**PARTE 1**

¿Cuál es el orden del grupo U(Z114)?

Toni:

#existe un comando para conseguir el orden de un grupo llamado G.order(), pero también podemos usar que el orden de un grupo cíclico es la función fi de euler

print("Z\_144 tiene orden " + str(euler\_phi(114)))

Tempra:

#sabemos que el orden del grupo de unidades de Z\_n es el número de coprimos con n menores que él, osea la función phi de Euler

print("Z\_144 tiene orden " + str(euler\_phi(114)))

¿Pertenece la clase [a]114[a]114 al grupo U(Z114)U(Z114)? ¿ Y  [c]114[c]114?

#[a]\_n pertenecerá a las unidades de Z\_m si y sólo si n y m son coprimos

if gcd(114,a)==1:

print("[a]\_n pertenece")

else:

print("[a]\_n no pertenece")

if gcd(114,c)==1:

print("[c]\_n pertenece")

else:

print("[c]\_n no pertenece")

Calcula el inverso multiplicativo de [c]114[c]114 en U(Z114)U(Z114) (usando Bezout con el comando xgcd)

#recurrimos a la identidad de Bézout para obtener el inverso

print(xgcd(114,c))[2]

A continuación puedes encontrar una función que calcula el orden multiplicativo de [n]m[n]m en U(Zm)U(Zm). Calcula el orden multiplicativo de la clase [c]114[c]114 en U(Z114)U(Z114).

def omultenZ\_m(n,m):

A=Integers(m)

n=A(n)

o=euler\_phi(m)

for i in range(1,m+1):

if n^i==1:

return i

print(omultenZ\_m(c,114))

Calcula el orden multiplicativo de la clase de [c]114[c]114 en Z114Z114.

Calcula los coeficientes de torsión de U(Z114)U(Z114).

En la variable **p**tiene asignada un cierto número primo. Encontrar cinco generadores distintos del grupo cíclico U(Zp)=Zp∖{[0]}U(Zp)=Zp∖{[0]}.

#U(Z\_p) = Z\_p-1

def gensdeUdeZp(p):

lst=[]

k=1

while len(lst)<5 and k<p-1:

if gcd(k,p-1)==1:

lst.append(k)

k= k+1

return lst

Si [x]p[x]p es uno de los generadores que has encontrado en la respuesta anterior, comprueba si [x]p2[x]p2 o [x+p]p2[x+p]p2 es un generador de U(Zp2)U(Zp2).

def gensdeUdeZp(p):

lst=[]

k=1

while k<p-1:

if gcd(k,p-1)==1:

lst.append(k)

k= k+1

return lst

POR HACER YO:

TORSION

KLEIN Y S4 SUB

KLIEN Y S4 NORM

<https://turing.mat.ucm.es:8080/home/antsev01/1/>

<http://doc.sagemath.org/html/en/thematic_tutorials/group_theory.html>

<https://turing.mat.ucm.es:8080/home/antsev01/3/>

<http://verso.mat.uam.es/~pablo.angulo/doc/laboratorio/b3s2.html>

"""

def UnidadesDeZ\_114():

result=[]

for n in range(1,114):

if gcd(114,n)==1:

result.append(n)

return result

def omultenUds(n,m):

A=Integers(m)

n\_0=n

generados=[]

while len(generados)<len(UnidadesDeZ\_114()) and A(n\*n\_0) not in generados:

n=A(n\*n\_0)

generados.append(n)

return len(generados)

print len(UnidadesDeZ\_114())

ordenes=[]

for i in UnidadesDeZ\_114():

ordenes.append(omultenUds(i,114))

print omultenUds(5,114)

print Set(ordenes)

"""

Ç

UK=[]

for i in K:

UK.append(i)

UK.pop(0)

def omultenUK(x):

K=POL.quotient(3\*t^3 + 11\*t^2 + 1)

x\_0=x

generados=[]

while len(generados)<len(UK) and K(x\*x\_0) not in generados:

x=K(x\*x\_0)

generados.append(x)

return len(generados)

omultenUK(3)

**PARTE 2**

A modo de ejemplo, a continuación definimos el grupo de permutaciones de 3 elementos y calculamos el producto de dos transposiciones.

 L=[(1,2),(2,3),(1,3)]

S3=PermutationGroup(L)

s=S3((1,2))

t=S3((2,3))

print s\*t

Define el grupo de permutaciones de 10 elementos.

#sabemos que el grupo d permutaciones esta generado por la transposición y la permutación d todos los elementos

L=[(1,2),(1,2,3,45,6,7,8,9,10)]

 S10=PermutationGroup(L)

Definir el grupo de permutaciones de 4 elementos. Usando el comando **set**define un conjunto que contenga los elementos del subgrupo de Klein. Comprobar que efectivamente es un subgrupo de S\_4.

P=[(1,2),(1,2,3,4)]

Klein = KleinFourGroup()

ElementosDeKlein=set(Klein.list())

(Ejercicio no obligatorio) Probar que el subgrupo de Klein es un subgrupo normal de S4S4.

¿Cuál es el orden del grupo U(Z114)U(Z114)?

Si en la siguiente casilla escribes print a y print c te aparecerán números entero.

#sabemos que el orden del grupo de unidades de Z\_n es el número de coprimos con n menores que él, es decir, la función phi de Euler

print(euler\_phi(114))

36

¿Pertenece la clase [a]114[a]114 al grupo U(Z114)U(Z114)? ¿ Y [c]114[c]114?

#si n y m son coprimos entonces n pertenece a la clase de las unidades de Zm

if gcd(a,114)==1:

print(True)

else:

print(False)

if gcd(c,114)==1:

print(True)

else:

print(False)

False

True

Calcula el inverso multiplicativo de [c]114[c]114 en U(Z114)U(Z114) (usando Bezout con el comando xgcd)

(xgcd(c,114))[1]

5

A continuación puedes encontrar una función que calcula el orden multiplicativo de [n]m[n]m en U(Zm)U(Zm). Calcula el orden multiplicativo de la clase [c]114[c]114 en U(Z114)U(Z114).

Calcula el orden multiplicativo de la clase de [c]114[c]114 en Z114Z114.

omult(c,114)

18

Calcula los coeficientes de torsión de U(Z114)U(Z114).

En la variable p tiene asignada un cierto número primo. Encontrar cinco generadores distintos del grupo cíclico U(Zp)=Zp∖{[0]}U(Zp)=Zp∖{[0]}.

211

#buscamos aquellos valores coprimos con p, es decir, generadores de Zp

def generadores(p):

i=1

conseguidos=0

lista=[]

while conseguidos<5:

if gcd(p-1,i)==1:

lista.append(i)

i=i+1

conseguidos=conseguidos+1

else:

None

i=i+1

return lista

generadores(p)

Si [x]p[x]p es uno de los generadores que has encontrado en la respuesta anterior, comprueba si [x]p2[x]p2 o [x+p]p2[x+p]p2 es un generador de U(Zp2)U(Zp2).

#en primer lugar encontramos todos los generadores de Zp^2, y entre ellos buscamos si está x o x+p

def generadores2(p):

i=1

conseguidos=0

lista=[]

while i<p^2:

if gcd((p^2)-1,i)==1:

lista.append(i)

i=i+1

conseguidos=conseguidos+1

else:

None

i=i+1

return lista

def evaluar(x,p):

i=0

while i<len(generadores2(p)):

if x == generadores2(p)[i]:

return True

i=i+1

return False

ejercicio de permutaciones:

Klein\_group=direct\_product\_permgroups([CyclicPermutationGroup(2),CyclicPermutationGroup(2)])

Klein\_group.list()

def UZm(m):

lista=[]

i=0

while i<m:

if gcd(i,m)==1:

lista.append(i)

i=1+i

return lista

def omul2(n,m):

A=Integers(m)

i=1

k=n

lista=[]

continuar=True

while len(lista)<=len(UZm(m)) and continuar:

k=A(n\*k)

i=i+1

continuar=True

for h in lista:

if h==k:

continuar=False

lista.append(k)

return i-2

def ordenes(q):

ords=[]

for p in UZm(q):

anadir=True

for w in ords:

if w==omul2(p,q):

anadir=False

if anadir:

ords.append(omul2(p,q))

return ords

ordenes(114)