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

$b \equiv c \; mod \, n \quad si \quad n \; divide \; a \; (c - b)$

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:*

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

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:*

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

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

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 $O(\log_2(b))$

Máximo común divisor

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

- ¿Cómo podemos calcular MCD(a, b)?

Máximo común divisor

Sea $\text{MCD}(a, b)$ el máximo común divisor de los números a y b

- ¿Cómo podemos calcular $\text{MCD}(a, b)$?

Proposición

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

Cálculo de máximo común divisor

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

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

Cálculo de máximo común divisor

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

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

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, $b \bmod a$)

Cálculo de máximo común divisor

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

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

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* $(a \bmod b) < \frac{a}{2}$

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):

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

$\square$

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

Una noción importante: inverso modular

Definición

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

Una noción importante: inverso modular

Definición

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

Ejemplo

37 es inverso de 13 en módulo 60

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?

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

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:

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

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:

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

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 $MCD(a, n) = 1$

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.

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

¿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 $\mathrm{MCD}(a, b) = s \cdot a + t \cdot b$

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 $\mathrm{MCD}(a, b) = r_{k-1}$

El Algoritmo Extendido de Euclides

Al mismo tiempo calculamos sucesiones s_i, t_i tales que:

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

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

El Algoritmo Extendido de Euclides

Al mismo tiempo calculamos sucesiones s_i, t_i tales que:

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

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

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

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

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:

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

Por lo tanto:

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

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:

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

Por lo tanto:

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

Concluimos que:

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

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:

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

Por lo tanto:

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

Concluimos que:

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

Definimos entonces:

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

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:

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

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:

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:

$$r_3 = 5 \qquad s_3 = -1 \qquad t_3 = 5$$

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:

$$\begin{array}{lll} r_3 = 5 & s_3 = -1 & t_3 = 5 \\ r_4 = 3 & s_4 = 2 & t_4 = -9 \end{array}$$

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:

$$\begin{array}{lll} r_3 = 5 & s_3 = -1 & t_3 = 5 \\ r_4 = 3 & s_4 = 2 & t_4 = -9 \\ r_5 = 2 & s_5 = -3 & t_5 = 14 \end{array}$$

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:

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

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:

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

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**

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

$\quad s_0 := s_1$

$\quad s_1 := aux\_s$

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

$\quad t_0 := t_1$

$\quad t_1 := aux\_t$

$\quad r_0 := r_1$

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

return t_1