TP (5déc17) Utilisation de la réduction de Smith programmée pour la résolution d'exercices

Fontions définies dans le corrigé du devoir

```
def showop(o): # affichage d'une liste d'opérations
    l=""
    for op in o:
        if op[0]==1:l=l+"C"+str(op[1])+" + "+str(op[3])+"C"+str(op\
   [2])+" → C"+str(op[1])+"\n"
        elif op[0]==2:l=l+'L'+str(op[1])+' + '+str(op[3])+'*L'+str(\
   op[2])+' → L'+str(op[1])+"\n"
        elif op[0]==3:l=l+'C'+str(op[1])+"  C"+str(op[2])+"\n"
        elif op[0]==4:l=l+'L'+str(op[1])+"  L"+str(op[2])+"\n"
    return(l)
```

```
import copy #B=copy.deepcopy(A)
def transf(A,o): #transformation d'une matrice A suivant la liste d'\
   opération o
    B=copy.deepcopy(A) #sinon A est modifié
    for op in o:
        if op[0]==1:B[:,op[1]]=B[:,op[1]]+op[3]*B[:,op[2]]
        elif op[0]==2:B[op[1],:]=B[op[1],:]+op[3]*B[op[2],:]
        elif op[0]==3:
            C=B[:,op[1]];B[:,op[1]]=B[:,op[2]];B[:,op[2]]=C
        elif op[0]==4:
            L=B[op[1],:];B[op[1],:]=B[op[2],:];B[op[2],:]=L
    return(B)
```

```
def fsmith(A,verif=false):
    if verif:show(A)
    p=A.nrows();q=A.ncols()
    L=[abs(A[i][j]) for i in range(p) for j in range(q)]
    if p*q==0 or max(L)==0:
        #print([]);
        return([])
    else:
        a=min(L[i] for i in range(p*q) if L[i]!=0)
        k=L.index(a);i0=k//q;j0=k%q;a=A[i0][j0] #a avec signe
        l=[i for i in range(p*q) if L[i]%a!=0]
        if verif:print '(verif) min,position =',[a,i0,j0]#verif
        if l==[]:  #pivot suivant a
            o=[[1,j,j0,-A[i0][j]/a] for j in [0..q-1] if j!=j0]\
                + [[2,i,i0,-A[i][j0]/a] for i in [0..p-1] if i!=i0]
            if j0!=0:o=o+[[3,0,j0]]
            if i0!=0:o=o+[[4,0,i0]]
            if verif:print '(verif) pivot',showop(o);show(transf(A,o\
   ))
            return(o+[[op[0],1+op[1],1+op[2]]+op[3:] for op in \
   fsmith(transf(A,o).submatrix(1,1),verif)])#
        else: #vers l'apparition du pgcd des coeff de A comme coeff \
   de A
```

```
            k=min(l);i1=k//q;j1=k%q;b=A[i1][j1]
            if verif:print '(verif) coef non multiple de min, \
    position =',[b,i1,j1]
            if A[i0][j1]%a != 0:
                o=[[1,j1,j0,-(A[i0][j1]//a)]]
            elif A[i1][j0]%a != 0:
                o=[[2,i1,i0,-(A[i1][j0]//a)]]
            else:
                o=[[2,i1,i0,-(A[i1][j0]//a)],[2,i0,i1,1],\
                   [1,j1,j0,-(((1-A[i1][j0]//a)*A[i0][j1]+b)//a)]]
            if verif:print '(verif) vers pgcd',showop(o);show(transf\
    (A,o))
            return(o+fsmith(transf(A,o),verif))
```

Q. Que font les fonctions définies ci-dessus ?

Ex 1. L'application $f : \mathbb{Z} \to \mathbb{Z}/24 \times \mathbb{Z}/15$ est elle surjective ? Si ce n'est pas le cas, donner un paramétrage de l'image.

Ecrire une fonction qui teste si le couple des classes de deux entiers a, b est dans l'image de f

Ex 2. Donner une base et une équation (c-a-d matrice) du $\mathbb{Z}$-module formé des éléments de $\mathbb{Z}^5$ qui sont combinaisons linéaires à coefficients rationnels des deux vecteurs $(1, 2, 3, 4, 5)$ et $(6, 7, 8, 9, 10)$

Ex 3. Répondre aux exercices du sujet de session2 2016-17

Réponse.

Rq. matrice? pour une aide sur la contruction de matrices, matrix.<tab >pour voir les méthodes qui s'applique à une matrice

Q. L'exemple d'execution du corrigé du devoir Smith et de la réponse à l'exercice 2 ci-dessous montre que showop affiche de façon conventionnelle une suite d'opérations encodées ; la définition de showop indique l'encodage retenu pour les opérations :

[1,i,j,a] pour l'opération sur les colonnes $C_i + a * C_j \to C_i$

[3,i,j] pour l'opération sur les colonnes $C_i \leftrightarrow C_j$

[2,i,j,a] et [4,i,j] pour ces mêmes opérations sur les lignes.

La fonction transf(A,o) transforme une matrice A suivant la liste d'opérations o. Sa définition est récursive.

La fonction fsmith(A) rend une liste d'opération o telle que transf(A,o) est la réduite de Smith de A. Sa définition est également récursive.

Ex1: L'application est surjective ssi l'application linéaire $\mathbb{Z}^3 \to \mathbb{Z}^2$ de matrice $\begin{pmatrix} 1 & 24 & 0 \\ 1 & 0 & 15 \end{pmatrix}$ est surjective, ce qui équivaut à ce que la réduite de Smith n'ait que des éléments inversibles sur la diagonale.

Image, équation

```
A=matrix([[1,24,0],[1,0,15]])
op=fsmith(A)
print
```

```
show(transf(A,op))
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix}$$

Ex2:

```
M=matrix(2,5,[[1,2,3,4,5],[6,7,8,9,10]]).transpose();show(M)
o=fsmith(M)
print
show(transf(M,o))
print 'Operations:\n',showop(o)
```

$$\begin{pmatrix} 1 & 6 \\ 2 & 7 \\ 3 & 8 \\ 4 & 9 \\ 5 & 10 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & -5 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Operations:
C1 + -6C0 → C1
L1 + -2*L0 → L1
L2 + -3*L0 → L2
L3 + -4*L0 → L3
L4 + -5*L0 → L4
L2 + -2*L1 → L2
L3 + -3*L1 → L3
L4 + -4*L1 → L4

```
#liste des opérations sur les colonnes extraite de fsmith(M) : les \
   op tq op[0] soit impair d'après l'encodage des opérations tel qu'\
   il apparaît dans la fonction showop
oc=[op for op in o if op[0]%2==1]
print showop(oc)
#liste des opérations sur les lignes
ol=[op for op in o if op[0]%2==0]
#matrices de passage
P=transf(matrix.identity(M.nrows()),ol)
Q=transf(matrix.identity(M.ncols()),oc)
show('P=',P,', Q=',Q,', P*M*Q=',P*M*Q)
```

C1 + -6C0 → C1

$$P=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 2 & -3 & 0 & 1 & 0 \\ 3 & -4 & 0 & 0 & 1 \end{pmatrix}, Q=\begin{pmatrix} 1 & -6 \\ 0 & 1 \end{pmatrix}, P*M*Q=\begin{pmatrix} 1 & 0 \\ 0 & -5 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

D'après la forme de PMQ l'image de PMQ est engendré par e1 et 5e2 où e1 et e2 sont les deux premiers vecteurs de la base canonique de $\mathbb{Z}^5$, donc le $\mathbb{Z}$-module cherché est engendrée par P-1e1 et P-1e2, c'est à dire par les deux premières colonnes de P-1 (l'image de M est elle engendrée par la première colonne et 5 fois la seconde colonne de P-1)

P-1 peut se calculer avec l'instruction Sagemath P^(-1) ou bien en transformant la matrice identité suivant la réciproque de la composée des opérations sur les lignes.

```
P^(-1)
```

```
[1 0 0 0 0]
[2 1 0 0 0]
[3 2 1 0 0]
[4 3 0 1 0]
[5 4 0 0 1]
```

Une équation du $\mathbb{Z}$-module en question est donnée par l'annulation des trois dernières coordonnées de P*(x1,,x5) donc par la matrice formée des trois dernières lignes de P

```
P[2:]
```

```
[ 1 -2  1  0  0]
[ 2 -3  0  1  0]
[ 3 -4  0  0  1]
```

```
#matrix.block?
```