Licence d'Informatique 2014–2015 **Semestre 6**
Probabilités et Combinatoire **J1IN6016**

Algorithmes randomisés

Exercice 1 Vous disposez d'une source de bits aléatoires qui retourne 1 avec la probabilité p, et 0 avec la probabilité $1-p$. *La probabilté p est inconnue.* Proposer un algorithme qui retourne les bits aléatoires 0 et 1 avec les probabilités *exactement égales* à $1/2$ et $1/2$. Combien de fois en moyenne votre algorithme s'adresse-t-il à la source initiale pour retourner un bit ?

Exercice 2 Il y a trois polynômes $P(x)$, $Q(x)$, $R(x)$, $\deg P = \deg Q = n$, $\deg R = 2n$. On affirme que le produit $P \cdot Q = R$; notre objectif est de vérifier si cette affirmation est vraie ou fausse. Deux méthodes sont proposées :

1. Première méthode : multiplier P par Q et voir si le résultat est égal à R.
2. Deuxième méthode : répéter k fois le test suivant :
   – choisir au hasard un entier $x \in [-10n, 10n]$;
   – calculer $y_1 = P(x)$, $y_2 = Q(x)$ et $z = R(x)$ et vérifier si $y_1 y_2 = z$.
   Si au moins une fois on obtient $y_1 y_2 \neq z$ on conclue que l'affirmation $PQ = R$ est fausse. Si chaque fois on obtient $y_1 y_2 = z$ on conclue que l'affirmation est vraie.

Comparer la complexité des deux algorithmes. Quelle est la probabilité que le deuxième algorithme se trompe ? (Trouver un majorant pour cette probabilité.)

Exercice 3 Appliquer le test de primalité de Fermat à $n = 39$, $x = 4$.

Exercice 4 La méthode ρ de Pollard pour déterminer un facteur non-trivial d'un entier n consiste en les opérations suivantes. Soit la fonction

$$f(x) \;=\; x^2 + 1 \bmod n\,.$$

– On tire un entier x_0 dans $[0, n-1]$ et on pose $y_0 = x_0$;
– on calcule les termes successifs des suites x_i et y_i définies, pour $i \geq 1$, par les relations de récurrence $x_i = f(x_{i-1})$, $y_i = f(f(y_{i-1}))$;
– à chaque étape on calcule $d_i = \mathrm{pgcd}(|x_i - y_i|, n) \bmod n$, et on s'arrête quand on obtient $d_i \neq 1$;
– si $d_i = 0$ l'algorithme échoue ; sinon, d_i est le facteur cherché.

Justifier l'algorithme. Appliquer cet algorithme au cas $n = 77$ en choisissant pour x_0 la valeur 0. Appliquer l'algorithme au cas $n = 1001$ en prenant la valeur de x_0 à votre choix. On précisera à chaque fois le facteur retourné et le nombre d'itérations.

QuickSort randomisé

Il s'agit d'un algorithme de tri qui se présente comme suit. Soit un ensemble S dont les éléments sont deux à deux comparables, $|S| = n$. (Pour faciliter l'analyse supposons qu'il n'y aient pas d'éléments égaux.)

1. Si $n = 1$ retourner S.
2. Sinon, choisir un élément-pivot $z \in S$ au hasard.
3. Passer en revue tous les éléments $s \in S$ en les comparant avec z. Mettre l'ensemble $X = \{s \in S \mid s < z\}$ "à gauche", puis mettre z lui-même et, enfin, mettre l'ensemble $Y = \{s \in S \mid s > z\}$ "à droite".
4. Appliquer récursivement le même algorithme aux ensembles X et Y.

Notez que deux éléments $x \in X$ et $y \in Y$, une fois passés, à l'étape 3, l'un à gauche, l'autre à droite du pivot, ne sont plus jamais comaparés entre eux.

Exercice 5 Soit $T = S$ le même ensemble, et soient $t_1 < t_2 < \ldots < t_n$ ses éléments pris en ordre croissant. (Avant l'application de l'algorithme cet ensemble ordonné n'existe qu'en notre imagination. On ne peut donc l'utiliser dans l'algorithme, mais on peut porter sur lui notre reflexion.) Soient $t_i, t_j \in T$ les éléments de T de rangs $i < j$. Quelle est la probabilité que ces deux éléments serons comparés entre eux pendant l'application de l'algorithme ?

Exercice 6 Soit $X_{i,j}$, $i < j$, une variable aléatoire,

$$X_{i,j} = \begin{cases} 1 & \text{si } t_i \text{ et } t_j \text{ sont comparés entre eux,} \\ 0 & \text{sinon,} \end{cases}$$

et soit

$$X = \sum_{1 \le i < j \le n} X_{i,j}$$

le nombre total de comparaisons effectuées par l'algorithme QuickSort.

1. Calculer $\mathrm{E}(X)$.
2. Proposer un plus simple majorant pour $\mathrm{E}(X)$.

Exercice 7 Montrer que $\displaystyle\sum_{k=2}^{n} \frac{1}{k} \le \int_1^n \frac{1}{x} dx$.

Exercice 8 En utilisant les résultats des exercices 5–7, estimer la complexité moyenne de l'algorithme QuickSort randomisé.