TD d'Algo n° 4

EPITA ING1 2013

Dans ce TD nous faisons de la compression de fichier, en changeant juste l'encodage de chaque caractère.

1 Histogramme et encodage de taille fixe

Pour simplifier, on suppose que tout le fichier à compresser est représenté par un tableau $data[0..n-1]$ de n caractères (sur 8 bits).

1. Proposez un algorithme calculant l'histogramme des caractères dans $data$. C'est-à-dire que $\text{HIST}(data, n)$ doit retourner un tableau de 256 entiers indiquant le nombre d'occurrence de chaque lettre.
2. Quelle est la complexité de HIST en fonction de n ?
3. Quelle est la complexité du calcul de p, le nombre de caractères différents qui apparaissent dans $data$.
4. Combien de bits sont réellement nécessaires pour distinguer p caractères différents ? Notons b cette valeur.
5. On décide que le fichier compressé que nous produisons commence par 256 bits indiquant pour chacun si le i^e caractère est utilisé, ou non. (Ces 256 bits permettent de retrouver p, b, et donc les encodages de chaque caractère.) Cet entête est suivi des n caractères, chacun sur b bits. Calculez le taux de compression en fonction de n et p. Quelle est la limite de ce taux quand $n \to \infty$?

2 Code préfixe

On considère maintenant un encodage de taille variable : chaque caractère peut être représenté par un nombre différent de bits. Il est évidement conseillé d'utiliser des codes courts pour les caractères fréquents, et de réserver les codes longs pour les caractères rares.

Il faut d'autre part qu'aucun code ne soit préfixe d'un autre. Par exemple si l'on avait choisi d'encoder "`a`" par 11 et "`b`" par 111, on ne saurait plus si 11111 désigne "`ab`" ou "`ba`".

1. Considérons le code suivant :

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

   Décodez `11011100110001001101`.
2. Proposez un algorithme qui prenne une telle table d'encodage en entrée, ainsi qu'une chaîne de n bits, et qui décode cette chaîne.
3. Combien existe-t-il d'encodages possibles pour p lettres ?
4. Supposons $p > 1$. Justifiez que si l'un des codes utilise p bits (ou plus), alors il existe forcément un meilleur codage.
5. On veut maintenant choisir un encodage optimal pour la compression. Supposons que le fichier à compresser contienne 10000 caractères choisis parmi `abcdefg` avec les fréquences suivantes.

lettre	a	b	c	d	e	f	g
fréquence	15%	42%	7%	11%	13%	9%	3%

   Proposer un encodage de ces lettres de façon à ce que le fichier encodé fasse 3062,5 octets (plusieurs encodages sont possibles). On ne peut pas avoir une taille plus petite sur cet exemple.

3 Huffman

L'algorithme de Huffman propose une façon de calculer un encodage optimal. L'encodage est représenté par un arbre dont les feuilles sont les lettres à encoder : le code d'une lettre se lit en suivant la branche de cette lettre à partir de la racine (fils gauche = 0, fils droit = 1). Chaque nœud possède un champ *freq* qui indique la fréquence des feuilles du sous-arbres (ou bien leur nombre d'occurrences c'est proportionnel).

Lignes 1–5 l'algo commence par construire une forêt F dans laquelle il "plante" un arbre (constitué d'une feuille) pour chaque caractère qui apparaît dans le fichier à compresser.

Les deux arbres dont les racines ont les plus faible fréquences sont ensuite combinés, et cela est répété lignes 6–11 jusqu'à ce qu'il ne reste plus qu'un arbre dans la forêt.

```
HUFFMAN(letters, freqs, p)
 1  for i ← 0 to p − 1 do
 2      z ←NEWLEAF(letters[i])
 3      z.freq ← freqs[i]
 4      INSERT(F, z)
 5  for i ← 1 to p − 1 do
 6      z ←NEWNODE()
 7      z.left ←EXTRACTMIN(F)
 8      z.right ←EXTRACTMIN(F)
 9      z.freq ← z.left.freq + z.right.freq
10      INSERT(F, z)
11  return EXTRACTMIN(F)
```

1. Déroulez cet algorithme avec les fréquences de la question précédente.
2. Quelle structure de données devrait être utilisée pour représenter F afin que EXTRACTMIN et INSERT soient efficaces ?
3. En déduire la complexité de Huffman en fonction du nombre de caractères p.
4. Écrire une procédure récursive qui prend l'arbre retourné par Huffman, et affiche la liste des lettres et leur code associé (les lettres n'ont pas besoin d'être dans l'ordre alphabétique). Quelle est la complexité de cet algorithme ?
5. Pour pouvoir décompresser un fichier encodé avec un code variable, il faut évidement connaître la correspondance code-caractère. Proposez une façon de sauvegarder l'arbre de (dé)codage utilisant au plus $10p - 1$ bits.
6. Si l'histogramme est passé sous la forme d'une liste triée, montrez que la forêt F peut être représentée plus efficacement avec deux listes triées (une liste contenant les arbres réduits à des feuilles, et une liste contenant les arbres ayant des nœuds internes). Que devient la complexité de l'algorithme dans ce cas ?
7. À quoi ressemble le codage de Huffman pour n lettres dont les fréquences sont les n premières valeurs de la suite de Fibonacci ?