Calculabilité, Complexité et Machine de Turing

Sommaire

1 La calculabilité **2**
- 1.1 Récursivité 2
- 1.2 Problèmes décidables - indécidables 2
- 1.3 Exemple des équations diophantiennes 3

2 La machine de Turing **3**

3 La complexité **5**
- 3.1 Définition 5
- 3.2 Classes P et NP 5
- 3.3 L'exemple de la primalité 6

4 Conclusion **7**

1 La calculabilité

Certains problèmes n'auront jamais de solutions.

Comment les classer ? Comment mathématiser cette notion ?

Une solution apportée dans les années 1930: La calculabilité.

1.1 Récursivité

On dit qu'une fonction est récursive lorsque elle est définie à partir d'elle-même.

La fonction factorielle est récursive: $\forall n \in \mathbb{N},\ n! = n \times (n-1)!$

Les fonctions récursives sont calculables $\leftrightarrow$ On peut trouver la factorielle de n'importe quel entier naturel

1.2 Problèmes décidables - indécidables

Trois notions équivalentes:

Fonction récursive $\Longleftrightarrow$ Fonction Calculable $\Longleftrightarrow$ Problème décidable

Comment passer de fonction calculable à problème décidable ?

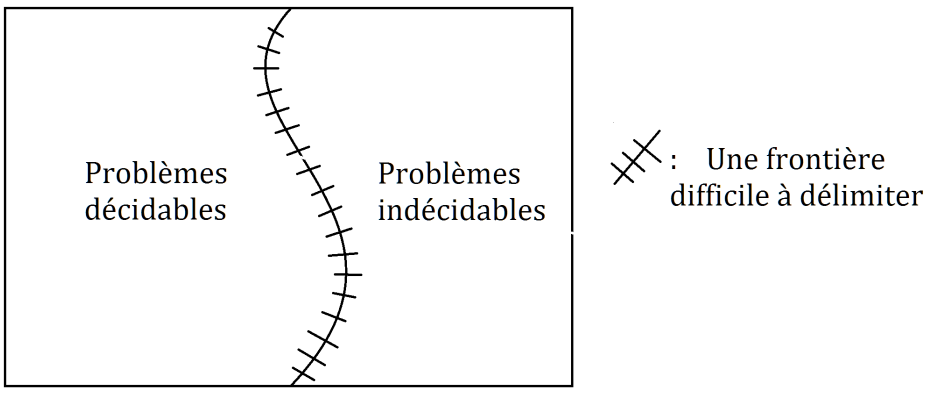
Soit P une propriété.

On lui associe sa fonction caracéristique χ_p

$\chi_p = 1$ si P est vrai.

$\chi_p = 0$ si P est faux.

Lorsque χ_p est une fonction calculable alors P est une propriété décidable.



1.3 Exemple des équations diophantiennes

L'équation $<< ax + by = cz >>$ est diophantienne $\Longleftrightarrow (a, b, c) \in \mathbb{Z}^3$

On définit les équations diophantiennes:

Du premier degré $<< ax + by = cz >> (a, b, c) \in \mathbb{Z}^3$
Du second degré $<< ax^2 + by^2 = cz >> (a, b, c) \in \mathbb{Z}^3$
Du n-ièmme degré $<< ax^n + by^n = cz >> (a, b, c) \in \mathbb{Z}^3$

Notons $D^n_{a,b,c}$ l'équation diophantienne à coefficients entiers a,b,c de degré n et P la propriété "'L'équation admet une solution."'

On lui associe sa fonction caractéristique χ_p :

$$\chi_p(D^n_{a,b,c}) = 1 \text{ si } P(D^n_{a,b,c}) \text{ est vraie.}$$
$$\chi_p(D^n_{a,b,c}) = 0 \text{ si } P(D^n_{a,b,c}) \text{ est fausse.}$$

Il n'existe pas d'algorithme capable de retourner l'existence ou la non-existence de solution pour une équation dophantienne de degré supérieur ou égal à 5

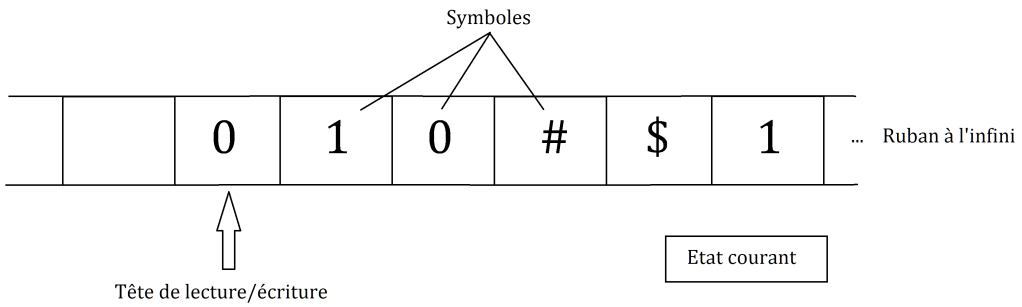
$\forall (a, b, c) \in \mathbb{Z}^3, \forall n \in [\![0, 3]\!]$, χ_p est calculable $\Longleftrightarrow$ P est décidable

$\forall (a, b, c) \in \mathbb{Z}^3, \forall n \in [\![5, +\infty]\!]$, χ_p n'est pas calculable $\Longleftrightarrow$ P est indécidable

2 La machine de Turing

Une machine théorique inventée par Alan Turing en 1936.
Capable de simuler n'importe quel algorithme, Python n'existait pas !

Fonctionnement:



Une machine de Turing peut-être définie par les données (Q, Σ, E, I_0) avec:

- Q l'ensemble des états possibles
- Σ l'ensemble des symboles utilisés
- E l'ensemble des transitions
- I_0 l'état initial de la machine

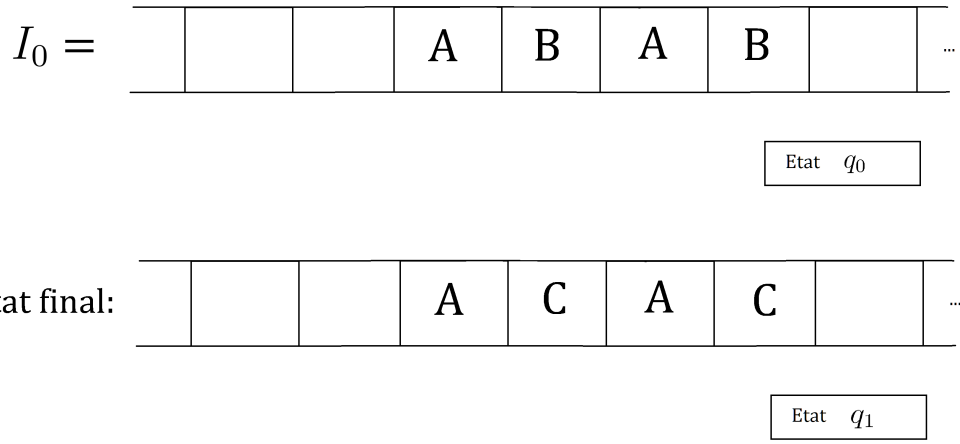
La thèse de Church-Turing: Pour tout algorithme, il existe une machine de turing capable de l'éxécuter.

Exemple:
On essaye de coder un algorithme qui transforme la chaîne de caractère ABAB en ACAC

$Q = (q_0, q_1, stop)$
$\Sigma = (A, B, C, \emptyset)$
E=

1. $(q_0, \emptyset, \emptyset, \longrightarrow, q_0)$
2. $(q_0, A, A, \longrightarrow, q_1)$
3. $(q_0, B, C, \longrightarrow, q_1)$
4. $(q_1, A, A, \longrightarrow, q_1)$
5. $(q_1, B, C, \longrightarrow, q_1)$
6. $(q_0, \emptyset, \emptyset, stop, q_1)$

Les instructions sont du type:
(état courant, symbole lu, symbole inscrit, déplacement, nouvel état courant)



3 La complexité

Parmi les problèmes décidables (dont il existe un algorithme de résolution) certains n'ont pourtant pas de solutions connues. Il est alors nécessaire d'estimer l'efficacité de l'algorithme, c'est le rôle de la complexité.

3.1 Définition

La complexité juge l'efficacité d'un algorithme pour traiter un problème donné. Elle peut être de plusieurs natures:

- Temps de calcul
- Place mémoire
- Nombre d'opérations
- Nombre de déplacement d'une tête de lecture/écriture sur le ruban d'une machine de Turing

Le but: Etablir une distinction entre les problèmes que l'on saura résoudre et ceux pour lesquels il faudra renoncer, faute de temps ou de place.

Pour un problème dont la taille des données est n on parlera de:

- Complexité exponentielle si elle est en $O(k^n)$ $k \in \mathbb{N}$
- Complexité polynomiale si elle est en $O(n^k)$ $k \in \mathbb{N}$

Seuls les algorithmes en complexité polynomiale sont utiles pour des grandes données d'entrées.

3.2 Classes P et NP

Deux classes de problèmes:

- P
- NP

La classe P est formée des problèmes décidables dont la solution est donnée par un algorithme de complexité polynomiale.

La classe NP est formée des problèmes qui peuvent être résolus par un algorithme de complexité polynomiale non déterministe.

3.3 L'exemple de la primalité

On note P le prédicat "'n est premier"'

Soit n$\in \mathbb{N}$
$P(n)$ vrai$\Longleftrightarrow$ n est premier
$P(n)$ faux$\Longleftrightarrow$ n n'est pas premier

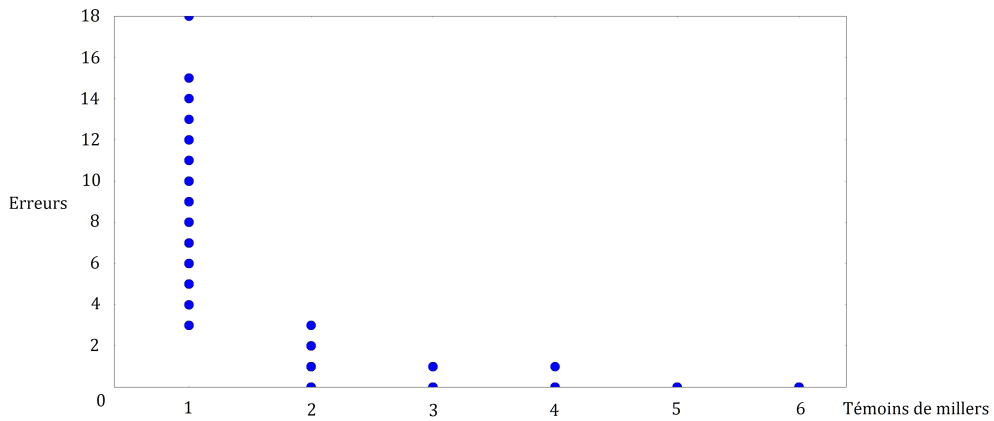
Plusieurs méthodes de résolution:

- Le crible d'Eratosthène
- Miller Rabin
- AKS

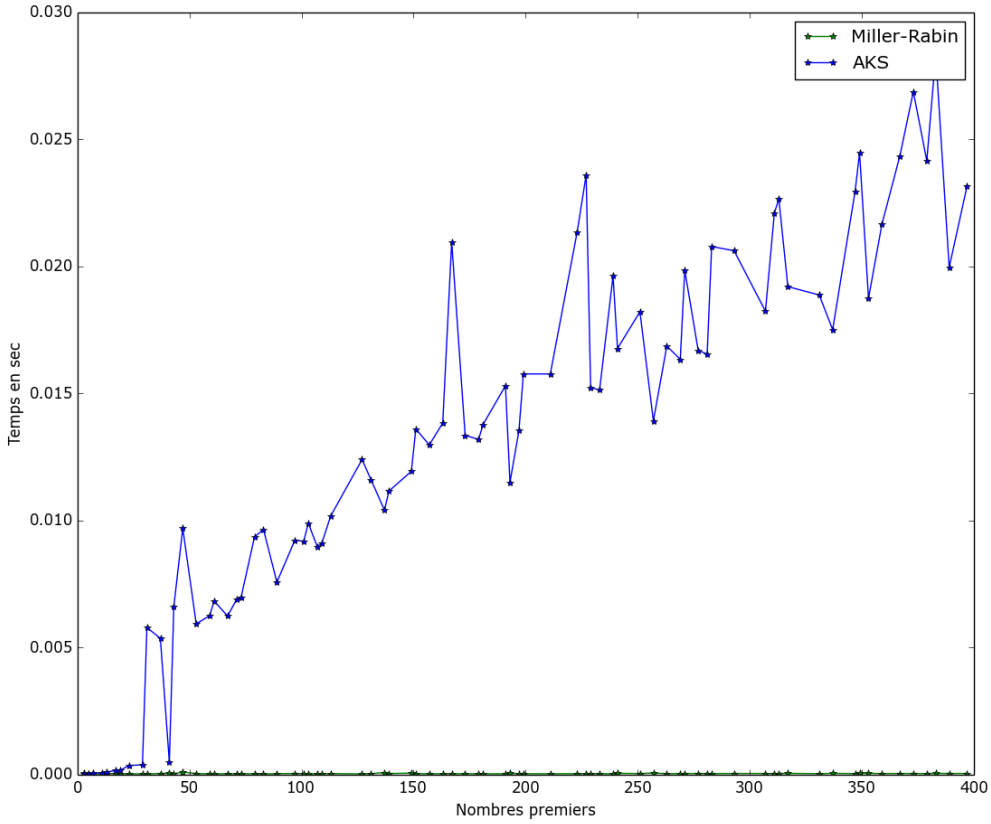
	Année	Déterministe	Polynomial
Crible d'Ératosthène	≈ -240	✔	✖
Miller-Rabin	1976	✖	✔
AKS	2002	✔	✔

L'algorithme de Miller-Rabin n'est pas déterministe.

Il commet donc des erreurs:
Certains nombre composés sont déclarés premiers.



Cependant, sa complexité en temps est très inférieure à celle de AKS



En cryptographie, de très grands nombres premiers sont utilisés et aucune erreur ne peut être tolérée.

Pour l'algorithme de MillerRabin (1 témoin de miller), la probabilité qu'un nombre soit premier avec:

- une itération: 99.1%
- cinq itérations: $(1 - (0.09)^5) \cdot 100 \approx 99.999999994\%$

4 Conclusion

- Différentes classes de problèmes ont été définies et ont permis de trier les algorithmes
- La machine théorique de Turing a servi de support pour définir la calculabilité et la complexité
- De nombreux problèmes restent inclassables