PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES

JOSÉ LUIS CÁNOVAS SÁNCHEZ

Tutores
LEANDRO MARÍN MUÑOZ
ANTONIO JOSÉ PALLARÉS RUIZ

Facultad de Matemáticas
Universidad de Murcia

José Luis Cánovas Sánchez: *Pruebas de Conocimiento Cero y sus Aplicaciones,* Junio 2017

RESUMEN

TODO

Short summary of the contents in -English- Spanish... a great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

ABSTRACT

1500 words of Introduction in English...

ÍNDICE GENERAL

1 INTRODUCCIÓN 1

2 PRELIMINARES 3

2.1 Preliminares de Álgebra 3

2.2 Preliminares de Computación 5

2.3 Preliminares de Probabilidad 9

2.4 Preliminares de Grafos 9

3 RESIDUOS CUADRÁTICOS 11

3.1 Primeras propiedades 11

3.2 Símbolo de Legendre 12

3.3 Símbolo de Kronecker-Jacobi 13

3.4 El problema de residuosidad cuadrática 14

4 PRUEBAS DE CONOCIMIENTO CERO 15

4.1 Definición 15

4.2 Estudio de ZKPs 15

5 APLICACIONES DE ZKP 17

6 IMPLEMENTACIONES 19

Appendix 21

A CÓDIGO FUENTE 23

BIBLIOGRAFÍA 25

ÍNDICE DE FIGURAS

Figura 1 Conjetura de las relaciones entre clases **NP**, **co-NP**, **NPC**, **P**. 9

ÍNDICE DE TABLAS

LISTINGS

Listing 1 A floating example (`listings` manual) 23

ACRONYMS

1 INTRODUCCIÓN

2 PRELIMINARES

Para poder comprender los resultados de los siguientes capítulos necesitaremos recordar ciertas definiciones y propiedades de álgebra que se cursan durante el grado, e introducir otros preliminares de algoritmia que formalizan el estudio. No incluiremos demostraciones, pues son los conceptos básicos de donde partiremos para desarrollar el resto del trabajo, pero el lector que quiera conocerlas puede consultar las referencias en **TODO: cite**.

2.1 PRELIMINARES DE ÁLGEBRA

ARITMÉTICA ELEMENTAL

Definición 2.1. Un entero d se dice que es el **máximo común divisor** de dos enteros a y b si es el mayor entero que divide a ambos. Lo denotaremos $d = \mathrm{mcd}(a, b)$.

Euclides describió en su obra *Los Elementos* un método para calcular el mcd de dos enteros, que hoy en día se conoce como *Algoritmo de Euclides*. La propiedad que utiliza el algoritmo para calcular el mcd es la siguiente:

Proposición 2.2. *Sean a, $b \in \mathbb{Z}$. Entonces, para todo $\alpha \in \mathbb{Z}$ se tiene:*

$$\mathrm{mcd}(a, b) = \mathrm{mcd}(a, b - \alpha a) = \mathrm{mcd}(a - \alpha b, b).$$

En particular, cuando $b \neq 0$ y la división entera de a entre b es $a = bq + r$, tenemos que $\mathrm{mcd}(a, b) = \mathrm{mcd}(b, r)$.

Algoritmo 2.3 (Euclides)**.** Encuentra el mcd de a, $b \in \mathbb{Z}$:

1. Inicializa $r_0 = a$ y $r_1 = b$.

2. Calcula las siguientes divisiones euclídeas

$$\begin{aligned} r_0 &= q_1 r_1 + r_2 \\ r_1 &= q_2 r_2 + r_3 \\ &\dots \\ r_{n-3} &= q_{n-2} r_{n-2} + r_{n-1} \\ r_{n-2} &= q_{n-1} r_{n-1} + r_n \end{aligned}$$

   hasta que se obtenga un $r_n = 0$, con $r_{n-1} \neq 0$.

3. Como $b = r_1 > r_2 > ... \geqslant 0$ y cada r_i es entero, para $i = 1, 2, ...$, se obtiene $r_n = 0$ en un número finito de pasos y acaba el algoritmo con $\mathrm{mcd}(a, b) = r_{n-1}$.

A partir del *Algoritmo de Euclides* se puede expresar d como una "combinación $\mathbb{Z}$-lineal" de a y b:

$$d = as + bt$$

conocida como la *Identidad de Bézout*.

El algoritmo se conoce como *Algoritmo de Euclides extendido*:

Algoritmo 2.4 (Euclides extendido)**.** Encuentra el $d = \mathrm{mcd}$ de a, $b \in \mathbb{Z}$ y valores s, $t \in \mathbb{Z}$ tal que $d = as + bt$.

1. Inicializa $r_0 \leftarrow a, r_1 \leftarrow b, s_0 \leftarrow 1, t_0 \leftarrow 0, s_1 \leftarrow 0, t_1 \leftarrow 1$
   $i \leftarrow 1$
2. Mientras $r_i \neq 0$:
   Calcule la división euclídea $r_{i-1} = q_i r_i + r_{i+1}$
   $s_{i+1} \leftarrow s_{i-1} - q_i s_i$
   $t_{i+1} \leftarrow t_{i-1} - q_i t_i$
   $i \leftarrow i + 1$
3. $d = r_{i-1} \quad s = s_{i-1} \quad t = t_{i-1}$

Observación. Los valores de s y t no tienen por qué ser únicos:
$a(s - kb) + b(t + ka) = as - kba + bt + kba = as + bt = d$

Utilizaremos la Identidad de Bézout para calcular los inversos en aritmética modular.

GRUPOS Y ANILLOS

CONGRUENCIAS

Definición 2.5. Sean a, b, $n \in \mathbb{Z}$, $n \neq 0$, diremos que a y b son **congruentes módulo** n, y lo escribiremos $a \equiv b \bmod n$, si la diferencia $a - b$ es múltiplo de n.

Cuando $a \equiv b \bmod n$ decimos que b es un ***residuo*** *de* a *módulo* n.

Proposición 2.6. *La relación de* **congruencia módulo** n *es una relación de equivalencia, es decir, es reflexiva, simétrica y transitiva.*

Esto establece una relación de equivalencia en $\mathbb{Z}$, en la que la **clase** de un entero a módulo n es $\overline{a} = \{a + kn\}_{k \in \mathbb{Z}}$. Cuando no exista confusión, escribiremos la clase de equivalencia $\overline{a}$ como a. El correspondiente conjunto cociente, de las *clases de resto módulo* n, es $\mathbb{Z}_n = \{\overline{0}, \overline{1}, \ldots \overline{n-1}\}$, y hereda la suma y producto de $\mathbb{Z}$ convirtiéndose en un anillo conmutativo con neutros $\overline{0}$ para la suma y $\overline{1}$ para el producto.

Teorema 2.7. *El anillo* $\mathbb{Z}_n$ *es un cuerpo cuando* n *es un número primo.*

Teorema 2.8 (Euler)**.** *Si* x *es coprimo con*

Si tenemos ahora dos enteros a y b coprimos, es decir, $\mathrm{mcd}(a, b) = 1$, usando el *algoritmo de Euclides extendido* podemos encontrar r y s de la *Identidad de Bézout* tales que:

$$as + bt = 1$$

Si a esta igualdad le aplicamos módulo b, obtenemos el inverso de a en $\mathbb{Z}_b$:

$$\begin{array}{ccccc} (as + bt) \bmod b & \equiv & as \bmod b & \equiv & 1 \bmod b \\ \overline{as + bt} & = & \overline{as} & = & \overline{1} \end{array}$$

Así hemos demostrado el siguiente resultado:

Proposición 2.9. *Si* $\mathrm{mcd}(a, n) = 1$, *entonces el elemento inverso* a^{-1}, $0 < a^{-1} < n$, *existe y* $aa^{-1} \equiv 1 \bmod n$.

2.2 PRELIMINARES DE COMPUTACIÓN

Teoría de complejidad algorítmica, problema de P NP, problema RSA de factorizar N, problemas de decisión, estadística usada en el estudio de los ZKP (*ensembles*), *probabilistic computations*, ...

La mayor parte está en los primeros capítulos de Fundamentals of Computer Security, y de sus referencias se podrá sacar más detallado.

En esta sección introducimos unos preliminares de complejidad computacional que nos permitirán entender de manera básica por qué se utilizan ciertos problemas matemáticos en la seguridad informática. Comencemos con unas definiciones:

Definición 2.10. Llamamos *problema* a la descripción general de una tarea que depende de unos parámetros. La *definición* de un problema consta de dos partes. La primera da el escenario del problema, describiendo los parámetros necesarios. La segunda parte indica una pregunta de la que se espera una respuesta o solución.

Ejemplo 2.11. Vamos a considerar el problema de multiplicar dos matrices. La definición del problema sería:

Nombre:	Problema multiplicación de matrices.
Parámetros:	Dos matrices A_1 y A_2.
Pregunta:	¿Cuál es la matriz A tal que $A = A_1 \cdot A_2$?

Definición 2.12. Una *instancia* de un problema es el caso particular de un problema al que se le han dado valores a los parámetros.

Definición 2.13. Un *algoritmo* es una lista de instrucciones que para una instancia produce una respuesta correcta. Se dice que un algoritmo *resuelve* un problema si devuelve respuestas correctas para todas las instancias del problema.

Definición 2.14. Se llama *problema de decisión* a un problema cuya respuesta está en el conjunto $\mathcal{B} = \{\mathtt{Verdadero}, \mathtt{Falso}\}$.

No todos los problemas son de decisión, pero en general es fácil transformarlos en un problema de decisión cambiando la *pregunta*, y los algoritmos que resuelven un problema de decisión suelen también ser fáciles de adaptar para el problema original.

Ejemplo 2.15. El problema del ejemplo anterior se puede transformar a un problema de decisión:

Nombre:	Problema multiplicación de matrices.
Parámetros:	Tres matrices A, A_1 y A_2.
Pregunta:	¿$A = A_1 \cdot A_2$?

Un algoritmo para solucionarlo puede primero comprobar las dimensiones de las matrices, y si no concuerdan con las de la multiplicación, puede responder $\mathtt{Falso}$ sin más operaciones, pero cuando las dimensiones concuerden, deberá realizar la multiplicación de A_1 por A_2 y luego comparar A con el producto obtenido.

NOTACIONES ASINTÓTICAS Se utilizan para estudiar cómo crece el tiempo de ejecución, la memoria usada o cualquier otro recurso, de un algoritmo dependiendo del tamaño de los datos de entrada. Nos interesa en particular una:

Definición 2.16. Sea $f : \mathbb{N} \to \mathbb{R}^+$. Definimos la notación *O grande* u *orden* de f al conjunto de funciones de $\mathbb{N}$ a $\mathbb{R}^+$ acotadas superiormente por un múltiplo positivo de f a partir de cierto valor $n \in \mathbb{N}$:

$O(f) = \{t : \mathbb{N} \to \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N} \ : \ t(n) \leqslant c \cdot f(n) \quad \forall n \geqslant n_0\}$.

Para estudiar el tiempo de ejecución, $t : \mathbb{N} \to \mathbb{R}^+$ (el tiempo promedio, el caso más desfavorable, ...), de un algoritmo, buscaremos una función f que acote a t lo más de cerca posible. Para ello definimos la relación de orden:

Definición 2.17. Sean $f, g : \mathbb{N} \to \mathbb{R}^+$. Diremos que $O(f) \leqslant O(g)$ si $\forall t \in O(f)$ se tiene que $t \in O(g)$, es decir, $O(f) \subseteq O(g)$.

Comparación de órdenes: $O(1) \leqslant O(\ln n) \leqslant O(\sqrt{n}) \leqslant O(n) \leqslant O(n \ln n) \leqslant O(n^c) \leqslant O(n^{\ln n}) \leqslant O(c^n) \leqslant O(n!) \leqslant O(n^n)$, *donde* $c > 1$ *constante.*

Para acotar t buscaremos el menor $O(f) : t \in O(f)$.

CLASES DE COMPLEJIDAD Para estudiar los problemas de decisión los organizamos en clases de complejidad, según sus mejores algoritmos que los solucionan.

Definición 2.18. Se llama algoritmo de *tiempo de cálculo polinomial* al algoritmo cuya función de tiempo del caso más desfavorable es de orden $O(n^k)$, donde n es el tamaño de la entrada y k una constante. Cualquier algoritmo cuyo tiempo de ejecución no puede acotarse de esa manera se llama de *tiempo de cálculo exponencial*.

A veces se denominan *buenos* o *eficientes* a los algoritmos polinomiales, e *ineficientes* a los exponenciales. Sin embargo, hay casos particulares donde no ocurre. Lo más importante al tratar con algoritmos polinomiales es el grado del polinomio, k, que indica su orden de complejidad. Por ejemplo, consideremos un problema cuyo mejor algoritmo tarda n^{100} instrucciones, es de orden polinómico, pero prácticamente intratable a partir de $n = 2$, y por otra parte, un problema que utilice $2^{0,00001n}$ instrucciones podría tratar casos hasta de $n = 10^6$ sin dificultad. Por esto, en criptografía se debe considerar el caso promedio sobre el caso más desfavorable.

Definición 2.19. El conjunto de problemas de decisión que se pueden resolver en tiempo polinomial se llama clase **P**.

Definición 2.20. Se llama clase de complejidad **NP** al conjunto de problemas de decisión donde una respuesta que sea $\mathsf{Verdadero}$ se puede verificar en tiempo polinomial, dada cierta información extra, que llamaremos *certificado*.

Definición 2.21. Se llama clase de complejidad **co-NP** al conjunto de problemas de decisión donde una respuesta que sea Falso se puede verificar en tiempo polinomial, dado el correspondiente *certificado*.

Es inmediato que que **P** $\subseteq$ **NP** y **P** $\subseteq$ **co-NP**.

Ejemplo 2.22 (Problema en **NP**). Consideremos el siguiente problema de decisión:

Nombre:	Problema del entero compuesto.
Parámetros:	Un entero positivo n.
Pregunta:	¿Es n compuesto? Es decir, ¿existen enteros $a, b > 1$ tal que $n = ab$?

El problema pertenece a **NP** porque se puede comprobar en tiempo polinomial que n es compuesto si nos dan su *certificado*, un divisor a de n, tal que $1 < a < n$. Basta usar la división euclídea de n entre a y ver que el resto es 0. Sin embargo, aún no se conoce si pertenece a **P**.

Definición 2.23. Sean dos problemas de decisión $L_1, L_2 \in$ **NP**, con correspondientes conjuntos de instancias I_1 e I_2. Sean I_1^+ e I_2^+ los subconjuntos de todas las instancias "$\mathtt{Verdaderas}$" de I_1 e I_2, respectivamente. Decimos que L_1 es *reducible en tiempo polinomial* a L_2, $L_1 \leqslant_P L_2$, si existe una función $f : I_1 \Rightarrow I_2$ que cumple:

1. Es ejecutable en tiempo polinomial.
2. $x \in I_1^+ \Leftrightarrow f(x) \in I_2^+$.

De manera más informal, si $L_1 \leqslant_P L_2$, entonces L_2 es computacionalmente tan difícil como L_1, o de otro modo, L_1 no es más difícil que L_2.

Definición 2.24. Un problema de decisión L se dice que es **NP**-completo, o **NPC** si:

(i) $L \in$ **NP**, y

(ii) $L_1 \leqslant_P L \quad \forall L_1 \in$ **NP**.

Los problemas **NPC** son los más difíciles entre los **NP** en el sentido de que son al menos tan difíciles como el resto de problemas en **NP**. Veamos el problema **NPC** más característico:

Nombre:	Problema de satisfacibilidad booleana (SAT).
Parámetros:	Una colección finita C de expresiones booleanas con variables y sin cuantificadores.
Pregunta:	¿Hay alguna asignación de las variables que haga $\mathtt{Verdadero}$ a C?

Teorema 2.25 (Teorema de Cook (1971))**.** *El problema de satisfacibilidad booleana es* ***NPC****.*

Para conocer más sobre el problema y la demostración se puede consultar [1].

O expresado de otra manera:

Teorema 2.26. *Todo problema $Q \in$ **NP** se puede reducir en tiempo polinomial al problema de satisfacibilidad booleana.*
$\forall Q \in$ ***NP***, $Q \leqslant_P \mathrm{SAT}$.

A día de hoy conocemos ciertas propiedades sobre las clases de equivalencia, pero quedan muchas cuestiones sin resolver:

- Uno de los siete problemas del milenio, ¿**P** = **NP**?
- ¿**NP** = **co-NP**?
- ¿**P** = **NP** ∩ **co-NP**?

La opinión de los expertos en base a ciertas evidencias es que la respuesta a las tres es NO, pero hasta que se encuentren demostraciones formales de cada una, quedarán en conjeturas. Y basándose en esas conjeturas es donde se apoya la seguridad informática.

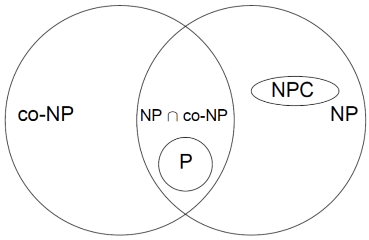

Figura 1: Conjetura de las relaciones entre clases **NP**, **co-NP**, **NPC**, **P**.

Existen tres problemas que se conoce que pertenecen a **NP**, pero se desconoce a día de hoy si pertenecen a **P** o **NPC**: el isomorfismo de grafos, el logaritmo discreto y la factorización de enteros.

En los siguientes capítulos los estudiaremos como base de diferentes pruebas de conocimiento cero.

ALGORITMOS PROBABILÍSTICOS Para intentar resolver la infactibilidad computacional de ciertos problemas, surge un modelo alternativo de computación que utiliza métodos probabilísticos. Estos métodos no pueden asegurar cotas superiores absolutas de tiempo, e incluso pueden devolver una respuesta errónea. Sin embargo, dadas unas cotas muy pequeñas de errores, en la práctica, ciertos métodos probabilísticos son más eficientes que los algoritmos conocidos, pues el tiempo de ejecución *esperado* del método, calculado probabilísticamente, es menor que el orden del algoritmo original.

2.3 PRELIMINARES DE PROBABILIDAD

2.4 PRELIMINARES DE GRAFOS

3 RESIDUOS CUADRÁTICOS

TODO : Párrafo de introducción al capítulo y referencias

Teoría de símbolos de Lebesgue, ..., residuos cuadráticos, cálculo de raíz discreta?

3.1 PRIMERAS PROPIEDADES

Definición 3.1. Sea $a \in \mathbb{Z}_n^*$. Se dice que a es un *residuo cuadrático* módulo n, o un *cuadrado* módulo n, si existe un $x \in \mathbb{Z}_n^*$ tal que $x^2 \equiv a \bmod n$. Si no existe dicho x, entonces a se llama un *no-residuo cuadrático* módulo n.

El conjunto de todos los residuos cuadráticos módulo n de $\mathbb{Z}_n^*$ los denotaremos como Q_n o bien como $\mathbb{Z}_n^{Q+}$. Al conjunto de los no-residuos cuadráticos lo denotamos como $\overline{Q_n}$.

Observación. Por definición $0 \notin \mathbb{Z}_n^*$, y por tanto $0 \notin Q_n$ y $0 \notin \overline{Q_n}$.

Definición 3.2. Sea $a \in Q_n$. Si $x \in \mathbb{Z}_n^*$ satisface $x^2 \equiv a \bmod n$, entonces x se llama *raíz cuadrada* módulo n de a.

Proposición 3.3. *Sea p un primo impar. Se cumple que $|Q_p| = \frac{p-1}{2}$ y $|\overline{Q_p}| = \frac{p-1}{2}$, es decir, la mitad de los elementos de $\mathbb{Z}_p^*$ son residuos cuadráticos, y la otra mitad no-residuos cuadráticos.*

Demostración. Sea $\alpha \in \mathbb{Z}_p^*$ un generador de $\mathbb{Z}_p^*$. Un elemento $a \in \mathbb{Z}_p^*$ es un residuo cuadrático módulo p sii $a \equiv \alpha^i \bmod p$ donde i es un entero par. Como p es primo, $\phi(p) = p - 1 = |\mathbb{Z}_p^*|$, que es un entero par, y de ahí se sigue el enunciado. □

Ejemplo 3.4. Para $p = 13$ tenemos que $\alpha = 6$ es un generador de $\mathbb{Z}_{13}^*$. Las potencias de α módulo 13 son:

i	1	2	3	4	5	6	7	8	9	10	11	12
$6^i \bmod 13$	6	10	8	9	2	12	7	3	5	4	11	1

Lo que nos da $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ y $\overline{Q_{13}} = \{2, 5, 6, 7, 8, 11\}$.

Proposición 3.5. *Sea n un producto de dos primos impares p y q, $n = pq$. Entonces $a \in \mathbb{Z}_p^*$ es un residuo cuadrático módulo n, $a \in Q_n$ si y solo si $a \in Q_p$ y $a \in Q_q$. Se sigue que $|Q_n| = |Q_p| \cdot |Q_q| = \frac{(p-1)(q-1)}{4}$, y por tanto $|\overline{Q_n}| = |\mathbb{Z}_n^*| - |Q_n| = \frac{3(p-1)(q-1)}{4}$.*

Demostración. Si a es un residuo cuadrático módulo $n = pq$, $a \equiv x^2 \bmod n$, es inmediato que en módulos p y q se cumple $a \equiv x^2 \bmod p$, $a \equiv x^2 \bmod q$.

Si tenemos que a es un residuo cuadrático módulo p, $a \equiv x_p^2 \bmod p$, y también módulo q, $a \equiv x_q^2 \bmod q$, por el Teorema Chino de los Restos existe un x tal que:
$x \equiv x_p \bmod p$
$x \equiv x_q \bmod q$
De modo que, elevando al cuadrado:
$x^2 \equiv x_p^2 \equiv a \bmod p$
$x^2 \equiv x_q^2 \equiv a \bmod q$
Por lo que $x^2 \equiv a \bmod n$.

□

3.2 SÍMBOLO DE LEGENDRE

Para identificar los residuos cuadráticos disponemos de una herramienta muy útil:

Definición 3.6. Dados un primo impar p y un entero a, se define el *símbolo de Lebesgue* $\left(\frac{a}{p}\right)$ como

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & si\ a \equiv 0 \bmod p \\ 1, & si\ a \in Q_p \\ -1, & si\ a \in \overline{Q_p} \end{cases}$$

Veamos ahora algunas propiedades del símbolo de Legendre:

Teorema 3.7 (Criterio de Euler)**.** *Sea* p *un primo impar. Sea* $a \not\equiv 0 \bmod p$. *Entonces:*

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \bmod p$$

Demostración. Observemos primero que las raíces de 1 módulo p son 1 y $-1 \bmod p$. También que por el Teorema de Euler, $a^{\phi(p)} \equiv a^{p-1} \equiv 1 \bmod p$.

De este modo, tenemos que $a^{\frac{p-1}{2}} \equiv 1$ ó $-1 \bmod p$.

Ahora demostrar el teorema es equivalente a demostrar que $a^{(p-1)/2} \equiv 1 \bmod p$ sii a es un residuo cuadrático.

Supongamos que a es un residuo cuadrático módulo p. Sea x tal que $x^2 \equiv a \bmod p$. Entonces, $a^{\frac{p-1}{2}} \equiv x^{(p-1)} \equiv 1 \bmod p$, de nuevo por el Teorema de Euler.

Sea ahora $a^{\frac{p-1}{2}} \equiv 1 \bmod p$.

Tomamos g un generador de $\mathbb{Z}_p^*$, de modo que $a \equiv g^r \bmod p$. Sustituyendo: $g^{r\frac{p-1}{2}} \equiv 1 \bmod p$, y como g tiene orden $p-1$, queda

$g^{\frac{r}{2}} \equiv 1 \bmod p$, de donde deducimos que necesariamente r es un entero par, $r = 2s$.

Construimos $x \equiv g^s \bmod p$, que cumple: $x^2 \equiv g^{2s} \equiv g^r \equiv a \bmod p$, de modo que a es un residuo cuadrático módulo p.

□

Proposición 3.8 (Propiedad multiplicativa del símbolo de Lebesgue)**.** *Sean* a *y* b *enteros coprimos con* p*, un primo impar. Entonces:*

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right).$$

En particular, el producto de dos no-residuos cuadráticos es un residuo cuadrático.

Demostración. Utilizando el Criterio de Euler:

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = a^{(p-1)/2} \cdot b^{(p-1)/2} = (ab)^{(p-1)/2} = \left(\frac{ab}{p}\right)$$

□

Teorema 3.9 (Ley de reciprocidad cuadrática)**.** *Sean* p *y* q *primos impares distintos, se cumple:*

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{(p-1)(q-1)/4}.$$

O de otro modo:

$$\left(\frac{p}{q}\right) = \begin{cases} \left(\frac{q}{p}\right) & \text{Si} \quad p \equiv 1 \bmod 4 \quad \text{ó} \quad q \equiv 1 \bmod 4 \\ -\left(\frac{q}{p}\right) & \text{Si} \quad p \equiv q \equiv 3 \bmod 4 \end{cases}.$$

3.3 SÍMBOLO DE KRONECKER-JACOBI

El símbolo de Legendre está definido para módulos un primo impar p. Ahora vamos a ver una generalización del concepto para cualquier módulo N .

Definición 3.10. Sean $a, N \in \mathbb{Z}$, con $N = p_1 p_2 \cdots p_r$, donde los p_i son primos, no necesariamente distintos, incluyendo el 2 y el -1 para el signo.

Definimos el *Símbolo de Kronecker-Jacobi* $\left(\frac{a}{N}\right)$ como

$$\left(\frac{a}{N}\right) = \prod_{i=1}^{r} \left(\frac{a}{p_i}\right)$$

donde $\left(\frac{a}{p_i}\right)$ es el Símbolo de Legendre para los $p_i > 2$, y para los casos $p = 2$ y $p = -1$ definimos:

$$\left(\frac{a}{2}\right) = \begin{cases} 0, & \text{si } a \text{ es par.} \\ (-1)^{(a^2-1)/8}, & \text{si } a \text{ es impar.} \end{cases}$$

y

$$\left(\frac{a}{2}\right) = \begin{cases} 1, & \text{si } a \geqslant 0 \\ -1, & \text{si } a < 0. \end{cases}$$

Observación. A diferencia del Símbolo de Legendre, el Símbolo de Jacobi $\left(\frac{a}{N}\right)$ no indica si a es un residuo cuadrático módulo N. Es cierto que si $a \in Q_N$, su Símbolo de Jacobi será $\left(\frac{a}{N}\right) = 1$, pero el contrario no se cumple.

Igual que antes, veamos algunas propiedades del Símbolo de Jacobi:

Teorema 3.11. *Propiedades del Símbolo de Jacobi:*

(i) $\left(\frac{a}{N}\right) = 0$ *si y sólo si* $\mathrm{mcd}(a, N) = 1$.

(ii) *Para cada* a, b *y* c *enteros, tenemos:*

$$\left(\frac{ab}{c}\right) = \left(\frac{a}{c}\right)\left(\frac{b}{c}\right), \quad \left(\frac{a}{bc}\right) = \left(\frac{a}{b}\right)\left(\frac{a}{c}\right) \quad \text{si } bc \neq 0.$$

(iii) *Fijado* $N > 0$, *el símbolo* $\left(\frac{a}{N}\right)$ *es periódico en* a *con periodo* N *si* $N \not\equiv 2 \bmod 4$, *en otro caso, es periódico con periodo* $4N$.

(iv) *Fijado* $a \neq 0$, *el símbolo* $\left(\frac{a}{N}\right)$ *es periódico en* N *con periodo* $|a|$ *si* $a \equiv 0$ ó $1 \bmod 4$, *en otro caso, es periódico con periodo* $4|a|$.

(v) *Las fórmulas del Teorema 3.9 se siguen verificando si* p *y* q *son enteros impares positivos, ya no necesitan ser primos.*

3.4 EL PROBLEMA DE RESIDUOSIDAD CUADRÁTICA

Indicar el problema, que es NP y que se puede reducir al de encontrar la factorización de N

4 PRUEBAS DE CONOCIMIENTO CERO

4.1 Definición

4.2 Estudio de ZKPs

Una subsección por cada ZKP analizado.

5 APLICACIONES DE ZKP

?

Aplicación de ZKP en los certificados de Idemix. Analizar cómo realizan pruebas de Y, O, etc.

6 IMPLEMENTACIONES

?

Implementaciones de símbolos, raíz discreta, ZKPs, ...

APPENDIX

A

CÓDIGO FUENTE

Listing 1: A floating example (`listings` manual)

```
for i:=maxint downto 0 do
begin
{ do nothing }
end;
```

BIBLIOGRAFÍA

[1] David S. Johnson Michael R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. First Edition. Series of Books in the Mathematical Sciences. W. H. Freeman, 1979. ISBN: 0716710455,9780716710455. URL: http://gen.lib.rus.ec/book/index.php?md5=77F747B4A3CC87AAF94F6A9EA1CBC1CF.

DECLARACIÓN DE ORIGINALIDAD

Yo, José Luis Cánovas Sánchez, autor del TFG PRUEBAS DE CONOCIMIENTO CERO Y SUS APLICACIONES, bajo la tutela de los profesores Leandro Marín Muñoz y Antonio José Pallarés Ruiz, declaro que el trabajo que presento es original, en el sentido de que ha puesto el mayor empeño en citar debidamente todas las fuentes utilizadas.

Murcia, Junio 2017

José Luis Cánovas Sánchez