Informatique II

Le sujet porte sur des algorithmes de *génération aléatoire* : dans une famille d'objets, par exemple les arbres couvrants d'un graphe G donné, on veut choisir un élément au hasard, de façon à ce que tous les éléments aient la même probabilité d'apparaître. Ce type d'algorithmes est utilisé par exemple pour l'étude expérimentale de configurations aléatoires en physique statistique ou pour la détermination de seuils de significativité dans l'analyse des génomes.

Le sujet comporte des préliminaires, puis un exercice et un problème indépendants. La lecture des préliminaires est utile pour traiter l'exercice ou le problème. À l'intérieur d'une partie, il est possible d'admettre les résultats d'une question pour traiter la suite. On prendra soin de justifier les réponses données, y compris le fonctionnement des algorithmes et en particulier leur terminaison.

Préliminaires

On supposera dans toute le sujet qu'on dispose d'un générateur parfait BA de bits aléatoires : chaque appel de BA renvoie un bit aléatoire 0 ou 1 avec probabilité 1/2. On note cela $\Pr(\mathtt{BA} = 0) = \Pr(\mathtt{BA} = 1) = 1/2$. Notre objectif est d'étudier des algorithmes qui font appel au générateur BA. Considérons par exemple l'algorithme suivant, qui tire une suite de bits aléatoires et s'arrête après avoir vu k zéros.

---

algo `stop(d)`
- $\ell := 0$;
- tant que $\ell < d$ faire { si BA()$= 0$ alors $\ell := \ell + 1$;}

Cet algorithme n'est pas déterministe : deux appels identiques `stop(10)` peuvent provoquer deux exécutions différentes. L'algorithme `stop` peut *a priori* ne pas terminer, par exemple si la suite de bits renvoyés par le générateur est une suite infinie de 1. Cependant ceci n'est pas le comportement qu'on s'attend à observer : en pratique l'algorithme `stop` termine « toujours » et on voudrait une notion de terminaison qui rende compte de cette intuition.

Question 1. Fixons une suite $s = (b_1, \ldots, b_i)$ de i bits. Quelle est la probabilité $p(s)$ qu'une suite de i appels à BA donne exactement s ?

Question 2. Décrire l'ensemble $S_d(i)$ des suites de i bits associées aux exécutions de `stop(d)` qui terminent après exactement i appels de BA.

Question 3. Calculer la probabilité $p_d(i) = \sum_{s \in S_d(i)} p(s)$ qu'une exécution de `stop(d)` termine après exactement i appels de BA et en déduire la probabilité que l'exécution de `stop(d)` termine. (On rappelle la formule du binôme : pour $d \geq 0$, et $|x| < 1$, on a $\sum_{n \geq 0} \binom{n+d-1}{d-1} x^n = (1-x)^{-d}$.)

On dira dans la suite qu'un algorithme $\mathcal{A}$ *termine avec probabilité 1* sur l'ensemble des données $\mathcal{D}$, si pour tout $d \in \mathcal{D}$, la probabilité de terminaison de l'appel $\mathcal{A}(d)$ est 1. Dans le même ordre d'idées, nous allons considérer la *complexité en moyenne* plutôt que la complexité dans le pire cas (qui serait infinie pour l'algorithme `stop`).

Question 4. Montrer que le nombre moyen de bits aléatoires utilisés par l'appel `stop(d)` s'écrit $\sum_{i \geq 0} i \, p_d(i)$ et que cette quantité est un $O(d)$.

Un *générateur aléatoire uniforme* pour un ensemble fini E est un algorithme ΓE qui renvoit un élément de E et tel que, pour tout $e \in E$, $\Pr(\Gamma E = e) = 1/|E|$.

Question 5. Donner un générateur aléatoire uniforme $\mathtt{Rand}(n)$ pour $\{0, \ldots, n-1\}$, d'abord avec $n = 2^k$, puis pour n quelconque. Indiquer le nombre moyen de bits aléatoires utilisés.

Exercice

Soit S_n l'ensemble des permutations de $\{1, \ldots, n\}$; on rappelle qu'une permutation de $\{1, \ldots, n\}$ est une bijection de cet ensemble dans lui-même, et que $|S_n| = 1 \cdot 2 \cdot 3 \cdots n = n!$. Par ailleurs, l'ensemble $\mathcal{B}$ des arbres binaires complets est défini récursivement : un arbre binaire complet est formé d'un nœud (appelé racine de l'arbre) et de deux sous-arbres ; chaque sous-arbre est formé d'une feuille ou d'un arbre binaire complet. Soit $\mathcal{B}_n$ l'ensemble des arbres binaires complets à n nœuds (et, par construction, à $n+1$ feuilles).

Les 3 questions sont indépendantes. On précisera dans les réponses à ces 3 questions la représentation choisie pour les structures manipulées, la complexité en moyenne des algorithmes et le nombre moyen de bits aléatoires utilisés.

Question 6. Donner un générateur aléatoire uniforme `permute`(n) de permutations de S_n.

Question 7. Proposer une construction ϕ pour fabriquer à partir
- d'un triplet (A, s, c) tel que $A \in \mathcal{B}_n$, s est un nœud ou une feuille de A et $c \in \{\text{gauche}, \text{droit}\}$,
- une paire (A', f) telle que $A' \in \mathcal{B}_{n+1}$ et f est une feuille de A',

de façon que chaque triplet corresponde à une unique paire et réciproquement.

En déduire (même si vous n'avez pas trouvé la bijection) une récurrence pour le nombre d'arbres dans $\mathcal{B}_n$, et un générateur aléatoire uniforme `Binaire`(n).

Question 8. Donner une borne inférieure sur le nombre de bits aléatoires nécessaires à la génération d'une permutation de S_n et à celle d'un arbre binaire complet de $\mathcal{B}_n$.

Problème : Cycles eulériens aléatoires

Soit $G = (V, E)$ un graphe orienté : $V = \{1, \ldots, n\}$ est l'ensemble des sommets, $E = \{1, \ldots, m\}$ est l'ensemble des arcs, et à tout arc $i \in E$ est associé une origine $i^- \in V$ et une extrémité $i^+ \in V$. En particulier les boucles ($i^+ = i^-$) et les arcs multiples ($i^+ = j^+$ et $i^- = j^-$ pour $i \neq j$) sont autorisées. Le degré sortant du sommet x est $d_-(x) = |\{i \in E \mid i^- = x\}|$ et son degré entrant est $d_+(x) = |\{i \in E \mid i^+ = x\}|$.

Une *marche* de x à y de longueur k est une suite d'arcs $(a_1, \ldots, a_k)$ de E tels que $a_1^- = x$, $a_k^+ = y$ et pour $1 \leq i < k$, $a_i^+ = a_{i+1}^-$. Un *circuit* issu de x est une marche de x à x. Un *chemin* (respectivement un *lacet*) est une marche (respectivement un circuit) dont les sommets sont distincts deux à deux. Dans tout le problème, on suppose le graphe G *fortement connexe* : pour tout x et y de V, il existe une marche de x à y.

Un graphe est un *arbre* s'il contient un sommet *racine* x tel que de tout sommet y part une unique marche vers x. Un *arbre couvrant* de G est un ensemble d'arcs $A \subset E$ tel que le graphe (V, A) soit un arbre. (Nos arbres ont donc tous leurs arcs orientés « vers la racine ».)

Question 9. Donner un algorithme qui construit un arbre couvrant du graphe G.

Un *circuit eulérien* est un circuit qui emprunte exactement une fois chaque arc de E (il est donc de longueur m). Le graphe G est dit *eulérien* s'il existe un circuit eulérien sur G.

Question 10. Donner une condition nécessaire et suffisante sur les degrés pour que G soit eulérien.

Question 11. Donner une bijection entre les circuits eulériens de G issus de x et les couples $(A, (O_y)_{y \in V})$ formés d'un arbre couvrant A de racine x et pour chaque sommet $y \in V$ d'un ordre O_y sur les arcs sortants de y qui n'appartiennent pas à A.

Marche à lacets effacés et empilement de lacets

Étant donnée une liste C de longueur ℓ, on note $C_{[j]}$ le j-ème élément, $C_{[<j]}$ la liste des $j-1$ premiers éléments et $C_{[\geq j]}$ la liste des $\ell - j + 1$ derniers éléments. Enfin, la concaténation d'un élément x à la fin d'une liste C est notée $C \cdot x$. On considère l'algorithme suivant, connu

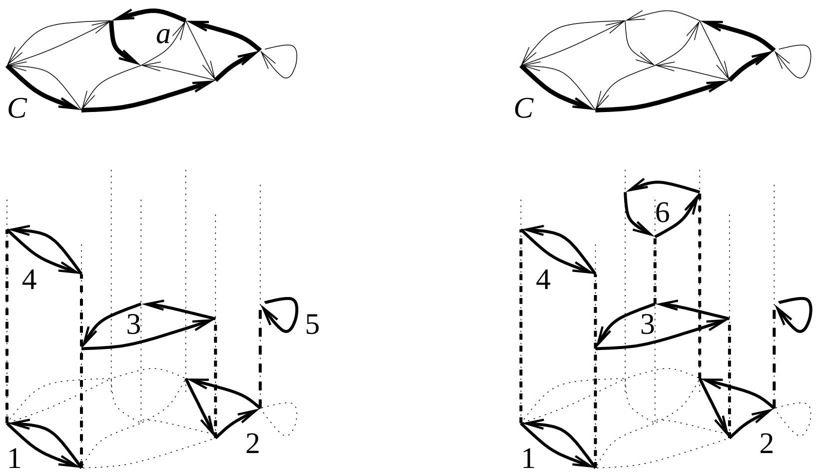

FIG. 1 – Un pas dans l'exécution de l'algorithme `efface`. En haut : le chemin C dans le graphe G. En dessous, la suite des lacets déjà formés représentée comme empilement de lacets. À gauche, le prochain arc a de la marche M est indiqué. À droite, le parcours de l'arc a a créé un sixième lacet. Les lacets maximaux sont 4 et 5 dans l'empilement de gauche ; 4, 5 et 6 dans celui de droite. On vérifie que dans les deux cas $\mathcal{B} \cdot C$ est une pyramide car C intersecte tous les chemins maximaux de $\mathcal{B}$.

sous le nom de *marche à lacets effacés*. On note $V(M)$ l'ensemble des sommets parcourus par une marche M issue de x_0 ; par convention $V(M) = \{x_0\}$ si la marche M est vide.

---

algo `Efface(M)` : la donnée est une marche $M = (a_1, \dots, a_k)$ d'origine $x_0 = a_1^-$.

– $C := ()$;	(C est un chemin d'origine x_0)
$\mathcal{B} := ()$;	($\mathcal{B}$ est une liste de lacets)
– pour i de 1 à k, faire	
– si $a_i^+ \in V(C)$	(détection d'un lacet)
alors soit j tel que $C_{[j]}^- = a_i^+$;	(le lacet est $C_{[\geq j]} \cdot a_i$)
$\mathcal{B} := \mathcal{B} \cdot (C_{[\geq j]} \cdot a_i)$; $C := C_{[<j]}$;	(ajout du lacet à $\mathcal{B}$ et effacement)
sinon $C := C \cdot a_i$;	
– renvoyer C et $\mathcal{B}$.	

Question 12. Montrer que la donnée du chemin C et de la liste de lacets $\mathcal{B}$ permet de retrouver la marche M.

On dit que deux suites de marches sont *équivalentes* si on peut passer de l'une à l'autre par une suite de commutations de la forme $(B_1, \dots, B_i, B_{i+1}, \dots, B_k) \sim (B_1, \dots, B_{i+1}, B_i, \dots, B_k)$ avec B_i et B_{i+1} des marches disjointes (*i.e.* $V(B_i) \cap V(B_{i+1}) = \emptyset$). Un *empilement de marches* est une classe d'équivalence de suites de marches. Une marche B est *maximale* dans l'empilement $\widetilde{\mathcal{B}}$ s'il existe un représentant de la classe $\widetilde{\mathcal{B}}$ tel que B soit la dernière marche de la liste. Une *pyramide de marches* est un empilement de marches ayant un unique élément maximal.

Question 13. Soit C un chemin et $\widetilde{\mathcal{B}}$ un empilement de lacets tels que la concaténation $\widetilde{\mathcal{B}} \cdot C$ forme une pyramide. Montrer qu'il existe une unique marche M dont l'image par `Efface` est formé du chemin C et d'un représentant $\mathcal{B}$ de la classe $\tilde{\mathcal{B}}$.

Arbres couvrants et circuits eulériens aléatoires.

À chaque sommet y de G on associe un générateur aléatoire uniforme $\mathtt{Arc}(y)$ qui renvoie un arc choisi uniformément parmi les arcs issus de y : $\Pr(\mathtt{Arc}(y) = e)$ vaut $1/d^-(y)$ si $e^- = y$ et 0 sinon. L'algorithme suivant produit un arbre couvrant de G et un empilement de lacets.

---

algo `Arbre(G)`
- $V_1 = \{n\}$; $A := \emptyset$;
- Pour x de 1 à $n-1$ répéter
  - $M_x = ()$; $y := x$; (construction d'une marche
  - tant que $y \notin V_x$ répéter de x à un sommet de V_x :
    - $e := \mathtt{Arc}(y)$; – choix d'un arc issu de y,
    - $M_x := M_x \cdot e$; $y := e^+$; – avancée d'un pas.)
  - $(C_x, \mathcal{B}_x) :=$`efface`(M_x) ;
  - $V_{x+1} := V_x \cup V(C_x)$;
  - ajouter les arcs de C_x à A ;
- renvoyer A et l'empilement $\mathcal{B} = \mathcal{B}_1 \cdots \mathcal{B}_{n-1}$.

Question 14. Montrer que les couples $(A, \mathcal{B})$ formés d'un arbre couvrant de racine n et d'un empilement de lacets qui évitent le sommet n sont en bijection avec les exécutions possibles de l'algorithme `Arbre(G)`.

Question 15. En déduire que deux arbres couvrants de G ont la même probabilité d'être renvoyés par l'appel `Arbre(G)`.

Question 16. Soit G un graphe eulérien. Donner un algorithme probabiliste $\mathtt{euler}(G)$ qui renvoie un circuit eulérien choisi uniformément parmi tous les circuits eulériens sur G.

Application aux séquences aléatoires contraintes

Le problème suivant est une version simplifiée d'une question issue du traitement des séquences génomiques. Étant donné un fragment s d'ARN, on souhaite produire un échantillon de séquences aléatoires « proches » du fragment s pour en étudier les propriétés statistiques. Une définition fréquente de séquences « proches » consiste à imposer la même distribution des facteurs de longueurs q pour un $q \geq 2$ fixé. L'objectif est de juger la pertinence d'un « évènement » (par exemple, mon fragment contient un motif répété très long) : si le même genre d'évènements se produit dans des séquences aléatoires proches, alors ça n'est sans doute pas un évènement significatif.

Soit $q \geq 2$ un entier et $\mathcal{A}$ un alphabet fini. À un mot w de longueur m sur $\mathcal{A}$, on associe l'application $\theta_w : \mathcal{A}^q \to \mathbb{N}$ qui compte le nombre d'occurrences des facteurs de longueur q dans le mot w. Pour simplifier on complète w cycliquement de façon à avoir m facteurs au total. (En d'autres termes pour $w = w_1 \cdots w_m$, on promène une fenêtre de longueur q sur le mot $w_1 \cdots w_m w_1 \cdots w_{q-1}$ et on compte les facteurs apparaissants.)

Ainsi pour $q = 3$ et $w = TATATCG$, les 7 facteurs de longueur 3 sont successivement TAT, ATA, TAT, ATC, TCG, CGT et GTA. D'où en particulier $\theta_w(TAT) = 2$, $\theta_w(AAA) = 0$.

Question 17. Soit w un mot de longueur m sur $\mathcal{A}$ et W l'ensemble des mots tels de même q-composition que w, c'est-à-dire, $W = \{w' \mid \theta_{w'} = \theta_w\}$. Donner un algorithme de génération aléatoire uniforme des mots de W.