info528 : Mathématiques pour l'informatique
TD 3 : générateurs aléatoires

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Exercice 1 : Générateurs pseudo-aléatoires : congruences linéaires

Une méthode simple pour générer des nombres pseudo-aléatoire entre 0 et m est d'utiliser la formule

$$x_{n+1} = ax_n + c \pmod m$$

avec un choix approprié de a, c, m et x_0.

Question 1. Un tels générateur est toujours périodique. Pourquoi ?

Question 2. Calculez les nombres générés ainsi que la période si on choisit $a = 6$, $c = 2$, $m = 24$ et $x_0 = 5$.

Question 3. Les vieilles implémentations de la fonction `rand` dans la GLibc utilisaient $m = 2^{31}$. Pourquoi ?

Quels problèmes cela pouvait-il poser ?

Question 4. La page de manuel de la fonction `rand` sur mon système contient la note suivante :

```
However,  on older rand() implementations, and on current
implementations on different systems, the lower-order bits
are much less random than the higher-order bits.
```

Regardez ce qui se passe sur le bit de poid faible de x lorsqu'on utilise une puissance de 2 pour m, et expliquer cette remarque.

Question 5. Comment peut-on utiliser cette méthode pour générer des *bits* aléatoires avec cette méthode ?

Pourquoi ne peut-on pas utiliser $m = 2$?

Note : il ne faut pas utiliser un tels générateur pour un usage cryptographique. En effet, il est assez facile de retrouver m, a et c à partir de quelques valeurs de x.

Exercice 2 : générateurs pseudo-aléatoires : registres à décalage et rétroaction

Question 1. Les générateurs de type *"Linear feedback shift register"* ("Registre à décalage à rétroaction linéaire") permettent de générer des suites de bits ressemblant à des suites aléatoires. Les plus simples sont de type *Fibonacci* où chaque nouveau bit est obtenu en faisant le XOR de quelques bits précédents. Par exemple :

$$b_n = b_{n-1} \oplus b_{n-4} \oplus b_{n-6} \oplus b_{n-30}$$

Dans cet exemple, on peut stocker les 30 bits précédents dans un nombre entier (32 bits). Le bit généré est calculé, puis on fait un décalage à gauche et on ajoute le bit généré dans le bit de poids faible (ou alors on fait un décalage vers la droite, et on ajoute le bit généré dans le 30ème bit).

Supposons que `state` contienne un l'entier correspondant aux 30 bits précédents. Trouvez une suite d'instructions C la plus courte possible qui calcule le bit généré ainsi que le nouvel état.

Question 2. Un tels générateur est forcément périodique. Pourquoi ?

Question 3. Calculez, à la main, les bits générés par

$$b_n = b_{n-3} \oplus b_{n-4}$$

si on commence avec un état qui ne contient que des 1. Quelle est la période du générateur ?

Idem, mais si on commence avec un état qui ne contient que des 0.

Question 4. Écrivez un programme en C pour calculer la période de l'exemple donné ci dessus si on commence avec `state = 0x3fffffff`.

Question 5. Quelle est la période maximale d'un tels générateur qui utilise les valeurs des n bits précédents ?

Question 6. Un autre type de LFSR fonctionne de la manière suivante : à chaque étape, l'état est décalé vers la droite. Si le bit qui disparait à droite est `1`, alors certains bits de l'état sont inversés, par exemple, les bits 1 et 4.

Calculez les bits générés si l'on part de `1111`.

Question 7. Écrivez un programme en C pour calculer la période du générateur si on modifie les bits 1, 4, 6 et 30 en partant de la droite (en partant de l'état `state = 0x3fffffff`).

Essayez de trouver la séquence d'instruction la plus courte possible pour calculer les états successifs. (*Note :* `-1` en complément à 2 est egal à `0xffffffff`, càd que la représentation de `-1` ne contient que des `1`.)

Note : il ne faut pas utiliser un tels générateur pour un usage cryptographique. En effet, il est assez facile de retrouver la formule à partir de quelques bits générés (algorithme de Berlekamp-Massey). Pour obtenir un générateur cryptographiquement plus intéressant, il peut par exemple coupler deux registres à rétroaction (*shrinking generator*).