

Algorithmique des structures arborescentes

Algorithmique et programmation fonctionnelle

L2 Info, Math-info, CMI ISI & OPTIM, 2022–23

Marc Zeitoun

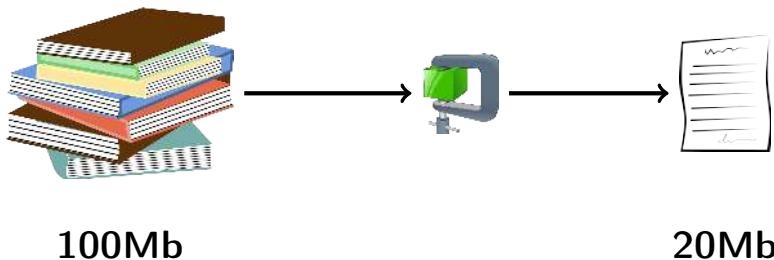
9 mars 2023

Plan

Compression de texte sans perte

Codage de Huffman

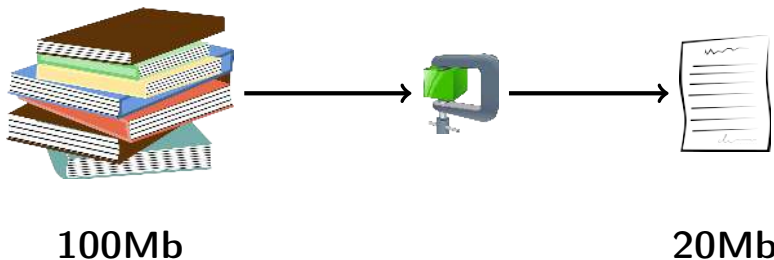
Compression sans perte



Objectifs :

- Réduire la taille des fichiers.

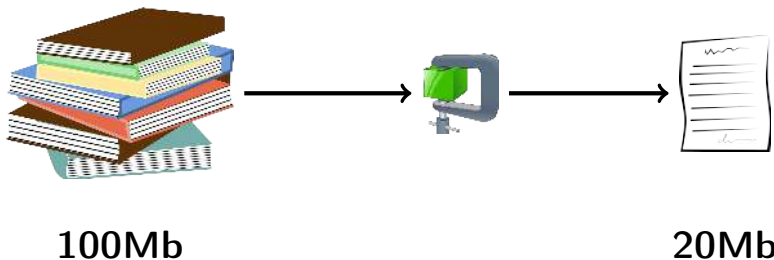
Compression sans perte



Objectifs :

- Réduire la taille des fichiers.
- **Sans perte** : on veut pouvoir reconstruire le fichier **original**.

Compression sans perte



Objectifs :

- Réduire la taille des fichiers.
- **Sans perte** : on veut pouvoir reconstruire le fichier **original**.
- Aujourd'hui : algorithme de Huffman.
Implémentation utilisant des arbres binaires complets.

Intérêt de la compression

Intérêt de la compression

- Gain d'espace sur disque → moins de disques nécessaires.

Intérêt de la compression

- Gain d'espace sur disque → moins de disques nécessaires.



Disque IBM de 5Mb en 1956
il en faut ~ 25000 pour faire 128Gb

Intérêt de la compression

- Gain d'**espace sur disque** → moins de disques nécessaires.



Disque IBM de 5Mb en 1956
il en faut ~ 25000 pour faire 128Gb

- Gain de **temps** en transfert (via le réseau).

Intérêt de la compression

- Gain d'**espace sur disque** → moins de disques nécessaires.



Disque IBM de 5Mb en 1956
il en faut ~ 25000 pour faire 128Gb

- Gain de **temps** en transfert (via le réseau).
- Gain d'**énergie**, diminution de l'empreinte carbone.

Codages de caractères

- Un fichier sur disque est une suite de caractères.

Codages de caractères

- Un fichier sur disque est une suite de caractères.
- Un codage associe à chaque caractère une **représentation** par un ou plusieurs **octets**.

Codages de caractères

- Un fichier sur disque est une suite de caractères.
- Un codage associe à chaque caractère une **représentation** par un ou plusieurs **octets**.
- Il y a plusieurs codages existants. Par exemple,
 - ASCII : représente 128 caractères chacun **sur un octet**.

Codages de caractères

- Un fichier sur disque est une suite de caractères.
- Un codage associe à chaque caractère une **représentation** par un ou plusieurs **octets**.
- Il y a plusieurs codages existants. Par exemple,
 - ASCII : représente 128 caractères chacun **sur un octet**.
 - ISO 8859-1 (latin1) l'étend à 191 caractères **sur un octet**.

Codages de caractères

- Un fichier sur disque est une suite de caractères.
- Un codage associe à chaque caractère une **représentation** par un ou plusieurs **octets**.
- Il y a plusieurs codages existants. Par exemple,
 - ASCII : représente 128 caractères chacun **sur un octet**.
 - ISO 8859-1 (latin1) l'étend à 191 caractères **sur un octet**.
 - UTF-8 pour les caractères du standard Unicode : 1 à 4 octets.
⇒ Plus difficile de décoder un fichier UTF-8.

Plan

Compression de texte sans perte

Codage de Huffman

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes **courts**.

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes **courts**.
- Caractères **peu fréquents** représentés par des codes **longs**.

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes **courts**.
- Caractères **peu fréquents** représentés par des codes **longs**.
- **Difficulté** : décodage, *i.e.*, découpage non ambigu.
- Par exemple, si on code le caractère **a** par **0** et **b** par **00**, on ne pourra pas décoder la suite **000** : elle peut représenter,

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes **courts**.
- Caractères **peu fréquents** représentés par des codes **longs**.
- **Difficulté** : décodage, *i.e.*, découpage non ambigu.
- Par exemple, si on code le caractère **a** par **0** et **b** par **00**, on ne pourra pas décoder la suite **000** : elle peut représenter,
 - ab, ou

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes **courts**.
- Caractères **peu fréquents** représentés par des codes **longs**.
- **Difficulté** : décodage, *i.e.*, découpage non ambigu.
- Par exemple, si on code le caractère **a** par **0** et **b** par **00**, on ne pourra pas décoder la suite **000** : elle peut représenter,
 - ab, ou
 - ba, ou

Compression de Huffman : principe

- **Idée** : coder les caractères sur un **nombre variable de bits**, éventuellement moins que 8.
- Caractères **fréquents** représentés par des codes **courts**.
- Caractères **peu fréquents** représentés par des codes **longs**.
- **Difficulté** : décodage, *i.e.*, découpage non ambigu.
- Par exemple, si on code le caractère **a** par **0** et **b** par **00**, on ne pourra pas décoder la suite **000** : elle peut représenter,
 - ab, ou
 - ba, ou
 - aaa.

Compression de Huffman : étapes

1. Calcul des fréquences de caractères dans le fichier à compresser.

Compression de Huffman : étapes

1. Calcul des fréquences de caractères dans le fichier à compresser.
2. Calcul du codage de chaque caractère, avec 2 contraintes :
 - caractères les plus fréquents représentés par des codes courts,

Compression de Huffman : étapes

1. Calcul des fréquences de caractères dans le fichier à compresser.
2. Calcul du codage de chaque caractère, avec 2 contraintes :
 - caractères les plus fréquents représentés par des codes courts,
 - possibilité de décoder.

Compression de Huffman : étapes

1. Calcul des fréquences de caractères dans le fichier à compresser.
2. Calcul du codage de chaque caractère, avec 2 contraintes :
 - caractères les plus fréquents représentés par des codes courts,
 - possibilité de décoder.
3. Dans le fichier compressé, écrire le codage utilisé.

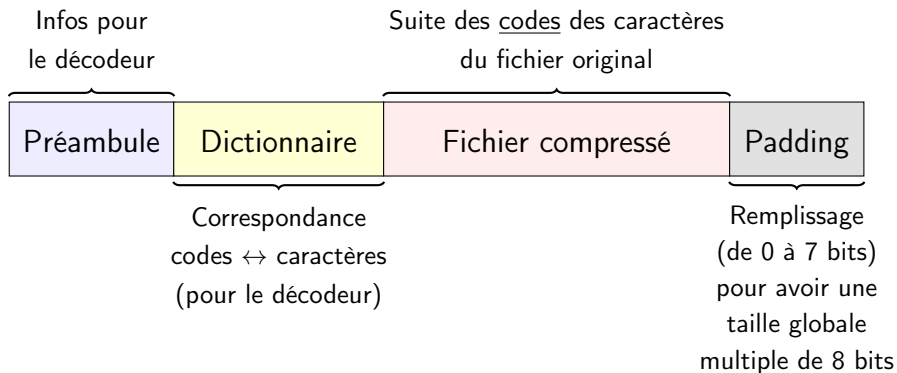
Compression de Huffman : étapes

1. Calcul des fréquences de caractères dans le fichier à compresser.
2. Calcul du codage de chaque caractère, avec 2 contraintes :
 - caractères les plus fréquents représentés par des codes courts,
 - possibilité de décoder.
3. Dans le fichier compressé, écrire le codage utilisé.
4. Enfin, relire le fichier original, et pour chacun de ses caractères, écrire son code dans le fichier compressé.

Compression de Huffman : étapes

1. Calcul des fréquences de caractères dans le fichier à compresser.
2. Calcul du codage de chaque caractère, avec 2 contraintes :
 - caractères les plus fréquents représentés par des codes courts,
 - possibilité de décoder.
3. Dans le fichier compressé, écrire le codage utilisé.
4. Enfin, relire le fichier original, et pour chacun de ses caractères, écrire son code dans le fichier compressé.
5. Un remplissage (padding) peut être nécessaire en fin de fichier pour ramener le nombre de bits écrits à un multiple de 8.

Format possible du fichier compressé



Comment décoder un texte : les codes préfixes

- Appelons **mot** une suite de **0** et de **1**. Par exemple **01001**.

Comment décoder un texte : les codes préfixes

- Appelons **mot** une suite de **0** et de **1**. Par exemple **01001**.
- Un mot x est **préfixe** d'un mot y si y commence par x .
Par exemple, **01** est préfixe de **01001**.

Comment décoder un texte : les codes préfixes

- Appelons **mot** une suite de **0** et de **1**. Par exemple **01001**.
- Un mot x est **préfixe** d'un mot y si y commence par x .
Par exemple, **01** est préfixe de **01001**.
- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- Exemples.
 - $P = \{000, 010, 011, 10, 11\}$ est un code préfixe.

Comment décoder un texte : les codes préfixes

- Appelons **mot** une suite de **0** et de **1**. Par exemple **01001**.
- Un mot x est **préfixe** d'un mot y si y commence par x .
Par exemple, **01** est préfixe de **01001**.
- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- Exemples.
 - $P = \{000, 010, 011, 10, 11\}$ est un code préfixe.
 - $Q = \{0, 00\}$ n'est **pas** un code préfixe car 0 est préfixe de 00.

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010110000110001000001011000
code le mot

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite **000**010110000110001000001011000
code le mot **A**

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000**0**1**0**110000110001000001011000
code le mot A B

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010**11**0000110001000001011000
code le mot A B R

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 00001011**000**0110001000001011000
code le mot A B R A

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 00001011000**0**110001000001011000
code le mot A B R A C

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 00001011000011**000**1000001011000
code le mot **A B R A C A**

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010111000011000**1**000001011000
code le mot A B R A C A D

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 0000101100001100010**000**01011000
code le mot A B R A C A D A

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 0000101100001100010000**010**11000
code le mot **A B R A C A D A B**

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 0000101100001100010000010**1**1000
code le mot **A B R A C A D A B R**

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010110000110001000001011**000**
code le mot **A B R A C A D A B R A**

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010110000110001000001011000
code le mot **A B R A C A D A B R A**
30 bits au lieu de $11 \times 8 = 88$ bits. **Peut-on faire mieux ?**

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010110000110001000001011000
code le mot **A B R A C A D A B R A**
30 bits au lieu de $11 \times 8 = 88$ bits. **Peut-on faire mieux ?**
 - **Oui** : 26 bits en échangeant les codes de A et de D.

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010110000110001000001011000
code le mot **A B R A C A D A B R A**
30 bits au lieu de $11 \times 8 = 88$ bits. **Peut-on faire mieux ?**
 - **Oui** : 26 bits en échangeant les codes de A et de D.
 - **Encore mieux** : 25 bits en codant A par 00 au lieu de 000.

Codes préfixes

- Un ensemble de mots P est un **code préfixe** si aucun mot de P n'est préfixe d'un autre mot de P .
- On peut utiliser un code préfixe pour **coder** :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R

- Un mot ne peut pas se décomposer de 2 façons différentes.
On décode en lisant la suite de 0 et de 1 **de gauche à droite**.
- **Exemple** : la suite 000010110000110001000001011000
code le mot **A B R A C A D A B R A**
30 bits au lieu de $11 \times 8 = 88$ bits. **Peut-on faire mieux ?**
 - **Oui** : 26 bits en échangeant les codes de A et de D.
 - **Encore mieux** : 25 bits en codant A par 00 au lieu de 000.
 - **Encore mieux** : 23 bits, voir plus loin (et 23 est optimal).

Codes préfixes et arbres binaires

- Un **code préfixe** se représente par un **arbre binaire**.
 - Chaque feuille correspond à un mot du code : on part de la racine jusqu'à la feuille, en écrivant **0** à chaque fois qu'on descend à gauche et **1** à chaque fois qu'on descend à droite.

Codes préfixes et arbres binaires

- Un **code préfixe** se représente par un **arbre binaire**.
 - Chaque feuille correspond à un mot du code : on part de la racine jusqu'à la feuille, en écrivant **0** à chaque fois qu'on descend à gauche et **1** à chaque fois qu'on descend à droite.
 - Chaque feuille correspond à un caractère du fichier à coder.

Codes préfixes et arbres binaires

- Un **code préfixe** se représente par un **arbre binaire**.
 - Chaque feuille correspond à un mot du code : on part de la racine jusqu'à la feuille, en écrivant **0** à chaque fois qu'on descend à gauche et **1** à chaque fois qu'on descend à droite.
 - Chaque feuille correspond à un caractère du fichier à coder.
 - Exemple :

$$P = \{000, 010, 011, 10, 11\}$$

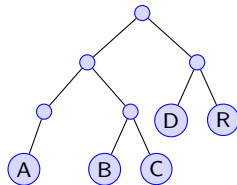
A B C D R

Codes préfixes et arbres binaires

- Un **code préfixe** se représente par un **arbre binaire**.
 - Chaque feuille correspond à un mot du code : on part de la racine jusqu'à la feuille, en écrivant **0** à chaque fois qu'on descend à gauche et **1** à chaque fois qu'on descend à droite.
 - Chaque feuille correspond à un caractère du fichier à coder.
 - Exemple :

$$P = \{000, 010, 011, 10, 11\}$$

A B C D R



Principe de la compression de Huffman

Principe

1. Calculer pour chaque caractère du fichier son code binaire.

Principe de la compression de Huffman

Principe

1. Calculer pour chaque caractère du fichier son code binaire.
2. Le fichier compressé est composé :

Principe de la compression de Huffman

Principe

1. Calculer pour chaque caractère du fichier son code binaire.
2. Le fichier compressé est composé :
 - du “dictionnaire” donnant le code de chaque caractère.

Principe de la compression de Huffman

Principe

1. Calculer pour chaque caractère du fichier son code binaire.
2. Le fichier compressé est composé :
 - du “dictionnaire” donnant le code de chaque caractère.
 - du fichier où chaque caractère est remplacé par son code.

Principe de la compression de Huffman

Principe

1. Calculer pour chaque caractère du fichier son code binaire.
2. Le fichier compressé est composé :
 - du “dictionnaire” donnant le code de chaque caractère.
 - du fichier où chaque caractère est remplacé par son code.

Deux difficultés techniques

- Séparer ces deux sections dans le fichier compressé.

Principe de la compression de Huffman

Principe

1. Calculer pour chaque caractère du fichier son code binaire.
2. Le fichier compressé est composé :
 - du “dictionnaire” donnant le code de chaque caractère.
 - du fichier où chaque caractère est remplacé par son code.

Deux difficultés techniques

- Séparer ces deux sections dans le fichier compressé.
- Avoir en tout un multiple de 8 bits.

Codage de Huffman : exemple

Création du “dictionnaire” pour **ABRACADABRA**.

- Tri selon fréquences → arbres avec poids : 

Codage de Huffman : exemple

Création du “dictionnaire” pour **ABRACADABRA**.

- Tri selon fréquences \rightarrow arbres avec poids : 

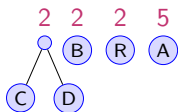
Remplacement des 2 arbres les plus légers par un unique arbre.
Nouveau poids = somme des anciens poids.

Codage de Huffman : exemple

Création du “dictionnaire” pour **ABRACADABRA**.

- Tri selon fréquences \rightarrow arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$

Remplacement des 2 arbres les plus légers par un unique arbre.
Nouveau poids = somme des anciens poids.



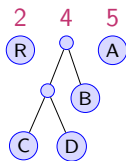
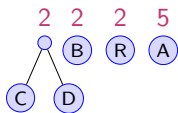
Codage de Huffman : exemple

Création du "dictionnaire" pour **ABRACADABRA**.

- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}}$ $\overset{1}{\text{D}}$ $\overset{2}{\text{B}}$ $\overset{2}{\text{R}}$ $\overset{5}{\text{A}}$

Remplacement des 2 arbres les plus légers par un unique arbre.

Nouveau poids = somme des anciens poids.

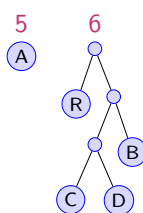
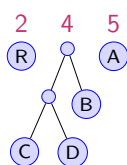
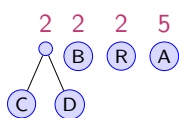


Codage de Huffman : exemple

Création du "dictionnaire" pour **ABRACADABRA**.

- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$

Remplacement des 2 arbres les plus légers par un unique arbre.
Nouveau poids = somme des anciens poids.



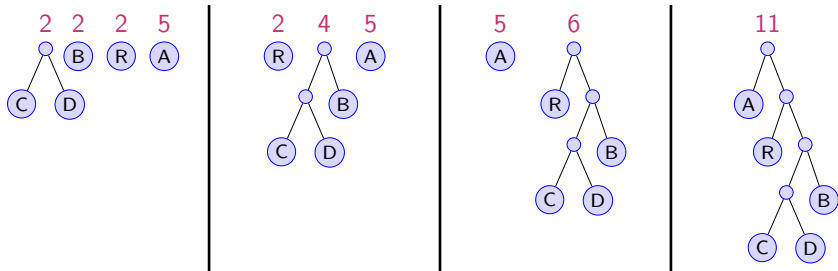
Codage de Huffman : exemple

Création du “dictionnaire” pour **ABRACADABRA**.

- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$

Remplacement des 2 arbres les plus légers par un unique arbre.

Nouveau poids = somme des anciens poids.

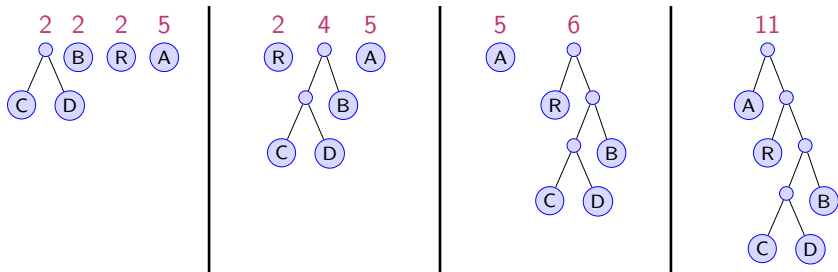


Codage de Huffman : exemple

Création du "dictionnaire" pour **ABRACADABRA**.

- Tri selon fréquences \rightarrow arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$

Remplacement des 2 arbres les plus légers par un unique arbre.
Nouveau poids = somme des anciens poids.



A : 0, R : 10, B : 111, C : 1100, D : 1101

Longueur du texte codé : $\underbrace{5 \times 1}_A + \underbrace{2 \times 2}_R + \underbrace{2 \times 1}_B + \underbrace{1 \times 4}_C + \underbrace{1 \times 4}_D = 23$.

Codage de Huffman : exemple

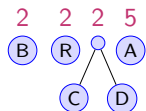
Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

- Tri selon fréquences → arbres avec poids : 

Codage de Huffman : exemple

Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

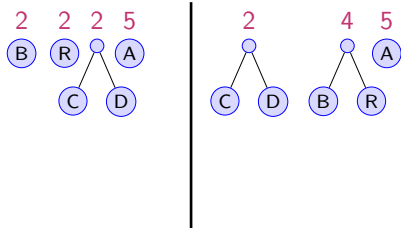
- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$



Codage de Huffman : exemple

Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

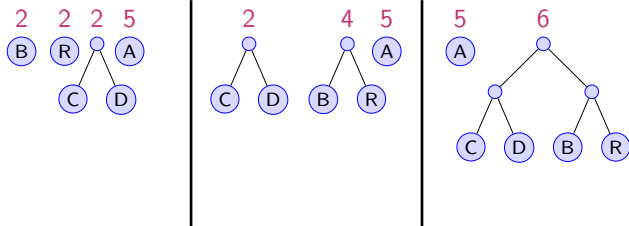
- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$



Codage de Huffman : exemple

Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

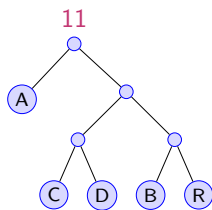
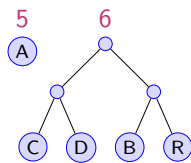
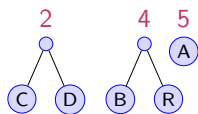
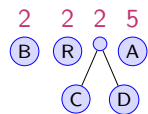
- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$



Codage de Huffman : exemple

Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

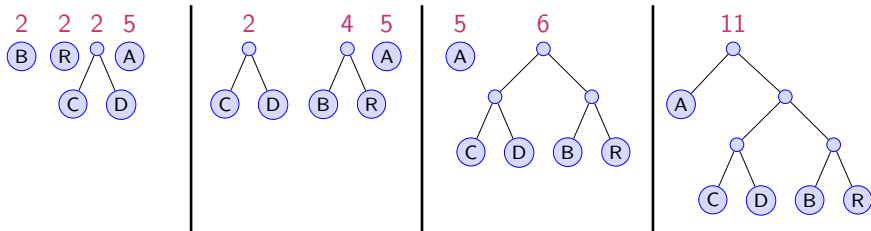
- Tri selon fréquences → arbres avec poids : $\overset{1}{\text{C}} \overset{1}{\text{D}} \overset{2}{\text{B}} \overset{2}{\text{R}} \overset{5}{\text{A}}$



Codage de Huffman : exemple

Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

- Tri selon fréquences \rightarrow arbres avec poids : $\overset{1}{C} \overset{1}{D} \overset{2}{B} \overset{2}{R} \overset{5}{A}$



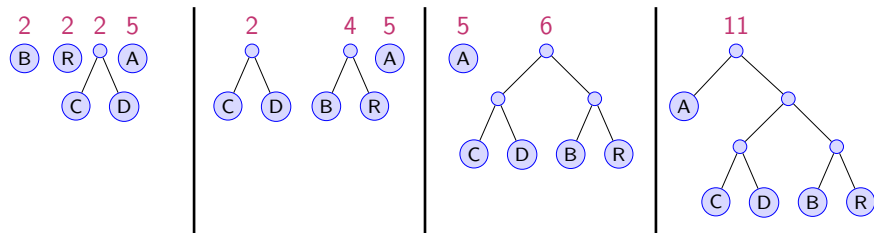
A : 0, R : 111, B : 110, C : 100, D : 101

Longueur du texte codé : $\underbrace{5 \times 1}_A + \underbrace{2 \times 3}_R + \underbrace{2 \times 3}_B + \underbrace{1 \times 3}_C + \underbrace{1 \times 3}_D = 23.$

Codage de Huffman : exemple

Dictionnaire pour **ABRACADABRA**, choix **alternatifs**.

- Tri selon fréquences → arbres avec poids : $\overset{1}{C} \overset{1}{D} \overset{2}{B} \overset{2}{R} \overset{5}{A}$



A : 0, R : 111, B : 110, C : 100, D : 101

Longueur du texte codé : $\underbrace{5 \times 1}_A + \underbrace{2 \times 3}_R + \underbrace{2 \times 3}_B + \underbrace{1 \times 3}_C + \underbrace{1 \times 3}_D = 23.$

On obtient la même longueur : 23. C'est en fait optimal !

Codage de Huffman : exercice

Coder le texte **MISSISSIPPI MAP**

Longueur du texte donné par un code

Dans la suite, on fixe :

- L'ensemble des lettres distinctes du texte original : $\{s_1, \dots, s_n\}$.

Longueur du texte donné par un code

Dans la suite, on fixe :

- L'ensemble des lettres distinctes du texte original : $\{s_1, \dots, s_n\}$.
- f_1, f_2, \dots, f_n : les fréquences correspondantes.

Longueur du texte donné par un code

Dans la suite, on fixe :

- L'ensemble des lettres distinctes du texte original : $\{s_1, \dots, s_n\}$.
- f_1, f_2, \dots, f_n : les fréquences correspondantes.

Soit T = arbre de code aux feuilles étiquetées sur $\{s_1, \dots, s_n\}$, avec :

Longueur du texte donné par un code

Dans la suite, on fixe :

- L'ensemble des lettres distinctes du texte original : $\{s_1, \dots, s_n\}$.
- f_1, f_2, \dots, f_n : les fréquences correspondantes.

Soit T = arbre de code aux feuilles étiquetées sur $\{s_1, \dots, s_n\}$, avec :

- $\ell_1, \ell_2, \dots, \ell_n$: les longueurs des codes de s_1, \dots, s_n .

Longueur du texte donné par un code

Dans la suite, on fixe :

- L'ensemble des lettres distinctes du texte original : $\{s_1, \dots, s_n\}$.
- f_1, f_2, \dots, f_n : les fréquences correspondantes.

Soit T = arbre de code aux feuilles étiquetées sur $\{s_1, \dots, s_n\}$, avec :

- $\ell_1, \ell_2, \dots, \ell_n$: les longueurs des codes de s_1, \dots, s_n .

Comment la longueur du texte codé selon T s'exprime-t-elle ?

Longueur du texte donné par un code

Dans la suite, on fixe :

- L'ensemble des lettres distinctes du texte original : $\{s_1, \dots, s_n\}$.
- f_1, f_2, \dots, f_n : les fréquences correspondantes.

Soit T = arbre de code aux feuilles étiquetées sur $\{s_1, \dots, s_n\}$, avec :

- $\ell_1, \ell_2, \dots, \ell_n$: les longueurs des codes de s_1, \dots, s_n .

Comment la longueur du texte codé selon T s'exprime-t-elle ?

$$L_T = f_1 \ell_1 + \dots + f_n \ell_n.$$

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Propriété intuitive. Échanger s_a et s_b dans T donne un meilleur code.

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Propriété intuitive. Échanger s_a et s_b dans T donne un meilleur code.

Preuve. Soit T' obtenu par échange des feuilles s_a et s_b dans T .

$$L_{T'} - L_T = f_a \ell_b + f_b \ell_a - (f_a \ell_a + f_b \ell_b).$$

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Propriété intuitive. Échanger s_a et s_b dans T donne un meilleur code.

Preuve. Soit T' obtenu par échange des feuilles s_a et s_b dans T .

$$\begin{aligned} L_{T'} - L_T &= f_a \ell_b + f_b \ell_a - (f_a \ell_a + f_b \ell_b). \\ &= (f_a - f_b) (\ell_b - \ell_a) \end{aligned}$$

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Propriété intuitive. Échanger s_a et s_b dans T donne un meilleur code.

Preuve. Soit T' obtenu par échange des feuilles s_a et s_b dans T .

$$\begin{aligned} L_{T'} - L_T &= f_a \ell_b + f_b \ell_a - (f_a \ell_a + f_b \ell_b). \\ &= \underbrace{(f_a - f_b)}_{\leq 0} \underbrace{(\ell_b - \ell_a)}_{\geq 0} \end{aligned}$$

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Propriété intuitive. Échanger s_a et s_b dans T donne un meilleur code.

Preuve. Soit T' obtenu par échange des feuilles s_a et s_b dans T .

$$\begin{aligned} L_{T'} - L_T &= f_a \ell_b + f_b \ell_a - (f_a \ell_a + f_b \ell_b). \\ &= \underbrace{(f_a - f_b)}_{\leq 0} \underbrace{(\ell_b - \ell_a)}_{\geq 0} \\ &\leq 0. \end{aligned}$$

Amélioration d'un code par échange de deux clés

Soit T un arbre donnant un code et s_a, s_b deux lettres telles que :

- $f_a \leq f_b$ (s_a est moins fréquent que s_b), et
- $\ell_a \leq \ell_b$ (le code de s_a est plus court que celui de s_b).

Propriété intuitive. Échanger s_a et s_b dans T donne un meilleur code.

Preuve. Soit T' obtenu par échange des feuilles s_a et s_b dans T .

$$\begin{aligned} L_{T'} - L_T &= f_a \ell_b + f_b \ell_a - (f_a \ell_a + f_b \ell_b). \\ &= \underbrace{(f_a - f_b)}_{\leq 0} \underbrace{(\ell_b - \ell_a)}_{\geq 0} \\ &\leq 0. \end{aligned}$$

Donc $L_{T'} \leq L_T$: T' donne effectivement un meilleur code que T .

Arbres optimaux

On dit que T est **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$ si L_T est de longueur **minimale** parmi tous les codages obtenus par des arbres.

Arbres optimaux

On dit que T est **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$ si L_T est de longueur **minimale** parmi tous les codages obtenus par des arbres.

Conséquence de la propriété d'amélioration par échange.

Il existe un arbre optimal dont les 2 premières lettres choisies par l'algorithme de Huffman ont le même père.

Arbres optimaux

On dit que T est **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$ si L_T est de longueur **minimale** parmi tous les codages obtenus par des arbres.

Conséquence de la propriété d'amélioration par échange.

Il existe un arbre optimal dont les 2 premières lettres choisies par l'algorithme de Huffman ont le même père.

En effet :

- Partons d'un arbre T optimal (*il en existe un, pourquoi ?*).

Arbres optimaux

On dit que T est **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$ si L_T est de longueur **minimale** parmi tous les codages obtenus par des arbres.

Conséquence de la propriété d'amélioration par échange.

Il existe un arbre optimal dont les 2 premières lettres choisies par l'algorithme de Huffman ont le même père.

En effet :

- Partons d'un arbre T optimal (*il en existe un, pourquoi ?*).
- Construisons T' à partir de T en échangeant les 2 lettres les moins fréquentes avec 2 feuilles « sœurs » les plus profondes.

Arbres optimaux

On dit que T est **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$ si L_T est de longueur **minimale** parmi tous les codages obtenus par des arbres.

Conséquence de la propriété d'amélioration par échange.

Il existe un arbre optimal dont les 2 premières lettres choisies par l'algorithme de Huffman ont le même père.

En effet :

- Partons d'un arbre T optimal (*il en existe un, pourquoi ?*).
- Construisons T' à partir de T en échangeant les 2 lettres les moins fréquentes avec 2 feuilles « sœurs » les plus profondes.
D'après la diapo précédente, on a : $L_{T'} \leq L_T$.

Arbres optimaux

On dit que T est **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$ si L_T est de longueur **minimale** parmi tous les codages obtenus par des arbres.

Conséquence de la propriété d'amélioration par échange.

Il existe un arbre optimal dont les 2 premières lettres choisies par l'algorithme de Huffman ont le même père.

En effet :

- Partons d'un arbre T optimal (*il en existe un, pourquoi ?*).
- Construisons T' à partir de T en échangeant les 2 lettres les moins fréquentes avec 2 feuilles « sœurs » les plus profondes.
D'après la diapo précédente, on a : $L_{T'} \leq L_T$.
- Donc T' est optimal lui aussi, et il a la propriété voulue.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Soit T un arbre **optimal** et H un arbre de Huffman pour un texte.

On veut montrer que :

$$L_H \leq L_T \quad (\text{donc en fait, } L_H = L_T).$$

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Soit T un arbre **optimal** et H un arbre de Huffman pour un texte.
On veut montrer que :

$$L_H \leq L_T \quad (\text{donc en fait, } L_H = L_T).$$

Preuve. Récurrence sur le nombre de lettres distinctes.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Soit T un arbre **optimal** et H un arbre de Huffman pour un texte.
On veut montrer que :

$$L_H \leq L_T \quad (\text{donc en fait, } L_H = L_T).$$

Preuve. Récurrence sur le nombre de lettres distinctes.

Cas de base : il y a une seule lettre $s_1 = a$ dans le texte. Alors :

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Soit T un arbre **optimal** et H un arbre de Huffman pour un texte.
On veut montrer que :

$$L_H \leq L_T \quad (\text{donc en fait, } L_H = L_T).$$

Preuve. Récurrence sur le nombre de lettres distinctes.

Cas de base : il y a une seule lettre $s_1 = a$ dans le texte. Alors :

- $T = \textcircled{a}$ puisque T est optimal.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Soit T un arbre **optimal** et H un arbre de Huffman pour un texte.
On veut montrer que :

$$L_H \leq L_T \quad (\text{donc en fait, } L_H = L_T).$$

Preuve. Récurrence sur le nombre de lettres distinctes.

Cas de base : il y a une seule lettre $s_1 = a$ dans le texte. Alors :

- $T = \textcircled{a}$ puisque T est optimal.
- $H = \textcircled{a}$ d'après l'algorithme de Huffman.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Soit T un arbre **optimal** et H un arbre de Huffman pour un texte.
On veut montrer que :

$$L_H \leq L_T \quad (\text{donc en fait, } L_H = L_T).$$

Preuve. Récurrence sur le nombre de lettres distinctes.

Cas de base : il y a une seule lettre $s_1 = a$ dans le texte. Alors :

- $T = \textcircled{a}$ puisque T est optimal.
- $H = \textcircled{a}$ d'après l'algorithme de Huffman.

Donc $T = H$, et en particulier, $L_T = L_H$.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Preuve, **hérédité** :

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Soit n lettres s_1, s_2, \dots, s_n de fréquences $f_1 \leq \dots \leq f_n$, et :

- H un arbre de Huffman pour $(s_1, f_1), \dots, (s_n, f_n)$.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Soit n lettres s_1, s_2, \dots, s_n de fréquences $f_1 \leq \dots \leq f_n$, et :

- H un arbre de Huffman pour $(s_1, f_1), \dots, (s_n, f_n)$.
- T un arbre **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.

Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Soit n lettres s_1, s_2, \dots, s_n de fréquences $f_1 \leq \dots \leq f_n$, et :

- H un arbre de Huffman pour $(s_1, f_1), \dots, (s_n, f_n)$.
- T un arbre **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$.

Dans H , s_1 et s_2 ont même père. On peut choisir T où c'est le cas.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.



Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Soit n lettres s_1, s_2, \dots, s_n de fréquences $f_1 \leq \dots \leq f_n$, et :

- H un arbre de Huffman pour $(s_1, f_1), \dots, (s_n, f_n)$.
- T un arbre **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$.

Dans H , s_1 et s_2 ont même père. On peut choisir T où c'est le cas.

Dans H et T , on remplace le sous-arbre  par un unique nœud , où s_0 est une nouvelle lettre. On obtient ainsi H' et T' .

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.



Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Soit n lettres s_1, s_2, \dots, s_n de fréquences $f_1 \leq \dots \leq f_n$, et :

- H un arbre de Huffman pour $(s_1, f_1), \dots, (s_n, f_n)$.
- T un arbre **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$.

Dans H , s_1 et s_2 ont même père. On peut choisir T où c'est le cas.

Dans H et T , on remplace le sous-arbre  par un unique nœud , où s_0 est une nouvelle lettre. On obtient ainsi H' et T' .

H' est un arbre de Huffman sur $(s_0, f_1 + f_2), (s_3, f_3), \dots, (s_n, f_n)$.
Par hypothèse de récurrence, H' est optimal : $L_{H'} \leq L_{T'}$.

Optimalité du codage de Huffman

Propriété. L'arbre de l'algorithme de Huffman est **optimal**.



Preuve, **hérédité** :

Supposons optimal tout arbre de Huffman sur $n-1$ lettres distinctes

Soit n lettres s_1, s_2, \dots, s_n de fréquences $f_1 \leq \dots \leq f_n$, et :

- H un arbre de Huffman pour $(s_1, f_1), \dots, (s_n, f_n)$.
- T un arbre **optimal** pour $(s_1, f_1), \dots, (s_n, f_n)$.

Dans H , s_1 et s_2 ont même père. On peut choisir T où c'est le cas.

Dans H et T , on remplace le sous-arbre  par un unique nœud , où s_0 est une nouvelle lettre. On obtient ainsi H' et T' .

H' est un arbre de Huffman sur $(s_0, f_1 + f_2), (s_3, f_3), \dots, (s_n, f_n)$.
Par hypothèse de récurrence, H' est optimal : $L_{H'} \leq L_{T'}$.

On obtient alors : $L_H = L_{H'} + f_1 + f_2 \leq L_{T'} + f_1 + f_2 = L_T$. ✓