Algortimos en teoría de números

Parte II

Segundo semestre 2022

IIC2283

Prof. Nicolás Van Sint Jan

Recordatorio: Cálculo de máximo común divisor

Se cumple la siguiente identidad para $a > 0$:

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

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 ?

Recordatorio: Inverso modular

Definición

Para $n \in \mathbb{N}$ y $a, b \in \mathbb{Z} - \{0\}$. Decimos que b es el **inverso de a en módulo n** si

$$a \cdot b \equiv 1 \pmod{n}$$

Teorema

Sea $n \in \mathbb{N}$ y $a \in \mathbb{Z} - \{0\}$. Luego a tiene inverso en módulo n si y sólo si $\gcd(a, n) = 1$

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:

$$\gcd(a, b) \quad = \quad s \cdot a + t \cdot b$$

Outline

Algoritmo extendido de Euclides

Test de primalidad: primer intento

¿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

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

Vamos a usar este algoritmo para calcular **inversos modulares**

El Algoritmo Extendido de Euclides

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

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

y calculamos esta sucesión hasta un número k tal que $r_k = 0$.

¿ A qué corresponde el valor r_{k-1} ?

R: $r_{k-1} = \gcd(a, b)$

El Algoritmo Extendido de Euclides

Al mismo tiempo podemos ir calculando dos sucesiones s_i, t_i tales que:

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

Tenemos que:

$$\gcd(a, b) = r_{k-1} = s_{k-1} \cdot a + t_{k-1} \cdot b$$

Sean:

$$\begin{array}{ll} s_0 = 1 & \qquad t_0 = 0 \\ s_1 = 0 & \qquad t_1 = 1 \end{array}$$

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

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

Por lo tanto:

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

Concluimos que:

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

Definimos entonces:

$$\begin{aligned} s_{i+1} &= s_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot s_i \\ t_{i+1} &= t_{i-1} - \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor \cdot t_i \end{aligned}$$

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:

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

El Algoritmo Extendido de Euclides

Ejemplo

[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**

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

$s_0 := s_1$

$s_1 := aux\_s$

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

$t_0 := t_1$

$t_1 := aux\_t$

$r_0 := r_1$

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

return t_1

Outline

Algoritmo extendido de Euclides

Test de primalidad: primer intento

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

Pequeño Teorema de Fermat

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

$$a^p \equiv a \pmod{p}$$

Un primer ingrediente

Corolario

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

$$a^{p-1} \equiv 1 \pmod{p}$$

Demostración

Por teorema anterior sabemos que

$$a^p \equiv a \pmod{p}$$

Por lo tanto, existe $\alpha \in \mathbb{Z}$ tal que

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

Un primer ingrediente

Demostración

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

Por lo tanto, dado que $a \in \{1, \ldots, p-1\}$ y p es un número primo, se concluye que $a \mid \alpha$.
Entonces $(a^{p-1} - 1) = \frac{\alpha}{a} \cdot p$, donde $\frac{\alpha}{a}$ es un número entero.

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

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

$$a^{n-1} \equiv 1 \pmod{n}$$

2. Si n es compuesto, entonces **existe** $a \in \{1, \ldots, n-1\}$ tal que

$$a^{n-1} \not\equiv 1 \pmod{n}$$

Demostración

Suponga que n es compuesto. Sea $a \in \{1, \ldots, n-1\}$ tal que $\gcd(a, n) > 1$.

Luego a no tiene inverso en módulo n.

Concluimos que

$$a^{n-1} \not\equiv 1 \pmod{a}$$

dado que a^{n-2} no puede ser inverso de a en módulo n $\square$

Test de primalidad: primera versión

Para $n \geq 2$, defina el conjunto $\mathbb{Z}_n^*$ como:

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

Es decir, $\mathbb{Z}_n^*$ es el conjunto de todos los primos relativos de n que son menores que él.

Suponga que n es compuesto. Luego si $a \in \{1, \ldots, n-1\} - \mathbb{Z}_n^*$, entonces $a^{n-1} \not\equiv 1 \bmod n$.

Test de primalidad entonces depende de qué tan grande es $\mathbb{Z}_n^*$.

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

Test de primalidad: primera versión

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

¿Es esta suposición correcta?

Considere la **función de Euler** $\phi : \mathbb{N} \to \mathbb{N}$ definida como

$$\phi(n) = \begin{cases} 0 & n = 1 \\ |\mathbb{Z}_n^*| & n > 1 \end{cases}$$

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