C.R. T.P. 8 (suite)

2 Le test de Fermat

2.1 Exponentiation modulaire

Comme dit par l'énoncé, on reprend simplement l'algorithme d'exp. rapide, en réduisant modulo m à chaque étape.

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

Cette algorithme est *très rapide*, car on travaille toujours modulo m à chaque étape (et que le nombre d'étapes est en $O(\log N)$ par l'exp. rapide).

2.2 Le test de Fermat et son efficacité

Le petit théorème de Fermat dit que si n est un nombre premier alors pour tout $a \in \mathbb{N}$, $a^n \equiv a\,[n]$. Ce théorème fournit donc une **C.N.** pour qu'un nombre soit premier. Si on a un a tel que $a^n \not\equiv a\,[n]$, on est sûr que n n'est pas premier. On dira que n ne *passe pas* le test de Fermat pour la valeur a en question (appelé aussi *témoin de Fermat*) et donc n'est pas premier.

a) Ecrire une fonction `test_Fermat` qui prend en argument un nombre `n` et un *argument facultatif* `a`, qui sinon admet `a=2` comme valeur par défaut, et qui teste si `n` « passe » le test de Fermat pour a, en renvoyant un booléen.

```
def test_fermat(n,a=2):
       return (expo_mod(a,n,n)==a%n)
```

b) (i) Faisons la liste des nombres inférieurs à 10000 qui ne sont pas premiers mais qui passent le test de Fermat pour $a = 2$.

```
L2=[]
for i in range(2,10000):
        if test_fermat(i)!=estpremier(i):
            L2.append(i)
print(L2)
```

On obtient : `[341, 561, 645, 1105, 1387, 1729, 1905, 2047, 2465, 2701, 2821, 3277, 4033, 4369, 4371, 4681, 5461, 6601, 7957, 8321, 8481, 8911]`

Remarque : l'énoncé proposait plutôt d'utiliser `Erato` que `estpremier` mais Erato renvoie la liste des nombres premiers, il serait plus commode d'utiliser plutôt la liste des nombres non premiers, qui sont les indices des entrées `False` du tableau renvoyé par `erato0`. Autrement dit, on fait une fonction `NonPremier` dont le code est identique à celui `Erato` sauf qu'on garde les numéro des entrées `False`. On utilise alors la liste `L=NonPremier(10000)`.

(ii) Faisons la liste des nombres inférieurs à 10000 qui ne sont pas premiers mais qui passent le test de Fermat pour $a = 3$.

On peut faire la même chose pour $a = 3$. On obtient :

`[6, 66, 91, 121, 286, 561, 671, 703, 726, 949, 1105, 1541, 1729, 1891, 2465, 2665, 2701, 2`

(iii) Finalement et là cela devient très intéressant : la liste des nombres inférieurs à 10000 qui ne sont pas premiers mais qui passent le test de Fermat pour $a = 2$ et pour $a = 3$ se calcule avec :

```
for i in range(2,10000):
    p=estpremier(i)
    if  p==False and (test_fermat(i) and test_fermat(i,3)):
        L23.append(i)
print(L23)
```

Ce qui donne le résultat très intéressant suivant : `[561, 1105, 1729, 2465, 2701, 2821, 6601, 8911]`

c) Comparer la rapidité de la réponse des deux fonctions `test_Fermat` et `testpremiernaif` sur des grands nombres : fabriquer pour cela une liste de grands nombres impairs consécutifs (par exemple des nombres 100 chiffres) et tester la fonction `testpremiernaif` sur ces nombres : lorsqu'elle « rame » sur un nombre, essayez le test de Fermat sur ce nombre.

La réponse était dans l'énoncé : *Par exemple en commençant à* `N=10**(100)+1` *j'ai trouvé* `N+36` *qui résistait à* `testpremiernaif` *mais pas à Fermat avec le témoin* 2*..*

d) Il suffit de faire à chaque étape, pour chaque entier n, le test de Fermat pour tous les $a \in [\![2, n-1]\!]$ (car le cas de $a = 0$ et $a = 1$) est trivial. La fonction Carmichael suivante renvoie `True` ssi `n` est nombre non premier qui passe le test de Fermat pour tous les $a \in \mathbb{Z}$ i.e. tous les $a \in [\![2, n-1]\!]$.

```
def Carmichael(n):
    if estpremier(n)==True :
        return False #"nombre premier"
    else:
        i=2
        while (i<n) and (expo_mod(i,n,n)==i):
            i+=1
        if i==n :
            return True
        else :
            return False
```

Comme indiqué dans l'énoncé la liste qu'on trouve pour les nombres de Carmichael inférieurs à 10000 est presque la même que celle des nombres non premiers qui passent le test de Fermat pour $a = 2$ et $a = 3$, il y en a seulement un de moins !