Algorithmique : TD 8

Licence M/I

14 novembre 2005

1 Algorithme de Huffman statique

Soit A un ensemble (ou *alphabet*) de lettres. Chacun des symboles de A occupe *a priori* une place identique en mémoire (par exemple 1 ou 2 octets). L'idée de l'algorithme de compression de Huffman est, pour un texte t donné, d'encoder les lettres par un nombre variable de bits : peu pour les lettres fréquentes dans t, plus pour les lettres rares.

Lors du premier passage dans le texte, on calcule le nombre d'occurrences n_a de chaque lettre a rencontrée.

On construit des arbres dont les nœuds internes sont étiquetés par des entiers et les feuilles par des couples lettre/entier. Le poids d'un arbre est l'entier contenu dans sa racine. Une forêt (liste d'arbres) est triée si les arbres sont rangés par ordre croissant de poids.

L'algorithme est initialisé de la façon suivante : pour chaque élément a de A_t, on crée l'arbre réduit à une feuille étiquetée par le couble (a, n_a). On considère la forêt triée formée à partir de ces arbres.

Tant que la forêt a au moins deux arbres, on fusionne les deux arbres minimaux en ajoutant un nœud racine dont le poids est égal à la somme des poids des arbres fusionnés, puis on trie la forêt.

Exercice 1: Que doit-on faire pour trier la forêt ?

Exercice 2: Comment cet arbre permet-il de coder le texte ? Quelle structure de données faut-il utiliser pour que ce codage soit efficace ?

Exercice 3: Comment gérer le fait qu'un fichier doit faire un nombre entier d'octets ?

Exercice 4: Comment décoder le fichier compressé ? Y a-t-il besoin de placer des séparateurs entre les codes successifs des différentes lettres ?

Exercice 5: Décrire le fichier compressé de *abracadabra*.

2 Algorithme de Huffman dynamique

L'algorithme de Huffman statique présente deux inconvénients. D'une part, on est obligé d'effectuer un premier passage pour construire la fonction de compression, ce qui nécessite en particulier de connaître tout le texte avant de commencer la compression proprement dite. D'autre part, le fichier compressé doit comporter un préambule qui contient la description de la fonction de compression.

L'algorithme de Huffman dynamique permet de pallier ces inconvénients. L'arbre de codage utilisé dépend dans ce cas uniquement du texte déjà lu et évolue en fonction des symboles qu'on rencontre. Pour cela, on manipule des arbres binaires qui vérifient la propriété suivante : on peut numéroter les nœuds $c_1, c_2, ...c_{2n-1}$ de telle sorte que

1. le poids d'un nœud interne est la somme des poids de ses fils ;
2. les poids des nœuds $p(c_1)$, $p(c_2)$,... forment une suite croissante ;

3. pour tout i dans $[1; n]$, les nœuds $p(c_{2i-1})$ et $p(c_{2i})$ sont frères.

On initialise l'arbre avec une feuille de poids 0 qui ne contient aucun symbole. On maintiendra une telle feuille dans l'arbre tout au long de la compression.

On suppose qu'on à déjà lu le texte t, qu'on a formé l'arbre $\mathcal{A}_t$ et qu'on lit une lettre a.

– *Si on a déjà rencontré a,* on met à jour l'arbre $\mathcal{A}_t$:
  1. c_i est la feuille correspondant à a,
  2. on incrémente le poids de c_i de 1,
  3. si la suite des poids n'est plus croissante, on échange c_i avec le plus grand c_j de poids inférieur à c_i,
  4. on passe au père de c_i, et on recommence à partir de l'étape 2 jusquà la racine.

– *Si on n'a jamais rencontré a,* on remplace la feuille 0 par un arbre à deux feuilles, celle de gauche vide avec poids 0, celle de droite contenant a avec poids 1. On réorganise ensuite l'arbre à partir de la feuille contenant a de la même façon que dans le premier cas.

Exercice 6: Décrire comment déduire de ces manipulations un algorithme de compression et de décompression.

Exercice 7: Décrire le fichier compressé de *abracadabra*.

— ∘ —

Exercice 8: Que fait l'algorithme suivant :

```
x := Racine();
repeter{
   si Existe-fils(x) alors
       x := Fils-aine(x);
   sinon{
       tant que non Est-racine(x) et non Existe-cadet(x) faire
           x := Pere(x);
       si Existe-cadet(x) alors
           x := Cadet(x);
   }
}jusqu'a Est-racine(x);
```

Peut-on l'adapter pour obtenir un parcours préfixe, suffixe ou infixe de l'arbre ?

Exercice 9: Algorithme RSA

Cet algorithme est un algorithme de cryptage à clé publique, ce qui signifie qu'un individu A rend publique une *clé* qui permet à n'importe qui de coder un message que seul A peut décoder. Le principe est le suivant :

a) A choisit en secret deux entiers premiers (grands) p et q et calcule $n = pq$. Il choisit un entier k (petit).

b) A rend publics les entiers n et k

c) Pour coder un message $m \in [0; n-1]$ destiné à A, un individu B doit calculer $m^k \mod n$.

1. Que doit faire A pour décoder le message ? Quelles sont les conditions sur k pour que cet algorithme fonctionne correctement ? Que doit faire un "espion" pour décoder un message qui ne lui est pas destiné ?

2. Comment A peut-il *signer* un message, c'est-à-dire diffuser publiquement un message dont n'importe qui puisse s'assurer qu'il provient bien de A.