

# Tests de primalité

Robert CONTIGNON      Paulo PORTINHA

UNIVERSITÉ DU LUXEMBOURG -  
Année académique 2016 - 2017 (Semestre d'été)

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Les tests de primalité</b>	<b>3</b>
2.1	Notions utiles préparatoires . . . . .	3
2.2	Les tests de primalité . . . . .	4
2.2.1	Le test de Lucas-Lehmer . . . . .	4
2.2.2	Le test de Miller-Rabin . . . . .	8
2.2.3	Le test AKS . . . . .	12
<b>3</b>	<b>Étude informatique des tests de primalité</b>	<b>14</b>
3.1	Le test de Lucas-Lehmer . . . . .	14
3.1.1	Code source . . . . .	15
3.1.2	Exemples . . . . .	15
3.1.3	Analyse de la rapidité de l'algorithme . . . . .	17
3.2	Le test de Miller-Rabin . . . . .	18
3.2.1	Code source . . . . .	18
3.2.2	Exemples . . . . .	18
3.2.3	Analyse de la rapidité de l'algorithme . . . . .	21
3.3	Le test AKS . . . . .	22
3.3.1	Code source . . . . .	22
3.3.2	Exemples . . . . .	23
3.3.3	Analyse de la rapidité de l'algorithme . . . . .	24
<b>4</b>	<b>Conclusion</b>	<b>25</b>
<b>5</b>	<b>Bibliographie</b>	<b>26</b>

# 1 Introduction

Nous rencontrons les nombres premiers très tôt dans notre vie : À l'école primaire déjà, l'on apprend la notion de nombre premier (un nombre naturel est premier s'il n'a que 2 diviseurs : 1 et lui-même) et on dresse la liste des nombres premiers inférieurs à 100. Puis au lycée, on découvre qu'il existe plein d'autres nombres premiers que ceux rencontrés à l'école primaire et l'on apprend ce que les mathématiciens appellent le théorème fondamental de l'algèbre (ici on considère une version simplifiée du théorème) : Tout nombre naturel se décompose de façon unique en produit fini de puissances naturelles de nombres premiers. Ainsi, toute personne qui finit la scolarité obligatoire a au moins une fois de sa vie entendu parler des nombres premiers.

Mais beaucoup d'entre nous ont dû se demander à l'époque comment on peut trouver des nombres premiers qui sont "très grands" à partir des quelques notions que l'on a apprises dans l'enseignement primaire et secondaire. Et bien, cette réponse a été fournie par plusieurs mathématiciens tout au fil des siècles. Néanmoins, la recherche d'algorithmes permettant de trouver des nombres premiers a connu une nette révolution avec l'apparition des ordinateurs. En effet, ces machines électroniques sont entre autres des super-calculatrices et sont capables d'effectuer une énorme quantité de calculs de en une seconde, ce qui a largement contribué à leur succès. C'est d'ailleurs grâce aux ordinateurs que beaucoup d'algorithmes concernant la recherche de nombres premiers, que l'on appelle communément les tests de primalité, ont vu le jour et ont pu être testés facilement.

À ce jour, il existe plusieurs tests de primalité, mais dans le cadre de notre projet, nous allons nous limiter à l'étude de 3 différents tests de primalité: le test de Lucas-Lehmer, de Miller-Rabin et AKS.

## 2 Les tests de primalité

### 2.1 Notions utiles préparatoires

Dans cette sous-section, nous allons regrouper toutes les notions vues jusqu'à présent à l'université qui serviront par la suite pour expliquer les divers tests et ainsi créer nos programmes informatiques.

**Définition 2.1.** On appelle **nombre premier** un nombre naturel qui n'est divisible que par 1 et par lui-même.

*Définition équivalente :* On appelle **nombre premier** un nombre naturel dont aucun des nombres naturels strictement compris entre 1 et ce nombre naturel en est un diviseur.

En langage mathématique, cette définition est donnée ainsi :  
Soit  $p \in \mathbb{N}^*$ . On dit que  $p$  est premier si  $\forall i \in [2, 3, \dots, p-1] : i \nmid p$ .

**Définition 2.2.** On appelle **test de primalité** un algorithme mathématique (informatique) qui traite un ou plusieurs nombres naturels non nuls et qui dit, en retournant soit la valeur *True* soit la valeur *False*, si ce ou ces nombres sont premiers.

**Théorème 2.1.** *Il existe une infinité de nombres premiers.*

**Définition 2.3.** Soit  $n$  un entier naturel. Deux entiers relatifs  $a$  et  $b$  sont dits **congrus modulo  $n$**  si leur différence est divisible par  $n$ , c'est-à-dire si  $a$  est de la forme  $b + kn$  avec  $k$  entier.

Si  $n=0$ , alors la congruence entre  $a$  et  $b$  devient simplement une égalité.

Deux entiers  $a$  et  $b$  sont alors congrus modulo  $n$  si et seulement si le reste de la division euclidienne de  $a$  par  $n$  est égal à celui de la division de  $b$  par  $n$ .

On note la **congruence** à l'aide du symbole  $\equiv$ . Ainsi,  $a$  et  $b$  sont congrus modulo  $n$  peut s'écrire sous la forme suivante :  $a \equiv b \pmod{n}$ .

## 2.2 Les tests de primalité

### 2.2.1 Le test de Lucas-Lehmer

Le test de Lucas-Lehmer est l'un des premiers véritables tests de primalité. Il a été mis en oeuvre par le mathématicien français Édouard Lucas en 1878 et a été modifié en une version plus forte par Derrick Lehmer, mathématicien américain important du  $XX^{me}$  siècle, environ 60 ans plus tard. Les 2 ayant travaillé sur ce même test malgré qu'ils n'étaient pas de la même époque, ce test porte jusqu'à ce jour le nom de test de Lucas-Lehmer. Ce test est un test que l'on dénomme non généraliste et déterministe, car il permet de trouver, et ce avec une certitude absolue, uniquement les nombres premiers de Mersenne dont l'explication sera fournie un peu plus loin dans ce paragraphe.

Avant d'expliquer le principe de ce test, nous allons d'abord brièvement traiter la version originale du test de Lucas. Le mathématicien français  $a$ , en 1878, découvrit un test de primalité basé sur le petit théorème de Fermat, mais l'a amélioré en 1891. Voici donc la vraie version du test de Lucas :

**Théorème 2.2.** *Soit  $p > 1$ ,  $p \in \mathbb{N}$ . Supposons qu'il existe un  $a > 1$  ( $a \in \mathbb{N}$ ) tel que :*

1.  $a^{N-1} \equiv 1 \pmod{p}$
2.  $a^m \not\equiv 1 \pmod{p} \forall m < p$ , tel que  $m|(p-1)$

*Alors  $p$  est premier.*

*Proof.* Il suffit de montrer que tout entier  $m$ ,  $1 \leq m < p$ , est premier à  $p$ , soit,  $\varphi(p) = p - 1$ . Pour cette tâche, il suffit de montrer qu'il existe  $a$ ,  $1 \leq a < p$ ,  $\text{pgcd}(a, p) = 1$ , tel que l'ordre de  $a \bmod p$  est  $p-1$ . C'est exactement ce qu'on déchiffre dans l'hypothèse. □

Même si ce test a l'air efficace vu sa simplicité, le principal problème est que  $\forall p$ , il faut toujours connaître la décomposition en facteurs premiers de  $p - 1$ , ce qui devient rapidement problématique pour les grands nombres.

Pour remédier à ce problème, il a fallu attendre environ 40 ans avant que Derrick Lehmer, fameux mathématicien américain du  $XX^{me}$  siècle, poursuive le travail de Lucas jusqu'à donner une version plus puissante du test de Lucas. Mais avant d'en arriver à l'énoncé du test de Lucas-Lehmer, nous allons introduire la notion de nombre premier de Mersenne (mathématicien et religieux érudit français du  $XVII^{me}$  siècle), notion sur laquelle le test est essentiellement basé.

**Définition 2.4.** • Un **nombre de Mersenne** est un terme de la suite  $(M_n)_{n \in \mathbb{N}} = 2^n - 1$ .

- Un **nombre premier de Mersenne** est un nombre  $2^p - 1$  où  $p$  est premier. On le note  $M_p$ .

À présent, on peut énoncer le principe du test de Lucas-Lehmer :

**Théorème 2.3.** (*Théorème/Test de Lucas-Lehmer*)

Soit la suite d'entiers  $(s_i)_{i \geq 0}$  définie par récurrence de la façon suivante :

$$\begin{cases} s_0 & = & 4 \\ s_{i+1} & = & (s_i)^2 - 2 \end{cases}$$

Soit  $p$  un nombre premier impair.

Alors le nombre de Mersenne  $M_p = 2^p - 1$  est premier  $\leftrightarrow M_p | (s_{p-2})$ .

Le nombre  $(s_{p-2}) \bmod M_p$  est appelé le résidu de Lucas-Lehmer de  $p$ .

La réciproque de ce théorème est aussi vraie.

*Proof.* Avant d'entamer la preuve du théorème, nous allons énoncer (sans démonstration) quelques assertions intermédiaires dont on aura besoin par la suite pour la preuve.

**Lemme 2.1.** Soit  $0, 1 \neq a \in \mathbb{Z}$  sans facteur carré.

1. L'application  $\varphi : \mathbb{Z}[X] \rightarrow \mathbb{Z}[\sqrt{a}]$ ,  $f(X) \mapsto f(\sqrt{a})$  est un épimorphisme d'anneau avec  $\ker \varphi = (X^2 - a)$ .

2. Soit  $M \in \mathbb{N} \geq 1$  un entier positif et notons par  $I$  l'idéal principal  $I = M \cdot \mathbb{Z}[\sqrt{a}] \leq \mathbb{Z}[\sqrt{a}]$ .  
Alors on a un isomorphisme d'anneau naturel  $(\mathbb{Z}/M\mathbb{Z})[X]/(X^2 - a) \simeq \mathbb{Z}[X]/(M, X^2 - a) \xrightarrow{\overline{f(X)} \mapsto f(\sqrt{a})} \mathbb{Z}[\sqrt{a}]/I$ , où  $a$  est la classe modulo  $M$ .
3. Soit  $M \geq 3$  un nombre premier tel que  $(\frac{a}{m}) = -1$ . Alors  $\mathbb{Z}[\sqrt{a}]/I$  est un corps fini avec  $M^2$  éléments.

**Théorème 2.4.** (Loi de réciprocité quadratique)

Soit  $p \neq q$  2 nombres premiers impairs distincts.

1.  $(\frac{-1}{p}) = (-1)^{\frac{p-1}{2}} = \{1 \text{ si } p \equiv 1 \pmod{4}, -1 \text{ si } p \equiv 3 \pmod{4}\}$ .
2.  $(\frac{2}{p}) = (-1)^{\frac{p^2-1}{8}} = \{1 \text{ si } p \equiv 1, 7 \pmod{8}, -1 \text{ si } p \equiv 3, 5 \pmod{8}\}$ .
3.  $(\frac{q}{p}) = (\frac{p}{q}) \cdot (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}$ . En particulier, si  $p \equiv 1 \pmod{4}$  ou  $q \equiv 1 \pmod{4}$ , alors  $(\frac{q}{p}) = (\frac{p}{q})$ .

**Proposition 2.1.** (Euler)

Soit  $p > 2$  un nombre premier et  $a \in \mathbb{Z}$ . Alors on a la congruence  $(\frac{a}{p}) \equiv a^{\frac{p-1}{2}} \pmod{p}$

À présent, nous pouvons démontrer le théorème de Lucas-Lehmer :

On travaille dans l'anneau  $\mathbb{Z}[\sqrt{3}]$  et un anneau quotient quelconque. Écrivons  $\omega = 2 + \sqrt{3}, \bar{\omega} = 2 - \sqrt{3} \in \mathbb{Z}[\sqrt{3}]$ . Nous avons  $\omega \cdot \bar{\omega} = 4 - 3 = 1$  (tel que  $\omega$  est une unité) et  $\omega = \frac{1}{2} \cdot (1 + \sqrt{3})^2$ .

$$\text{Montrons d'abord par récurrence : } S_k = \omega^{2^{k-1}} + \bar{\omega}^{2^{k-1}} \quad (E_1)$$

Le cas  $k = 1$  est simplement l'égalité  $S_1 = 4 = \omega + \bar{\omega}$ .

Le cas général est le calcul suivant :

$$S_{k+1} = (S_k)^2 - 2 = (\omega^{2^{k-1}} + \bar{\omega}^{2^{k-1}})^2 - 2 = \omega^{2^k} + \bar{\omega}^{2^k} + 2(\omega \cdot \bar{\omega})^{2^{k-1}} - 2 = \omega^{2^k} + \bar{\omega}^{2^k}.$$

$\longleftarrow$  :

Supposons  $M = M_p | S_{p-1}$ , mais que  $M$  n'est pas un nombre premier. Soit alors  $q | M$  un nombre premier tel que  $q \leq \sqrt{M} = \sqrt{2^p - 1} < 2^{\frac{p}{2}}$ .

La supposition  $M | S_{p-1}$  combinée avec  $(E_1)$  implique  $S_{p-1} = \omega^{2^{p-2}} + \bar{\omega}^{2^{p-2}} \equiv 0 \pmod{M \cdot \mathbb{Z}[\sqrt{3}]}$ , à partir duquel on conclue  $\omega^{2^{p-2}} \equiv -\bar{\omega}^{2^{p-2}} \pmod{M \cdot \mathbb{Z}[\sqrt{3}]}$  et par conséquent  $\omega^{2^{p-1}} \equiv -1 \pmod{M \cdot \mathbb{Z}[\sqrt{3}]}$ .

Ceci implique en particulier  $\omega^{2^{p-1}} \equiv -1 \pmod{q \cdot \mathbb{Z}[\sqrt{3}]}$ , d'où l'image de  $\omega$  est d'ordre  $2^p$  dans  $(\mathbb{Z}/q\mathbb{Z})[\sqrt{3}]$ . Ainsi,  $2^p < q^2 < (2^{\frac{p}{2}})^2 = 2^p$ , ce qui est la contradiction recherchée.

$\rightarrow$  : Nous supposons maintenant que  $M = M_p$  est un nombre premier. On peut montrer que  $\left(\frac{3}{2^{p-1}}\right) = \frac{3}{M} = -1$ , d'où le lemme précédent montre que  $R = \mathbb{Z}[\sqrt{3}]/M \cdot \mathbb{Z}[\sqrt{3}] \simeq \mathbb{F}_M[X]/(X^2 - \bar{3})$  est le corps avec  $M^2$  éléments.

Montrons d'abord la congruence  $\omega^{2^{p-1}} \equiv -1 \pmod{M \cdot \mathbb{Z}[\sqrt{3}]}$  ( $E_2$ ).

La définition  $M = 2^p - 1$  peut être reformulée ainsi :  $2^{p-1} = \frac{M+1}{2}$ . Par la proposition d'Euler et la loi de réciprocité quadratique, on a  $\left(\frac{1}{2}\right)^{\frac{M-1}{2}} \equiv \left(\frac{2}{M}\right) \equiv 1 \pmod{M}$ , vu que  $M \equiv -1 \pmod{8}$ .

En plus de cela, par le fait mathématique que l'on a utilisé auparavant au début de cette partie de preuve, on voit que  $3^{\frac{M-1}{2}} \equiv \left(\frac{3}{M}\right) \equiv -1 \pmod{M}$ .

On effectue le calcul suivant dans le corps  $R$ , qui est de caractéristique  $M$  (tel que  $(a+b)^M = a^M + b^M$ ).

$$\begin{aligned} \omega^{2^{p-1}} &= \left(\frac{1}{2} \cdot (1 + \sqrt{3})^2\right)^{2^{p-1}} = \left(\frac{1}{2}\right)^{\frac{M+1}{2}} \cdot (1 + \sqrt{3})^{M+1} = \left(\frac{1}{2}\right)^{\frac{M-1}{2}} \cdot \frac{1}{2} \cdot (1 + \sqrt{3}) \cdot (1 + \sqrt{3})^M \\ &= \frac{1}{2} \cdot (1 + \sqrt{3}) \cdot (1 + \sqrt{3})^M = \frac{1}{2} \cdot (1 + \sqrt{3}) \cdot (1 + \sqrt{3}^M) = \frac{1}{2} \cdot (1 + \sqrt{3}) \cdot (1 + 3^{\frac{M-1}{2}} \cdot \sqrt{3}) \\ &= \frac{1}{2} \cdot (1 + \sqrt{3}) \cdot (1 - \sqrt{3}) = -1. \end{aligned}$$

On utilise à présent ( $E_2$ ) pour déduire que  $-\bar{\omega}^{2^{p-2}} = \omega^{2^{p-1}} \cdot \bar{\omega}^{2^{p-2}} = \omega^{2^{p-2}} \cdot \omega^{2^{p-2}} \cdot \bar{\omega}^{2^{p-2}} = \omega^{2^{p-2}}$  et ainsi le tant recherché  $s_{p-1} = \omega^{2^{p-2}} + \bar{\omega}^{2^{p-2}} = 0$  dans le corps  $R$ . □

Le test de Lucas-Lehmer a pour particularité de fournir tous les nombres premiers de Mersenne vu que l'on teste en fait chaque nombre de Mersenne. Toutefois, ce test a l'inconvénient de ne fournir que ces nombres particuliers et omet ainsi plein d'autres nombre premiers. Mais en gros, ce test est pratique de part sa précision et son exactitude.

Grâce à ce théorème, on peut maintenant déduire un algorithme mathématique que l'on utilisera dans la prochaine section.

Décrivons-le en termes de phrases et par étape :

1. Choisir un nombre naturel  $p$  qui est à la fois impair et premier.
2. Calculer le nombre de Mersenne  $M_p$  correspondant.
3. Calculer  $s_{p-2}$  à l'aide de la boucle définie comme suit :
  - (a) Initialiser la variable  $s_i$  en lui affectant la valeur 4, où  $i$  est une variable naturelle.
  - (b) Créer la boucle de telle manière qu'elle parcourt l'indice  $i$  de 0 à  $p - 2$ .

- (c) Dans le corps de la boucle, l'on utilisera la relation de récurrence fournie par le théorème 2.3 afin d'effectuer les calculs des  $s_i$  jusqu'à avoir la valeur de  $s_{p-2}$ .
4. Calculer la division  $\frac{s_{p-2}}{M_p}$  et évaluer son reste. S'il vaut 0, alors  $M_p$  est un nombre premier (de Mersenne), sinon c'en n'est pas un.

### 2.2.2 Le test de Miller-Rabin

Le test de Miller-Rabin est un test qui a été en grande partie développé par le mathématicien américain Gary Miller jusqu'en 1976 et dont certaines notions proviennent de celles employées par le mathématicien polonais d'origine israélienne Rabin, d'où le nom de ces 2 hommes pour ce test de primalité. Le test de Miller-Rabin est un test de primalité dit "probabiliste" en raison de l'utilisation de l'hypothèse de Riemann généralisée (par Rabin, car Miller avait à l'origine créé un test déterministe) qui n'a pas été vraiment démontrée et que l'on évoquera seulement ici pour comprendre le contexte dans lequel on se trouve. Puisque ce test est basé sur une hypothèse non-démontrée, les résultats provenant du test de Miller-Rabin sont vrais avec une certaine probabilité. Plus précisément, si un certain  $n$  naturel est premier, alors il l'est probablement. Sinon, il ne l'est pas avec une certitude absolue.

Le test de Miller-Rabin utilise plein de notions nouvelles que nous allons immédiatement définir en vue de comprendre le théorème qui décrit ce test.

**Définition 2.5.** Un **nombre pseudo-premier** est un nombre premier probable (un entier naturel qui partage une propriété commune à tous les nombres premiers) qui n'est en fait pas premier (probabilité d'être premier est comprise entre 0 et 1). Les nombres pseudo-premiers peuvent être classés selon la propriété qu'ils satisfont.

Citons par exemple la famille des nombres pseudo-premiers de Mersenne. La propriété satisfaite par ces nombres est qu'ils sont tous des termes de la suite de Mersenne.

**Définition 2.6.** Soit  $N$  un naturel,  $N - 1 = 2^s d$ , avec  $s \geq 0, d$  impair. Soit  $1 < a < N$  avec  $\text{pgcd}(a, N) = 1$ . On dit alors que  $a$  est un **témoin de Miller** pour  $N$  si  $a^d \not\equiv 1 \pmod{N}$  et  $a^{2^r d} \not\equiv 1 \pmod{N} \forall r, 0 \leq r < s$ .

**Définition 2.7.** Un **nombre composé** est un entier naturel différent de 0 qui possède un diviseur positif autre que 1 ou lui-même. Par définition, chaque entier plus grand que 1 est donc soit un nombre premier, soit un nombre composé, et les nombres 0 et 1 ne sont ni premiers ni composés.

Autre définition : Un nombre composé est le produit d'au moins deux nombres premiers (qu'ils soient distincts ou identiques).

Le théorème suivant établit les relations entre les 3 notions précédemment introduites. Il servira à mieux comprendre le test de Miller-Rabin.

**Théorème 2.5.** *Si  $a$  est un témoin de Miller, alors  $N$  est composé.*

*Si  $N$  est composé, si  $1 < a < N$ , si  $\text{pgcd}(a,N)=1$  et si  $a$  n'est pas un témoin de Miller, alors  $N$  est un pseudopremier.*

*Réciproquement, si  $N$  est impair et  $N$  est un pseudopremier, alors  $a$  n'est pas un témoin de Miller pour  $N$ .*

Nous présentons maintenant une proposition qui permet de déduire la méthode principale du test de Miller-Rabin pour tester la primalité d'un naturel non nul quelconque, à savoir trouver plusieurs et suffisamment de témoins de Miller pour montrer que le nombre concerné est composé respectivement pseudopremier.

**Proposition 2.2.** *Pour un nombre impair composé  $N$ ,  $\frac{3}{4}$  au moins des entiers  $a$ ,  $1 < a < N$ , sont des témoins de Miller pour  $N$ .*

Nous pouvons à présent énoncer le théorème de Miller qui est à l'origine du test de Miller :

**Théorème 2.6.** *Soit  $N$  un naturel impair. S'il existe  $a$  tel que  $\text{pgcd}(a, N) = 1$ ,  $1 < a < 2(\log N)^2$ , qui est un témoin de Miller pour  $N$ , alors  $N$  est composé. Sinon,  $N$  est premier.*

Étant donné que ce test est déterministe, il ne correspond pas encore tout à fait au test de Miller-Rabin, et ce pour la simple raison que Rabin a formulé un autre théorème qui a rendu le test de Miller probabiliste, ce qui a donné naissance au fameux test de Miller-Rabin.

**Théorème 2.7.** *Soit  $n$  un entier impair composé  $> 9$ , avec  $n - 1 = 2^s \cdot d$  pour  $d$  impair. Alors il existe au plus  $\frac{\varphi(n)}{4}$  menteurs forts  $a$  associés au pseudo-premier  $n$  ( $\text{pgcd}(a,n)=1$ ), pour  $1 < a < n$ , c'est-à-dire des entiers  $a$  dans cet intervalle vérifiant soit  $a^d \equiv 1 \pmod n$ , soit,  $a^{d2^r} \equiv -1 \pmod n$  pour un certain  $r$  tel que  $0 \leq r < s$  ( $\varphi$  est la fonction indicatrice d'Euler).*

*La réciproque de ce théorème est également vraie.*

*Proof.* Comme dans la preuve du test de Lucas-Lehmer, nous allons ici d'abord énumérer quelques assertions ainsi qu'une définition utile dans le cadre de cette preuve, et en admettant toutes ces assertions vraies sans les démontrer :

**Théorème 2.8.** *(Théorème des restes chinois)*

*Soient  $n_1, \dots, n_k$  des entiers 2 à 2 premiers entre eux.*

*Alors pour tout entier  $a_1, \dots, a_k$ , il existe un entier  $x$ , unique modulo  $n = \prod_{i=1}^k n_i$ , tel que  $x \equiv a_i \pmod{n_i} \forall i \in \{1, 2, \dots, k\}$ .*

Une autre version existe lorsque l'on travaille dans l'anneau  $\mathbb{Z}/N\mathbb{Z}$ . La voici :

Si  $n_1, \dots, n_k$  sont 2 à 2 premiers entre eux, alors, en notant  $n$  le PPCM des  $n_i$ , c'est-à-dire dans le cas présent le produit des  $n_i$ , l'application (à valeurs dans l'anneau produit) :

$$\phi : \mathbb{Z} \rightarrow \mathbb{Z}/(n_1)\mathbb{Z} \times \dots \times \mathbb{Z}/(n_k)\mathbb{Z}$$

$$\alpha[n] \mapsto (\alpha[n_1], \dots, \alpha[n_k])$$

est un isomorphisme d'anneaux.

**Théorème 2.9.** (Application du théorème des restes chinois (TRC))

Soit  $N = (p_1)^{e_1} \cdot \dots \cdot (p_k)^{e_k}$  un entier impair dans sa décomposition en facteurs premiers. Alors dans  $\mathbb{Z}/N\mathbb{Z}$  l'équation  $X^2 - 1$  a  $2^k$  solutions : à savoir, sous le TRC  $\mathbb{Z}/N\mathbb{Z} \simeq \mathbb{Z}/(p_1)^{e_1}\mathbb{Z} \times \dots \times \mathbb{Z}/(p_k)^{e_k}\mathbb{Z}$  les solutions sont précisément données par les éléments  $(a_1, \dots, a_k)$  avec  $a_i \in \{1, -1\} \forall 1 \leq i \leq k$  (notons que dans les anneaux  $\mathbb{Z}/(p_i)^{e_i}\mathbb{Z}$ , l'équation  $X^2 - 1 = 0$  a exactement 1 et -1 comme solutions vu que le groupe des unités est cyclique par le lemme introduit dans la preuve du théorème de Lucas-Lehmer.

Ainsi, on obtient le critère de primalité suivant :

$N$  est premier  $\leftrightarrow X^2 - 1 = 0$  admet seulement 1, -1 comme solutions dans  $\mathbb{Z}/N\mathbb{Z}$ .

**Définition 2.8.** Un nombre naturel  $n \in \mathbb{N}$  est appelé **nombre de Carmichael** si  $n$  n'est pas premier et  $a^{n-1} \equiv 1 \pmod n \forall a \in \mathbb{Z}$  tel que  $\gcd(a, n) = 1$ .

**Proposition 2.3.** Soit  $N \in \mathbb{N}_{\geq 2}$ . Les assertions suivantes sont équivalentes :

1.  $N$  est premier ou un nombre de Carmichael.
2. L'exposant de  $(\mathbb{Z}/N\mathbb{Z})^\times$  est un diviseur de  $N - 1$ .
3. Pour tout nombre premier  $p$  tel que  $p$  divise  $N$  :
  - $p^2 \nmid N$  (nombres qui ne sont pas divisible par le carré d'un quelconque nombre premier sont appelés sans facteur carré)
  - $(p - 1) | (N - 1)$ .

**Proposition 2.4.** Pour tout nombre de Carmichael est impair et a au moins 3 diviseurs premiers.

Nous pouvons maintenant passer à la preuve du théorème de Miller-Rabin:

$\rightarrow$  :

Supposons que  $N$  est un nombre premier et soit  $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ . Écrivons  $r = \text{ord}(a)$ , c'est un diviseur de  $N - 1$ , qui est l'ordre de  $(\mathbb{Z}/N\mathbb{Z})^\times$ .

1<sup>er</sup> cas :  $r$  est impair

Alors  $r|m$  et par conséquent  $a^m \equiv 1 \pmod{N}$ .

2<sup>nd</sup> cas :  $r$  est pair

Alors  $r = 2^s \cdot r'$  avec un nombre impair  $r' \in \mathbb{N}$  et nécessairement  $r'|m$ .  
Puisque  $(a^{2^{s-1} \cdot r'})^2 \equiv 1 \pmod{N}$ , il s'ensuit que  $a^{2^{s-1} \cdot r'} \equiv -1 \pmod{N}$ , parce que dans un corps 1 et -1 sont les uniques éléments dont le carré est 1.

← :

Supposons que  $N$  n'est pas premier. Dans tout les cas, on obtient en élevant au carré que  $a^{N-1} \equiv 1 \pmod{N} \forall a \in (\mathbb{Z}/N\mathbb{Z})^\times$ , d'où  $N$  est un nombre de Carmichael et ainsi sans facteur carré,  $N = p_1 \cdot \dots \cdot p_k$  avec au moins 3 nombres premiers impairs distincts  $p_1, \dots, p_k$  tels que  $(p_i - 1) | (N - 1) \forall 1 \leq i \leq k$  par les 2 propositions précédentes.

On utilise à présent le théorème des restes chinois (TRC) en vue de construire un  $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ , violant ainsi toutes les conditions. Par l'isomorphisme  $\mathbb{Z}/N\mathbb{Z} \simeq \mathbb{Z}/p_1\mathbb{Z} \times \dots \times \mathbb{Z}/p_k\mathbb{Z}$  à partir du TRC on définit  $a$  en tant que  $(-1, 1, 1, \dots, 1)$ . Puisque  $m$  est impair, il s'ensuit que  $a^m \neq 1$  dans  $(\mathbb{Z}/N\mathbb{Z})^\times$  car la première composante reste égale à -1 ; d'où la première condition est violée. Il est également évident que la seconde condition n'est pas respectée (étant donné que -1 n'est pas une puissance de 1).

□

Remarque : menteur fort = "opposé" de témoin de Miller

Grâce à ce théorème et aux assertions et définitions de cette section, on peut en déduire un algorithme mathématique qui nous permettra dans la prochaine section de programmer le test de Miller-Rabin.

Décrivons-le, tout comme dans le test de Lucas-Lehmer, à l'aide de phrases et par étapes :

1. Choisir un nombre naturel  $p$  impair non nul quelconque ( $p$  ne doit pas forcément être composé et  $>9$  à l'avance, car on ne sait pas toujours si c'est le cas, surtout pour les très grands nombres).
2. Résoudre l'équation de l'énoncé du théorème 2.6 d'inconnues  $s$  et  $d$ , sachant que  $s > 0$  et  $d$  est impair :

$$p - 1 = 2^s \cdot d$$

3.  $\forall 1 < a < p$ , vérifier si  $a$  est témoin de Miller ou pas.
4. Si on en a suffisamment assez (minoré par la Proposition 2.1), alors d'après les théorèmes 2.4 et 2.5,  $p$  est composé.

5. Sinon,  $p$  est probablement premier, c'est-à-dire pseudo-premier.

### 2.2.3 Le test AKS

Le test AKS, dont les lettres correspondent au nom de famille des 3 mathématiciens indiens Agrawal, Kayal et Saxena ayant formulé ce test, est l'un des tests de primalité les plus récents de l'histoire mathématique. En effet, il a vu le jour au début des années 2000. Ce test a pour énorme avantage que, contrairement au test de Miller-Rabin dont on a parlé dans la sous-section précédente, c'est un test déterministe. En d'autres mots, le test de primalité AKS dit avec certitude (probabilité égale à 1) si un nombre naturel quelconque est premier ou pas. En plus de cela, il fonctionne pour n'importe quel naturel non nul.

Tout comme le test de Miller-Rabin, le test AKS dit si un nombre naturel donné est premier ou composé.

En ce qui concerne le principe de ce test, il est surtout basé sur le théorème suivant (généralisation du petit théorème de Fermat) :

**Théorème 2.10.** *Soit  $n \geq 2, n \in \mathbb{N}$ . Soit  $a \in \mathbb{N}$  tel que  $\text{pgcd}(a, n) = 1$  (on dit alors que  $a$  est premier à  $n$ ).*

*Alors :  $n$  est premier  $\leftrightarrow (X + a)^n \equiv (X^n + a) \pmod{n}$ .*

Ce théorème est certes très utile vu qu'on le qualifie de tel, mais n'empêche que si l'on utilise ce résultat dans l'algorithme, le programme ne sera pas efficace en raison du temps de calcul nécessaire de tous les coefficients (il y en a  $n$  par  $(X + a)^n$ ) pour comparer et ainsi savoir si  $n$  est premier ou composé. C'est donc pour cette raison que la généralisation du petit théorème de Fermat a été réétudiée de manière approfondie dans le but de déduire un nouveau théorème ou simplement une variante, mais qui allège nettement le temps de calcul dont on a parlé auparavant.

Ceci a été réalisé avec succès par 2 mathématiciens indiens, qui ne sont ni Agrawal ni Saxena et ni Kayal. Les 2 hommes ont modifié le théorème 2.10 de manière à obtenir une nouvelle formulation de ce théorème, mais avec la principale différence que le temps de calcul est clairement réduit par rapport à la version initiale du théorème en question.

La nouvelle formulation du théorème 2.10 est la suivante :

**Théorème 2.11.** *Soit  $n \geq 2, n \in \mathbb{N}$ . Soit  $a \in \mathbb{N}$  tel que  $\text{pgcd}(a, n) = 1$ .*

*Alors  $n$  est premier  $\leftrightarrow \exists r, r \leq n$  tel que  $(X + a)^n \equiv X^n + a \pmod{X^r - 1, n}$ .*

Cette version de la généralisation du petit théorème de Fermat a été à cette période une avancée décisive dans la réalisation du test de primalité AKS avec comme seul et sacré problème qu'elle n'a pas été démontrée. Ceci

a été le travail de Kayal et Saxena qui l'ont prouvé peu de temps après la publication du résultat.

À partir de cette version de théorème, ces 2 mathématiciens indiens avaient à l'époque déduit une assertion particulièrement importante. La voici :

**Proposition 2.5.** *Soit  $n \in \mathbb{N}, r \in \mathbb{N}$  tel que  $r \leq n$ . Si  $r \nmid n$  et si  $(X+1)^n \equiv X^n + 1 \pmod{X^r - 1, n}$ , alors  $n$  est soit premier soit on a  $n^2 \equiv 1 \pmod{r}$ .*

Grâce au théorème 2.11, à la dernière proposition et au cas spécial, on peut rédiger un algorithme pour le test de primalité AKS (l'étude détaillée se fera au cours de la prochaine section). Néanmoins, avant d'en arriver au pseudo-code source, on va d'abord éliminer un cas spécial pour lequel  $n$  est composé.

Le voici :

Si  $n = m^k$  ( $m \in \mathbb{N}, m \geq 2, k \in \mathbb{N}^* - \{1\}$ ), alors  $n$  est composé.

En plus de cela, il est primordial de savoir que le trio de mathématiciens indiens est également parvenu à prouver que si  $n$  est composé, alors  $\exists r \leq (\log_2(n))^5, \exists a \leq \sqrt{\varphi_{Euler}(r) \log_2(n)^2}$  tels que l'équation dans le théorème 2.11 n'est pas vérifiée. La fonction  $\varphi_{Euler}(r)$  est appelée indicatrice d'Euler et associe au nombre  $r$  le nombre d'entiers entre 1 et  $r$  inclus qui sont premiers à  $r$ . Ceci est une assertion qui découle entre autre des théorèmes 2.10 et 2.11.

Le test AKS donne, contrairement au test de Lucas-Lehmer, tous les nombres premiers possibles. Décrivons son algorithme en mots et phrases et par étapes :

1. Choisir un quelconque nombre naturel  $p$  supérieur ou égal à 2.
2. On définit une fonction qui effectue le test AKS. Décrivons-la en détail :
  - (a) S'il existe un nombre naturel  $q$  supérieur ou égal à 2 et un  $k \in \mathbb{N}^* - \{1\}$  tels que  $p = q^k$ , alors  $n$  est composé.
  - (b) Sinon, on sort de cette instruction conditionnelle et on cherche le plus petit  $r$  tel que  $p^i = 1 \pmod{r}$  avec  $1 \leq i \leq B, B$  est le plus grand entier inférieur ou égal à  $(\log_2(p))^2$  et  $r \geq 2$  majoré par le plus grand entier inférieur ou égal à  $(\log_2(p))^5$ .
  - (c) Étudier ensuite un cas particulier où un nombre donné peut être composé ; pour chaque nombre, appelons-le  $a$ , entre 1 et  $r$  inclus, on regarde si le pgcd de  $a$  et  $p$  sont coprimiers. Si oui, alors  $p$  est composé.
  - (d) Étudier encore un cas spécial, à savoir si  $p$  est inférieur ou égal à  $r$ . Si c'est le cas, alors  $p$  est premier.

- (e) Définir une borne sur la future variable  $a$  en lui affectant la valeur  $\sqrt{\varphi_{Euler}(r) \log_2(p)^2}$ ,  $r$  étant un entier supérieur ou égal à 2.
  - (f) Créer une boucle **for** de la manière suivante :
  - (g) Calculer  $(x + a)^p \bmod (x^r - 1)$ .
  - (h) Calculer  $n \bmod r$ .
  - (i) Si le premier résultat est différent du second résultat, alors  $p$  est composé.
  - (j) Sinon, on continue de parcourir la boucle. Si l'on a parcouru toute la boucle sans vérifier une seule fois la condition précédente, alors  $p$  est premier.
3. Appliquer la fonction prédéfinie au nombre  $p$  choisi au départ. La fonction retourne donc soit le message *premier* soit le message *composé*.

### 3 Étude informatique des tests de primalité

Comme le titre de cette section l'indique, nous allons maintenant nous intéresser aux programmes informatiques sur les 3 tests de primalité que nous avons créés nous-même.

Pour chaque test de primalité, nous allons en premier lieu présenter le code source du programme sous forme de capture d'écran, le code source contenant également des explications des diverses étapes du programme. Ensuite, nous allons présenter 2 exemples en affichant le résultat final ; le premier exemple est celui d'un nombre qui est ou fournit un nombre premier tandis que le second est celui d'un nombre qui n'en est ou fournit pas un. Pour terminer l'étude, nous allons analyser la complexité informatique de chaque programme, plus clairement nous allons analyser la rapidité du programme en détaillant les endroits du programme où l'on perd du temps.

Enfin, nous précisons au début de chaque sous-section quel est le plus grand nombre premier que l'on a trouvé.

#### 3.1 Le test de Lucas-Lehmer

Le plus grand nombre premier de Mersenne que nous avons obtenu pendant un laps de temps correct (moins de 30 secondes) est  $2^{44497} - 1$ .

### 3.1.1 Code source

```
1 *
2 | #test de Lucas-Lehmer:
3 |
4 | 3
5 | 4
6 | 5 n=44497 #n doit être un entier naturel plus grand que 1.
7 | 6
8 | 7 def Lucas-Lehmer(p): #On définit une fonction qui effectue le test de primalité de Lucas-Lehmer
9 | 8 M_p = (2^p)-1 #M_p est le nombre de Mersenne de la variable p sur laquelle on veut faire le test de Lucas-Lehmer.
10 | 9 resultat = 4 #On initialise le nombre s_0 de la suite (s_n) du théorème de Lucas-Lehmer
11 | 10 for k in [1..(p-2)]: #On effectue les calculs de s_{(p-2)} et de s_{(p-2)}M_p en une ligne
12 | 11 resultat=(power_mod(resultat,2,M_p)-2)M_p
13 | 12 if resultat == 0: #Si le reste de cette dernière division est nul, alors on a le message ci-après
14 | 13 return 'Le nombre de Mersenne', M_p, 'est un nombre premier'
15 | 14 elif M_p == 3: #Sinon, si M_p = 3 (et donc si p=2), alors on a le message ci-après
16 | 15 else: #Sinon, on a le message ci-après
17 | 16 return 'Le nombre de Mersenne', M_p, 'n'est pas un nombre premier'
18 | 17
19 | 18 Lucas-Lehmer(n) #On applique le test de Lucas-Lehmer au nombre n choisi au début de ce programme
20 | 19
21 | 20
22 | 21
23 | 22
24 | 23
25 | 24
26 | 25
27 | 26
28 | 27
29 | 28
30 | 29
31 | 30
32 | 31
33 | 32
34 | 33
35 | 34
36 | 35
37 | 36
38 | 37
39 | 38
40 | 39
41 | 40
42 | 41
43 | 42
44 | 43
45 | 44
46 | 45
47 | 46
48 | 47
49 | 48
50 | 49
51 | 50
52 | 51
53 | 52
54 | 53
55 | 54
56 | 55
57 | 56
58 | 57
59 | 58
60 | 59
61 | 60
62 | 61
63 | 62
64 | 63
65 | 64
66 | 65
67 | 66
68 | 67
69 | 68
70 | 69
71 | 70
72 | 71
73 | 72
74 | 73
75 | 74
76 | 75
77 | 76
78 | 77
79 | 78
80 | 79
81 | 80
82 | 81
83 | 82
84 | 83
85 | 84
86 | 85
87 | 86
88 | 87
89 | 88
90 | 89
91 | 90
92 | 91
93 | 92
94 | 93
95 | 94
96 | 95
97 | 96
98 | 97
99 | 98
100 | 99
101 | 100
102 | 101
103 | 102
104 | 103
105 | 104
106 | 105
107 | 106
108 | 107
109 | 108
110 | 109
111 | 110
112 | 111
113 | 112
114 | 113
115 | 114
116 | 115
117 | 116
118 | 117
119 | 118
120 | 119
121 | 120
122 | 121
123 | 122
124 | 123
125 | 124
126 | 125
127 | 126
128 | 127
129 | 128
130 | 129
131 | 130
132 | 131
133 | 132
134 | 133
135 | 134
136 | 135
137 | 136
138 | 137
139 | 138
140 | 139
141 | 140
142 | 141
143 | 142
144 | 143
145 | 144
146 | 145
147 | 146
148 | 147
149 | 148
150 | 149
151 | 150
152 | 151
153 | 152
154 | 153
155 | 154
156 | 155
157 | 156
158 | 157
159 | 158
160 | 159
161 | 160
162 | 161
163 | 162
164 | 163
165 | 164
166 | 165
167 | 166
168 | 167
169 | 168
170 | 169
171 | 170
172 | 171
173 | 172
174 | 173
175 | 174
176 | 175
177 | 176
178 | 177
179 | 178
180 | 179
181 | 180
182 | 181
183 | 182
184 | 183
185 | 184
186 | 185
187 | 186
188 | 187
189 | 188
190 | 189
191 | 190
192 | 191
193 | 192
194 | 193
195 | 194
196 | 195
197 | 196
198 | 197
199 | 198
200 | 199
201 | 200
202 | 201
203 | 202
204 | 203
205 | 204
206 | 205
207 | 206
208 | 207
209 | 208
210 | 209
211 | 210
212 | 211
213 | 212
214 | 213
215 | 214
216 | 215
217 | 216
218 | 217
219 | 218
220 | 219
221 | 220
222 | 221
223 | 222
224 | 223
225 | 224
226 | 225
227 | 226
228 | 227
229 | 228
230 | 229
231 | 230
232 | 231
233 | 232
234 | 233
235 | 234
236 | 235
237 | 236
238 | 237
239 | 238
240 | 239
241 | 240
242 | 241
243 | 242
244 | 243
245 | 244
246 | 245
247 | 246
248 | 247
249 | 248
250 | 249
251 | 250
252 | 251
253 | 252
254 | 253
255 | 254
256 | 255
257 | 256
258 | 257
259 | 258
260 | 259
261 | 260
262 | 261
263 | 262
264 | 263
265 | 264
266 | 265
267 | 266
268 | 267
269 | 268
270 | 269
271 | 270
272 | 271
273 | 272
274 | 273
275 | 274
276 | 275
277 | 276
278 | 277
279 | 278
280 | 279
281 | 280
282 | 281
283 | 282
284 | 283
285 | 284
286 | 285
287 | 286
288 | 287
289 | 288
290 | 289
291 | 290
292 | 291
293 | 292
294 | 293
295 | 294
296 | 295
297 | 296
298 | 297
299 | 298
300 | 299
301 | 300
302 | 301
303 | 302
304 | 303
305 | 304
306 | 305
307 | 306
308 | 307
309 | 308
310 | 309
311 | 310
312 | 311
313 | 312
314 | 313
315 | 314
316 | 315
317 | 316
318 | 317
319 | 318
320 | 319
321 | 320
322 | 321
323 | 322
324 | 323
325 | 324
326 | 325
327 | 326
328 | 327
329 | 328
330 | 329
331 | 330
332 | 331
333 | 332
334 | 333
335 | 334
336 | 335
337 | 336
338 | 337
339 | 338
340 | 339
341 | 340
342 | 341
343 | 342
344 | 343
345 | 344
346 | 345
347 | 346
348 | 347
349 | 348
350 | 349
351 | 350
352 | 351
353 | 352
354 | 353
355 | 354
356 | 355
357 | 356
358 | 357
359 | 358
360 | 359
361 | 360
362 | 361
363 | 362
364 | 363
365 | 364
366 | 365
367 | 366
368 | 367
369 | 368
370 | 369
371 | 370
372 | 371
373 | 372
374 | 373
375 | 374
376 | 375
377 | 376
378 | 377
379 | 378
380 | 379
381 | 380
382 | 381
383 | 382
384 | 383
385 | 384
386 | 385
387 | 386
388 | 387
389 | 388
390 | 389
391 | 390
392 | 391
393 | 392
394 | 393
395 | 394
396 | 395
397 | 396
398 | 397
399 | 398
400 | 399
401 | 400
402 | 401
403 | 402
404 | 403
405 | 404
406 | 405
407 | 406
408 | 407
409 | 408
410 | 409
411 | 410
412 | 411
413 | 412
414 | 413
415 | 414
416 | 415
417 | 416
418 | 417
419 | 418
420 | 419
421 | 420
422 | 421
423 | 422
424 | 423
425 | 424
426 | 425
427 | 426
428 | 427
429 | 428
430 | 429
431 | 430
432 | 431
433 | 432
434 | 433
435 | 434
436 | 435
437 | 436
438 | 437
439 | 438
440 | 439
441 | 440
442 | 441
443 | 442
444 | 443
445 | 444
446 | 445
447 | 446
448 | 447
449 | 448
450 | 449
451 | 450
452 | 451
453 | 452
454 | 453
455 | 454
456 | 455
457 | 456
458 | 457
459 | 458
460 | 459
461 | 460
462 | 461
463 | 462
464 | 463
465 | 464
466 | 465
467 | 466
468 | 467
469 | 468
470 | 469
471 | 470
472 | 471
473 | 472
474 | 473
475 | 474
476 | 475
477 | 476
478 | 477
479 | 478
480 | 479
481 | 480
482 | 481
483 | 482
484 | 483
485 | 484
486 | 485
487 | 486
488 | 487
489 | 488
490 | 489
491 | 490
492 | 491
493 | 492
494 | 493
495 | 494
496 | 495
497 | 496
498 | 497
499 | 498
500 | 499
501 | 500
502 | 501
503 | 502
504 | 503
505 | 504
506 | 505
507 | 506
508 | 507
509 | 508
510 | 509
511 | 510
512 | 511
513 | 512
514 | 513
515 | 514
516 | 515
517 | 516
518 | 517
519 | 518
520 | 519
521 | 520
522 | 521
523 | 522
524 | 523
525 | 524
526 | 525
527 | 526
528 | 527
529 | 528
530 | 529
531 | 530
532 | 531
533 | 532
534 | 533
535 | 534
536 | 535
537 | 536
538 | 537
539 | 538
540 | 539
541 | 540
542 | 541
543 | 542
544 | 543
545 | 544
546 | 545
547 | 546
548 | 547
549 | 548
550 | 549
551 | 550
552 | 551
553 | 552
554 | 553
555 | 554
556 | 555
557 | 556
558 | 557
559 | 558
560 | 559
561 | 560
562 | 561
563 | 562
564 | 563
565 | 564
566 | 565
567 | 566
568 | 567
569 | 568
570 | 569
571 | 570
572 | 571
573 | 572
574 | 573
575 | 574
576 | 575
577 | 576
578 | 577
579 | 578
580 | 579
581 | 580
582 | 581
583 | 582
584 | 583
585 | 584
586 | 585
587 | 586
588 | 587
589 | 588
590 | 589
591 | 590
592 | 591
593 | 592
594 | 593
595 | 594
596 | 595
597 | 596
598 | 597
599 | 598
600 | 599
601 | 600
602 | 601
603 | 602
604 | 603
605 | 604
606 | 605
607 | 606
608 | 607
609 | 608
610 | 609
611 | 610
612 | 611
613 | 612
614 | 613
615 | 614
616 | 615
617 | 616
618 | 617
619 | 618
620 | 619
621 | 620
622 | 621
623 | 622
624 | 623
625 | 624
626 | 625
627 | 626
628 | 627
629 | 628
630 | 629
631 | 630
632 | 631
633 | 632
634 | 633
635 | 634
636 | 635
637 | 636
638 | 637
639 | 638
640 | 639
641 | 640
642 | 641
643 | 642
644 | 643
645 | 644
646 | 645
647 | 646
648 | 647
649 | 648
650 | 649
651 | 650
652 | 651
653 | 652
654 | 653
655 | 654
656 | 655
657 | 656
658 | 657
659 | 658
660 | 659
661 | 660
662 | 661
663 | 662
664 | 663
665 | 664
666 | 665
667 | 666
668 | 667
669 | 668
670 | 669
671 | 670
672 | 671
673 | 672
674 | 673
675 | 674
676 | 675
677 | 676
678 | 677
679 | 678
680 | 679
681 | 680
682 | 681
683 | 682
684 | 683
685 | 684
686 | 685
687 | 686
688 | 687
689 | 688
690 | 689
691 | 690
692 | 691
693 | 692
694 | 693
695 | 694
696 | 695
697 | 696
698 | 697
699 | 698
700 | 699
701 | 700
702 | 701
703 | 702
704 | 703
705 | 704
706 | 705
707 | 706
708 | 707
709 | 708
710 | 709
711 | 710
712 | 711
713 | 712
714 | 713
715 | 714
716 | 715
717 | 716
718 | 717
719 | 718
720 | 719
721 | 720
722 | 721
723 | 722
724 | 723
725 | 724
726 | 725
727 | 726
728 | 727
729 | 728
730 | 729
731 | 730
732 | 731
733 | 732
734 | 733
735 | 734
736 | 735
737 | 736
738 | 737
739 | 738
740 | 739
741 | 740
742 | 741
743 | 742
744 | 743
745 | 744
746 | 745
747 | 746
748 | 747
749 | 748
750 | 749
751 | 750
752 | 751
753 | 752
754 | 753
755 | 754
756 | 755
757 | 756
758 | 757
759 | 758
760 | 759
761 | 760
762 | 761
763 | 762
764 | 763
765 | 764
766 | 765
767 | 766
768 | 767
769 | 768
770 | 769
771 | 770
772 | 771
773 | 772
774 | 773
775 | 774
776 | 775
777 | 776
778 | 777
779 | 778
780 | 779
781 | 780
782 | 781
783 | 782
784 | 783
785 | 784
786 | 785
787 | 786
788 | 787
789 | 788
790 | 789
791 | 790
792 | 791
793 | 792
794 | 793
795 | 794
796 | 795
797 | 796
798 | 797
799 | 798
800 | 799
801 | 800
802 | 801
803 | 802
804 | 803
805 | 804
806 | 805
807 | 806
808 | 807
809 | 808
810 | 809
811 | 810
812 | 811
813 | 812
814 | 813
815 | 814
816 | 815
817 | 816
818 | 817
819 | 818
820 | 819
821 | 820
822 | 821
823 | 822
824 | 823
825 | 824
826 | 825
827 | 826
828 | 827
829 | 828
830 | 829
831 | 830
832 | 831
833 | 832
834 | 833
835 | 834
836 | 835
837 | 836
838 | 837
839 | 838
840 | 839
841 | 840
842 | 841
843 | 842
844 | 843
845 | 844
846 | 845
847 | 846
848 | 847
849 | 848
850 | 849
851 | 850
852 | 851
853 | 852
854 | 853
855 | 854
856 | 855
857 | 856
858 | 857
859 | 858
860 | 859
861 | 860
862 | 861
863 | 862
864 | 863
865 | 864
866 | 865
867 | 866
868 | 867
869 | 868
870 | 869
871 | 870
872 | 871
873 | 872
874 | 873
875 | 874
876 | 875
877 | 876
878 | 877
879 | 878
880 | 879
881 | 880
882 | 881
883 | 882
884 | 883
885 | 884
886 | 885
887 | 886
888 | 887
889 | 888
890 | 889
891 | 890
892 | 891
893 | 892
894 | 893
895 | 894
896 | 895
897 | 896
898 | 897
899 | 898
900 | 899
901 | 900
902 | 901
903 | 902
904 | 903
905 | 904
906 | 905
907 | 906
908 | 907
909 | 908
910 | 909
911 | 910
912 | 911
913 | 912
914 | 913
915 | 914
916 | 915
917 | 916
918 | 917
919 | 918
920 | 919
921 | 920
922 | 921
923 | 922
924 | 923
925 | 924
926 | 925
927 | 926
928 | 927
929 | 928
930 | 929
931 | 930
932 | 931
933 | 932
934 | 933
935 | 934
936 | 935
937 | 936
938 | 937
939 | 938
940 | 939
941 | 940
942 | 941
943 | 942
944 | 943
945 | 944
946 | 945
947 | 946
948 | 947
949 | 948
950 | 949
951 | 950
952 | 951
953 | 952
954 | 953
955 | 954
956 | 955
957 | 956
958 | 957
959 | 958
960 | 959
961 | 960
962 | 961
963 | 962
964 | 963
965 | 964
966 | 965
967 | 966
968 | 967
969 | 968
970 | 969
971 | 970
972 | 971
973 | 972
974 | 973
975 | 974
976 | 975
977 | 976
978 | 977
979 | 978
980 | 979
981 | 980
982 | 981
983 | 982
984 | 983
985 | 984
986 | 985
987 | 986
988 | 987
989 | 988
990 | 989
991 | 990
992 | 991
993 | 992
994 | 993
995 | 994
996 | 995
997 | 996
998 | 997
999 | 998
1000 | 999
1001 | 1000
1002 | 1001
1003 | 1002
1004 | 1003
1005 | 1004
1006 | 1005
1007 | 1006
1008 | 1007
1009 | 1008
1010 | 1009
1011 | 1010
1012 | 1011
1013 | 1012
1014 | 1013
1015 | 1014
1016 | 1015
1017 | 1016
1018 | 1017
1019 | 1018
1020 | 1019
1021 | 1020
1022 | 1021
1023 | 1022
1024 | 1023
1025 | 1024
1026 | 1025
1027 | 1026
1028 | 1027
1029 | 1028
1030 | 1029
1031 | 1030
1032 | 1031
1033 | 1032
1034 | 1033
1035 | 1034
1036 | 1035
1037 | 1036
1038 | 1037
1039 | 1038
1040 | 1039
1041 | 1040
1042 | 1041
1043 | 1042
1044 | 1043
1045 | 1044
1046 | 1045
1047 | 1046
1048 | 1047
1049 | 1048
1050 | 1049
1051 | 1050
1052 | 1051
1053 | 1052
1054 | 1053
1055 | 1054
1056 | 1055
1057 | 1056
1058 | 1057
1059 | 1058
1060 | 1059
1061 | 1060
1062 | 1061
1063 | 1062
1064 | 1063
1065 | 1064
1066 | 1065
1067 | 1066
1068 | 1067
1069 | 1068
1070 | 1069
1071 | 1070
1072 | 1071
1073 | 1072
1074 | 1073
1075 | 1074
1076 | 1075
1077 | 1076
1078 | 1077
1079 | 1078
1080 | 1079
1081 | 1080
1082 | 1081
1083 | 1082
1084 | 1083
1085 | 1084
1086 | 1085
1087 | 1086
1088 | 1087
1089 | 1088
1090 | 1089
1091 | 1090
1092 | 1091
1093 | 1092
1094 | 1093
1095 | 1094
1096 | 1095
1097 | 1096
1098 | 1097
1099 | 1098
1100 | 1099
1101 | 1100
1102 | 1101
1103 | 1102
1104 | 1103
1105 | 1104
1106 | 1105
1107 | 1106
1108 | 1107
1109 | 1108
1110 | 1109
1111 | 1110
1112 | 1111
1113 | 1112
1114 | 1113
1115 | 1114
1116 | 1115
1117 | 1116
1118 | 1117
1119 | 1118
1120 | 1119
1121 | 1120
1122 | 1121
1123 | 1122
1124 | 1123
1125 | 1124
1126 | 1125
1127 | 1126
1128 | 1127
1129 | 1128
1130 | 1129
1131 | 1130
1132 | 1131
1133 | 1132
1134 | 1133
1135 | 1134
1136 | 1135
1137 | 1136
1138 | 1137
1139 | 1138
1140 | 1139
1141 | 1140
1142 | 1141
1143 | 1142
1144 | 1143
1145 | 1144
1146 | 1145
1147 | 1146
1148 | 1147
1149 | 1148
1150 | 1149
1151 | 1150
1152 | 1151
1153 | 1152
1154 | 1153
1155 | 1154
1156 | 1155
1157 | 1156
1158 | 1157
1159 | 1158
1160 | 1159
1161 | 1160
1162 | 1161
1163 | 1162
1164 | 1163
1165 | 1164
1166 | 1165
1167 | 1166
1168 | 1167
1169 | 1168
1170 | 1169
1171 | 1170
1172 | 1171
1173 | 1172
1174 | 1173
1175 | 1174
1176 | 1175
1177 | 1176
1178 | 1177
1179 | 1178
1180 | 1179
1181 | 1180
1182 | 1181
1183 | 1182
1184 | 1183
1185 | 1184
1186 | 1185
1187 | 1186
1188 | 1187
1189 | 1188
1190 | 1189
1191 | 1190
1192 | 1191
1193 | 1192
1194 | 1193
1195 | 1194
1196 | 1195
1197 | 1196
1198 | 1197
1199 | 1198
1200 | 1199
1201 | 1200
1202 | 1201
1203 | 1202
1204 | 1203
1205 | 1204
1206 | 1205
1207 | 1206
1208 | 1207
1209 | 1208
1210 | 1209
1211 | 1210
1212 | 1211
1213 | 1212
1214 | 1213
1215 | 1214
1216 | 1215
1217 | 1216
1218 | 1217
1219 | 1218
1220 | 1219
1221 | 1220
1222 | 1221
1223 | 1222
1224 | 1223
1225 | 1224
1226 | 1225
1227 | 1226
1228 | 1227
1229 | 1228
1230 | 1229
1231 | 1230
1232 | 1231
1233 | 1232
1234 | 1233
1235 | 1234
1236 | 1235
1237 | 1236
1238 | 1237
1239 | 1238
1240 | 1239
1241 | 1240
1242 | 1241
1243 | 1242
1244 | 1243
1245 | 1244
1246 | 1245
1247 | 1246
1248 | 1247
1249 | 1248
1250 | 1249
1251 | 1250
1252 | 1251
1253 | 1252
1254 | 1253
1255 | 1254
1256 | 1255
1257 | 1256
1258 | 1257
1259 | 1258
1260 | 1259
1261 | 1260
1262 | 1261
1263 | 1262
1264 | 1263
1265 | 1264
1266 | 1265
1267 | 1266
1268 | 1267
1269 | 1268
1270 | 1269
1271 | 1270
1272 | 1271
1273 | 1272
1274 | 1273
1275 | 1274
1276 | 1275
1277 | 1276
1278 | 1277
1279 | 1278
1280 | 1279
1281 | 1280
1282 | 1281
1283 | 1282
1284 | 1283
1285 | 1284
1286 | 1285
1287 | 1286
1288 | 1287
1289 | 1288
1290 | 1289
1291 | 1290
1292 | 1291
1293 |
```



### 3.1.3 Analyse de la rapidité de l'algorithme

Dans cette sous-section tout comme dans les sous-sections du même nom dans les autres sections, nous allons en premier lieu présenter un tableau qui décrit en gros le nombre considéré et le temps de calcul mis par notre algorithme pour afficher le résultat. Nous allons ensuite expliquer et interpréter en détail le tableau. Enfin, nous fournirons encore d'autres tableaux qui donneront d'autres types de renseignements.

n	Mesure 1	Mesure 2	Moyenne
<b>11213</b>	0.85 s	0.34 s	0.595 s, <1s
<b>21701</b>	3.05 s	3.53 s	3.29 s
<b>44497</b>	16.34s	13.44 s	14.89 s
<b>110503</b>	2 min 23.31 s	2 min 25.88 s	2 min 24.595 s
<b>216091</b>			>15 min

Figure 8: 1er tableau expérimental sur le test de Lucas-Lehmer

Dans ce premier tableau, nous avons pris des nombres  $n$  qui donnent des nombres premiers de Mersenne. On voit bien que le temps d'exécution croît exponentiellement, car si on prend par exemple  $n = 110503$ , on voit que ceci est environ trois fois plus grand que  $n = 44497$ , mais le temps que l'ordinateur prend pour nous donner le résultat est environ 15 fois plus.

On a eu des difficultés à mesurer le temps pour  $n = 11213$ , car le temps d'exécution n'est même pas d'une seconde, ce qui engendre facilement des erreurs de mesure de l'ordre d'une seconde, le temps que nous réagissions.

On voulait aussi faire l'expérience pour  $n = 216091$ , mais comme après 15 minutes l'ordinateur n'avait toujours pas trouvé la solution on a arrêté l'expérience.

n	Mesure 1	Mesure 2	Moyenne1
<b>16457</b>	1.91 s	1.8 s	1.855 s
<b>33099</b>	6.83 s	7.08 s	6.955 s
<b>77500</b>	51.01 s	51.33 s	51.17 s
<b>163297</b>	9 min 34.59 s	9 min 31.72 s	9 min 33.155 s

Figure 9: 2ème tableau expérimental sur le test de Lucas-Lehmer

Pour ce deuxième tableau nous avons pris des  $n$  qui se trouvent entre les  $n$  du premier tableau, car pour  $n = 11213$ , le temps d'exécution était trop court et pour  $n = 216091$  le temps d'exécution était trop long.

On voit bien que si on prend un  $n$  du deuxième tableau qui se trouve entre deux  $n$  du premier tableau, le temps d'exécution va rester entre le temps pris par l'ordinateur pour les  $n$  du premier tableau. Mais ceci est facile à expliquer car si on regarde notre programme sur **SAGE**, on voit que la boucle va être toujours exécutée si le nombre  $n$  donne un nombre premier de Mersenne ou pas. La seule chose qui change dans l'exécution est la condition

*if*. Mais comme le *if* ne contient pas de boucle, il prend un temps constant pour être exécuté.

On a arrêté à  $n = 163297$  car l'ordinateur a pris *9min33.155 s* pour trouver la solution et en plus le nombre de Mersenne que nous avons trouvé n'était pas entièrement affiché. **SAGE** a laissé le message suivant :

WARNING: Output: 49216 truncated by MAX\_STDOUT\_SIZE to 40000

## 3.2 Le test de Miller-Rabin

Le plus grand nombre premier que nous avons trouvé à l'aide du test de Miller-Rabin dans un laps de temps correct est  $2^{44497} - 1$ .

### 3.2.1 Code source

```

1 -
2 1 #Test de Miller-Rabin:
3 2
4 3 from random import randint
5 4 n=100000016531 #n doit être un entier impair plus grand que 3.
6 5 k=1 #k doit être un entier plus grand ou égal à 1.
7 6
8 7 def temoin_de_miller(n,a): #On définit une fonction qui vérifie si on a un témoin de Miller ou pas
9 8 s=(n-1).valuation(2)
10 9 d=(n-1) // 2*s
11 10 #pour les deux premières lignes dans cette fonction, on calcule s et d de l'équation n-1=(2^s)*d
12 11 x= power_mod(a,d,n) #on prend x=a^d (mod n)
13 12 if x==1 or x==n-1: #Si x vaut 1 ou (n-1),alors on n'a pas un témoin de Miller
14 13 return False
15 14 while s>1: #Aussi longtemps que s>1, on effectue le calcul ci-après
16 15 x = power_mod(x,2,n)
17 16 if x==n-1: #Si x vaut (n-1),alors on n'a pas de témoin de Miller
18 17 return False
19 18 s = s-1
20 19 return True #Sinon, on a un témoin de Miller
21 20
22 21 def miller_rabin(n,k):
23 22 for i in [1..k]: #on veut répéter k fois
24 23 a=randint(2,n-2) #a est un entier aléatoire entre 2 et n-2.
25 24 if temoin_de_miller(n,a)==True: #Si a est un témoin de Miller pour n, alors n est composé
26 25 return n,'est composé'
27 26 else:
28 27 for p in primes_first_n(k): #Sinon,on va chercher tous les nombres premiers plus petit que k

```

Figure 10: Code source de l'algorithme de Miller-Rabin (1ère partie)

```

27 27 for p in primes_first_n(k): #Sinon,on va chercher tous les nombres premiers plus petit que k
28 28 if Mod(n,p)==0:
29 29 if n==p:
30 30 return n,'est premier'
31 31 return n,'est probablement premier si k est suffisamment grand'
32 32
33 33 miller_rabin(n,k)
34 34

```

Figure 11: Code source de l'algorithme de Miller-Rabin (2ème partie)

### 3.2.2 Exemples

Cas où le nombre préalablement choisi est probablement premier :

```

1 *
2 1 #Test de Miller-Rabin:
3 2
4 3 from random import randint
5 4 n= 100000016531 #n doit être un entier impair plus grand que 3.
6 5 k=50000 #k doit être un entier plus grand ou égal à 1.
7 6
8 * 7 def temoin_de_miller(n,a): #On définit une fonction qui vérifie si on a un témoin de Miller ou pas
9 8 s=(n-1).valuation(2)
10 9 d=(n-1) // 2^s
11 10 #pour les deux premières lignes dans cette fonction, on calcule s et d de l'équation n-1=(2^s)*d
12 11 x= power_mod(a,d,n) #on prend x=a^d (mod n)
13 * 12 if x==1 or x == n-1: #Si x vaut 1 ou (n-1),alors on n'a pas un témoin de Miller
14 13 return False
15 * 14 while s>1: #Aussi longtemps que s>1, on effectue le calcul ci-après
16 15 x = power_mod(x,2,n)
17 * 16 if x==n-1: #Si x vaut (n-1),alors on n'a pas de témoin de Miller
18 17 return False
19 18 s = s-1
20 19 return True #Sinon, on a un témoin de Miller
21 20
22 * 21 def miller_rabin(n,k):
23 * 22 for i in [1..k]: #on veut répéter k fois
24 23 a=randint(2,n-2) #a est un entier aléatoire entre 2 et n-2.
25 * 24 if temoin_de_miller(n,a)==True: #Si a est un témoin de Miller pour n, alors n est composé
26 25 return n,'est composé'
27 * 26 else:
28 * 27 for p in primes_first_n(k): #Sinon,on va chercher tous les nombres premiers plus petit que k

```

Figure 12: 1ère partie du programme

```

28 * 27 for p in primes_first_n(k): #Sinon,on va chercher tous les nombres premiers plus petit que k
29 * 28 if Mod(n,p)==0:
30 * 29 return n,'est premier'
31 30 return n,'est probablement premier si k est suffisamment grand'
32 31
33 32 miller_rabin(n,k)
34 33
35 *
36 * (100000016531, 'est probablement premier si k est suffisamment grand')

```

Figure 13: 2ème partie du programme

Cas où le nombre préalablement choisi est composé :

```

1 *
2 1 #Test de Miller-Rabin:
3 2
4 3 from random import randint
5 4 n= 57933 #n doit être un entier impair plus grand que 3.
6 5 k=100 #k doit être un entier plus grand ou égal à 1.
7 6
8 * 7 def temoin_de_miller(n,a): #On définit une fonction qui vérifie si on a un témoin de Miller ou pas
9 8 s=(n-1).valuation(2)
10 9 d=(n-1) // 2^s
11 10 #pour les deux premières lignes dans cette fonction, on calcule s et d de l'équation n-1=(2^s)*d
12 11 x= power_mod(a,d,n) #on prend x=a^d (mod n)
13 * 12 if x==1 or x == n-1: #Si x vaut 1 ou (n-1),alors on n'a pas un témoin de Miller
14 13 return False
15 * 14 while s>1: #Aussi longtemps que s>1, on effectue le calcul ci-après
16 15 x = power_mod(x,2,n)
17 * 16 if x==n-1: #Si x vaut (n-1),alors on n'a pas de témoin de Miller
18 17 return False
19 18 s = s-1
20 19 return True #Sinon, on a un témoin de Miller
21 20
22 * 21 def miller_rabin(n,k):
23 * 22 for i in [1..k]: #on veut répéter k fois
24 23 a=randint(2,n-2) #a est un entier aléatoire entre 2 et n-2.
25 * 24 if temoin_de_miller(n,a)==True: #Si a est un témoin de Miller pour n, alors n est composé
26 25 return n,'est composé'
27 * 26 else:
28 * 27 for p in primes_first_n(k): #Sinon,on va chercher tous les nombres premiers plus petit que k

```

Figure 14: 1ère partie du programme

```

28 * 27         for p in primes_first_n(k): #Sinon, on va chercher tous les nombres premiers plus petit que k
29 * 28         if Mod(n,p)==0:
30 * 29             if n==p:
31 * 30                 return n, 'est premier'
32 * 31             return n, 'est probablement premier si k est suffisamment grand'
33 * 32
34 * 33 miller_rabin(n,k)
35 *
36 *

```

Figure 15: 2ème partie du programme

Nous allons montrer dans l'exemple qui suit l'importance et le rôle de la variable  $k$  dans notre programme.

```

1 *
2 * #Test de Miller-Rabin:
3 *
4 * from random import randint
5 * n= 1117 #n doit être un entier impair plus grand que 3.
6 * k=186 #k doit être un entier plus grand ou égal à 1.
7 *
8 * def temoin_de_miller(n,a): #On définit une fonction qui vérifie si on a un témoin de Miller ou pas
9 *     s=(n-1).valuation(2)
10 *     d=(n-1)//2^s
11 *     #Pour les deux premières lignes dans cette fonction, on calcule s et d de l'équation n-1=(2^s)*d
12 *     #Si x vaut 1 ou (n-1), alors on n'a pas un témoin de Miller
13 *     if x==1 or x==n-1:
14 *         return False
15 *     #Aussi longtemps que s>1, on effectue le calcul ci-après
16 *     while s>1:
17 *         x = power_mod(x,2,n)
18 *         #Si x vaut (n-1), alors on n'a pas de témoin de Miller
19 *         return False
20 *     s = s-1
21 *     return True #Sinon, on a un témoin de Miller
22 *
23 * def miller_rabin(n,k):
24 *     for i in [1..k]: #On veut répéter k fois
25 *         a=randint(2,n-2) #a est un entier aléatoire entre 2 et n-2.
26 *         if temoin_de_miller(n,a)==True: #Si a est un témoin de Miller pour n, alors n est composé
27 *             return n, 'est composé'
28 *     else:
29 *         #n est probablement premier si k est suffisamment grand

```

Figure 16: 1117 est probablement premier si  $k \leq 186$  (1ère partie)

```

26 * 25         return n, 'est composé'
27 * 26     else:
28 * 27         for p in primes_first_n(k): #Sinon, on va chercher tous les nombres premiers plus petit que k
29 * 28         if Mod(n,p)!=0:
30 * 29             if n==p:
31 * 30                 return n, 'est premier'
32 * 31             return n, 'est probablement premier si k est suffisamment grand'
33 * 32
34 * 33 miller_rabin(n,k)
35 *
36 *
37 * 1
38 * 4

```

Figure 17: 1117 est probablement premier si  $k \leq 186$  (2ème partie)

```

1 *
2 * #Test de Miller-Rabin:
3 *
4 * from random import randint
5 * n= 1117 #n doit être un entier impair plus grand que 3.
6 * k=187 #k doit être un entier plus grand ou égal à 1.
7 *
8 * def temoin_de_miller(n,a): #On définit une fonction qui vérifie si on a un témoin de Miller ou pas
9 *     s=(n-1).valuation(2)
10 *     d=(n-1)//2^s
11 *     #Pour les deux premières lignes dans cette fonction, on calcule s et d de l'équation n-1=(2^s)*d
12 *     #Si x vaut 1 ou (n-1), alors on n'a pas un témoin de Miller
13 *     if x==1 or x==n-1:
14 *         return False
15 *     #Aussi longtemps que s>1, on effectue le calcul ci-après
16 *     while s>1:
17 *         x = power_mod(x,2,n)
18 *         #Si x vaut (n-1), alors on n'a pas de témoin de Miller
19 *         return False
20 *     s = s-1
21 *     return True #Sinon, on a un témoin de Miller
22 *
23 * def miller_rabin(n,k):
24 *     for i in [1..k]: #On veut répéter k fois
25 *         a=randint(2,n-2) #a est un entier aléatoire entre 2 et n-2.
26 *         if temoin_de_miller(n,a)==True: #Si a est un témoin de Miller pour n, alors n est composé
27 *             return n, 'est composé'
28 *     else:
29 *         #n est probablement premier si k est suffisamment grand

```

Figure 18: 1117 est premier si  $k \geq 187$  (1ère partie)

```

28 * 27      for p in primes_first_n(k): #Sinon, on va chercher tous les nombres premiers plus petit que k
29 * 28      if mod(n,p) == 0:
30 * 29          if n == p:
31 * 30              return n, 'est premier'
32 * 31          return n, 'est probablement premier si k est suffisamment grand'
33 * 32
34 * 33      miller_rabin(n,k)
35 * 34      (1117, 'est premier')
36 * 35
37 * 36

```

Figure 19: 1117 est premier si  $k \geq 187$  (2ème partie)

### 3.2.3 Analyse de la rapidité de l'algorithme

Comme nous l'avons dit dans l'introduction de la sous-section 3.1.3, nous allons procéder ici de la même manière que dans la sous-section mentionnée.

n	k	Mesure 1	Mesure 2	Moyenne
$2^{21701}$	100	4.87 s	4.91 s	4.89 s
$2^{21701}$	$10^6$	5.4 s	5.28 s	5.34 s
$2^{44497}$	100	26.58 s	26.63 s	26.605 s
$2^{44497}$	$10^6$	27.45 s	27.98 s	27.715 s

Figure 20: 1er tableau expérimental sur le test de Miller-Rabin

Dans ce premier tableau on voulait tester des nombres composés. On a essayé des nombres dans les  $10^{50}$ , mais le temps d'exécution était toujours inférieur à une seconde. Donc on a décidé de prendre des nombres de la forme des nombres de Mersenne augmentés de 1 (c'est-à-dire de la forme  $n = 2^x - 1$  où  $x$  est un entier naturel).

On a donc pris les nombres  $n = 221701$  et  $244497$  et pour  $k$  on a aussi pris deux  $k$  différents, à savoir  $k = 100$  et  $k = 106$ .

On a conclu que si on prend un plus grand nombre, l'ordinateur prend plus de temps à le calculer (ce qui est évident). Mais on a aussi vu que si on augmente la valeur de  $k$ , le temps d'exécution augmente aussi.

n	k	Mesure 1	Mesure 2	Moyenne
$2^{11213}-1$	1000	1.36 s	1.24 s	1.3 s
$2^{21701}-1$	1000	5.38 s	5.28 s	5.33 s
$2^{44497}-1$	1000	26.85 s	26.3 s	26.575 s
$2^{110503}-1$	1000	4 min 9.93 s	4 min 10.06 s	4 min 9.995 s

Figure 21: 2ème tableau expérimental sur le test de Miller-Rabin

Dans ce deuxième tableau on a pris les nombres de Mersenne que l'on a testé dans le test de Lucas-Lehmer avec  $k = 1000$ . Donc la réponse du programme était toujours que notre  $n$  est probablement premier si  $k$  est suffisamment grand.

Notre but dans ce tableau était de comparer le test de Lucas-Lehmer avec le test de Miller-Rabin. On a pu constater que le test de Lucas-Lehmer est beaucoup plus rapide et nous donne une information exacte tandis que le test de Miller-Rabin nous dit seulement que c'est probablement premier. On pourrait aussi augmenter la valeur de  $k$  pour obtenir une réponse exacte,

mais si on faisait cela, le temps d'exécution augmenterait aussi. En guise de conclusion, nous avons que le test de Lucas-Lehmer est beaucoup plus efficace pour trouver les nombres premiers de Mersenne (et les nombres premiers en général) que le test de Miller-Rabin.

n	k	Mesure 1	Mesure 2	Moyenne
178439	100000	5.33 s	5.03 s	5.18 s
558179	100000	13.52 s	13.81 s	13.665 s
912727	100000	20.78 s	20.59 s	20.685 s
1297333	100000	28.54 s	28.56 s	28.55 s

Figure 22: 3ème tableau expérimental sur le test de Miller-Rabin

Dans ce dernier tableau, notre but était de mesurer le temps d'exécution nécessaire pour avoir un résultat précis. Donc pour chaque nombre on a pris le même  $k$  pour ne pas avoir d'erreur de mesure à cause du  $k$ . Dans chaque exécution on a eu la réponse : *n est premier*

La distance entre chaque nombre  $n$  est toujours d'environ 400000. Si on regarde le temps d'exécution pour chaque  $n$ , on voit que le temps augmente toujours de plus ou moins 8 s. Donc le temps semble être proportionnel au nombre  $n$ .

### 3.3 Le test AKS

Le plus grand nombre premier que nous avons obtenu pendant un lap de temps correct est 178439 .

#### 3.3.1 Code source

```

1 -
2 1 n=13
3 2
4 3 def AKS(n):
5 4 #Cas: mod k.
6 5 for p in [2..(ceil(log(n,2)))]:
7 6 if is_prime(p)==true:
8 7   l=ln(valuation(p))
9 8   if (n == p^k) and (k>1):
10 9     return "composé"
11 10
12 11 #on va chercher le plus petit r tel que n^l = 1 mod r avec 1 <= l<borne.
13 12 borne=(1/2*floor(log(n,2)^5)) #on prend le maximum entre 3 et floor(log(n,2)^5) car on a r = 2 et pour la boucle on a r plus petit ou égal au maximum.
14 13 r = 2
15 14 while r <= n:
16 15   compt = 0
17 16   borne = floor(log(n,2)^2)
18 17   for i in [1..borne]:
19 18     if ((n^i)%r==1):
20 19       compt = compt+1
21 20   if(compt==l):#après la première itération si le compteur est encore nul alors on arrête la boucle.
22 21     break
23 22   r*=4
24 23
25 24 for a in [1..r]:
26 25   if lgcd(a,n) and gcd(a,n)<n:
27 26     return "composé"

```

Figure 23: Code source de l'algorithme AKS (1ère partie)





On voit bien que pour des nombres composés, le programme AKS est rapide, même plus vite que celui de Miller-Rabin.

On a aussi essayé avec des nombres  $n$  de la forme  $n = p^k$ , où  $p$  est un entier plus grand que 0 et  $k$  un entier plus grand que 1, mais les résultats étaient encore une fois presque immédiats.

## 4 Conclusion

Après avoir longuement étudié avec précision les 3 tests de primalité, nous pouvons dès à présent les comparer entre eux en énumérant leurs points forts et points faibles.

Tout d'abord, en ce qui concerne la rapidité de l'algorithme, le test de primalité de Lucas-Lehmer est de loin le plus rapide des 3. Vient ensuite le test de Miller-Rabin et enfin, loin derrière, le test AKS.

Vient ensuite l'exactitude des résultats du programme ; les algorithmes de Lucas-Lehmer et AKS sont tous les 2 aussi efficaces dans ce domaine étant donné qu'ils disent avec certitude si un nombre donné est premier ou pas. Par contre, le test de Miller-Rabin ne fournit en général qu'une probabilité à un nombre donné d'être premier tant que l'on n'a pas choisi un  $k$  suffisamment grand (pour la signification du  $k$ , référez vous au paragraphe 3.1.1), ce qui place ce test en dernière position parmi les 3.

Pour terminer les comparaisons, en ce qui concerne la quantité de nombres premiers trouvables à l'aide de ces 3 algorithmes, le test de Lucas-Lehmer est le plus désavantageux. En effet, là où les tests de primalité de Miller-Rabin et AKS fournissent tous les nombres premiers possibles, le test de Lucas-Lehmer ne fournit que les nombres premiers de Mersenne qui sont les nombres d'une suite particulière.

À partir de ces comparaisons, l'on constate que suivants les critères que l'on prend en compte, certains tests de primalités sont plus à privilégier que d'autres. Ainsi par exemple, si une personne veut réaliser un test de primalité qui trouve très rapidement avec exactitude si un nombre donné, aussi grand soit-il, est premier, alors dans ce cas, le test de Lucas-Lehmer sera largement recommandé. C'est d'ailleurs avec ce test que des mathématiciens sont parvenus, en janvier 2016, à trouver le plus grand nombre premier connu à ce jour, à savoir  $2^{74207281} - 1$ .

Ce qu'il faut donc retenir de tout cela, c'est qu'il y a des tests de primalité pour tous les goûts et qu'il est de nos jours encore difficile de recommander un test de primalité en particulier malgré que certains tests comme celui de Lucas-Lehmer soient extrêmement efficaces. Tout dépendra simplement de l'objectif recherché de la personne par l'utilisation de ces tests de primalité.

## 5 Bibliographie

- Entretiens personnels avec Prof. Wiese du 8 et 22 mars, du 27 avril et du 18 mai 2017
- Paulo RIBENBOIM - The little book of bigger primes (Second edition of Springer edition)
- [https://fr.wikipedia.org/wiki/Test\\_de\\_primalit \\_de\\_Lucas-Lehmer\\_pour\\_les\\_nombres\\_de\\_Mersenne](https://fr.wikipedia.org/wiki/Test_de_primalit _de_Lucas-Lehmer_pour_les_nombres_de_Mersenne)
- [https://fr.wikipedia.org/wiki/ douard\\_Lucas](https://fr.wikipedia.org/wiki/ douard_Lucas)
- [https://fr.wikipedia.org/wiki/Derrick\\_Lehmer](https://fr.wikipedia.org/wiki/Derrick_Lehmer)
- [https://fr.wikipedia.org/wiki/Test\\_de\\_primalit \\_de\\_Lucas-Lehmer](https://fr.wikipedia.org/wiki/Test_de_primalit _de_Lucas-Lehmer)
- [https://fr.wikipedia.org/wiki/Nombre\\_de\\_Mersenne\\_premier](https://fr.wikipedia.org/wiki/Nombre_de_Mersenne_premier)
- Notes de cours de "Th orie des nombres et applications   la cryptographie" du Prof. Wiese
- [https://fr.wikipedia.org/wiki/Test\\_de\\_primalit \\_AKS](https://fr.wikipedia.org/wiki/Test_de_primalit _AKS)
- [https://fr.wikipedia.org/wiki/Gary\\_L.\\_Miller](https://fr.wikipedia.org/wiki/Gary_L._Miller)
- [https://fr.wikipedia.org/wiki/Michael\\_Rabin](https://fr.wikipedia.org/wiki/Michael_Rabin)
- [https://fr.wikipedia.org/wiki/Test\\_de\\_primalit \\_de\\_Miller-Rabin](https://fr.wikipedia.org/wiki/Test_de_primalit _de_Miller-Rabin)
- [https://fr.wikipedia.org/wiki/Congruence\\_sur\\_les\\_entiers](https://fr.wikipedia.org/wiki/Congruence_sur_les_entiers)
- [https://fr.wikipedia.org/wiki/Nombre\\_pseudo-premier](https://fr.wikipedia.org/wiki/Nombre_pseudo-premier)
- [https://fr.wikipedia.org/wiki/Nombre\\_compos ](https://fr.wikipedia.org/wiki/Nombre_compos )
- <http://www.trigofacile.com/maths/curiosite/primarite/aks/pdf/algorithmme-aks.pdf>
- <https://de.wikipedia.org/wiki/AKS-Primzahltest>
- [https://en.wikipedia.org/wiki/Miller-Rabin\\_primality\\_test](https://en.wikipedia.org/wiki/Miller-Rabin_primality_test)
- <https://www.math.univ-toulouse.fr/~hallouin/Documents/Primalite.pdf>
- [https://fr.wikipedia.org/wiki/Entier\\_sans\\_facteur\\_carr ](https://fr.wikipedia.org/wiki/Entier_sans_facteur_carr )
- [https://fr.wikipedia.org/wiki/Loi\\_de\\_r ciprocit \\_quadratique](https://fr.wikipedia.org/wiki/Loi_de_r ciprocit _quadratique)
- [https://fr.wikipedia.org/wiki/Th or me\\_des\\_restes\\_chinois](https://fr.wikipedia.org/wiki/Th or me_des_restes_chinois)

- <https://perso.univ-rennes1.fr/romain.basson/pdf/Crypto.pdf>
- [http://compoasso.free.fr/primelistweb/page/prime/liste\\_online.php](http://compoasso.free.fr/primelistweb/page/prime/liste_online.php)
- <http://www.bibmath.net/dossiers/index.php?action=affiche&quoi=dossier1/page4.html>
- <http://math.uni.lu/eml/projects/reports/Bachelor-Thesis-Notarnicola-Luca.pdf>
- [https://fr.wikipedia.org/wiki/Indicatrice\\_d'Euler](https://fr.wikipedia.org/wiki/Indicatrice_d'Euler)