lundi 19 décembre 2016
LUR, LRG

E304A : Laboratoire de concepts informatiques

Labo 5 : Compression de Huffman

Dans ce cinquième labo, vous allez implémenter un algorithme de compression sans perte qui exploite des structures de données vues au cours de conception orientée objet. Vous allez notamment manipuler des dictionnaires et des arbres.

Objectifs

- ☐ Comprendre le fonctionnement de la compression de Huffman
- ☐ Être capable d'implémenter un arbre binaire de manière récursive
- ☐ Être capable d'implémenter un algorithme qui utilise des structures de données

Codage de Huffman

Le principe du codage de Huffman consiste à associer un code à tous les symboles possibles, et coder un message revient à remplacer chaque symbole par son code. Voyons son fonctionnement pour compresser une chaine de caractères, en partant de l'exemple `BIENVENUE` :

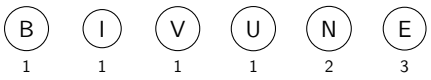
1. La première étape consiste à construire un tableau de fréquences d'apparition des différentes lettres de l'alphabet présentes dans le mot.

   Pour la chaine de caractères d'exemple, on obtient le tableau suivant, si on classe les fréquences par ordre croissant :

Symbole	B	I	V	U	N	E
Fréquence	1	1	1	1	2	3

2. La deuxième étape consiste à construire un arbre binaire, de manière incrémentale. On crée d'abord un arbre par symbole, en lui attachant sa fréquence.

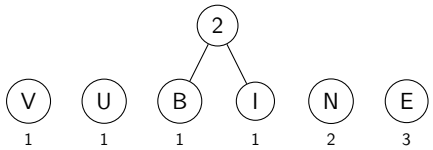
   Pour la chaine de caractères d'exemple, on se retrouve donc avec les six arbres suivants :

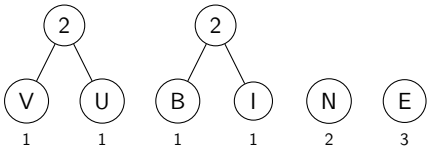
   Jusqu'à ce qu'il ne reste qu'un seul arbre, on va itérativement choisir deux arbres qui ont la plus petite fréquence associée et les fusionner en un nouvel arbre binaire prenant les deux arbres choisis en fils gauche et droit et ayant comme fréquence la somme de celles des arbres fusionnés.

   Pour la chaine de caractères d'exemple, l'algorithme se poursuit donc ainsi :

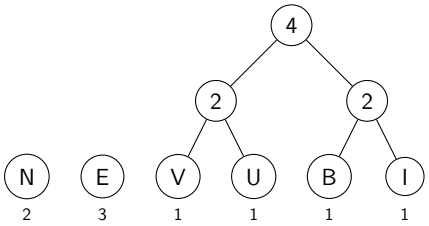
   (a)

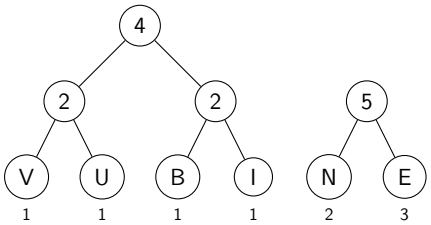
(b)



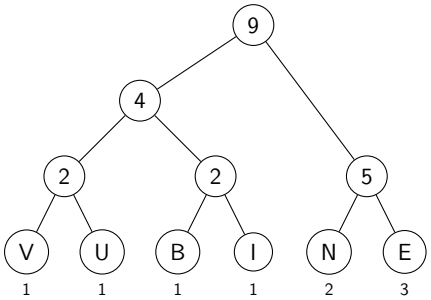
(c)



(d)



(e)



3. La troisième étape consiste à construire la table des mots code à partir de l'arbre construit à l'étape 2. Pour cela, on va effectuer des parcours dans l'arbre, partant de la racine vers les feuilles, et on considérant que prendre un fils gauche correspond à 0 et un fils droit à 1. Le mot code associé à chaque symbole est la séquence de 0 et de 1 rencontrés lors du parcours.
   Pour la chaine de caractères d'exemple, la table de mots code obtenue est la suivante :

Symbole	**Code**
V	000
U	001
B	010
I	011
N	10
E	11

4. Pour compresser une suite de symboles, il suffit simplement de remplacer chaque symbole par le mot code correspondant.

   Pour la chaine de caractères d'exemple, on obtient la suite de mots codes suivante :

   `010011111000011100011`1

   Si on avait stocké la chaine de caractères au format ASCII, cela aurait un total de neuf octets, chaque caractère occupant 8 bits de données. La forme compressée fait 22 bits, ce qu'on stockera donc sur trois octets. Le taux de compression est donc de 33,33...%.

5. Pour décompresser une suite de bits, il suffit de parcourir plusieurs fois l'arbre en suivant la chaine de bits. Chaque fois que l'on atteint une feuille de l'arbre, on note le symbole obtenu et on repart ensuite depuis la racine.

Travail à réaliser

Vous devez réaliser, dans n'importe quel langage, un programme qui va compresser une chaine de caractères en utilisant le codage de Huffman tel que présenté ci-dessus. Pour cela, vous devrez implémenter un arbre binaire, de manière récursive (voir cours).

Pour votre curiosité, on peut se poser quelques questions sur le codage de Huffman :

— Si on souhaite compresser des données, les envoyer à quelqu'un et s'assurer qu'il saura les décompresser, on devra également lui envoyer l'arbre ou au moins la table des fréquences. Cela occupera évidemment de la place dans le fichier et réduira le taux de compression obtenu. Pour un gros fichier, l'espace additionnel occupé est évidemment négligeable.

— Pour éviter de devoir la table de fréquences, on pourrait décider d'utiliser la même table de fréquences dans le monde entier (ou par exemple une par langue naturelle). Sauf pour des applications spécifiques, ce n'est généralement pas une bonne idée car une table spécifique sera plus efficace.

— Pour être certain que le résultat de la décompression soit unique, l'arbre construit par l'algorithme de Huffman possède une caractéristique appelée *suffix-free*. Cela signifie qu'aucun mot code n'est suffixe d'un autre mot code, c'est-à-dire que lorsqu'on atteint une feuille lors de la décompression, on est certain qu'il n'existe pas un autre chemin dans l'arbre qui prendrait un bit de plus.

— Enfin, on remarquera que pour avoir le meilleur taux de compression possible, les symboles les plus fréquents sont associés à des mots codes plus courts.