Université de Caen Normandie — Année 2016-2017
Master 2 e-secure

Outils Algorithmiques
Devoir donné par Etienne Grandjean le 14 octobre 2016
A rendre impérativement, individuellement
et au plus tard le jeudi 3 novembre 2016

Ce devoir donnera lieu ensuite à une soutenance orale individuelle de 10 à 15 minutes.

Consignes à respecter, à lire attentivement :

- Commencez dès maintenant votre devoir !
- Vous traiterez tous *l'exercice court 1* et *l'exercice long 4 (Test probabiliste pour l'isomorphisme d'arbres)* et vous traiterez *au choix* (choix à préciser en tête de votre copie) *l'un des deux exercices 2 (Election de leader) et 3 (Exponentiation modulaire)* qui demandent chacun un peu de programmation et d'expérimentation.
- Vous rendrez votre devoir sous forme *manuscrite* ou sous la forme du tirage papier (ou envoi email) d'un *fichier pdf. Pas d'autre format !* De plus, pour l'exercice choisi 2 ou 3, vous rendrez, en plus du fichier de votre devoir, le *code du programme* (Python, C++, Java, ou...) demandé ainsi que le *listing des résultats* obtenus par votre programme.
- Dans tous les cas, lisez d'abord, très attentivement et jusqu'à la fin, chaque énoncé. Justifiez clairement vos réponses. La clarté de vos réponses sera un élément déterminant de la notation.
- Il y a beaucoup d'indications pour vous aider (c'est pourquoi le sujet est long, mais les réponses sont souvent courtes...). Utilisez ces indications ! Les questions sont très proches de questions traitées en cours et TD. Utilisez aussi vos notes de cours !
- Si vous ne savez pas traiter une question (après avoir cherché), passez aux suivantes en admettant la réponse. Cela vaut mieux que de donner une réponse que vous ne maîtrisez pas, ce que je pourrai contrôler lors de votre soutenance...

Exercice 1. **Un couple veut avoir une fille** (d'après le livre [Probability and Computing : Randomized Algorithms and Probabilistic Analysis, Mitzenmacher & Upfal], exercice 2.8, page 39).

Question 1. Alice et Bob décident d'avoir des enfants *jusqu'à* ce qu'ils aient *exactement* une fille. En supposant que c'est possible (ils ne limitent pas le nombre de leurs enfants !), quel nombre moyen de garçons ont-ils alors avant d'avoir une fille ?
Remarque : Pour cette question, comme pour la question suivante, on fait les deux hypothèses suivantes : à chaque naissance, la probabilité d'avoir une fille ou un garçon est exactement $1/2$ pour chaque sexe et il n'y a jamais de naissance multiple.

Question 2. Supposons maintenant qu'Alice et Bob décident d'avoir des enfants jusqu'à ce qu'ils aient exactement une fille ou k enfants, pour un entier $k \geq 1$ fixé (par exemple, $k = 3$). Dans ce cas, combien ont-ils de filles en moyenne ? Combien ont-ils de garçons en moyenne ? Comparez ces résultats avec ceux de la question précédente. Ces résultats correspondent-ils à l'intuition ?

Pour chacune de ces deux questions, vous présenterez votre raisonnement avec soin ; le calcul à faire n'est pas très difficile mais doit être bien présenté et justifié.

Exercice 2. **Election de leader : étude théorique et expérimentation.**
Nous avons étudié en cours le problème de l'élection de leader dans un anneau de n processeurs anonymes et identiques : pour ce problème, nous avons présenté un protocole/algorithme probabiliste de type Las Vegas qui, en rompant la symétrie de l'anneau, permet d'élire un unique leader parmi les n processeurs : ce protocole est la répétition d'une même suite d'instructions ; cette suite d'instructions est appelée une "phase" du protocole.

Question 1. Rappelez quel est ce protocole dans le cas général et ce à quoi il se ramène dans le cas $n = 2$. On note T_n la variable aléatoire qui représente le nombre de phases de l'algorithme pour n processeurs. Quelles valeurs peut prendre T_n ? Pourquoi ?

Question 2. Quel est, dans le cas particulier $n = 2$, le nombre moyen de phases ainsi que la variance et l'écart-type de ce nombre, donc $E(T_n)$, $V(T_n)$ et $\sqrt{V(T_n)}$? On pourra utiliser des éléments du cours, mais toujours en les justifiant.

Question 3. *Programmation :* Ecrivez (dans votre langage favori : Python, C++, Java...) un programme très simple qui réalise l'élection de leader dans le cas général, donc pour un entier fixé quelconque $n > 1$. Evidemment, votre programme ne fera que "simuler" de façon séquentielle le protocole probabiliste (pas de parallélisme dans votre programme alors que le protocole probabiliste est réparti sur les n processeurs de l'anneau).

Question 4. *Expérimentation :* Exécutez votre programme pour chacune des valeurs de n suivantes : 2, 3, 5, 10, 100 et 1000. Pour chacune de ces valeurs de n, votre programme calculera le nombre T_n de phases, exécutera le protocole 1000 fois et calculera la moyenne expérimentale de T_n pour ces 1000 exécutions. Que constatez vous ?

Exercice 3. **Exponentiation modulaire : mesurer le nombre de multiplications et la complexité.** Comme on l'a vu en cours, l'exponentiation modulaire est l'une des opérations de base les plus utilisées en cryptographie (test de Miller-Rabin, protocoles RSA et El Gamal) et doit donc être programmée de la façon la plus efficace possible. Il s'agit de calculer, pour trois entiers naturels a, p et $n \geq 1$ tels que $a < n$, l'entier $a^p \mod n$.

Question 1. Rappelez l'algorithme classique qui calcule l'exponentiation modulaire.

Question 2. Si l'exposant $p \geq 1$ s'écrit en binaire sur exactement ℓ bits (donnez la valeur exacte de ℓ en fonction de p), combien de multiplications modulaires l'algorithme effectue-t-il pour calculer $a^p \mod n$? Justifiez. Donnez l'intervalle des valeurs possibles de ce nombre de multiplications en fonction de la longueur ℓ de p, c'est-à-dire le maximum et le minimum de ce nombre de multiplications. Pour quelles valeurs de p obtient-on respectivement ce maximum et ce minimum ? Donnez le nombre exact de multiplications effectuées en fonction de la longueur ℓ de la représentation binaire de p et du nombre u de bits 1 de cette représentation.

Question 3. Donnez, en fonction de ℓ la complexité du calcul de $a^p \mod n$ pour des entiers a, p et n, chacun écrit sur ℓ bits, sachant qu'on réalise chaque multiplication en utilisant la FFT (transformée de Fourier rapide) et que la multiplication de deux entiers de ℓ bits s'effectue (avec la FFT) en temps $O(\ell \log \ell)$ (admis). Donnez l'ordre de grandeur de cette complexité pour $\ell \approx 10\ 000$. Si la taille ℓ est multipliée par 2, par combien environ le temps de calcul de l'exponentiation modulaire est-il multiplié ? Justifiez.

Question 4. Programmez l'algorithme classique (que vous avez rappelé à la question 1) en lui faisant calculer aussi le nombre de multiplications modulaires qu'il réalise.

Question 5. Avec votre programme, calculez $a^p \mod n$ (pour des valeurs a et n fixées à votre convenance) pour 1000 valeurs de p choisies aléatoirement parmi les entiers qui ont exactement 20 bits, donc prises au hasard dans l'intervalle $[2^{19}, 2^{20} - 1]$ et calculez avec votre programme la moyenne du nombre de multiplications effectuées. (Vous pourrez renouveler l'expérience plusieurs

fois.) Quelle(s) moyenne(s) trouvez-vous ? Expliquez votre résultat en utilisant la conclusion de la question 2.

Exercice 4. **Test probabiliste de polynôme et isomorphisme d'arbres** (d'après le livre [The Design and Analysis of Algorithms, Kozen], Chapitre 40, pages 211-215).
Nous avons étudié en cours le test probabiliste de Schwartz (1980) et Zippel (1979) qui teste si un polynôme donné sous une forme quelconque (typiquement, avec des produits, sommes et différences, mais *sans exposant*) est identiquement nul.

Question 1. Rappelez l'énoncé du théorème de Schwartz et Zippel sur lequel s'appuie leur test probabiliste. Prouvez ce théorème dans le cas où le polynôme p à tester porte sur une seule variable et est de degré d quelconque. De même, prouvez ce théorème dans le cas où le polynôme p est de degré $d = 1$ avec un nombre quelconque m de variables. (Les preuves sont très simples.)

Question 2. Rappelez le test probabiliste de Schwartz et Zippel qui teste si un polynôme p est identiquement nul. Précisez les conditions à réaliser pour que ce test probabiliste fonctionne, c'est-à-dire, pour borner la probabilité d'erreur. Dans quel cas le test peut-il se tromper ? Avec quelle erreur ? Comment faire pour que la probabilité d'erreur soit inférieure à 10^{-15} ?

Question 3. Comment utiliser le test de Schwartz et Zippel pour tester si deux polynômes p_1 et p_2 sont identiques ?

Comme on a commencé à le voir dans le cours, on voudrait utiliser le test probabiliste de Schwartz et Zippel pour tester si deux arbres enracinés non ordonnés sont isomorphes. On donne ci-dessous une définition récursive des *arbres enracinés non ordonnés*, qu'on appellera pour simplifier des *arbres*. Pour une meilleure formalisation, on présente chaque arbre sous la forme d'un mot sur l'alphabet à quatre symboles $\{f, r, (,)\}$:

Définition

1. f est un *arbre* de *hauteur* 0 et de *taille* 0 ;
2. si pour un entier $k > 0$, $A_1, \ldots, A_k$ sont des *arbres* de *hauteurs* respectives $h_1, \ldots, h_k$, alors $r(A_1, \ldots, A_k)$ est un *arbre* A de *hauteur* $1 + \max(h_1, \ldots, h_k)$ et de *taille* $k + \text{taille}(A_1) + \ldots + \text{taille}(A_k)$; on dit que r est la *racine* de l'arbre A et que $A_1, \ldots, A_k$ sont les *sous-arbres immédiats* de A.

Exemple d'arbre : $A = r(r(f), r(r(f)), r(f, r(f, f, f)))$. Les sous-arbres immédiats de A sont $r(f)$, $r(r(f))$ et $r(f, r(f, f, f))$.

— Représentez cet arbre A sous la forme graphique habituelle d'un arbre.
— Quelle est la hauteur de cet arbre A ? Quelle est la taille de A (au sens de la définition ci-dessus [1]) ?

Encore quelques définitions ou remarques :

— Le *nombre de feuilles* d'un arbre est son nombre de symboles f (6 pour l'exemple).
— Remarquez que la *taille* d'un arbre est exactement son nombre de liens parent-enfant (11 pour l'exemple). Pourquoi ? (Voir définition donnée ci-dessus en 1-2.)
— Les *sous-arbres* d'un arbre sont ses sous-arbres immédiats et les *sous-arbres* de ses sous-arbres immédiats ; pour notre exemple d'arbre A, ce sont $r(f)$, $r(r(f))$ et $r(f, r(f, f, f))$ (les sous-arbres immédiats de A), et aussi f et $r(f, f, f)$ (certains de ces sous-arbres apparaissent plusieurs fois dans A).

Intuitivement, deux arbres sont isomorphes s'ils ont la même forme, c'est-à-dire, s'ils représentent le même arbre, à permutation près de leurs sous-arbres. Plus formellement, on définit cette notion par induction :

1. Cette notion de taille est différente de la notion habituelle de taille d'un arbre qui est son nombre de noeuds.

1. deux arbres de hauteur 0 (donc égaux à f) sont *isomorphes* ;
2. deux arbres, de hauteurs non nulles, $A = r(A_1, \ldots, A_k)$ et $B = r(B_1, \ldots, B_{k'})$, sont *isomorphes* si les deux conditions suivantes sont vraies : ils ont le même nombre de sous-arbres immédiats, donc $k = k'$, et il existe une permutation π de l'ensemble des indices $\{1, \ldots, k\}$ telle que A_i est *isomorphe* à $B_{\pi(i)}$, pour tout indice $i \in \{1, \ldots, k\}$.

Exemple : Vérifiez que l'arbre $A = r(r(f), r(r(f)), r(f, r(f, f, f)))$ donné en exemple est isomorphe à l'arbre $B = r(r(r(f, f, f), f), r(f), r(r(f)))$. Pour cela, constatez que les sous-arbres immédiats de A sont $r(f)$, $r(r(f))$ et $r(f, r(f, f, f))$ qui sont, à permutation près et isomorphie près, les mêmes que les sous-arbres immédiats de B, qui sont $r(r(f, f, f), f)$, $r(f)$ et $r(r(f))$.

Comme nous l'avons vu en cours, on peut associer à chaque arbre A (arbre enraciné non ordonné) de hauteur h un polynôme, noté p_A, portant sur les $h + 1$ variables $x_0, \ldots, x_h$. Ce polynôme est défini par induction sur la hauteur de l'arbre :

— si l'arbre A est de hauteur 0, donc est égal à f, son polynôme associé est $p_A = x_0$;
— si l'arbre A est de hauteur $h \geq 1$ et donc de la forme $A = r(A_1, \ldots, A_k)$, avec $k > 0$, alors son polynôme associé est $p_A = (x_h - p_{A_1}) \ldots (x_h - p_{A_k})$, où $p_{A_1}, \ldots, p_{A_k}$ sont les polynômes associés aux k sous-arbres immédiats $A_1, \ldots, A_k$ de A.

Exemple : Vérifiez que le polynôme associé à l'arbre $r(f, r(f, f, f))$ est $(x_2 - x_0)(x_2 - (x_1 - x_0)^3)$ qu'on écrira plutôt $(x_2 - x_0)(x_2 - (x_1 - x_0)(x_1 - x_0)(x_1 - x_0))$.

Question 4. Quel est le polynôme p associé à l'arbre $A = r(r(f), r(r(f)), r(f, r(f, f, f)))$ donné en exemple ci-dessus ?
Attention ! Donnez seulement l'expression de p, comme dans l'exemple précédent, *sans* développer cette expression !

Question 5. Démontrez par induction que, pour tout arbre A, le degré de son polynôme associé p_A est égal au nombre de feuilles de A.

Indication : Remarquez que, pour un arbre A de hauteur non nulle, les termes $p_{A_1}, \ldots, p_{A_k}$ de son polynôme $p_A = (x_h - p_{A_1}) \ldots (x_h - p_{A_k})$ sont tous de degré au moins 1 et ne contiennent pas la variable x_h (pourquoi ?).

On appelle *taille* du polynôme p_A associé à un arbre A le nombre d'occurrences de variables dans p_A quand on écrit p_A sans utiliser de puissance (qui est une abréviation). Par exemple, on associe à l'arbre $r(f, r(f, f, f))$ le polynôme $(x_2 - x_0)(x_2 - (x_1 - x_0)(x_1 - x_0)(x_1 - x_0))$, de taille 9.

Question 6. Démontrez (toujours par induction) que, pour tout arbre A, la taille de son polynôme associé p_A est égale à la somme taille(A) + nombreFeuilles(A).

En déduire qu'on a taille(p_A) = O(nombreNoeuds(A)) où nombreNoeuds(A) est le nombre de noeuds de A, c'est-à-dire, le nombre total de symboles r (noeuds internes) et f (feuilles) de l'arbre A.

Question 7. Soient deux arbres A et B et soient p_A et p_B leurs polynômes associés. Prouvez que A et B sont isomorphes si et seulement si les conditions suivantes sont toutes deux réalisées :

1. A et B sont de même hauteur h et de même taille et ont le même nombre de feuilles d ;
2. les polynômes associés p_A et p_B sont identiques (ou, de façon équivalente, le polynôme $p_A - p_B$ est identiquement nul).

Indication : Vous utiliserez (sans le prouver) le fait que la factorisation d'un polynôme est unique, à l'ordre des facteurs près. En particulier, si les polynômes $p_A = (x_h - p_{A_1}) \ldots (x_h - p_{A_k})$ et $p_B = (x_h - p_{B_1}) \ldots (x_h - p_{B_{k'}})$ associés à des arbres A et B de hauteur $h \geq 1$ sont identiques, alors $k = k'$ et les k facteurs de p_A sont identiques aux k facteurs de p_B, à permutation près de ces facteurs.

Question 8. En déduire un test probabiliste pour tester si deux arbres A et B sont isomorphes.

Question 9. Quelle est la complexité de ce test ? Vous supposerez que chaque tirage aléatoire d'un nombre et chaque opération arithmétique (soustraction, multiplication) s'effectuent en temps constant 1.