ÉCOLE NORMALE SUPÉRIEURE DE LYON

Concours d'admission session 2014

Filière universitaire : Second concours

COMPOSITION D'INFORMATIQUE

Durée : 3 heures

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.

⋆ ⋆ ⋆

Le sujet comporte trois parties et quatre pages. Bien que ces trois parties aient un rapport entre elles, elles sont largement indépendantes et peuvent être traitées comme telles. Il est toujours possible d'admettre les résultats énoncés dans les questions antérieures pour répondre à une question. Toutes vos réponses doivent être justifiées rigoureusement, et prouvées formellement lorsque cela est demandé ou que vous l'estimez nécessaire. La qualité de la rédaction, la précision et la concision des arguments seront largement prises en comptes dans l'évaluation.

Quand une question demande d'écrire un programme, ou une fonction, il est attendu une réponse écrite dans un langage de programmation de haut niveau du programme (Caml, Maple, Pascal). À défaut, on pourra proposer un programme écrit dans un autre langage de haut niveau d'usage largement répandu (C, Java, Python, etc.) que l'on indiquera.

Quand une question demande un algorithme, on donnera une description suffisamment détaillée, qui peut être indifféremment de type itératif ou récursif.

Exponentiation et logarithme discrets

DÉFINITIONS, CONVENTIONS, NOTATIONS COMMUNES À TOUT LE SUJET

Dans tout le sujet, p est un nombre premier fixé. D'un point de vue programmation, on supposera que p est une constante, accessible en lecture depuis toutes les fonctions que le sujet demandera d'écrire.

On note $\mathscr{E}$ l'ensemble $(\mathbb{Z}/p\mathbb{Z})^*$, que l'on identifiera à $\{1,\ldots,p-1\}$ – les éléments de $\mathscr{E}$ seront donc représentés en machine par des entiers, et on supposera que chaque élément de $\mathscr{E}$ occupe un espace 1 en mémoire.

On notera $x \cdot y$ le produit de deux éléments de $\mathscr{E}$ modulo p. Informatiquement, on supposera donnée une fonction `mul` prenant deux arguments x, y de $\mathscr{E}$ et renvoyant le produit $x \cdot y \in \mathscr{E}$. Cette opération sera supposée avoir un coût d'une unité de temps. Par exemple,

– En CAML :
```
let mul x y = x * y mod p;;
```
– En Pascal :
```
function mul(x, y : integer) : integer;
begin
 mul := x * y mod p
end;
```
– En Maple :
```
mul := (x, y) -> x * y mod p;
```

Si $g \in \mathscr{E}$ et t est un entier positif ou nul, on note g^t le produit de t éléments de $\mathscr{E}$ égaux à g, avec la convention usuelle $g^0 = 1$. Les règles de calcul usuelles sur les puissances s'appliquent.

On note $E(\cdot)$ la fonction partie entière d'un nombre réel, et pour $x, n \in \mathbb{Z}$, on notera $x\%n$ le reste de la division euclidienne de x par n.

Si x est un nombre entier non nul et $x = \sum_{j=0}^{n} x_j 2^j$ son écriture binaire, avec $x_n = 1$, $x_i \in \{0,1\}$, on pose $s(x) = n+1$ la longueur de sa représentation binaire. Par convention, $s(0) = 0$.

Enfin, nous noterons $\mathscr{E}^{\mathscr{E}}$ l'ensemble des fonctions de $\mathscr{E}$ dans $\mathscr{E}$, dont le cardinal est $(p-1)^{p-1}$.

Par convention, on supposera toujours qu'un tableau est initialisé à 0, sans que cela ait un coût en terme de calcul. On supposera que les cases d'un tableau de taille ℓ sont numérotées de 0 à $\ell - 1$, et que $t[i]$ désigne le contenu de la case indexée par i. Par exemple, si t est le tableau $[5,2,3,4,1,0]$, $t[0]$ vaut 5 et $t[3]$ vaut 4.

1. Exponentiation discrète

Dans cette partie, on s'intéresse à des algorithmes pour évaluer efficacement l'application $(g,x) \mapsto g^x$, où $g \in \mathscr{E}$ et x est un entier positif ou nul. On mesurera leur efficacité en terme de nombre de multiplications dans $\mathscr{E}$.

1. Écrire une fonction prenant en entrée deux arguments $g \in \mathscr{E}$ et x un entier positif ou nul, et renvoyant un tableau t de taille x tel que $t[i]$ contienne l'élément $g^i \in \mathscr{E}$. Combien de multiplications effectue cette fonction ?

2. Écrire une fonction `Question2` prenant en argument $g \in \mathscr{E}$ et un entier k positif ou nul et renvoyant l'élément $g^{2^k} \in \mathscr{E}$, en faisant au plus k multiplications.

Quand x est une puissance de 2, on sait donc évaluer g^x en $s(x)-1$ multiplications. La suite de la partie vise à se rapprocher de cette situation pour un x général.

3. Vérifier l'identité

$$g^x = \left(\left(g^{E(x/2)} \right)^2 \cdot g^{x\%2} \right).$$

En déduire un algorithme de calcul de g^x. Si $c(x)$ est le nombre de multiplications effectuées par cet algorithme sur l'entrée x, on pose $C(n) = \max_{x/s(x)=n} c(x)$. Montrer que $C(n) \leqslant C(n-1)+2$, en déduire que $C(n) \leqslant \max(0, 2n-2)$.

Soit k un paramètre entier que l'on choisira ultérieurement, et qu'à l'instar de p on traitera comme une constante d'un point de vue informatique.

4. Que calcule l'algorithme suivant ? On justifiera la réponse.

Fonction `Question4`
Entrées : $g \in \mathscr{E}$, x entier positif ou nul, T tableau contenant $[1, g, g^2, \ldots, g^{2^k-1}]$
Faire :
- $y \leftarrow E(x/2^k)$
- Si $y = 0$ alors renvoyer $T[x\%2^k]$.
- $h \leftarrow$ `Question4`(g,y)
- $\ell \leftarrow$ `Question2`(h,k)
- **Renvoyer** $\ell \cdot T[x\%2^k]$

5. Déduire de la question 4 une fonction prenant en argument $g \in \mathscr{E}$, x un entier positif ou nul et renvoyant g^x.

En définissant $c_k(x)$ le nombre de multiplications effectuées par cette fonction sur une entrée x, et $C_k(n) = \max_{x/s(x)=n} c_k(x)$, on s'attachera à ce que la fonction écrite vérifie $C_k(n) \leqslant 2^k + n(1+1/k)$ (et on indiquera pourquoi c'est le cas). Comment choisir k (en fonction de n) de sorte que $C_k(n) \leqslant n + o(n)$ quand $n \to +\infty$?

2. Logarithme discret – une première approche

Dans toute la suite, on s'intéresse à écrire des fonctions qui calculent l'inverse de l'application de la première partie, à savoir étant donné $g,h \in \mathscr{E}$, trouver un entier y (s'il existe) tel que $g^y = h$. L'entier y est appelé un logarithme discret de h en base g.

6. Écrire une fonction de complexité $O(p)$ prenant en argument deux entiers g,h, renvoyant un entier $y \in [0, p-1]$ tel que $g^y = h$, ou -1 si un tel entier n'existe pas.

Dans la suite, on fixe un paramètre entier k, que l'on choisira ultérieurement.

7. Soit $g,h \in \mathscr{E}$. Montrer qu'on a équivalence entre
- Il existe un entier $y \in [0, p-1]$ tel que $h = g^y$;
- Il existe des entiers $r \in [1,k]$, $q \in [1,(p-1)/k+1]$ tels que $0 \leqslant qk - r < p$ et $h \cdot g^r = g^{kq}$.

8. On propose de représenter l'ensemble $A_1 := \{h \cdot g^r, 1 \leqslant r \leqslant k\}$ et l'ensemble $A_2 := \{g^{kq}, q \in [1,(p-1)/k+1]\}$ respectivement par
- un tableau t_1 de taille k tel que $t_1[r-1] = h \cdot g^r$
- et un tableau t_2 de taille $E((p-1)/k)+1$ tel que $t_2[q-1] = g^{kq}$.

Montrer qu'on peut calculer t_1 et t_2 en temps total $O(k + p/k)$.

9. Écrire une fonction de complexité $O(\max(m,n))$ prenant en argument deux tableaux d'entiers positifs triés et leurs longueurs respectives n et m, et renvoyant
- un élément se trouvant dans les deux tableaux s'il en existe un ;
- -1 sinon

10. En déduire que l'on peut résoudre le problème du logarithme discret par un algorithme de complexité $O(\max(k, p/k)\log\max(k, p/k))$.

11. Discuter l'occupation mémoire de l'algorithme de la question 9. Montrer qu'il suffit de stocker t_1 ou t_2 et en déduire un algorithme de complexité $O(\min(k, p/k))$ en espace et

$$O(\max(k, p/k)\log\min(k, p/k))$$

en temps.

12. Quel est le choix de k minimisant la complexité en temps de la méthode décrite dans cette partie ? À l'inverse, si l'on borne l'espace mémoire disponible à m, quelle complexité en temps peut-on obtenir ? Si $m = p$, montrer qu'en modifiant la représentation de A_1 on peut obtenir un algorithme de complexité $O(\sqrt{p})$ – on rappelle qu'on peut toujours supposer qu'un tableau est entièrement initialisé à 0, sans coût de calcul.

3. Logarithme discret – un algorithme heuristique

Dans cette partie, on pose $\mathscr{E}_1 = \mathscr{E} \cap [1, p/3[$, $\mathscr{E}_2 = \mathscr{E} \cap [p/3, 2p/3[$, $\mathscr{E}_3 = \mathscr{E} \cap [2p/3, p[$, et, pour $x \in \mathscr{E}$ on définit

$$f(x) = \begin{cases} h \cdot x, & \text{si } x \in \mathscr{E}_1 \\ x^2, & \text{si } x \in \mathscr{E}_2 \\ g \cdot x & \text{si } x \in \mathscr{E}_3 \end{cases}$$

13. On considère la suite $x_0 = 1$, $x_{n+1} = f(x_n)$. Montrer qu'il existe des suites a_n, b_n telles que pour tout n, $x_n = g^{a_n} h^{b_n}$. Écrire une fonction qui prend en entrée un entier n et renvoie le tableau $[x_n, a_n, b_n]$. Quel est l'ordre de grandeur de sa complexité ?

14. Montrer qu'il existe $0 \leqslant n < n' < p$ tels que $x_n = x_{n'}$. Donner un algorithme qui renvoie $(a_n - a_{n'}, b_{n'} - b_n)$ en une complexité $O(n')$ en temps. Quelle est la complexité en espace de l'algorithme ?

Si t est le plus petit entier strictement positif tel que $g^t = 1$ et si $\text{pgcd}(b_{n'} - b_n, t) = 1$, on calcule alors le logarithme discret comme $(a_n - a_{n'})/(b_{n'} - b_n) \bmod t$.

Dans la suite, nous analysons un modèle de l'algorithme ci-dessus, qui consiste à modéliser la fonction f par une fonction aléatoire de $\mathscr{E}$ dans $\mathscr{E}$. Pour ce faire, étant donnée une fonction $\varphi \in \mathscr{E}^{\mathscr{E}}$, nous définissons la suite $y_0 = 1, y_{n+1} = \varphi(y_n)$ et

$$\kappa(\varphi) = \min\{j \in \mathbb{N} / \exists r > 0, y_{r+j} = y_j\}, T(\varphi) = \min\{r > 0 / y_{r+\kappa(\varphi)} = y_{\kappa(\varphi)}\}.$$

15. Montrer que $n = \kappa(f)$, $n' = \kappa(f) + T(f)$ conviennent dans la question **14.**.

16. Montrer qu'il existe un entier $\kappa(f) \leqslant e(f) \leqslant \kappa(f) + T(f)$ tel que $x_e = x_{2e}$. En déduire un algorithme de complexité $O(\kappa(f) + T(f))$ en temps et de complexité $O(1)$ en mémoire pour résoudre le problème du logarithme discret.

À une fonction $\varphi \in \mathscr{E}^{\mathscr{E}}$, on associe un vecteur $V(\varphi) \in \mathscr{E}^{p-1}$ défini par

$$V(\varphi) = (\varphi(1), \varphi^2(1), \ldots, \varphi^{\kappa(\varphi)+T(\varphi)}(1), \varphi(z_0), \ldots, \varphi(z_{p-\kappa(\varphi)-T(\varphi)-2})),$$

où $z_0 < \cdots < z_{p-\kappa(\varphi)-T(\varphi)-2}$ sont les éléments de $\mathscr{E} - \{1, \varphi(1), \varphi^2(1), \ldots, \varphi^{\kappa(\varphi)+T(\varphi)-1}(1)\}$.

17. Étant donné un vecteur $U = (u_1, \ldots, u_{p-1}) \in \mathscr{E}^{p-1}$ et $s \in [0, p-2]$ tels que $1, u_1, \ldots, u_s$ sont deux-à-deux distincts, montrer qu'il existe une unique fonction $\varphi : \mathscr{E} \to \mathscr{E}$ telle que $\kappa(\varphi) + T(\varphi) > s$ et $U = V(\varphi)$.

18. En déduire que

$$\operatorname{card}\{\varphi \in \mathscr{E}^{\mathscr{E}} / \kappa(\varphi) + T(\varphi) - 1 \geqslant s\} = \frac{(p-2)!}{(p-2-s)!}(p-1)^{(p-1-s)}.$$

19. En déduire que la valeur moyenne de $\kappa + T$, définie comme

$$1 + \sum_{0 \leqslant s \leqslant p-2} \frac{\operatorname{card}\{\varphi \in \mathscr{E}^{\mathscr{E}} / \kappa(\varphi) + T(\varphi) - 1 \geqslant s\}}{(p-1)^{p-1}}$$

vaut

$$1 + \int_0^\infty \left(1 + \frac{t}{p-1}\right)^{p-2} \exp(-t)\,\mathrm{d}t \sim_{p\to\infty} \sqrt{\frac{\pi p}{2}}.$$

On pourra utiliser les identités

$$k! = \int_0^\infty t^k \exp(-t)\,\mathrm{d}t, \qquad \int_0^\infty \exp(-t^2/2)\,\mathrm{d}t = \sqrt{\pi}.$$

En déduire, dans le modèle présenté ici, la complexité de l'algorithme heuristique de la partie 3.