Algorithmique : TD 4
Licence M/I - 11 octobre 2003

# 1 Recherche de motif : algorithme de Rabin-Karp

On se place sur l'alphabet $\Sigma = \{0, 1, \cdots, 9\}$. Le motif $w$ de longueur $m$ peut donc être considéré comme un entier $v(w)$ à $m$ chiffres. Pour le texte $t$, on note $t_i$ le facteur de $t$ de longueur $m$ qui commence à la position $i$, qui est aussi un entier $v(t_i)$ à $m$ chiffres. On cherche donc une position $i$ telle que $v(t_i) = v(w)$.

**Exercice 1:** Comment calculer $v(t_0)$ avec un minimum d'opérations de base $(+, *)$ ?

**Exercice 2:** Comment calculer $v(t_1)$ à partir de $v(t_0)$ rapidement ?

**Exercice 3:** Donner un algorithme de recherche de motif efficace.

En pratique, on ne peut pas gérer des entiers de taille aussi grande que l'on veut. On va donc travailler modulo $q$, en ne gardant que $v(t_i)\%q$ et $v(w)\%q$.

**Exercice 4:** Comment modifier l'algorithme dans ce cas ?

**Exercice 5:** Donner son temps d'exécution dans le pire cas et un exemple où ce temps est atteint.

# 2 Algorithme de compression de Lempel et Ziv

Cet algorithme est un algorithme de compression de fichiers : l'objectif est de pouvoir transformer un mot en un autre mot plus court, puis de pouvoir faire l'opération inverse pour retrouver le mot original.

**Exercice 6:** Pourquoi un algorithme de compression qui n'entraîne pas une perte d'information ne peut-il pas comprimer tous les fichiers ?

On dispose d'un texte $s$ sur l'alphabet $\Sigma = \{a, b\}$ qu'on désire comprimer en un texte $d$. Pour cela, on construit un arbre binaire au fur et à mesure qu'on lit le texte de la manière suivante :

1. Au départ, l'arbre binaire n'a qu'un nœud numéroté 0.
2. On se place à la racine de l'arbre ($i = 0$).
3. On lit une lettre du texte, s'il s'agit d'un $a$, on essaie d'aller dans le fils gauche du nœud courant, sinon dans le fils droit. Si ce nœud existe, on s'y place et on recommence l'étape 3.
4. Sinon, on incrémente $i$, on écrit dans $d$ le numéro du nœud dans lequel on se trouve suivi de la lettre qu'on vient de lire. On crée un nœud $i$ et on l'ajoute comme fils (gauche ou droit selon la lettre lue) au nœud courant. On retourne ensuite en 2.
5. Lorsqu'il n'y a plus de lettre à lire, on écrit dans $d$ le numéro du nœud courant.

**Exercice 7:** Comment est compressé le mot *abbaabbaababbaaaabaabba* ?

**Exercice 8:** Comment peut-on décoder le mot $0a0b1a2b3a2a5a5b$ ?

**Exercice 9:** Est-ce que tous les mots sur $\Sigma = \{a, b\}$ peut être compressés par l'algorithme ? Est-ce que tous les mots contenant des $a$, des $b$ et des nombres peuvent être décompressés ?

En réalité, les "lettres" que l'on lit sont des bits. Et les numéros que l'on insère dans le fichier sont aussi représentés sous forme binaire. Ainsi, un fichier compressé de la forme $0a1b0b2a2b4b$... s'écrira, si on n'y prend garde 0011011001011001... (en remplaçant $a$ par 0, $b$ par 1, et les nombres par leur écriture binaire) ; ce qui risque d'être fort difficile à décoder.

**Exercice 10:** Proposer une solution pour résoudre ce problème. Comment gérer la fin du fichier ?