TP-2024-03-04

March 4, 2024

[179]:
```
a=123456789
a^10
```

[179]:
```
8225262591471025795047611436615355477641378922955141680937016996764162077997366
01
```

[180]:
```
100.factorial()
```

[180]:
```
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000
```

[181]:
```
p=next_prime(100);p
```

[181]:
```
101
```

[182]:
```
a
```

[182]:
```
123456789
```

[183]:
```
a^p-a
```

[183]:
```
174989783605563173243440446299561186459186612335579908529280871249557869892184078705505922376106132177072247153007403146691610365728216304864868752326467482393879560410045821252643772005630892226196896539023698221693080304619967556969246349154878795457520881378126191493106222698390921771774936043515220039538023854965287553783589999952514493567837677397827530695070937182386368319743145702224013326487774680575400966901504572120438845928131976726788153459243112941734063511088680971330976142429581006838491526049252841824966532452468629876070463818595812674542904489304403932407297760071183329430521109356968785790752669484772638158352178073261431075183885881993156990990751232920902846862191119792465326795937771160818253457760907224067340472100518777923607618123070809785600740768718553547299197022782424106575574000
```

[184]:
```
(a^p-a)%p
```

[184]:
```
0
```

[185]:
```
Set([(a^p-a)%p for a in range(1000)])
```

[185]:
```
{0}
```

[186]:
```
(a%p)^p-a # voir la différence avec 3 lignes au-dessus
```

[186]:
```
94289621266793848694096320012199761041022814726349889332543715058143870788665105
41984630554864971721886937021055506058873744729825761012875773303676396608352648
7871336
```

[187]:
```
# ON définit l'anneau Z/pZ
Zp=IntegerModRing(p)
Zp
```

[187]:
```
Ring of integers modulo 101
```

[188]:
```
a,Zp(a),a%p
```

[188]:
```
(123456789, 45, 45)
```

[189]:
```
A=Zp(a);B=a%p;
A.parent(),B.parent()
```

[189]:
```
(Ring of integers modulo 101, Integer Ring)
```

[190]:
```
A,B,A^5,B^5
```

[190]:
```
(45, 45, 14, 184528125)
```

[191]:
```
Zp.unit_group()
```

[191]:
```
Multiplicative Abelian group isomorphic to C100
```

[192]:
```
Set([a.multiplicative_order() for a in Zp.unit_group()])
```

[192]:
```
{1, 2, 100, 5, 4, 10, 50, 20, 25}
```

[193]:
```
ld=(p-1).divisors();ld
```

[193]:
```
[1, 2, 4, 5, 10, 20, 25, 50, 100]
```

[194]:
```
[[d,euler_phi(d),len([u for u in Zp if u!=0 and u.multiplicative_order()==d])]␣
↪for d in ld]
```

[194]:
```
[[1, 1, 1],
 [2, 1, 1],
 [4, 2, 2],
 [5, 4, 4],
 [10, 4, 4],
```

```
 [20, 8, 8],
 [25, 20, 20],
 [50, 20, 20],
 [100, 40, 40]]
```

[195]:
```
euler_phi(100)
```

[195]:
```
40
```

0.0.1 Question 2. Ordre de 2 dans Z/nZ

[196]:
```
[n for n in range(3,51)]
```

[196]:
```
[3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
 18,
 19,
 20,
 21,
 22,
 23,
 24,
 25,
 26,
 27,
 28,
 29,
 30,
 31,
 32,
 33,
 34,
 35,
 36,
```

```
 37,
 38,
 39,
 40,
 41,
 42,
 43,
 44,
 45,
 46,
 47,
 48,
 49,
 50]
```

[197]:
```
n=25;# ordre de 2 modulo 25 ?
IntegerModRing(n)(2).multiplicative_order()
```

[197]: 20

[198]:
```
[2^i%n for i in range(21)]
```

[198]: [1, 2, 4, 8, 16, 7, 14, 3, 6, 12, 24, 23, 21, 17, 9, 18, 11, 22, 19, 13, 1]

[199]:
```
[IntegerModRing(2*n+1)(2).multiplicative_order() for n in range(1,24)]
```

[199]:
```
[2,
 4,
 3,
 6,
 10,
 12,
 4,
 8,
 18,
 6,
 11,
 20,
 18,
 28,
 5,
 10,
 12,
 36,
 12,
 20,
 14,
```

```
12,
23]
```

1 Question : quand est-ce que 2 est un générateur de $(\mathbf{Z}/n\mathbf{Z})^*$

[200]:
```
[2*n+1 for n in range(1,50) if IntegerModRing(2*n+1)(2).
↪multiplicative_order()==euler_phi(2*n+1)]
```

[200]:
```
[3, 5, 9, 11, 13, 19, 25, 27, 29, 37, 53, 59, 61, 67, 81, 83]
```

1.1 Exercice 2

[201]:
```
p=31
(p-1).factor()
```

[201]:
```
2 * 3 * 5
```

[202]:
```
(p-1).prime_factors(),(p-1).prime_divisors()
```

[202]:
```
([2, 3, 5], [2, 3, 5])
```

[203]:
```
lp=(p-1).prime_divisors();lp
```

[203]:
```
[2, 3, 5]
```

[204]:
```
p=31
Zp=IntegerModRing(p)
n=(p-1)
lp=n.prime_divisors();lp
a=Zp(2)
[a^((p-1)/d) for d in lp]
```

[204]:
```
[1, 1, 2]
```

[205]:
```
a=Zp(3)
[a^(n/d) for d in n.prime_divisors()]
```

[205]:
```
[30, 25, 16]
```

[206]:
```
product([x-1 for x in _])
```

[206]:
```
24
```

[207]:
```
def test(a,p):
    Zp=IntegerModRing(p)
```

```
    return(product([Zp(a)^(n/d)-1 for d in n.prime_divisors()])!=0 and
 ↪Zp(a)^(p-1)==1)
# defaut : recalcule Z/pZ et les diverus de p-1 à chaque étape.
```

[208]:
```
test(3,p),test(2,p)
```

[208]: (True, False)

[209]:
```
[a for a in Zp if a!=0 and a.multiplicative_order()==p-1]
```

[209]: [3, 11, 12, 13, 17, 21, 22, 24]

[211]:
```
[a for a in Zp if a!=0 and test(a,p)] # ce sont les génarateurs de $(Z/pZ)^*$
```

[211]: [3, 11, 12, 13, 17, 21, 22, 24]

[212]:
```
p=next_prime(10^20)
Zp=IntegerModRing(p)
n=(p-1)
lp=n.prime_divisors();print(p,lp)
a=Zp(2)
[a^((p-1)/d) for d in lp]
```

```
100000000000000000039 [2, 3, 32839, 507526619771207]
```

[212]: [1, 80592505861524411514, 33352432661614321066, 51125734533859006629]

[213]:
```
test(2,p),test(3,p)
```

[213]: (False, True)

[214]:
```
a=Zp.random_element();a,test(a,p)
```

[214]: (87687877344214321730, False)

[215]:
```
# calcul de la probabilité de trouver un génarateur au hasard
N=10000
c=0
for i in range(N):
    a=Zp.random_element()
    if test(a,p):
        c=c+1
(c/N).n()
```

[215]: 0.335800000000000

Cette probalité vaut $\varphi(p-1)/(p-1)$

[217]:
```
(euler_phi(p-1)/(p-1)).n()
```

[217]:
```
0.333323182800937
```

[218]:
```
p=2^127-1;
n=p-1;
lp=n.prime_divisors()
Zp=IntegerModRing(p)
lp
```

[218]:
```
[2, 3, 7, 19, 43, 73, 127, 337, 5419, 92737, 649657, 77158673929]
```

[219]:
```
a=Zp(2)
[a^(n/d) for d in lp],p,n,lp
```

[219]:
```
([1, 1, 1, 1, 1, 1, 68719476736, 1, 1, 1, 1, 1],
 170141183460469231731687303715884105727,
 170141183460469231731687303715884105726,
 [2, 3, 7, 19, 43, 73, 127, 337, 5419, 92737, 649657, 77158673929])
```

[220]:
```
p=2^127-1
n=p-1
lp=n.prime_divisors()
Zp=IntegerModRing(p)
N=10000
c=0
for i in range(N):
    a=Zp.random_element()
    #print(a)
    if product([a^(n/d)-1 for d in lp])!=0 and a^(n)==1:
        c=c+1
(c/N).n()
```

[220]:
```
0.259200000000000
```

[221]:
```
p.factor()
```

[221]:
```
170141183460469231731687303715884105727
```

[222]:
```
(euler_phi(n)/n).n()
```

[222]:
```
0.257888531813542
```

[224]:
```
# on s'arrête lorsqu'on a un générateur.
p=2^127-1
n=p-1
lp=n.prime_divisors()
```

```
Zp=IntegerModRing(p)
N=100
c=0
for i in range(N):
    a=Zp.random_element()
    c=c+1
    #print(a)
    if product([a^(n/d)-1 for d in lp])!=0 and a^(n)==1:
        break
c
```

[224]: 1

[228]:
```
# calcul du nombre moyen de coups  avant de trouver un générateur
NN=1000
c=0
for i in range(NN):
    for j in range(N):
        a=Zp.random_element()
        c=c+1
    #print(a)
        if product([a^(n/d)-1 for d in lp])!=0 and a^(n)==1:
            break
(c/NN).n()
```

[228]: 3.87400000000000

A comparer avec $(p-1)/\varphi(p-1)$

[227]:
```
(n/euler_phi(n)).n()
```

[227]: 3.87764431775128

1.2 Exercice 8.4

[143]:
```
p=37;q=41;n=p*q;phin=(p-1)*(q-1);
Zn=IntegerModRing(n)
for i in range(10):
    e=ZZ(randint(1,phin-1))
    if e.gcd(phin)==1:
        break

d=IntegerModRing(phin)(e)^(-1)
```

[144]:
```
e,d
```

[144]: (791, 71)

[146]:
```
x=Zn.random_element()
y=x^e
z=y^d
z-x
```

[146]: 0

[157]:
```
N=10;
p=next_prime(10^N);q=next_prime(p+1000);n=p*q;
p,q
```

[157]: (10000000019, 10000001041)

[158]:
```
m=sqrt(n).n().floor()
```

[162]:
```
m+1,(n-(m+1)^2).factor()
```

[162]: (10000000530, -1 * 7^2 * 73^2)

[173]:
```
N=35
p=next_prime(randint(10^N,10^(N+1)))
q=next_prime(randint(10^N,10^(N+1)))
n=p*q
print(n,p,q)
```

```
269579992865524374107630041875266048292639682392654922603782661026232391
566554663934344972916179331319459967 475823446573488229865722733785404473
```

[174]:
```
%time n.factor()
```

```
CPU times: user 1min 1s, sys: 563 ms, total: 1min 2s
Wall time: 1min 1s
```

[174]: 475823446573488229865722733785404473 * 566554663934344972916179331319459967

[175]:
```
#p=37;q=41;n=p*q;
phin=(p-1)*(q-1);
Zn=IntegerModRing(n)
for i in range(10):
    e=ZZ(randint(1,phin-1))
    if e.gcd(phin)==1:
        break

d=IntegerModRing(phin)(e)^(-1)
```

[176]:
```
e,d
```

[176]:
```
(151123694052989058957197235634339648209548370025140485998217618978836057,
 113593471454075285182526405213651683442522309311376977395122021488163929)
```

[177]:
```
x=Zn.random_element()
y=x^e
z=y^d
z-x
```

[177]:
```
0
```

[ ]:
```

```