Licence M/I

Algorithmique avancée: TD 3

10 octobre 2005

Exercice 1: Algorithme de Rabin-Karp

Dans tout ce qui suit, le texte est dénoté par t, de longueur n, le motif est w, de longueur m. i désigne une position dans w et j la position de w par rapport à t. Soit d la taille de l'alphabet et L la taille du plus grand entier représentable en mémoire ($L = 2^{32} - 1$ par exemple).

Le principe de l'algorithme de Rabin et Karp est le suivant : on crée une fonction h qui associe à n'importe quel mot de longueur m un entier de $[0; L]$ (fonction de hachage). Pour chaque facteur u de t de longueur m, on calcule $h(u)$. Si $h(u) = h(w)$, on vérifie (naïvement) si $u = w$.

i) Pourquoi faut-il vérifier que $u = w$?

ii) Proposer une fonction de hachage telle qu'on peut calculer la valeur du facteur commençant en $j + 1$ à partir de celle du facteur commen̄cant en j sans relire toutes les lettres.

iii) Ecrire un algorithme qui utilise cette fonction de hachage (sans calculer de tableau qui contiendrait les valeurs des facteurs de t).

Exercice 2: Algorithme de Lempel et Ziv

Cet algorithme est un algorithme de compression de fichiers. On supposera dans ce qui suit que l'alphabet a deux lettres (a et b), par exemple, le fichier à comprimer est un fichier binaire. On dispose d'un texte t qu'on désire comprimer en un fichier c. Pour cela, on construit un arbre binaire au fur et à mesure qu'on lit le texte.

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 c 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 c le numéro du nœud courant.

i) De manière générale, pourquoi un algorithme de compression qui n'entraîne pas une perte d'information ne peut-il pas comprimer tous les fichiers ?

ii) Faire tourner l'algorithme de Lempel et Ziv sur le mot *abbaabbaaabaab*.

iii) Comment peut-on décoder le fichier c ?

iv) 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. Proposer une solution. Comment gérer la fin du fichier ? Par exemple, un fichier fait un nombre entier d'octets, il faut donc que le nombre total de bits soit un multiple de 8.