Université de Bordeaux
Mathématiques

Préparation à l'agrégation
2021–2022

Algorithmes de détection de cycles

1. Le lièvre et la tortue

Soit E un ensemble fini à p éléments et une fonction $f\colon E \to E$. On considère une suite $(x_n) \in E^{\mathbb{N}}$ définie par x_0 et la relation de récurrence $x_{n+1} = f(x_n)$, $n \geqslant 0$. Elle est ultimement périodique puisqu'il existe $i \neq j \leqslant p$ tels que $x_i = x_j$ (principe des tiroirs : il y a $p+1$ indices et p valeurs possibles), d'où il suit que $x_{i+s} = x_{j+s}$ pour tout $s \geqslant 0$. Si $i < j$ sont les deux indices minimaux tels que $x_i = x_j$, $P := i$ est la prépériode et $T := j - i$ la période de la suite.

Un algorithme naïf pour détecter une telle collision examine les paires (x_i, x_j) pour $j = 1, 2, \ldots$ et $0 \leqslant i < j$ et s'arrête à la première collision, c'est-à-dire quand $j = T + P$ et $i = P$. Il utilise donc $O((T+P)^2) = O(p^2)$ comparaisons et stocke $T + P = O(p)$ élément de la suite. Le lemme suivant permet un algorithme en $O(T+P) = O(p)$, et ne stocke que 2 éléments :

Lemme. *Soient $0 \leqslant i < j$ les indices minimaux tels que $x_i = x_j$. Il existe $0 < k \leqslant j$ tel que $x_k = x_{2k}$.*

Preuve. On pose $T = j - i$ (période). Pour tout $\lambda \geqslant 0$ et $s \geqslant 0$ entiers, on a

$$x_{i+s} = x_{j+\lambda T+s}\ ;$$

on veut poser $k = i + s$ et $2k = j + \lambda T + s$, pour un $s \geqslant 0$ le plus petit possible. Il faut donc choisir s et λ tels que $2(i+s) = j + \lambda T + s$, soit $s = j - 2i + \lambda T$.

- Si $j \geqslant 2i$, on peut choisir $\lambda = 0$ et on a bien $s = j - 2i \geqslant 0$ et $k = j - i \leqslant j$.
- Sinon, le $s \geqslant 0$ minimal est le reste de la division euclidienne de $j - 2i < 0$ par T : il est strictement inférieur à T, et le quotient $-\lambda$ est bien négatif. On trouve $k = s + i < T + i = j$.

On remarque qu'on a bien $k > 0$ (sinon $i = j = 0$). □

Il suffit donc de considérer les paires (x_k, x_{2k}) pour $k = 1, 2, \ldots$ jusqu'à obtenir une collision. L'algorithme s'arrête après au plus $T + P$ valeurs. La preuve du lemme montre qu'on peut choisir $k < j$ sauf si $i = 0$ (prépériode nulle). Dans ce dernier cas, $j = T$ et $x_0 = x_T$ est la première collision ; l'équation $x_k = x_{2k}$ est équivalente à $T \mid 2k - k = k$ donc le plus petit k possible est T et le lemme est optimal pour ce cas.

On appelle souvent cette méthode « l'algorithme du lièvre et de la tortue », le lièvre parcourant la suite deux fois plus vite que la tortue, jusqu'à la rattraper contrairement à celui de la fable. Elle apparaît dans la littérature à la fin des années 1960s (en 1969, Donald Knuth l'attribue à Robert Floyd, mais sans doute à tort ; c'est un algorithme du folklore).

2. Un autre algorithme

Il utilise une variante du lemme ci-dessus, qui repose sur un principe différent : il s'agit de déterminer la plus petite puissance de 2 supérieure à P et T.

Lemme. *Si* $2^j > \max(P,T)$*, alors* $x_{2^j-1} = x_{2^j+i}$ *pour un indice* $0 \leqslant i < 2^j$*.*

Preuve. Comme $2^j - 1 \geqslant P$, on a $x_{2^j-1} = x_{2^j-1+T}$. On pose $i = T - 1 < 2^j$. □

L'algorithme est maintenant le suivant :

- Soit $y \leftarrow x_0$ et $z \leftarrow y$.
- Pour $j = 0, 1, \ldots$ (ici, y vaut x_{2^j-1})
  - Pour $i = 0, 1, \ldots, 2^j - 1$:
    - $z \leftarrow f(z)$ $(= x_{2^j+i})$.
    - si $z = y$, afficher $(2^j - 1, 2^j + i)$ et stopper.
  - $y \leftarrow z$

Pour $j = 1, 2, ...$, on compare x_{2^j-1} à x_{2^j+i} pour $0 \leqslant i < 2^j$. Remarquons que la dernière valeur de $2^j + i$ dans la boucle intérieure, pour $i = 2^j - 1$, vaut $2^{j+1} - 1$ qui est la prochaine valeur de référence à comparer avec les valeurs suivantes de la suite. Ainsi, on ne calcule jamais deux fois le même x_k ! L'algorithme s'arrête quand 2^j est la plus petite puissance de 2 qui soit $> \max(P,T)$, pour l'indice i du lemme.

C'est de nouveau un algorithme utilisant $O(P+T)$ comparaisons et ne stockant que 2 éléments de la liste. On peut montrer qu'il ne calcule jamais plus d'éléments de la suite que l'algorithme du lièvre et de la tortue. Cet algorithme est dû à Richard Brent. D'autres variantes utilisent plus de mémoire pour déterminer exactement période et prépériode (en gagnant au mieux un facteur constant sur le temps).

3. Application à la factorisation des entiers

Soit $N > 0$ l'entier qu'on désire factoriser et soit $p \mid N$ un diviseur strict de N, par exemple le plus petit diviseur premier de N (p est inconnu). On choisit une fonction $f\colon \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$ polynomiale, par exemple $f(x) = x^2 + 1$ et un $X_0 \in \mathbb{Z}/N\mathbb{Z}$ et on considère la suite définie par la récurrence $X_{n+1} = f(X_n)$. La projection canonique $\mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ permet de définir $x_n = X_n \mod p$ (on sait calculer les X_n mais pas les x_n). Motivés par le paradoxe des anniversaires on *espère* que la suite x_n a une période T et une prépériode P qui sont toutes deux $O(\sqrt{p})$.

Dans ce cas, on applique l'un ou l'autre algorithme à la suite des X_n, en remplaçant le test $x_i = x_j$ dans $\mathbb{Z}/p\mathbb{Z}$ (p étant inconnu, on ne sait pas le faire) par le test $\mathrm{pgcd}(X_i - X_j, N) > 1$ (si $x_i = x_j$, alors p divise ce pgcd). On *espère* que ce pgcd ne vaut pas N, auquel cas on a factorisé N !