ALGORITHMES PROBABILISTES

Introduction

Il existe quatre grands types d'algorithmes probabilistes :

1. Les algorithmes numériques. Par exemple, on utilise l'aléatoire pour découper un intervalle de calcul.
2. Les méthodes de Monte-Carlo. Il s'agit de calculer une valeur incorrecte en un temps déterministe avec une probabilité d'erreur mesurable et pouvant être rendue arbitrairement petite.
3. Les méthodes de Las Vegas. C'est un algorithme qui renvoie une valeur toujours correcte en un temps aléatoire (avec une probabilité souvent très faible, le programme peut ne jamais se terminer).
4. Les algorithmes de randomisation. On introduit de l'aléatoire dans les données en entrée pour faire baisser la complexité moyenne.

1. Générateurs de nombres pseudo-aléatoires

Les nombres pseudo-aléatoires ainsi générés sont utiles dans plusieurs domaines :

1. les algorithmes probabilistes ;
2. la cryptographie ;
3. les tests (paramètres d'entrées choisis aléatoirement).

Un générateur de nombres pseudo-aléatoires est une fonction qui renvoie $x \in [0, 1[$.
En python, il y a le module *random*. Un *import* random donne accès à :
— *random.uniform(a,b)* qui renvoie un réel $a \leq x < b$ (0 et 1 par défaut pour a et b) ;
— *random.randinit(a,b)* qui renvoie un entier $a \leq x \leq b$.
— *random.choice(seq)* qui renvoie un élément parmi *seq* ;
— *random.random()* qui renvoie $x \in [0, 1[$.

Il faut cependant initialiser toutes ces méthodes avec *random.seed(x)*. Par défaut c'est l'heure du système.

Définition 1.1
Un générateur est une suite de nombres réels qui vont se comporter statistiquement comme une suite de nombres aléatoires sur $[0, 1[$.

Exemple. — Soit $(x_n)_{n\in\mathbf{N}}$ une suite d'entiers. On pose x_0 la graine. On définit la relation de récurrence par

$$x_{n+1} = Ax_n \mod M$$

avec $A, M \in \mathbf{N}$. La valeur retournée est x_n/M.

Proposition 1.2
La suite $(x_n)_{n\in\mathbf{N}}$ est périodique de période $P < M$. Si M est premier, $P = M - 1$ si, et seulement si, A vérifie $A^{M-1} = 1 \mod M$ et $A^k \neq 1 \mod M$ pour tout $k < M$.

Exemple. — Avec $M = 2^{31} - 1$ et $A = 397\ 204\ 094$.
Pour faciliter les calculs, on utilise généralement un M sous la forme $M = 2^\beta$.

Proposition 1.3
Si M est de cette forme, alors la période maximale est $2^{\beta-2}$ et est réalisée si $A = \pm 3 \mod 8$ et $x_0 = \pm 1 \mod 8$.

2. Monte-Carlo

Définition 2.1
Une variable aléatoire Y est une fonction qui associe à chaque résultat d'une expérience aléatoire un entier k. La loi de Y est une suite $(p_k)_{k\in K} \in [0, 1]$ traduisant $\mathbf{P}[Y = y_k] = p_k$ avec $\sum p_k = 1$.

Exemple. — Pour les dés, $\{y_k\} = \{1, 2, \ldots, 6\}$ et $p_k = 1/6$.

Définition 2.2
L'espérance de Y, notée, $\mathbf{E}[Y]$, est la somme :

$$\mathbf{E}[Y] = \sum_{k\geq 0} y_k p_k$$

si cette série converge.
La variance de Y est

$$\mathrm{Var}(Y) = \sum_{k\geq 0} p_k y_k^2 = \mathbf{E}[Y]^2.$$

Théorème 2.3 (Loi des grands nombres)
Soit (Y_i) une suite de variables aléatoires indépendantes et de même loi. Alors

$$m_n = \frac{1}{N}\sum_{i=1}^{N} Y_i \overset{N\to\infty}{\longrightarrow} \mathbf{E}[Y].$$

2.1. Exemple d'algorithme de Monte-Carlo

C'est un algorithme qui termine mais éventuellement en donnant une réponse inexacte (c'est une proportion de cas à minimiser).

Problème. — Il s'agit pour un entier n donné, de savoir si n est premier et s'il ne l'est pas, une décomposition de n. Nous allons étudier le teste de primalité de Miller-Rabin.

Théorème 2.4 (Petit théorème de Fermat)
Soit p un entier non nul. Si p est premier alors pour $a \in \{1, \dots, p-1\}$

$$a^{p-1} = 1 \mod p.$$

Démonstration
Soit a non nul, $\mathbf{Z}/p\mathbf{Z}$ est un corps à p éléments. Si $a^{p-1} \neq 1$ alors on a aussi $a^{p-1} \neq 0$ et on a pour tout $n \leq p-1$: $a^{p-1} \neq 1$. Ainsi, il y a $p-1$ nombres entre 2 et $p-1$, absurde.

Proposition 2.5
Soit $n \in \mathbf{N}^*$. Supposons que $n = 2^k m + 1$ où m est impair. Si l'une des conditions suivantes est vraie :
— $a^{n-1} \neq 1 \mod n$;
— $a^{n-1} = 1 \mod n$ et $a^m \neq 1 \mod n$ et pour tout $i \leq k$,

$$a^{2^i m} \neq -1 \mod n.$$

Alors n est composé.

Démonstration
Soit n telle que la seconde condition soit vérifiée. Soit $i \leq k$ le plus grand entier tel que

$$a^{2^i m} \neq 1 \mod n.$$

On pose $b = a^{2^i m}$, $b \neq \pm 1 \mod n$. Ainsi, n ne divise ni $b-1$ ni $b+1$. Or

$$b^2 = a^{2^{i+1} m} = 1 \mod n$$

et donc

$$b^2 - 1 = (b-1)(b+1) = 0 \mod n.$$

Or n ne peut diviser $b-1$ ou $b+1$ et donc n ne peut être premier. Donc n est composé.

Remarque. — Les nombres vérifiant la proposition précédente sont appelés « nombre de Carmichael ».

```
1  function Temoin(a,b,d,s)
2          x = a**d % n
3          if x = 1 then return false
4          for r from 1 to s-1 do
5                  x = a**(2**r) % n
6                  if x = n-1 then
7                          return false
8          return true
9
10 function MillerRabin(n,k=10)
11         n - 1 = (2**s)*d
12         for i from 1 to k do
13                 a = rand(2,n-2)
14                 if Temoin(a,n,d,s) then
15                         return false
16         return true
```

Considérons le sous-groupe :

$$B_i = \left\{ a \,\middle|\, a \leq n-1, \text{pgcd}(a,n) = 1 \text{ et } a^{2^i m} = \pm 1 \mod n \right\}$$

où

$$i = \max \left\{ i \leq k \,\middle|\, \exists a \in \{1, \ldots, n-1\}, \text{pgcd}(a,n) = 1 \text{ et } a^{2^i m} \neq -1 \mod n \right\}.$$

B_i est sous-groupe strict du groupe $\{a \,|\, a \leq n-1, \text{pgcd}(a,n) = 1\}$. Par le théorème de Lagrange, $|B_i|$ divise le groupe principal et donc $|B_i| \leq 1/2$.

Ainsi, en répétant k fois l'algorithme, la probabilité de se tromper est de $1/2^k$.

L'algorithme. — L'entrée est $n \in \mathbf{N}^*$. On pose

$$n - 1 = 2^k \times m$$

avec m impair.

On choisit aléatoirement $a \in \{1, \ldots, n-1\}$. Si $\text{pgcd}(a,n) \neq 1$ alors on renvoie « composé ».

Si $a^m = 1 \mod n$ alors on renvoie « premier ».

Pour $i = 1, \ldots, k-1$ alors si

$$a^{2^i m} = -1 \mod n$$

alors on renvoie « premier ».

Sinon, on renvoie « composé ».

3. Algorithme de Las Vegas

Un tel algorithme est probabiliste, il donne toujours la bonne réponse quand il se termine mais peut ne jamais se terminer.

Exemple : élection d'un chef. — On a n candidats et l'objectif est d'élire un chef parmi ces candidats. C'est un algorithme en plusieurs tours :

1. Chaque candidat choisit un nombre (de manière aléatoire) entre 1 et le nombre de candidats.
2. Soit t le nombre de 1 tirés par l'ensemble des votants.
   — Si $t = 1$ alors on recommence.
   — Si $t > 1$, seuls les candidats qui ont tirés un 1 restent en jeu.
   — Si $t = 1$, le candidat qui a tiré un 1 est élu.

On peut montrer que l'espérance du nombre de tours nécessaires est constante.