Algoritmos en teoría de números

IIC2283

Para recordar: aritmética modular

Dados dos números $a, b \in \mathbb{Z}$, si $b > 0$ entonces existen $\alpha, \beta \in \mathbb{Z}$ tales que $0 \leq \beta < b$ y

$$a = \alpha \cdot b + \beta$$

Además, estos números α, β son únicos

β es llamado el resto de la división entera entre a y b, y es denotado como $a \bmod b$

- Por ejemplo, $8 \bmod 3 = 2$, $9 \bmod 3 = 0$ y $(-8) \bmod 3 = 1$

Para recordar: aritmética modular

Definición

$b \equiv c \bmod n$ *si* n *divide a* $(c - b)$

Usamos la notación $n|m$ para indicar que n divide a m

- $b \equiv c \bmod n$ si $n|(c - b)$

Para recordar: algunas propiedades básicas

Proposición

1. $a \equiv b \bmod n$ *si y sólo si* $a \bmod n = b \bmod n$
2. $a \equiv (a \bmod n) \bmod n$
3. *Si* $a \equiv b \bmod n$ *y* $c \equiv d \bmod n$, *entonces:*

$$\begin{aligned} (a + c) &\equiv (b + d) \bmod n \\ (a \cdot c) &\equiv (b \cdot d) \bmod n \end{aligned}$$

Ejercicios

1. Demuestre la proposición
2. Demuestre que un número n es divisible por 3 si y sólo si la suma de sus dígitos es divisible por 3

Algoritmos básicos en teoría de números

Vamos a estudiar tres algoritmos fundamentales en el área:

- Exponenciación rápida
- El algoritmo de Euclides para el cálculo del máximo común divisor
- El algoritmo de Euclides extendido y el cálculo del inverso modular

Exponenciación rápida: calculando $a^b \bmod n$

Utilizamos el siguiente algoritmo para calcular $a^b \bmod n$, el cual es llamado exponenciación rápida:

EXP$(a,\ b,\ n)$
    if $b = 1$ **then return** $a \bmod n$
    else if b es par **then**
        $val := \mathbf{EXP}(a, \frac{b}{2}, n)$
        return $(val \cdot val) \bmod n$
    else
        $val := \mathbf{EXP}(a, \frac{b-1}{2}, n)$
        return $(val \cdot val \cdot a) \bmod n$

La complejidad de **EXP**

Ejercicio

Considerando la multiplicación de enteros y el cálculo de la función $x \bmod y$ como las operaciones básicas a contar, demuestre que **EXP**(a, b, n) en el peor caso es $O(\log_2(b))$

Máximo común divisor

Sea MCD(a, b) el máximo común divisor de los números a y b

- ¿Cómo podemos calcular MCD(a, b)?

Proposición

Si $b > 0$, *entonces* $MCD(a, b) = MCD(b, a \bmod b)$

Demostración: Vamos a demostrar que un número c divide a a y b si y sólo si c divide a b y $a \bmod b$

- De esto se concluye que MCD$(a, b) =$ MCD$(b, a \bmod b)$

Máximo común divisor

Sabemos que $a = \alpha \cdot b + (a \bmod b)$

($\Rightarrow$) Suponga que $c|a$ y $c|b$.

Dado que $(a \bmod b) = a - \alpha \cdot b$, concluimos que $c|(a \bmod b)$.

($\Leftarrow$) Suponga que $c|b$ y $c|(a \bmod b)$.

Dado que $a = \alpha \cdot b + (a \bmod b)$, tenemos que $c|a$ $\square$

Cálculo de máximo común divisor

De lo anterior, concluimos la siguiente identidad para $a > 0$:

$$\text{MCD}(a, b) = \begin{cases} a & b = 0 \\ \text{MCD}(b, a \bmod b) & b > 0 \end{cases}$$

Usamos esta identidad para generar un algoritmo para calcular el máximo común divisor, el cual es conocido como Algoritmo de Euclides:

MCD(a, b)
    if $a = 0$ **and** $b = 0$ **then return** error
    else if $a = 0$ **then return** b
    else if $b = 0$ **then return** a
    else if $a \geq b$ **then return MCD**(b, $a \bmod b$)
    else return MCD(a, $b \bmod a$)

¿Cuál es la complejidad del algoritmo?

La complejidad del algoritmo

Lema

Si $a \geq b$ *y* $b > 0$, *entonces* $(a \bmod b) < \frac{a}{2}$

Demostración: Si $b > \frac{a}{2}$:

$$a \bmod b \;=\; a - b \;<\; a - \frac{a}{2} \;=\; \frac{a}{2}$$

La complejidad del algoritmo

Si $b < \frac{a}{2}$, entonces:

$$a \bmod b \ < \ b \ < \ \frac{a}{2}$$

Si $b = \frac{a}{2}$ (a debe ser par):

$$a \bmod b \ = \ 0 \ < \ b \ = \ \frac{a}{2}$$

□

La complejidad del algoritmo

Ejercicio

Suponga que la operación básica para el algoritmo **MCD** es el cálculo de la función $x \bmod y$. Muestre entonces que el algoritmo en el peor caso es $O(\log_2(\text{máx}\{a, b\}))$, suponiendo que la entrada es (a, b)

- Vale decir, **MCD** es de orden lineal en el tamaño de la entrada en el peor caso

Una noción importante: inverso modular

Definición

b es inverso de a en módulo n si $a \cdot b \equiv 1 \bmod n$

Ejemplo

37 es inverso de 13 en módulo 60

¿Todo número tiene inverso modular?

- No, 2 no tiene inverso en módulo 4

¿Bajo qué condiciones *a* tiene inverso en módulo *n*?

Una identidad útil

Identidad de Bézout

Para cada $a, b \in \mathbb{N}$ tales que $a \neq 0$ o $b \neq 0$, existen $s, t \in \mathbb{Z}$ tales que:

$$\text{MCD}(a, b) \quad = \quad s \cdot a + t \cdot b$$

Ejercicio

Demuestre la identidad de Bézout.

Existencia de inverso modular y la Identidad de Bézout

Teorema

a tiene inverso en módulo n si y sólo si $MCD(a, n) = 1$

Demostración: ($\Rightarrow$) Suponga que b es inverso de a en módulo n

- Entonces: $a \cdot b \equiv 1 \bmod n$

Se deduce que $a \cdot b = \alpha \cdot n + 1$, por lo que $1 = a \cdot b - \alpha \cdot n$

Concluimos que si $c|a$ y $c|n$, entonces $c|1$

- Por lo tanto c debe ser igual a 1, de lo que concluimos que $\text{MCD}(a, n) = 1$

Existencia de inverso modular y la Identidad de Bézout

($\Leftarrow$) Suponga que $\text{MCD}(a, n) = 1$

Por la identidad de Bézout existen $s, t \in \mathbb{Z}$ tales que:

$$1 \quad = \quad s \cdot n + t \cdot a$$

Por lo tanto: $a \cdot t \equiv 1 \bmod n$

- Concluimos que a tiene inverso en módulo n $\square$

¿Cómo podemos calcular el inverso modular?

Sabemos que **MCD** es un algoritmo eficiente para calcular el máximo común divisor entre dos números.

¡Pero este algoritmo puede hacer más!

- Puede ser extendido para calcular s y t tales que $\text{MCD}(a, b) = s \cdot a + t \cdot b$

Vamos a usar este algoritmo para calcular *inversos modulares*

El Algoritmo Extendido de Euclides

Suponga que $a \geq b$, y defina la siguiente sucesión:

$$\begin{aligned} r_0 &= a \\ r_1 &= b \\ r_{i+1} &= r_{i-1} \bmod r_i \qquad (i \geq 2) \end{aligned}$$

Calculamos esta sucesión hasta un número k tal que $r_k = 0$

- Tenemos que $\text{MCD}(a, b) = r_{k-1}$

El Algoritmo Extendido de Euclides

Al mismo tiempo calculamos sucesiones s_i, t_i tales que:

$$r_i = s_i \cdot a + t_i \cdot b$$

Tenemos que: $\text{MCD}(a, b) = r_{k-1} = s_{k-1} \cdot a + t_{k-1} \cdot b$

Sean:

$$\begin{aligned} s_0 &= 1 & t_0 &= 0 \\ s_1 &= 0 & t_1 &= 1 \end{aligned}$$

Se tiene que:

$$\begin{aligned} r_0 &= s_0 \cdot a + t_0 \cdot b \\ r_1 &= s_1 \cdot a + t_1 \cdot b \end{aligned}$$

El Algoritmo Extendido de Euclides

Dado que $r_{i-1} = \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot r_i + r_{i-1} \bmod r_i$, tenemos que:

$$r_{i-1} = \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot r_i + r_{i+1}$$

Por lo tanto:

$$s_{i-1} \cdot a + t_{i-1} \cdot b = \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot (s_i \cdot a + t_i \cdot b) + r_{i+1}$$

Concluimos que:

$$r_{i+1} = (s_{i-1} - \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot s_i) \cdot a + (t_{i-1} - \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot t_i) \cdot b$$

Definimos entonces:

$$s_{i+1} = s_{i-1} - \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot s_i$$

$$t_{i+1} = t_{i-1} - \lfloor \frac{r_{i-1}}{r_i} \rfloor \cdot t_i$$

El Algoritmo Extendido de Euclides

Ejemplo

Vamos a usar el algoritmo para $a = 60$ y $b = 13$

Inicialmente:

$$\begin{array}{lll} r_0 = 60 & s_0 = 1 & t_0 = 0 \\ r_1 = 13 & s_1 = 0 & t_1 = 1 \end{array}$$

Entonces tenemos que:

$$\begin{aligned} r_2 &= r_0 \bmod r_1 \\ s_2 &= s_0 - \lfloor \frac{r_0}{r_1} \rfloor \cdot s_1 \\ t_2 &= t_0 - \lfloor \frac{r_0}{r_1} \rfloor \cdot t_1 \end{aligned}$$

El Algoritmo Extendido de Euclides

Example (Continuación)

Por lo tanto:

$$r_2 = 8 \qquad s_2 = 1 \qquad t_2 = -4$$

Y el proceso continua:

$$\begin{array}{lll} r_3 = 5 & s_3 = -1 & t_3 = 5 \\ r_4 = 3 & s_4 = 2 & t_4 = -9 \\ r_5 = 2 & s_5 = -3 & t_5 = 14 \\ r_6 = 1 & s_6 = 5 & t_6 = -23 \\ r_7 = 0 & s_7 = -13 & t_7 = 60 \end{array}$$

Tenemos que: $1 = 5 \cdot 60 + (-23) \cdot 13$

El Algoritmo Extendido de Euclides y el inverso modular

Dados dos números naturales a y n, con $n \geq 2$, si el inverso de a en módulo n existe el siguiente algoritmo lo retorna, y en caso contrario indica que no existe.

Inverso$(a,\ n)$
    if MCD$(a, n) > 1$ **then** *no_existe_inverso*
    else
        $s_0 := 1$
        $t_0 := 0$
        $s_1 := 0$
        $t_1 := 1$
        $r_0 := n$
        $r_1 := a$

El Algoritmo Extendido de Euclides y el inverso modular

while $r_1 > 1$ **do**

$aux\_s := s_0 - \lfloor \frac{r_0}{r_1} \rfloor \cdot s_1$

$s_0 := s_1$

$s_1 := aux\_s$

$aux\_t := t_0 - \lfloor \frac{r_0}{r_1} \rfloor \cdot t_1$

$t_0 := t_1$

$t_1 := aux\_t$

$r_0 := r_1$

$r_1 := s_1 \cdot n + t_1 \cdot a$

return t_1

Un problema fundamental: verificación de primalidad

Vamos a ver un algoritmo aleatorizado para verificar si un número es primo.

- Este algoritmo es mucho más eficiente que los algoritmos sin componentes aleatorias para este problema

El ingrediente fundamental para el algoritmo es el uso de aritmética modular.

Un primer ingrediente

Teorema (Fermat)

Sea p un número primo. Si $a \in \{0, \ldots, p-1\}$, entonces $a^p \equiv a \bmod p$

Demostración: Por inducción en a

Para $a = 0$ y $a = 1$ se cumple trivialmente. Suponga que $a^p \equiv a \bmod p$ y $2 \leq (a+1) < p$

Sabemos que:

$$(a+1)^p = \sum_{k=0}^{p} \binom{p}{k} a^k$$

Un primer ingrediente

Por lo tanto:

$$(a+1)^p - (a+1) \quad = \quad (a^p - a) + \sum_{k=1}^{p-1} \binom{p}{k} a^k$$

Lema

Si $k \in \{1, \ldots, p-1\}$, *entonces* $p | \binom{p}{k}$

Demostración: Sabemos que:

$$\binom{p}{k} = \frac{p!}{k! \cdot (p-k)!} = \frac{p \cdot (p-1) \cdot \ldots \cdot (p-k+1)}{k!}$$

Como $k \in \{1, \ldots, p-1\}$ y p es un número primo:

$\frac{(p-1)\cdot\ldots\cdot(p-k+1)}{k!}$ es un número entero

Un primer ingrediente

Por lo tanto: $\binom{p}{k} = p \cdot \alpha$, donde α es un número entero

- Concluimos que $p | \binom{p}{k}$ □

Del lema concluimos que $p \left| \sum_{k=1}^{p-1} \binom{p}{k} a^k \right.$

Por lo tanto, dado que $p|(a^p - a)$ por hipótesis de inducción, tenemos que: $p|((a+1)^p - (a+1))$

- Concluimos que $(a+1)^p \equiv (a+1) \bmod p$ □

Un primer ingrediente

Corolario (Fermat)

Sea p un número primo. Si $a \in \{1, \ldots, p-1\}$, entonces $a^{p-1} \equiv 1 \, mod \, p$

Demostración: Por teorema anterior sabemos que

$$a^p \equiv a \bmod p$$

Por lo tanto: existe un número entero α tal que

$$a^p - a \quad = \quad \alpha \cdot p$$

Un primer ingrediente

Dado que $a|(a^p - a)$, se tiene que $a|(\alpha \cdot p)$

Por lo tanto, dado que $a \in \{1, \ldots, p-1\}$ y p es un número primo, se concluye que $a|\alpha$

Entonces: $(a^{p-1} - 1) = \frac{\alpha}{a} \cdot p$, donde $\frac{\alpha}{a}$ es un número entero.

- Concluimos que $a^{p-1} \equiv 1 \bmod p$ □

Test de primalidad: primera versión

El test de primalidad que vamos a estudiar está basado en estas propiedades ($n \geq 2$):

1. Si n es primo y $a \in \{1, \ldots, n-1\}$, entonces $a^{n-1} \equiv 1 \bmod n$
2. Si n es compuesto, entonces existe $a \in \{1, \ldots, n-1\}$ tal que $a^{n-1} \not\equiv 1 \bmod n$

Demostración de 2. Sea $a \in \{1, \ldots, n-1\}$ tal que $\text{MCD}(a, n) > 1$

- a no tiene inverso en módulo n

Concluimos que $a^{n-1} \not\equiv 1 \bmod a$

- Dado que a^{n-2} no puede ser inverso de a en módulo n □

Test de primalidad: primera versión

Para $n \geq 2$, sea:

$$\mathbb{Z}_n^* \quad = \quad \{a \in \{1, \ldots, n-1\} \mid \text{MCD}(a, n) = 1\}$$

Sabemos que para n compuesto: Si $a \in (\{1, \ldots, n-1\} \setminus \mathbb{Z}_n^*)$, entonces $a^{n-1} \not\equiv 1 \bmod n$

Test de primalidad depende de cuan grande es $\mathbb{Z}_n^*$

- Suponemos que $|\mathbb{Z}_n^*| \leq \lfloor \frac{n}{2} \rfloor$ para cada número compuesto $n \geq 2$

Test de primalidad: primera versión

En nuestros algoritmos consideramos $n \geq 2$

TestPrimalidad1(n)
    sea a un número elegido de manera uniforme desde $\{1, \ldots, n-1\}$
    if EXP$(a, n-1, n) \neq 1$
        then return COMPUESTO
    else
        return PRIMO

Algunas propiedades de **TestPrimalidad1**

Ejercicios

Recuerde que estamos suponiendo que $|\mathbb{Z}_n^*| \leq \lfloor \frac{n}{2} \rfloor$ para cada número compuesto $n \geq 2$

1. Demuestre que la probabilidad de error de **TestPrimalidad1** es menor o igual a $\frac{1}{2}$
2. Demuestre que **TestPrimalidad1** funcionan en tiempo polinomial.
   - Recuerde que el tiempo es medido en función del tamaño de la entrada, que en este caso es $\lfloor \log_2(n) \rfloor + 1$ si suponemos que la entrada está dada como una palabra sobre el alfabeto $\{0, 1\}$
3. De un algoritmo que reciba como parámetros a dos números enteros $n \geq 2$ y $k \geq 1$, y determina si n es un número primo con probabilidad de error menor o igual a $\left(\frac{1}{2}\right)^k$

Una solución al tercer ejercicio

TestPrimalidad2$(n,\ k)$
    sea $a_1, \ldots, a_k$ una secuencia de números elegidos de
                manera uniforme e independiente desde $\{1, \ldots, n-1\}$
    for $i := 1$ **to** k **do**
        if **EXP**$(a_i, n-1, n) \neq 1$
            then return COMPUESTO
    return PRIMO

¿Pero la probabilidad de error de **TestPrimalidad2** está bien acotada?

Supusimos que $|\mathbb{Z}_n^*| \leq \lfloor \frac{n}{2} \rfloor$ para cada número compuesto $n \geq 2$

- ¿Es esta suposición correcta?

Función de Euler: $\phi(1) = 0$ y $\phi(n) = |\mathbb{Z}_n^*|$ para $n \geq 2$

- Necesitamos acotar el valor de esta función

Una cota inferior para la función ϕ de Euler

Teorema

$$\phi(n) \quad \in \quad \Omega\left(\frac{n}{\log_2(\log_2(n))}\right)$$

Conclusión

Para cada número n, el conjunto $\mathbb{Z}_n^*$ tiene un número de elementos cercano a n

- No es cierto que $|\mathbb{Z}_n^*| \leq \lfloor \frac{n}{2} \rfloor$ para cada número compuesto $n \geq 2$
- No podemos basar nuestro test en los elementos del conjunto $(\{1, \ldots, n-1\} \setminus \mathbb{Z}_n^*)$

Test de primalidad: segunda versión

Una observación importante: si n es compuesto, entonces puede existir $a \in \mathbb{Z}_n^*$ tal que $a^{n-1} \not\equiv 1 \bmod n$

- Por ejemplo: $3^{15} \bmod 16 = 11$

En lugar de considerar $\mathbb{Z}_n^*$ en el test de primalidad, consideramos:

$$J_n \quad = \quad \{a \in \mathbb{Z}_n^* \mid a^{n-1} \equiv 1 \bmod n\}$$

Si demostramos que para cada número compuesto n se tiene que $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$, entonces tenemos un test de primalidad.

- Puesto que para p primo: $|J_p| = |\mathbb{Z}_p^*| = p - 1$

Test de primalidad: segunda versión

Recuerde que en nuestros algoritmos consideramos $n \geq 2$

TestPrimalidad3(n, k)
    sea $a_1, \ldots, a_k$ una secuencia de números elegidos de
                manera uniforme e independiente desde $\{1, \ldots, n-1\}$
    for $i := 1$ **to** k **do**
        if MCD$(a_i, n) > 1$ **then return** COMPUESTO
        else
            if EXP$(a_i, n-1, n) \neq 1$
                then return COMPUESTO
    return PRIMO

Algunas consideraciones sobre **TestPrimalidad3**

Ejercicio

1. Demuestre que **TestPrimalidad3** funcionan en tiempo polinomial.

2. Suponiendo que para cada número compuesto n se tiene que $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$, demuestre que la probabilidad de error de **TestPrimalidad3** es menor o igual a $\left(\frac{1}{2}\right)^k$

¿Qué enfoque podríamos usar para demostrar que $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$ para cada número n compuesto?

- Teoría de grupos también juega un papel fundamental en el desarrollo del test de primalidad

Teoría de grupos

Definición

Un conjunto G y una función (total) $\circ : G \times G \to G$ forman un grupo si:

1. *Para cada $a, b, c \in G$:* $(a \circ b) \circ c = a \circ (b \circ c)$
2. *Existe $e \in G$ tal que para cada $a \in G$:* $a \circ e = e \circ a = a$
3. *Para cada $a \in G$, existe $b \in G$:* $a \circ b = b \circ a = e$

Propiedades básicas

- Neutro es único: Si e_1 y e_2 satisfacen 2, entonces $e_1 = e_2$
- Inverso de cada elemento a es único: Si $a \circ b = b \circ a = e$ y $a \circ c = c \circ a = e$, entonces $b = c$

Teoría de grupos: algunos ejemplos

Ejercicios

Muestre que los siguientes son grupos:

1. $(\mathbb{Z}_n, +)$, donde $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ y $+$ es la suma en módulo n
2. $(\mathbb{Z}_n^*, \cdot)$, donde $\cdot$ es la multiplicación en módulo n
3. $(J_n, \cdot)$, donde $\cdot$ es la multiplicación en módulo n

Teoría de grupos: subgrupos

Definition

$(H, \circ)$ es un subgrupo de un grupo $(G, \circ)$, para $\emptyset \subsetneq H \subseteq G$, si $(H, \circ)$ es un grupo.

Ejercicio

Demuestre que $(J_n, \cdot)$ es un subgrupo de $(\mathbb{Z}_n^*, \cdot)$

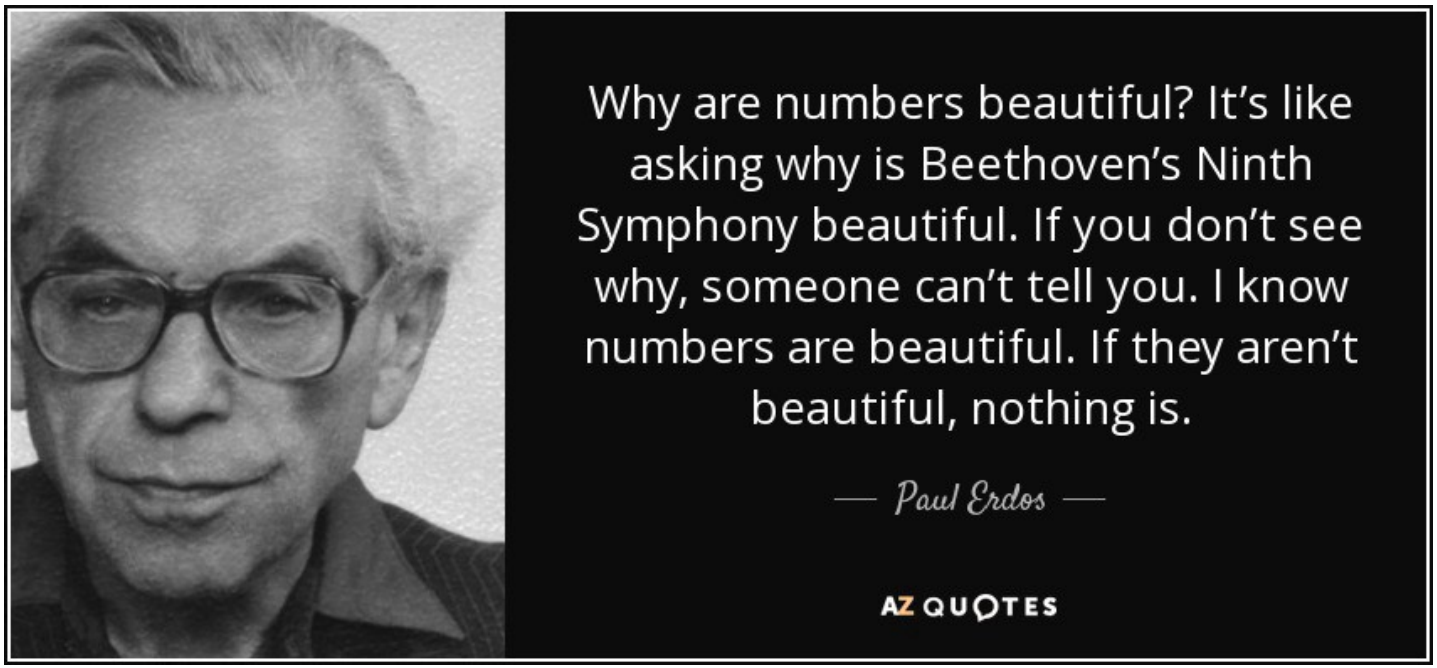
Propiedades básicas

- Si e_1 es el neutro en $(G, \circ)$ y e_2 es el neutro de $(H, \circ)$, entonces $e_1 = e_2$
- Para cada $a \in H$, si b es el inverso de a en $(G, \circ)$ y c es el inverso de a en $(H, \circ)$, entonces $c = b$

Teoría de grupos: una propiedad fundamental

Teorema (Lagrange)

Si $(G, \circ)$ *es un grupo finito y* $(H, \circ)$ *es un subgrupo de* $(G, \circ)$*, entonces* $|H|$ *divide a* $|G|$



Teorema de Lagrange: demostración

Suponga que e es el elemento neutro de $(G, \circ)$ y a^{-1} es el inverso de a en $(G, \circ)$

Sea $\sim$ una relación binaria sobre G definida como:

$$a \sim b \text{ si y sólo si } b \circ a^{-1} \in H$$

Lema

$\sim$ es una relación de equivalencia.

Teorema de Lagrange: demostración del primer lema

- $a \sim a$ ya que $a \circ a^{-1} = e$ y $e \in H$

- Suponga que $a \sim b$
  - Tenemos que demostrar que $b \sim a$

Dado que $a \sim b$: $b \circ a^{-1} \in H$

- Tenemos que demostrar que $a \circ b^{-1} \in H$

Tenemos que:

$$\begin{aligned}(b \circ a^{-1}) \circ (a \circ b^{-1}) &= (b \circ (a^{-1} \circ a)) \circ b^{-1} \\ &= (b \circ e) \circ b^{-1} \\ &= b \circ b^{-1} \\ &= e\end{aligned}$$

Teorema de Lagrange: demostración del primer lema

De la misma forma concluimos que $(a \circ b^{-1}) \circ (b \circ a^{-1}) = e$

- Por lo tanto: $(b \circ a^{-1})^{-1} = a \circ b^{-1}$

Concluimos que $a \circ b^{-1}$ está en H, ya que $(H, \circ)$ es un subgrupo de $(G, \circ)$

- Suponga que $a \sim b$ y $b \sim c$
  - Tenemos que demostrar que $a \sim c$

Por hipótesis: $b \circ a^{-1} \in H$ y $c \circ b^{-1} \in H$

- Tenemos que demostrar que $c \circ a^{-1} \in H$

Pero $(c \circ b^{-1}) \circ (b \circ a^{-1}) = c \circ a^{-1}$ y $\circ$ es cerrada en H

- Por lo tanto: $c \circ a^{-1} \in H$ □

Teorema de Lagrange: demostración

Sea $[a]_\sim$ la clase de equivalencia de $a \in G$ bajo la relación $\sim$

Lema

1. $[e]_\sim = H$
2. *Para cada* $a, b \in G$*:* $|[a]_\sim| = |[b]_\sim|$

Del lema se concluye el teorema.

- Puesto que las clases de equivalencia de $\sim$ particionan G

Teorema de Lagrange: demostración del segundo lema

1. Se tiene que:

$$\begin{aligned} a \in [e]_{\sim} &\Leftrightarrow e \sim a \\ &\Leftrightarrow a \circ e^{-1} \in H \\ &\Leftrightarrow a \circ e \in H \\ &\Leftrightarrow a \in H \end{aligned}$$

2. Sean $a, b \in G$, y defina la función f de la siguiente forma:

$$f(x) = x \circ (a^{-1} \circ b)$$

Teorema de Lagrange: demostración del segundo lema

Se tiene que:

$$
\begin{aligned}
x \in [a]_\sim \quad &\Rightarrow \quad a \sim x \\
&\Rightarrow \quad x \circ a^{-1} \in H \\
&\Rightarrow \quad (x \circ a^{-1}) \circ e \in H \\
&\Rightarrow \quad (x \circ a^{-1}) \circ (b \circ b^{-1}) \in H \\
&\Rightarrow \quad (x \circ (a^{-1} \circ b)) \circ b^{-1} \in H \\
&\Rightarrow \quad f(x) \circ b^{-1} \in H \\
&\Rightarrow \quad b \sim f(x) \\
&\Rightarrow \quad f(x) \in [b]_\sim
\end{aligned}
$$

Por lo tanto: $f : [a]_\sim \to [b]_\sim$

- Vamos a demostrar que f es una biyección, de lo cual concluimos que $|[a]_\sim| = |[b]_\sim|$

Teorema de Lagrange: demostración del segundo lema

f es 1-1:

$$
\begin{aligned}
f(x) = f(y) \quad &\Rightarrow \quad x \circ (a^{-1} \circ b) = y \circ (a^{-1} \circ b) \\
&\Rightarrow \quad (x \circ (a^{-1} \circ b)) \circ (b^{-1} \circ a) = \\
&\qquad\quad (y \circ (a^{-1} \circ b)) \circ (b^{-1} \circ a) \\
&\Rightarrow \quad x \circ (a^{-1} \circ (b \circ b^{-1}) \circ a) = \\
&\qquad\quad y \circ (a^{-1} \circ (b \circ b^{-1}) \circ a) \\
&\Rightarrow \quad x \circ ((a^{-1} \circ e) \circ a) = y \circ ((a^{-1} \circ e) \circ a) \\
&\Rightarrow \quad x \circ (a^{-1} \circ a) = y \circ (a^{-1} \circ a) \\
&\Rightarrow \quad x \circ e = y \circ e \\
&\Rightarrow \quad x = y
\end{aligned}
$$

Teorema de Lagrange: demostración del segundo lema

f es sobre:

$$\begin{aligned} y \in [b]_\sim &\Rightarrow b \sim y \\ &\Rightarrow y \circ b^{-1} \in H \\ &\Rightarrow (y \circ b^{-1}) \circ (a \circ a^{-1}) \in H \\ &\Rightarrow ((y \circ b^{-1}) \circ a) \circ a^{-1} \in H \\ &\Rightarrow a \sim ((y \circ b^{-1}) \circ a) \\ &\Rightarrow ((y \circ b^{-1}) \circ a) \in [a]_\sim \end{aligned}$$

Sea $x = ((y \circ b^{-1}) \circ a)$. Tenemos que:

$$\begin{aligned} f(x) &= x \circ (a^{-1} \circ b) \\ &= ((y \circ b^{-1}) \circ a) \circ (a^{-1} \circ b) \\ &= y \circ (b^{-1} \circ (a \circ a^{-1}) \circ b) \\ &= y \circ ((b^{-1} \circ e) \circ b) \\ &= y \circ (b^{-1} \circ b) \\ &= y \circ e \\ &= y \end{aligned}$$

Test de primalidad: segunda versión (continuación)

Pregunta pendiente: ¿Qué enfoque podríamos usar para demostrar que $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$?

- ¡Usamos el Teorema de Lagrange!

Dado que $(J_n, \cdot)$ es un subgrupo de $(\mathbb{Z}_n^*, \cdot)$:

Si existe $a \in (\mathbb{Z}_n^* \setminus J_n)$, entonces $|J_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$

¿Tenemos entonces nuestro test de primalidad?

- Lamentablemente no todavía: números de Carmichael
- Pero lo que hemos aprendido va a ser fundamental para desarrollar el test de primalidad

Test de primalidad: segunda versión (continuación)

Definition

Un número n es de Carmichael si $n \geq 2$, n es compuesto y $|J_n| = |\mathbb{Z}_n^*|$

Ejemplo

561, 1105 y 1729 son números de Carmichael.

Teorema (Alford-Granville-Pomerance)

Existe un número infinito de números de Carmichael.

Test de primalidad: tercera version

Conclusión: el test basado en J_n no va a funcionar.

¿Qué hacemos entonces?

- En lugar de utilizar J_n, vamos a usar las herramientas que desarrollamos sobre el siguiente conjunto (n impar):

$$S_n = \{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv 1 \bmod n \text{ ó } a^{\frac{n-1}{2}} \equiv -1 \bmod n\}$$

¡Esto sí funciona!