Université de Bordeaux **Algèbre et calcul formel – Agrégation**
Mathématiques 2022–2023

FEUILLE D'EXERCICES n° 12

Logarithme discret

Soit G un groupe cyclique et soit g un générateur de G. Soit $a \in G$. Le problème est de trouver un entier k tel que $a = g^k$. On va s'intéresser au cas où $G = (\mathbb{Z}/p\mathbb{Z})^*$, où p désigne un nombre premier.

La commande existe dans `sage` : c'est `log`, appliqué à un élément de G.

À titre d'exercice, on va quand même programmer un algorithme pour ce problème, de complexité algébrique meilleure que l'algorithme naïf (dont la complexité est en $O(p)$) : c'est l'objet de l'exercice 2. L'exercice 1 consiste à retrouver certaines commandes sage en rapport avec le problème.

Exercice 1 – [Commandes sage]
Cet exercice consiste à trouver une commande sage pour chacune des questions.

1) Définir $A = \mathbb{Z}/2027\mathbb{Z}$. Soit G son groupe multiplicatif. Comment trouver un générateur de G par une commande sage ? Soit g un tel générateur. En utilisant `log`, trouver k tel que $456 = g^k$.

2) Comment trouver l'ordre de 76 dans G ?

3) Soit L une liste d'éléments de G. Comment décider si un élément q est ou non dans L ? Si oui, comment déterminer l'indice correspondant ?

Exercice 2 – [Algorithme : Pas de bébé, pas de géant]
Soit G un groupe cyclique de générateur g, et soit a un élément de G. On décrit un algorithme pour trouver k tel que $ag^k = 1$.

Soit N l'ordre de G et $b = \left\lceil \sqrt{N-1} \right\rceil$. Alors $b^2 \geqslant k$. Soit $L = [a, ag, \ldots, ag^{b-1}]$. On trie la liste L : cela va permettre des recherches plus rapides dans L. On calcule ensuite $c = g^{-b}$. Pour $q \in \{0, 1, \ldots, b\}$, on calcule c^q et on cherche si cet élément se trouve dans la liste L. Si c'est le cas, on a trouvé $r \in [[0, b-1]]$ tel que $c^q = ag^r$, c'est-à-dire tel que $ag^{bq+r} = 1$.

1) Expliquer pourquoi l'algorithme termine et est correct (i.e., pourquoi on trouvera toujours un $q \in \{0, 1, \ldots, b\}$ et un $r \in \{0, \ldots, b-1\}$ tels que $c^q = ag^r$).

2) Quelle est la complexité de l'algorithme en terme d'opérations dans G et de comparaison d'éléments de G (on suppose ici que les éléments de G ont une représentation qui permet de définir un ordre total). Votre complexité est-elle meilleure que l'algorithme naïf en $O(|G|)$ opérations dans G ?

Indice : on pourra admettre qu'il existe des algorithmes de tri (par exemple le tri fusion) qui trient une liste de taille ℓ en faisant $O(\ell \log \ell)$ comparaisons.

Indice 2 : on peut trouver un élément dans une liste triée de taille ℓ en $O(\log \ell)$ comparaisons. Comment ?

3) Programmer (et tester) cet algorithme dans le cas où $G = (\mathbb{Z}/p\mathbb{Z})^*$.

4) Vérifier expérimentalement la complexité de l'algorithme dans des groupes $G = \mathbb{Z}/p\mathbb{Z}$ avec des p de plus en plus grands.