EjerciciosSemana2

March 5, 2022

1 Ejercicio 3

Dado tu número m (de 30 cifras o mas) de la lista publicada:

1. **Calcula a^{m-1} mod m para los 5 primeros primos.**
2. **Calcula el test de Solovay-Strassen para los 5 primeros primos.**
3. **Calcula el test de Miller-Rabin para esas 5 bases.**
4. **¿Qué deduces sobre la primalidad de tu número?**

[1]:
```
m=493525498337311292187187890733
```

Vamos a volver a utilzar el algoritmo de exponenciación rápida por lo que lo recuperamos de los ejercicios anteriores.

[2]:
```
def exponenciacion_rapida_izda_dcha (a,exp,m,imprime=False):
    c=0
    num_binario= bin(exp)[2:]
    acu=1
    for i in num_binario:
        c=2*c
        acu=(acu*acu)%m

        if i=='1':
            c+=1
            acu=(acu*a)%m

    return acu
```

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

for i in primos:
    print("Probamos con la base",i)
    resultado=exponenciacion_rapida_izda_dcha(i,m-1,m)
    print(resultado)
    print("")
```

```
Probamos con la base 2
1
```

```
Probamos con la base 3
1

Probamos con la base 5
1

Probamos con la base 7
1

Probamos con la base 11
1
```

Como podemos ver nuestro número es un posible primo de Fermat para todas las bases probadas.

Ahora vamos a comprobar si para estos mismos números se verifica el Test de Solovay-Strassen, para ello vamos a recuperar el algoritmo para calcular el símbolo de Jacobi.

[4]:
```python
def simbolo_jacobi(a,n):
    t=1
    m=abs(n)
    b=a%m

    while a != 0:
        while a%2==0:
            a=a/2
            if m%8==3 or m%8==5:
                t=-t

        aux=a
        a=m
        m=aux
        if a%4==m%4==3:
            t=-t

        a=a%m

    if m==1:
        return t
    else:
        return 0
```

Recordemos que El Test de Solovay-Strassen es el 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.

Calculemos primero el símbolo de Jacobi con los 5 primeros primos:

[5]:
```
for i in primos:
    s=simbolo_jacobi(i,m)
    print("simbolo de Jacobi obtenido para", i,"/ m = ",s)
```

```
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
```

Por otro lado calculamos la parte derecha de la igualdad:

[6]:
```
exp=(m-1)//2

print("El exponente es:",exp)
print("")

for i in primos:
    print("Probamos con la base:", i)
    resultado=exponenciacion_rapida_izda_dcha(i,exp,m)
    print(resultado)
    print("")
```

```
El exponente es: 24676274916865564609359394536 6

Probamos con la base: 2
49352549833731129218718789 0732

Probamos con la base: 3
1

Probamos con la base: 5
49352549833731129218718789 0732

Probamos con la base: 7
49352549833731129218718789 0732

Probamos con la base: 11
1
```

Como podemos comprobar pasa el test de Solovay-Strassen para todos los primos probados.

A continuación vamos a comprobar el test de **Millner-Rabin**, el cual se basa en que dado que $n-1=2^r m$ con m impar, para una base a, se calcula la siguiente a-sucesión para n:

$$a^m,\ a^{2m},\ \ldots a^{2^r m} = a^{n-1}$$

Si esta sucesión no acaba en 1 o bien hay un 1 precedido de un número que no es +-1, se tendría que n es compuesto.

Por lo tanto en primer lugar calculamos los exponentes que necesitaremos para Millner-Rabin:

[7]:
```
def obtener_exponentes_millner_rabin(n):
    valor=n-1
    resultado=[valor]
    while valor%2==0:
        valor=valor//2
        resultado.append(valor)

    return resultado
```

[8]:
```
exp=obtener_exponentes_millner_rabin(m)
print(exp)
```

```
[4935254983373112921871878907 32, 24676274916865564609359394536 6,
12338137458432782304679697268 3]
```

En segundo lugar pasamos a probar el test con los 5 primeros primos como bases.

[9]:
```
for j in primos:
    for i in reversed(exp):
        print("Probamos con la base",j)
        resultado=exponenciacion_rapida_izda_dcha(j,i,m)
        print(resultado)
        print("")
```

```
Probamos con la base 2
48855204116439375377362441635 5

Probamos con la base 2
4935254983373112921871878907 32

Probamos con la base 2
1

Probamos con la base 3
1

Probamos con la base 3
1

Probamos con la base 3
1
```

```
Probamos con la base 5
4973457172917538413563474378

Probamos con la base 5
4935254983373112921871878907 32

Probamos con la base 5
1

Probamos con la base 7
4973457172917538413563474378

Probamos con la base 7
4935254983373112921871878907 32

Probamos con la base 7
1

Probamos con la base 11
4935254983373112921871878907 32

Probamos con la base 11
1

Probamos con la base 11
1
```

Como todas las sucesiones acaban en 1 para todas las bases tenemos que nuestro número sería un pseudoprimo fuerte para estas 5 bases probadas, y por lo tanto contestando a la última pregunta del ejercicio podemos concluir que es muy probable que nuestro número sea primo pues la probabilidad de que lo sea es de $1 - 4^{-5} = 0.9990234375$.

Esta probabilidad anterior se calcula teniendo en cuenta que el número de mentirosos fuertes no puede ser nunca mayor que $\frac{n-1}{4}$ y que la probabilidad de que un n compuesto pase el test para m bases elegidas es de 4^{-m}, luego la probabilidad de que sea primo es la complementaria.

2 Ejercicio 4

Dado tu número n de 8 cifras del ejercicio 2:

1. Factoriza n-1 aplicando el método ρ de Polard. ¿Cuántas iteraciones necesitas?

2. Si es necesario aplica recursivamente Lucas-Lehmer para certificar factores primos de n-1 mayores de 4 cifras.

3. Aplica Lucas-Lehmer para encontrar un certificado de primalidad de n.

NOTA: Debes encontrar el natural más pequeño cuya clase sea primitiva.

En primer lugar vamos a implementar el método ρ de polard:

[10]:
```
from math import gcd

n=26505013
```

[11]:
```
def f(x):
    return x*x+1
```

[12]:
```
def rho_de_polard(n,imprime=False):
    x=1
    y=1
    contador=0
    resultado=1

    if imprime:
        print("Iteracion ", contador)
        print("x: ",x," y:",y , " mcd: ", resultado)

    while resultado==1 or resultado == n:
        x=f(x)%n
        y=f(f(y))%n
        resultado=gcd(x-y,n)
        contador+=1

        if imprime:
            print("Iteracion ", contador)
            print("x: ",x," y:",y , " mcd: ", resultado)
        if 1<resultado<n:
            return resultado

    return "No hay divisores"
```

[13]:
```
primos_4_cifras=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,␣
 ↪59, 61, 67, 71, 73, 79,
83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163,
167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251,
257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443,
449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557,
563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643,
647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743,
751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853,
857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953,
```

```
967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049,
1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129,
1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229, 1231,
1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319,
1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409, 1423, 1427, 1429, 1433, 1439,
1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 1523,
1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609,
1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 1669, 1693, 1697, 1699, 1709,
1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811,
1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913,
1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999, 2003, 2011, 2017,
2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083, 2087, 2089, 2099, 2111, 2113,
2129, 2131, 2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237,
2239, 2243, 2251, 2267, 2269, 2273, 2281, 2287, 2293, 2297, 2309, 2311, 2333,
2339, 2341, 2347, 2351, 2357, 2371, 2377, 2381, 2383, 2389, 2393, 2399, 2411,
2417, 2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539,
2543, 2549, 2551, 2557, 2579, 2591, 2593, 2609, 2617, 2621, 2633, 2647, 2657,
2659, 2663, 2671, 2677, 2683, 2687, 2689, 2693, 2699, 2707, 2711, 2713, 2719,
2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803, 2819,
2833, 2837, 2843, 2851, 2857, 2861, 2879, 2887, 2897, 2903, 2909, 2917, 2927,
2939, 2953, 2957, 2963, 2969, 2971, 2999, 3001, 3011, 3019, 3023, 3037, 3041,
3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169,
3181, 3187, 3191, 3203, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271,
3299, 3301, 3307, 3313, 3319, 3323, 3329, 3331, 3343, 3347, 3359, 3361, 3371,
3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463, 3467, 3469, 3491,
3499, 3511, 3517, 3527, 3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581,
3583, 3593, 3607, 3613, 3617, 3623, 3631, 3637, 3643, 3659, 3671, 3673, 3677,
3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761, 3767, 3769, 3779, 3793,
3797, 3803, 3821, 3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907,
3911, 3917, 3919, 3923, 3929, 3931, 3943, 3947, 3967, 3989, 4001, 4003, 4007,
4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073, 4079, 4091, 4093, 4099, 4111,
4127, 4129, 4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229,
4231, 4241, 4243, 4253, 4259, 4261, 4271, 4273, 4283, 4289, 4297, 4327, 4337,
4339, 4349, 4357, 4363, 4373, 4391, 4397, 4409, 4421, 4423, 4441, 4447, 4451,
4457, 4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561,
4567, 4583, 4591, 4597, 4603, 4621, 4637, 4639, 4643, 4649, 4651, 4657, 4663,
4673, 4679, 4691, 4703, 4721, 4723, 4729, 4733, 4751, 4759, 4783, 4787, 4789,
4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909, 4919,
4931, 4933, 4937, 4943, 4951, 4957, 4967, 4969, 4973, 4987, 4993, 4999, 5003,
5009, 5011, 5021, 5023, 5039, 5051, 5059, 5077, 5081, 5087, 5099, 5101, 5107,
5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227, 5231, 5233,
5237, 5261, 5273, 5279, 5281, 5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381,
5387, 5393, 5399, 5407, 5413, 5417, 5419, 5431, 5437, 5441, 5443, 5449, 5471,
5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531, 5557, 5563, 5569,
5573, 5581, 5591, 5623, 5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683,
5689, 5693, 5701, 5711, 5717, 5737, 5741, 5743, 5749, 5779, 5783, 5791, 5801,
5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857, 5861, 5867, 5869, 5879,
```

```
5881, 5897, 5903, 5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037,
6043, 6047, 6053, 6067, 6073, 6079, 6089, 6091, 6101, 6113, 6121, 6131, 6133,
6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211, 6217, 6221, 6229, 6247, 6257,
6263, 6269, 6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343,
6353, 6359, 6361, 6367, 6373, 6379, 6389, 6397, 6421, 6427, 6449, 6451, 6469,
6473, 6481, 6491, 6521, 6529, 6547, 6551, 6553, 6563, 6569, 6571, 6577, 6581,
6599, 6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703,
6709, 6719, 6733, 6737, 6761, 6763, 6779, 6781, 6791, 6793, 6803, 6823, 6827,
6829, 6833, 6841, 6857, 6863, 6869, 6871, 6883, 6899, 6907, 6911, 6917, 6947,
6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019, 7027,
7039, 7043, 7057, 7069, 7079, 7103, 7109, 7121, 7127, 7129, 7151, 7159, 7177,
7187, 7193, 7207, 7211, 7213, 7219, 7229, 7237, 7243, 7247, 7253, 7283, 7297,
7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417, 7433, 7451,
7457, 7459, 7477, 7481, 7487, 7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541,
7547, 7549, 7559, 7561, 7573, 7577, 7583, 7589, 7591, 7603, 7607, 7621, 7639,
7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717, 7723, 7727, 7741,
7753, 7757, 7759, 7789, 7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877,
7879, 7883, 7901, 7907, 7919, 7927, 7933, 7937, 7949, 7951, 7963, 7993, 8009,
8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089, 8093, 8101, 8111, 8117,
8123, 8147, 8161, 8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237,
8243, 8263, 8269, 8273, 8287, 8291, 8293, 8297, 8311, 8317, 8329, 8353, 8363,
8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431, 8443, 8447, 8461, 8467, 8501,
8513, 8521, 8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623,
8627, 8629, 8641, 8647, 8663, 8669, 8677, 8681, 8689, 8693, 8699, 8707, 8713,
8719, 8731, 8737, 8741, 8747, 8753, 8761, 8779, 8783, 8803, 8807, 8819, 8821,
8831, 8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941,
8951, 8963, 8969, 8971, 8999, 9001, 9007, 9011, 9013, 9029, 9041, 9043, 9049,
9059, 9067, 9091, 9103, 9109, 9127, 9133, 9137, 9151, 9157, 9161, 9173, 9181,
9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283, 9293,
9311, 9319, 9323, 9337, 9341, 9343, 9349, 9371, 9377, 9391, 9397, 9403, 9413,
9419, 9421, 9431, 9433, 9437, 9439, 9461, 9463, 9467, 9473, 9479, 9491, 9497,
9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623, 9629, 9631,
9643, 9649, 9661, 9677, 9679, 9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749,
9767, 9769, 9781, 9787, 9791, 9803, 9811, 9817, 9829, 9833, 9839, 9851, 9857,
9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941, 9949, 9967, 9973]
```

[14]:
```python
def factores_primos(n):
    fact_primos=[]
    finaliza=False
    x=0
    y=0
    while finaliza==False:
        x=rho_de_polard(n)
        print("El factor encontrado es: ",x)
        if x not in fact_primos:
            fact_primos.append(x)
        y=n//x
```

```
        print("--> Vamos a ver si ",y," es primo")
        if es_primo(y,primos_4_cifras):
            finaliza=True

        n=y

    return fact_primos
```

[15]:
```
def es_primo(n, primos):
    print("Analizando si ",n," es primo")
    primo=False

    if n<10000:
        print("El posible primo que estudiamos es ",n)
        primo= n in primos
        print(primo)
        return primo
    else:
        print("---->Usamos Lucas_Lehmer para ver si ", n , " es primo")
        return Lucas_Lehmer(n)
```

Como se nos dice en el primer apartado vamos a ejecutar el método ρ de polard sobre $n-1$. Esto podemos hacerlo con garantías de que $n-1$ es compuesto porque es par y por tanto el método acabará en un número finito de pasos.

[16]:
```
rho_de_polard(n-1,True)
```

```
Iteracion  0
x:  1  y: 1  mcd:  1
Iteracion  1
x:  2  y: 5  mcd:  3
```

[16]: 3

[17]:
```
rho_de_polard((n-1)//3,True)
```

```
Iteracion  0
x:  1  y: 1  mcd:  1
Iteracion  1
x:  2  y: 5  mcd:  1
Iteracion  2
x:  5  y: 677  mcd:  4
```

[17]: 4

Tenemos que $n-1 = 26505012$ y como es divisible entre 2 se tiene que $n-1 = 2*2*6626253$. Por otro lado 6626253 es divisible entre 3 por lo que $n-1 = 2*2*3*2208751$, luego nuestro problema

es saber si 2208751 es un número primo o no. Primero veamos si pasa el test de Fermat con los primeros 5 primos.

Como podemos ver, contestando a la primera pregunta, necesitamos muy pocas iteraciones con el método de Polard para romper el número, en el primer caso basta con 2 y en el segundo con 3 iteraciones.

[18]:
```
primos=[2,3,5,7,11]

for i in primos:
    r=exponenciacion_rapida_izda_dcha(i,n-1,n)
    print(r)
```

```
1
1
1
1
1
```

Como vemos es un posible primo para las 5 primeras bases, por lo que no es recomendable aplicar el algoritmo ρ de Polard de primeras pues de ser primo no acabaría nunca. Vamos antes a calcular la probabilidad de que nuestro nuevo número sea primo usando **Millner-Rabin**.

[19]:
```
exp=obtener_exponentes_millner_rabin(2208751)
print(exp)
```

```
[2208750, 1104375]
```

[20]:
```
for j in primos:
    for i in reversed(exp):
        print("Probamos con la base",j)
        resultado=exponenciacion_rapida_izda_dcha(j,i,2208751)
        print(resultado)
        print("")
```

```
Probamos con la base 2
1

Probamos con la base 2
1

Probamos con la base 3
2208750

Probamos con la base 3
1

Probamos con la base 5
1
```

```
Probamos con la base 5
1

Probamos con la base 7
1

Probamos con la base 7
1

Probamos con la base 11
1

Probamos con la base 11
1
```

Como vemos pasa el Test de Millner-Rabin para todas las bases, luego la probabilidad de que sea primo es de $1 - 4^{-5} = 0.9990234375$. Por ello vamos a intentar demostrar que es primo usando **Lucas-Lehmer**.

Por ello necesitamos Calcular los divisores primos de p-1 (siendo p=2208751).

[21]:
```
p=2208751
rho_de_polard(p-1)
```

[21]:
```
3
```

[22]:
```
(p-1)/3
```

[22]:
```
736250.0
```

Luego $p - 1 = 3 * 736250$, y volvemos a aplicar ρ de Polard porque 736250 es compuesto al ser par.

[23]:
```
rho_de_polard(736250)
```

[23]:
```
2
```

Obtenemos así que $p - 1 = 2 * 3 * 368125$.

A simple vista podemos ver que 368125 es compuesto pues es divisible por 5, luego volvemos a probar con el método de Polard.

[24]:
```
rho_de_polard(368125,True)
```

```
Iteracion  0
x:  1  y: 1  mcd:  1
Iteracion  1
x:  2  y: 5  mcd:  1
Iteracion  2
```

```
x:  5  y: 677  mcd:  1
Iteracion  3
x:  26  y: 275151  mcd:  3875
```

[24]: 3875

Obtenemos que $p-1 = 2*3*3875 * 95.0 = 2*3*5*19*3875$, todos los factores son primos salvo 3875, luego seguimos aplicando el método:

[25]:
```
rho_de_polard(3875,True)
```

```
Iteracion  0
x:  1  y: 1  mcd:  1
Iteracion  1
x:  2  y: 5  mcd:  1
Iteracion  2
x:  5  y: 677  mcd:  1
Iteracion  3
x:  26  y: 26  mcd:  3875
Iteracion  4
x:  677  y: 1080  mcd:  31
```

[25]: 31

Obtenemos que $p-1 = 2*3*3875*95.0 = 2*3*5*19*31*125 = 2*3*5*19*31*125 = 2*3*5^4*19*31$ luego ya tenemos todos los divisores primos de 2208750. A continuación probamos el Test de **Lucas-Lehmer**.

[26]:
```
def comprobacion_lucas_lehmer(a,divisores,n):
    resultado=True

    for i in divisores:
        if exponenciacion_rapida_izda_dcha(a,(n-1)//i,n) == 1:
            resultado=False

    return resultado
```

[27]:
```
def Lucas_Lehmer(n,divisores):
    i=1
    test1=False
    test2=False
    res=0

    while i>0:
        i+=1
        res=exponenciacion_rapida_izda_dcha (i,n-1,n)
```

```
        if res==1:
            test1=True

        if comprobacion_lucas_lehmer(i,divisores,n):
            test2=True

        if test1 & test2:
            print("El natural más pequeño cuya clase es primitiva: ", i)
            return True
        else:
            test1=False
            test2=False
```

[28]:
```
divisores=[2,3,5,19,31]
if Lucas_Lehmer(2208751,divisores):
    print("Es primo")
```

```
El natural más pequeño cuya clase es primitiva:  3
Es primo
```

Por lo tanto la descomposición en factores primos del número $n-1$ asignado es $n-1 = 2^2 * 3 * 2208751$

Finalmente, para el último apartado del ejercicio vamos a buscar con el método de **Lucas-Lehmer** un certificado de primalidad del número $n = 26505013$ asignado.

[29]:
```
divisores=[2,3,2208751]

if Lucas_Lehmer(n,divisores):
    print("Es primo")
```

```
El natural más pequeño cuya clase es primitiva:  5
Es primo
```

Como podemos ver sería el 5 pues:

$5^{n-1}\ mod\ n \equiv 1$

[30]:
```
exponenciacion_rapida_izda_dcha(5,n-1,n)
```

[30]: 1

Y por otro lado:

$5^{\frac{n-1}{2}}\ mod\ n \equiv 26505012 \neq 1$

$5^{\frac{n-1}{3}}\ mod\ n \equiv 17175518 \neq 1$

$5^{\frac{n-1}{2208751}}\ mod\ n \equiv 5595508 \neq 1$

[31]:
```
for i in divisores:
    print(exponenciacion_rapida_izda_dcha(5,(n-1)//i,n))
```

```
26505012
17175518
5595508
```

Que son todos distintos de 1, luego por el Teorema de Lucas-Lehmer 26505013 es primo y podríamos usar el 2 como certificado de primalidad.