
2. El menor número natural r que verifique $ord_r(n) > 4(\log_2 n)^2$ puede no ser primo y por lo tanto, encontrarse antes que en el caso de exigir su primalidad.

Nótese que en concreto lo que se acelera es el Paso 2. En la literatura de algoritmos eficientes podemos encontrar información sobre otras mejoras de AKS utilizando los algoritmos más conocidos para la aritmética de polinomios (ver [3]).

3.3.2. Tamaño de r_0

Acabamos de ver unas mejoras del algoritmo que propusimos en un principio, y se han acortado los tiempos de ejecución de los pasos 2 y 3. No obstante, si queremos obtener una mejora clara en este último paso (que si nos fijamos

es el que requiere una mayor cantidad de tiempo, el llamado *cuello de botella* del algoritmo), debemos centrarnos en el tamaño de r_0, obtener una mejor cota superior. Recordamos que $r_0 = r(n, \lfloor 4(\log_2 n)^2 \rfloor)$ se definió como el menor número primo tal que $ord_r(n) > 4(\log_2 n)^2$ y nuestra cota original era del orden $O((\log_2 n)^{10})$, sin embargo, el siguiente resultado (del que no proporcionamos una prueba) nos permite mejorarla.

Lema 3.26. *Para cada $\varepsilon > 0$ se tiene que $r(n,k) = O((k^2 \log_2 n)^{1+\varepsilon})$.*

Observación 3.27. Nótese que esta es una versión más fuerte del Teorema 3.24

Corolario 3.28. *El número r_0 verifica $r_0 = O((\log_2 n)^{5+\varepsilon})$.*

Aunque ε es un número positivo arbitrariamente pequeño, esta cota se puede mejorar aún más si, tal y como comentábamos, no exigimos que r_0 sea primo:

$$r_0 = O((\log_2 n)^5)$$

Para estudiar en profundidad estas cotas haría falta investigar algunos teoremas y cuestiones más profundas de la teoría de números.

3.3.3. Primos de Sophie Germain. Conjetura de Hardy y Littlewood. Fouvry

A partir de ahora fijamos n y seguimos buscando un número r para el cual $ord_r(n)$ es "grande". Del Teorema 1.22 (Pequeño Teorema de Fermat) se deduce que $ord_r(n)$ es un divisor de $\varphi(r)$, luego, sería útil encontrar números para los cuales $\varphi(r)$ tenga grandes divisores primos. El estudio de los números primos r para los cuales $\varphi(r) = r - 1$ tiene grandes divisores primos, se remonta casi dos siglos, al trabajo de Sophie Germain, que estudió los números primos p para los cuales $kp+1$ también es primo, siendo k un número par. De todos estos, destaca un caso importante en el que $k = 2$.

Definición 3.29. *Un número primo p se denomina primo de Sophie Germain si $2p + 1$ también es primo.*

En relación a la función recuento de primos $\pi(m)$, definimos la función recuento de primos de Sophie Germain:

$$\begin{aligned} s\colon \mathbb{Z}^+ &\longrightarrow \mathbb{Z}^+ \\ n &\longmapsto s(m) = |\{p \text{ primo de Sophie Germain } / p \leq m\}| \\ &\qquad\quad = |\{p \text{ primo } / 2p+1 \text{ primo y } p \leq m\}| \end{aligned}$$

En 1922, los matemáticos ingleses Hardy y Littlewood formularon la siguiente conjetura:

Conjetura 3.30 (Conjetura de Hardy y Littlewood). Existe una constante C tal que $s(m) = O(\frac{m}{(\log_2 m)^2})$

Si esto es cierto, entonces se podría probar que $r_0 = O((\log_2 n)^2 + \varepsilon)$, con $\varepsilon > 0$ arbitrariamente pequeño, pero desafortunadamente, a día de hoy aún no se ha demostrado.

También cabe destacar que existe un teorema de Fouvry (ver [6]) de 1985 que estudia los números primos q tales que $q - 1$ tiene un factor primo que es mucho mayor que $\sqrt{q}$. Aunque no sean primos de Sophie Germain, al menos verifican esta propiedad y utilizando este resultado profundo y extremadamente difícil de la teoría analítica de números, se podría probar que $r = O((\log_2 n)^3)$. Esto significa una mejora considerable en nuestra cota inicial, y más teniendo en cuenta que se consideran tiempos prácticamente aceptables $r = O((\log_2 n)^4)$.

3.3.4. Variantes del algoritmo AKS

Poco después de que Agrawal, Kayal y Saxena publicaran su artículo, varias personas desarrollaron mejoras. En un principio, con los ajustes realizados por Lenstra y Pomerance (ver [11]), el AKS se situó en un tiempo de ejecución $r = O((\log_2 n)^{4'5})$. Posteriormente también realizaron cambios en el algoritmo Berstein, así como Berrizbeitia (ver [15]). Sólo comentaremos brevemente el algoritmo de Lenstra y Pomerance. La idea básica es, esencialmente, la misma que la de AKS, pero para probar las congruencias del Paso 3 se utiliza un polinomio Q diferente. Mejora considerablemente el tiempo de ejecución pero a cambio, la demostración es considerablemente más difícil.

3.4. Tests de primalidad en software comerciales

A pesar de todas las mejoras de AKS, los algoritmos de primalidad que se utilizan en el software comercial son los probabilistas, dado que sus tiempos de ejecución son mucho mejores y tienen baja probabilidad de error. De hecho, con algunas modificaciones se puede garantizar el acierto para todos los enteros positivos menores que un valor determinado. Esto se consigue comprobando todos esos números hasta la cifra en cuestión, y corrigiendo aquellos que dan error. El más utilizado en la actualidad es el test de Miller y Rabin. Todavía no se conoce una prueba de primalidad determinista que pueda remotamente competir con este algoritmo. Miller-Rabin no sólo es el más rápido en la práctica, sino que también es el más fácil de implementar. Pero no debemos olvidar que para números grandes, aunque la probabilidad sea muy baja, el éxito no está garantizado. Si quisiéramos realmente garantizar el acierto, deberíamos utilizar un

algoritmo determinista como el AKS, pero perderíamos en tiempo de ejecución.

Aunque hayamos visto que el problema de primalidad se puede resolver en tiempo polinomial, no se conoce ningún algoritmo que resuelva el problema de factorizar en primos en tiempo polinomial. A día de hoy, la seguridad de muchos protocolos criptográficos de clave pública se basa en este hecho, luego, si se encontrara tal algoritmo, habría que reconstruir muchos sistemas de seguridad de la información.

Bibliografía

[1] Agrawal, M. & Kayal, N. & Saxena, N. *PRIMES is in P.* Annals of Mathematics, 160 (2004), 781-793.

[2] Alford, W.R. & Granville, A. & Pomerance, C. *There are infinitely many Carmichael numbers.* Annals of Mathematics, Second Series, Vol. 139, No. 3 (May, 1994), pp. 703-722

[3] Bernstein, D. J. *Proving primality after Agrawal, Kayal and Saxena.* Departament of Mathematics, Statistics and Computer Science, The University of illinois at Chicago, 2003.

[4] Bressoud, D. E. *Factorization and Primality Testing.* Springer, 1989.

[5] Crandall, R. & Pomerance, C. *Prime numbers: A computational perspective.* Second edition. Springer, New York, 2005.

[6] Fouvry, E. *Théorème de Brun-Titchmarsh; application au théorème de Fermat.* Inventiones mathematicae 79 (1985) 383-408.

[7] Goto, T. & Ohno, Y. *Odd perfect numbers have a prime factor exceeding* 10^8. Mathematics of Computation, Vol. 77, Nº 263 (Jul. 2008), pp. 1859-1868.

[8] Iannucci, D. E. *The second largest prime divisor of an odd perfect number exceeds ten thousand.* Mathematics of Computation, Vol. 68, Nº 228 (1999), pp. 1749-1760.

[9] Iannucci, D. E. *The third largest prime divisor of an odd perfect number exceeds one hundred.* Mathematics of Computation, Vol. 69, Nº 230 (1999), pp. 867-879.

[10] Knuth, D. M. *The art of computer programming.* Vols. 1-4, Reading, Massachusetts: Addison-Wesley.

[11] Lenstra, H. W. Jr. & Pomerance, C. *Primality testing with Gaussian periods.* Preprint, 2005, revised April 2011.

[12] Nielsen, P. P. *Odd perfect numbers have at least nine distinct prime factors.* Mathematics of Computation, Vol. 76, Nº 260 (Oct. 2007), pp. 2109-2126-1868.

[13] Ocheml, P. & Rao, M. *Odd perfect numbers are greater than* 10^{1500}. Mathematics of Computation, Vol. 81, Nº 279 (Jul. 2012), pp. 1869-1877.

[14] Rempe-Gillen, L. & Waldecker, R. *Primality Testing for Beginners.* American Mathematical Society, Providence, Rhode Island, 2014.

[15] Santiago Zaragoza, A. C. & Santiago Puertas, M. J. *La teoría elemental de números y su historia.* Aebius, 2012.

Primality tests: the AKS algorithm

Sandra Beatriz Jiménez Carballo
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100842418@ull.edu.es

Abstract

This work begins with an introductory chapter about prime numbers, some families of special numbers, primality tests and some important results about these topics. This constitutes a basis for the main objective of our memory: to study in detail the AKS Algorithm. This is the first deterministic and polynomial-time primality test. This test is built as a generalization for polynomials of Fermat's Little Theorem. After a series of technical results, we state and prove the Theorem of Agrawal, Kayal and Saxena; the key result for obtaining the algorithm. As a consequence, we get that the problem of testing primality of an integer is in the complexity class P (of polynomial-time solvable problems). Finally, we comment on some improvements of the algorithm provided by other researchers.

1. Introduction

A problem always present throughout the history of mathematics is the one of determining whether a number is prime or not. To solve it, various primality tests have been developed. These tests used to be either deterministic or efficient, but not both. Thus, one of the great dilemmas in the intersection of number theory and computational complexity theory is: is the problem of testing primality in P? Recently, in 2002, the computer scientists Agrawal, Kayal and Saxena provide a positive answer to this question by publishing their paper SPRIMES is in P [1]. There, they present the first deterministic and efficient primality test, the so-called AKS algorithm.

2. Prime numbers

Definition 1 *Let $p \in \mathbb{Z}$, with $p \geq 2$. We say that p is a prime number if it is divisible only by itself and 1. A non-prime number is called a composite number.*

Theorem 1 (Fermat's Little Theorem) *Let p be prime and $a \in \mathbb{Z}$. Then $a^p \equiv a \pmod p$.*

Two important primality tests are the classical Sieve of Eratosthenes and the Miller-Rabin algorithm. The first one is deterministic and inefficient, unlike the second, which is probabilistic and fast.

Algorithm 1 The Sieve of Eratosthenes
Require: A natural number $n \geq 2$.
Ensure: All prime numbers less than or equal to n.
1: **for** $i = 2$ **to** n **do**
2: $a_i \leftarrow i$
3: **end for**
4: $j \leftarrow 2$
5: **while** $j^2 \leq n$ **do**
6: **if** $a_j \neq 0$ **then**
7: **for** $i = 2$ **to** $\lfloor n/j \rfloor$ **do**
8: $a_{i \cdot j} \leftarrow 0$
9: **end for**
10: **end if**
11: $j \leftarrow j + 1$
12: **end while**
13: **for** $i = 2$ **to** n **do**
14: **if** $a_i \neq 0$ **then**
15: write i
16: **end if**
17: **end for**

Algorithm 2 Miller-Rabin Test
Require: A natural number $n > 1$.
1: If n is even and $n > 2$, or if n is a power of some other natural number, answer "n is composite".
2: Otherwise, write $n-1 = d \cdot 2^l$ and choose a number $a \in \{1, 2, \ldots, n-1\}$ at random.
3: If g.c.d.$(a, n) \neq 1$, answer "n is composite".
4: Otherwise, compute $b := a^d$ (mód n).
5: If $b = 1$, then answer "n is probably prime".
6: Otherwise, compute $b, b^2, b^4, \ldots, b^{2^{l-1}}$ (mod n).
7: If none of these numbers are congruent to -1 modulo n, then answer "n is composite".
8: Otherwise, answer "n is probably prime".

3. AKS algorithm

The starting point is an extension of Fermat's Little Theorem:

Theorem 2 (Fermat for polynomials) *Let p be a prime number. Then for all polynomials $P(x)$ with integer coefficients:*

$$\overline{P}(x)^p = \overline{P}(x^p) \in \mathbb{Z}_p[x]$$

The following theorem constitutes the key result to derive the algorithm:

Theorem 3 ***(Theorem of Agrawal, Kayal and Saxena)*** *Let n be a composite number, let p be a prime divisor of n and let r be a prime that does not divide n and such that $ord_r(n) > 4(\log_2 n)^2$. Also set $Q(x) = x^r - 1$. If n is not a power of p, then there are at most $r-1$ polynomials of the form $P(x) = x + a$, with $a \in \{0, 1, \ldots, p-1\}$, that satisfy:*

$$[\overline{P}(x)^n]_{\overline{H}} = [\overline{P}(x^n)]_{\overline{H}} \in \mathbb{Z}_p[x]/(\overline{H})$$

Algorithm 3 AKS algorithm
Require: A natural number $n \geq 2$
1: If n is a perfect power, then answer "n is composite".
2: Otherwise do the following for $r = 2, 3, 4, \cdots$:
(a) Test whether r is prime (for example by using the Sieve of Eratosthenes).
(b) If $r \mid n$ and $r < n$, answer "n is composite".
(c) If $r = n$, answer "n is prime".
(d) Compute $ord_r(n)$.
(e) If r is prime and $ord_r(n) > 4(\log_2 n)^2$, set $Q(x) := x^r - 1$ and continue with Step 3.
3: Test the equalities:

$$[(x + \overline{a})^n]_{\overline{Q}} = [x^n + \overline{a}]_{\overline{Q}} \in \mathbb{Z}_n[x]/(\overline{Q})$$

for all $\overline{a} \in \mathbb{Z}_n$ from $\overline{1}$ to $\overline{r-1}$.
4: If one of these congruences is not satisfied, answer "n is composite". Otherwise answer "n is prime".

4. The primality problem

As a consequence of the AKS algorithm we get the following.

Theorem 4 *The primality problem is in P.*

After [1] was published, other researchers such as Lenstra, Pomerance, Berstein or Berrizbeitia, have developed improvements of the algorithm with lower execution times.
Despite the theoretical interest of AKS, primality tests that are implemented nowadays in commercial software and in cryptographic security are often probabilistic. The main reason is that the execution time of probabilistic methods is much lower meanwhile the probability of error is very low.

References

[1] AGRAWAL, M. & KAYAL, N. & SAXENA, N. *PRIMES is in P.* Annals of Mathematics, 160 (2004), 781-793.
[2] BERNSTEIN, D. J. *Proving primality after Agrawal, Kayal and Saxena.* Departament of Mathematics, Statistics and Computer Science, The University of illinois at Chicago, 2003.
[3] BRESSOUD, D. E. *Factorization and Primality Testing.* Springer, 1989.
[4] KNUTH, D. M. *The art of computer programming.* Vols. 1-4, Reading, Massachusetts: Addison-Wesley.
[5] REMPE-GILLEN, L. & WALDECKER, R. *Primality Testing for Beginners.* American Mathematical Society, Providence, Rhode Island, 2014.