Algorithmique Avancée - 2010/2011
Devoir de programmation : Huffman Dynamique
**Devoir à faire en binôme**
**Soutenance en TME semaine du 6 décembre ; rapport à rendre le 13 décembre.**

## 1. Présentation

La méthode de Huffman compresse un texte de façon optimale en codant les caractères les plus fréquents par les codes les plus courts.L'algorithme de Huffman dynamique construit un arbre de Huffman adaptatif qui évolue au fur et à mesure de la lecture et du traitement (compression ou décompression) des symboles. L'algorithme de compression (compresseur) et l'algorithme de décompression (décompresseur) modifient l'arbre de la même façon, de telle sorte qu' à chaque instant ils utilisent les mêmes codes (mais ces codes changent au cours du temps).

Initialement le compresseur démarre avec un arbre vide : aucun symbole n'a encore reçu de code. Le premier symbole lu est transmis avec son codage initial (ASCII, ou codage sur 5 bits si le texte ne contient que les 26 lettres de l'alphabet). Ce symbole est alors ajouté à l'arbre et reçoit ainsi un code (de longueur variable). La prochaine fois que ce symbole sera rencontré, le compresseur transmettra son code actuel et incrémentera sa fréquence. Il se peut alors que l'arbre ne soit plus un arbre de Huffman adaptatif (AHA) et si c'est le cas, le compresseur le modifie, ce qui implique une modification des codes.

Le décompresseur travaille de la même façon : lorsqu'il lit le code initial d'un symbole, il l'ajoute à l'arbre et lui affecte ainsi un code ; lorsqu'il lit un code de longueur variable, il utilise l'arbre courant pour déterminer le symbole correspondant, puis il modifie l'arbre de la même manière que le compresseur.

Il faut bien sûr que le décompresseur sache si le code qu'il lit est un codage initial ou un code de longueur variable : pour cela chaque codage initial est précédé d'un code spécial d'échappement, de longueur variable. Ainsi lorsque le décompresseur lit ce code d'échappement, il sait que les $k$ bits suivants ($k = 8$ pour l'ASCII, $k = 5$ pour un codage initial sur 5 bits) correspondent au codage initial d'un symbole qui n'est pas encore apparu dans le texte compressé lu jusque là.

La valeur du code spécial d'échappement évolue aussi au cours du traitement : pour ce faire, on ajoute à l'arbre une feuille spéciale #, de fréquence 0, dont la position dans l'arbre varie au cours du temps.

## 2. Modification de l'arbre

**Question 0**. Montrer les propriétés suivantes
**Propriété 1**. Soit $H$ un AHA avec $n$ feuilles et $n-1$ nœuds internes ; dans la numérotation hiérarchique GDBH *GaucheDroiteBasHaut* $x_1, x_2, \ldots, x_{2n-1}$ des nœuds, on a :

(1) $W(x_1) \leq W(x_2) \leq \ldots \leq W(x_{2n-1})$, où $W(x_i)$ est le poids du nœud $x_i$.

(2) Pour $1 \leq i \leq n-1$, les nœuds $x_{2i-1}$ et $x_{2i}$ sont frères (i.e. ont le même père dans l'arbre).

**Propriété 2**. Etant donné un AHA et une feuille $f$, de numéro $x_{i_0}$, dont le chemin à la racine est $\Gamma_f = x_{i_0}, x_{i_1}, \ldots, x_{i_k}$ ($i_k = 2n-1$), alors l'arbre résultant de l'incrémentation de $W(f)$ sera encore un AHA ssi $W(x_{i_j}) < W(x_{i_j+1})$, pour $0 \leq j \leq k-1$. On dira dans ce cas que tous les nœuds du chemin $\Gamma_f$ sont *incrémentables*.

L'algorithme suivant effectue la modification de l'AHA $H$ après lecture du symbole $s$. Il produit en résultat un AHA $H'$ dans lequel la fréquence de $s$ a été augmentée de 1, et les autres poids ont été modifiés en conséquence.
En plus des primitives classiques sur les arbres, on dispose d'une fonction `finBloc(H,m)` qui, étant donné un nœud $m$ numéroté $x_m$ dans un AHA $H$, renvoie le nœud $b$ tel que $W(x_m) = W(x_{m+1}), \ldots = W(x_b)$ et $W(x_b) < W(x_{b+1})$.

*Modification : AHA * Symbole → AHA*
**Fonction** `Modification` (H, s)
. **Si** $s$ n'est pas dans $H$
. **Soit** $Q$ le père de la feuille spéciale # dans $H$
. Remplacer # par un noeud interne de poids 1, dont
. le fils gauche est la feuille "#"
. le fils droit est une feuille "s", de poids 1
. **Sinon**
. **Soit** $Q$ la feuille correspondant à $s$ dans $H$
. **Si** $Q$ est frère de # et `pere`(Q)=`finBloc`(H,Q),
. **Alors** Q=`pere`(Q) **Fin Si**
. **Fin Si**
. **Retourne** `Traitement`(H,Q)
**Fin Fonction** `Modification`

*Traitement : AHA * noeud → AHA*
**Fonction** Traitement (H,Q)
. **Soit** $Q, Q_{i1}, ..., Q_{ik} = \Gamma_Q$ le chemin de $Q$ à la racine
. **Soit** $x_{i0}, x_{i1}, ..., x_{ik}$ la suite des numéros des nœuds de $\Gamma_Q$
. **Si** tous les nœuds de $\Gamma_Q$ sont incrémentables
. ajouter 1 à tous les poids sur le chemin $\Gamma_Q$
. **Retourne** $H$
. **Sinon**
. **Soit** $m$ le premier indice de $\Gamma_Q$ t.q. $W(x_m) = W(x_{m+1})$
. **Soit** $b =$ `finBloc(H,m)`
. ajouter 1 à tous les poids du chemin de $Q$ à $Q_m$
. changer dans $H$ les sous-arbres enracinés en $Q_m$ et $Q_b$
. **Retourne** `Traitement(H,pere(`$Q_m$`))`
**Fin Fonction** `Traitement`

**Question 1**. Montrer que l'algorithme n'échange jamais un nœud avec un de ses ancêtres.
**Question 2**. Quel est (en fonction de la taille de l'alphabet des symboles) le nombre maximal d'échanges réalisés lors d'une modification de l'arbre ?

## 3. Compression et Décompression

La compression se fait selon l'algorithme suivant.

**Procédure** Compression ($T$ :Texte)
**var** $H$ : Huffman, s : Symbole
. $H$=feuille spéciale #
. **Répéter**
. $s$= symbole suivant de $T$
. **Si** $s \in H$ **Alors**
. transmettre le code de $s$ dans $H$
. **Sinon**
. transmettre le code de # dans $H$ puis le code initial de $s$
. **Fin Si**
. $H$=Modification ($H$, $s$)
. **Jusqu'à** $T$ terminé
**FinProcédure** `Compression`

**Question 1**. Quelle est la complexité de cet algorithme, en temps et en place ?

**Question 2**. Pour $T = abracadabra$, avec le code initial sur 5 bits :
$a = 00000, b = 00001, c = 00010, d = 00011, \ldots, r = 10001, \ldots$
le texte compressé est 00000 000001 0010001 0 10000010 0 110000011 0 110 110 0 (les espaces servent à séparer les différentes transmissions).

Expliciter les différentes étapes de la production du texte compressé, en construisant au fur et à mesure l'arbre de Huffman.
**Question 3**. Ecrire l'algorithme de décompression. Vérifier qu'en suivant cet algorithme, la décompression de l'exemple précédent produit bien le texte *abracadabra*.

## 4. Implantation

Vous pouvez implanter les algorithmes dans le langage de votre choix (C, C++, Caml, Java, ...) et faire appel à des bibliothèques existantes pour la gestion des bits.
**Question 1**. Discuter en détails les structures de données nécessaires à l'implantation des algorithmes de compression et de décompression.
**Question 2**. Faire tourner les programmes de compression et décompression sur différents jeux de tests (qui vous seront communiqués sur www-master.ufr-info-p6.jussieu.fr/2010/algav) et analyser les résultats.

## 5. Cahier des charges

Le **rapport** (à rendre le 13 décembre) doit contenir les réponses aux questions posées dans l'énoncé, une justification des choix technologiques (langage, structures de donnée), la description de l'architecture du logiciel, un tableau de tests avec lien avec la complexité théoriques des courbes et analyse des résultats.

Lors de la **soutenance** (en TME semaine du 6 décembre), chaque binôme devra expliquer son travail en 10 minutes : présentation sur 2 ou 3 transparents et démonstration.