L3 Alg effective feuille 3

3 octobre 2016

Quelques algorithmes pour le TD-TP2

Pour chacun des algorithme, écrire une preuve de terminaison et une preuve de ce que l'agorithme calcule bien ce qu'on devine. (Corriger l'algorithme au besoin.)

```
def factoriel(n):
    if not n in NN:print(str(n)+" n'est pas de type NN")
    elif n==0:return(1)
    else:return(n*factoriel(n-1))
```

```
factoriel(-1)
```

```
-1 n'est pas de type NN
```

```
def fpgcd(a,b): #pgcd de deux entiers
    print("fpgcd("+str(a)+","+str(b)+")") #pour analyse de l'\
   algorithme
    if not (a in ZZ and b in ZZ):return(str((a,b))+" n'est pas de \
   type ZxZ") # restriction sur l'argument
    elif b==0:return(abs(a))
    else:return(fpgcd(b,a%b))
```

```
fpgcd(105,132)
```

```
fpgcd(105,132)
fpgcd(132,105)
fpgcd(105,27)
fpgcd(27,24)
fpgcd(24,3)
fpgcd(3,0)
3
```

```
def lpgcd(a): #pgcd d'une liste d'entiers
    print("lpgcd"+str(a)) #analyse de l'algorithme
    n=len(a)
    if n==0:return(0)
    elif n==1:return(abs(a[0]))
    elif a[n-1]==0:return(lpgcd([a[i] for i in [0..n-2]]))
    else:return(lpgcd([a[i] for i in [0..n-3]]+[a[n-1],a[n-2]%a[n\
   -1]]))
```

```
lpgcd([-2,6,4])
```

```
lpgcd[-2, 6, 4]
lpgcd[-2, 4, 2]
lpgcd[-2, 2, 0]
lpgcd[-2, 2]
lpgcd[2, 0]
lpgcd[2]
2
```

```
def rel (a,b): #pgcd,coeff d'une relation de Bezout, nbre d'ité\
```

```
    rations pour deux entiers
    if b==0:return(a,1,0,1)
    else:
        d,u,v,c=rel(b,a%b)
        #print(d,u,v)
        return(d,v,u-v*(a//b),c+1)
```

```
def rell (l): #pgcd, relation de Bezout pour une liste d'entiers ; \
    fait appel à rel
    n=len(l)
    if n==0:return(0,[])
    elif n==1:return(l[0],[1])
    else:
        a,b=rell(l[0:n-1])
        d,u,v,c=rel(a,l[n-1])
        return(d,[u*b[i] for i in [0..n-2]]+[v])
```

```
gcd(0,0)
```
```
0
```

```
def solpart (a,b): #sol particulière de l'eq. a[0]*x[0]+...=b
    if a==[]:return([])
    else:
        d,k=rell(a)
        if d==0:
            if b!=0:return([])
            else:return(k)
        elif b%d!=0:return([])
        else:return([vector([b//d*k[i] for i in [0..len(k)-1]])])
```

```
a=[12,30,20]
x=solpart(a,10);x
```
```
[(30, -15, 5)]
```

```
30*12+(-15)*30+5*20
```
```
10
```

```
matrix(ZZ,[a])*x[0]
```
```
(10)
```

```
#Plongement de Z^n dans Z^(n+1) : un exemple
x=vector([1,2,3,4]);x
vector([x[i] for i in [0..len(x)-1]]+[0])
```
```
(1, 2, 3, 4)
(1, 2, 3, 4, 0)
```

```
def sol0(a):
    n=len(a)
    if n==0:return([])
    else:
        d=lpgcd(a)
        if d==0:return("all")
```

```
        else:
            a=[a[i]/d for i in [0..n-1]]
            d,k=rell([a[i] for i in [0..n-2]])
            return("pas fini")
```