Práctica 1
Aritmética modular

Índice

Para entregar

- Carpeta "aritmod" con
  - El código de las funciones *euclides*(), *euclidesext*(), *invmod*(), *potmod*() y *tchino*() completado.
  - Problemas de la sección 3 resueltos.

Nota: Después de cada algoritmo hay propuestos ejercicios con sus respectivas soluciones. Servirán para comprobar que los programas funcionan correctamente, antes de realizar la entrega.

1. Introducción

Dedicaremos esta práctica a programar algunas operaciones frecuentes en teoría de números y a efectuar algunas operaciones modulares. Los enteros involucrados en Criptografía son números muy grandes, de ahí que ciertas operaciones puedan producir fácilmente *overflow* (por ejemplo, la potenciación modular). Hemos elegido algoritmos sencillos que tratan de evitar este problema.

2. Algoritmos

2.1. Algoritmo de Euclides

Dados dos números enteros a y b, el algoritmo de Euclides permite calcular el valor d del máximo común divisor de a y b.

Se comienza dividiendo a entre b y después se divide sucesivamente cada divisor por el resto:

$$\begin{array}{rc}
a = q_1 b + r_1, & 0 < r_1 < b \\
b = q_2 r_1 + r_2, & 0 < r_2 < r_1 \\
r_1 = q_3 r_2 + r_3, & 0 < r_3 < r_2 \\
\vdots & \vdots \\
r_i = q_{i+2} r_{i+1} + r_{i+2}, & 0 < r_{i+2} < r_{i+1} \\
\vdots & \vdots \\
r_{k-2} = q_k r_{k-1} + r_k, & 0 < r_k < r_{k-1} \\
r_{k-1} = q_{k+1} r_k + 0 &
\end{array}$$

Como cada vez se obtienen restos más pequeños, en algún momento el resto es 0.

$$b > r_1 > r_2 > \cdots > r_{k-1} > r_k > 0 \; (= r_{k+1}).$$

Veamos que $\mathrm{mcd}(a, b) = \mathrm{mcd}(b, r_1)$. Sea $d = \mathrm{mcd}(a, b)$.

- d es un *divisor común* de b y r_1:

  $$d = \mathrm{mcd}(a, b) \Rightarrow d \mid a \text{ y } d \mid b.$$

  Como $r_1 = a - q_1 b$, se tiene que $d \mid r_1$. Por tanto,

  $$d \mid b \text{ y } d \mid r_1.$$

- d es el *máximo* común divisor de b y r_1, es decir, cualquier divisor común de b y r_1 divide a d:

  Sea c un divisor común de b y r_1. Entonces $c \mid b$ y $d \mid r_1$.

  Como $a = q_1 b + r_1$ se tiene que $c \mid a$. Luego c es un divisor común a y b. Pero cualquier divisor común de a y b divide a $\mathrm{mcd}(a, b) = d$. Por tanto,

  $$c \mid d.$$

De la misma forma se prueba que

$$\mathrm{mcd}(b, r_1) = \mathrm{mcd}(r_1, r_2), \quad \mathrm{mcd}(r_1, r_2) = \mathrm{mcd}(r_2, r_3), \quad \ldots.$$

Es decir,

$$\text{mcd}(a,b) = \text{mcd}(b,r_1) = \text{mcd}(r_1,r_2) = \cdots = \text{mcd}(r_{k-1},r_k) = \text{mcd}(r_k,0) = r_k.$$

Como consecuencia, el último resto distinto de 0 es el $\text{mcd}(a,b)$.

Algoritmo

Entrada: Números enteros no negativos a y b.

Salida: Un número entero $d = \text{mcd}(a,b)$.

Paso 1. Mientras $b \neq 0$

Paso 1.1. Hacer $r = a \mod b$.

Paso 1.2. Hacer $a = b$.

Paso 1.3. Hacer $b = r$.

Paso 2. Salida a.

- Programar el algoritmo anterior (*euclides*()).

EJERCICIOS.

1. Probar la función con ejemplos sencillos: $\text{mcd}(3,5)$, $\text{mcd}(2,4)$, ...

2. Calcular: $\text{mcd}(2730,2926)$; $\text{mcd}(5510,8246)$; $\text{mcd}(1547,560)$.
   Solución: 14; 38; 7.

2.2. Algoritmo de Euclides extendido

Dados dos números enteros a y b, el algoritmo de Euclides extendido proporciona dos números enteros u y v tales que $au + bv = d$, donde $d = \text{mcd}(a,b)$.

Tendremos en cuenta que cada uno de los restos obtenidos en el algoritmo de Euclides para el cálculo de $\text{mcd}(a,b)$ se puede expresar como una combinación lineal de a y b:

$$a = q_1 b + r_1 \Rightarrow r_1 = a - q_1 b = 1 \cdot a + (-q_1)b = u_1 a + v_1 b,$$

donde $u_1 = 1, \quad v_1 = -q_1$,

$$b = q_2 r_1 + r_2 \Rightarrow r_2 = b - q_2 r_1 = (-q_2)u_1 a + (1 - q_2 v_1)b = u_2 a + v_2 b,$$

donde $u_2 = -q_2, \quad v_2 = 1 - q_2 v_1,$

$$r_1 = q_3 r_2 + r_3 \Rightarrow r_3 = r_1 - q_3 r_2 = (u_1 - q_3 u_2)a + (v_1 - q_3 v_2)b = u_3 a + v_3 b,$$

donde $u_3 = u_1 - q_3 u_2, \quad v_3 = v_1 - q_3 v_2,$

$$r_2 = q_4 r_3 + r_4 \Rightarrow r_4 = r_2 - q_4 r_3 = (u_2 - q_4 u_3)a + (v_2 - q_4 v_3)b = u_4 a + v_4 b,$$

donde $u_4 = u_2 - q_4 u_3, \quad v_4 = v_2 - q_4 v_3.$

En general, si denotamos $r_{-1} = a$ y $r_0 = b$, se tiene

$$r_i = u_i a + v_i b,$$

donde

$$u_{-1} = 1, \quad v_{-1} = 0, \quad u_0 = 0, \quad v_0 = 1$$

y, para $i \geq 1$,

$$u_i = u_{i-2} - q_i u_{i-1}, \quad v_i = v_{i-2} - q_i v_{i-1}.$$

Denotando ahora

$$t_i = u_{i-2}, \quad s_i = u_{i-1}, \quad g_i = v_{i-2}, \quad h_i = v_{i-1},$$

se tiene:

$$t_1 = 1, \quad s_1 = 0, \quad g_1 = 0, \quad h_1 = 1,$$

$$u_i = t_i - q_i s_i, \quad s_{i+1} = u_i, \quad t_{i+1} = s_i,$$

$$v_i = g_i - q_i h_i, \quad g_{i+1} = h_i, \quad h_{i+1} = v_i.$$

Algoritmo

Entrada: Números enteros no negativos a y b.

Salida: Números enteros d, u, v tales que $d = \text{mcd}(a, b)$ y $d = au + bv$.

Paso 1. Hacer $(t, s, g, h) = (1, 0, 0, 1)$

Paso 2. Mientras $b > 0$

  Paso 2.1. Hacer $q = \text{ent}(a/b)$; $r = a - qb$; $u = t - qs$; $v = g - qh$

  Paso 2.2. Hacer $a = b$; $b = r$; $t = s$; $s = u$; $g = h$; $h = v$

Paso 3. Salida (a, t, g)

- Programar el algoritmo anterior ($euclidesext()$).

  Observación: Por ent(a/b) denotamos la parte entera del cociente de la división de a entre b. La sentencia $a\,\%/\,\%b$ permite obtenerla.

EJERCICIOS.

1. Probar la función con ejemplos sencillos: $\text{mcd}(3,5) = 1 = 2 \cdot 3 + (-1) \cdot 5$, $\text{mcd}(2,4) = 2 = 1 \cdot 2 + 0 \cdot 4$, ...

2. Calcular $d = \text{mcd}(5510, 8246)$ y obtener u, v tales que $d = 5510u + 8246v$.
   Solución: $d = 38$, $u = 3$, $v = -2$.

3. Calcular $d = \text{mcd}(1547, 560)$ y obtener u, v tales que $d = 1547u + 560v$.
   Solución: $d = 7$, $u = 21$, $v = -58$.

2.3. Inversos módulo n

Ahora se trata de implementar el cálculo del inverso modular.

Si queremos calcular $a^{-1} \mod n$ usaremos el algoritmo de Euclides extendido programado anteriormente.

Recordemos que la salida de $euclidesext(a, n)$ es un vector de enteros (d, u, v), donde $d = \text{mcd}(a, n)$ y $d = ua + vn$.

Si existe $a^{-1} \mod n$, se tendrá que $d = 1$, luego $1 = ua + vn$ y por tanto

$$ua \equiv 1 \mod n,$$

de donde

$$a^{-1} \equiv u \mod n.$$

Entonces, el programa para el cálculo del inverso modular consistirá simplemente en devolver como salida el valor de u, es decir, la segunda componente del vector $euclidesext(a, n)$ (mód n):

```
source("aritmod/euclidesext.R")

invmod <- function(a,n)
  # Entrada: a, n (enteros positivos, relativamente primos)
  # Salida:  a^{-1} (mod n)
  {
    #Chequeos
    if(a<=0|n<=0){stop("a,n deben ser positivos")}
    x<-euclidesext(a,n)
    if(x[1]!=1){stop("los números no son primos relativos")}

    #Código
    inv <- x[2]
    inv <- inv%%n
    return(inv)
  }
```

- Programar el algoritmo anterior (*invmod*()).
  Observación: La orden

  ```
  inv <- inv%%n
  ```

  tiene como fin que el algoritmo devuelva un valor entre 0 y $n-1$.

EJERCICIOS.

1. Probar la función con ejemplos sencillos: $3^{-1} \mod 5$, $3^{-1} \mod 4$, ...

2. Calcular: $1426^{-1} \mod 1653$; $7084^{-1} \mod 87$.
   Solución: 1369; 40.

2.4. Potenciación modular

Dados números enteros a, x, n con $x \geq 0$, $n > 1$, vamos a calcular la potencia modular $a^x \mod n$ por el método denominado potenciación por cuadrados, o "*repeated squaring method*", teniendo en cuenta que

$$a^{2^i} = (a^{2^{i-1}})^2.$$

Para calcular a^x, utilizaremos la representación binaria de x. Por ejemplo,

$$x = 27 = 11011_{(2} = 2^4 + 2^3 + 2 + 1.$$

$$a^{27} = a^{2^4+2^3+2+1} = a^{2^4} \cdot a^{2^3} \cdot a^2 \cdot a = (((a^2 \cdot a)^2)^2 \cdot a)^2 \cdot a.$$

En general,

$$x = 2^{l-1}b_{l-1} + ... + 2b_1 + b_0.$$

$$\begin{aligned}
a^x &= a^{2^{l-1}b_{l-1}+2^{l-2}b_{l-2}+2^{l-3}b_{l-3}+...+2b_2+2b_1+b_0} \\
&= (\underbrace{a^{b_{l-1}}}_{Z_1})^{2^{l-1}} \cdot (a^{b_{l-2}})^{2^{l-2}} \cdot (a^{b_{l-3}})^{2^{l-3}} \cdot ... \cdot (a^{b_2})^{2^2} \cdot (a^{b_1})^2 \cdot a^{b_0} \\
&= (\underbrace{(a^{b_{l-1}})^2 \cdot a^{b_{l-2}}}_{Z_2})^{2^{l-2}} \cdot (a^{b_{l-3}})^{2^{l-3}} \cdot ... \cdot (a^{b_2})^{2^2} \cdot (a^{b_1})^2 \cdot a^{b_0} \\
&= (\underbrace{((a^{b_{l-1}})^2 \cdot a^{b_{l-2}})^2 \cdot a^{b_{l-3}}}_{Z_3})^{2^{l-3}} \cdot (a^{b_{l-4}})^{2^{l-4}} \cdot ... \cdot (a^{b_2})^{2^2} \cdot (a^{b_1})^2 \cdot a^{b_0} \\
&\vdots \\
&= (\underbrace{((\ldots((a^{b_{l-1}})^2 \cdot a^{b_{l-2}})^2 \cdot a^{b_{l-3}})^2 \cdot a^{b_{l-4}} ... \, a^{b_2})^2 a^{b_1}}_{Z_{l-1}})^2 a^{b_0}
\end{aligned}$$

Se puede dar una fórmula recurrente:

$$Z_0 = 1,$$

$$Z_k = Z_{k-1}^2 a^{b_{l-k}}, \quad k = 1, 2, \ldots, l$$

Entonces,

$$a^x = Z_l.$$

Observaciones:

1. Es de notar que los coeficientes b_{l-k} sólo pueden valer 0 o 1. Si $b_{l-k} = 0$, entonces $Z_k = Z_{k-1}^2$ y si $b_{l-k} = 1$, entonces $Z_k = Z_{k-1}^2 a$.
2. Para que el algoritmo permita operar con números más grandes, tras efectuar cada producto intermedio debemos hacer la reducción modular.

Algoritmo

Entrada: Enteros a, x, n ($x \geq 0$, $n > 1$).

Salida: $z = a^x \mod n$.

Paso 0. Representar x en forma binaria: $x = 2^{l-1}b_{l-1} + ... + 2b_1 + b_0$.
$(xbin = \left(xbin[1] \quad xbin[2] \quad \ldots xbin[l]\right), \quad xbin[i] = b_{l-i}, \ i = 1, 2, \ldots, l).$

Paso 1. Hacer $z = 1$.

Paso 2. Para $i = 1, ..., l$

Paso 2.1. Hacer $z = z^2 \mod n$.

Paso 2.2. Si $xbin[i] = 1$ hacer $z = za \mod n$.

Paso 3. Salida z.

- Programar el algoritmo anterior (*potmod*()).

  Observación: Para representar x en forma binaria podemos utilizar la función *cambiobase*().

EJERCICIOS.

1. Probar la función con ejemplos sencillos: $3^2 \mod 5$, $2^3 \mod 7$, $6^5 \mod 6$, $5^0 \mod 8$, $\ldots$.

2. Calcular: $38^{75} \mod 103$; $2^{1000000} \mod 7$.
   Solución: 79; 2.

2.5. Teorema chino del resto

El Teorema chino del resto permite resolver ciertos sistemas de ecuaciones modulares. Tiene numerosas aplicaciones en Criptografía, por ejemplo en el algoritmo RSA y en procedimientos criptográficos para compartir secretos.

Teorema (T. chino del resto)
Sean $p_1, p_2, \ldots, p_r$ primos dos a dos y $a_1, a_2, \ldots, a_r$ números enteros. Entonces el sistema de ecuaciones modulares

$$\begin{array}{rcl} x & \equiv & a_1 \mod p_1, \\ x & \equiv & a_2 \mod p_2, \\ & \vdots & \\ x & \equiv & a_r \mod p_r, \end{array}$$

posee una única solución módulo $n = p_1 \ldots p_r$, que viene dada por

$$x \equiv (\sum_{i=1}^{r} \frac{n}{p_i} y_i a_i) \mod n,$$

siendo $y_i \equiv (\frac{n}{p_i})^{-1} \mod p_i, \ i = 1, \ldots, r$.

La unicidad módulo n significa que existe una única solución en $\mathbb{Z}_n$, y que si x es una solución entonces también lo es $x + kn$ para cuaquier entero k.

Algoritmo

Entrada: $a = (a_1, \ldots, a_r)$ (vector de enteros)
$p = (p_1, \ldots, p_r)$ (vector de enteros primos dos a dos)

Salida: (x, n), donde
x es un entero tal que $x \equiv a_i \mod p_i$, $i = 1, \ldots, r$
$n = p_1 \ldots p_r$.

Paso 0. Hacer $n = p_1 \ldots p_r$.

Paso 1. Hacer $x = 0$.

Paso 2. Para $i = 1, ..., r$

Paso 2.1. Hacer $y \equiv (\frac{n}{p_i})^{-1} \mod p_i$.

Paso 2.2. Hacer $x \equiv (x + (\frac{n}{p_i}) y a_i) \mod n$.

Paso 3. Salida (x, n).

- Programar el algoritmo anterior (*tchino*()).

EJERCICIOS.

1. Probar la función con ejemplos sencillos, como

$$\left.\begin{array}{l} x \equiv 4 \mod 7 \\ x \equiv 2 \mod 3 \end{array}\right\}$$

2. Resolver:

$$\left.\begin{array}{l} x \equiv 87 \mod 127, \\ x \equiv 91 \mod 255. \end{array}\right\}$$

Solución: $x \equiv 31456 \mod 32385$.

3. Resolver:

$$\left.\begin{array}{l} x \equiv 67 \mod 92 \\ x \equiv 42 \mod 87 \\ x \equiv 24 \mod 77 \end{array}\right\}$$

Solución: $551883 \mod 616308$.

3. Problemas

Para resolver algunos de los problemas, además de utilizar las funciones anteriores, habrá que hacer algún razonamiento. Hay que escribir la solución en el fichero "problemasaritmod.R", escribiendo comentados todos los pasos necesarios para la resolución del problema. A modo de ejemplo, el primer problema está resuelto.

1. *a*) Demostrar que existe $207^{-1} \pmod{1000}$.

   b) Resolver la ecuación: $207x = 10 \pmod{1000}$.

2. *a*) Demostrar que existe $357^{-1} \pmod{1210}$.

   b) Resolver la ecuación: $357x = 2 \pmod{1210}$.

   c) Calcular $357^{10} \pmod{1210}$ y $357^{-10} \pmod{1210}$.

3. *a*) Demostrar que $p = 1925$, $q = 1728$ son primos relativos.

   b) Resolver:

$$\left.\begin{array}{l} x \equiv 48 \pmod{1925} \\ x \equiv 148 \pmod{1728} \end{array}\right\}$$