EjerciciosSemana1

February 28, 2022

1 Ejercicios Semana 1 Teoría de Números y Criptografía

1.1 Ejercicio 1

Dado tu número n=26504975.

1- Mientras sea múltiplo de 2,3,5,7 u 11 le sumas uno. De forma que tu nuevo n no tenga esos divisores primos.

[1]:
```
n=26504975
```

[1]:
```
26504975
```

[2]:
```
primos = [2,3,5,7,11]
no_es_multiplo = true

while no_es_multiplo
    no_es_multiplo = false
        for p in primos
            if mod(n,p) == 0
                n += 1
                no_es_multiplo = true
            end
        end
end

print("El valor del n que no es múltiplo de $primos es n = $n")
```

```
El valor del n que no es múltiplo de [2, 3, 5, 7, 11] es n = 26504977
```

2- Calcula a^{n-1} mod n, para cada uno de esas 5 bases usando sucesivamente el algoritmo de izda-dcha y de dcha a izda.

En primer lugar usaremos el algoritmo de exponenciación rápida de derecha a izquierda para calcular a^{n-1}.

[3]:
```
function exponenciacion_rapida_dcha_izda(a, exp,m, print=false)
    base=a
    acu=1
    num=exp
```

```
    num_binario= string(num; base=2)
    contador=0

    println("Numero en binario=$num_binario")

    if print==true
        println("Iteración: $contador")
        println("acu: $acu")
        println("base: $base")
        println("")
    end

    for i in reverse(num_binario)
        if i!='0'
            acu=mod(acu*base,m)
        end

        base=mod(base*base,m)

        contador+=1

        if print==true
            println("Iteración: $contador")
            println("acu: $acu")
            println("base: $base")
            println("")
        end
    end

    println("El resultado es $acu")
end
```

[3]: exponenciacion_rapida_dcha_izda (generic function with 2 methods)

Vamos a probar con el ejemplo de la ayuda.

[4]:
```
exponenciacion_rapida_dcha_izda(2,27213646,27213647,true)
```

```
Numero en binario=1100111110011111101001110
Iteración: 0
acu: 1
base: 2

Iteración: 1
acu: 1
base: 4

Iteración: 2
```

```
acu: 4
base: 16

Iteración: 3
acu: 64
base: 256

Iteración: 4
acu: 16384
base: 65536

Iteración: 5
acu: 16384
base: 22424717

Iteración: 6
acu: 16384
base: 12167649

Iteración: 7
acu: 14796941
base: 26585339

Iteración: 8
acu: 14796941
base: 9779482

Iteración: 9
acu: 120175
base: 15067521

Iteración: 10
acu: 24905736
base: 12424177

Iteración: 11
acu: 9320126
base: 14159809

Iteración: 12
acu: 10785431
base: 15283

Iteración: 13
acu: 682094
base: 15860913

Iteración: 14
```

```
acu: 11508854
base: 2314287

Iteración: 15
acu: 11508854
base: 6452299

Iteración: 16
acu: 11508854
base: 17649979

Iteración: 17
acu: 15616437
base: 11515927

Iteración: 18
acu: 8921297
base: 9800221

Iteración: 19
acu: 10593740
base: 23701151

Iteración: 20
acu: 1995822
base: 21932449

Iteración: 21
acu: 4173990
base: 3214080

Iteración: 22
acu: 4173990
base: 9845200

Iteración: 23
acu: 4173990
base: 760573

Iteración: 24
acu: 16105485
base: 18007697

Iteración: 25
acu: 1
base: 13932984

El resultado es 1
```

Como podemos ver da el mismo resultado del ejemplo en cada iteración. Vamos a realizar ahora el apartado 2.

[5]:
```
for i in primos
    println("Probamos con la base $i")
    println("")
    exponenciacion_rapida_dcha_izda(i,n-1,n,false)
    println("")
end
```

```
Probamos con la base 2

Numero en binario=1100101000110111100010000
El resultado es 19574020

Probamos con la base 3

Numero en binario=1100101000110111100010000
El resultado es 12215788

Probamos con la base 5

Numero en binario=1100101000110111100010000
El resultado es 5558275

Probamos con la base 7

Numero en binario=1100101000110111100010000
El resultado es 14115151

Probamos con la base 11

Numero en binario=1100101000110111100010000
El resultado es 4065986
```

Ahora realizamos los mismos experimentos con la exponenciación rápida de izquierda a derecha.

[6]:
```
function exponenciacion_rapida_izda_dcha(a,exp,m,print=false)
    base=a
    acu=1
    num=exp
    num_binario= string(num; base=2)
    contador=0

    if print==true
        println("Iteración: $contador")
        println("acu: $acu")
```

```
        println("")
    end

    println(num_binario)
    for i in num_binario
        if i!='0'
            acu=mod(acu*acu*base,m)
        else
            acu=mod(acu*acu,m)
        end
        contador+=1

        if print==true
            println("Iteración: $contador")
            println("acu: $acu")
            println("")
        end

    end

    println("El resultado es: $acu")

end
```

[6]: exponenciacion_rapida_izda_dcha (generic function with 2 methods)

[7]:
```
exponenciacion_rapida_izda_dcha(2,27213646,27213647,true)
```

```
Iteración: 0
acu: 1

11001111100111111101001110
Iteración: 1
acu: 2

Iteración: 2
acu: 8

Iteración: 3
acu: 64

Iteración: 4
acu: 4096

Iteración: 5
acu: 6340785
```

```
Iteración: 6
acu: 6967674

Iteración: 7
acu: 2913255

Iteración: 8
acu: 5278505

Iteración: 9
acu: 8390032

Iteración: 10
acu: 21530122

Iteración: 11
acu: 20718507

Iteración: 12
acu: 6234754

Iteración: 13
acu: 14354021

Iteración: 14
acu: 18664779

Iteración: 15
acu: 21238793

Iteración: 16
acu: 9159785

Iteración: 17
acu: 1897988

Iteración: 18
acu: 6353813

Iteración: 19
acu: 22747524

Iteración: 20
acu: 12082479

Iteración: 21
acu: 22922644
```

```
Iteración: 22
acu: 15226206

Iteración: 23
acu: 8424067

Iteración: 24
acu: 1

Iteración: 25
acu: 1

El resultado es: 1
```

[8]:
```
for i in primos
    println("Probamos con la base $i")
    println("")
    exponenciacion_rapida_izda_dcha(i,n-1,n)
    println("")
end
```

```
Probamos con la base 2

11001010001101111100010000
El resultado es: 19574020

Probamos con la base 3

11001010001101111100010000
El resultado es: 12215788

Probamos con la base 5

11001010001101111100010000
El resultado es: 5558275

Probamos con la base 7

11001010001101111100010000
El resultado es: 14115151

Probamos con la base 11

11001010001101111100010000
El resultado es: 4065986
```

3- ¿Es un Posible primo de Fermat para alguna de ellas? ¿Es n un pseudoprimo para

alguna de ellas?

Como podemos ver el número n no es primo, pues para toda base probada a^{n-1} mod n era distinto de 1. Por lo tanto no puede ser un posible primo de Fermat ni un pseudoprimo para ninguna base.

1.1.1 Ejercicio 2

Dado tu número n=26505013 de ocho cifras de la lista publicada.

1- Usa el algoritmo Manual para calcular el símbolo de Jacobi $(\frac{p}{n})$, para p cada uno de los 5 primeros primos.

En primer lugar vamos a programar el algoritmo para poder comprobar que son correctos los resultados obtenidos a mano:

[9]:
```
function simbolo_jacobi(a,n)
    t=1
    m=abs(n)
    b=mod(a,m)

    while a != 0
        while mod(a,2)==0
            a=a/2
            if mod(m,8)==3 || mod(m,8)==5
                t=-t
            end
        end
        aux=a
        a=m
        m=aux
        if mod(a,4)==mod(m,4)==3
            t=-t
        end

        a=mod(a,m)

    end

    if m==1
        return t
    else
        return 0
    end


end
```

[9]:
```
simbolo_jacobi (generic function with 1 method)
```

Una vez programado el Algoritmo vamos a proceder a calcular a mano el símbolo de Jacobi para

nuestro n con los cinco primeros primos:

- Para $p = 2$:

  Como $n \equiv 5 \bmod 8$ tenemos que $(\frac{2}{n}) = -1$ aplicando la primera propiedad del símbolo de Jacobi.

- Para $p = 3$:

  En este caso tanto n como p son impares, luego deben ser congruentes a 1 o a 3 módulo 4.

  Como $n \equiv 1 \bmod 4 \Rightarrow (\frac{3}{n}) = (\frac{n}{3})$

  Por otro lado como $n \equiv 1 \bmod 3 \Rightarrow (\frac{3}{n}) = (\frac{1}{3}) = 1$ (ya que $2^2 \equiv 1 \bmod 3$).

- Para $p = 5$:

  Como son ambos impares, al ser $5 \equiv 1 \bmod 4$ se tiene que $(\frac{5}{n}) = (\frac{n}{5})$.

  Del mismo modo como $n \equiv 3 \bmod 5$ se tiene que $(\frac{n}{5}) = (\frac{3}{5})$.

  Como $5 \equiv 1 \bmod 4$ $(\frac{3}{5}) = (\frac{5}{3})$ y como $5 \equiv 2 \bmod 3$ se tiene que $(\frac{5}{3}) = (\frac{2}{3})$.

  Finalmente como $3 \equiv 3 \bmod 8$ se tiene que $(\frac{2}{3}) = -1$ por las primeras propiedades del símbolo de Jacobi.

- Para $p = 7$:

  Ambos son impares y como $n \equiv 1 \bmod 4$ $(\frac{7}{n}) = (\frac{n}{7})$, por otro lado $n \equiv 3 \bmod 7$, luego $(\frac{n}{7}) = (\frac{3}{7})$.

  En este caso ambos números son congruentes con 3 módulo 4, luego las propiedades del símbolo de Jacobi nos dicen que $(\frac{3}{7}) = -(\frac{7}{3}) = -(\frac{1}{3}) = (-1)(1) = -1$. Pues ya habíamos calculado antes el símbolo de Jacobi de $\frac{1}{3}$.

- Para $p = 11$:

  Como $n \equiv 1 \bmod 4 \Rightarrow (\frac{11}{n}) = (\frac{n}{11})$, a su vez $(\frac{n}{11}) = (\frac{7}{11})$ pues $n \equiv 7 \bmod 11$. Como tanto 7 como 11 son congruentes con 3 módulo 4 se tiene que $(\frac{7}{11}) = -(\frac{11}{7}) = -(\frac{4}{7}) = -1$ pues $2^2 \equiv 4 \bmod 11$.

Finalmente comprobemos que con el algoritmo implementado se obtienen los mismos resultados:

[10]:
```
for i in primos
    s=simbolo_jacobi(i,26505013)
    println("simbolo de Jacobi obtenido para $i / 26505013 = $s")
end
```

```
simbolo de Jacobi obtenido para 2 / 26505013 = -1
simbolo de Jacobi obtenido para 3 / 26505013 = 1
simbolo de Jacobi obtenido para 5 / 26505013 = -1
simbolo de Jacobi obtenido para 7 / 26505013 = -1
simbolo de Jacobi obtenido para 11 / 26505013 = -1
```

Como podemos apreciar los resultados coinciden.

2- Si para alguna de esas bases tu número sale posible primo de Fermat, comprueba si además es posible primo de Euler.

Vamos a utilizar el algoritmo de exponenciación rápida de izquierda a derecha implementado en el ejercicio anterior:

[11]:
```
for i in primos
    println("Probamos con la base $i")
    println("")
    exponenciacion_rapida_izda_dcha(i,26505012,26505013)
    println("")
end
```

```
Probamos con la base 2

11001010001101111100110100
El resultado es: 1

Probamos con la base 3

11001010001101111100110100
El resultado es: 1

Probamos con la base 5

11001010001101111100110100
El resultado es: 1

Probamos con la base 7

11001010001101111100110100
El resultado es: 1

Probamos con la base 11

11001010001101111100110100
El resultado es: 1
```

Como podemos ver, tras las 25 iteraciones nuestro número sale posible primo de Fermat para cada base escogida, veamos ahora si es posible primo de Euler.

Para ver si es posible primo de Euler debemos comprobar el test de SOLOVAY-STRASSEN de manera que si se cumple lo siguiente:

$$(\frac{a}{n}) \equiv a^{(n-1)/2} \; mod \; n$$

Se dirá que n es un posible primo de Euler para a, y en caso contrario se tendrá que n es compuesto.

Por un lado la parte izquierda de la equivalencia la hemos calculado ya, veamos la parte derecha.

De nuevo nos ayudaremos del algoritmo de exponenciación rápida.

A continuación probamos el algoritmo con los valores del ejercicio

[12]:
```
m=13252506
n=26505013
println("El exponente es: $m")

for i in primos
    println("Probamos con la base $i")
    println("")
    exponenciacion_rapida_izda_dcha(i,m,n)
    println("")
end
```

```
El exponente es: 13252506
Probamos con la base 2

11001010001011111110011010
El resultado es: 26505012

Probamos con la base 3

11001010001011111110011010
El resultado es: 1

Probamos con la base 5

11001010001011111110011010
El resultado es: 26505012

Probamos con la base 7

11001010001011111110011010
El resultado es: 26505012

Probamos con la base 11

11001010001011111110011010
El resultado es: 26505012
```

Como podemos observar coinciden con el valor del símbolo de Jacobi para cada primo, luego podemos decir que el número n es un posible primo de Euler para 2,3,5,7 y 11.

Como curiosidad, he estado investigando y existe una forma más eficiente de realizar la exponenciación rápida ayudándose de las operaciones lógicas de Julia que operan bit a bit y nos podemos evitar convertir el número a binario de forma explícita pues Julia trabajaría con el exponente en

binario con estas operaciones, de esta forma usaremos el and a nivel de bits "&" con 1 para identificar si el valor actual del exponente es par (devuelve 0) o impar (devuelve 1) y en cada iteración desplazaremos a la derecha el valor del exponente a nivel de bits, esto es, por ejemplo en la iteración i-ésima el valor en binario del exponente es 1101, en la iteración siguiente será 110.

De esta forma el algoritmo quedaría de la siguiente manera:

[13]:
```
function exponenciacion_rapida_modificado(base,exp,m)
    acu=1

    while exp>0
        if (exp&1)>0
            acu=mod(acu*base,m)
        end

        exp>>=1
        base=mod(base*base,m)

    end

    println("El resultado es: $acu")
end
```

[13]:
```
exponenciacion_rapida_modificado (generic function with 1 method)
```

Como podemos observar se producen los mismos resultados para el ejercicio:

[14]:
```
m=13252506
n=26505013
println("El exponente es: $m")

for i in primos
    println("Probamos con la base $i")
    exponenciacion_rapida_modificado(i,m,n)
    println("")
end
```

```
El exponente es: 13252506
Probamos con la base 2
El resultado es: 26505012

Probamos con la base 3
El resultado es: 1

Probamos con la base 5
El resultado es: 26505012

Probamos con la base 7
El resultado es: 26505012
```

```
Probamos con la base 11
El resultado es: 26505012
```

3- ¿Es tu número n pseudoprimo de Fermat o de Euler para alguna de las bases?

En nuestro caso sería un pseudoprimo de Euler y de Fermat para todas las bases probadas, lo que nos permite concluir que nuestro n sería primo con cierta probabilidad.