C.R. T.P. 6 partie 2

2 L'algorithme d'exponentiation rapide

Le but de cet algorithme est de calculer la puissance N-ième x^N d'un nombre x qui peut être un entier ou un flottant, avec un minimum d'opérations.

Cet algo. s'applique à différents types de nombres x : flottants, entiers.... on l'appliquera aussi dans les anneaux de congruences, où il est encore plus efficace, et qu'on s'en servira pour des problèmes d'arithmétiques...

a) L'idée essentielle : Si $N = a_0 + a_1 2 \cdots + a_n 2^n$, écriture en base deux, avec $a_i \in \{0, 1\}$ alors :

$$x^N = x^{a_0}(x^2)^{a_1} \dots (x^{2^n})^{a_n}.$$

Ensuite deux points de vue possibles, cela dépend si on connait déjà l'écriture en base 2 de N ou pas.

b) **1ère méthode (des poids faibles vers les poids forts)**
L'algorithme qui suit incorpore l'algorithme d'obtention des chiffres successifs de l'écriture en base deux, des poids faibles vers les poids forts, obtenu au § 1, ainsi on n' a pas à connaître à l'avance l'écriture en base deux.
Pour mettre en oeuvre l'algorithme on utilise trois variables qu'on va appeler `res` et `aux` comme résultat et auxiliaire et la variable `N` qui au départ contient comme valeur l'exposant N et qui va permettre à chaque étape de calculer le chiffre du développement en base 2 de N.

```
def exprapide(x,N):
    """retourne le calcul de xˆN par la méthode d'exp. rapide"""
    aux=x
    res=1
    while N!=0 :
        if N%2==1:
            res=res*aux
        aux=aux*aux
        N=N//2
    return res
```

c) Cette fois on suppose connue l'écriture en base deux de `N`. Par exemple en python, on peut l'obtenir avec `bin`. Attention cependant `bin` renvoie une *chaîne de caractères*, qui commence par `'0b'`.

```
def rapidePF(x,N):
    res=1
    tab=bin(N)[2:]# chaîne de carac. contenant
    # l'écriture en base deux de N
    for i in range(len(tab)):
        if int(tab[i])==1: # int nécessaire car tab[i] est un str
            res=res*res*x
        else:
            res=res*res
    return res
```