Práctica 5
Primalidad. Factorización

Índice

Para entregar

- Carpeta "primfact" con el código de las funciones $milrab()$, $pollard()$, $fermat()$ completado.

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

En Criptografía de clave pública se utilizan números primos grandes. Asegurar de manera determinista que uno de tales números es primo supone comprobar que no es divisible por ninguno de los primos $2, 3, 5, 7, \ldots$ hasta el mayor entero primo menor o igual que $[\sqrt{n}]$ ($[x]$ es la parte entera de x), lo cual es computacionalmente imposible. Se impone así la necesidad de obtener números primos de manera probabilística. Para ello se desarrollan los llamados *Tests de primalidad*. Nos permiten concluir que un número entero es primo con una cierta probabilidad, que puede ser tan grande como queramos.

Por otra parte, el problema de la factorización de enteros es uno de los problemas de matemáticas que se considera *intratable*, lo cual significa que no se conoce ningún algoritmo que permita factorizar un entero en tiempo polinomial. En realidad, su complejidad computacional se desconoce.

El **problema de la factorización de enteros** consiste en lo siguiente:

Dado un entero positivo n, hallar su descomposición en factores primos

$$n = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r},$$

donde $p_1, p_2, \dots, p_r$ son números primos distintos y $e_i \geq 1$, $i = 1, \dots, r$.

Un sencillo algoritmo de factorización consiste de nuevo en probar si el número dado n es divisible por alguno de los primos menores o iguales que $[\sqrt{n}]$, pero para enteros grandes este cálculo no se puede llevar a cabo.

El hecho de que en la actualidad se considere que la factorización de enteros es un problema intratable permite desarrollar en base a él un criptosistema de clave pública: el RSA. El criptosistema se basa en el siguiente problema que es un caso particular del problema de factorización de números enteros.

El problema RSA:

Dado un entero positivo n del que se sabe que es producto de dos números primos p y q, hallar sus factores.

2. Test de primalidad de Miller-Rabin

El test de primalidad de Miller-Rabin se basa en el siguiente resultado:

Teorema 1 *Sea n un número entero impar y sea $n - 1 = 2^s t$ con t impar.*

1. *Si n es primo, entonces para cualquier entero a tal que $\mathrm{mcd}(a, n) = 1$ se cumple que*

$$\left.\begin{array}{l} a^t \equiv 1 \mod n \\ o \\ a^{2^j t} \equiv n - 1 \mod n \text{ para algún } j,\ 0 \leq j < s. \end{array}\right\} \quad (1)$$

2. *Si n es compuesto, entonces la condición (1) se satisface como máximo para $\frac{1}{4}$ de los enteros a tales que $0 < a < n$. (De hecho, si $n \neq 9$, el número máximo es $\frac{\phi(n)}{4}$).*

Test:

Sea n un número entero impar y sea $n - 1 = 2^s t$ con t impar.

Se elige un entero a, $1 < a < n - 1$.

1. Si $a^t \not\equiv 1 \mod n$ y $a^{2^j t} \not\equiv n - 1 \mod n$, para todo j, $0 \leq j < s$, entonces n es compuesto.

2. Si $a^t \equiv 1 \mod n$ ó $a^{2^j t} \equiv n-1 \mod n$, para algún $j,\ 0 \leq j < s$, entonces $p(n \text{ es primo}) > \frac{3}{4}$.

Si se cumple la segunda condición, se tiene que $p(n \text{ es compuesto}) < \frac{1}{4}$. Así, si pasamos el test dos veces (para distintos valores de a), obtendremos que $p(n \text{ es compuesto}) < \frac{1}{4^2}$. Si pasamos el test k veces, $p(n \text{ es compuesto}) < 4^{-k}$ y por tanto $p(n \text{ es primo}) > 1 - 4^{-k}$.

Observaciones:

1. Notemos que

$$a^{2t} = (a^t)^2, \quad a^{2^2 t} = (a^{2t})^2, \quad a^{2^3 t} = (a^{2^2 t})^2, \quad \ldots.$$

   Es decir, si $y_j = a^{2^j t}$, entonces

$$y_{j+1} = y_j^2, \quad j = 0, 1, \ldots.$$

2. Con esta notación, si $y_j = 1$ entonces $y_k = 1 \neq n-1$ para $k \geq j$.

Algoritmo

Entrada: n (número entero impar $n \geq 3$),
k (parámetro de seguridad $k \geq 1$).

Salida: "n es compuesto" o "n es primo con probabilidad mayor que $1 - 4^{-k}$".

Paso 1. Calcular enteros (t, s) tales que $n - 1 = 2^s t,\ t$ impar.

Paso 2. Hacer $fin = FALSE$.

Paso 3. Hacer *contador* $= 0$.

Paso 4. Mientras *contador* $< k$ y fin sea $FALSE$

Paso 4.1. Elegir entero aleatorio $a,\ 1 < a < n-1$.

Paso 4.2. Calcular $y \equiv a^t \mod n$.

Paso 4.3. Si $y \neq 1$ e $y \neq n-1$

Paso 4.3.1. Hacer $j = 1$.

Paso 4.3.2. Mientras $j < s$ e $y \neq n-1$ y fin sea $FALSE$

Paso 4.3.2.1. Hacer $y \equiv y^2 \mod n$.

Paso 4.3.2.2. Si $y = 1$

Paso 4.3.2.2.1. Hacer $fin = TRUE$.

Paso 4.3.2.3. Hacer $j = j + 1$.

Paso 4.3.3. Si $y \neq n - 1$

Paso 4.3.3.1. Hacer $fin = TRUE$.

Paso 4.4. Hacer $contador = contador + 1$.

Paso 5. Si fin es $TRUE$

Paso 5.1. Hacer $mensaje$="n es compuesto".

Paso 6. En otro caso

Paso 6.1. Hacer $mensaje$="n es primo con probabilidad mayor que $1-4^{-k}$".

Paso 7. Salida $mensaje$.

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

  Observaciones:

  1. Se sugiere el uso del siguiente bloque de código para implementar la factorización $n - 1 = 2^s t$:

     ```
     s <- 0
     t <- n-1
     while(t%%2==0)
       {
         s <- s+1
         t <- t/2
       }
     ```

  2. Para elegir un entero aleatorio a, $1 < a < n - 1$ podemos usar la orden

     ```
     a <- sample(2:(n-2), 1)
     ```

  3. En el paso 2.2 será necesario utilizar la función $potmod()$.

EJERCICIOS.

1. Probar la función con ejemplos sencillos: 13, 25, ...

2. Estudiar si 41353 y 38737 son primos.
   Solución: 41353 es compuesto y 38737 es primo.

3. Algoritmos de factorización

Existen numerosos algoritmos que tratan de encontrar los factores de un número entero compuesto para ciertos tipos de enteros. Vamos a ver dos de ellos que tratan de resolver el problema RSA. Son:

1. El método rho de Pollard.
2. El método de factorización de Fermat.

3.1. Método rho de Pollard

Este método permite encontrar un factor pequeño de un entero compuesto. Consiste en lo siguiente:

- Se elige un polinomio $p(x)$ con coeficientes en $\mathbb{Z}_n$.
- Se elige un entero x_0 (por ejemplo $x_0 = 1$ ó $x_0 = 2$).
- Se calcula la sucesión $x_0, x_1, \ldots$, con

$$x_i = p(x_{i-1}) \mod n, \quad i = 1, \ldots.$$

- Se hacen comparaciones entre los valores calculados hasta encontrar $x_j,\ x_k$ tales que $x_k \not\equiv x_j \mod n$ y $x_k \equiv x_j \mod d$ para algún divisor propio d de n. Es decir, tales que

$$n \not| x_k - x_j, \quad d \mid x_k - x_j, \quad d \mid n.$$

- Una vez obtenidos los números $x_j,\ x_k$ cumpliendo la anterior condición, se tiene que $\mathrm{mcd}(x_k - x_j, n)$ es un divisor propio de n.

Observaciones:

1. Los polinomios más comunes utilizados son de la forma $p(x) = x^2 + c$, con $c \neq 0, -2$.
2. Por si el método fracasa, es conveniente limitar por entrada el número de pasos a efectuar. En caso de que no proporcione resultado, antes de abandonar se debe probar con otra semilla u otro polinomio.

Algoritmo

Entrada: n (número entero impar $n \geq 3$),
p (vector de coeficientes de un polinomio sobre $\mathbb{Z}_n$, con el coeficiente de grado más alto a la izquierda),
x_0 (semilla),
N (número máximo de pasos).

Salida: (f_1, f_2) (factores de n).

Paso 1. Hacer $factores =$ "No se ha encontrado la factorización".

Paso 2. Hacer $b = x_0$.

Paso 3. Hacer $contador = 0$.

Paso 4. Hacer $factoresencontrados = FALSE$.

Paso 5. Mientras $contador < N$ y $factoresencontrados$ sea $FALSE$

Paso 5.1. Calcular $a \equiv p(x_0) \mod n$.

Paso 5.2. Hacer $b = \text{concatenar}(b, a)$.

Paso 5.3. Hacer $j = 1$.

Paso 5.4. Mientras $j <$ longitud(b) y $factoresencontrados$ sea $FALSE$

Paso 5.4.1. Calcular $d = \text{mcd}(a - b[j], n)$.

Paso 5.4.2. Si $1 < d < n$

Paso 5.4.2.1. Hacer $factores = (d, \frac{n}{d})$.

Paso 5.4.2.2. Hacer $factoresencontrados = TRUE$.

Paso 5.4.3. Hacer $j = j + 1$.

Paso 5.5. Si $factoresencontrados$ es $FALSE$.

Paso 5.5.1. Hacer $x0 = a$.

Paso 5.5.2. Hacer $contador = contador + 1$.

Paso 6. Salida $factores$.

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

  Observación: El algoritmo necesitará hacer uso de las funciones $evalpol()$ para evaluar un polinomio y $euclides()$ para calcular el máximo común divisor de dos enteros. Si hemos programado el algoritmo de Euclides de forma que la entrada sean números no negativos, al calcular $\text{mcd}(a - b(j), n)$, deberemos tener en cuenta que $a - b(j)$ puede ser negativo.

EJERCICIOS.

1. Probar la función con ejemplos sencillos: 21, 35, ...

2. Factorizar $n_1 = 455459$, $n_2 = 4087$.
   Solución: $455459 = 743 \cdot 613$; $4087 = 67 \cdot 61$.

3.2. Método de factorización de Fermat

Se basa en el siguiente resultado:

Teorema 2 *Sea n entero positivo impar. Entonces existe una correspondencia uno a uno entre una factorización $n = ab$, donde $a \geq b > 0$ y una representación de n de la forma $n = t^2 - s^2$ donde s y t son enteros no negativos. La correspondencia viene dada por*

$$t = (a+b)/2, \quad s = (a-b)/2; \qquad a = t+s, \quad b = t-s.$$

Si $n = ab$, con a, b próximos, entonces $s = (a-b)/2$ pequeño y t poco mayor que $\sqrt{n}$. En ese caso, podemos obtener a y b probando con todos los valores de t empezando con $t = [\sqrt{n}] + 1$, hasta que encontremos uno para el que $t^2 - n = s^2$ sea un cuadrado perfecto.

Observación: Por si el método fracasa, es conveniente limitar por entrada el número de pasos a efectuar.

Algoritmo

Entrada: n (número entero impar $n > 0$).
N (número máximo de pasos).

Salida: (f_1, f_2) (factores de n).

Paso 1. Hacer $t = \lceil \sqrt{n} \rceil$ ($\lceil \cdot \rceil$ significa *techo*: el techo de x es mínimo entero no inferior a x).

Paso 2. Hacer $s = \sqrt{t^2 - n}$.

Paso 3. Hacer *contador* $= 0$.

Paso 4. Mientras *contador* $< N$ y s no sea entero

Paso 4.1. Hacer t=t+1.

Paso 4.2. Hacer $s = \sqrt{t^2 - n}$.

Paso 4.3. Hacer *contador* = *contador* $+ 1$.

Paso 5. Si *contador* es menor que N y $t - s$ no es 1

Paso 5.1. Hacer $factores = (t + s, t - s)$.

Paso 6. En otro caso

Paso 6.1. Hacer $factores =$ "No se ha encontrado la factorización".

Paso 7. Salida $factores$.

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

  Observación: Para calcular el techo de un número se puede utilizar la función $ceiling()$.

EJERCICIOS.

1. Probar la función con ejemplos sencillos: 21, 35, ...
2. Factorizar $n_1 = 455459$, $n_2 = 200819$, $n_3 = 141467$.
   Solución: $455459 = 743 \cdot 613$; $200819 = 491 \cdot 409$; $141467 = 587 \cdot 241$.