Algorithmique 2 2021-2022 Projet PreIng 2

La compression : Algorithme de Huffman

La compression de données est un aspect important en informatique.
Le but de ce projet est de comprendre et d'implémenter le comportement d'un algorithme par l'écriture un programme permettant de compresser des fichiers textes afin qu'ils occupent moins d'espace sur un disque. On demande également d'écrire un décompresseur qui devra restaurer le fichier original.
Il existe beaucoup de méthodes de compression. Celui proposé ici est l'algorithme de Huffman. Cet algorithme offre de bonnes performances et est utilisé dans de nombreux formats de compression (zip, jpg).

Algorithme de Huffman

La compression de Huffman est basée sur la remarque suivante : les caractères d'un fichier texte sont habituellement codés sur un octet, donc tous sur le même nombre de bits. Il serait plus économique, pour un fichier donné, de coder ses caractères sur un nombre variable de bits, en utilisant peu de bits pour les caractères fréquents et plus de bits pour les caractères rares. Notez que le codage choisi dépend donc du fichier à compresser.
Pour créer un code Huffman, il faut connaître la fréquence des caractères utilisés dans le fichier. Il faut donc lire tout le fichier avant de le compresser. En revanche, pour décompresser, il faut connaître la table de codage qui est jointe ou ajoutée devant le fichier compressé (d'ailleurs, cela diminue le taux de compression notamment pour les petits fichiers).
Pour illustrer le codage Huffman, on peut prendre un exemple.
Supposons un texte composé de 100 000 caractères a-f uniquement. Voici la fréquence de ces caractères, un encodage classique à 3 bits et un encodage Huffman à taille variable :

Caractères	a	b	c	d	e	f
Nbre d'occurrences (en milliers)	45	13	12	16	9	5
Mot de code (binaire)						
Taille fixe	000	001	010	011	100	101
Taille variable	0	101	100	111	1101	1100

Fig. 1 : Bilan (source : Introduction à l'algorithmique Cormen et ali.)

Avec l'encodage classique, le texte nécessite (45+13+12+16+9+5)*3 = 300 kbits.
Avec Huffman : 45*1 + 13*3 + 12*3 + 16*3 + 9*4 + 5*4 = 224 kbits.
Le codage de Huffman va attribuer un code plus court pour les caractères apparaissant le plus fréquemment. Le 'a', le plus fréquent, est codé sur 1 bit, tandis que les moins fréquents sont codés sur 4 bits.

Comment trouver le code Huffman ? L'algorithme construit de façon itérative un arbre de codage correspondant au code optimal, en regroupant à chaque itération les deux nœuds possédant le moins de fréquence dans le texte. Cet algorithme utilise donc les arbres binaires.

Afin de trouver le codage de chaque caractère, il faut d'abord compter, par caractère, leur fréquence d'apparition dans le texte (cf Fig. 1).

Pour chaque caractère ayant une fréquence d'apparition, on construit un nœud contenant le caractère et sa fréquence. On insère ces nœuds, de manière ordonnée croissante par rapport à la fréquence, dans une liste de nœuds (cf Fig. 2 (a)).

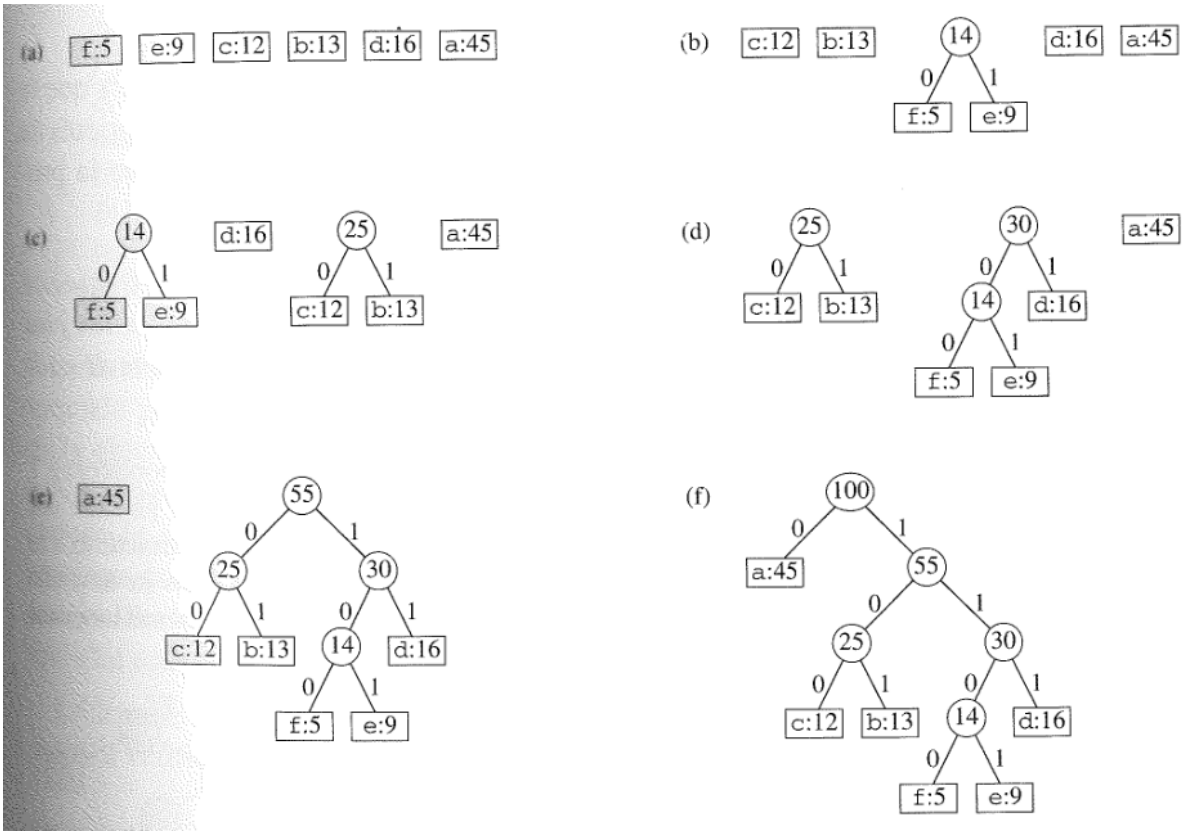

Fig. 2 : Étape de construction de l'arbre (source : Introduction à l'algorithmique Cormen et ali.)

On parcourt cette liste pour construire l'arbre, de la manière suivante :
On prend les deux premiers nœuds (en les enlevant de la liste), on les fusionne de sorte que leur nœud parent contient la somme de leurs fréquences, puis on l'insère dans la liste (Fig. 2 (b)).
On itère le processus (Fig. 2 (c), (d), (e)) jusqu'à la construction finale de l'arbre (Fig. 2 (f)).
Le code de chaque caractère est donné par le parcours de l'arbre, on note zéro lorsque l'on part à gauche, 1 lorsque l'on part à droite :

caractère	code
a	0
b	101
c	100
d	111
e	1101
f	1100

Ce codage ainsi obtenu est dit sans préfixe, de sorte que le code d'aucun caractère n'est préfixe d'un autre. Cela rend aisé le décodage.

L'algorithme génère un fichier : en en-tête, le nombre de caractères codés, les caractères et leur fréquence d'apparition ; puis, le contenu du fichier originel codé avec le nouveau code de chaque caractère.

La décompression lit l'en-tête, reconstruit l'arbre et reconstruit le texte originel par parcours successifs de l'arbre de codage selon la suite de caractères codés lus.

Cahier des charges :
Implémenter l'algorithme de Huffman pour coder des fichiers de texte et les décoder. Vous penserez à soigner le code (commenté, indenté, dépourvu de bugs, sur plusieurs fichiers...).
Si vous tenez à avoir une excellente note, vous pouvez ajouter à votre programme les améliorations indiquées ci-après (cf. « extensions possibles »).

Rendu
Vous rendrez une archive contenant les fichiers sources du code commenté ainsi qu'un fichier README contenant la ligne de compilation ainsi que les fonctionnalités d'exécution implémentées et un rapport expliquant votre démarche.
La date limite de rendu est le **vendredi 03 décembre 2021**
Vous réaliserez ce projet par groupe de **2-3 étudiants**.

Extensions possibles
Utiliser des options en ligne de commande :

1) L'option –c pour compresser, suivie du nom du fichier à compresser et du nom du fichier à créer. En plus de la suite de bits finale, l'arbre associé obtenu durant la phase de codage de Huffman est bien sûr nécessaire au décodage et doit être écrit dans (ou joint avec) le fichier compressé. Si la compression est réussie, le programme indique le temps d'exécution et le taux de compression. Le programme affiche un message d'erreur si le nombre de paramètres n'est pas correct.

Exemples :
> ./huffman monfichier.txt
Error: bad parameters.
Please use huffman –h for more information.

> ./huffman –c monfichier.txt monfichier.hfzip
Compressing monfichier.txt...
Done (2.43s).
3 658kb compressed to 3 119kb (85.2%).

2) L'option –d pour décompresser, suivie du nom du fichier à décompresser et le nom du fichier décompressé. Le programme affiche le temps d'exécution, ou un message d'erreur le cas échéant.

Exemple :
> ./huffman –d monfichier.hfzip monfichier.txt
Decompressing monfichier.hfzip...
Done (0.87s).

3) L'option –h qui affiche une aide d'utilisation.
4) Le fait d'enregistrer, non pas au format texte la compression, mais au format binaire.

RAPPEL : Le code de Huffman étant extrêmement connu, la copie de code trouvée sur Internet sera sanctionnée par 0, avec possibilité de conseil de discipline selon la gravité de la copie.