TP2: décomposition d'un entier en produit de facteurs premiers

Exercice 1:

Dans l'algorithme de décomposition d'un entier n (avec $n \geq 2$) en produit de facteurs premiers la première étape consiste à trouver le plus petit nombre premier p qui divise n . Notons $f(n)$ ce nombre premier.

1. Calculez $f(143)$, $f(6557)$ (vous pouvez utiliser une calculatrice). Quelle est la principale difficulté selon vous?
2. Si n n'est pas premier pourquoi doit-on avoir $f(n) \leq \sqrt{n}$?
3. Montrez que $f(n)$ est aussi le plus petit diviseur de n différent de 1.
4. Que peut-on dire de la parité de $f(n)$?
5. En utilisant les résultats des questions 2, 3 et 4 programmez la fonction f en python.
   (on rappelle qu'en Python le reste de la division euclidienne de a par b est $a\%b$ .Le quotient, quant à lui, s'obtient par $a//b$. Pour calculer une racine carrée il faut utiliser la fonction **sqrt** de la bibliothèque **math**:

```
from math import *

l=int(sqrt(39))

l
6
```

   dans cette suite de commande la fonction **int** permet d'obtenir la partie entière de $\sqrt{39}$ et d'avoir une variable entière **l**.)

6. En vous servant de l'algorithme de la question 5 calculez successivement:
   (a) $f(2^{11}-1)$ (rappel: 2^{11} s'écrit $2**11$ en Python.
   (b) $f(2^{21}-1)$
   (c) $f(2^{31}-1)$
   (d) $f(2^{41}-1)$
   (e) $f(2^{51}-1)$
   (f) $f(2^{61}-1)$

   (Remarque: en Python la taille des entiers n'est pas limitée. Par exemple, le nombre $2^{61}-1$ est codé sur 61 bits sans arrondi ou approximation)

Exercice 2:

On considère un nombre entier $n = p_1 \times p_2$ avec p_1 et p_2 deux nombres premiers tels que $2 < p_1 < p_2$:

1. que vaut $f(n)$?
2. Soit β le nombre de bits minimum servant à écrire p_1 en base 2:
   (a) Donnez un encadrement de p_1 en fonction de β.
   (b) Donnez un encadrement de $\sqrt{p_1}$ en fonction de β.
   (c) On considère la fonction f programmée en python dans l'exercice 1. Donnez, en fonction de β, un minorant du nombre de passages dans la boucle de la fonction f .
3. Donnez une estimation expérimentale de β_0 , la valeur à partir de laquelle l'algorithme $f(n)$ sera trop long.

(**Remarques:**
1) Dans l'algorithme de chiffrement RSA l'impossibilité de trouver p_1 lorsqu'on donne l'entier n (avec $n = p_1 \times p_2$) rend le décryptage impossible en pratique. Il faut bien sûr que p_1 soient encodés sur un nombre de bits suffisamment grand (quelques milliers en pratique).
2) Il existe des algorithmes plus efficaces que votre fonction f de l'exercice 1 (par exemple l'algorithme rho de Pollard) mais encore insuffisants pour les grands nombres premiers)

Exercice 3: l'algorithme de décomposition d'un nombre entier en produit de facteurs premiers

L'objectif de cet exercice est de programmer en python une fonction $g(n)$ qui accepte en entrée un entier $n \geq 2$ et dont la sortie est une liste dans laquelle les facteurs premiers apparaissent par ordre croissant et autant de fois que leur exposant dans la décomposition. On aura par exemple $g(48) = [2, 2, 2, 2, 3]$.

Pour réaliser ce programme vous pouvez utiliser la fonction f de l'exercice 1.

En Python, la concaténation de la liste $a = [1, 2, 3]$ avec la liste $b = [4, 5]$ s'obtient en écrivant simplement $a + b$ (et on obtient la liste $[1, 2, 3, 4, 5]$).