Compression de texte : algorithme de Huffman

F. Pascual

10 novembre 2018

Présentation

Compression de texte :

- le texte compressé doit prendre moins de place que le texte initial
- il n'y a pas de perte d'information
- le texte compressé peut être décompressé, sa décompression est unique, et on retrouve le texte initial.

Code de longueur variable

Avant compression, tous les caractères sont codés sur le même nombre de bits 0 ou 1.

Principe de la compression : on remplace chaque caractère par une suite de bits dont la taille dépend de la fréquence du caractère dans le texte :

- caractère fréquent $\rightarrow$ code court
- caractère rare $\rightarrow$ code long

Code de longueur variable, exemple

Avant compression, caractères codés sur 1 octet (8 bits).

Le mot `exercice` non compressé : 8 octets = 64 bits.

Code de longueur variable, exemple

Avant compression, caractères codés sur 1 octet (8 bits).

Le mot `exercice` non compressé : 8 octets = 64 bits.

On considère le codage :

Caractère	e	x	r	c	i
Code associé	00	010	011	10	11

“exercice” est codé : ”00010000111010111000”

Taille du codage après compression = 18 bits.

Décompression, exemple

On considère le codage :

Caractère	e	x	r	c	i
Code associé	00	010	011	10	11

Texte compressé : 000100001110111000

La décompression doit être unique :

$$\underbrace{00}_{e}\ \underbrace{010}_{x}\ \underbrace{00}_{e}\ \underbrace{011}_{r}\ \underbrace{10}_{c}\ \underbrace{11}_{i}\ \underbrace{10}_{c}\ \underbrace{00}_{e}$$

Texte décompressé : `exercice`

Code préfixe

Définition : Soit u et v deux mots de A^*, u est un préfixe de v s'il existe un mot h de A^* tel que $v = uh$.

bon est un préfixe de `bonjour`

Code préfixe

Définition : Soit u et v deux mots de A^*, u est un préfixe de v s'il existe un mot h de A^* tel que $v = uh$.

bon est un préfixe de `bonjour`

Définition : Un sous-ensemble C de A^* est un code préfixe si aucun mot de C n'est préfixe d'un autre mot de C.

$\{00, 010, 011, 10, 11\}$ est un code préfixe.

Code préfixe et décompression

Théorème : Soit C un code préfixe. Soit $u_1, u_2, \ldots, u_x$ et $v_1, v_2, \ldots, v_y$ des mots de C. Si $u_1 u_2 \ldots u_x = v_1 v_2 \ldots v_y$ alors $x = y$ et $u_i = v_i$ pour tout $i \in \{1, \ldots, x\}$.

Code préfixe et décompression

Théorème : Soit C un code préfixe. Soit $u_1, u_2, \ldots, u_x$ et $v_1, v_2, \ldots, v_y$ des mots de C. Si $u_1 u_2 \ldots u_x = v_1 v_2 \ldots v_y$ alors $x = y$ et $u_i = v_i$ pour tout $i \in \{1, \ldots, x\}$.

La décompression d'un texte compressé avec un code préfixe est unique et égale au texte initial.

Code préfixe et décompression

Théorème : Soit C un code préfixe. Soit $u_1, u_2, \ldots, u_x$ et $v_1, v_2, \ldots, v_y$ des mots de C. Si $u_1 u_2 \ldots u_x = v_1 v_2 \ldots v_y$ alors $x = y$ et $u_i = v_i$ pour tout $i \in \{1, \ldots, x\}$.

La décompression d'un texte compressé avec un code préfixe est unique et égale au texte initial.

Exemple :

Caractère	e	x	r	c	i
Code associé	00	010	011	10	11

00010000111011100 est la compression de exercice.
exercice est l'unique décompression de 00010000111011100.

Code préfixe et décompression

Théorème : Soit C un code préfixe. Soit $u_1, u_2, \ldots, u_x$ et $v_1, v_2, \ldots, v_y$ des mots de C. Si $u_1 u_2 \ldots u_x = v_1 v_2 \ldots v_y$ alors $x = y$ et $u_i = v_i$ pour tout $i \in \{1, \ldots, x\}$.

Code préfixe et décompression

Théorème : Soit C un code préfixe. Soit $u_1, u_2, \ldots, u_x$ et $v_1, v_2, \ldots, v_y$ des mots de C. Si $u_1 u_2 \ldots u_x = v_1 v_2 \ldots v_y$ alors $x = y$ et $u_i = v_i$ pour tout $i \in \{1, \ldots, x\}$.

La décompression d'un texte compressé avec un code préfixe est unique et égale au texte initial.

Codage de Huffman (1)

- Algorithme de compression mis au point en 1952 par David Albert Huffman

Codage de Huffman (1)

- Algorithme de compression mis au point en 1952 par David Albert Huffman
- Codage par symbole : chaque caractère du texte initial est remplacé par le symbole correspondant

Codage de Huffman (1)

- Algorithme de compression mis au point en 1952 par David Albert Huffman
- Codage par symbole : chaque caractère du texte initial est remplacé par le symbole correspondant
- Cet algorithme offre un taux de compression le meilleur possible pour un codage par symbole

Codage de Huffman (1)

- Algorithme de compression mis au point en 1952 par David Albert Huffman
- Codage par symbole : chaque caractère du texte initial est remplacé par le symbole correspondant
- Cet algorithme offre un taux de compression le meilleur possible pour un codage par symbole
- Il existe d'autres techniques de compression plus efficaces (codage arithmétique, Lempel-Ziv, etc.)

Codage de Huffman (2)

Données :

- Un texte à compresser sur un alphabet A.
- Une estimation de la fréquence des caractères de A dans le texte à compresser.

Codage de Huffman (2)

Données :

- Un texte à compresser sur un alphabet A.
- Une estimation de la fréquence des caractères de A dans le texte à compresser.

Principe du codage de Huffman :

- A partir de la fréquence des caractères dans le texte, on crée un arbre T, dit arbre de Huffman.
- Chaque caractère x est une feuille de cet arbre et plus x est fréquent, plus il est proche de la racine.

Codage de Huffman (2)

Données :

- Un texte à compresser sur un alphabet A.
- Une estimation de la fréquence des caractères de A dans le texte à compresser.

Principe du codage de Huffman :

- A partir de la fréquence des caractères dans le texte, on crée un arbre T, dit arbre de Huffman.
- Chaque caractère x est une feuille de cet arbre et plus x est fréquent, plus il est proche de la racine.
- A partir de cet arbre, on associe à chaque caractère un code de longueur égale à la profondeur de x dans T.

Codage de Huffman (2)

Données :

- Un texte à compresser sur un alphabet A.
- Une estimation de la fréquence des caractères de A dans le texte à compresser.

Principe du codage de Huffman :

- A partir de la fréquence des caractères dans le texte, on crée un arbre T, dit arbre de Huffman.
- Chaque caractère x est une feuille de cet arbre et plus x est fréquent, plus il est proche de la racine.
- A partir de cet arbre, on associe à chaque caractère un code de longueur égale à la profondeur de x dans T.

Compression : on remplace chaque lettre par le code correspondant.
Décompression : à partir de T et du texte compressé, on retrouve le texte initial.

Construction de l'arbre de Huffman

Soit $f(x)$ la fréquence du caractère x dans le texte à compresser.

Les feuilles de l'arbre sont les caractères, les noeuds internes contiennent le "poids" de leur sous-arbre.

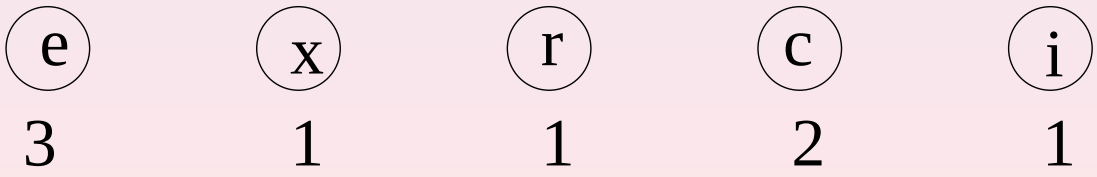
Algorithme :

- Pour chaque caractère $x \rightarrow$ feuille de poids $f(x)$
- Tant qu'il existe plusieurs arbres :
  - On choisit deux arbres T_1 et T_2 de poids minimal
  - On créé un arbre de racine poids(T_1)+poids(T_2) et ayant comme sous-arbres T_1 et T_2

Construction de l'arbre de Huffman, exemple

Mot à compresser : `exercice`

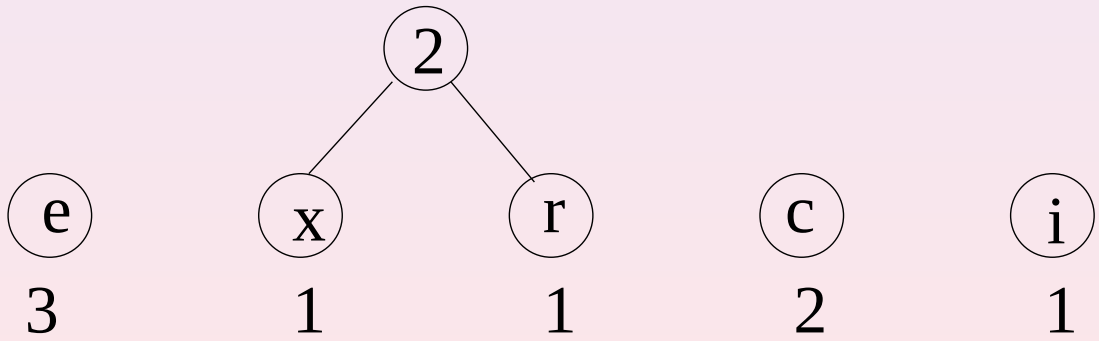
Caractère	e	x	r	c	i
Fréquence	3	1	1	2	1



Construction de l'arbre de Huffman, exemple

Mot à compresser : `exercice`

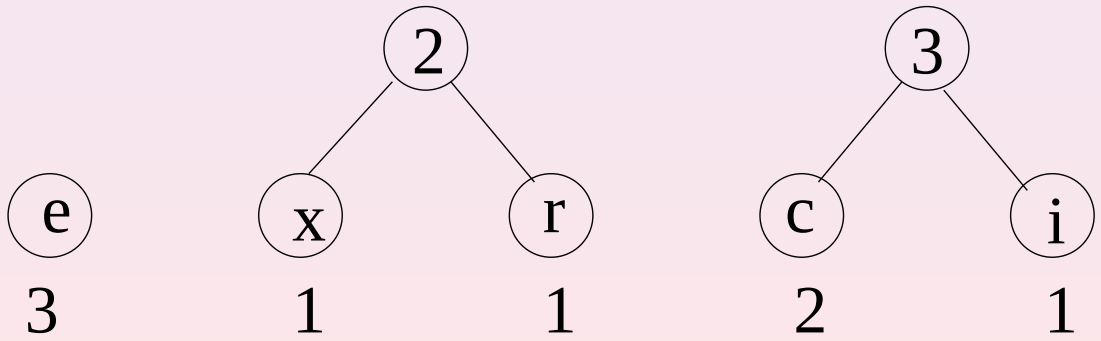
Caractère	e	x	r	c	i
Fréquence	3	1	1	2	1



Construction de l'arbre de Huffman, exemple

Mot à compresser : `exercice`

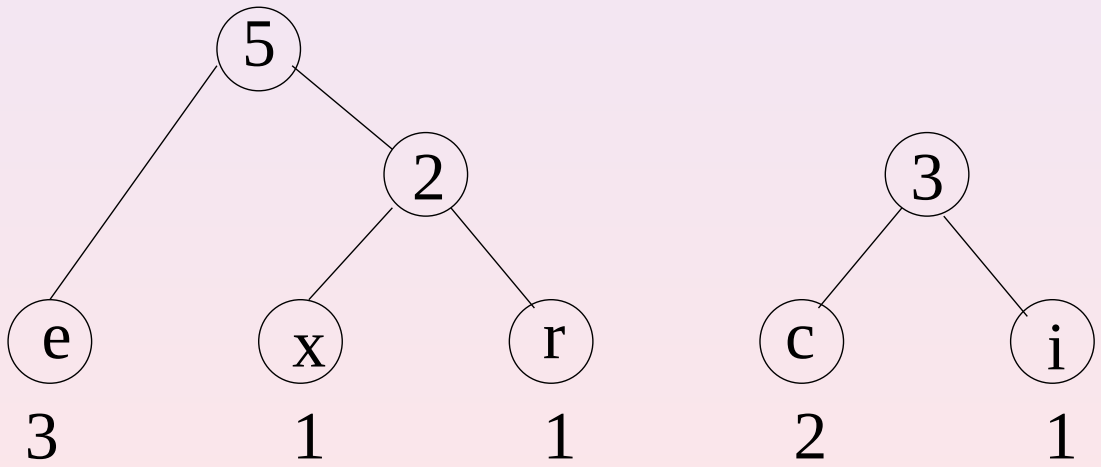
Caractère	e	x	r	c	i
Fréquence	3	1	1	2	1



Construction de l'arbre de Huffman, exemple

Mot à compresser : `exercice`

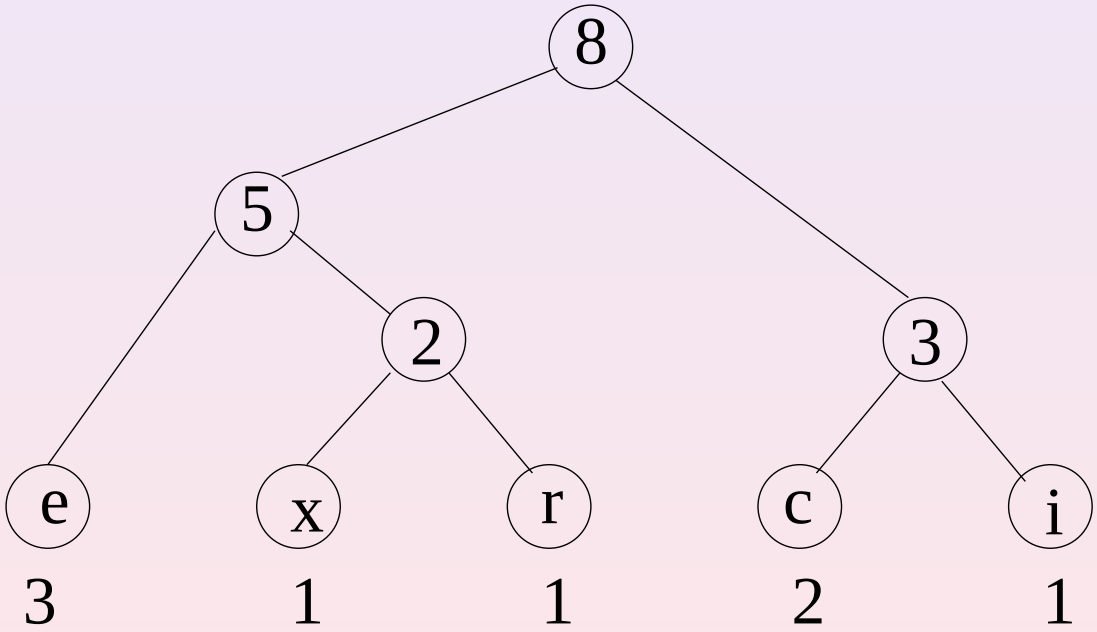
Caractère	e	x	r	c	i
Fréquence	3	1	1	2	1



Construction de l'arbre de Huffman, exemple

Mot à compresser : `exercice`

Caractère	e	x	r	c	i
Fréquence	3	1	1	2	1



Codage de Huffman

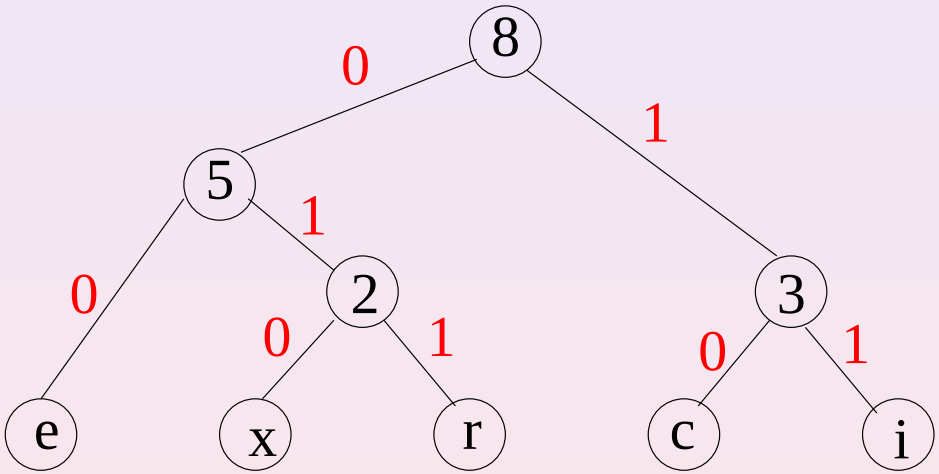
Le codage d'un caractère est obtenu en suivant le chemin de la racine au caractère dans l'arbre :

- 0 pour une descente à gauche
- 1 pour une descente à droite

Codage de Huffman

Le codage d'un caractère est obtenu en suivant le chemin de la racine au caractère dans l'arbre :

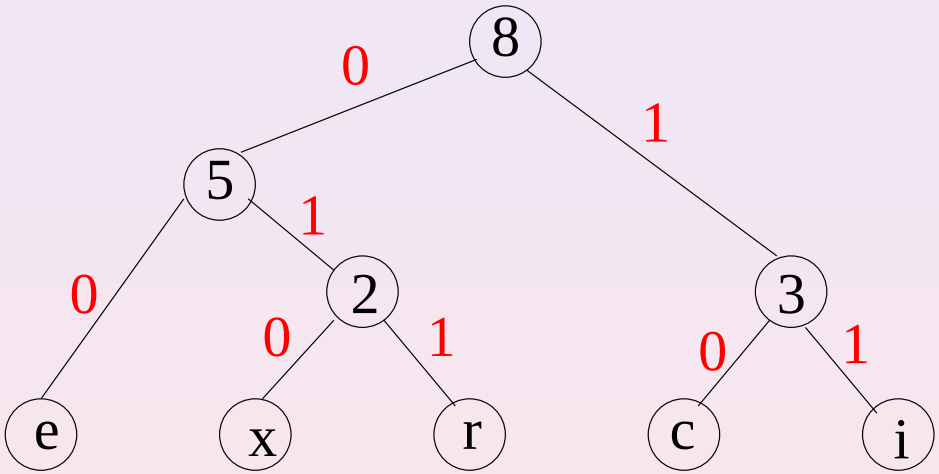
- 0 pour une descente à gauche
- 1 pour une descente à droite



Codage de Huffman

Le codage d'un caractère est obtenu en suivant le chemin de la racine au caractère dans l'arbre :

- 0 pour une descente à gauche
- 1 pour une descente à droite



Caractère	e	x	r	c	i
Code associé	00	010	011	10	11

Arbre de Huffman et codage préfixe

Théorème : L'ensemble des codages des caractères obtenus par construction d'un arbre de Huffman est un code préfixe.

Arbre de Huffman et codage préfixe

Théorème : L'ensemble des codages des caractères obtenus par construction d'un arbre de Huffman est un code préfixe.

La décomposition d'un texte compressé avec un codage de Huffman est donc unique et égale au texte initial.

Complexité de la construction de l'arbre de Huffman

Rappel de l'algorithme :

- Pour chaque caractère x créer une feuille de poids $f(x)$
- Tant qu'il existe plusieurs arbres :
  - Choisir deux arbres T_1 et T_2 de poids minimal
  - Créer un arbre de racine poids(T_1)+poids(T_2) et ayant comme sous-arbres T_1 et T_2

Complexité de la construction de l'arbre de Huffman

Rappel de l'algorithme :

- Pour chaque caractère x créer une feuille de poids $f(x)$
- Tant qu'il existe plusieurs arbres :
  - Choisir deux arbres T_1 et T_2 de poids minimal
  - Créer un arbre de racine poids(T_1)+poids(T_2) et ayant comme sous-arbres T_1 et T_2

Soit x le nombre de caractères.

Si on gère les arbres intermédiaires avec un tas : complexité en $O(x \log x)$.

Arbre optimal

Soit T un arbre ayant des feuilles de poids $w_i, \ldots, w_n$. Soit l_i la longueur du chemin de la racine au noeud w_i. La longueur pondérée $L(T)$ de l'arbre T est

$$L(T) = \sum_{i \text{ feuille de } T} l_i w_i$$

Arbre optimal

Soit T un arbre ayant des feuilles de poids $w_i, \ldots, w_n$. Soit l_i la longueur du chemin de la racine au noeud w_i. La longueur pondérée $L(T)$ de l'arbre T est

$$L(T) = \sum_{i \text{ feuille de } T} l_i w_i$$

$L(T)$ est la longueur du message après compression.

Arbre optimal

Soit T un arbre ayant des feuilles de poids $w_i, \ldots, w_n$. Soit l_i la longueur du chemin de la racine au noeud w_i. La longueur pondérée $L(T)$ de l'arbre T est

$$L(T) = \sum_{i \text{ feuille de } T} l_i w_i$$

$L(T)$ est la longueur du message après compression.

Un arbre T est optimal si $L(T) \leq L(T')$ pour tout arbre T' ayant les mêmes feuilles.

arbre de Huffman optimal

Théorème : Un arbre de Huffman est optimal.

Preuve par induction sur le nombre de feuilles de l'arbre.

arbre de Huffman optimal

Théorème : Un arbre de Huffman est optimal.

Preuve par induction sur le nombre de feuilles de l'arbre.

Propriété de choix glouton : Soit x le caractère ayant la moins grande fréquence, et y le caractère ayant la seconde moins grande fréquence. Il existe un codage préfixe optimal tel que x et y ont le même père.

arbre de Huffman optimal

Théorème : Un arbre de Huffman est optimal.

Preuve par induction sur le nombre de feuilles de l'arbre.

Propriété de choix glouton : Soit x le caractère ayant la moins grande fréquence, et y le caractère ayant la seconde moins grande fréquence. Il existe un codage préfixe optimal tel que x et y ont le même père.

Propriété de sous-structure optimale : On considère l'alphabet $A' = A - \{x, y\} + z$, où z est une nouvelle lettre ayant la fréquence $f(z) = f(x) + f(y)$. Soit T' un arbre optimal pour l'alphabet A'. L'arbre T obtenu à partir de T' en remplaçant la feuille associée à z par un noeud interne ayant x et y comme feuilles, est un arbre optimal pour A.