TP1_v3

MT10/P13

-

TP

Groupes d'ordre 4

1 Test d'une loi

1.1 Codage d'une loi sur un ensemble

Question 1

Nous utilisons des listes pour coder les lois des groupes (Z/4Z, +, 0) et (Z/2Z × Z/2Z, +, 0)

```
Z4=[[0,1,2,3],[1,2,3,0],[2,3,0,1],[3,0,1,2]]
```

```
Z2=[[0,1,2,3],[1,0,3,2],[2,3,0,1],[3,2,1,0]]
```

Pour Z/4Z la correspondance entre un entier et un élément de Z/Z4 est immédiate, mais pour Z/2Z × Z/2Z, il faudrait coder cela sur 2 entiers ce qui provoquerai par la suite des difficultés de programmation, pour cela, on utilise un dictionnaire, Dico_Z2, qui nous donnera une correspondance entre un entier et un couple d'éléments de Z/2Z × Z/2Z.

```
Dico_Z2={0:(0,0),1:(0,1),2:(1,0),3:(1,1)}
Dico_Z4={0:0,1:1,2:2,3:3}
```

Question 2

On crée une autre représentation de Z/4Z en utilisant un objet CyclicPermutationGroup, en explorant ses méthodes, on trouve la méthode cayley_table(), qui permet d'afficher ses tables de loi.

```
Z4_bis=CyclicPermutationGroup(4)
```

```
Z4_bis.cayley_table()
```

```
    *  a b c d
     +--------
```

```
a| a b c d
b| b c d a
c| c d a b
d| d a b c
```

1.2 Elément neutre

Question 3

On teste les éléments un par un. Jusqu'à trouver un élément neutre. Si on en a pas trouvé, on l'affiche et on ne renvoie rien.

```
def neutre(t):
    for a in range(len(t)):
        e = true
        for i in range(len(t)):
            if (t[a][i] != i or t[i][a] != i):
                e = false
        if e :
            return a
    return 'none'
```

```
Dico_Z2[neutre(Z2)]
```

```
(0, 0)
```

```
neutre(Z4)
```

```
0
```

```
hazard=[[1,3,3,3],[1,2,3,2],[1,1,2,1],[3,2,1,2]]
neutre(hazard)
```

```
'none'
```

1.3 Elément symétrique

Question 4

On cherche l'élément neutre, puis pour chaque élément, on cherche son symétrique si il existe.

```
def symetrie(t,dico):
    sym=[]
    e=neutre(t)
    for i in range(len(t)):
        for j in range(len(t)):
            if (t[i][j] == e and t[i][j] == e):
                sym.append((i,j))
    return sym
```

```
symetrie(Z4,Dico_Z4)
```

```
[(0, 0), (1, 3), (2, 2), (3, 1)]
```

```
symetrie(Z2,Dico_Z2)
```

```
[(0, 0), (1, 1), (2, 2), (3, 3)]
```

Table des symétriques

Pour Z/Z4

0	0
1	3
2	2
3	1

Pour Z/Z2 * Z/Z2

(0,0)	(0,0)
(0,1)	(0,1)
(1,0)	(1,0)
(1,1)	(1,1)

1.4 Associativité

Question 5

On crée une procédure de test en utilisant la définition de l'associativité :

$(\forall x, y, z \in E)(x * y) * z = x * (y * z)$

```
def associative(t):
    for i in range(len(t)):
        for j in range(len(t)):
            for k in range(len(t)):
                if (t[t[i][j]][k] != t[i][t[j][k]]):
                    return false
    return true
```

```
associative(Z2)
```

```
    True
```

```
associative(Z4)
```

```
    True
```

Ces instructions nous permettent d'obtenir tous les détails des fonctions suivantes :

```
#TestSuite??
```

```
#TestSuite(Z4_bis).run(verbose=True)
```

2. Test d'un morphisme

Question 7

1.
On crée une fonction qui teste si une application est un morphisme.

Pour cela il faut vérifier que $(\forall x, y \in G) f(x * y) = f(x) * f(y)$

```
def is_a_morphisme(f,t,y):
    for i in range(len(t)):
        for j in range(len(t)):
            if (f[t[i][j]] != y[f[i]][f[j]]):
                return false
    return true
```

2.

On teste toutes les applications bijectives possibles.

```
def automorphismes(Z):
    automorph=[]
    for f in Arrangements(range(len(Z)),len(Z)).list():
        if is_a_morphisme(f,Z,Z):
            automorph.append(f)
    return automorph
```

```
automorphismes(Z4)
```

```
[[0, 1, 2, 3], [0, 3, 2, 1]]
```

```
automorphismes(Z2)
```

```
[[0, 1, 2, 3], [0, 1, 3, 2], [0, 2, 1, 3], [0, 2, 3, 1], [0,
2], [0, 3, 2, 1]]
```

3.

On va tester toutes les applications bijectives possibles.

```
def isomorphe(Z1,Z2):
    morphisme=[]
    for f in Arrangements(range(len(Z1)),len(Z1)).list():
        if is_a_morphisme(f,Z1,Z2):
            morphisme.append(f) #On cherche tout les
morphismes de Z1 dans Z2
    for f in Arrangements(range(len(Z2)),len(Z2)).list():
        if is_a_morphisme(f,Z2,Z1): #On cherche les morphismes
```

```
de Z2 dans Z1
            for e in morphisme : #Si il correspond a un
morphisme inverse, on retournera true
                identite = true
                for i in range(len(e)) :
                    if (f[e[i]] != i or e[f[i]] != i) :
                        identite = false
                if identite :
                    return true
    return false
```

```
isomorphe(Z4,Z2)
```

```
    False
```

```
isomorphe(Z2,Z4)
```

```
    False
```

Nous n'avons pas trouvé de morphismes, Z/4Z et Z/2Z × Z/2Z ne sont donc pas isomorphes.

3 Où l'on identifie tous les groupes d'ordre 4

Question 8

Si on a deux fois le même élément sur la même ligne ou la même colonne alors cela signifierait que l'élément neutre n'est pas unique ce qui viendrait en contradiction de la définition de groupe

Question 9

1.

```
TestCarreLatin=[[1,1,2,3],[1,2,3,0],[2,3,0,1],[3,0,1,2]]
```

```
neutre(TestCarreLatin)
```

```
    'none'
```

```
tI=[[0,1,2,3],[1,0,3,2],[2,3,1,0],[3,2,0,1]]
Dico={0:0,1:1,2:2,3:3}
```

```
neutre(tI)
```

```
    0
```

```
symetrie(tI,Dico)
```

```
    [(0, 0), (1, 1), (2, 3), (3, 2)]
```

```
associative(tI)
```

```
    True
```

La table tI est bien une table de groupe

```
tII=[[0,1,2,3],[1,0,3,2],[2,3,0,1],[3,2,1,0]]
```

```
neutre(tII)
```
```
    0
```
```
symetrie(tII,Dico)
```
```
    [(0, 0), (1, 1), (2, 2), (3, 3)]
```
```
associative(tII)
```
```
    True
```
La table tII est bien une table de groupe
```
tIII=[[0,1,2,3],[1,2,3,0],[2,3,0,1],[3,0,1,2]]
```

```
neutre(tIII)
```
```
    0
```
```
symetrie(tIII,Dico)
```
```
    [(0, 0), (1, 3), (2, 2), (3, 1)]
```
```
associative(tIII)
```
```
    True
```

La table tIII est bien une table de groupe

```
tIV=[[0,1,2,3],[1,3,0,2],[2,0,3,1],[3,2,1,0]]
```

```
neutre(tIV)
```
```
    0
```
```
symetrie(tIV,Dico)
```
```
    [(0, 0), (1, 2), (2, 1), (3, 3)]
```
```
associative(tIV)
```
```
    True
```

La table tIV est bien une table de groupe.

Ces 4 tables sont bien des tables de groupes, en cherchant toutes les possibilités, on remarque que ce sont les seules.

2.

```
isomorphe(Z4,tI)
```
```
    True
```
```
isomorphe(Z4,tII)
```
```
    False
```

```
isomorphe(Z4,tIII)
```
```
True
```
```
isomorphe(Z4,tIV)
```
```
True
```

Z/Z4 est isomorphe à tI, tIII et tIV

```
isomorphe(Z2,tI)
```
```
False
```
```
isomorphe(Z2,tII)
```
```
True
```
```
isomorphe(Z2,tIII)
```
```
False
```
```
isomorphe(Z2,tIV)
```
```
False
```

tI, tIII, tIV ne sont pas isomorphes à Z/2Z × Z/2Z, ce résultat était prévisible car elles sont isomorphes à Z/4Z qui n'est pas isomorphe à Z/2Z × Z/2Z.
tII est isomorphe à Z/2Z × Z/2Z,

4 Utiliser les fonctionnalités de Sage

Question 10

```
C5=CyclicPermutationGroup(5)
C5.cayley_table()
```
```
*  a b c d e
 +----------
a| a b c d e
b| b c d e a
c| c d e a b
d| d e a b c
e| e a b c d
```
```
C6=CyclicPermutationGroup(6)
C6.cayley_table()
```
```
*  a b c d e f
 +------------
a| a b c d e f
b| b c d e f a
c| c d e f a b
d| d e f a b c
e| e f a b c d
f| f a b c d e
```
```
K4=KleinFourGroup()
K4.cayley_table()
```

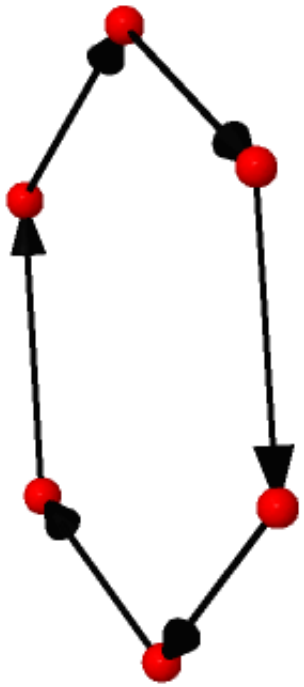
```
*  a b c d
 +--------
a| a b c d
b| b a d c
c| c d a b
d| d c b a
```

```
C6.caley_graph().show3d()
```

```
Traceback (click to the left of this block for traceback)
...
AttributeError: 'CyclicPermutationGroup_with_category' object
attribute 'caley_graph'
```

```
C6.cayley_graph().show3d()
```

Sleeping... Make Interactive



```
K4.cayley_graph().show3d()
```

Sleeping... Make Interactive