# info607 : Mathématiques pour l'informatique
# TD 3 : générateurs aléatoires

Pierre Hyvernat
François Boussion
Laboratoire de mathématiques de l'université Savoie Mont Blanc
bâtiment Chablais, bureau 17, poste : 94 22
email : Pierre.Hyvernat@univ-smb.fr
www : http://www.lama.univ-smb.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 $a = 6$, $c = 1$, $m = 24$ et $x_0 = 4$.

*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 expliquez cette remarque.

*Question 5.* Pourquoi ne peut-on pas utiliser $m = 2$ pour générer des *bits* aléatoires avec cette méthode ?

Comment générer des bits aléatoires à partir de congruences linéaires ?

**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 :

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

Calculez les bits générés si on commence avec un état qui ne contient que des 1.

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

*Question 2.* Un tels générateur est forcément périodique. Pourquoi ?
Quelle est la période maximale d'un tels générateur qui utilise les valeurs des $n$ bits précédents ?

*Question 3.* On considère maintenant le générateur

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

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).
En supposant que `state` contient un 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 4.* Écrivez un programme en C pour calculer la période de l'exemple donné ci dessus si on commence avec `state = 0x3fffffff`.
(Remarque : le générateur boucle sur l'état initial...)

**Exercice 3 : mélanger un tableau**

On considère l'algorithme suivant pour mélanger un tableau `T` de taille `t` :

```
for (int i=0; i<t; i++) {
    r = random() % t;      // on tire un nombre aléatoire entre 0 et t-1
    tmp = T[i]; T[i] = T[r]; T[r] = tmp;      // on échange T[i] et T[r]
}
```

*Question 1.* Combien de tirages aléatoires sont effectués ? Combien d'exécutions possibles cela donne t'il ?

*Question 2.* Il y a $\texttt{t}! = 1 \times 2 \times 3 \times \cdots \times \texttt{t}$ permutations distinctes de `T`. Est t'il possible que chacune d'entre elle soit obtenue le même nombre de fois par l'algorithme précédent ?
Qu'en pensez vous ?

On considère maintenant l'algorithme de Fisher-Yates

```
for (int i=t-1; i>0; i--) {
    r = random() % (i+1);      // on tire un nombre aléatoire entre 0 et i
    tmp = T[i]; T[i] = T[r]; T[r] = tmp;        // on échange T[i] et T[r]
}
```

*Question 3.* Combien y a t'il d'exécutions possibles ?
Qu'en pensez vous ?