

Fermat, Euler & Pseudoprimes

Universidad de Cuenca

Optativa 6 - Criptología

Dr. Diego Ponce

Capítulo 3

Freddy L. Abad L.
freddy.abadl@ucuenca.edu.ec

CAP 3

3.1) In the exercises 3.1 - 3.3, use the facts that $a \equiv b \pmod{m}$ if and only if m divides $a - b$

3.1. Prove that if $a \equiv x \pmod{m}$ and $b \equiv y \pmod{m}$ then $ab \equiv xy \pmod{m}$

Sei $a \geq b (\text{mod } m)$ entonces m divide $a - b$, si $a^b \in \mathbb{Z}^+$
 $\rightarrow a \text{ mod } m \equiv b \text{ mod } m$

$$\text{Sea } \begin{cases} a \equiv x \pmod{m} \\ b \equiv y \pmod{m} \end{cases} \rightarrow \begin{cases} a+b \equiv (x \pmod{m}) + (y \pmod{m}) \\ a+b \equiv (x+y) \pmod{m} \\ axb \equiv (x \pmod{m})(y \pmod{m}) \\ axb \equiv xy \pmod{m} \end{cases}$$

COMPLETOS
OBSERVACIONES
DE FERMAT

3.2 Prove that if $\gcd(x, m) = 1$ and, if $ax \equiv b \pmod{m}$, then $a \equiv b \pmod{m}$. Show that the conclusion does not necessarily follow if $\gcd(x, m)$ is not 1.

supongá $\gcd(x, m) = 1$

Sea $a \equiv b \pmod{m}$

Sea $x \in \mathbb{Z}$ entonces

erste un ersten $k/a = b/km$) d'chnig. In weiteren

7

$$Ax = (b + km) \alpha$$

$ax = bx \pmod{m}$, donde $kx \in \mathbb{Z}$

Por la relación de congruencia

$$ax \equiv bx \pmod{m}$$

luegs $a \equiv b \pmod{m}$

Se pueden eliminar los " x " de los coprinos es decir $\gcd(x_1, x_2)$, coprinos entre sí, entonces de la observación de Fermat.

$$x \equiv 1 \pmod{m}$$

De no cumplir esta congruencia no es posible afirmar.

$$ax \equiv bx \pmod{m} \quad = \quad a \equiv b \pmod{m}$$

3.3 Let $g = \gcd(x, m)$. Prove that $ax \equiv bx \pmod{m}$, if and only if $a \equiv b \pmod{m/g}$.

Sea $g = \gcd(x, m)$ $ax \equiv bx \pmod{m} \Leftrightarrow a \equiv b \pmod{m/g}$

Observación de Fermat: Si a es primo, $\phi(p-1)$
 q divide a $MCP = C$

$$\phi(3) = 2^3 - 1 = 7$$

entonces $q \equiv 1 \pmod{p}$

Sea $a \equiv b \pmod{m}$, por relación congruencia $ax \equiv bx \pmod{m}$, si g es el máximo común divisor de x^m , entonces puede existir la relación $a \equiv b \pmod{m/g}$, entonces:

$$ax \equiv bx \pmod{m} \Leftrightarrow a \equiv b \pmod{m/g}$$

3.4 Write a program to implement Alg 3.3 Use it to test whether 2212324139 and 19707683773 are pseudoprimes (base 2)

public class algoritmo 33 {

```
public static void main(String[] args) {
    /* Exponenciación rápida (método binario) */
    int a=5, b=64, m=19;
    int resultado = calculate(a, b, m);
    System.out.println("Resultado Exponenciación rápida: " + resultado);
}
```

```
public static int calculate(int a, int b, int m) {
    int exp=1;
    int xp=a%m;
    while (b>0) {
        if ((b%2)==0) {
            exp=(exp+xp)%m;
            xp=(xp*xp)%m;
            b=b/2;
        }
        return exp;
    }
}
```

35 Find ten 12-digit probable primes.

$$a = 100000000000$$

$$b = 999999999999$$

for i in range(a, b):

$$\text{valcr} = (2^{**700000}) \% i$$

$$\text{valcr2} = (2^{**1000}) \% i$$

$$\text{valcr3} = (2^{**100}) \% i$$

salir = False

mod = 1

cont = 0

if (i%2 != 0):

if (i%5 != 0):

while (salir == False):

if (cont + 100000 >= i-1):

if (cont + 100 >= i-1):

mod = (mod * 2) % i

cont += 1

if (cont == i-1):

salir = True

else:

cont += 100

mod *= valcr2

mod = mod % i

else:

cont += 1000

mod *= valcr3

mod = mod % i

else:

cont += 100000

mod *= valcr1

mod = mod % i

if (mod == 1):

print(i)

100000000003
100000000019
100000000057
100000000063
100000000069
100000000073
100000000091
100000001009
10000001099

3.6 Find a 100 digit probable prime.

Hacerlo toma mucha capacidad de procesamiento si se quiere hacer la comprobación de Fermat a por lo menos 1000 números de 100 dígitos y aun así no se garantiza el obtener un posible primo en ese rango.

Una solución podría ser coger 9 números del más anterior y multiplicarlos lo que nos daría un número pseudoprimo que cumple con las ecuaciones de Fermat.

$$\begin{aligned} & 1000000000009 * 100000000019 + 100000000057 * \dots \\ \dots & * 100000000063 * 100000000069 * 100000000073 + \dots \\ \dots & * 100000000091 * 10000001009 * 10000001099 = \\ & \underline{\underline{1 * 10^{199}}} \end{aligned}$$

35

3.7 Take the unfactored numbers from the answer 2.15 and test whether they are composite or probable primes.

$$[200000001 - 200000002 - 200000003]$$

⋮

$$[200000097 - 200000098 - 200000099]$$

Utilizando el algoritmo del ejercicio 3.5 ningún número dentro del rango es un # primo probable, todos son compuestos ya que no satisface la observación de Fermat.

(3.8) Prove that if the p is prime and $q = 2^p + 1$ and q divides $M(p)$, then q must be prime. (and $q \geq p$)
 As an example, verify that 1103 is a prime and that 2207 divides $M(1103)$. By the first part of this exercise, 2207 must be prime.

$$2^p - 1 \bmod q = 0 \quad \text{sea} \quad q = 2^p + 1$$

$$2^p = q - 1$$

$$(2^p + 1)(2^p - 1) \bmod q = 0$$

$$2^{2^p} - 1 \bmod q = 0$$

$$2^{q-1} - 1 \bmod q = 0$$

$$2^{q-1} \bmod q = 1 \bmod q$$

Completa el Teorema de Fermat para lo tanto que q es primo o pseudo-primo

1103 pasa la observación de Fermat y el test de primalidad dividiendo el número entre el rango de valores $[2, 1103^{1/(2)}]$ se comprueba que el número es primo.

(Trial Division)

$$\text{Sea } q = 2^* 1103 + 1 = 2207$$

2207 pasa la observación de Fermat y el test de primalidad dividiendo el número entre el rango de valores $[2, 1103^{1/(2)}]$ se comprueba que el número es primo.

(Trial Division)

Entonces:

$$M = ((2^{**} 1103) - 1) \% 2207$$

$$M = 0$$

3.9 Using the idea of Ex 3.8, show that if p is a prime q divides $M(p)$, and q is less than p^2 , then q must be a prime. As an example, verify that 1153 is a prime and that 267497 divides $M(1153)$ and so 267497 must be a prime.

a) Si " p " es primo, " q " divide $M(p)$ y $q < p^2$

y,

q debe ser primo

Por observación Fermat, si un primo " q " divide a $M(p)$

$$q \equiv 1 \pmod{p}$$

Los valores que puede tomar que son 1 hasta $p^2 - 1$

$$p^2 - 1 \equiv 1 \pmod{p}$$

$$\boxed{p^2 \equiv 0 \pmod{p}}$$

Como " p " es un numero primo, la igualdad se cumple para todos los primos " p ".

Y como " q " podía tomar cualquier valor menor o igual a $(p^2 - 1)$, " q " es primo para todos los valores de " p " que sean primos.

b) Usando el algoritmo de la "División trial" se verifica que 1153 es un numero primo ya que no tiene divisores en el rango $[2, \sqrt{1153}]$

$q = 267497$ divide a $M(1153)$

$$p = 1153 \Rightarrow p^2 = 1329409$$

$q < p^2$ se cumple.

Con la Observación de Fermat

$$q \equiv 1 \pmod{p}$$

$$\boxed{267497 \equiv 1 \pmod{1153}}$$

Se cumple la condición de Fermat.

3.10 Ex 3.9 implies the following primality test: If p is a prime factor of $q-1$ such that p^2 is larger than q and q divides $M(p)$, then q must be prime. Try using this test on the ten probable primes which you found in exercise 3.5. For how many of them does it work? When it fails, why does it fail? Does its failure mean that your probable primes is not prime?

Si " p " es un factor primo de " $q-1$ " tal que " p^2 " es más grande que " q " y " q " divide $M(p)$

Entonces " q " debe ser primo

$$q \equiv 1 \pmod{p}$$

1 00000000003
 1 000000000019
 1 000000000057
 1 000000000063
 1 000000000069
 1 000000000073
 1 000000000091
 1 000000000103
 1 000000000129

$$\textcircled{1} \quad n = 100000000003$$

Probando con el algoritmo de la "Division Trial" verificamos que el numero n es primo ya que no tiene factores

Como " n " es primo su única divisor " q " es el mismo valor de n

$$\therefore n = q$$

Ahora si " p " es un factor primo de $(q-1)$

$$\Rightarrow q-1 \equiv 100000000002$$

Factorizamos $(q-1)$ con la "Division Trial"

los factores " p " de $(q-1)$ son

$$p_1 = 2$$

$$p_2 = 3$$

$$p_3 = 7$$

$$p_4 = 2380952381$$

► Para el caso $p=2$

$$p^2 = 4$$

∴ $p^2 < q \Rightarrow$ No se cumple la condición del ejercicio.

► Para el caso $p=3$

$$p^2 = 9$$

∴ $p^2 < q \Rightarrow$ No se cumple la condición del ejercicio.

► Para el caso $p=7$

$$p^2 = 49$$

∴ $p^2 < q \Rightarrow$ No se cumple la condición del ejercicio.

► Para el caso $p=2380952381$

$$p^2 = 5668934240589569161$$

∴ $p^2 > q \Rightarrow$ Se cumple la condición del ejercicio.

⇒ El valor de "q" es primo.

Un análisis similar sigue para los restantes y enteros primos probables de 12 dígitos, ya que verificando con el algoritmo de la "División trial" los 9 números restantes también son primos.

¿Para cuantos de ellos trabaja?

El test funciona para todos los primos probables cuyo factor "p" de $(q-1)$ es el mayor factor encontrado con la "División trial".

¿Cuando falla el test y por qué?

El test falla cuando el factor "p" de $(q-1)$ es cualquier factor distinto del factor más grande encontrado.

¿(esta falla significa que tu primo probable no es un primo?)
No, el problema se verifica que el valor es primo con la división trial.

3.11 Verify that the following algorithm produces the binary expansion of n :

$$n = b_k \times 2^k + b_{k-1} \times 2^{k-1} + \dots + b_1 \times 2 + b_0$$

INITIALIZE: READ n

$$i \leftarrow -1$$

BINARY LOOP: WHILE $n \neq 0$ DO

$$i \leftarrow i + 1$$

$$b_i \leftarrow n \bmod 2$$

$$n \leftarrow \lceil n/2 \rceil$$

TERMINATE:

$$k \leftarrow 2$$

FOR $i=1$ to k DO,

WRITE b_i

public class algorithm-3-11 {

 public static void main (String [] args)

 {

 Integer n = 27;

 algoritmo(n);

 }

 public static void algoritmo (Integer n)

 {

 System.out.println ("INPUT: " + n);

 ArrayList < Integer > b = new ArrayList <>();

 // BINARY LOOP

 int i = 0;

 while (n != 0) {

 b.add (n % 2);

 n /= 2;

 i++;

 }

 // TERMINATE

 System.out.println ("Binary Expansion: ");

 for (int j = b.size () - 1; j >= 0; j--)

 System.out.print ("(" + b.get(j) + "*2^" + j + ") + ");

 System.out.println ("");

 }

3.12 Following variation of Alg 3.3 is often called peasant multiplication because it's has been used for centuries by European peasants who could use it to multiply without needing to memorize multiplication tables. Prove that it does provide the product of a and b as long as b is no negative, and that it's equivalent to multiplication in base 2.

INITIALIZED READ a, b
 $n \leftarrow 0$

BINARY_LOOP WHILE $b \neq 0$ DO
 IF b is odd THEN DO
 $n \leftarrow n + a$
 $b \leftarrow \lceil b/2 \rceil$
 $a \leftarrow a + a$

TERMINATE: WRITE n

```
public class algorithm_3_12 {
    public static void main(String[] args) {
        //INITIALIZE
        Integer a=10, b=10;
        algoritmo(a, b);

        //BINARY LOOP
        int n=0;
        while (b!=0) {
            if (isOdd(b)) {
                n+=a;
            }
            b/=2;
            a+=a;
        }

        //TERMINATE
        System.out.println("Multiplication(a*b) = " + n);
    }

    public static boolean isOdd(Integer n) {
        return n%2!=0;
    }
}
```

3.13 Use theorem 3.2 to prove that if $\gcd(b, 561) = 1$ then

$$\begin{aligned} b^{560} &\equiv 1 \pmod{3} \\ b^{560} &\equiv 1 \pmod{11} \\ b^{560} &\equiv 1 \pmod{17} \end{aligned}$$

It follows from these 3 congruences that

$$b^{560} \equiv 1 \pmod{561}$$

too 3.2 = Si p es un primo que no divide a b , entonces
 $b^{p-1} \equiv 1 \pmod{p}$

b puede ser 2, 3, 5

$$\begin{aligned} 2^{560} &\equiv 1 \pmod{3} & 2^{560} &\equiv 1 \pmod{561} \\ 2^{560} &\equiv 1 \pmod{11} \\ 2^{560} &\equiv 1 \pmod{7} \end{aligned}$$

$$\begin{aligned} 3^{560} &\equiv 1 \pmod{3} & 3^{560} &\equiv 1 \pmod{561} \\ 3^{560} &\equiv 1 \pmod{11} \\ 3^{560} &\equiv 1 \pmod{7} \end{aligned}$$

$$\begin{aligned} 5^{560} &\equiv 1 \pmod{3} & 5^{560} &\equiv 1 \pmod{561} \\ 5^{560} &\equiv 1 \pmod{11} \\ 5^{560} &\equiv 1 \pmod{7} \end{aligned}$$

Si: $b=2 \quad 561 \bmod 2 = 1 \quad \gcd(2, 561) = 1$

$b=3 \quad 561 \bmod 3 = 0 \quad \gcd(3, 561) = 3$

$b=5 \quad 561 \bmod 5 = 1 \quad \gcd(5, 561) = 1$

b puede ser todos los números enteros positivos menores a 561 excepto 3, 11, 17 que son factores de 561

3.14 Find the smallest pseudo prime (base 2) which is larger than a thousand. Prove that this number is in fact a Carmichael number.

Nº Carmichael

$$m \neq n \rightarrow \phi(m) = \phi(n) \rightarrow \phi(6) = \phi(4) = 2 \quad 674$$

Lista de # pseudo primos (orden de mil) = 3561

$$a^{n-1} \equiv 1 \pmod{n}$$

$$\Rightarrow a=2, n=3561$$

$$2^{n-1} \equiv 1 \pmod{n}$$

$$2^{3561-1} \equiv 1 \pmod{3561}$$

$$2^{3560} \equiv 1 \pmod{3561}$$

$$\therefore n=3561 \left\{ \begin{array}{l} * \text{Número menor a } 100 \\ * \text{Número Pseudo primo } (3561 = 3 \times 11 \times 17) \\ * \text{Número Carmichael (con base 2)} \\ (a^{n-1} \equiv 1 \pmod{n}) \end{array} \right.$$

3.15 Find the value of $\phi(n)$ for $n=9, 10, 11, 12$

$\phi(n)$ función de Euler (cantidad de # primos no factor de "n")

primos menores a 12: {2, 3, 5, 7, 11}

$$\phi(10) = \{1, 3, 7\} \quad \therefore \phi(10) = 3$$

$$\phi(11) = \{1, 2, 3, 5, 7\} \quad \therefore \phi(11) = 5$$

$$\phi(12) = \{1, 5, 7, 11\} \quad \therefore \phi(12) = 4$$

$$\phi(9) = \{1, 2, 5, 7\} \quad \therefore \phi(9) = 4/1$$



3.16 Write a program to compute $\phi(n)$ by counting one for each $i \leq n$ such that $\gcd(i, n) = 1$. List the values of $\phi(n)$ for n up to 200.

```
public class euler {
```

```
    public static int gcd(int num1, int num2) {
```

```
        while (num1 != num2) {
```

```
            if (num1 > num2) {
```

```
                num1 -= num2;
```

```
            } else {
```

```
                num2 -= num1;
```

```
            }
```

```
        }
```

```
        public static int funcionEuler (int n) {
```

```
            if (n == 1) {
```

```
                return 1;
```

```
            }
```

```
            int aux = 0;
```

```
            for (int i = 1; i < n; i++) {
```

```
                if ( $\gcd(i, n) == 1$ ) {
```

```
                    aux++;
```

```
                }
```

```
            }
```

```
            return aux;
```

```
}
```

```
    public static void main (String [] args) {
```

```
        System.out.println ( $\gcd(7, 31)$ );
```

```
}
```

3.17 What patterns can you pick out of table of values of $\phi(n)$?

Ejemplo:

$$\phi(3) = 2$$

$$\phi(57) = 36$$

$$\phi(122) = 60$$

$$\phi(999) = 648$$

tambien se da que a veces valores de $\phi(n)$ son grandes y los que le siguen a continuación son menores a su valor.

Ejemplo:

$$\phi(193) = 192$$

$$\phi(194) = 96$$

$$\phi(195) = 96$$

$$\phi(197) = 196$$

3.18 is $\phi(n)$ always even when n is larger than 2?
Prove your assertion.

Sí efectivamente siempre es así para lo podemos probar con algunos ejemplos

$$\phi(777) = 432 \quad \phi(21) = 12$$

$$\phi(27) = 18$$

3.19 prove that for any integer $x > 1$, we have that

$$\gcd(x^a - 1, x^b - 1) = x^{\gcd(a, b)} - 1$$

A partir de

$$x^k - 1 = (x-1)(1+x+x^2+\dots+x^{k-1})$$

y tomando $k = \gcd(a, b)$, donde $a = kp$ y $b = kp$
por lo tanto

$$x^a - 1 = (x-1)(1+x+x^2+\dots+x^{kp-2}+x^{kp-1})$$

$$x^a - 1 = (x-1)(x^{k(p-1)+k-1} + x^{k(p-1)+k-2} + \dots + x^{kp-1}) \\ + \dots + (x^{k-1} + x^{kp-2} + \dots + x+1)$$

$$x^a - 1 = (x-1)(x^{kp-1} + x^{kp-2} + \dots + x+1)(x^{k-1} + x^{kp-2} + \dots + x+1)$$

$$x^a - 1 = (x^k - 1)(x^{kp-1} + x^{kp-2} + \dots + x^k + 1)$$

Realizamos los mismos para b

$$x^b - 1 = (x^k - 1)(x^{k(q-1)} + x^{k(q-2)} + \dots + x^k + 1)$$

sabiendo $\gcd(p, q) = 1$ $\{p, q \text{ son coprimos}\}$ entonces

$$\gcd((x^{kp-1} + x^{kp-2} + \dots + x^k + 1), (x^{k(q-1)} + x^{k(q-2)} + \dots + x^k + 1)) \\ = 1$$

$$\gcd(x^a - 1, x^b - 1) = x^{\gcd(a, b)} - 1$$

3.20 Let p be a prime and b be an integer. Prove that if q is a prime divisor of $b^p - 1$ then

q divides $b-1$ or $q \equiv 1 \pmod{p}$

Si $p = b^p - 1$ y q es divisor.

$q \equiv 1 \pmod{p} \rightarrow$ Se quiere probar

A partir del teorema 3.1 y dividendo para que se obtiene

$$b^{q-1} - 1, b^p - 1$$

se sabe que

$$\gcd(x^a - 1, x^b - 1) = x^{\gcd(a, b)} - 1$$

Reemplazamos

$$\gcd(b^{q-1} - 1, b^p - 1) = b^{\gcd(p, q-1)} - 1$$

Consideramos a $q-1, p$ como primos relativos, lo que implica

$$\gcd(q-1, p) = 1 \text{ por lo tanto}$$

$$b^1 - 1 = \boxed{b-1}$$

por lo tanto q divide p y su resultado es $b-1$ o lo que significa $q \equiv 1 \pmod{p}$ //

LQDD

Freddy L. Abad L.

freddy.abad@ucuenca.edu.ec