Début : 15 sept.
Fin : 10 oct.

2014

ASD 2 Labo1 : Encodage de Huffman

Paul Ntawuruhunga & Valentin Minder

But : Implémenter un programme permettant de compresser un document à l'aide de l'algorithme de Huffman, puis de le décompresser en récupérant le fichier original. Tester l'algorithme sur divers fichiers et discuter les résultats de compression.

Table des matières

1. **Remarques sur l'implémentation** **3**
2. **Tests sur divers fichiers.** **3**
3. **Résultats des tests** **4**
4. **Interprétation des résultats** **5**

1. Remarques sur l'implémentation

Les fonctions à implémenter étant déjà commentées et documentées, la plupart de notre implémentation est relativement évidente et triviale à comprendre. Au besoin, quelques commentaires supplémentaires dans le code permettent de s'y retrouver.

Toutefois, il y a un détail concernant la récursion dans la décompression. Cette récursion cherche à récupérer un caractère en fonction d'une chaine de bits. A chaque étape de la récursion, la tête de chaine est lue, supprimée puis on avance dans l'arbre avec le reste de la chaine. Toutefois, la chaine est fournie octet par octet par les classes utiles fournies, jusqu'à l'appel de l'exception BitStreamTropCourt qui relancera la lecture pour un octet supplémentaire. Dans cette étape, la récursion déjà effectuée, début de descente dans l'arbre et les morceaux de chaine déjà lus sont perdus. Pour contrer cela, nous attrapons (catch) l'exception lancée à chaque étape de la récursion, rajoutons le bit supprimé à cette étape en tête de chaine, puis relançons l'exception pour l'étape antérieur, jusqu'au code appelant des classes utiles, afin de lire plus loin et de relancer le décodage avec la chaine non utilisée sans perte de bit.

Cette solution a l'avantage de la simplicité sans modifications des classes fournies. Toutefois, ce n'est de loin pas optimal car il faut sans cesse enlever et rajouter des bits en tête de chaine, en particulier si les chaines correspondant à un symbole est souvent supérieur à 8 bits.

2. Tests sur divers fichiers.

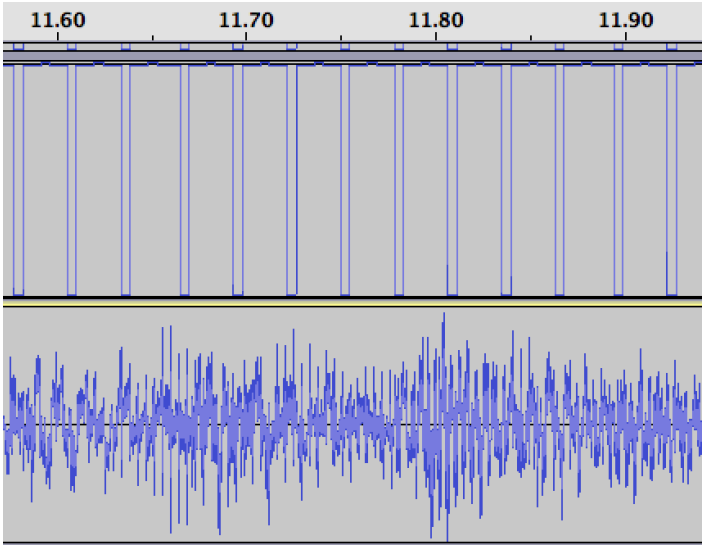
Nous avons testé la compression sur divers types de fichier. Tout d'abord, les deux fichiers en texte brut (.txt) fournis, l'un comportant une seule phrase, l'autre étant un très long texte (des conditions générales). Ensuite, nous nous sommes intéressés à des formats d'images. Pour cela, nous avons tout d'abord choisi un format sans compression (bitmap.bmp), et deux images, l'une étant unie (une seule couleur), l'autre étant très multicolore. Ensuite, nous avons compressé ces images en format jpg avec une qualité de 100% (sans fusion des couleurs très proches), afin de comparer la compression jpg face à celle implémentée dans ce labo. Nous avons également testé la compression sur ces fichiers jpg.

Figure 2 Image variée

Figure 1 Image unie

Nous avons également testé des formats de musique. Nous avons choisi deux extraits en format sans compression (wave form .wav), l'un étant un morceau de jazz, l'autre étant un bruit/beep constant sur 1 minute. Pour le beep, nous avons ensuite testé sur le même fichier encodé aux formats FLAC (Free Lossless Audio Codec – format de compression sans perete) et mp3 (MPEG Audio Layer 3 – compression avec pertes à 320kbps).

Figure 3 Comparaison (son beep et son jazzy)



3. Résultats des tests

Les résultats exhaustifs sont fournis dans le tableau annexe, comprenant la taille originale des fichiers, la taille compressée, le ratio de compression, le gain d'espace relatif, le temps de compression et de décompression. A noter que les tests ont tous été réalisé sur la même machine, même environnement (machine virtuelle Windows 7 avec Parrallel Desktop sur Mac OS X). En ce sens, le temps n'ont de sens que relativement aux autres fichiers.

Pour chaque fichier, nous avons utilisé la commande « diff fichierOriginal fichierDecompresse » pour vérifier que le fichier décompressé est strictement identique au fichier original. Ce test fut un succès pour l'ensemble des fichiers considérés.

4. Interprétation des résultats

Type	Nom du fichier	Taille [octets]		ratio	gain	temps [s]	
		original	compressés			compression	décompression
.txt	input_simple.txt	79	251	317.7%	-217.7%	0	0.015
	input_nddp.txt	1'039'351	589'499	56.7%	43.3%	8.877	10.701
.bmp	uni.bmp	86'454	24'777	28.7%	71.3%	0.358	0.266
	full.bmp	4'264'316	3'734'382	87.6%	12.4%	70.636	87.173
.jpg	uni.jpg	12'622	12'189	96.6%	3.4%	0.218	0.234
	full.jpg	1'284'700	1'283'637	99.9%	0.1%	22.807	31.34
.wav	jazzy.waw	3'115'242	3'033'249	97.4%	2.6%	55.738	75.894
	beep.waw	10'769'454	5'234'826	48.6%	51.4%	88.842	99.153
.flac	beep.flac	4'158'237	3'727'304	89.6%	10.4%	73.522	94.552
.mp3	beep.mp3	980'031	952'697	97.2%	2.8%	18.826	24.742

Vous trouverez nos fichiers de test dans le dossier > **Fichiers de test*

Les résultats sont sans surprise. Le fichier texte court ne gagne pas d'espace avec la compression, car le stockage de l'arbre en plus important que les données elles-mêmes. En revanche, le fichier texte long obtient une bonne compression, car le fichier est suffisamment long pour que les fréquences des lettres soient représentatives, et donc les lettres très fréquentes sont encodées de façon très courte.

Concernant les fichiers images, on obtient une bonne compression sur les fichiers bitmap car il s'agit d'un format non-compressé. Le fichier uni est une répétition de même information, sa compression est donc bien meilleure que celle du fichier multicolore. Il est à noter que notre compression est moins bonne que le format jpg de la même image, jpg étant un format spécialisé pour la compression des images. En revanche, jpg étant un format compressé, nous n'obtenons pas de compression significative par Hufmann sur ces fichiers.

Concernant les fichiers de musique, le morceau de jazz étant très varié, nous obtenons une compression insignifiante même sur le fichier non compressé wav. En revanche, pour le beep.wav, ce fichier étant constitué de répétitions, nous obtenons une très bonne compression. Le fichier flac est sensé être une compression sans perte optimale. Étonnamment, nous obtenons une compression supplémentaire de 10%, sans doute liée à la compression des index (un fichier flac est lisible depuis n'importe quelle position alors que le fichier .huf doit d'abord être décompressé). Concernant le mp3, ce format étant déjà compressé et spécialisé pour la musique, nous n'obtenons pas de résultats satisfaisants.