Projet de programmation fonctionnelle
Un compresseur Huffman

Juliusz Chroboczek
PPS
Université Paris-Diderot (Paris 7)

15 octobre 2014

Compression

$$250 \times 157 \times 4 = 157\,000$$

Compression

$250 \times 157 \times 4 = 157\,000$

```
$ ls -l spi.jpeg
-rw-r--- 1 jch jch 14766 Oct 15 18:31 spi.jpeg
```

Cette image est compressée.

Compression

La compression consiste à stocker des données en minimisant la quantité d'espace qu'elles occuppent. Deux types de compression :

- compression avec pertes, qui élimine les détails peu importants, adaptée aux photos (JPEG), à la musique (MPEG audio), aux films (MPEG vidéo).
- compression entropique, qui garde toutes les données (Zip, gzip, RAR, etc.).

Dans ce projet, nous nous intéressons à la compression entropique.

Compression entropique

Le dessein en est pris : je pars, cher Théramène,
Et quitte le séjour de l'aimable Trézène.
Dans le doute mortel dont je suis agité,
Je commence à rougir de mon oisiveté.

Deux inefficacités :

1. des séquences se répètent (« ène », « _le_ »)
   idée : coder la deuxième occurrence par un pointeur ;
2. chaque caractère est codé par au moins un octet :
   « e », apparaît 22 fois, et « z », une seule fois
   idée : moins de bits pour les caractères fréquents.

Dans ce projet, nous ne nous interesserons qu'aux codages de longueur variable (algorithme de Huffman).
Le codage des séquences répétées est hors sujet (algorithme de Lempel et Ziv, etc.).

Projet

Le but de ce projet est d'écrire un compresseur et un décompresseur utilisant l'algorithme de Huffman :

```
$ gzip -k huffman.tex
$ ./huffman huffman.tex*
$ ls -l huffman.*
-rw-r--r-- 1 jch jch 17145 Oct 15 17:30 huffman.tex
-rw-r--r-- 1 jch jch  6503 Oct 15 17:30 huffman.tex.gz
-rw-r--r-- 1 jch jch 10493 Oct 15 17:30 huffman.tex.hf
```

Le format de fichier vous est imposé, et l'interopérabilité est obligatoire.

```
$ ./huffman-etudiant huffman.tex
$ ./huffman -d huffman.tex.hf
```

Pour celà, je vous fournis un binaire fonctionnel.

Interopérabilité

L'interoperabilité est la capacité d'un programme à lire la sortie d'un autre :

- *OpenOffice* est interopérable avec *Microsoft Office* ;
- *Firefox* est interopérable *IIS* (le serveur *web* de Microsoft).

Dans ce projet, nous vous imposons le format de fichier et nous testerons l'interopérabilité de votre solution avec la mienne.

Si votre solution n'interopère pas avec la mienne, vous aurez un carton.

Codes de longueur variable

Code Morse :

Symbole	Code	Symbole	Code
A	• —	I	• •
B	— • • •	S	• • •
E	•	V	• • • —

Ce codage est ambigu :

• • • ⟶ S ou EEE ou IE ou EI ?

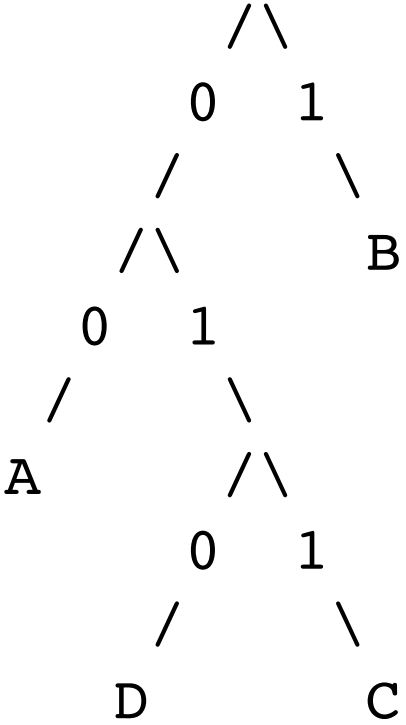
(Pas d'ambiguïté en pratique : l'opérateur humain fait une pause entre deux codes.

Codage préfixe et arbre

Un codage préfixe est un code où aucun code n'est préfixe d'un autre. Un codage préfixe n'est jamais ambigu.

Le code morse n'est pas un codage préfixe :
E (•) est préfixe de S (• • •).

Se donner un code préfixe est équivalent à se donner un arbre binaire :



Symbole	Code
A	00
B	1
C	011
D	010

Code préfixe optimal

Si on connaît les fréquences des différents symboles :

Symbole S	Fréquence f_s
A	143
B	73
C	83
...	...

alors le codage qui produit le plus petit fichier est le codage qui minimise

$$\sum_s f_s \cdot l_s$$

où l_s est la longueur du code correspondant à s. Un tel codage est dit optimal.

L’algorithme de Huffman

Étant donné un tableau des fréquences des caractères (un histogramme), l’algorithme de Huffman produit un codage préfixe optimal.

On aime tous l’algorithme de Huffman :

- optimal ;
- fait par un étudiant (le patron a conçu l’algorithme de Shannon et Fano, non-optimal) ;
- facile à expliquer, facile à implémenter.

Format de fichier

Le but de ce projet n'est pas seulement d'implémenter l'algorithme de Huffman — c'est de faire un compresseur et un décompresseur complets.

Format de fichier imposé :

- 4 octets magiques ;
- un octet de valeur n suivi de n octets ignorés ;
- la représentation de l'arbre ;
- la suite de codes compressés ;
- de 0 à 7 bits valant 0 ;
- un octet m suivi de m octets ignorés.

Si vous utilisez un autre format, vous aurez un carton.

Entrées sorties bit-à-bit

Sous Unix, un fichier est une suite d'octets. Votre compresseur produit une suite de bits :

$$\text{compresseur} : \text{suite d'octets} \longrightarrow \text{suite de bits}$$

$$\text{décompresseur} : \text{suite de bits} \longrightarrow \text{suite d'octets}$$

Entrées sorties bit-à-bit

Sous Unix, un fichier est une suite d'octets. Votre compresseur produit une suite de bits :

compresseur : suite d'octets $\longrightarrow$ suite de bits

décompresseur : suite de bits $\longrightarrow$ suite d'octets

Il faut donc simuler des suites de bits à l'aide de suites d'octets. Je vous fournis la bibliothèque `bitio.ml` :

```
let ch = Bitio.open_out_bit "toto" in
Bitio.output_bit 1; Bitio.output_bit 0; ...
Bitio.close_out_bit ch
```

Vous pouvez vous en servir.
(Ou pas, je ne suis pas susceptible.)

Binaires fournis

Ce projet, je l'ai fait. Deux fois :

- en C, il y a longtemps ;
- en CamI, le mois dernier.

Je vous fournis des binaires de mon implémentation :

- versions Linux/amd64 et Linux/i386, testées ;
- version Windows, jamais testée (sûrement fausse).

Servez vous-en pour vérifier l'interopérabilité.

Extensions

Le format de fichier est extensible :

- 4 octets magiques ;
- un octet de valeur n suivi de n octets ignorés ;
- la représentation de l'arbre ;
- la suite de codes compressés ;
- de 0 à 7 bits valant 0 ;
- un octet m suivi de m octets ignorés.

Ces octets ignorés peuvent servir à stocker des données auxquelles je n'ai pas pensé. Toute extension sera la bienvenue.

Résumé

- À rendre avant le 31 décembre 2014. Vous avez deux mois et demi (mais commencez un peu avant la Saint Sylvestre).
- Groupes de deux. 2. Pas trois. Deux.
- On veut des programmes qui marchent. Si votre programme ne fait rien, vous aurez un carton, même s'il y a plein de code.
- On veut des programmes interopérables. Si votre programme n'est pas interopérable, vous aurez un carton, même s'il marche.
- Il y a deux parties faciles ($\simeq$ TP 2) :
  - construction de l'histogramme ;
  - construction de l'arbre.