25 mai 2022
Master 1 M.F., Orsay

Partiel de M.A.O. Calcul Formel Durée : 3 heures

Lorsqu'on évoque le coût d'un algorithme, on attend toujours une réponse de la forme $O(\dots)$: déterminer la constante implicite dans le symbole $O(\dots)$ n'est pas demandé.

Il est autorisé d'admettre le résultat de certaines questions pour traiter les suivantes. Les parties 1 et 2 sont indépendantes. Elle sont utilisées dans les parties 3 et 4 pour étudier la complexité de certains algorithmes.

Partie 1 : Suites récurrentes d'ordre 1

Soit E un ensemble fini, non vide, de cardinal S. Soit $f : E \to E$ une fonction, et $u_0 \in E$. On considère la suite $(u_n)_{n\geq 0}$ d'éléments de E définie par :

$$\forall n \geq 0, \qquad u_{n+1} = f(u_n). \tag{1}$$

On note j_0 le plus petit entier strictement positif tel que $u_{j_0} \in \{u_0, u_1, \dots, u_{j_0-1}\}$. Comme E est fini de cardinal S, j_0 existe et $j_0 \leq S$. On note alors i_0 l'unique entier strictement inférieur à j_0 tel que $u_{i_0} = u_{j_0}$, et on pose $T = j_0 - i_0$.

1. Démontrer que $u_{n+T} = u_n$ pour tout $n \geq i_0$; on dit que la suite (u_n) est *ultimement périodique* de période T.
2. On pose $x_n = u_{2n}$ pour tout $n \in \mathbb{N}$. Montrer que pour tout $n \in \mathbb{N}^*$ on a l'équivalence suivante : $x_n = u_n$ si, et seulement si, n est multiple de T et $n \geq i_0$.
3. Déduire de la question précédente qu'il existe un entier n, compris au sens large entre i_0 et $j_0 - 1$, tel que $x_n = u_n$.

On veut maintenant comparer les deux algorithmes suivants, qui déterminent des entiers i et j vérifiant $0 \leq i < j$ et $u_i = u_j$. *Le but ici est simplement de trouver un tel couple (i, j), le fait qu'il puisse être distinct du couple (i_0, j_0) n'est pas un problème.*

- Algorithme 1 : calculer u_j pour $j = 1, 2, \dots$, jusqu'à trouver une valeur qu'on a déjà obtenue auparavant.
- Algorithme 2 : en notant $x_j = u_{2j}$, calculer le couple (u_j, x_j) pour $j = 1, 2, \dots$, jusqu'à ce que $u_j = x_j$.

4. En fonction de S, déterminer le coût en temps (mesuré en nombre d'évaluations de la fonction f) de chacun de ces deux algorithmes, et aussi leur coût en mémoire. Quelles conclusions peut-on en tirer ?
5. ($\star$) Implémenter l'algorithme 2 lorsque $E = \mathbb{Z}/p\mathbb{Z}$ avec $p = 10^9+7$, $u_0 = 2$, et $f : E \to E$ est donnée par $f(x) = x^2 + 1$. Quelles valeurs de i et j obtient-on ?

Partie 2 : Suites aléatoires

On considère à nouveau un ensemble fini E de cardinal $S \geq 1$. Soit $(X_n)_{n\geq 0}$ une suite de variables aléatoires à valeurs dans E, mutuellement indépendantes, qui suivent la loi uniforme. Cela signifie que pour tout n et pour tout $(x_0, \dots, x_n) \in E^{n+1}$, la probabilité d'avoir $X_0 = x_0$, $X_1 = x_1, \dots,$ et $X_n = x_n$ est égale à $1/S^{n+1}$. On note A l'évènement suivant : il existe $i, j \in \mathbb{N}$ tels que $0 \leq i < j \leq 1 + 4\sqrt{S}$ et $X_i = X_j$.

6. Pour n entier tel que $0 \leq n \leq S-1$, rappeler combien il y a de $(n+1)$-uplets $(x_0, \ldots, x_n) \in E^{n+1}$ formés d'éléments deux à deux distincts.
7. En déduire la probabilité de A.
8. En utilisant le fait que $e^{-8} < 10^{-3}$, démontrer que $P(A) > 0,999$.
9. ($\star$) Programmer une fonction `Sage` qui prend en entrée l'entier S et renvoie la probabilité de l'évènement A, calculée à la question 7. Combien obtient-on pour $S = 365$?
10. Si on étudie les dates d'anniversaire (supposées indépendantes et uniformément réparties dans l'année) d'un groupe de 78 personnes, quelle conclusion peut-on tirer des questions précédentes ?

Partie 3 : Une méthode de factorisation

Soit $N \geq 2$ un entier. On considère la suite $(u_n)_{n\geq 0} \in \mathbb{Z}^{\mathbb{N}}$ définie par

$$u_0 = 2 \text{ et } u_{n+1} = u_n^2 + 1 \text{ pour tout } n \geq 0.$$

On pose aussi $x_n = u_{2n}$ et $\delta_n = \text{pgcd}(x_n - u_n, N)$.

11. Montrer que pour calculer δ_n, il suffit de connaître les classes de congruence modulo N de x_n et de u_n. Pourquoi cette remarque est-elle fondamentale pour rendre faisable le calcul de δ_n lorsque n est grand ?
12. En supposant x_n et u_n connus (au moins modulo N), rappeler le coût du calcul de δ_n en nombre d'opérations arithmétiques dans $\mathbb{Z}$.

On note n_0 le plus petit entier $n > 0$ tel que $\delta_n > 1$; pour l'instant on admet que n_0 existe. Dans la suite de cette partie, on considère l'algorithme consistant à calculer δ_0, δ_1, δ_2, ..., jusqu'à obtenir δ_{n_0}. Cet algorithme permet d'exhiber un diviseur δ_{n_0} de N qui est différent de 1. Il est cependant possible que $\delta_{n_0} = N$; on dira alors que l'algorithme a échoué.

13. ($\star$) Implémenter cet algorithme en une fonction `Sage` qui prend en entrée l'entier N et renvoie un diviseur non trivial de N si l'algorithme en trouve un, et le mot `Echec` sinon.
14. ($\star$) En utilisant la fonction précédente, déterminer un diviseur non trivial de $F_5 = 2^{2^5}+1$.
15. Calculer en fonction de n_0 et N le coût de cet algorithme en nombre d'opérations arithmétiques dans $\mathbb{Z}$.

On note maintenant p le plus petit facteur premier de N.

16. En utilisant les résultats de la partie 1, montrer que n_0 existe et $n_0 < p$.
17. Proposer une hypothèse (même assez vague) qui, au vu des résultats des parties 1 et 2, incite à penser que $n_0 \leq 4\sqrt{p}$.
18. Déduire des questions 15 à 17 le coût de l'algorithme ci-dessus en fonction de p et de N (en nombre d'opérations arithmétiques dans $\mathbb{Z}$), selon qu'on fait (ou pas) l'hypothèse de la question 17.
19. ($\star$) En utilisant la fonction programmée à la question 13, déterminer un diviseur non trivial de $F_k = 2^{2^k}+1$ pour certaines valeurs de k comprises (au sens large) entre 6 et 12, *sauf pour $k = 7$,* en faisant afficher le temps d'exécution du calcul. *Il est conseillé de laisser la valeur $k = 8$ pour la fin, et de ne la traiter que si l'implémentation s'avère suffisamment rapide.*

20. ($\star$) Les données expérimentales de la question précédente semblent-elles valider l'hypothèse de la question 17 ?
21. Pour quel type d'entiers N cet algorithme est-il le plus utile ? Peut-il constituer une attaque déterminante contre le cryptosystème RSA ? Que peut-on dire, intuitivement, de la probabilité d'échec de l'algorithme ? En cas d'échec, que peut-on faire ?

Partie 4 : Calculs dans un groupe

Dans cette partie on considère un groupe cyclique G de cardinal $N \geq 2$, noté multiplicativement, et deux éléments ω et g de G. On suppose que ω est un générateur de G et on cherche à déterminer un entier i tel que $g = \omega^i$.

22. Si la décomposition de N en produit de facteurs premiers est connue, sous la forme $p_1^{e_1} \dots p_t^{e_t}$ avec $p_1, \dots, p_t$ premiers et deux à deux distincts et $e_1, \dots, e_t \geq 1$, rappeler brièvement comment on peut vérifier efficacement que ω est un générateur de G. Quel est le coût de cette vérification, en nombre de multiplications dans G ?
23. Dans cette question seulement, on suppose connus quatre entiers a, b, a', b' compris (au sens large) entre 0 et $N-1$, tels que $\mathrm{pgcd}(b'-b, N) = 1$ et $\omega^a g^b = \omega^{a'} g^{b'}$. Montrer comment en déduire un entier i tel que $g = \omega^i$, et préciser en fonction de N le coût du calcul de i (exprimé en nombre d'opérations arithmétiques dans $\mathbb{Z}$).
24. Modifier l'algorithme de la question précédente pour remplacer l'hypothèse $\mathrm{pgcd}(b'-b, N) = 1$ par $(a, b) \neq (a', b')$, de façon à obtenir un algorithme efficace lorsque $\mathrm{pgcd}(b'-b, N)$ est petit. Combien coûte alors ce calcul, en fonction de N et de $\mathrm{pgcd}(b'-b, N)$?

Dans toute la suite de cette partie on suppose que $G = A_0 \cup A_1 \cup A_2$ où A_0, A_1 et A_2 sont trois parties non vides de G, deux à deux disjointes. On suppose qu'étant donné un élément de G, on sait tester efficacement à laquelle de ces trois parties il appartient. On définit une fonction $f : G \to G$ en posant :

$$f(x) = \begin{cases} x^2 \text{ si } x \in A_0 \\ \omega x \text{ si } x \in A_1 \\ gx \text{ si } x \in A_2 \end{cases}$$

25. On suppose que la fonction f se comporte comme une fonction aléatoire. En s'inspirant des résultats des parties 1 et 2, et de la question 24, proposer un algorithme qui calcule un entier i tel que $g = \omega^i$ (sauf dans une situation que l'on précisera, où il échoue). En supposant que $\mathrm{pgcd}(b'-b, N)$ est effectivement petit lorsqu'on applique la question 24, pourquoi peut-on raisonnablement espérer qu'en pratique il n'échoue pas, et ait un coût en temps nettement inférieur à celui de l'algorithme naïf ?
26. ($\star$) Implémenter cet algorithme lorsque $p = 127$, $G = (\mathbb{Z}/p\mathbb{Z})^\times$, $\omega = 3$ et $g = 58$. Quelle valeur obtient-on pour i ?

Partie 5 : Bonus

Cette partie ne doit être traitée que si vous êtes certain d'avoir complètement résolu toutes les questions précédentes, et de vous être bien relu.

27. ($\star$) Reprendre la question 19 avec $k = 7$ et/ou $k = 13$, en modifiant l'algorithme pour qu'il soit plus efficace. *Une amélioration significative est nécessaire, optimiser les calculs de façon marginale ne devrait pas suffire.*