Théorie de l'information

M1 Informatique 2013-2014

Fiche de TD 2

Codes

Exercice 1. (Codes et algorithme de Sardinas et Patterson)

On rappelle qu'un ensemble C fini de mots est un *code* si, pour tous mots $u_i \in C$ et $v_j \in C$ tels que $i \in [n]$ et $j \in [m]$, l'égalité

$$u_1 \cdot u_2 \dots u_n = v_1 \cdot v_2 \dots v_m$$

implique $n = m$ et $u_i = v_i$ pour tout $i \in [n]$.

1. Construire un code qui contient quatre mots sur l'alphabet $\{\mathtt{a}, \mathtt{b}\}$.
2. Démontrer que l'ensemble $\{\mathtt{a}, \mathtt{ab}, \mathtt{ba}\}$ n'est pas un code.
3. Expliquer pourquoi l'ensemble $\{\mathtt{b}, \mathtt{ab}, \mathtt{aab}\}$ est un code.
4. Expliquer pourquoi l'ensemble $\{\mathtt{a}, \mathtt{ab}\}$ est un code.
5. Démontrer que tout ensemble C fini de mots est un code si et seulement si pour tous mots $u_i \in C$ et $v_j \in C$ tels que $i \in [n]$ et $j \in [m]$, l'égalité $u_1 \dots u_n = v_1 \dots v_m$ implique $u_1 = v_1$.
6. Démontrer que tout ensemble C de mots ayant tous la même longueur est un code.
7. Démontrer que tout ensemble C de mots tel qu'aucun mot n'est préfixe propre d'un autre est un code.

L'*algorithme de Sardinas et Patterson* permet de décider si un ensemble C fini de mots est un code. Il se décrit de la manière suivante :

(i) **initialisation :** $U_0 := C^{-1} \cdot C \setminus \{\epsilon\}$;

(ii) **itération :** $U_{i+1} := U_i^{-1} \cdot C \cup C^{-1} \cdot U_i$;

(iii) **arrêt** : si $\epsilon \in U_i$ alors C n'est pas un code et si $U_i = U_j$ avec $i > j$, alors C est un code.

On rappelle que si L et K sont deux ensembles de mots, l'ensemble $L^{-1} \cdot K$ est défini par

$$L^{-1} \cdot K := \{\text{mot } v : \text{ il existe } u \in L \text{ tel que } u \cdot v \in K\}.$$

8. Donner un argument pour justifier le fait que cette suite d'instructions termine sur l'entrée de tout ensemble fini de mots.

En appliquant l'algorithme de Sardinas et Patterson, décider si les ensembles suivants sont des codes :

9. $C_1 := \{\mathtt{ab}, \mathtt{baa}, \mathtt{abba}, \mathtt{aabaa}\}$;
10. $C_2 := \{\mathtt{aaa}, \mathtt{aba}, \mathtt{abb}, \mathtt{abaab}\}$;
11. $C_3 := \{\mathtt{a}, \mathtt{abba}\}$;
12. $C_4 := \{\mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}\}$;
13. $C_5 := \{\mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{baaababa}\}$;
14. $C_6 := \{\mathtt{a}, \mathtt{ab}, \mathtt{bba}, \mathtt{bbab}, \mathtt{bbbb}\}$.

Exercice 2. (Arbres, codes préfixes et longueur de mots)

Soit $A := \{\mathsf{a}_1, \ldots, \mathsf{a}_k\}$ un alphabet de taille k. On considère une suite de nombres entiers positifs

$$s := (s_1, s_2, \ldots, s_n)$$

telle que

$$\sum_{i=1}^{n} \frac{s_i}{k^i} \leq 1. \tag{1}$$

On souhaite montrer de manière constructive que si la suite s vérifie (1), alors on peut construire un code préfixe C_s sur l'alphabet A qui contient exactement s_i mots de longueur i.

1. Montrer que c'est possible si pour tout $i \in [n]$, on a $s_i \leq k - 1$.
2. Soit s une suite qui vérifie (1) et telle qu'il existe $r \in [n]$ tel que $s_r \geq k$. On définit $\Theta_r(s)$ comme la suite vérifiant

$$\Theta_r(s)_i := \begin{cases} s_r - k & \text{si } i = r, \\ s_{r-1} + 1 & \text{si } i = r - 1, \\ s_i & \text{sinon,} \end{cases}$$

   pour tout $i \in [n]$. Montrer que la suite $\Theta_r(s)$ vérifie (1).
3. Soit s une suite qui vérifie (1) et telle qu'il existe $r \in [n]$ tel que $s_r \geq k$. En supposant que l'on dispose d'un code préfixe $C_{\Theta_r(s)}$, déterminer un moyen de construire C_s.
4. En déduire un algorithme qui, étant donnée une suite s vérifiant (1) construit un code préfixe sur A avec exactement s_i mots de longueur i.

Exercice 3. (Algorithme de Varn)

L'*algorithme de Varn* consiste, sur l'entrée d'un alphabet $A := \{\mathsf{a}_1, \ldots, \mathsf{a}_k\}$, d'une fonction de coût $c : A \to \mathbb{N} \setminus \{0\}$ et d'un entier n de la forme $n = \ell(k-1) + 1$ où ℓ est un entier strictement positif, à construire un code préfixe de cardinal n sur l'alphabet A.

Cet algorithme construit un arbre T d'arité k de la manière suivante :

(1) l'arbre T est initialisé en une racine étiquetée par 0 ;

(2) Pour i allant de 1 à ℓ :
   (a) soit x une feuille de T avec une plus petite étiquette ;
   (b) pour chaque lettre a_i de A, construire un nœud étiqueté par l'étiquette de x plus $c(\mathsf{a}_i)$ et attacher ce nœud à x par une arête étiquetée par a_i.

1. Expliquer comment, à partir d'un arbre T produit par l'algorithme de Varn, on obtient un code préfixe X_T de cardinal n sur A.
2. Un code préfixe X est *maximal* si pour tout code préfixe Y tel que $X \subseteq Y \subseteq A^*$, on a $X = Y$. Montrer que tout code obtenu par l'algorithme de Varn est un code préfixe maximal.
3. Construire un code préfixe de cardinal 7 sur l'alphabet $A := \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ où l'on a $c(\mathsf{a}) := 1$, $c(\mathsf{b}) := 2$, $c(\mathsf{c}) := 3$ et $c(\mathsf{d}) := 4$ en appliquant l'algorithme de Varn.
4. Construire un code préfixe de cardinal 7 sur l'alphabet $A := \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ où l'on a $c(\mathsf{a}) := 2$, $c(\mathsf{b}) := 4$ et $c(\mathsf{c}) := 5$ en appliquant l'algorithme de Varn.
5. Construire un code préfixe de cardinal 13 sur l'alphabet $A := \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ où l'on a $c(\mathsf{a}) := 1$, $c(\mathsf{b}) := 3$ et $c(\mathsf{c}) := 4$ en appliquant l'algorithme de Varn.
6. Reformuler en détail la description de l'algorithme de Varn en explicitant les structures de données utilisées. Évaluer sa complexité en espace et en temps en fonction du cardinal n du code préfixe construit.
7. Le coût $c(X)$ d'un code préfixe X sur A est défini par

$$c(X) := \sum_{x \in X} c(x_1) + \cdots + c(x_{|x|}).$$

   Calculer les coûts des codes construits précédemment.
8. Un code préfixe X de cardinal n sur A est *optimal* s'il minimise $c(X)$. Démontrer que tout code construit par l'algorithme de Varn est optimal parmi les codes maximaux.