Algorithmique : TD 8
Licence M/I - 22 novembre 2004

1 Codage de Huffman statique

Exercice 1: On utilise le code suivant :

A	B	C	D	E	F	G	H
000	001	010	011	100	101	110	111

Quelle est la taille du codage de $BACADAEAFABBAAAGAH$?

Exercice 2: Même question avec le code :

A	B	C	D	E	F	G	H
01	00	1010	1011	1100	1101	1110	1111

Exercice 3: Donner un code plus efficace pour $BACADAEAFABBAAAGAH$ que ceux des questions 1 et 2.

Exercice 4: Comment décoder le mot 001011010011101101 avec le code

A	B	C	D	E	F
0	1	00	01	10	11

Exercice 5: Donner une solution au problème rencontré dans l'exercice précédent ? Le code donné dans la question 3 satisfaisait-il cette condition ?

On utilise l'algorithme de compression suivant :

1. établir une table des fréquence d'apparition des valeurs dans le document (combien de fois chaque lettre apparaît),
2. on construit un arbre de la manière suivante : initialement on a n arbres à un seul nœud correspondant aux n lettres de l'alphabet et dont le poids est la fréquence de la lettre. Ensuite à chaque étape on choisi les deux arbres (u et v) de poids minimum (p_u et p_v) et on les remplace par l'arbre w de poids $p_w = p_u + p_v$ pour lequel la racine est un nouveau nœud, dont le fils gauche est u et le fils droit est v.
3. Le code associé à une lettre est alors le mot obtenu en descendant vers la feuille correspondant à cette lettre, en écrivant 0 si l'on va vers la gauche et 1 vers la droite.

Exercice 6: encoder le mot $BACADAEAFABBAAAGAH$ avec cet algorithme.

Exercice 7: On considère un alphabet à n lettres dont les fréquences sont $1, 2, 4, \cdots, 2^{n-1}$. Donner l'arbre de Huffman pour $n = 5$, $n = 8$. Dans le cas général, combien de bits sont nécessaires pour coder chaque lettre ? Combien de bits sont nécessaires pour coder le texte entier ?

2 Codage dynamique

Cette fois l'arbre de codage va dépendre du texte déjà lu et évoluer en fonction des symboles qu'on rencontre. Pour celà, 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. les poids des nœuds $p(c_1)$, $p(c_2)$,... forment une suite croissante,
2. pour tout i dans $[1; n]$, les nœuds $p(c_{2i-1})$ et $p(c_{2i})$ sont frères.

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

- *On a déjà rencontré a.* Le code de a est alors égal au code donné par l'arbre $\mathcal{A}_t$. On met ensuite 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 2 jusqu'à la racine.
- *On n'a jamais rencontré a.* Le code de a correspond alors au parcours dans l'arbre de la racine à la feuille 0. On remplace ensuite cette feuille 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 8: Coder le mot *abracadabra* avec cet algorithme.

Exercice 9: Donner le principe de l'algorithme de décompression.

Exercice 10: Pourquoi cet algorithme nécessite-t-il de supposer que le codage de chaque symbole du texte de départ occupe une taille fixe connue ?