

Test de primalidad: tercera version

Conclusión: el test basado en J_n no va a funcionar.

¿Qué hacemos entonces?

- ▶ En lugar de utilizar J_n , vamos a usar las herramientas que desarrollamos sobre el siguiente conjunto (n impar):

$$S_n = \{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n} \text{ ó } a^{\frac{n-1}{2}} \equiv -1 \pmod{n}\}$$

¡Esto sí funciona!

Test de primalidad: un intento exitoso

Vamos a diseñar un test de primalidad considerando los conjuntos:

$$\begin{aligned} S_n^+ &= \{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n}\} \\ S_n^- &= \{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \equiv -1 \pmod{n}\} \\ S_n &= S_n^+ \cup S_n^- \end{aligned}$$

Para hacer esto necesitamos estudiar algunas propiedades de los conjuntos S_n^+ , S_n^- y S_n

- Consideramos primero el caso en que n es primo, y luego el caso en que n es compuesto

Una propiedad fundamental de S_n para n primo

Proposición

Si $n \geq 3$ es primo, entonces $S_n = \mathbb{Z}_n^*$

Demostración: Si $a \in \{1, \dots, n-1\}$, tenemos que $a^{n-1} \equiv 1 \pmod{n}$

Por lo tanto $(a^{\frac{n-1}{2}})^2 \equiv 1 \pmod{n}$, de lo cual se deduce que:

$$(a^{\frac{n-1}{2}} + 1) \cdot (a^{\frac{n-1}{2}} - 1) \equiv 0 \pmod{n}$$

Así, dado que n es primo se concluye que $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$ ó $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$

► ¿Por qué?



La estructura de S_n y las raíces modulares

Definition

b es una raíz cuadrada de a en módulo n si $b^2 \equiv a \pmod{n}$

Ejemplo

3 es una raíz cuadrada de 9 en módulo 16, y 3 es una raíz cuadrada de 4 en módulo 5.

Vamos a ver que hay una relación estrecha entre S_n y las raíces cuadradas modulares.

- Esta relación es fundamental para el test de primalidad

La estructura de S_n y las raíces modulares

Teorema

Sea $n \geq 3$ un primo y $a \in \{1, \dots, n-1\}$. Entonces a tiene raíz cuadrada en módulo n si y sólo si $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$

Demostración: ($\Rightarrow$) Suponga que a tiene raíz cuadrada en módulo n

► Sea $b \in \{1, \dots, n-1\}$ tal que $b^2 \equiv a \pmod{n}$

Se tiene que:

$$\begin{aligned} a^{\frac{n-1}{2}} &\equiv (b^2)^{\frac{n-1}{2}} \pmod{n} \\ &\equiv b^{n-1} \pmod{n} \\ &\equiv 1 \pmod{n} \end{aligned}$$

Existencia de raíces modulares: demostración

($\Leftarrow$) Para demostrar esta dirección del teorema, usamos el siguiente lema.

Sea $p(x)$ el polinomio:

$$p(x) = \sum_{i=0}^k a_i x^i,$$

donde $k \geq 1$, $a_k \in \{1, \dots, n-1\}$ y cada $a_j \in \{0, \dots, n-1\}$
($0 \leq j \leq k-1$)

Decimos que a es una raíz de $p(x)$ en módulo n si $p(a) \equiv 0 \pmod{n}$

Existencia de raíces modulares: demostración

Lema

$p(x)$ tiene a lo más k raíces en módulo n

Demostración: Decimos que dos polinomios $p_1(x)$ y $p_2(x)$ son congruentes en módulo n si para todo $a \in \{0, \dots, n-1\}$:

$$p_1(a) \equiv p_2(a) \pmod{n}$$

Notación

$$p_1(x) \equiv p_2(x) \pmod{n}$$

Sea a una raíz de $p(x)$ en módulo n

Existencia de raíces modulares: demostración

Vamos a demostrar que existe un polinomio $q(x)$ de grado $k - 1$ tal que:

$$p(x) \equiv (x - a) \cdot q(x) \pmod{n}$$

Pero antes de demostrar esto, vamos a mostrar que de esta propiedad se concluye que el lema es cierto.

Si c es una raíz de $p(x)$ en módulo n , entonces $p(c) \equiv 0 \pmod{n}$

- ▶ Como $p(x) \equiv (x - a) \cdot q(x) \pmod{n}$, concluimos que
 $(c - a) \cdot q(c) \equiv 0 \pmod{n}$

Dado que n es primo, si $d \cdot e \equiv 0 \pmod{n}$, entonces $d \equiv 0 \pmod{n}$ o $e \equiv 0 \pmod{n}$

- ▶ Tenemos entonces que $c \equiv a \pmod{n}$ o $q(c) \equiv 0 \pmod{n}$

Existencia de raíces modulares: demostración

Así, tenemos que c es la raíz a que ya habíamos identificado o es una raíz de $q(x)$ en módulo n

Concluimos que el número de raíces de $p(x)$ en módulo n es menor o igual a uno más el número de raíces de $q(x)$ en módulo n

- ▶ Como $q(x)$ tiene grado $k - 1$, si continuamos usando este argumento (o usamos inducción) concluimos que el número de raíces de $p(x)$ es menor o igual a k

Existencia de raíces modulares: demostración

Nótese que el argumento anterior no funciona si n es compuesto.

- Dado que podemos tener d y e tales que $d \cdot e \equiv 0 \pmod{n}$, $d \not\equiv 0 \pmod{n}$ y $e \not\equiv 0 \pmod{n}$

De hecho, si n es compuesto no es necesariamente cierto que el número de raíces de un polinomio está acotado superiormente por su grado.

Ejemplo

Si $n = 35$, tenemos que $5 \cdot 7 \equiv 0 \pmod{35}$, pero $5 \not\equiv 0 \pmod{35}$ y $7 \not\equiv 0 \pmod{35}$

En este caso tenemos cuatro raíces para el polinomio $p(x) = x^2 - 1$

- Ya que $1^2 \equiv 1 \pmod{35}$, $6^2 \equiv 1 \pmod{35}$, $29^2 \equiv 1 \pmod{35}$ y $34^2 \equiv 1 \pmod{35}$

Existencia de raíces modulares: demostración

Volvemos entonces a la demostración de que existe un polinomio $q(x)$ de grado $k - 1$ tal que:

$$p(x) \equiv (x - a) \cdot q(x) \pmod{n}$$

Definimos $q(x)$ como:

$$q(x) = \sum_{i=0}^{k-1} b_i x^i,$$

donde $b_i = a_{i+1} + a_{i+2} \cdot a + \cdots + a_k \cdot a^{k-1-i}$

Existencia de raíces modulares: demostración

Se tiene que:

$$\begin{aligned}(x - a) \cdot q(x) &= \left(\sum_{i=0}^{k-1} b_i x^{i+1} \right) + \left(\sum_{i=0}^{k-1} (-a \cdot b_i) x^i \right) \\&= \left(\sum_{i=1}^k b_{i-1} x^i \right) + \left(\sum_{i=0}^{k-1} (-a \cdot b_i) x^i \right) \\&= b_{k-1} \cdot x^k + \left(\sum_{i=1}^{k-1} (b_{i-1} - a \cdot b_i) x^i \right) - a \cdot b_0\end{aligned}$$

Así, dado que:

$$b_{k-1} = a_k$$

Existencia de raíces modulares: demostración

Y dado que para $i \in \{1, \dots, k-1\}$:

$$\begin{aligned}(b_{i-1} - a \cdot b_i) &= a_i + a_{i+1} \cdot a + \dots + a_k \cdot a^{k-i} - \\ &\quad a \cdot (a_{i+1} + a_{i+2} \cdot a + \dots + a_k \cdot a^{k-1-i}) \\ &= a_i + a_{i+1} \cdot a + \dots + a_k \cdot a^{k-i} - \\ &\quad a_{i+1} \cdot a - a_{i+2} \cdot a^2 - \dots - a_k \cdot a^{k-1} \\ &= a_i\end{aligned}$$

Concluimos que:

$$(x - a) \cdot q(x) = \left(\sum_{i=1}^k a_i \cdot x^i \right) - a \cdot b_0$$

Existencia de raíces modulares: demostración

Pero:

$$\begin{aligned} -a \cdot b_0 &= -a \cdot (a_1 + a_2 \cdot a + \dots + a_k \cdot a^{k-1}) \\ &= -a_1 \cdot a - a_2 \cdot a^2 - \dots - a_k \cdot a^k \end{aligned}$$

De lo cual deducimos que:

$$a_0 \equiv -a \cdot b_0 \pmod{n},$$

ya que $a_k \cdot a^k + \dots + a_1 \cdot a + a_0 \equiv 0 \pmod{n}$

Tenemos entonces que:

$$(x - a) \cdot q(x) \equiv p(x) \pmod{n}$$



Existencia de raíces modulares: demostración

Volvemos a la demostración inicial:

- ▶ Queremos demostrar que si $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$, entonces a tiene raíz en módulo n

$$\text{Sea } R = \{b^2 \mid 1 \leq b \leq \frac{n-1}{2}\}$$

Como $b^2 \equiv (n - b)^2 \pmod{n}$, se tiene que:

a tiene raíz en módulo n si y sólo si $a \in R$

Existencia de raíces modulares: demostración

Por ($\Rightarrow$) sabemos que:

$$R \subseteq \{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n}\}$$

Además, sabemos que si $1 \leq b < c \leq \frac{n-1}{2}$ y $b^2 \equiv c^2 \pmod{n}$:

$$(c - b) \cdot (c + b) \equiv 0 \pmod{n}$$

Así, dado que $2 \leq b + c \leq n - 1$, concluimos que $b \equiv c \pmod{n}$

► Dado que n es primo

Obtenemos una contradicción puesto que $1 \leq (c - b) \leq \frac{n-1}{2}$

► Por lo tanto: $|R| = \frac{n-1}{2}$

Existencia de raíces modulares: demostración

Sea $p(x) = x^{\frac{n-1}{2}} - 1$. También sabemos que $p(x)$ tiene a lo más $\frac{n-1}{2}$ raíces en módulo n .

► Por lo tanto: $|\{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n}\}| \leq \frac{n-1}{2}$

Concluimos que:

$$\frac{n-1}{2} = |R| \leq |\{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n}\}| \leq \frac{n-1}{2}$$

Por lo tanto:

$$R = \{a \in \{1, \dots, n-1\} \mid a^{\frac{n-1}{2}} \equiv 1 \pmod{n}\}$$

Así, si $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$, se tiene que a tiene raíz en módulo n



Una propiedad fundamental de S_n^+ y S_n^- para n primo

Si $n \geq 3$ es primo, dado que $S_n = \mathbb{Z}_n^*$, obtenemos por el teorema:

- Si a no tiene raíz en módulo n , entonces $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$

Así, obtenemos el siguiente corolario de los resultados anteriores.

Corolario

Si $n \geq 3$ es primo:

$$|S_n^+| = |S_n^-| = \frac{n-1}{2}$$

Una propiedad fundamental de S_n para n compuesto

Teorema

Sea $n = n_1 \cdot n_2$, donde $n_1, n_2 \geq 3$ y $\text{MCD}(n_1, n_2) = 1$. Si existe $a \in \mathbb{Z}_n^*$ tal que $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$, entonces:

$$|S_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$$

Para demostrar el teorema necesitamos el Teorema Chino del resto

Para recordar: un teorema muy útil

Teorema (Chino del Resto)

Suponga que $\text{MCD}(m, n) = 1$. Para todo a y b , existe c tal que:

$$c \equiv a \pmod{m}$$

$$c \equiv b \pmod{n}$$

Demostración: Dado que $\text{MCD}(m, n) = 1$, existen d y e tales que:

$$n \cdot d \equiv 1 \pmod{m}$$

$$m \cdot e \equiv 1 \pmod{n}$$

Sea $c = a \cdot n \cdot d + b \cdot m \cdot e$

Se tiene que:

$$c \equiv a \pmod{m}$$

$$c \equiv b \pmod{n}$$



La demostración del teorema inicial

Suponga que $a \in \mathbb{Z}_n^*$ y $a^{\frac{n-1}{2}} \equiv -1 \pmod n$

Por Teorema Chino del Resto, existe b tal que:

$$b \equiv a \pmod{n_1}$$

$$b \equiv 1 \pmod{n_2}$$

Entonces: $a = \alpha \cdot n_1 + b$ y $1 = \beta \cdot n_2 + b$

► Por lo tanto $\text{MCD}(b, n) = 1$, ya que $n = n_1 \cdot n_2$ y $a \in \mathbb{Z}_n^*$

La demostración del teorema inicial

Además, tenemos que:

$$\begin{array}{rclcl} b^{\frac{n-1}{2}} & \equiv & a^{\frac{n-1}{2}} & \equiv & -1 \pmod{n_1} \\ & & b^{\frac{n-1}{2}} & \equiv & 1 \pmod{n_2} \end{array}$$

Dado que $n = n_1 \cdot n_2$ concluimos que:

$$\begin{array}{rclcl} b^{\frac{n-1}{2}} & \not\equiv & 1 & \pmod{n} \\ b^{\frac{n-1}{2}} & \not\equiv & -1 & \pmod{n} \end{array}$$

La demostración del teorema inicial

Sea $c = (b \bmod n)$. Concluimos que $c \notin S_n$ y $c \in \mathbb{Z}_n^*$

► Por lo tanto: $S_n \subsetneq \mathbb{Z}_n^*$

Pero se tiene que $(S_n, \cdot)$ es un subgrupo de $(\mathbb{Z}_n^*, \cdot)$

► ¿Por qué?

Por Teorema de Lagrange: $|S_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$



Un test de primalidad aleatorizado

Ya tenemos los ingredientes esenciales para el test de primalidad

- ▶ Sólo nos falta implementar algunas funciones auxiliares

Necesitamos desarrollar un algoritmo eficiente para determinar si un número n es la potencia (no trivial) de otro número.

Verificando si un número es la potencia de otro

Primero necesitamos una función para calcular n^k

- Usamos el algoritmo de exponenciación rápida pero sin considerar el módulo

```
EXP( $n, k$ )  
  if  $k = 1$  then return  $n$   
  else if  $k$  es par then  
     $val := \mathbf{EXP}(n, \frac{k}{2})$   
    return  $val \cdot val$   
  else  
     $val := \mathbf{EXP}(n, \frac{k-1}{2})$   
    return  $val \cdot val \cdot n$ 
```

Verificando si un número es la potencia de otro

Dado un número natural $n \geq 2$, la siguiente función verifica si existen $m, k \in \mathbb{N}$ tales que $k \geq 2$ y $n = m^k$

EsPotencia(n)

if $n \leq 3$ **then return** no

else

for $k := 2$ **to** $\lfloor \log_2(n) \rfloor$ **do**

if **TieneRaízEntera**($n, k, 1, n$) **then return** sí

return no

Verificando si un número es la potencia de otro

La siguiente función verifica si existe $m \in \{i, \dots, j\}$ tal que $n = m^k$

- Vale decir, la llamada **TieneRaízEntera**($n, k, 1, n$) verifica si n tiene raíz k -ésima entera

TieneRaízEntera(n, k, i, j)

if $i = j$ then

if **EXP**(i, k) = n then return sí

else return no

else if $i < j$ then

$p := \lfloor \frac{i+j}{2} \rfloor$

$val := \mathbf{EXP}(p, k)$

if $val = n$ then return sí

else if $val < n$ then return **TieneRaízEntera**($n, k, p + 1, j$)

else return **TieneRaízEntera**($n, k, i, p - 1$)

else return no

La complejidad de **EsPotencia**

Consideramos la multiplicación de números enteros como la operación básica a contar

Tenemos que:

- ▶ En el peor caso **EsPotencia**(n) realiza $(\lfloor \log_2(n) \rfloor - 1)$ llamadas a la función **TieneRaízEntera**
- ▶ Existe $c \in \mathbb{N}$ tal que la llamada **TieneRaízEntera**($n, k, 1, n$) realiza en el peor caso a lo más $c \cdot \log_2(n)$ llamadas a la función **EXP**
- ▶ **EXP**(n, k) en el peor caso es $O(\log_2(k))$

La complejidad de **EsPotencia**

Concluimos que **EsPotencia**(n) en el peor caso es $O([\log_2(n)]^3)$

Vale decir, **EsPotencia** en el peor caso es de orden polinomial en el tamaño de la entrada

- ▶ Se puede llegar a la misma conclusión si consideramos todas las operaciones realizadas por **EsPotencia**

Un test de primalidad aleatorizado

El siguiente algoritmo aleatorizado determina si un número entero $n \geq 2$ es primo.

El algoritmo recibe como entrada un valor entero $k \geq 1$ que es usado para controlar la probabilidad de error.

Un test de primalidad aleatorizado

TestPrimalidad(n, k)

if $n = 2$ then return PRIMO

else if n es par then return COMPUESTO

else if EsPotencia(n) then return COMPUESTO

else

 sea $a_1, \dots, a_k$ una secuencia de números elegidos de
 manera uniforme e independiente desde $\{1, \dots, n - 1\}$

for $i := 1$ to k do

if $\text{MCD}(a_i, n) > 1$ then return COMPUESTO

else $b_i := \text{EXP}(a_i, \frac{n-1}{2}, n)$

$neg := 0$

for $i := 1$ to k do

if $b_i \equiv -1 \pmod n$ then $neg := neg + 1$

else if $b_i \not\equiv 1 \pmod n$ then return COMPUESTO

if $neg = 0$ then return COMPUESTO

else return PRIMO

Test de primalidad: probabilidad de error

TestPrimalidad se puede equivocar de dos formas:

- Suponga que $n \geq 3$ es primo. En este caso **TestPrimalidad** da una respuesta incorrecta si $b_i \equiv 1 \pmod{n}$ para todo $i \in \{1, \dots, k\}$

Dado que $|S_n^+| = |S_n^-| = \frac{n-1}{2}$:

- La probabilidad de que para un número a elegido con distribución uniforme desde $\{1, \dots, n-1\}$ se tenga que $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$ es $\frac{1}{2}$

Por lo tanto, la probabilidad de que **TestPrimalidad** diga COMPUESTO para $n \geq 3$ primo es $\left(\frac{1}{2}\right)^k$

Test de primalidad: probabilidad de error

- ▶ Suponga que n es compuesto, n es impar y n no es de la forma m^ℓ con $\ell \geq 2$
 - ▶ Si n es par o n es de la forma m^ℓ con $\ell \geq 2$, entonces **TestPrimalidad** da la respuesta correcta COMPUESTO

Tenemos entonces que $n = n_1 \cdot n_2$ con $n_1 \geq 3$, $n_2 \geq 3$ y $\text{MCD}(n_1, n_2) = 1$

Además debe existir $a \in \{1, \dots, n-1\}$ tal que $\text{MCD}(a, n) = 1$ y $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$

- ▶ Si esto no es cierto **TestPrimalidad** retorna COMPUESTO, dado que si **TestPrimalidad** logra llegar a la última instrucción **if** entonces neg necesariamente es igual a 0

Test de primalidad: probabilidad de error

Concluimos que $|S_n| \leq \frac{1}{2} \cdot |\mathbb{Z}_n^*|$

- Por la caracterización que dimos de S_n para n compuesto

Vamos a utilizar este resultado para acotar la probabilidad de error:

$$\Pr\left(\left(\bigwedge_{i=1}^k \text{MCD}(a_i, n) = 1 \wedge (b_i \equiv 1 \pmod{n} \vee b_i \equiv -1 \pmod{n})\right) \wedge \left(\bigvee_{j=1}^k b_j \equiv -1 \pmod{n}\right)\right)$$

Test de primalidad: probabilidad de error

Tenemos que:

$$\Pr\left(\left(\bigwedge_{i=1}^k \text{MCD}(a_i, n) = 1 \wedge (b_i \equiv 1 \bmod n \vee b_i \equiv -1 \bmod n)\right) \wedge \left(\bigvee_{j=1}^k b_j \equiv -1 \bmod n\right)\right) \leq$$
$$\Pr\left(\bigwedge_{i=1}^k \text{MCD}(a_i, n) = 1 \wedge (b_i \equiv 1 \bmod n \vee b_i \equiv -1 \bmod n)\right)$$

Por lo tanto sólo necesitamos una cota superior para la última expresión.

Test de primalidad: probabilidad de error

Tenemos que:

$$\begin{aligned} & \Pr\left(\bigwedge_{i=1}^k \text{MCD}(a_i, n) = 1 \wedge (b_i \equiv 1 \bmod n \vee b_i \equiv -1 \bmod n)\right) \\ &= \prod_{i=1}^k \Pr(\text{MCD}(a_i, n) = 1 \wedge (b_i \equiv 1 \bmod n \vee b_i \equiv -1 \bmod n)) \\ &= \prod_{i=1}^k \Pr((b_i \equiv 1 \bmod n \vee b_i \equiv -1 \bmod n) \mid \text{MCD}(a_i, n) = 1) \cdot \\ & \qquad \qquad \qquad \Pr(\text{MCD}(a_i, n) = 1) \\ &\leq \prod_{i=1}^k \Pr((b_i \equiv 1 \bmod n \vee b_i \equiv -1 \bmod n) \mid \text{MCD}(a_i, n) = 1) \\ &= \prod_{i=1}^k \Pr(a_i \in S_n \mid a_i \in \mathbb{Z}_n^*) \leq \prod_{i=1}^k \frac{1}{2} = \frac{1}{2^k} \end{aligned}$$

Test de primalidad: probabilidad de error

Concluimos que la probabilidad de que el test diga PRIMO para el valor compuesto n está acotada por $\left(\frac{1}{2}\right)^k$

En ambos casos (si n es primo o compuesto) la probabilidad de error del algoritmo está acotada por $\left(\frac{1}{2}\right)^k$

- ▶ ¡Si $k = 100$, esta probabilidad está acotada por $\left(\frac{1}{2}\right)^{100} \approx 7.9 \times 10^{-31}$!