TD 3 : Complexité de Kolmogorov et compression de données

1 Compression à base de dictionnaires

Exercice 1 Avant d'étudier des algorithmes de compression informatiques (LZW et variantes), discutez les avantages et inconvénients d'une compression de textes en langue naturelle à l'aide de dictionnaires "classiques", tels que le Littré [1] ou le TLFi [2].

1. Les grands dictionnaires classiques réclament avoir entre 80.000 et 100.000 "entrées". Quel est le problème avec ces chiffres en relation avec la compression de textes ?
2. Est-ce qu'un tel dictionnaire peut être utilisé sans adaptation pour coder n'importe quel texte en langage naturel (par exemple un article dans un journal) ?
3. Supposons un dictionnaire avec 100.000 entrées, et une longueur moyenne d'un mot de 4 caractères. Calculez le taux de compression que vous pouvez atteindre si vous codez un texte avec des références au dictionnaire au lieu de le coder en UTF-8. Prenez uniquement en compte les mots, négligez les séparateurs (espaces et signes de ponctuation).
4. En tant qu'experts de la théorie de l'information de Shannon, vous détectez certainement un potentiel d'optimisation si on parle de la longueur moyenne des mots. Lequel ?

Exercice 2 On devrait espérer que toute fonction de compression (`zip` ...) permet de réduire la taille des donnés que l'on souhaite comprimer. Le résultat suivant montre que ce n'est pas possible.

Soit $A \neq \{\}$ un alphabet et A^* l'ensemble de mots formés sur A.

1. Montrez que pour toute fonction $f : A^* \to A^*$ de compression, il existe au moins un mot $m \in A^*$ dont la longuer n'est pas réduite par la compression, donc $|f(m)| \geq |m|$, où $|m|$ est la longueur du mot m.
2. Montrez une propriété plus forte : pour toute fonction $f : A^* \to A^*$ de compression, pour toute longueur k, il existe au moins un mot $m \in A^*$ avec $|m| = k$ tel que $|f(m)| \geq |m|$. *Note :* utilisez le principe des tiroirs et le fait que la fonction de compression f doit être injective.

Exercice 3 Exécutez l'algorithme de compression (et ensuite, de décompression) sur les chaînes suivantes, avec un alphabet `a, b, c` :

1. `abbcabbc`
2. `aabbbaaca`

Exercice 4 Montrez que le dictionnaire construit par la procédure de compression de LZW est fermé par préfixes : Si un mot w' est dans le dictionnaire, alors aussi tout préfixe $w \preceq w'$ non vide.

Exercice 5 Nous nous intéressons à la compression d'une chaîne de caractères `aa ... a` (uniquement composée de lettres `a`) avec l'algorithme LZW, et en supposant que le dictionnaire initial comporte uniquement le caractère `a` (et pas d'autres caractères).

1. Quelle sera la structure du dictionnaire après lecture d'un certain nombre de caractères `a` ?
2. Quelle est la chaîne d'entrée si le résultat de la compression est la séquence des codes `[0, 1, 2, ...` $k-1$`]` ?

1. http://www.littre.org/
2. http://atilf.atilf.fr/tlf.htm

3. Inversement, pour une chaîne d'entrée de n lettres `a`, quelle est la structure de la séquence des codes ? Et par conséquent, sa taille ?
4. Peut-être, vous auriez attendu un taux de compression plus fort pour une chaîne à n lettres, donc une fonction $f(n)$ qui croît moins fortement. Quel est le taux de compression que vous pouvez atteindre en principe, si vous écrivez le nombre de `a` en décimal / en binaire ? Le format décimal est plus compact. Mais montrez que la représentation décimale se distingue uniquement par un facteur constant de la représentation binaire.

Exercice 6 Cet exercice construit explicitement des séquences qui ne peuvent pas être comprimés avec l'algorithme LZW. L'existence de telles séquences n'est pas surprenante, étant donné le résultat de l'exercice 2. On s'interroge aussi sur la croissance du dictionnaire, dont la taille peut empêcher un codage efficace.

1. Cet excercice est un jeu / un concours : qui arrive à construire la chaîne d'entrée la plus longue sur l'alphabet `a, b, c, d` qui n'admet aucune compression ? Autrement dit, votre solution doit être une séquence de codes qui est aussi longue après compression que la chaîne de caractères avant compression.
2. Généralisez cette observation : étant donné un alphabet avec n caractères, à partir de quelle longueur est-ce qu'une chaîne de caractères s est forcément comprimée, indépendamment de la forme particulière de s ?
3. Nous disons qu'un dictionnaire sur un alphabet $\{a_1 \dots a_n\}$ est *saturé* jusqu'au niveau k s'il contient "toutes les combinaisons possibles" d'entrées dont la taille est $\leq k$. Par exemple, pour l'alphabet $\{a, b, c\}$ et $k = 2$, un dictionnaire saturé est $\{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc\}$. Quelle est la cardinalité d'un dictionnaire saturé jusqu'au niveau k sur un alphabet de taille n ?
   (a) Calculez précisément la taille d'un dictionnaire saturé pour $n = 4$ et $k = 5$.
   (b) Donnez l'ordre de grandeur d'un dictionnaire saturé pour $n = 32$ et $k = 5$. Le résultat que vous obtenez est un argument *pratique* pour limiter la taille d'un dictionnaire.
4. Donnez un argument informel pour montrer que, pour tout alphabet $\{a_1 \dots a_n\}$ et tout niveau k, il existe des séquences s dont la compression produit un dictionnaire saturé.
5. Supposez que vous travaillez avec un dictionnaire saturé jusqu'au niveau k, et vous devez coder le mot m des k caractères suivants qui se trouvent dans le dictionnaire. Le code de ce mot a une représentation binaire. Quelle est sa taille ? Comparez avec la taille (en binaire) de ce mot si vous ne le codez pas avec le dictionnaire. Constat ? Effectuez le calcul concrètement pour $n = 4$ et $k = 5$.