Algortimos en teoría de números

Parte I

Segundo semestre 2022

IIC2283

Prof. Nicolás Van Sint Jan

Outline

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

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

Además, estos números α, β **son únicos**.

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

$$a \bmod b := \beta$$

Ejemplo

$8 \bmod 3 = 2 \qquad 9 \bmod 3 = 0 \qquad (-8) \bmod 3 = 1$

Para recordar: aritmética modular

Definición

$a \equiv b \pmod{n}$ si, y solo si, n divide a $(b - a)$

Usamos la notación $n \mid m$ para indicar que n divide a m

- $a \equiv b \pmod{n}$ si $n \mid (b - a)$

Para recordar: algunas propiedades básicas

Proposición

1. $a \equiv b \pmod{n}$ si y sólo si $a \bmod n = b \bmod n$
2. $a \equiv a \bmod n \pmod{n}$
3. Si $a \equiv b \pmod{n}$ y $c \equiv d \pmod{n}$, entonces:

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

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.

¿ Alguien recuerda alguno de estos **algoritmos** ?

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**
        $val :=$ **EXP**$(a, \frac{b}{2}, n)$
        return $(val \cdot val) \bmod n$
    else
        $val :=$ **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 $O(\log_2(b))$

Outline

Aritmética modular

Máximo común divisor

Inversos modulares

Máximo común divisor

Definición

Sea $a, b \in \mathbb{Z} - \{0\}$. Se define el **máximo común divisor** $\gcd(a, b)$ de a y b como el mayor número d tal que $d \mid a$ y $d \mid b$.

Ejemplos

$$\gcd(8, 12) = 4 \qquad \gcd(24, 36) = 12 \qquad \gcd(54, 24) = 6$$

En otras palabras, $\gcd(a, b)$ es el máximo del conjunto

$$D_{a,b} = \{c \in \mathbb{Z} \mid c \mid a \wedge c \mid b\}$$

¿ Cómo podemos calcular $\gcd(a, b)$?

Máximo común divisor

Proposición

Para todo $a, b \in \mathbb{Z} - \{0\}$, $\gcd(a, b) = \gcd(b, (a \bmod b))$

Demostración

Vamos a demostrar que para $c \in \mathbb{Z} - \{0\}$

$$c \mid a \text{ y } c \mid b \quad \Leftrightarrow \quad c \mid b \text{ y } c \mid (a \bmod b)$$

De esto se concluye que $\gcd(a, b) = \gcd(b, a \bmod b)$.

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

($\Rightarrow$) Suponga que $c \mid a$ y $c \mid b$.
Dado que $(a \bmod b) = a - \alpha \cdot b$, concluimos que $c \mid (a \bmod b)$.

($\Leftarrow$) Suponga que $c \mid b$ y $c \mid (a \bmod b)$.
Dado que $a = \alpha \cdot b + (a \bmod b)$, tenemos que $c \mid a$. $\square$

Cálculo de máximo común divisor

De lo anterior, concluimos 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 ?

La complejidad del algoritmo

Lema

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

Demostración

Si $b > \frac{a}{2}$:

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

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

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

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

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

$\square$

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 $O(\log_2(\max\{a, b\}))$, suponiendo que la entrada es (a, b)

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

Outline

Una noción importante: inverso modular

Definición

b es **inverso de a en módulo n** si

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

Ejemplo

37 es inverso de 13 en módulo 60

¿ Todo número tiene inverso modular ?

R: 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:

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

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 $\gcd(a, n) = 1$

Demostración

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

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

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

Concluimos que si $c \mid a$ y $c \mid n$, entonces $c \mid 1$. Por lo tanto c debe ser igual a 1, de lo que concluimos que $\gcd(a, n) = 1$

($\Leftarrow$) Suponga que $\gcd(a, n) = 1$. Por la identidad de Bézout existen $s, t \in \mathbb{Z}$ tales que:

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

Entonces $a \cdot t \equiv 1 \bmod n$. Así 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

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

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