Prologin 2013

Concours national d'informatique

Épreuve écrite d'algorithmique
Paris I

Dimanche 10 février 2013

The Two-Ring Machine

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (3 heures 30).

Conseils

- Lisez bien tout le sujet avant de commencer.
- **Soignez la présentation** de votre copie.
- N'hésitez pas à poser des questions.
- Si vous avez fini en avance, relisez bien, ou préparez votre présentation pour l'entretien.
- N'oubliez pas de passer une bonne journée.

Remarques

- Le barème est donné à titre indicatif uniquement.
- Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
- Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
- Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe, sinon ça va barder.
- Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
- Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 Sujet

Introduction

Soit N un nombre entier supérieur à 4. On considère une machine à deux anneaux [1] marqués de 0 à $N-1$, communément nommés *lièvre* et *tortue*. Ces anneaux, initialisés à 0, peuvent pivoter, de sorte que les variables `lièvre` et `tortue` évoluent entre 0 et $N-1$. Comme nous allons le voir, cette machine permet de détecter des cycles ou de factoriser des nombres et de plus, il s'avère qu'elle constitue un excellent butoir de porte.

Les seules actions possibles pour les anneaux sont les opérations suivantes, appelées *pas* [4] :

```
lièvre = f(lièvre) # mais il ne sera pas rare de faire lièvre = f(f(lièvre))
tortue = f(tortue)
```

L'unique fonction f qui sera utilisée dans tout le sujet sera la suivante :

$$f(x) = (x^2 + 4) \bmod N.$$

Par exemple, si $N = 5$ et que `tortue` vaut 0, après un pas, `tortue` vaudra $(0^2+4) \bmod 5 = 4$. Au pas suivant, `tortue` vaudra $(4^2 + 4) \bmod 5 = 20 \bmod 5 = 0$, et ainsi de suite.

Question 1 (2 points)

> Que se passe-t-il au bout d'un certain nombre de pas ? Pour $N = 9$, écrire les neuf premières valeurs de `tortue`, sachant qu'elle est initialisée à 0.

On peut représenter l'évolution de la tortue en traçant des flèches allant de x à $f(x)$ pour chaque x de 0 à $N-1$, comme sur la Figure 1 ci-dessous qui représente le cas $N = 6$:

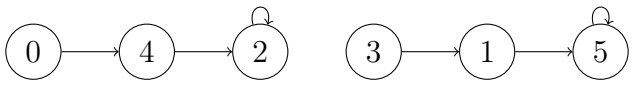

FIGURE 1 – Graphe pour $N = 6$. Par exemple, $4 \to 2$ car $f(4) = (4^2+4) \bmod 6 = 20 \bmod 6 = 2$ et il y a une flèche reliant 4 à $f(4) = 2$.

Question 2 (2 points)

> Pour $N = 9$, calculer $f(x)$ pour tout x de 0 à 8, et dessiner un tel graphe.

Question 3 (3 points)

> Écrire une fonction qui remplit un tableau `t` représentant les différentes valeurs de `tortue` au fur et à mesure des pas (`t[0] = 0`, `t[1] = f(0)`, `t[2] = f(f(0))`, etc.), jusqu'à ce que la tortue tombe sur une case déjà visitée `t[j] = t[i]` avec `i < j`. Ainsi les m éléments `i` à `j - 1` forment un *cycle* tandis que les n éléments `0` à `i - 1` forment une *queue*. Votre fonction devra déterminer m et n.

1. *two-ring* [2] *machine*, en anglais.
2. *two-ring* ne prend pas de S, c'est un *two-word* [3] *modifier*.
3. *two-word* est aussi un *two-word* [3] *modifier*.
4. Quand je dis « appelées *pas* », je ne dis pas qu'on ne les appelle pas, je dis juste qu'on les appelle *pas*.

On dit qu'un algorithme est two-ring *complet* si on peut l'implémenter sur la *two-ring machine*™ [5], c'est-à-dire qu'on ne peut que :

- effectuer un (ou plusieurs) pas sur un anneau (`lièvre` ou `tortue`) ;
- réinitialiser un anneau à 0 ;
- tester si les anneaux sont différents (si `lièvre != tortue`) ;
- faire une boucle `while` ;
- initialiser ou incrémenter un compteur ;
- avoir une mémoire constante (les tableaux de N entiers sont donc proscrits).

On ne peut donc pas comparer un compteur à un autre, ou comparer un anneau à un compteur.

Par exemple, l'algorithme suivant dit *de La Fontaine* est tout à fait *two-ring* complet :

```
# Au départ, lièvre = tortue = 0
lièvre = f(f(lièvre))
tortue = f(tortue)
while lièvre != tortue:
  lièvre = f(f(lièvre))
  tortue = f(tortue)
```

En revanche, celui que vous avez écrit à la question 2 ne l'est pas, visiblement [6].

Question 4 (2 points)

> Pour $N = 9$, et à l'aide de votre dessin à la question 2, écrire un tableau des différentes valeurs que prennent les variables `lièvre` et `tortue` de la *two-ring* machine implémentant l'algorithme de La Fontaine, au fur et à mesure qu'elles évoluent :

Tour	`lièvre`	`tortue`
0	0	0
1	...	...
⋮		

Question 5 (4 points)

> ⋆ Montrer que pour N quelconque, l'algorithme de La Fontaine termine forcément [7].

Question 6 (4 points)

> En vous appuyant sur l'algorithme de La Fontaine, écrire un algorithme *two-ring* complet (dans votre langage favori) qui retourne pour un certain N la longueur m du cycle.

Question 7 (5 points)

> ⋆ Écrire un algorithme *two-ring* complet qui retourne pour un certain N la longueur n de la queue.

5. ™ signifie *two-ring machine*™.
6. En effet, il manipule un tableau qui peut contenir jusqu'à N éléments.
7. N'hésitez pas à faire des dessins, mais soyez rigoureux.

On s'intéresse à présent à déterminer un diviseur d'un nombre N, possiblement grand.

Question 8 (3 points)

> Écrire une fonction qui retourne le PGCD [8] de deux entiers a et b positifs.

Soit k un diviseur connu d'un nombre M. À partir du graphe pour M, il est simple d'obtenir le graphe pour k en remplaçant chaque élément x par $(x \bmod k)$ et de fusionner les différentes occurrences de $(x \bmod k)$ obtenues [9]. Ces fusions font qu'il est possible que `lièvre` et `tortue` arrivent sur la même case dans le graphe pour k alors qu'ils ne sont pas sur la même case dans le graphe pour M : on a `lièvre - tortue` divisible par k mais pas par M.

Question 9 (5 points)

> Si `lièvre` − `tortue` est divisible par p pour un certain p (inconnu) qui divise N, comment pouvez-vous vous aider de votre fonction PGCD, de `lièvre` et de `tortue` pour déterminer un multiple de p divisant N ? ⋆ En déduire une fonction *two-ring* complète [10] susceptible de trouver un diviseur non trivial de N.

Vous pouvez attaquer les questions suivantes si et seulement si vous pouvez attaquer les questions suivantes.

Question bonus 10 (2 points)

Dans quels cas votre programme à la question 9 peut échouer ? Que faire dans ce cas ?

Question bonus 11 (1 point)

Êtes-vous une machine ? Si oui, `:(){ :|: & };:`. Sinon, prouvez-le.

Question bonus 12 (1 point)

Combien y a-t-il de boucles infinies dans ce sujet ? **Attention.** Cette question peut être autoréférente, se référer à la question bonus 12 pour plus d'informations [11].

Question bonus 13 (0 point)

Déduire des questions précédentes un algorithme capable de détecter une boucle infinie d'un autre algorithme. Ça vous servira pour cette après-midi.

Question bonus 14 (1 point)

Entre un lièvre et une Tortue Ninja, lequel est le plus rapide ? Commentez.

Le sujet est sur 30 points, et les questions bonus rapportent au total 5 points, plus 1 point de présentation.

8. Plus grand commun diviseur. Il est interdit de retourner $ab/\mathrm{PPCM}(a, b)$.
9. La preuve de ce résultat n'est pas demandée.
10. À ceci près qu'on vous autorise à utiliser la fonction PGCD.
11. Beaucoup plus.