Tests de primalités

Un nombre premier est un nombre qui n'a d'autres diviseurs que 1 et lui-même.
Ex : 2, 3, 5, 11, 97 sont premiers mais 0, 1, 4 ne le sont pas

I. La division jusqu'à $\sqrt{n}$

Soit $d \in \mathbb{N}^*$ tel que d | n
On voit facilement que n/d divise aussi n.
Par disjonction de cas, on peut montrer que soit d, soit n/d est inférieur ou égal à racine de n :

- Si $d \geq \sqrt{n} \Rightarrow \frac{1}{d} \leq \frac{1}{\sqrt{n}} \Rightarrow \frac{n}{d} \leq \sqrt{n}$
- De même si $d \leq \sqrt{n}$

Donc on peut chercher tous les diviseurs d de n jusqu'à racine de n et à chaque fois prendre n/d comme un autre diviseur

NB : Il est possible d'accélérer un peu cet algorithme en testant si le nombre est divisible par 2 puis seulement les nombres impairs

II. Test de primalité de Fermat

Ce test se base sur le petit théorème de Fermat (1) qui stipule :
$$p \in \mathbb{P}, PGCD(a,p) = 1 \Rightarrow a^{p-1} \equiv 1[p]$$

Autrement dit pour tout entier « a » premier avec « p » (un nombre premier) le reste de la division euclidienne de « a » à la puissance (p-1) par « p » est 1.

Le test de primalité pour un entier « p » est le suivant :

- On choisit plusieurs fois un entier « a » tel que 2 ≤ a ≤ p-2
  - Pour chaque « a », si on n'a pas la relation (1) alors on déduit par contraposée que « p » n'est pas premier (NB : on peut vérifier la relation avec l'exponentiation binaire)
- Si on a toujours la relation, alors « p » est PROBABLEMENT premier

En effet, cet algorithme est probabiliste, donc il ne fonctionne pas toujours (exemple : nombres de Charmichael, dont il en existe 646 inférieurs à 10^9).

III. Test de primalité de Miller-Rabin

Le test de primalité de Miller-Rabin est une « amélioration » de celui de Fermat. Il se base sur une remarque assez simple :
Si n impair, alors on peut écrire (n-1) sous la forme : $n-1=2^s\cdot d$ avec d impair.
On peut donc factoriser le petit théorème de Fermat :

$$\begin{aligned} a^{n-1}\equiv 1 \bmod n \quad &\iff a^{2^s d}-1\equiv 0 \bmod n \\ &\iff (a^{2^{s-1}d}+1)(a^{2^{s-1}d}-1)\equiv 0 \bmod n \\ &\iff (a^{2^{s-1}d}+1)(a^{2^{s-2}d}+1)(a^{2^{s-2}d}-1)\equiv 0 \bmod n \\ &\ \ \vdots \\ &\iff (a^{2^{s-1}d}+1)(a^{2^{s-2}d}+1)\cdots(a^{d}+1)(a^{d}-1)\equiv 0 \bmod n \end{aligned}$$

Si n est premier, alors n doit diviser un de ces facteurs (car on aura montré que ce facteur est congru à 0 mod n donc que l'ensemble est congru à 0 mod n, donc que $a^{n-1}\equiv 1[n]$ car c'est équivalent). On teste donc cela pour plusieurs bases « a » avec 2 ≤ a ≤ n-2. S'il existe une base telle qu'aucun des facteurs ne soit congru à 0 mod n, alors on a la PREUVE que n est composé.

Le programme obtient très rapidement de très bons résultats en fonction du nombre d'itérations (nombre de bases aléatoires choisies). Soit N ce nombre de bases, alors la probabilité que l'algorithme renvoie le mauvais résultat est de 4^{-N}, donc décroît exponentiellement.