Coder l'information

Informatique pour tou(te)s - semaines 46 & 47

Guillaume CONNAN

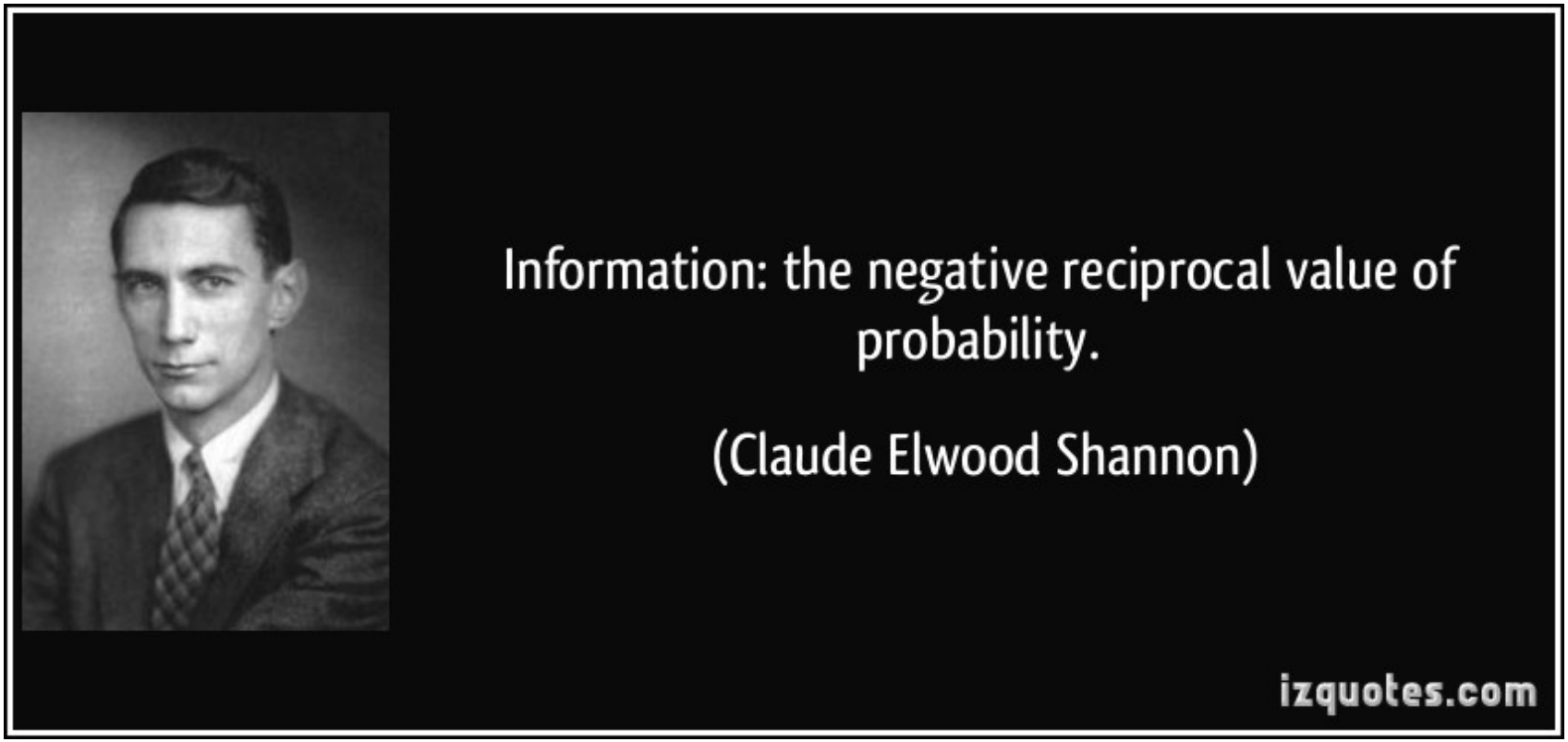
Lycée Clemenceau - MP / MP*

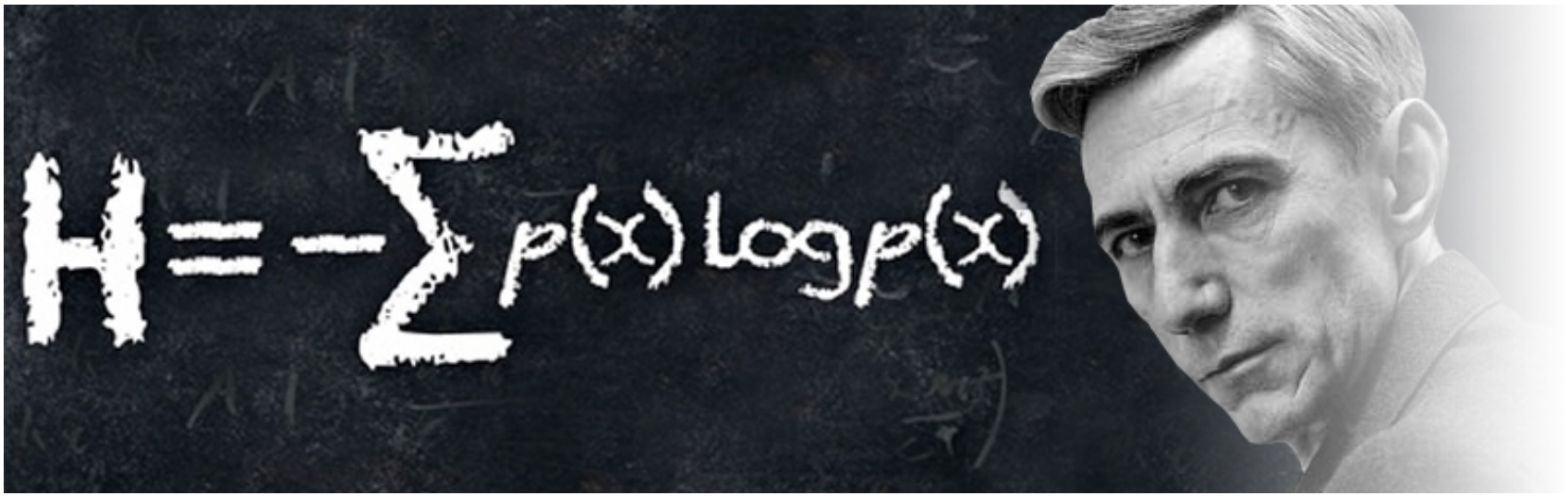
novembre 2015

UNIVERSITÉ DE NANTES

Sommaire

Sommaire


Information: the negative reciprocal value of probability.
(Claude Elwood Shannon)
izquotes.com


H=-∑p(x) log p(x)

- Claude Shannon

- Claude Shannon
- 1916 - 2001

- Claude Shannon
- 1916 - 2001
- MIT, Bell Labs

- Claude Shannon
- 1916 - 2001
- MIT, Bell Labs
- Mathématiques, Électricité

- Claude Shannon
- 1916 - 2001
- MIT, Bell Labs
- Mathématiques, Électricité
- Master : liens entre algèbre de Bool et interrupteurs

- Claude Shannon
- 1916 - 2001
- MIT, Bell Labs
- Mathématiques, Électricité
- Master : liens entre algèbre de Bool et interrupteurs
- Théorie de l'information

N°1 NOUVEAU ✦ 3F50 PRIX DE LANCEMENT ✦ CHAQUE MERCREDI

INFOS DU MONDE

Hebdomadaire d'information • du 19 au 26 mars 1984 • N°1

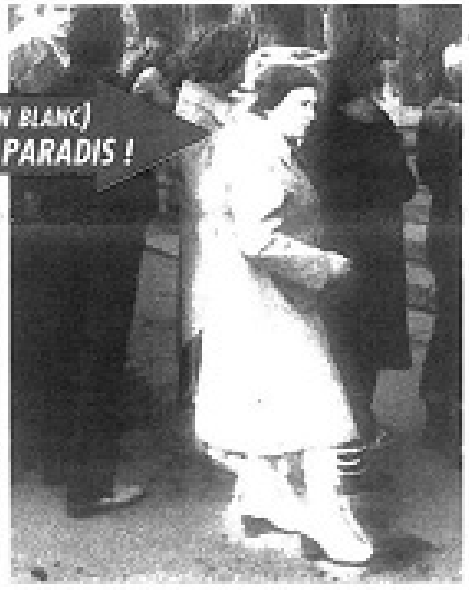
Le boucher anglais : il a aussi tué 12 Françaises ! Page 3

Où irez-vous ?

ENFER OU PARADIS ?

UN APPAREIL VOUS LE REVELE !

Découverte scientifique capitale, l'appareil photo mis au point par le professeur Herbert Von Krish permet de savoir si vous êtes destiné à l'enfer ou au paradis. PAGE 11



ELISE (EN BLANC) IRA AU PARADIS !

TÉMOIGNAGE

ELLE N'A PAS DORMI DEPUIS 32 ANS !

Marie-Claude, 39 ans, est atteinte par une maladie inconnue ! Page 5

ils font empailler leur patron mort ! Page 8

L'HOMME QUI N'A QUE DES POUCES !

Un étonnant handicap Page 6

UNIVERSITÉ DE NANTES

- fonction décroissante de la probabilité d'un évènement

- fonction décroissante de la probabilité d'un évènement
- La quantité d'information de deux évènements indépendants est la somme de leurs quantités d'information

- fonction décroissante de la probabilité d'un évènement
- La quantité d'information de deux évènements indépendants est la somme de leurs quantités d'information
- $I(B \cap C) = I(B) + I(C)$

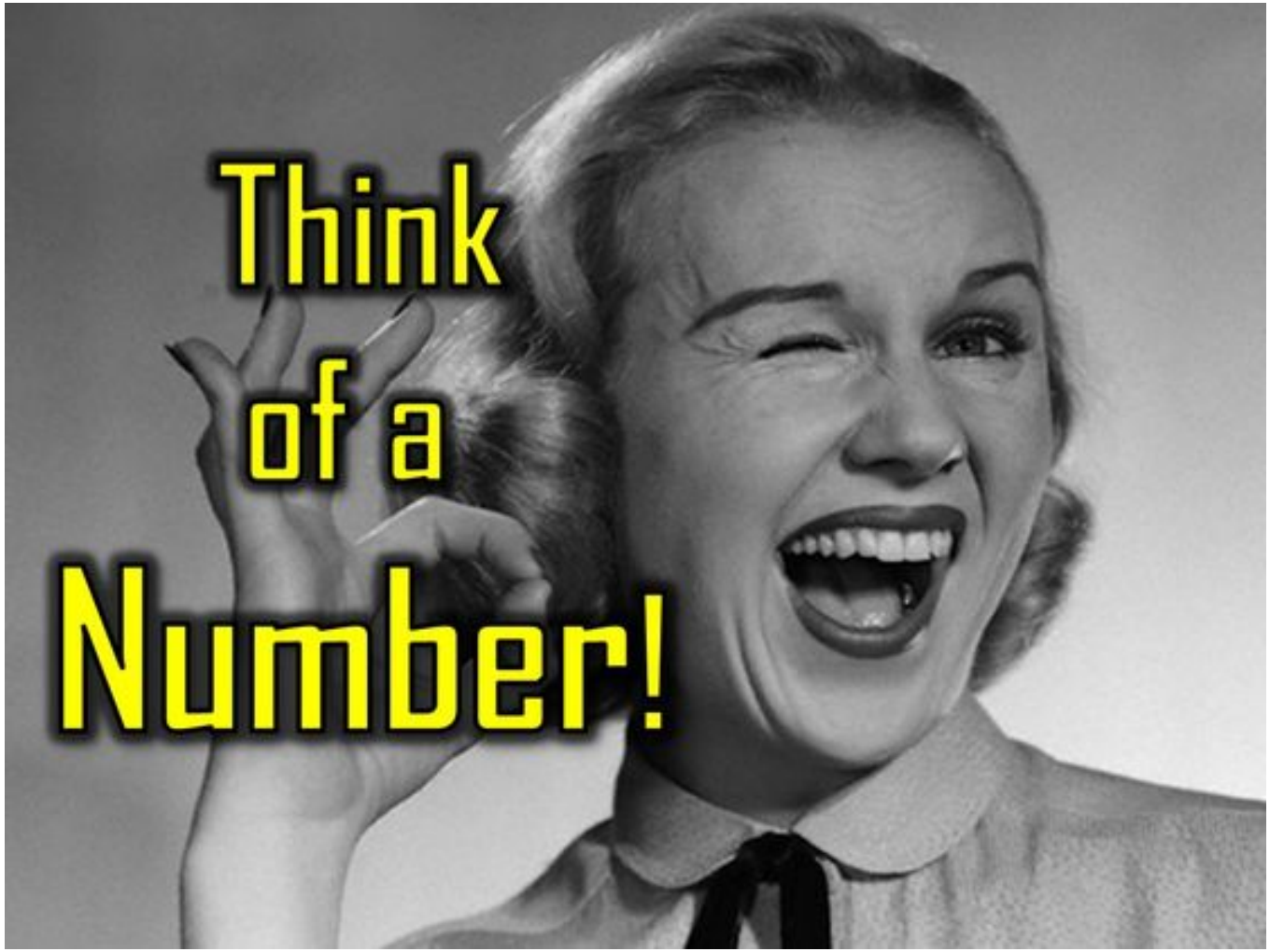
- fonction décroissante de la probabilité d'un évènement
- La quantité d'information de deux évènements indépendants est la somme de leurs quantités d'information
- $I(B \cap C) = I(B) + I(C)$
- $\varphi(b \cdot c) = \varphi(b) + \varphi(c)$

- fonction décroissante de la probabilité d'un évènement
- La quantité d'information de deux évènements indépendants est la somme de leurs quantités d'information
- $I(B \cap C) = I(B) + I(C)$
- $\varphi(b \cdot c) = \varphi(b) + \varphi(c)$
- φ continue de $[0, 1]$ dans $[0, +\infty]$

- fonction décroissante de la probabilité d'un évènement
- La quantité d'information de deux évènements indépendants est la somme de leurs quantités d'information
- $I(B \cap C) = I(B) + I(C)$
- $\varphi(b \cdot c) = \varphi(b) + \varphi(c)$
- φ continue de $[0, 1]$ dans $[0, +\infty]$
- $\varphi(Pr(E)) = -\log_2(Pr(E))$

Quantité d'information : version pratique

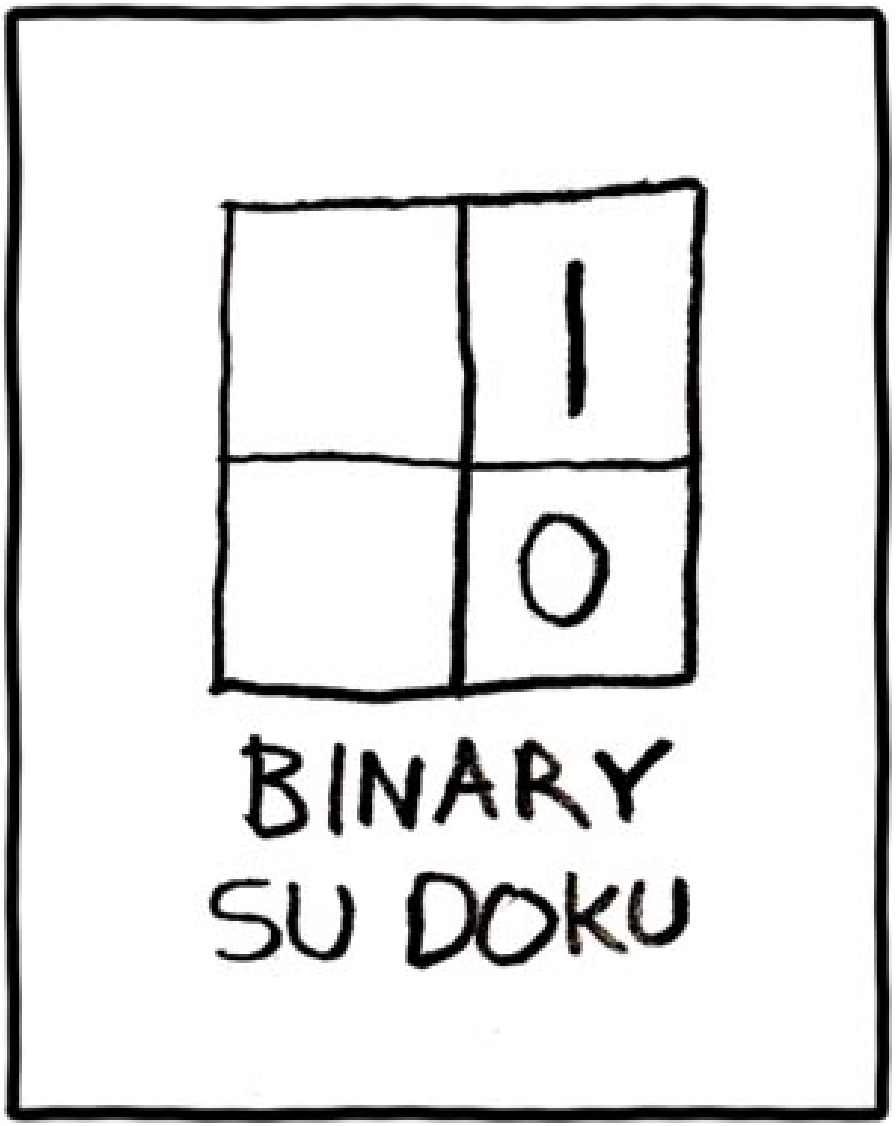
Nombre de questions auxquelles on ne peut répondre que par oui ou non qu'il faut poser pour être certain de retrouver les données.


Think
of a
Number!

- 1 bit = 1 question

- 1 bit = 1 question
- Je choisis une lettre de l'alphabet : combien de questions ?

- 1 bit = 1 question
- Je choisis une lettre de l'alphabet : combien de questions ?
- Je choisis une lettre au hasard dans *Notre-Dame de Paris* de Victor Hugo ?


1
0
BINARY
SU DOKU

sémantique

- *Quelque part sur la Terre* : 25 caractères soit environ 200 bits.

sémantique

- *Quelque part sur la Terre* : 25 caractères soit environ 200 bits.
- *À Nantes* : 8 caractères soit environ 64 bits.

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)
- Traditionnellement :

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)
- Traditionnellement :
  - un *kilo*octet représente 2^{10} octets

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)
- Traditionnellement :
  - un *kilo*octet représente 2^{10} octets
  - un *méga*octet représente 2^{20} octets

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)
- Traditionnellement :
  - un *kilo*octet représente 2^{10} octets
  - un *méga*octet représente 2^{20} octets
  - un *giga*octet représente 2^{30} octets

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)
- Traditionnellement :
  - un *kilo*octet représente 2^{10} octets
  - un *méga*octet représente 2^{20} octets
  - un *giga*octet représente 2^{30} octets
  - ...

Les unités

- Un octet (Byte) représente 8 bits (BInary digiT)
- Traditionnellement :
  - un *kilo*octet représente 2^{10} octets
  - un *méga*octet représente 2^{20} octets
  - un *giga*octet représente 2^{30} octets
  - ...
  - Depuis 1998, il fau*drai*t utiliser les préfixes *kibi*, *mébi*, *gibi*...et garder les autres pour les puissances de 10 par paquets de 3.

bit ≠ Byte

bit ≠ Byte

- 1 Mb = 1 mégabit = 10^6b

bit ≠ Byte

- 1 Mb = 1 mégabit = 10^6b
- 1 MB = 1 méga*byte* = 1 mégaoctet = 8 Mb

Numérique / Analogique

Numérique / Analogique

Échantillonnage On découpe l'information (temps / espace)

Numérique / Analogique

Échantillonnage On découpe l'information (temps / espace)

Quantification Approximation de chaque échantillon par une valeur choisie dans un ensemble fini

Numérique / Analogique

Échantillonnage On découpe l'information (temps / espace)

Quantification Approximation de chaque échantillon par une valeur choisie dans un ensemble fini

Encodage On associe un nombre à chacune des valeurs

Numérique / Analogique

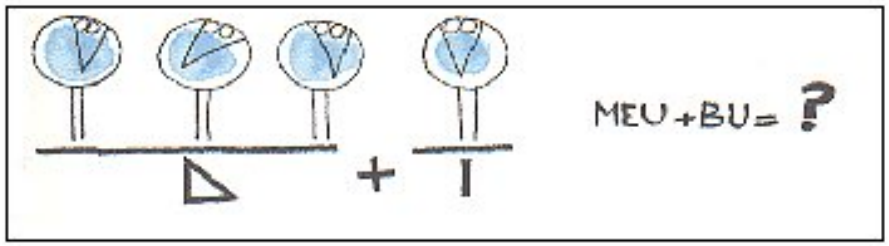
Échantillonnage On découpe l'information (temps / espace)

Quantification Approximation de chaque échantillon par une valeur choisie dans un ensemble fini

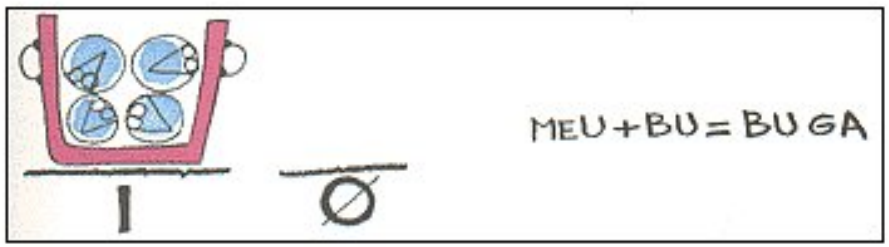
Encodage On associe un nombre à chacune des valeurs

Options compression, détection/correction d'erreurs

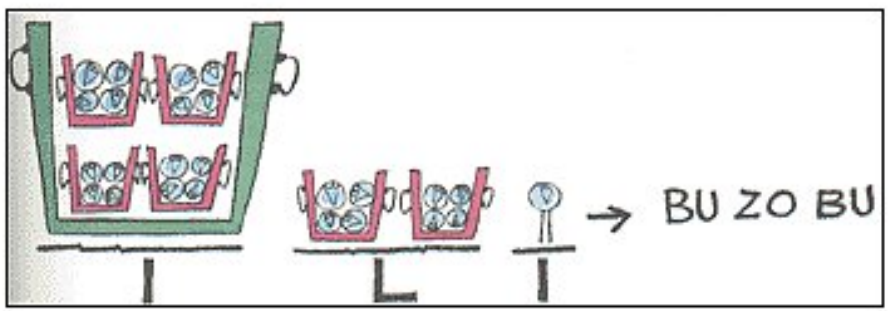
Sommaire



schema n°1



schema n°2



schema n°3

UNIVERSITÉ DE NANTES

- Entier non signé sur 32 bits $\longrightarrow$ 8 paquets de 4 bits $\longrightarrow$ 4 octets
- 1001 0111 1110 1010 0011 1011 1111 0101
- 9 7 E A 3 B F 5

Entiers signés

- Le bit de poids fort est réservé pour le signe : 1 pour les négatifs, 0 pour les positifs.
- Mais cela pose des problèmes si on se contente de ce codage. Par exemple, sur quatre bits, $3 + (-3)$ serait égal à...

```
  0 0 1 1
+ 1 0 1 1
---------
  1 1 1 0
```

...−6

Complément à 2^n

- Sur un octet par exemple, 2^8 est représenté par 8 zéros car on n'a pas la place pour écrire le 1 en neuvième position.
- Or $2^8 - 1$ s'écrit normalement 1111 1111
- Un nombre plus son complément à 1 s'écrit 1111 1111 avec l'algorithme usuel
- $\left(b_7\,b_6\,b_5\,b_4\,b_3\,b_2\,b_1\,b_0 + \bar{b_7}\,\bar{b_6}\,\bar{b_5}\,\bar{b_4}\,\bar{b_3}\,\bar{b_2}\,\bar{b_1}\,\bar{b_0}\right) + 1 = 1111\,1111 + 1 = (1)0000\,0000$
- donc $b_7\,b_6\,b_5\,b_4\,b_3\,b_2\,b_1\,b_0 + \left(\bar{b_7}\,\bar{b_6}\,\bar{b_5}\,\bar{b_4}\,\bar{b_3}\,\bar{b_2}\,\bar{b_1}\,\bar{b_0} + 1\right) = 0$
- On choisira donc comme opposé d'un nombre son complément à 1 plus 1.

UNIVERSITÉ DE NANTES

Exemple 1 (–27 sur 8 bits)

Valeur absolue $|-27|$ s'écrit 00011011

Complément à 1 11100100

On ajoute 1 11100101

Opposé ? on vérifie :

$$\begin{array}{rr} & 00011011 \\ + & 11100101 \\ \hline (1) & 00000000 \end{array}$$

Avec k bits, quels sont les entiers ainsi représentables ?

Sommaire

UNIVERSITÉ DE NANTES


OUPS !

```
In [1]: 3 * 0.1
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 3 * 0.1
Out[1]: 0.30000000000000004
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 3 * 0.1
Out[1]: 0.30000000000000004
In [2]: sum([0.1 for k in range(10000000)])
.
```

```
In [1]: 3 * 0.1
Out[1]: 0.30000000000000004
In [2]: sum([0.1 for k in range(10000000)])
Out[2]: 999999.9998389754
```

LES NOMBRES RÉELS N'EXISTENT PAS.

LES NOMBRES RÉELS N'EXISTENT PAS.

TOUT CE QUE VOUS AVEZ VU EN MATHS N'EST QU'ILLUSION !

UNIVERSITÉ DE NANTES

William Kahan (1933)

$$v = (-1)^s \times m \times 2^E$$

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32*

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32*

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64*

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64*

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7*

UNIVERSITÉ DE NANTES

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7*

UNIVERSITÉ DE NANTES

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

Oups ! $1 + 11 + 53 = 65$...En fait, on représente un nombre de $\mathbb{V}_{64}$ sous **forme normale** :

$$v = (-1)^s \times 1, f \times 2^E$$

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1} - 1$.

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1} - 1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1} - 1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E} - 1$.

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1} - 1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E} - 1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1} - 1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E} - 1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1} - 1$...

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1} - 1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E} - 1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1} - 1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

UNIVERSITÉ DE NANTES

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

UNIVERSITÉ DE NANTES

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E$$

UNIVERSITÉ DE NANTES

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1$$

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

UNIVERSITÉ DE NANTES

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

$$2 \times 0,75 = b_1 + b_2 \times 2^{-1} + \cdots$$

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

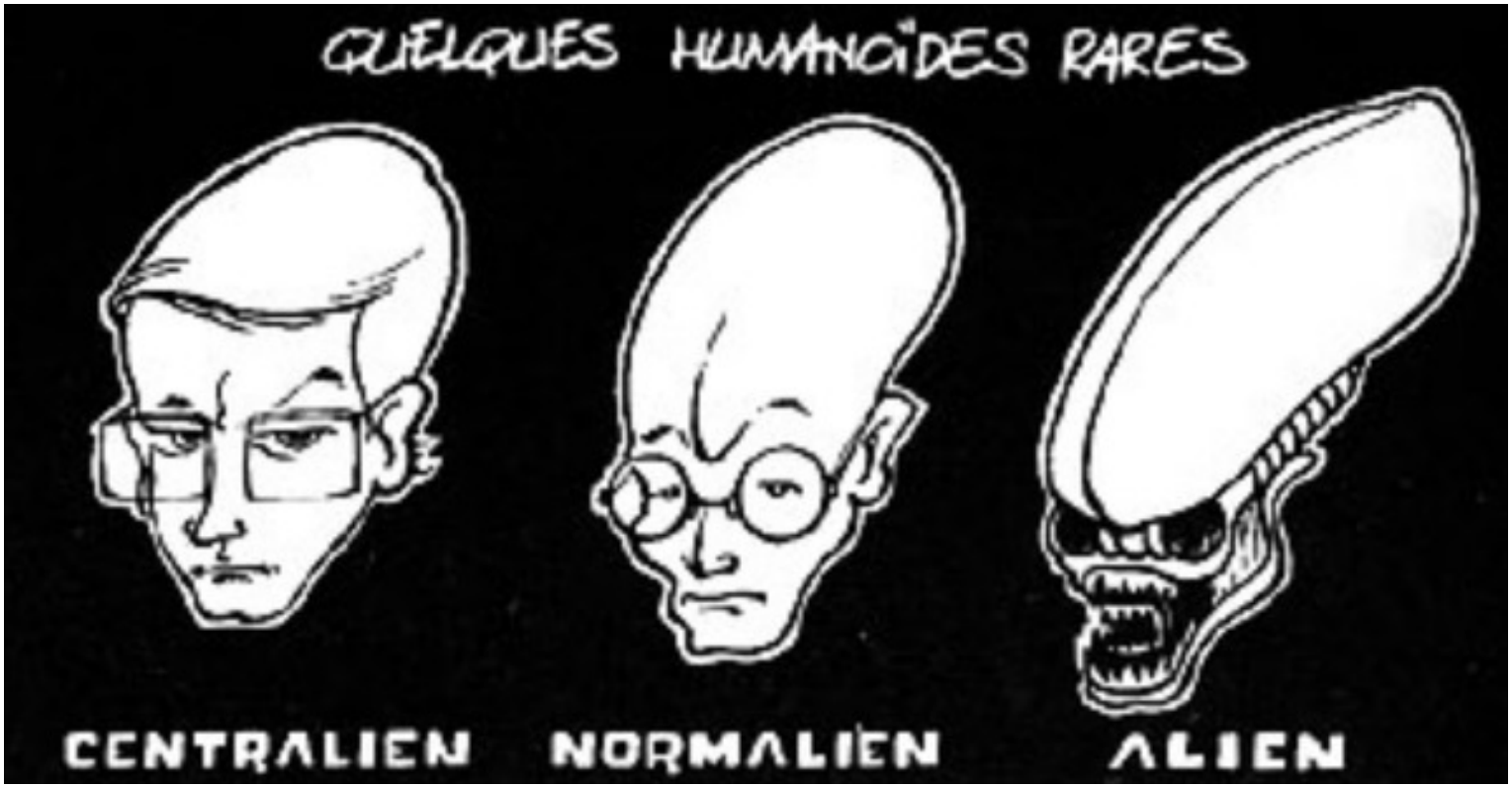
$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

$$2 \times 0,75 = b_1 + b_2 \times 2^{-1} + \cdots$$

On voit poindre un bel algo...

Exercice 1

Pourquoi le Patriot a raté son Scud de 600m ?
Représentez 1/10 avec une mantisse de 24 bits et tronquez le résultat.
Calculez l'erreur en seconde puis après 100 heures d'utilisation.
Sachant qu'un Scud vole à $1676\,m\,s^{-1}$*, quelle est environ l'erreur commise en mètres ?*
Et si on avait arrondi au plus proche ?


QUELQUES HUMANOÏDES RARES
CENTRALIEN
NORMALIEN
ALIEN

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
1	0	0	0	0	0	0

```
In [1]: x = 1e500

.
.
.
.
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: x = 1e500

In [2]: 1+x
.
.
.
.
.
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
.
.
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
.
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
Out[4]: 0.0
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
Out[4]: 0.0
In [5]: 4 - x
.
```

UNIVERSITÉ DE NANTES

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
Out[4]: 0.0
In [5]: 4 - x
Out[5]: -inf
```

$$\lim_{x \to +\infty} x^2 - x$$

$$\lim_{x \to +\infty} x^2 - x$$

```
In [9]: x**2 - x
.
```

UNIVERSITÉ DE NANTES

$$\lim_{x \to +\infty} x^2 - x$$

```
In [9]: x**2 - x
Out[9]: nan
```


NAN (PAIN INDIEN FOURRÉ AU FROMAGE)
MENU
cheese NAN KEBAB
MENU
cheese NAN CHIK'N

```
In [10]: x
Out[10]: inf

In [11]: x - x
Out[11]: nan

In [12]: x / x
Out[12]: nan

In [13]: x - x == x - x
Out[13]: False

In [14]: x**2 - x
Out[14]: nan

In [15]: x * (x - 1)
Out[15]: inf

In [16]: x * (x - 1) == x**2 - x
Out[16]: False
```

UNIVERSITÉ DE NANTES

Exercice 2

1. *Comment expliquer les résultats suivants :*

```
*Main> let f(x) = x^2 / sqrt(x^3 + 1)
*Main> f(1e100)
1.0e50
*Main> f(1e150)
0.0
*Main> f(1e200)
NaN
```

2. *Comment éviter le dernier NaN ?*

UNIVERSITÉ DE NANTES

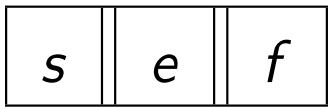
```
In [16]: f = lambda x: x**2 / sqrt(x**3 + 1)

In [17]: f(1e100)
Out[17]: 1e+50

In [18]: f(1e150)
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call
         ↪   last)
<ipython-input-18-79775d85f31a> in <module>()
----> 1 f(1e150)

<ipython-input-16-eed0b81ef932> in <lambda>(x)
----> 1 f = lambda x: x**2 / sqrt(x**3 + 1)

OverflowError: (34, 'Numerical result out of range')

In [19]: f(1e200)
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call
         ↪   last)
<ipython-input-19-c1d077e9d28e> in <module>()
----> 1 f(1e200)
```

```
In [53]: x = 1e-500

In [54]: x
Out[54]: 0.0

In [55]: x / x
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call
        ↪   last)
<ipython-input-55-fd52a7f8b5f1> in <module>()
----> 1 x / x

ZeroDivisionError: float division by zero
```

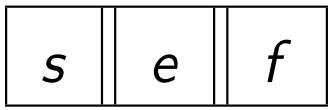
```
In [56]: sqrt(-1.0)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call
        ↪  last)
<ipython-input-56-d1c09f21b443> in <module>()
----> 1 sqrt(-1.0)

ValueError: math domain error
```

```
*Main> let x = 1e500
*Main> x - x
NaN
*Main> x / x
NaN
*Main> sqrt(-1)
NaN
*Main> 0 / 0
NaN
```

Arbre de lecture



Arbre de lecture



$e = 0...0$	$f = 0$	$\rightarrow$	$v = (-1)^s \times 0.0$
$e = 0...0$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 0.f \times 2^{1-E_{\max}}$
$e = 1...1$	$f = 0$	$\rightarrow$	$v = (-1)^s \times \infty$
$e = 1...1$	$f \neq 0$	$\rightarrow$	NaN
$0...0 < e < 1...1$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 1.f \times 2^{e-E_{\max}}$

UNIVERSITÉ DE NANTES

Successeur

$$n = \#f$$

Successeur

$$n = \#f$$

toy7,

Successeur

$$n = \#f$$

$toy7,\ n = 3$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32,

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64,

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\text{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

UNIVERSITÉ DE NANTES

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Nom	E	f	$E_{\min}$	$E_{\max}$	ε_m	λ	μ	Ω
toy7	3	3	-2	3	$2^{-3} = 1/8$	$2^{-2} = 1/4$	$2^{-5} = 1/32$	$1,111 \times 2^{3} = 15$
bin32	8	23	-126	127	2^{-23} $\approx 1,2 \times 10^{-7}$	2^{-126} $\approx 1,2 \times 10^{-38}$	$2^{-126-23}$ $\approx 1,4 \times 10^{-45}$	$(2^{24}-1) \times 2^{12}$ $\approx 3,4 \times 10^{38}$
bin64	11	52	-1022	1023	2^{-52} $\approx 2,2 \times 10^{-16}$	2^{-1022} $\approx 2,2 \times 10^{-308}$	2^{-1074} $\approx 5 \times 10^{-324}$	$(2^{53}-1)2^{1023-}$ $\approx 1,8 \times 10^{308}$

UNIVERSITÉ DE NANTES

```
In [1]: 1 + 1e-16 == 1
.
.
.
.
.
.
.
.
.
.
.
.
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True
.
.
.
.
.
.
.
.
.
.
.
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16
.
.
.
.
.
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
.
.
.
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

.
.
.
.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
.

.
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
Out[4]: 0.9999999999999999

.
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
Out[4]: 0.9999999999999999

In [5]: 1e-16 + 1e-18
.
```

UNIVERSITÉ DE NANTES

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
Out[4]: 0.9999999999999999

In [5]: 1e-16 + 1e-18
Out[5]: 1.01e-16
```

Sommaire

$$\mathbb{V}_b$$

$$\mathbb{V}_b$$

$$\overline{\mathbb{V}_b}$$

Comparaison

Il est très simple et rapide de comparer deux VF : comment la machine procède-t-elle ?
Quel est l'avantage de ce stockage des VF ?

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

En *toy7* : $1,1 + 0,0111$

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

UNIVERSITÉ DE NANTES

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Les arrondis

Définition 2

Les arrondis

Définition 2

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;

Les arrondis

Définition 2

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;

Les arrondis

Définition 2

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;

Les arrondis

Définition 2

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Les arrondis

Définition 2

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Les arrondis

Définition 2

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

1,100
0,001*11*
– – ––
1,101*11*

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,10111$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,10111$ VF en *toy7* ?

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,10111$ VF en *toy7* ?

$$1,10100 \leqslant \underbrace{1,10111}_{x} \leqslant 1,11000$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,10111$ VF en *toy7* ?

$$1,10100 \leqslant \underbrace{1,10111}_{x} \leqslant 1,11000$$

$$\underbrace{1,100 + 1,00\,ulp(x)}_{v} \leqslant \underbrace{1,100 + 1,11\,ulp(x)}_{x} \leqslant \underbrace{1,100 + 10,00\,ulp(x)}_{\mathbf{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,10111$ VF en *toy7* ?

$$1,10100 \leqslant \underbrace{1,10111}_{x} \leqslant 1,11000$$

$$\underbrace{1,100 + 1,00ulp(x)}_{v} \leqslant \underbrace{1,100 + 1,11ulp(x)}_{x} \leqslant \underbrace{1,100 + 10,00ulp(x)}_{\mathsf{succ}(v)}$$

$|x - v| = 0,11ulp(x)$ $|x - \mathsf{succ}(v)| = 0,01ulp(x)$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,10111$ VF en *toy7* ?

$$1,10100 \leqslant \underbrace{1,10111}_{x} \leqslant 1,11000$$

$$\underbrace{1,100 + 1,00\,ulp(x)}_{v} \leqslant \underbrace{1,100 + 1,11\,ulp(x)}_{x} \leqslant \underbrace{1,100 + 10,00\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

1,100
0,001*11*
– – ––
1,101*11*

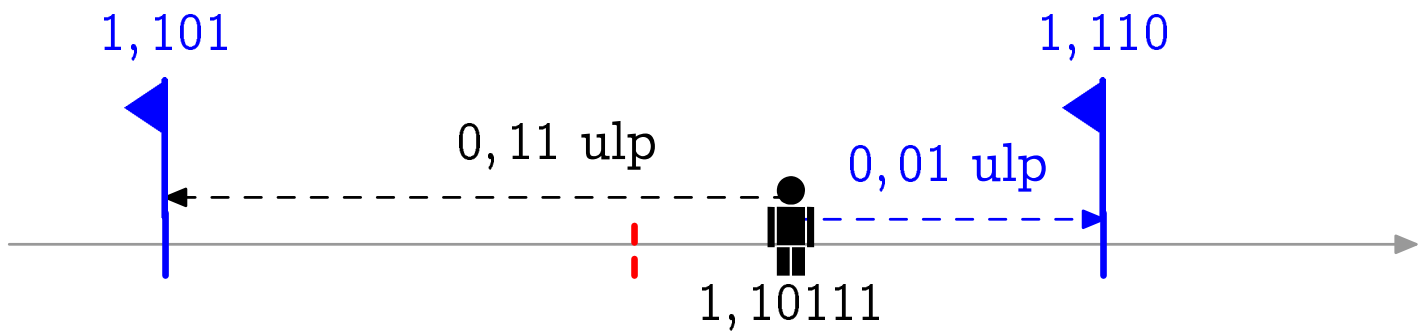
$x = 1,10111$ VF en *toy7* ?

$$1,10100 \leqslant \underbrace{1,10111}_{x} \leqslant 1,11000$$

$$\underbrace{1,100 + 1,00ulp(x)}_{v} \leqslant \underbrace{1,100 + 1,11ulp(x)}_{x} \leqslant \underbrace{1,100 + 10,00ulp(x)}_{\mathsf{succ}(v)}$$

$|x - v| = 0,11ulp(x)$ $|x - \mathsf{succ}(v)| = 0,01ulp(x)$ donc $RN(x) = \mathsf{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$


1, 101
1, 110
0, 11 ulp
0, 01 ulp
1, 10111

1. Est-ce que l'addition des VF est associative ?

1. Est-ce que l'addition des VF est associative ?

```
In [14]: (1 + 1e-16) + 1e-16
Out[14]: 1.0

In [15]: 1 + (1e-16 + 1e-16)
Out[15]: 1.0000000000000002
```

UNIVERSITÉ DE NANTES

1. Est-ce que l'addition des VF est associative ?

```
In [14]: (1 + 1e-16) + 1e-16
Out[14]: 1.0

In [15]: 1 + (1e-16 + 1e-16)
Out[15]: 1.0000000000000002
```

2. Est-on sûr de l'ordre dans le quel un compilateur calcule $a + b + c + d$?

UNIVERSITÉ DE NANTES

Multiplication

1. on « xore » les bits de signe ;

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

$101,1 \times (-10,01)$

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| =$

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| =$

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011\,ulp(x)$ et $|x - 1,101| = 0,101\,ulp(x)$

$101,1 \times (-10,01)$

101,1 :	0	1	0	1	0	1	1
-10,01 :	1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011\,ulp(x)$ et $|x - 1,101| = 0,101\,ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

UNIVERSITÉ DE NANTES

1. Est-ce que la multiplication des VF est associative ?

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

UNIVERSITÉ DE NANTES

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
In [3]: 1e15 * (1e-16 + 1)
Out[3]: 1000000000000000.0

In [4]: (1e15 * 1e-16) + (1e15 * 1)
Out[4]: 1000000000000000.1
```

UNIVERSITÉ DE NANTES

Sommaire

```python
def base1(a):
    if (a + 1.0) - a != 1.0:
        return a
    else:
        return base1(2.0 * a)

def base2(a, b):
    if (a + b) - a == b:
        return b
    else:
        return base2(a, b + 1.0)
```

```python
def base1(a):
    if (a + 1.0) - a != 1.0:
        return a
    else:
        return base1(2.0 * a)

def base2(a, b):
    if (a + b) - a == b:
        return b
    else:
        return base2(a, b + 1.0)
```

????!!!!????!!???

95 % of the folks out there are completely clueless about floating-point

James Gosling *(M. Java) - 28 février 1998*

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,177777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Feel nervous, but feel in control. It's not dark magic, it's science.

Florent DE DINECHIN

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left| \frac{x - \widehat{x}}{x} \right| \leqslant \left| \frac{x - \widehat{x}}{m \times 2^E} \right|$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left| \frac{x - \widehat{x}}{x} \right| \leqslant \left| \frac{x - \widehat{x}}{m \times 2^E} \right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left| \frac{x - \widehat{x}}{x} \right| \leqslant \left| \frac{x - \widehat{x}}{m \times 2^E} \right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m}$$

UNIVERSITÉ DE NANTES

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left| \frac{x - \widehat{x}}{x} \right| \leqslant \left| \frac{x - \widehat{x}}{m \times 2^E} \right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée

UNIVERSITÉ DE NANTES

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée

UNIVERSITÉ DE NANTES

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\mathbf{min}}-1-n}$

UNIVERSITÉ DE NANTES

NE FAITES PAS DES TESTS D'ÉGALITÉ MAIS DES TESTS D'APPARTENANCE À DES INTERVALLES DE LARGEUR L'ε MACHINE !

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Mouais...

Élimination

$$\widehat{a} = a(1 + \eta_a),$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b),$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a - b$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a - b$ et $\widehat{x} = \widehat{a - b}$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a - b$ et $\widehat{x} = \widehat{\widehat{a} - \widehat{b}} = \widehat{a} - \widehat{b}$.

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a - b$ et $\widehat{x} = \widehat{\widehat{a} - \widehat{b}} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x - \widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a - b}\right|$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a - b$ et $\widehat{x} = \widehat{\widehat{a} - \widehat{b}} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x - \widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\,\frac{|a| + |b|}{|a-b|}$$

Catastrophe

```
def deriv(f,h,x):
    return (f(x+h) - f(x)) / h
```

Catastrophe

```
def deriv(f,h,x):
    return (f(x+h) - f(x)) / h
```

```
def derivn(f,x,h,n):
    if n == 0:
        return f(x)
    else:
        return derivn(lambda x: deriv(f,h,x),x,h,n-1)
```

Catastrophe

```
In [1]: [derivn(lambda x: x**4,1,1e-7,k) for k in range(5)]
Out[1]:
[1,
 4.000000601855902,
 12.012613126444194,
 -222044.6049250313,
 2220446049250.313]
```

Catastrophe

```
In [1]: [derivn(lambda x: x**4,1,1e-7,k) for k in range(5)]
Out[1]:
[1,
 4.000000601855902,
 12.012613126444194,
 -222044.6049250313,
 2220446049250.313]
```

```
In [2]: [derivn(lambda x: x**4,1,1e-8,k) for k in range(5)]
Out[2]: [1, 4.000000042303498, 11.102230246251565, 0.0,
         ↪   2.220446049250313e+16]
```

Catastrophe

```
In [3]: [derivn(lambda x: x**4,1,1e-9,k) for k in range(5)]
Out[3]: [1, 4.000000330961484, 0.0, 0.0, 0.0]
```

Catastrophe

```
In [3]: [derivn(lambda x: x**4,1,1e-9,k) for k in range(5)]
Out[3]: [1, 4.000000330961484, 0.0, 0.0, 0.0]
```

```
In [4]: [derivn(lambda x: x**4,1,1e-3,k) for k in range(6)]
Out[4]:
[1,
 4.006004000999486,
 12.024014000244776,
 24.03599941303014,
 24.001245435556484,
 -2.4424906541753444]
```

UNIVERSITÉ DE NANTES

Catastrophe

```
In [5]: [derivn(lambda x: x**4,1,2**-10,k) for k in range(6)]
Out[5]: [1, 4.005863190628588, 12.02345085144043, 24.03515625, 24.0,
         ↪  0.0]
```

Catastrophe

```
In [5]: [derivn(lambda x: x**4,1,2**-10,k) for k in range(6)]
Out[5]: [1, 4.005863190628588, 12.02345085144043, 24.03515625, 24.0,
         ↪   0.0]
```

```
In [6]: [derivn(lambda x: x**4,1,1.0/1026.,k) for k in range(6)]
Out[6]:
[1,
 4.005851753982072,
 12.023405112200917,
 24.035087928668187,
 23.999573957547014,
 0.755876563500351]
```

Catastrophe

```
In [7]: [derivn(lambda x: x**4,1,2**-20,k) for k in range(6)]
Out[7]: [1, 4.000005722045898, 12.0, 0.0, 268435456.0,
         ↪   -844424930131968.0]
```

Catastrophe

```
In [7]: [derivn(exp,0,2**-5,k) for k in range(10)]
Out[7]:
[1.0,
 1.0157890399712883,
 1.0318273737254913,
 1.0481189373895177,
 1.0646677284967154,
 1.081477828323841,
 1.0985534191131592,
 1.1158447265625,
 1.1376953125,
 0.953125]
```

Catastrophe

```
In [8]: [derivn(exp,0,2**-20,k) for k in range(10)]
Out[8]: [1.0, 1.0000004768371582, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
         ↪   0.0]
```

TROP DE PRÉCISION TUE LA PRÉCISION !

TROP DE PRÉCISION TUE LA PRÉCISION ! DIVISEZ PAR DES PUISSANCES DE 2...

TROP DE PRÉCISION TUE LA PRÉCISION !
DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

Exercice 3

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ *avec* $a \neq 0$*...*
Les racines, si elles existent, sont données par une formule bien connue dépendant de a*,* b *et* c *:*

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

Exercice 3

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ *avec* $a \neq 0$*...*
Les racines, si elles existent, sont données par une formule bien connue dépendant de a*,* b *et* c *:*

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque* $b^2 \gg |4ac|$ *? En quoi la formule* $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ *peut aider ?*

Exercice 3

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*

UNIVERSITÉ DE NANTES

Exercice 3

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*

UNIVERSITÉ DE NANTES

Exercice 3

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*

UNIVERSITÉ DE NANTES

Exercice 3

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*

UNIVERSITÉ DE NANTES

Élimination

```
def expn(n):
    return (1. + 1./n)**n
```

Élimination

```
In [1]: [(exp(1) - expn(10.0**k))
        ↪   for k in range(20)]
Out[1]:
[0.7182818284590451,
 0.12453936835904278,
 0.01346799903751661,
 0.0013578962234515046,
 0.000135901634119584,
 1.359126674760347e-05,
 1.359363291708604e-06,
 1.3432696333026684e-07,
 3.011168736577474e-08,
```

```
-2.2355251516614771e-07,
-2.2477574246337895e-07,
-2.248980650598753e-07,
-0.00024166757819266138,
0.002171794372144209,
0.0021717943720220845,
-0.31675337809021675,
1.718281828459045,
1.718281828459045,
1.718281828459045,
1.718281828459045]
```

UNIVERSITÉ DE NANTES

Élimination

```
In [2]: [(exp(1) -
        ↪   expn(2.0**(3*k))) for
        ↪   k in range(20)]
Out[2]:
[0.7182818284590451,
 0.1524973145086972,
 0.020936875893946105,
 0.0026498282900537795,
 0.0003317472693793455,
 4.147652875108321e-05,
 5.1846929989274315e-06,
 6.480886076687398e-07,
 8.101110671177025e-08,
```

```
1.0126388616527038e-08,
1.2657985770658797e-09,
1.582245445774788e-10,
1.977795704988239e-11,
2.4722446312352986e-12,
3.090860900556436e-13,
3.863576125695545e-14,
4.884981308350689e-15,
4.440892098500626e-16,
1.718281828459045,
1.718281828459045]
```

```
In [89]: [2.0**53 + k for k in range(10)]
Out[89]:
[9007199254740992.0,
 9007199254740992.0,
 9007199254740994.0,
 9007199254740996.0,
 9007199254740996.0,
 9007199254740996.0,
 9007199254740998.0,
 9007199254741000.0,
 9007199254741000.0,
 9007199254741000.0]
```

```
In [91]: [2.0**55 + k for k in range(10)]
Out[91]:
[3.602879701896397e+16,
 3.602879701896397e+16,
 3.602879701896397e+16,
 3.602879701896397e+16,
 3.602879701896397e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16]
```

Attention !

Pour éviter de déduire des lemmes suivants des théorèmes totalement faux, n'oubliez pas que DANS CE QUI SUIT X ET Y SONT DES NOMBRES À VIRGULE FLOTTANTE !

Lemme 3 (Majoration de l'erreur d'une somme)

Posons $x \oplus y = x + y + \mathsf{err}(x \oplus y)$. *Alors, s'il n'y a pas de dépassement de capacité,*

$$|\mathsf{err}(x \oplus y)| \leqslant \min(|x|, |y|)$$

On a bien sûr un résultat analogue pour la différence

- $x \oplus y = x + y + \mathsf{err}(x \oplus y)$

- $x \oplus y = x + y + \mathsf{err}(x \oplus y)$
- $x = x + y + (-y)$

- $x \oplus y = x + y + \mathsf{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$

- $x \oplus y = x + y + \mathsf{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ est le VF le plus proche de $x + y$

- $x \oplus y = x + y + \mathsf{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ est le VF le plus proche de $x + y$
- $|\mathsf{err}(x \oplus y)| \leqslant |y|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ est le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ est le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ est le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

Faites un dessin...

UNIVERSITÉ DE NANTES

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ est le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

Faites un dessin...

À noter

De l'importance de disposer avec la IEEE 754 de la **meilleure approximation** !

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathsf{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

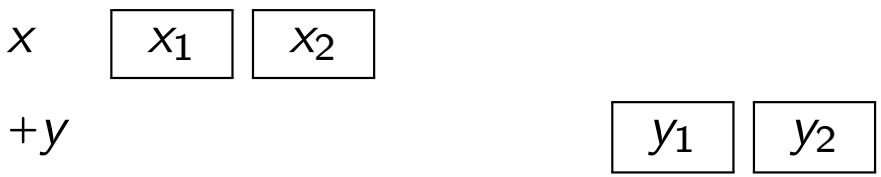
- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

  x $\boxed{x_1}$ $\boxed{x_2}$

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$



Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

$$
\begin{array}{lcccc}
x & \boxed{x_1} & \boxed{x_2} & & \\
+y & & & \boxed{y_1} & \boxed{y_2} \\
= & \boxed{x_1} & \boxed{x_2 + y_1} & \mathrm{err}(x \oplus y) &
\end{array}
$$

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

  x $\boxed{x_1}$ $\boxed{x_2}$

  $+y$ $\boxed{y_1}$ $\boxed{y_2}$

  $=$ $\boxed{x_1}$ $\boxed{x_2 + y_1}$ $\mathrm{err}(x \oplus y)$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$

UNIVERSITÉ DE NANTES

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

  x $\boxed{x_1}$ $\boxed{x_2}$

  $+y$ $\boxed{y_1}$ $\boxed{y_2}$

  $=$ $\boxed{x_1}$ $\boxed{x_2 + y_1}$ $\mathrm{err}(x \oplus y)$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$

UNIVERSITÉ DE NANTES

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* $p = ulp(y)$ *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

  x $\boxed{x_1}$ $\boxed{x_2}$

  $+y$ $\boxed{y_1}$ $\boxed{y_2}$

  $=$ $\boxed{x_1}$ $\boxed{x_2 + y_1}$ $\mathrm{err}(x \oplus y)$

- $|\mathrm{err}(x \oplus y)| \leqslant |y|$donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

UNIVERSITÉ DE NANTES

Lemme 5

Supposons que $|x + y| \leqslant \min(|x|, |y|)$*, alors* $x \oplus y = x + y$.
On obtient un résultat analogue pour la soustraction.

Lemme 5

Supposons que $|x + y| \leqslant \min(|x|, |y|)$*, alors* $x \oplus y = x + y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$

Lemme 5

Supposons que $|x+y| \leqslant \min(|x|,|y|)$*, alors* $x \oplus y = x + y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

Lemme 5

Supposons que $|x+y| \leqslant \min(|x|,|y|)$*, alors* $x \oplus y = x + y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

DANGER

Nous avons démontré nos lemmes en considérant des VF x et y.
Est-ce que le résultat suivant contredit notre dernier lemme ?

```
In [86]: 1 - 0.9
Out[86]: 0.09999999999999998
```

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

- Concrètement, à quoi correspond cette condition $\frac{x}{2} \leqslant y \leqslant 2x$?

- Concrètement, à quoi correspond cette condition $\frac{x}{2} \leqslant y \leqslant 2x$?
- Demandez-vous ce que l'on peut dire de l'écart maximum entre les exposants de x et y.

```
In [9]: 2.**53
Out[9]: 9007199254740992.0

In [10]: 2.**53 + 1
Out[10]: 9007199254740992.0

In [11]: 2.**53 + 2
Out[11]: 9007199254740994.0

In [12]: 2.**53 + 3
Out[12]: 9007199254740996.0

In [13]: 2.**53 + 4
Out[13]: 9007199254740996.0

In [14]: 2.**53 + 5
Out[14]: 9007199254740996.0
```

UNIVERSITÉ DE NANTES

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53}$$

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53}$$

0 10000110100 0000000000000000000000000000000000000000...00000

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53}$$

0 10000110100 0000000000000000000000000000000000000000...00000

$$1 = (-1)^0 \times 1,000...000 \times 2^0$$

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53}$$

0 10000110100 0000000000000000000000000000000000000000...00000

$$1 = (-1)^0 \times 1,000...000 \times 2^0$$

0 01111111111 0000000000000000000000000000000000000000...00000

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53}$$

0 10000110100 0000000000000000000000000000000000000000...00000

$$1 = (-1)^0 \times 1,000...000 \times 2^0$$

0 01111111111 0000000000000000000000000000000000000000...00000

$$1 = 0,\underbrace{000...000}_{52 bits}\ 1 \times 2^{53}$$

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53}$$

```
0 10000110100 0000000000000000000000000000000000000...00000
```

$$1 = (-1)^0 \times 1,000...000 \times 2^0$$

```
0 01111111111 0000000000000000000000000000000000000...00000
```

$$1 = 0,\underbrace{000...000}_{52 bits}\ 1 \times 2^{53}$$

```
1,0000000000000000000000000000000000000000000000000000|
0,0000000000000000000000000000000000000000000000000000|1
----------------------------------------------------------
1,0000000000000000000000000000000000000000000000000000|1
```

Unit in the Last Place

Unit in the Last Place

Mauvaise nouvelle : une addition entre deux nombres flottants peut donc, dans certains cas, créer une erreur.

Unit in the Last Place

Mauvaise nouvelle : une addition entre deux nombres flottants peut donc, dans certains cas, créer une erreur.

Bonne nouvelle : on peut récupérer cette erreur.

Théorème 7 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

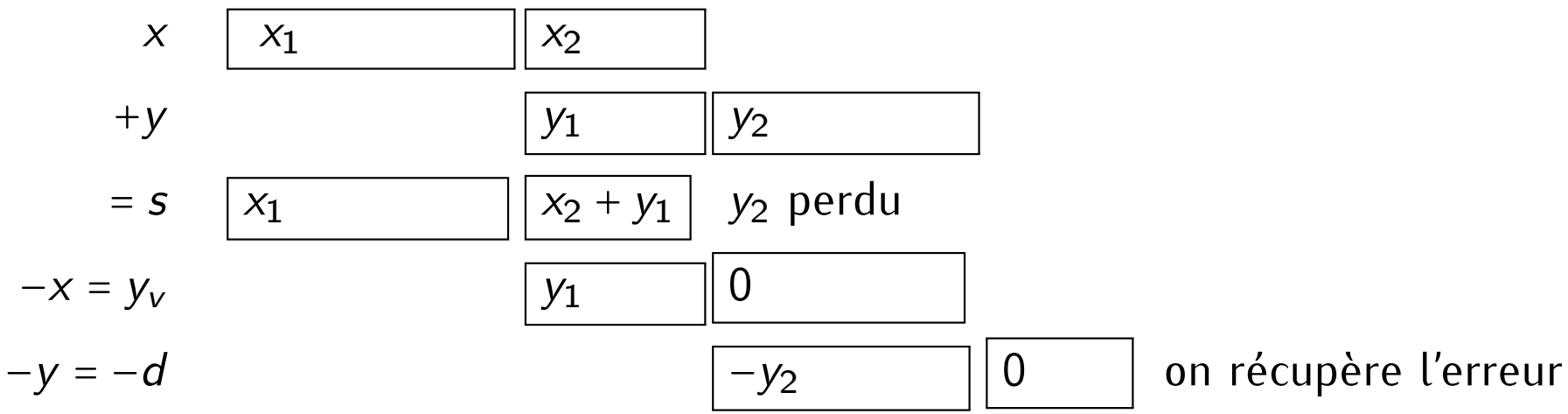
```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme de Sterbenz ;

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme de Sterbenz ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$.

Somme compensée



```
def fast2sum(x,y):
    if abs(x) >= abs(y):
        s  = x + y
        yv = s - x
        d  = y - yv
        return (s,d)
    else:
        return fast2sum(y,x)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [104]: fast2sum(2.0**54,5.0)
Out[104]: (1.8014398509481988e+16, 1.0)
```

Théorème 8 (2Sum - Knuth)

On considère deux VF x et y et l'algorithme suivant :

1 $s \leftarrow x \oplus y$

2 $y_v \leftarrow s \ominus x$

3 $x_v \leftarrow s \ominus y_v$

4 $y_a \leftarrow y \ominus y_v$

5 $x_a \leftarrow x \ominus x_v$

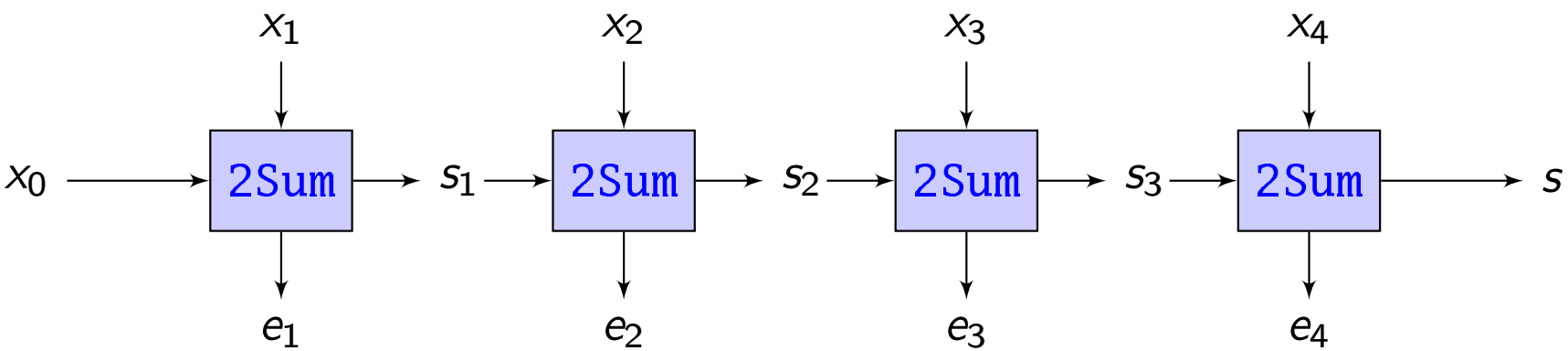
6 $d \leftarrow x_a \oplus y_a$

7 $\texttt{Retourner } (s, d)$

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

UNIVERSITÉ DE NANTES

Somme compensée d'un nombre quelconque de flottants



UNIVERSITÉ DE NANTES

Somme compensée d'un nombre quelconque de flottants

```
def sommeKahan(liste):
    (s,c) = (0.,0.)
    for x in liste:
        (s,c) = fast2sum(s, x + c)
    return s
```

Somme compensée d'un nombre quelconque de flottants

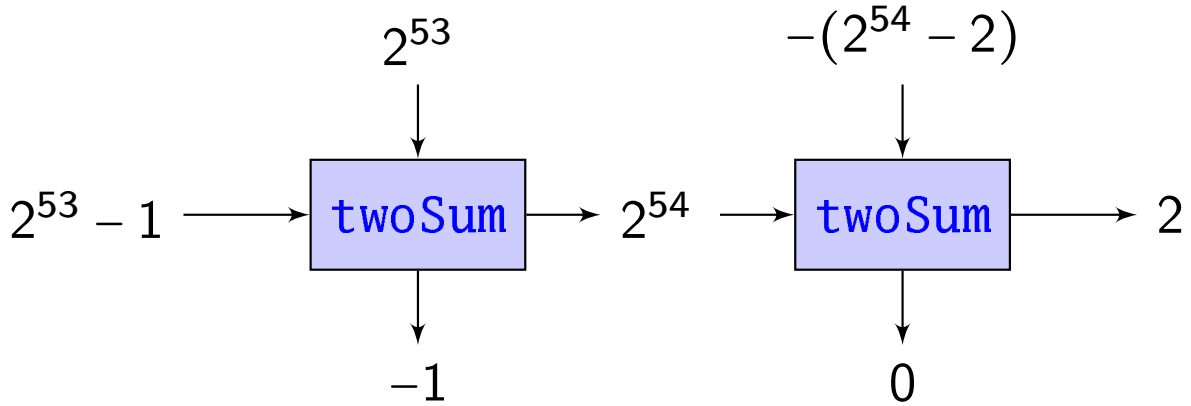
```
def sommeKahan(liste):
    (s,c) = (0.,0.)
    for x in liste:
        (s,c) = fast2sum(s, x + c)
    return s
```

```
def sommePichat(liste):
    s,e = 0.,0.
    for x in liste:
        (s,c) = fast2sum(s, x)
        e += c
    return s + e
```

UNIVERSITÉ DE NANTES

$x_0 = 2^{53} - 1$, $x_1 = 2^{53}$ et $x_2 = -(2^{54} - 2)$

$x_0 = 2^{53} - 1$, $x_1 = 2^{53}$ et $x_2 = -(2^{54} - 2)$



Exercice 4

Expliquez les résultats trouvés. Que va faire l'algorithme de somme compensée ? Et celui de somme en cascade ?

```
In [105]: sommePichat([1. / n for n in range(1,100001)])
Out[105]: 12.090146129863427

In [106]: sommeKahan([1. / n for n in range(1,100001)])
Out[106]: 12.090146129863427

In [107]: sum([1. / n for n in range(1,100001)])
Out[107]: 12.090146129863335

In [108]: sum([1. / n for n in range(100000,0,-1)])
Out[108]: 12.090146129863408
```

```
In [105]: sommePichat([1. / n for n in range(1,100001)])
Out[105]: 12.090146129863427

In [106]: sommeKahan([1. / n for n in range(1,100001)])
Out[106]: 12.090146129863427

In [107]: sum([1. / n for n in range(1,100001)])
Out[107]: 12.090146129863335

In [108]: sum([1. / n for n in range(100000,0,-1)])
Out[108]: 12.090146129863408
```

```
sage: sum(1. / n, n , 1 , 100000).n(64)
12.090146129863427
```

Banque chaotique

Exemple dû à Jean-Michel Muller.

Banque chaotique

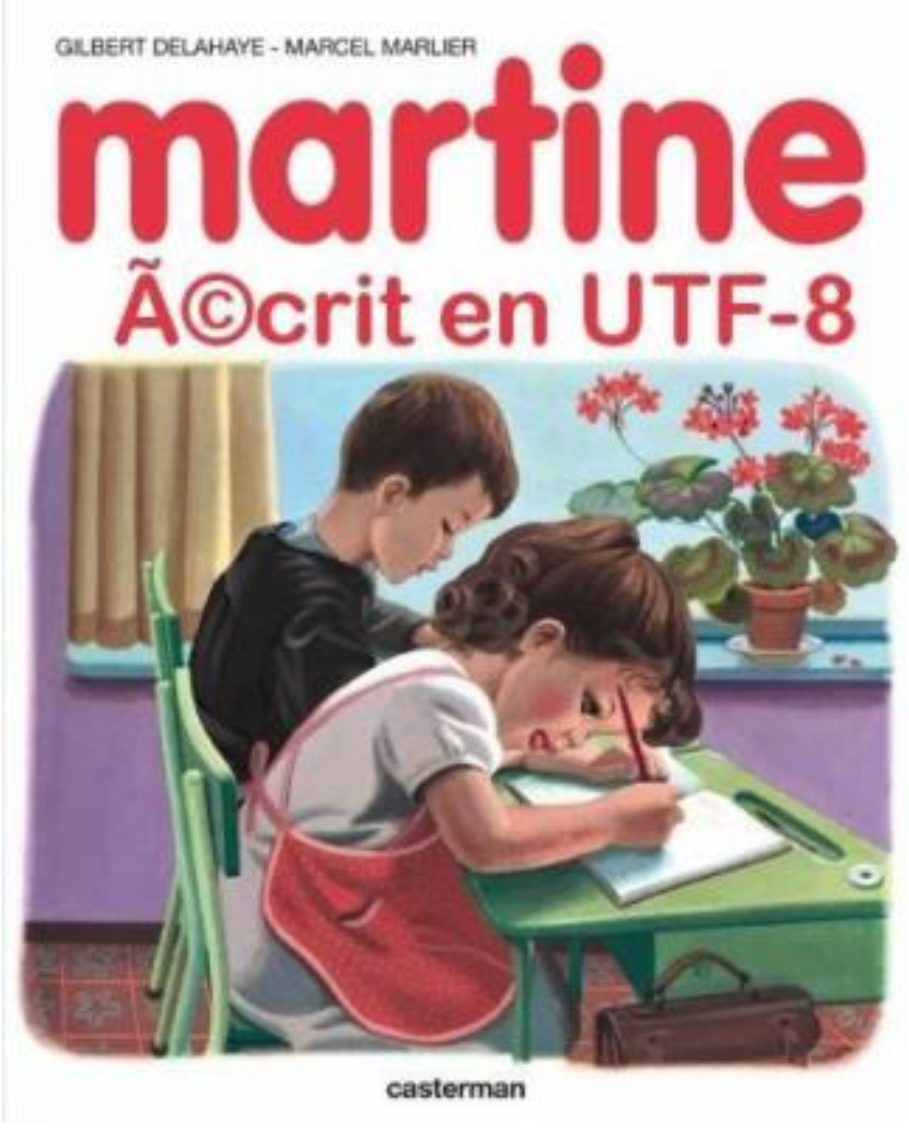
Exemple dû à Jean-Michel Muller.
M. X a récemment été à sa banque (Chaotic Bank Society), pour connaître les nouvelles offres proposées aux meilleurs clients. Son banquier lui propose l'offre suivante : « vous déposez tout d'abord e - 1 euros sur votre compte (où e = 2.7182818... est la base du logarithme népérien). La première année, nous prenons 1 euro sur votre compte de frais de gestion. Par contre, la deuxième année est plus intéressante pour vous, car nous multiplions votre capital restant par 2 et prenons 1 euro de frais de gestion. La troisième année est encore plus intéressante, car nous multiplions votre capital par 3 et prenons 1 euro de frais de gestion. Et ainsi de suite : la n-ième année, nous multiplions votre capital par n et prenons 1 euro de frais de gestion. Intéressant, non ? » Pour prendre sa décision, M. X décide de demander l'aide d'un informaticien et se retourne vers vous.

```python
def chaotic(n):
    cpt = exp(1) - 1
    for i in range(1,n + 1):
        cpt = i*cpt - 1
    return cpt
```

```
def chaotic(n):
    cpt = exp(1) - 1
    for i in range(1,n + 1):
        cpt = i*cpt - 1
    return cpt
```

$$
\begin{aligned}
c_n &= n! \times \left(c_0 - 1 - \frac{1}{2!} - \frac{1}{3!} - \cdots - \frac{1}{n!} \right) \\
&= n! \times \left(c_0 - (e-1) + \frac{1}{(n+1)!} + \frac{1}{(n+2)!} + \cdots \right)
\end{aligned}
$$

Sommaire


GILBERT DELAHAYE - MARCEL MARLIER
martine
Ã©crit en UTF-8
casterman

ASCII

- **American** Standard Code for Information Interchange

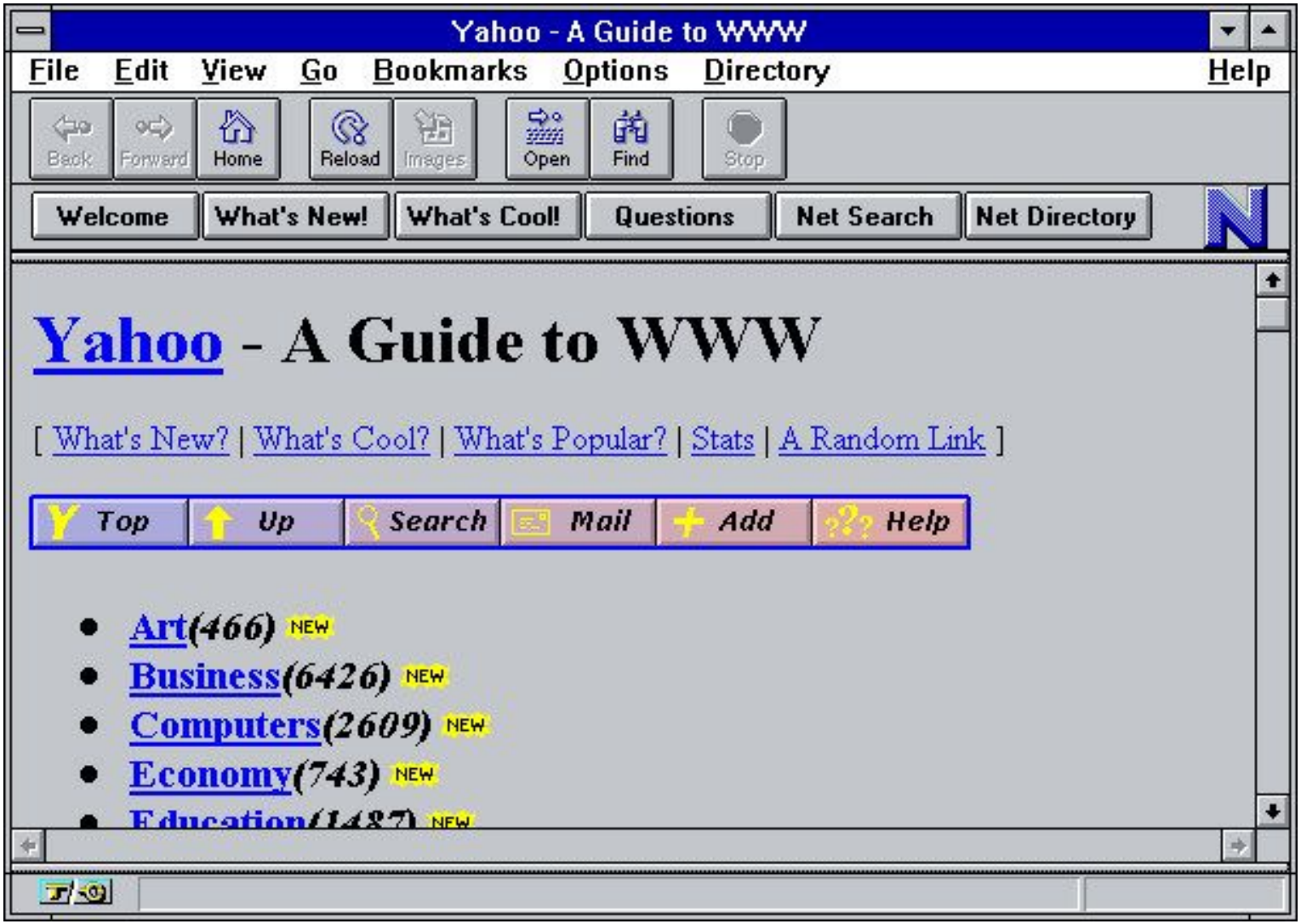
ASCII

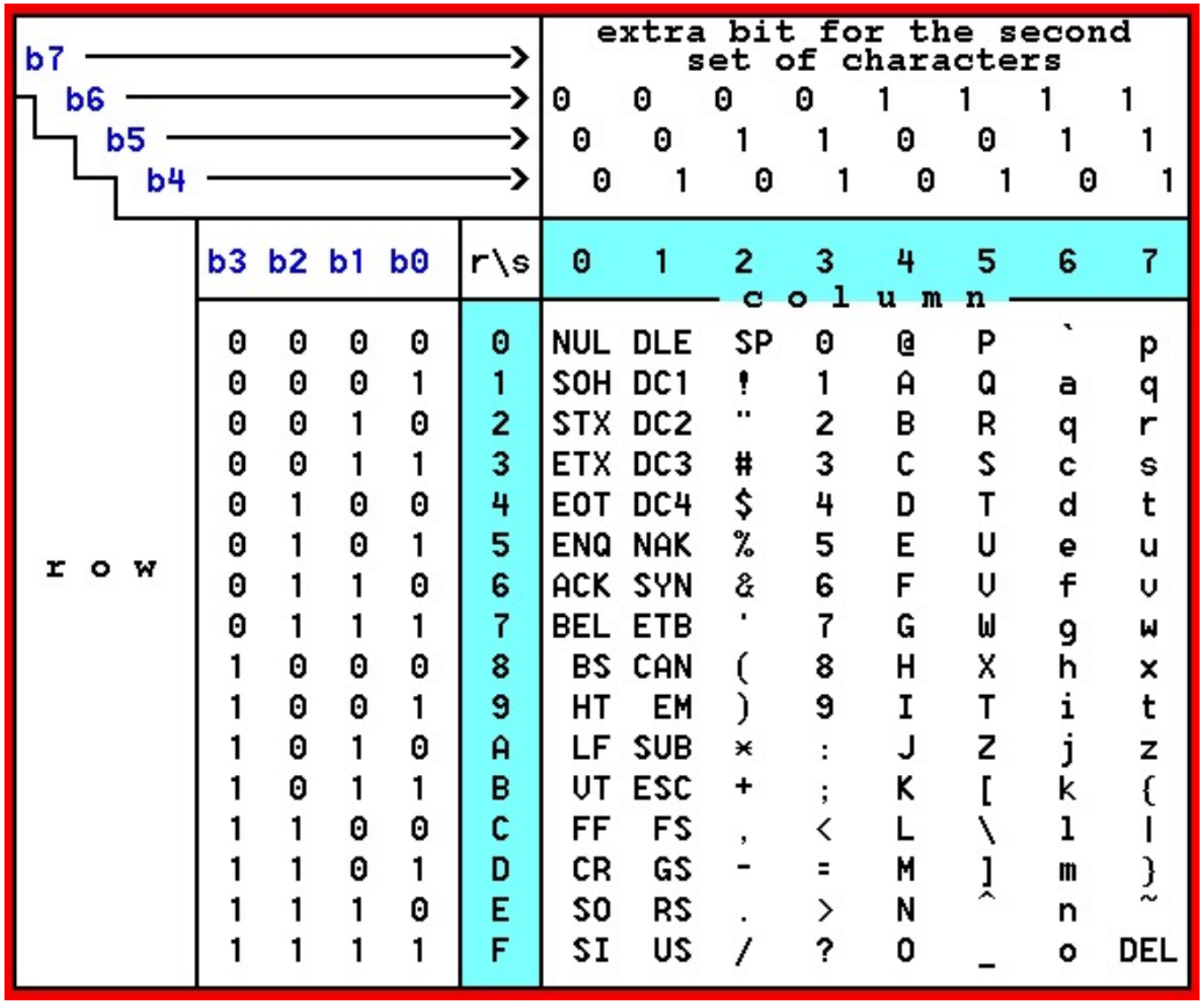
- **American** Standard Code for Information Interchange
- 128 caractères pour la langue anglaise

ASCII

- **American** Standard Code for Information Interchange
- 128 caractères pour la langue anglaise
- sur combien de bits ?


Pages Jaunes 3611
(c) 2003
France telecom
Nom/Marque:
Activité:
Localité:
Vous pouvez préciser
Département:
Adresse:
Prénom:
Infos éditeur
Comment remplir
autre recherche


Yahoo - A Guide to WWW
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Find Stop
Welcome What's New! What's Cool! Questions Net Search Net Directory
Yahoo - A Guide to WWW
[What's New? | What's Cool? | What's Popular? | Stats | A Random Link]
Top Up Search Mail Add Help
Art(466) NEW
Business(6426) NEW
Computers(2609) NEW
Economy(743) NEW
Education(1487) NEW



b7 →				extra bit for the second set of characters							
b6 →				0	0	0	0	1	1	1	1
b5 →				0	0	1	1	0	0	1	1
b4 →				0	1	0	1	0	1	0	1
	b3	b2	b1	b0	r\s	0	1	2	3	4	5	6	7
						column							
row	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	q	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	T	i	t
	1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	B	VT	ESC	+	;	K	[	k	{
	1	1	0	0	C	FF	FS	,	<	L	\	l	|
	1	1	0	1	D	CR	GS	-	=	M	]	m	}
	1	1	1	0	E	SO	RS	.	>	N	^	n	~
	1	1	1	1	F	SI	US	/	?	O	_	o	DEL

UNIVERSITÉ DE NANTES

ISO-8859-1 / Latin-1

- International Organization for Standardization

ISO-8859-1 / Latin-1

- International Organization for Standardization
- 8 bits pour les langues d'Europe de l'Ouest

ISO-8859-1																
	x0	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

UNIVERSITÉ DE NANTES

Unicode

Espace pouvant contenir $0x10ffff$ soit $1\,114\,112$ codes différents.

UTF-8

- Unicode Transformation Format : un standard pour encoder les points de code.

UTF-8

- Unicode Transformation Format : un standard pour encoder les points de code.
- Peut représenter tous les points de code de l'Unicode

UTF-8

- Unicode Transformation Format : un standard pour encoder les points de code.
- Peut représenter tous les points de code de l'Unicode
- Un caractère est représenté sur 1 à 4 octets

UTF-8

- Unicode Transformation Format : un standard pour encoder les points de code.
- Peut représenter tous les points de code de l'Unicode
- Un caractère est représenté sur 1 à 4 octets
- Les caractères les plus fréquents sont représentés sur les codes de plus petite taille

UTF-8

- Unicode Transformation Format : un standard pour encoder les points de code.
- Peut représenter tous les points de code de l'Unicode
- Un caractère est représenté sur 1 à 4 octets
- Les caractères les plus fréquents sont représentés sur les codes de plus petite taille
- compatible avec les caractères ASCII

Fonctionnement de l'UTF-8

De	à	1er octet	2ème octet	3ème octet	4ème octet
0x0	0x7F	$0 b_7 b_6 b_5 b_4 b_3 b_2 b_1$	/	/	/
0x80	0x7FF	$110 b_{11} b_{10} b_9 b_8 b_7$	$10 b_6 b_5 b_4 b_3 b_2 b_1$	/	/
0x800	0xFFFF	$1110 b_{16} b_{15} b_{14} b_{13}$	$10 b_{12} b_{11} b_{10} b_9 b_8 b_7$	$10 b_6 b_5 b_4 b_3 b_2 b_1$	/
0x10000	0x10FFFF	$11110 b_{21} b_{20} b_{19}$	$10 b_{18} b_{17} b_{16} b_{15} b_{14} b_{13}$	$10 b_{12} b_{11} b_{10} b_9 b_8 b_7$	$10 b_6 b_5 b_4 b$

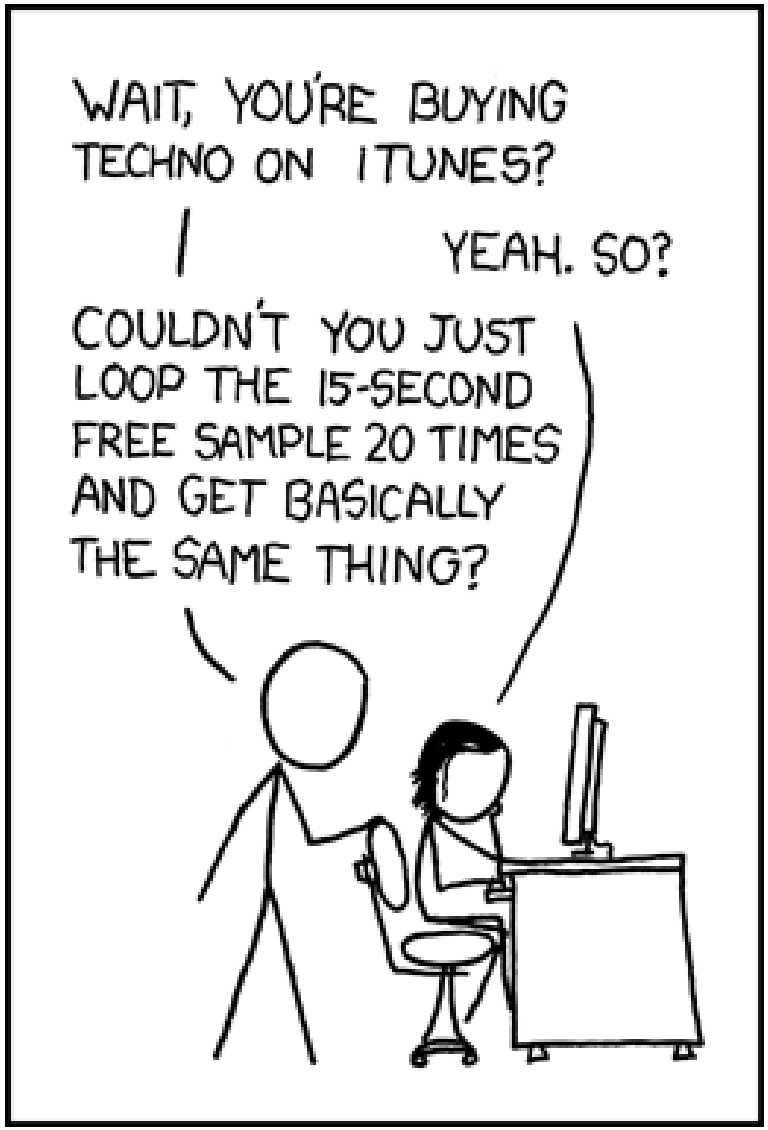
UNIVERSITÉ DE NANTES

Point de code $0x3B1 \longrightarrow$ caractère $\alpha \longrightarrow$ 11001110 10110001

Point de code 0x3B1 ⟶ caractère α ⟶ 11001110 10110001

```
In [4]: chr(0x3b1)
Out[4]: 'α'

In [5]: ord('é')
Out[5]: 233
```

Sommaire


WAIT, YOU'RE BUYING TECHNO ON ITUNES?
YEAH. SO?
COULDN'T YOU JUST LOOP THE 15-SECOND FREE SAMPLE 20 TIMES AND GET BASICALLY THE SAME THING?

0000000000000000111111111111111111

1111111111000000000000000111111111

0000000000000000111111111111000000

0000111111111111000000000100000111

0000000000001111111110000000111111

0000000011111111110000000111110000

0000000000000000111111111111111111

1111111111000000000000000011111111

0000000000000000111111111111000000

0000111111111111000000001000000111

0000000000001111111100000000111111

0000000011111111110000000011111000

$$\underset{17}{\underline{0000000000000000}}\underset{19}{\underline{1111111111111111111}}$$

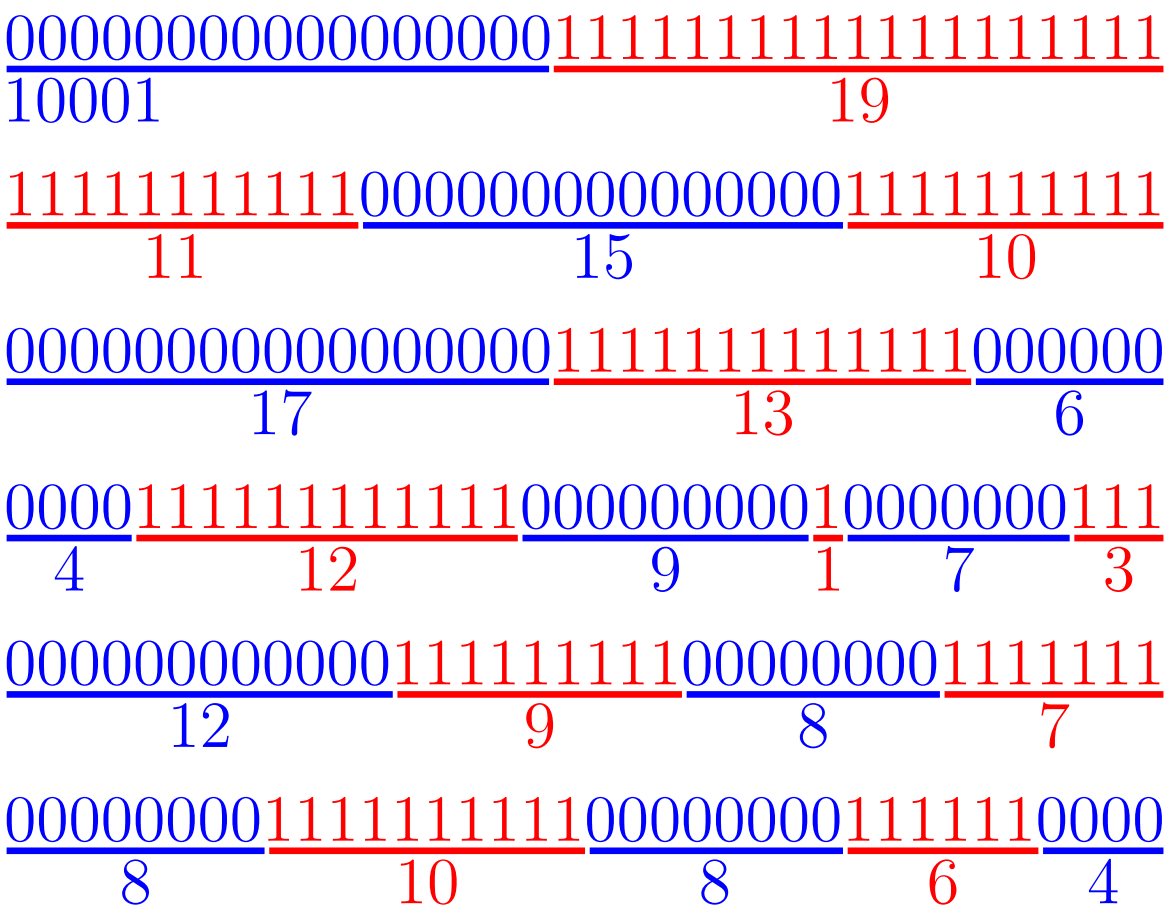
$$\underset{11}{\underline{11111111111}}\underset{15}{\underline{000000000000000}}\underset{10}{\underline{1111111111}}$$

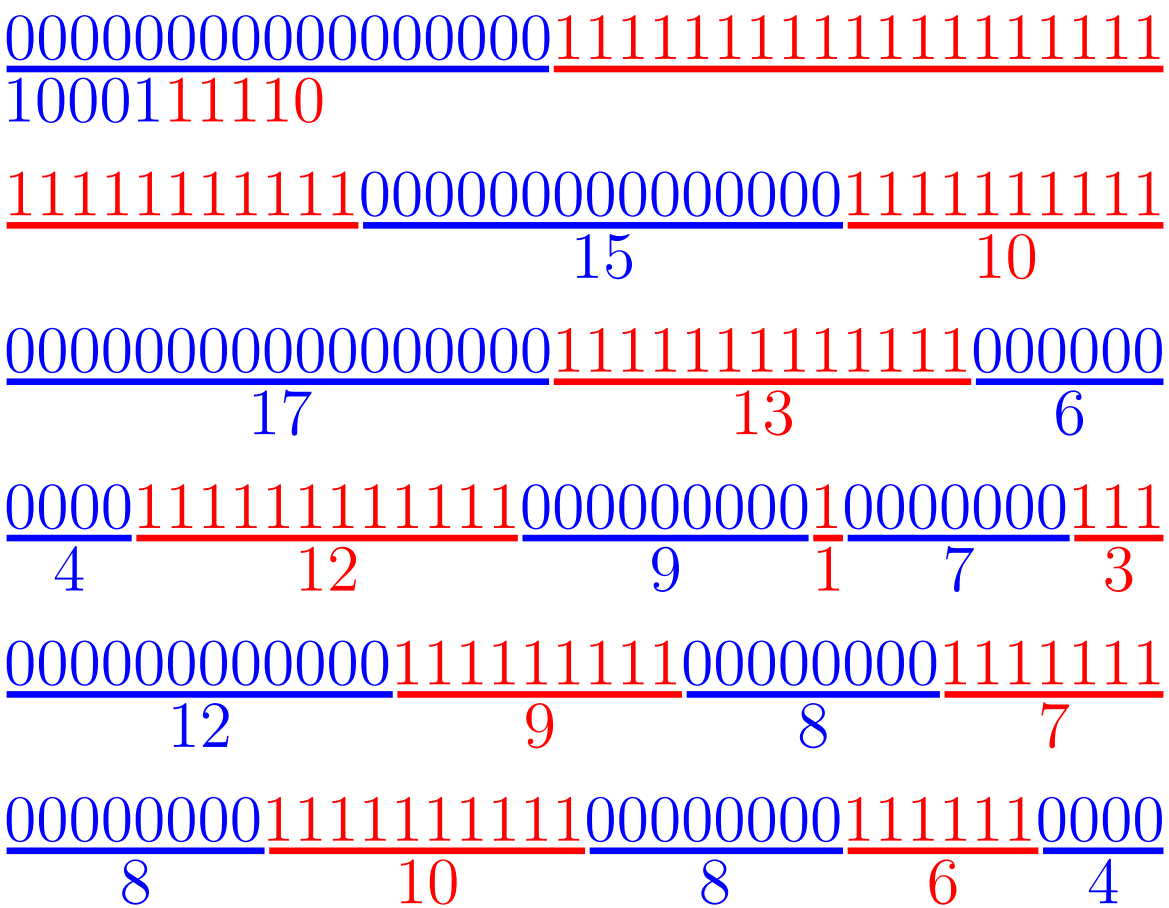
$$\underset{17}{\underline{0000000000000000}}\underset{13}{\underline{1111111111111}}\underset{6}{\underline{000000}}$$

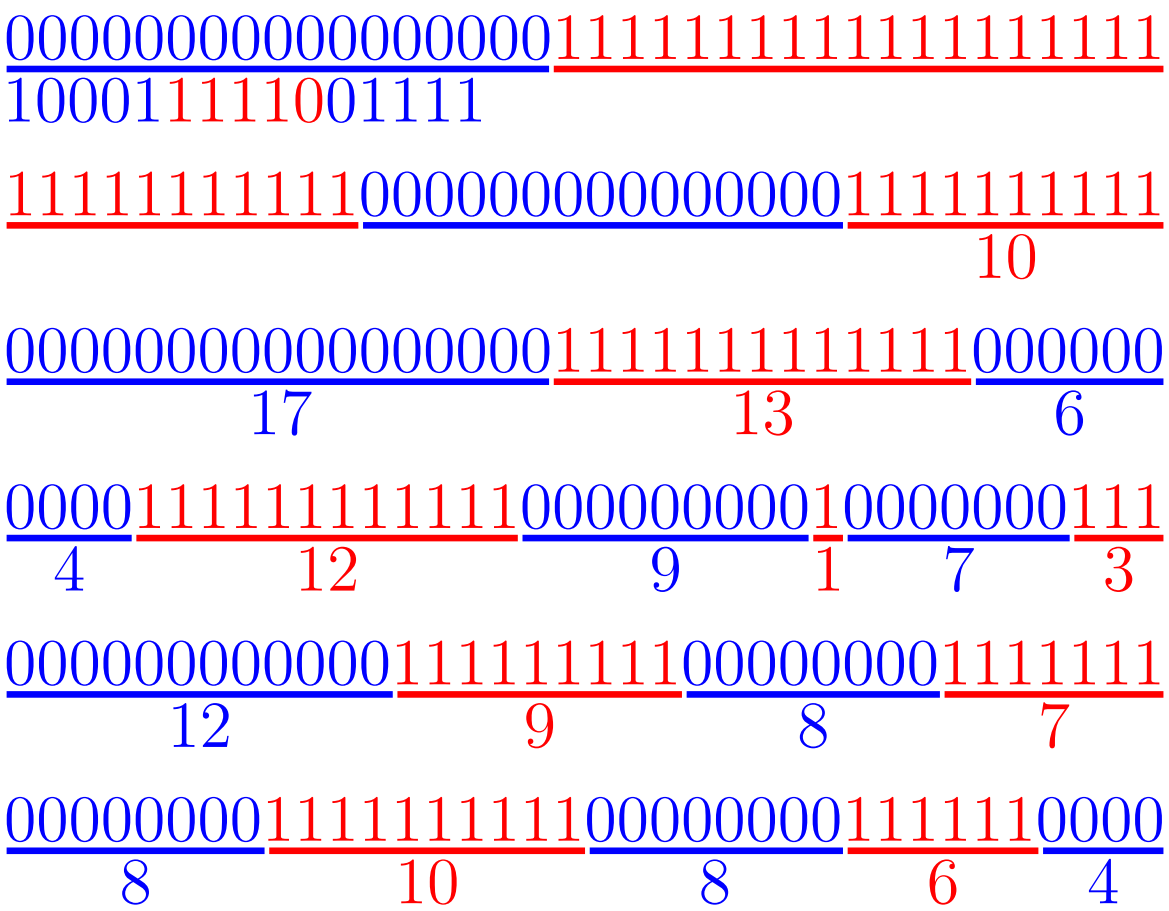
$$\underset{4}{\underline{0000}}\underset{12}{\underline{111111111111}}\underset{9}{\underline{000000000}}\underset{1}{\underline{1}}\underset{7}{\underline{0000000}}\underset{3}{\underline{111}}$$

$$\underset{12}{\underline{000000000000}}\underset{9}{\underline{111111111}}\underset{8}{\underline{00000000}}\underset{7}{\underline{1111111}}$$

$$\underset{8}{\underline{00000000}}\underset{10}{\underline{1111111111}}\underset{8}{\underline{00000000}}\underset{6}{\underline{111111}}\underset{4}{\underline{0000}}$$


0000000000000000 1111111111111111111
10001 19
11111111111 0000000000000000 1111111111
11 15 10
0000000000000000 1111111111111 000000
17 13 6
0000 111111111111 000000000 1 0000000 111
4 12 9 1 7 3
000000000000 111111111 00000000 1111111
12 9 8 7
00000000 111111111 00000000 111111 0000
8 10 8 6 4


0000000000000000 1111111111111111
1000111110
1111111111 0000000000000000 1111111111
15 10
0000000000000000 1111111111111 000000
17 13 6
0000 111111111111 000000000 1 0000000 111
4 12 9 1 7 3
000000000000 11111111 00000000 1111111
12 9 8 7
00000000 1111111111 00000000 111111 0000
8 10 8 6 4


0000000000000000 1111111111111111
1000111110011111
1111111111 0000000000000000 1111111111
10
0000000000000000 1111111111111 000000
17 13 6
0000 111111111111 000000000 1 0000000 111
4 12 9 1 7 3
000000000000 11111111 00000000 1111111
12 9 8 7
00000000 1111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
10001111100111101010

1111111111 0000000000000000 1111111111

0000000000000000 1111111111111 000000
17 13 6

0000 111111111111 000000000 1 0000000 111
4 12 9 1 7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 1111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001

1111111111100000000000000 1111111111

0000000000000000 1111111111111 000000
13 6

0000 111111111111 000000000 1 0000000 111
4 12 9 1 7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 1111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011 01

1111111111 0000000000000000 1111111111

0000000000000000 1111111111111 000000
6

0000 111111111111 000000000 1 0000000 111
4 12 9 1 7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 1111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011010 10

1111111111 0000000000000000 1111111111

0000000000000000 111111111111 000000

0000 111111111111 000000000 1 0000000 111
12 9 1 7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111111
10001111100111101010100010110101010 0

11111111111 0000000000000000 1111111111
1100

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111
9 1 7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111111
10001111100111101010100010110101010 0

11111111111 0000000000000000 1111111111
110001001

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111
1 7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111111
1000111110011110101010001011010101000

11111111111 0000000000000000 1111111111
1100010010000 1

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111
7 3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011010100

1111111111100000000000000 1111111111
1100010010000100111

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111
3

000000000000 11111111 00000000 1111111
12 9 8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011010100

1111111111100000000000000 1111111111
1100010010000100111000 11

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111

000000000000 111111111 00000000 1111111
12 9 8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111111
1000111110011110101010001011010101000

1111111111110000000000000001111111111
1100010010000100111000110110 0

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111

000000000000 11111111 00000000 1111111
9 8 7

00000000 111111111 10000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011010100

1111111111100000000000000011111111
1100010010000100111000110110001001

0000000000000000 111111111111 000000

0000 1111111111111 000000000 1 0000000 111

000000000000 11111111 00000000 1111111
8 7

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111111
1000111110011111010101000101101010100

11111111111 0000000000000000 1111111111
11000100100001001110001101100100101

0000000000000000 111111111111 000000
000

0000 1111111111111 000000000 1 0000000 111

000000000000 11111111 00000000 1111111
7

00000000 111111111 00000000 11111 10000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011010100

1111111111100000000000000011111111
1100010010000100111000110110001001 01

0000000000000000 111111111111 000000
00000111

0000 1111111111111 000000000 1 0000000 111

000000000000 11111111 00000000 1111111

00000000 111111111 00000000 111111 0000
8 10 8 6 4

0000000000000000 1111111111111111
1000111110011110101010001011010100

1111111111100000000000000011111111
1100010010000100111000110110001001 01

0000000000000000 111111111111 000000
0000011101000

0000 1111111111111 000000000 1 0000000 111

000000000000 11111111 00000000 1111111

00000000 111111111 00000000 111111 0000
10 8 6 4

0000000000000000111111111111111111
1000111110011110101010001011010100

1111111111100000000000000011111111
1100010010000100111000110110001001 01

0000000000000000111111111111000000
0000011101000010 10

0000111111111111000000001000000111

0000000000001111111110000000011111 11

0000000011111111110000000011111 10000
8 6 4

0000000000000000111111111111111111
1000111110011110101010001011010101000

1111111111100000000000000011111111111
1100010010000100111000110110001001001

0000000000000000111111111111000000
0000011101000010100100 0

0000111111111111000000001000000111

000000000000111111111000000001111111

00000000111111111100000000111111000 0
6 4

0000000000000000111111111111111111
1000111110011110101010001011010101000

1111111111100000000000000011111111111
1100010010000100111000110110001001010

0000000000000000111111111111000000
0000011101000010100100000110

0000111111111111000000001000000111

000000000000111111111000000001111111

000000001111111111000000001111110000
4

0000000000000000111111111111111111
1000111110011111010101000101101010100

1111111111100000000000000011111111111
1100010010000100111000110110001001001

0000000000000000111111111111000000
0000011101000010100100000110001 00

0000111111111111000000001000000111

000000000000111111111000000001111111

000000001111111111000000001111110000

00000000000000001111111111111111111
11111111111000000000000001111111111
00000000000000001111111111111000000
00001111111111111000000001000000111
00000000000011111111100000000111111
00000000111111111100000000111110000

10001111100111101010100010110101010 0
11000100100001001110001101100100101
00000111010000101001000011000100

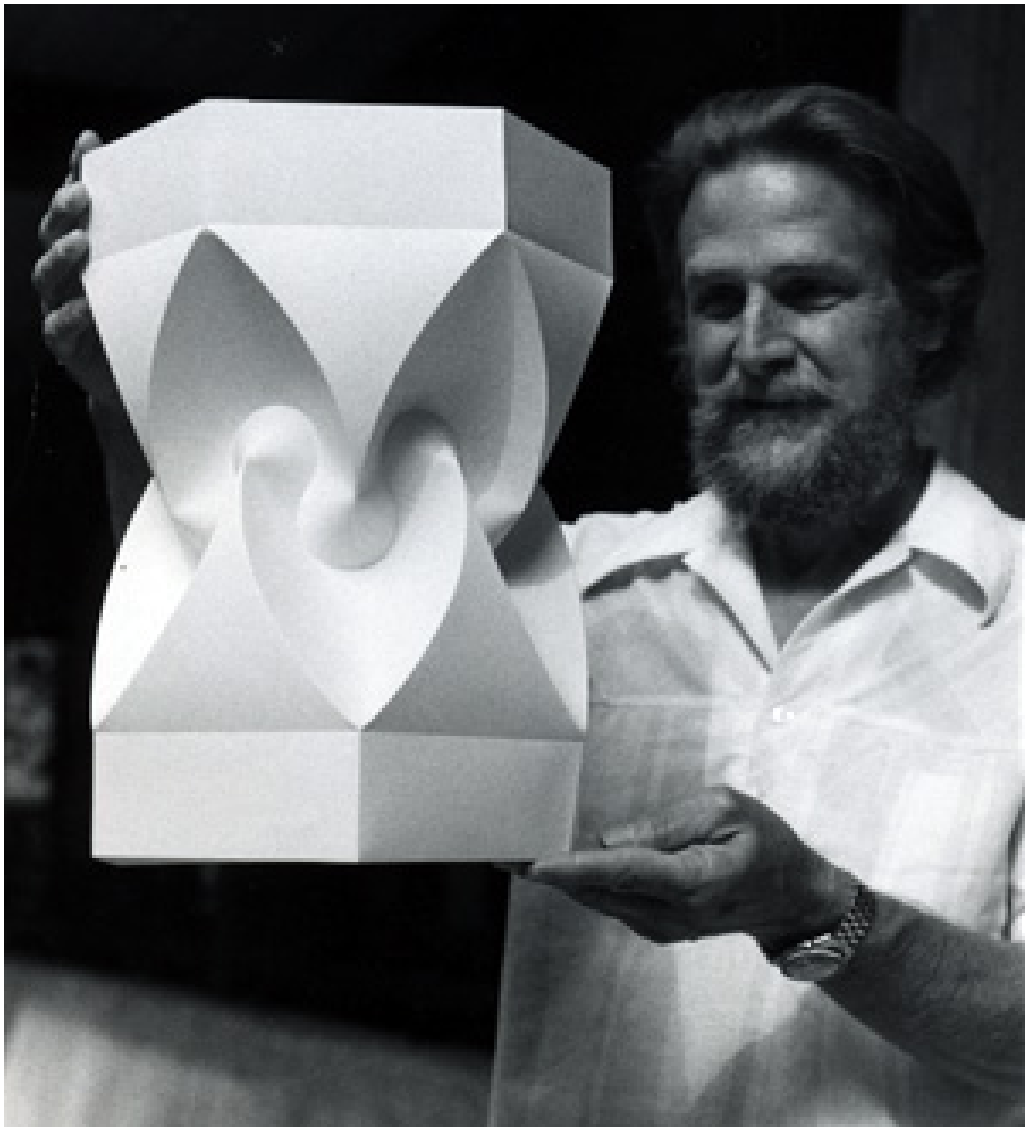
00000000000000001111111111111111111
11111111111000000000000001111111111
00000000000000001111111111111000000
00001111111111111000000001000000111
00000000000011111111100000000111111
00000000111111111100000000111110000

01011000111110011110101010001011010
10011000100100001001110001101100010
10100000111010000101001000011000100


MP3


DIVX

Sujet des Mines 2015...hors programme...Codage de Huffman



David Huffman (1925 - 1999)

Principe de l'algorithme de Huffman

Principe

Principe de l'algorithme de Huffman

Principe

- Les symboles (e.g octets) sont recodés avec des tailles variables : plus un symbole est fréquent, plus le mot de code le représentant sera court

Principe de l'algorithme de Huffman

Principe

- Les symboles (e.g octets) sont recodés avec des tailles variables : plus un symbole est fréquent, plus le mot de code le représentant sera court
- Pour permettre la décompression, une table (en fait un arbre) de correspondance entre les symboles et les mots de code correspondants est transmise avec la donnée compressée

Principe de l'algorithme de Huffman

Principe

- Les symboles (e.g octets) sont recodés avec des tailles variables : plus un symbole est fréquent, plus le mot de code le représentant sera court
- Pour permettre la décompression, une table (en fait un arbre) de correspondance entre les symboles et les mots de code correspondants est transmise avec la donnée compressée

Principe de l'algorithme de Huffman

Principe

- Les symboles (e.g octets) sont recodés avec des tailles variables : plus un symbole est fréquent, plus le mot de code le représentant sera court
- Pour permettre la décompression, une table (en fait un arbre) de correspondance entre les symboles et les mots de code correspondants est transmise avec la donnée compressée

Apport principal de Huffman (1952)

Une manière judicieuse de choisir quel mot de code associer à chaque symbole

Caractéristiques de l'algorithme de Huffman

Un code préfixe (quel intérêt ?)

Aucun mot de code n'est le préfixe (le début) d'un autre mot de code.

Caractéristiques de l'algorithme de Huffman

Un code préfixe (quel intérêt ?)

Aucun mot de code n'est le préfixe (le début) d'un autre mot de code.

Code préfixe ou non pour trois caractères

	a	b	c
Préfixe	0	10	11
Non préfixe	0	1	11

Caractéristiques de l'algorithme de Huffman

Un code préfixe (quel intérêt ?)

Aucun mot de code n'est le préfixe (le début) d'un autre mot de code.

Code préfixe ou non pour trois caractères

	a	b	c
Préfixe	0	10	11
Non préfixe	0	1	11

Caractéristiques de l'algorithme de Huffman

Un code préfixe (quel intérêt ?)

Aucun mot de code n'est le préfixe (le début) d'un autre mot de code.

Code préfixe ou non pour trois caractères

	a	b	c
Préfixe	0	10	11
Non préfixe	0	1	11

Un code optimal

En encodant les symboles séparément on ne peut pas compresser plus que par l'algorithme de Huffman !

L'algorithme de Huffman en trois étapes

Étape 1 : fréquence des symboles

On calcule la fréquence de chaque symbole de la donnée à compresser

L'algorithme de Huffman en trois étapes

Étape 1 : fréquence des symboles

On calcule la fréquence de chaque symbole de la donnée à compresser

Exemple

J'erre près des berges de l'Elster.

x	J	'	e	r	sp	p	è	s	d	b	g	l	E	t	.
f_x	1	2	7	5	5	1	1	4	2	1	1	2	1	1	1

L'algorithme de Huffman en trois étapes

Étape 1 : fréquence des symboles

On calcule la fréquence de chaque symbole de la donnée à compresser

Exemple

J'erre près des berges de l'Elster.

x	J	p	è	b	g	E	t	.	'	d	l	s	r	sp	e
f_x	1	1	1	1	1	1	1	1	2	2	2	4	5	5	7

L'algorithme de Huffman en trois étapes

Étape 1 : fréquence des symboles

On calcule la fréquence de chaque symbole de la donnée à compresser

Exemple

J'erre près des berges de l'Elster.

x	J	p	è	b	g	E	t	.	'	d	l	s	r	sp	e
f_x	1	1	1	1	1	1	1	1	2	2	2	4	5	5	7

Remarque

On peut utiliser des tables de fréquences pré-existantes, mais le code n'est alors plus optimal

UNIVERSITÉ DE NANTES

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

Construction de l'arbre

tant que il reste plus d'un symbole
 choisir x minimisant f_x
 choisir $y \neq x$ minimisant f_y
 détruire x et y
 créer $x\hat{y}$ de fréquence $f_x + f_y$
fin tant que
retourner le symbole restant

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

Construction de l'arbre

tant que il reste plus d'un symbole
 choisir x minimisant f_x
 choisir $y \neq x$ minimisant f_y
 détruire x et y
 créer $x\hat{y}$ de fréquence $f_x + f_y$
fin tant que
retourner le symbole restant

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

Construction de l'arbre

tant que il reste plus d'un symbole
 choisir x minimisant f_x
 choisir $y \neq x$ minimisant f_y
 détruire x et y
 créer $x\hat{y}$ de fréquence $f_x + f_y$
fin tant que
retourner le symbole restant

Exemple

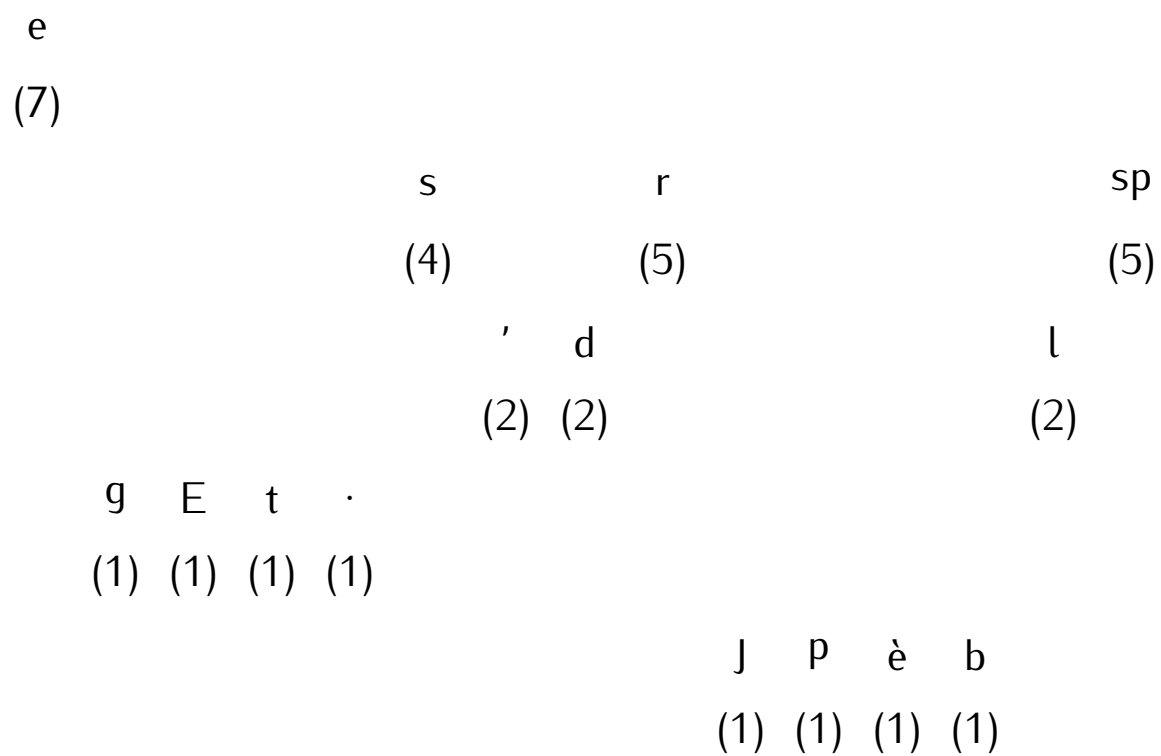
J'erre près des berges de l'Elster.

x	J	p	è	b	g	E	t
f_x	1	1	1	1	1	1	1

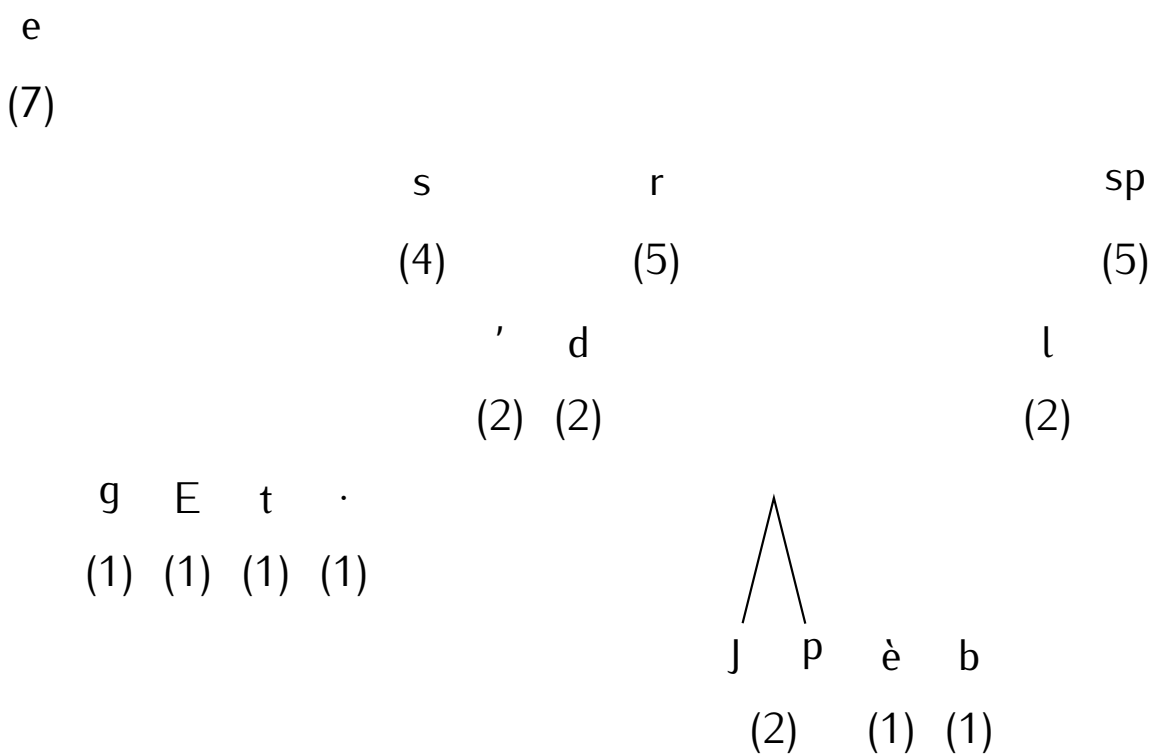
.	'	d	l	s	r	sp	e
1	2	2	2	4	5	5	7

UNIVERSITÉ DE NANTES

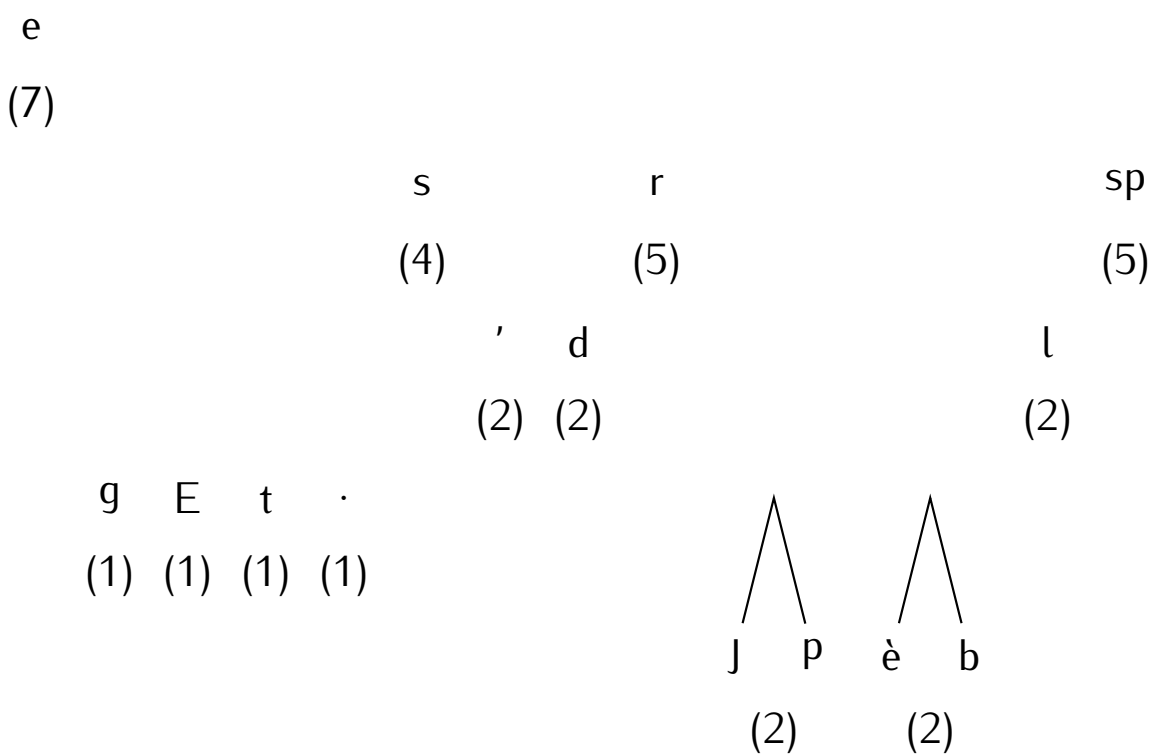
Construction de l'arbre



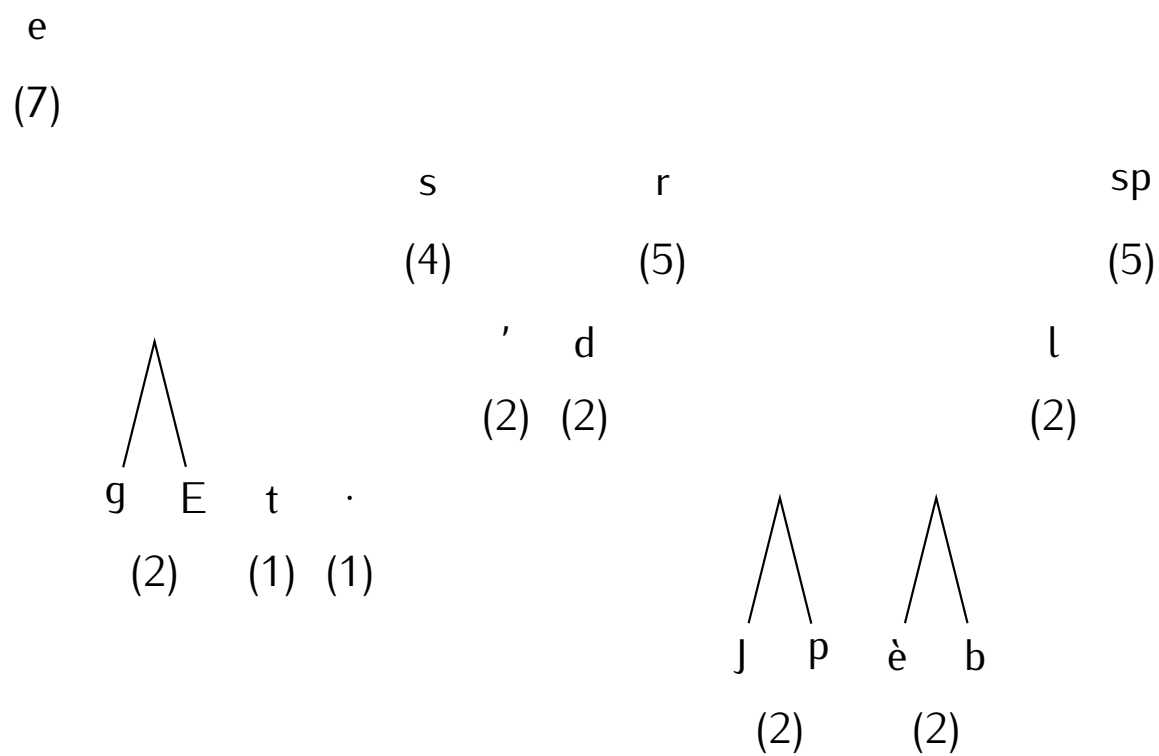
Construction de l'arbre



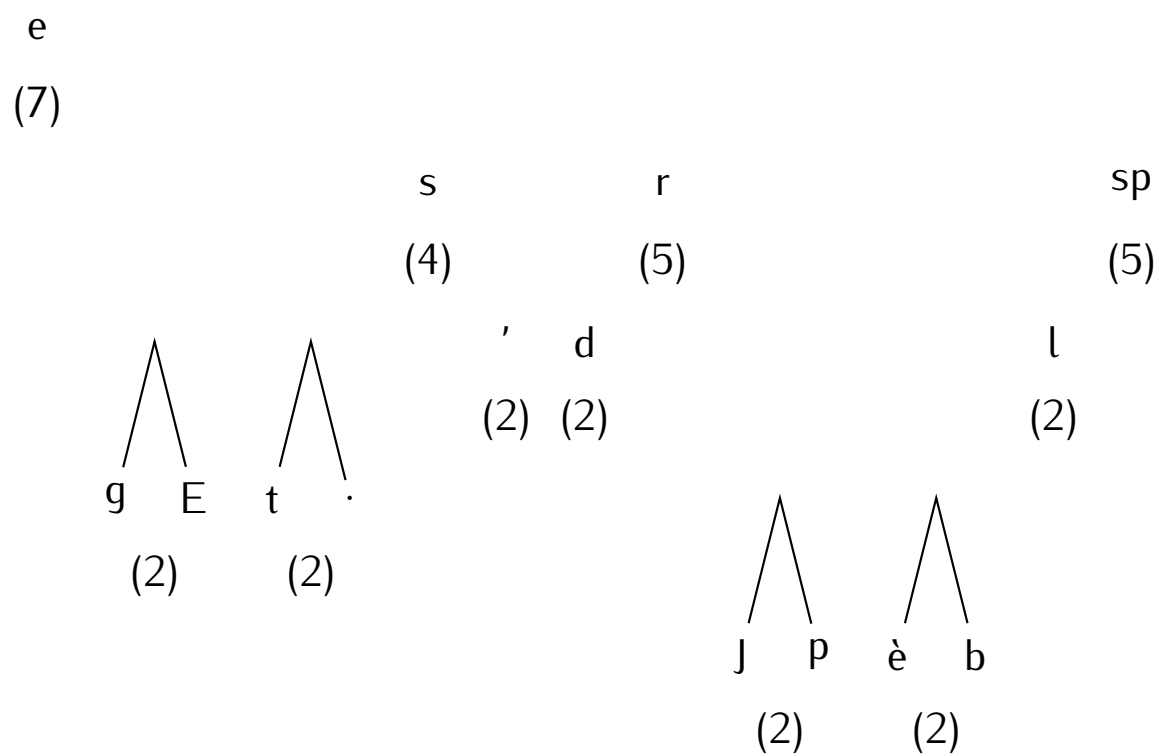
Construction de l'arbre



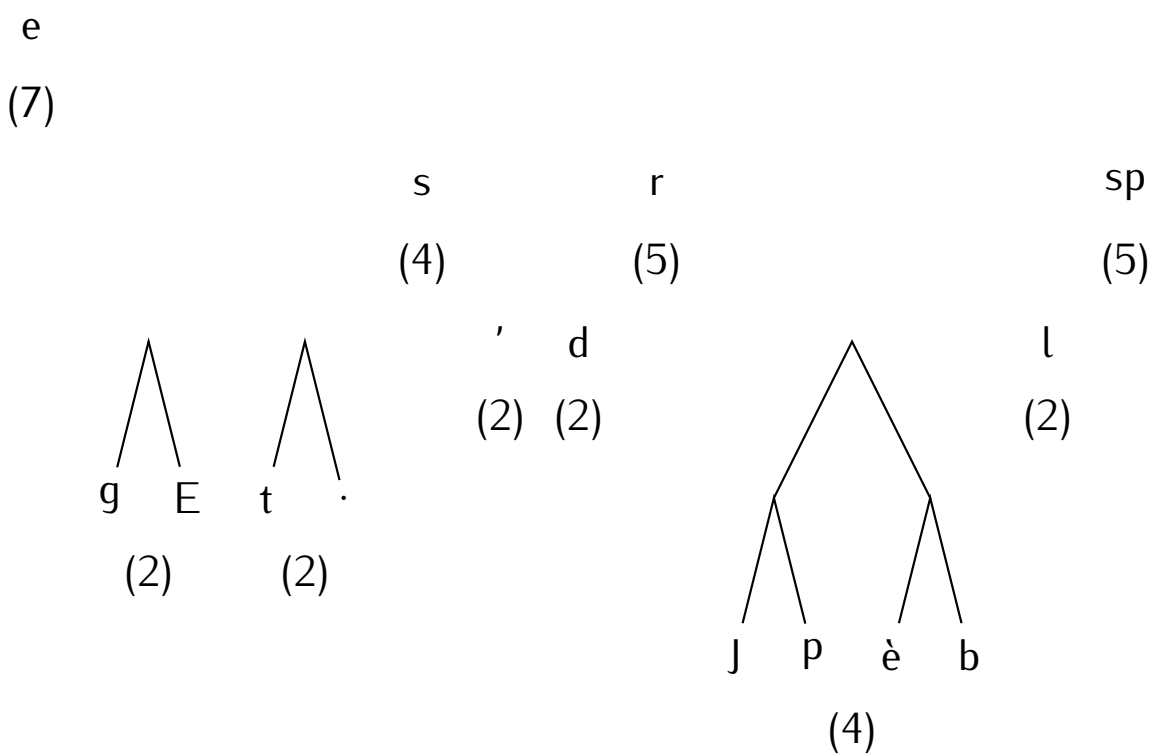
Construction de l'arbre



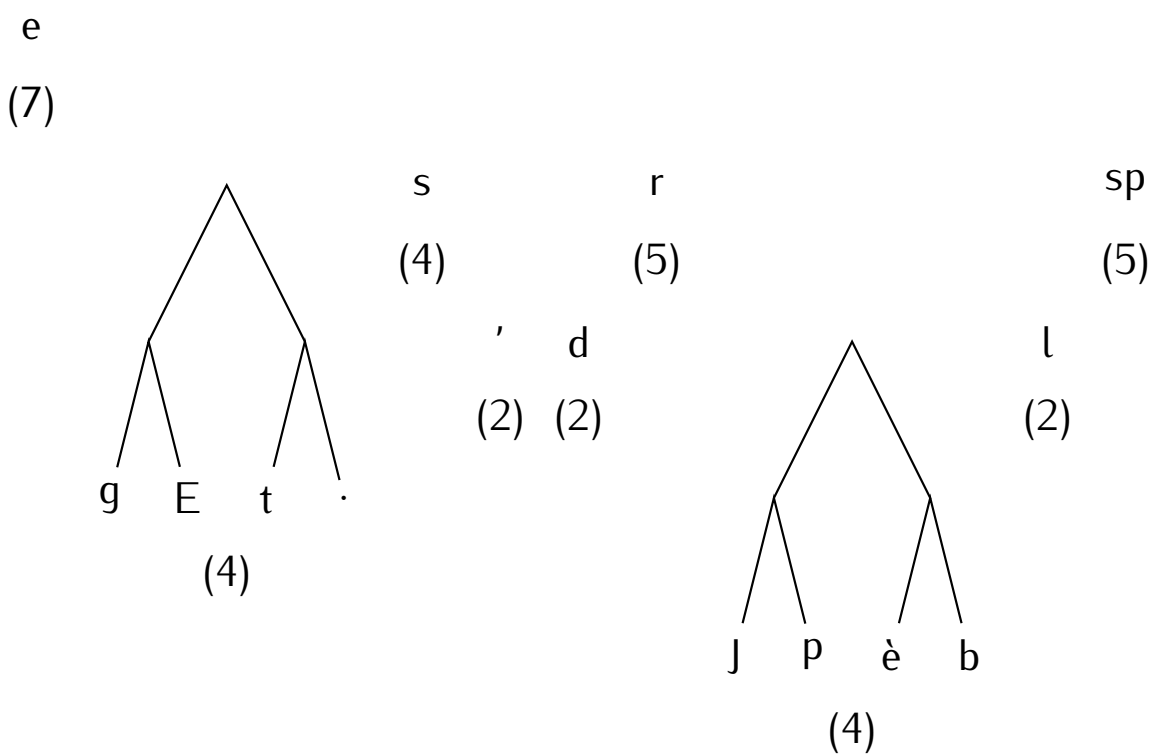
Construction de l'arbre



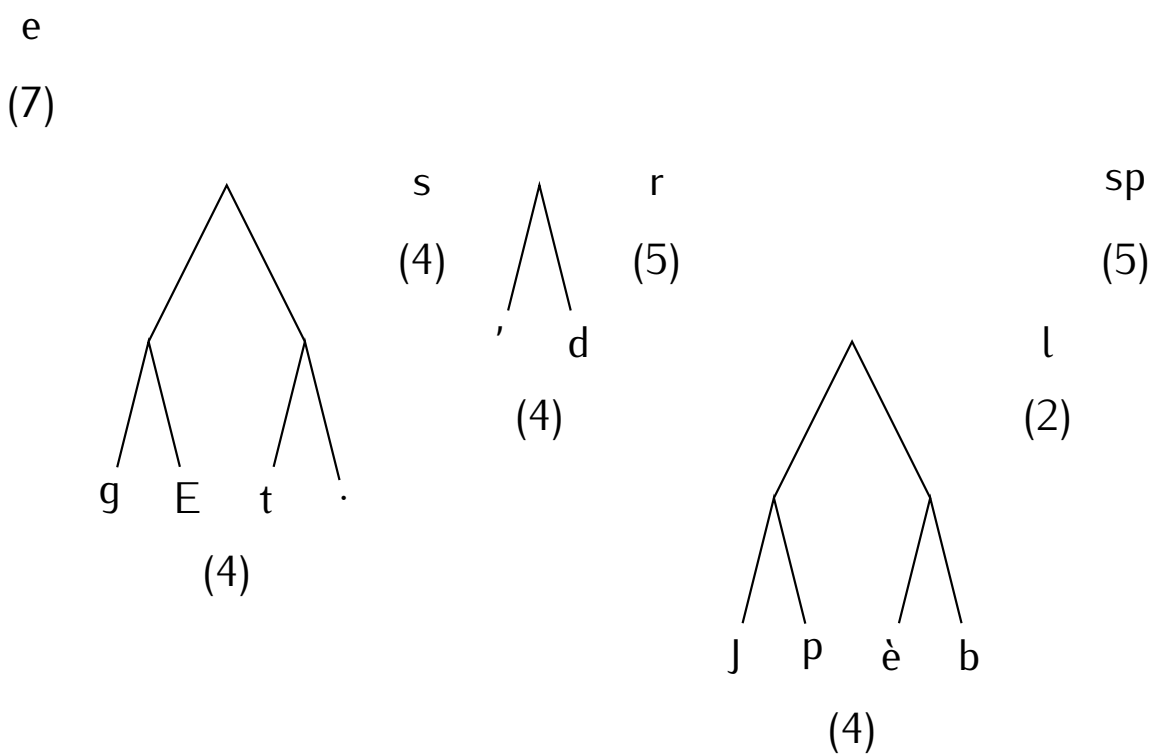
Construction de l'arbre



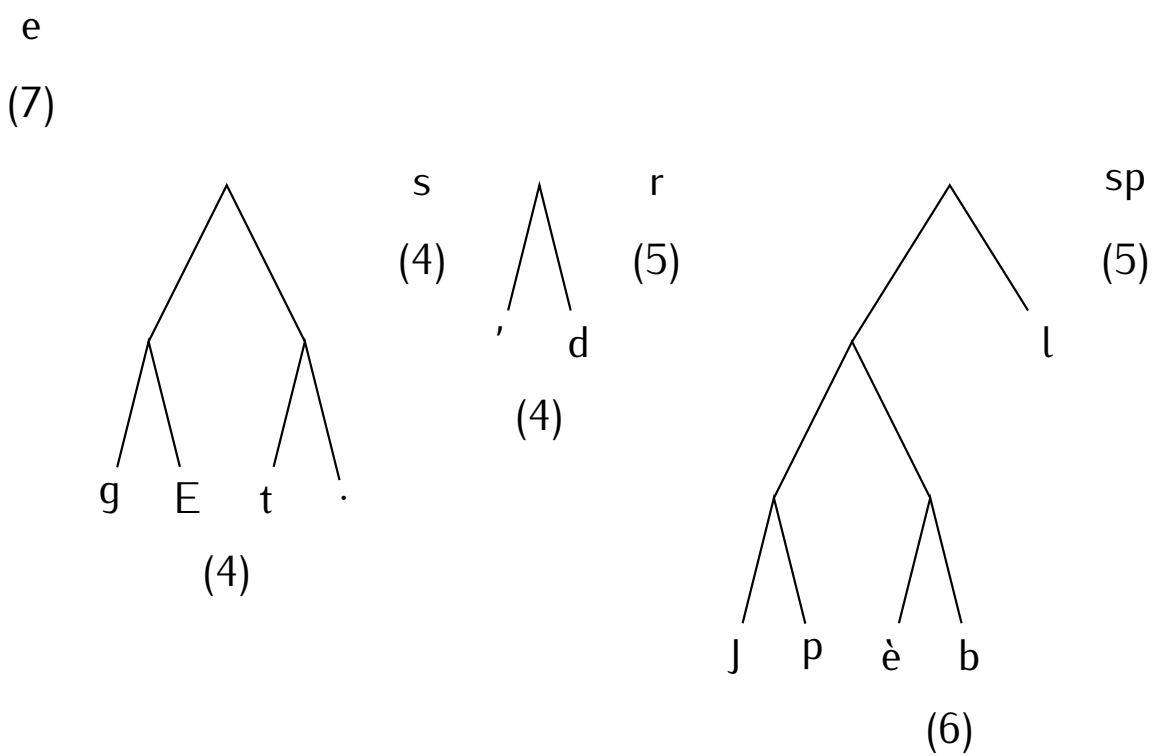
Construction de l'arbre



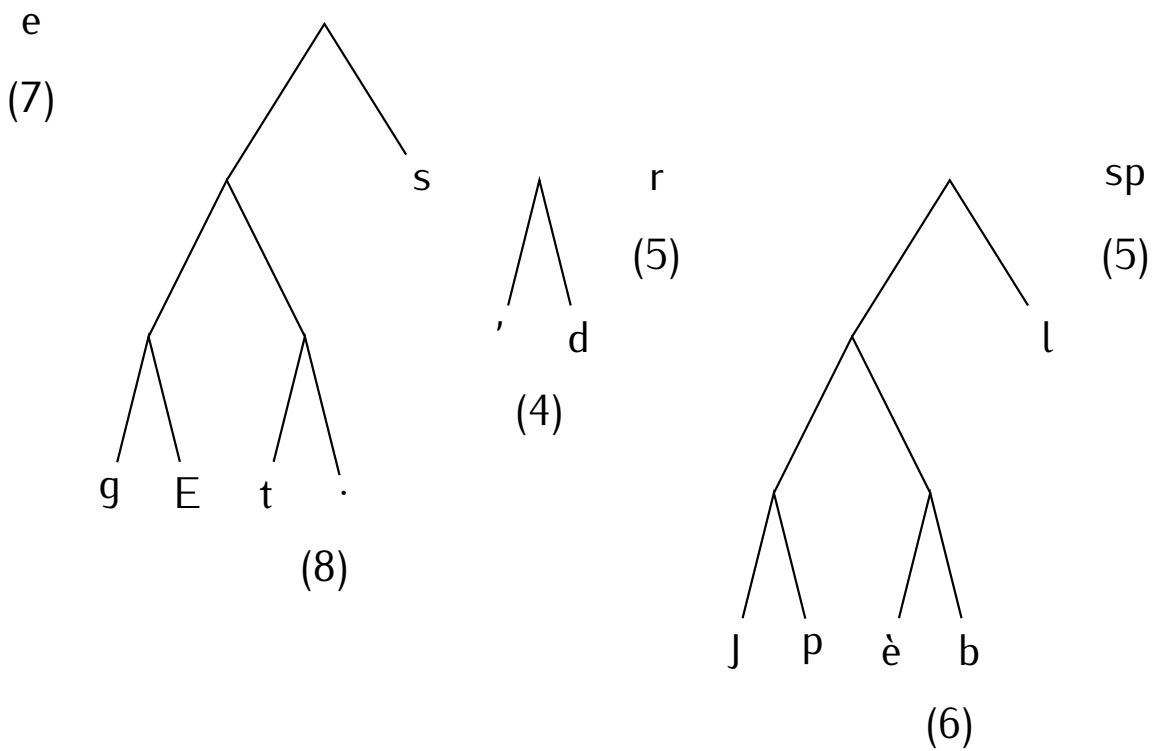
Construction de l'arbre



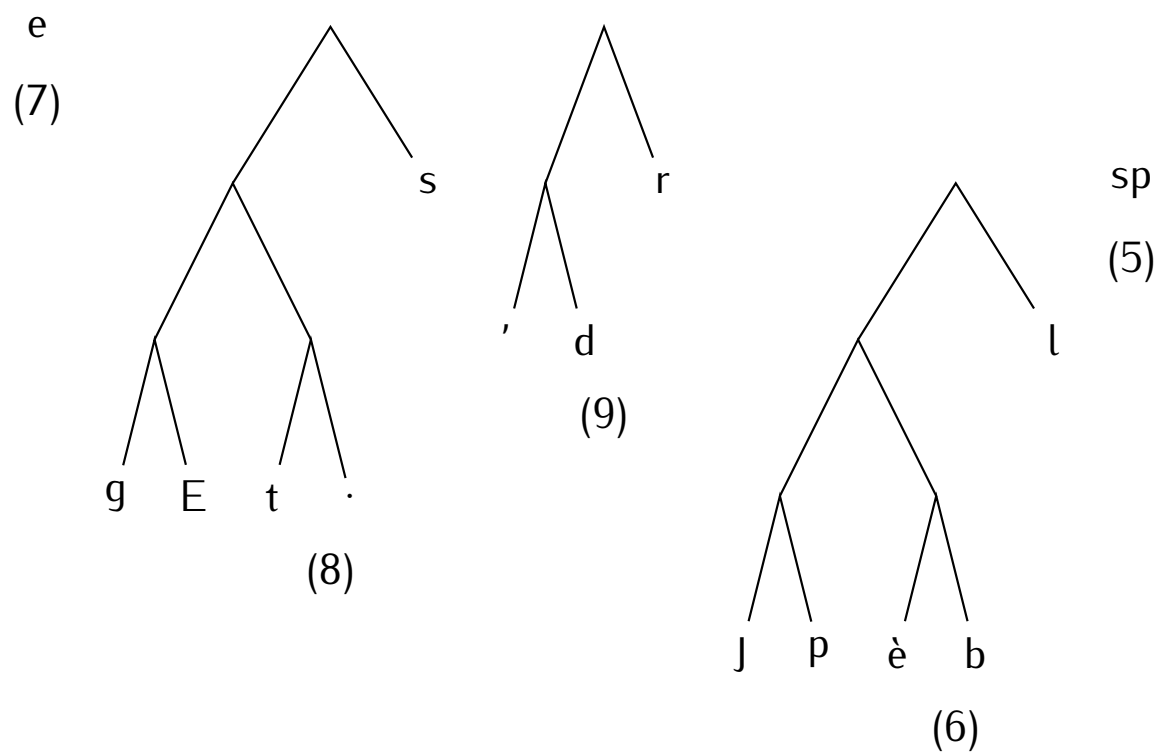
Construction de l'arbre



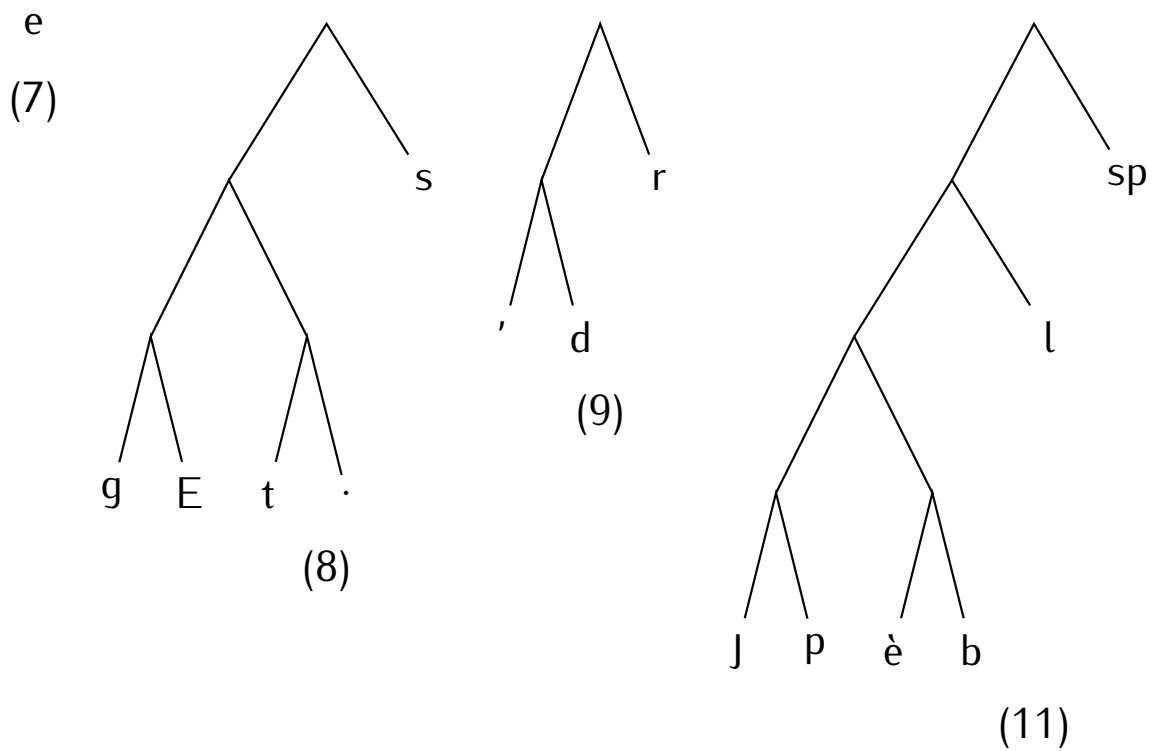
Construction de l'arbre



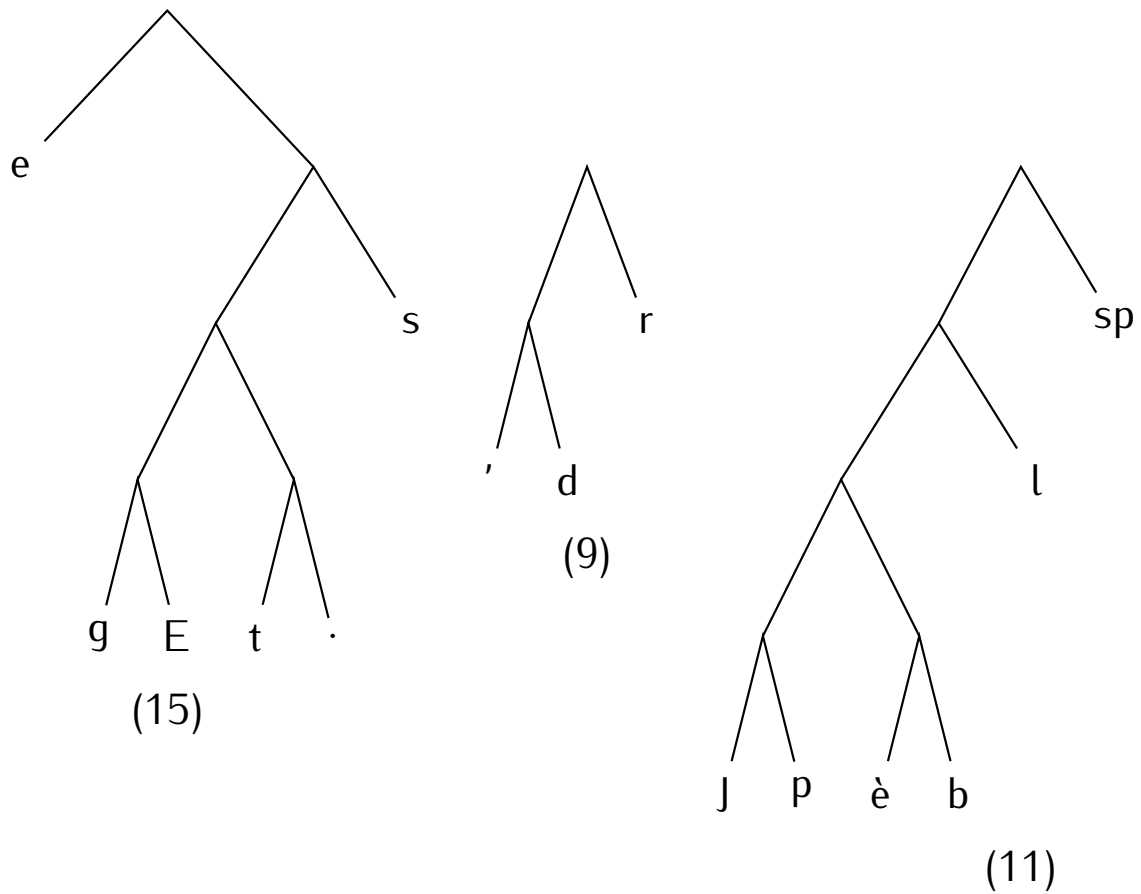
Construction de l'arbre



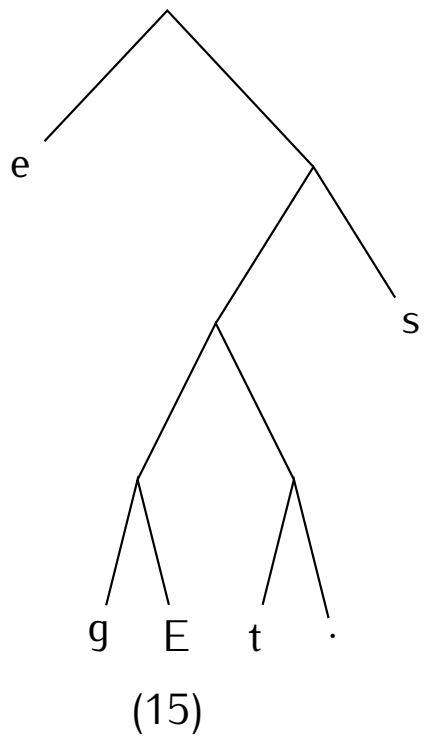
Construction de l'arbre



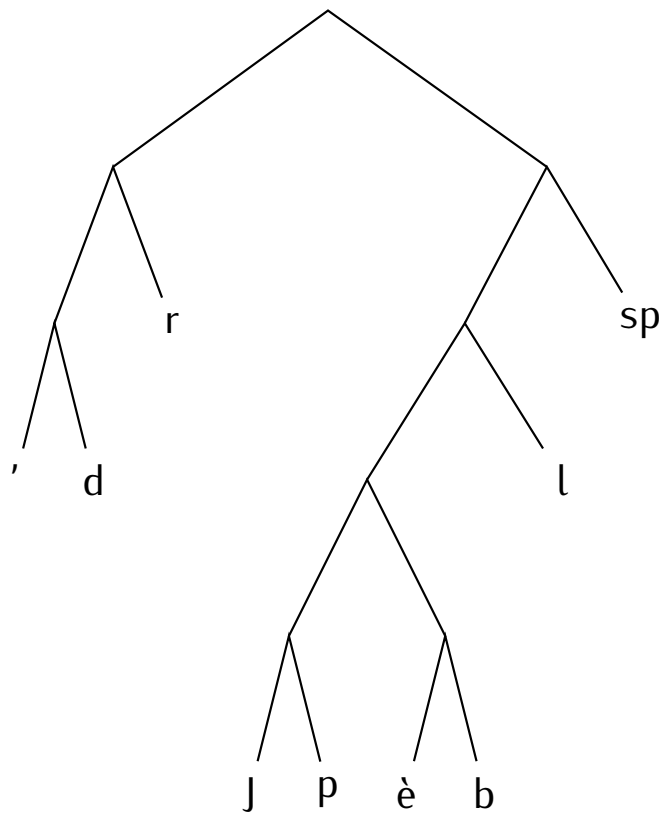
Construction de l'arbre



Construction de l'arbre

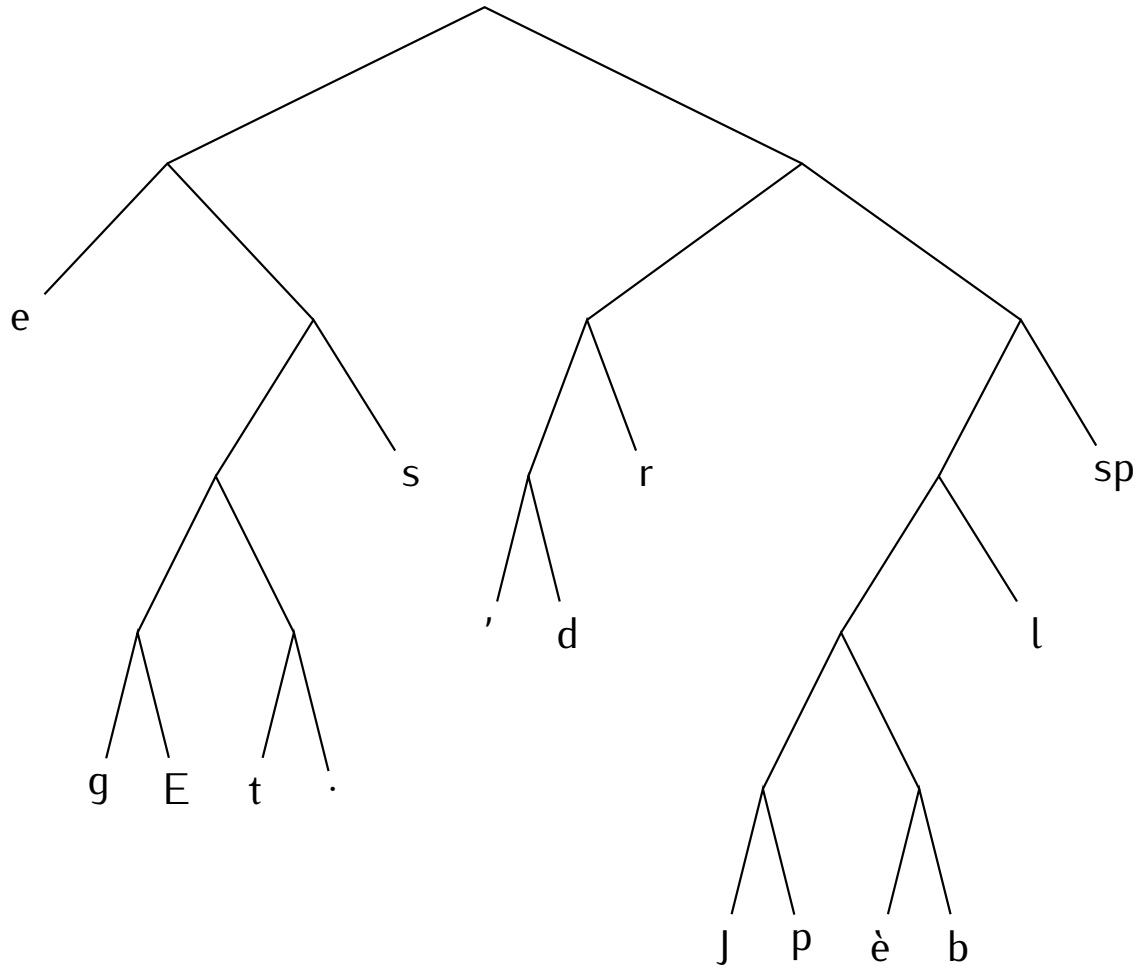


(15)



(20)

Construction de l'arbre



(35)

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

Construction de l'arbre

tant que il reste plus d'un symbole
 choisir x minimisant f_x
 choisir $y \neq x$ minimisant f_y
 détruire x et y
 créer $x\hat{y}$ de fréquence $f_x + f_y$
fin tant que
retourner le symbole restant

L'algorithme de Huffman en trois étapes

Étape 2

On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

Construction de l'arbre

tant que il reste plus d'un symbole
 choisir x minimisant f_x
 choisir $y \neq x$ minimisant f_y
 détruire x et y
 créer $x\hat{y}$ de fréquence $f_x + f_y$
fin tant que
retourner le symbole restant

L'algorithme de Huffman en trois étapes

Étape 2

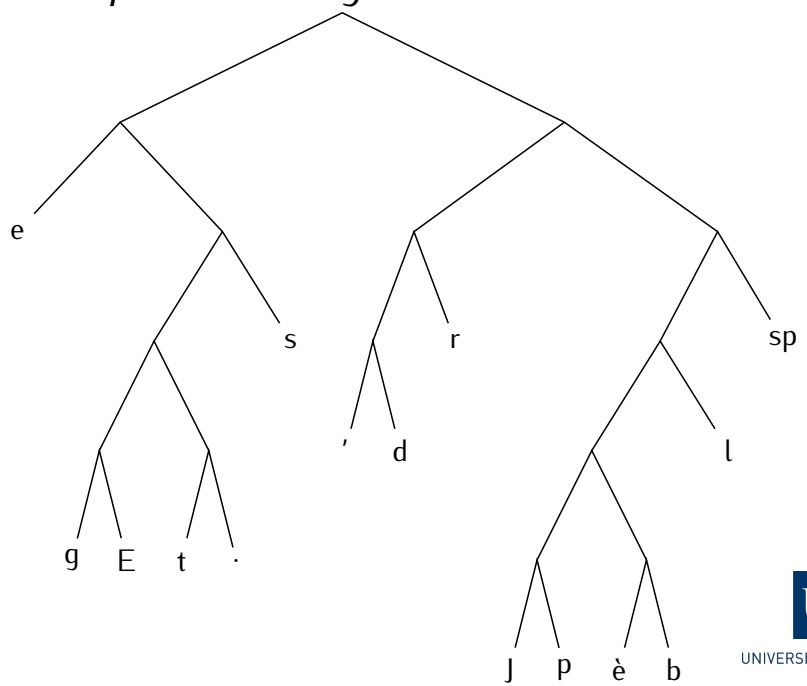
On construit un arbre à partir des fréquences, qui déterminera le mot de code correspondant à chaque symbole

Construction de l'arbre

tant que il reste plus d'un symbole
 choisir x minimisant f_x
 choisir $y \neq x$ minimisant f_y
 détruire x et y
 créer $x\hat{y}$ de fréquence $f_x + f_y$
fin tant que
retourner le symbole restant

Exemple

J'erre près des berges de l'Elster.



UNIVERSITÉ DE NANTES

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

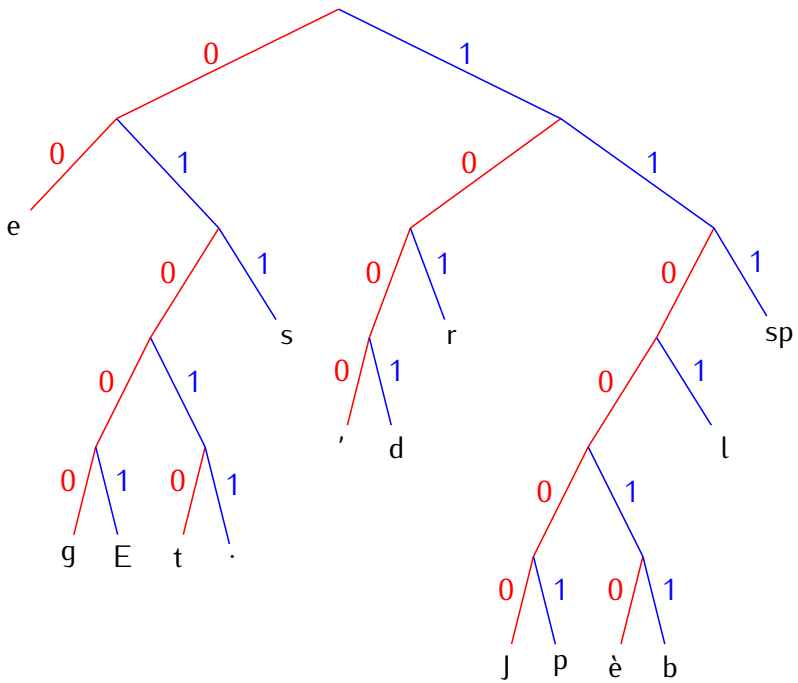
Exemple

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



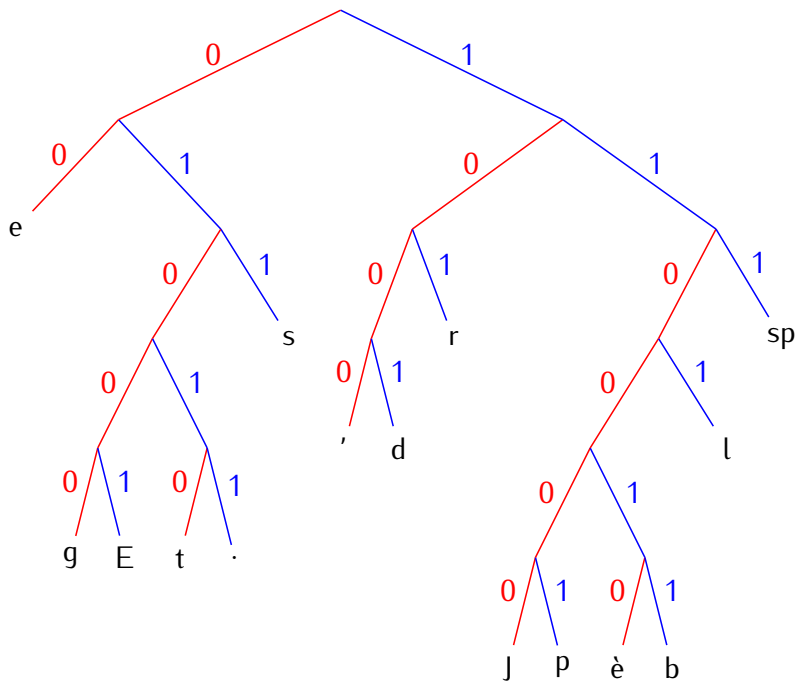
UNIVERSITÉ DE NANTES

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



J'erre près des berges de l'Elster.

J : 110000, ' : 1000, e : 00,
r : 101, sp : 111,

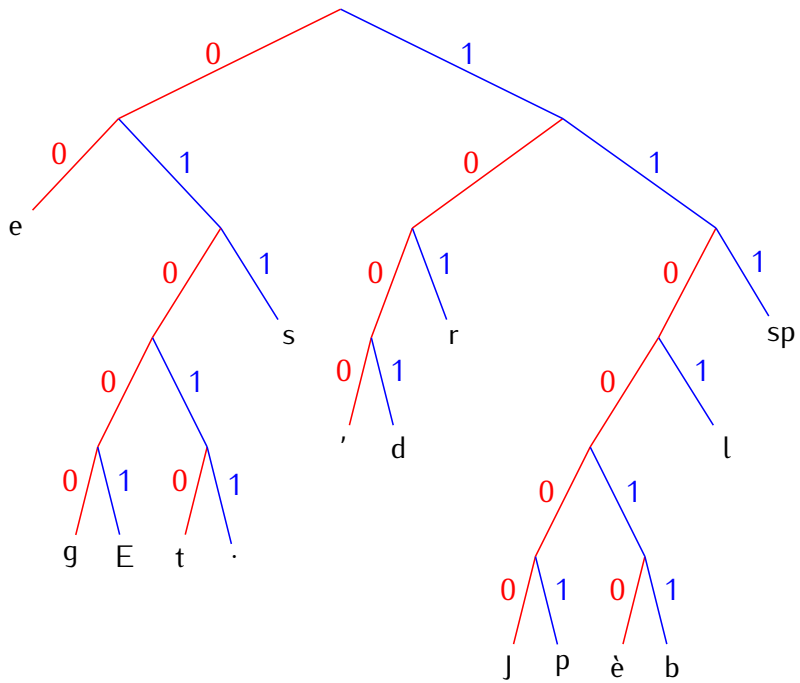
110000100000101101

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



J'erre près des berges de l'Elster.

J : 110000, ' : 1000, e : 00,
r : 101, sp : 111, p : 110001,

110000100000101101 00

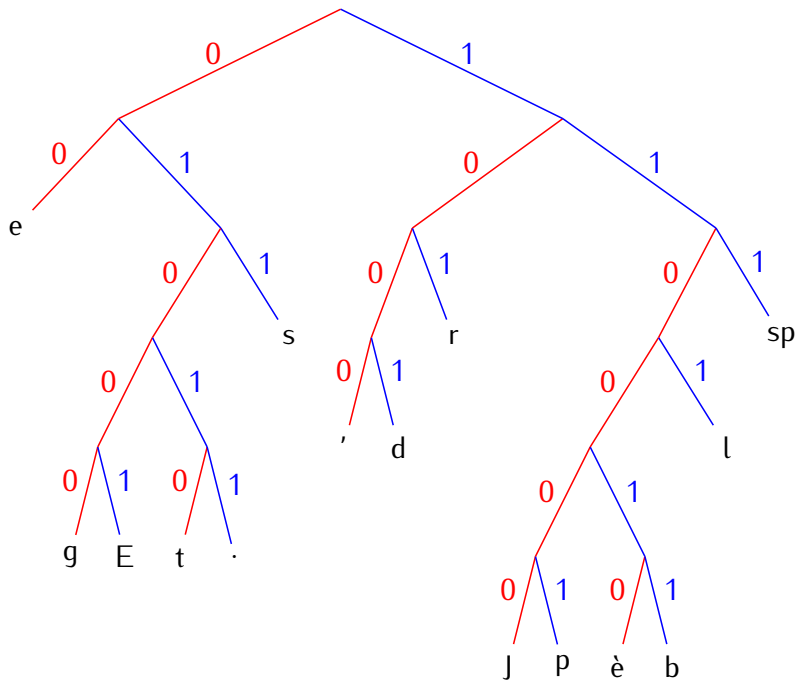
UNIVERSITÉ DE NANTES

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



J'erre près des berges de l'Elster.

J : 110000, ' : 1000, e : 00,
r : 101, sp : 111, p : 110001, etc

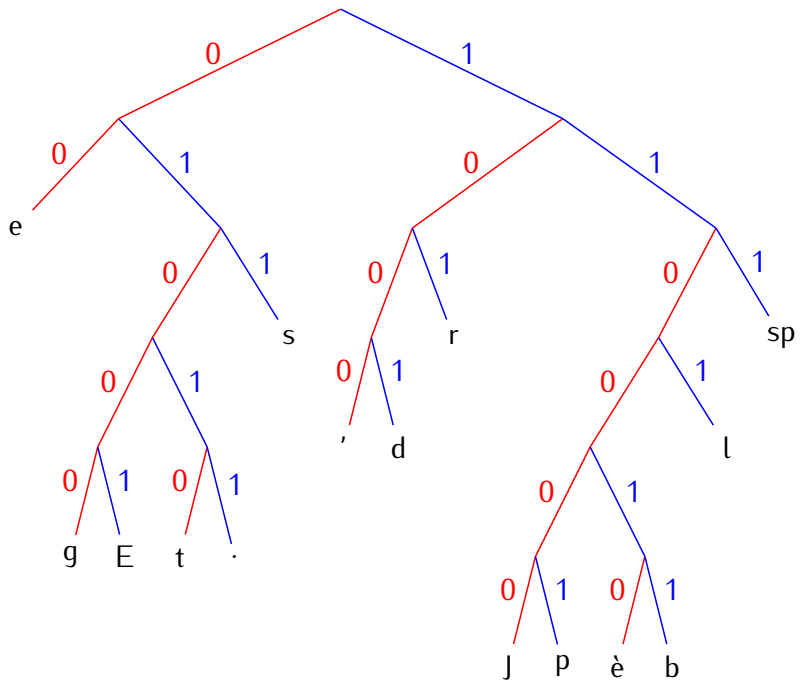
110000100000101101001111

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



J'erre près des berges de l'Elster.

J : 110000, ' : 1000, e : 00,
r : 101, sp : 111, p : 110001, etc

11000010000010110100111110001

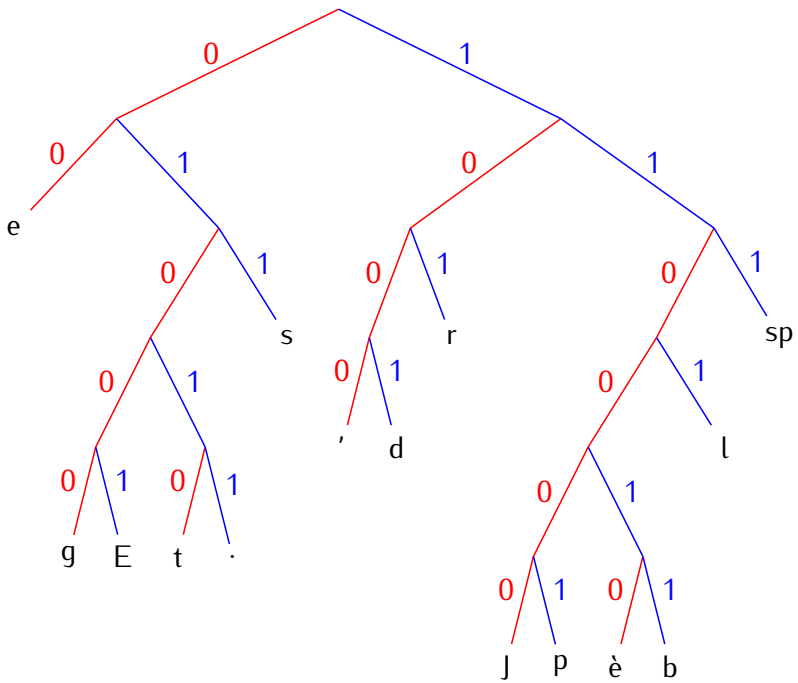
UNIVERSITÉ DE NANTES

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



J'erre près des berges de l'Elster.

J : 110000, ' : 1000, e : 00,
r : 101, sp : 111, p : 110001, etc

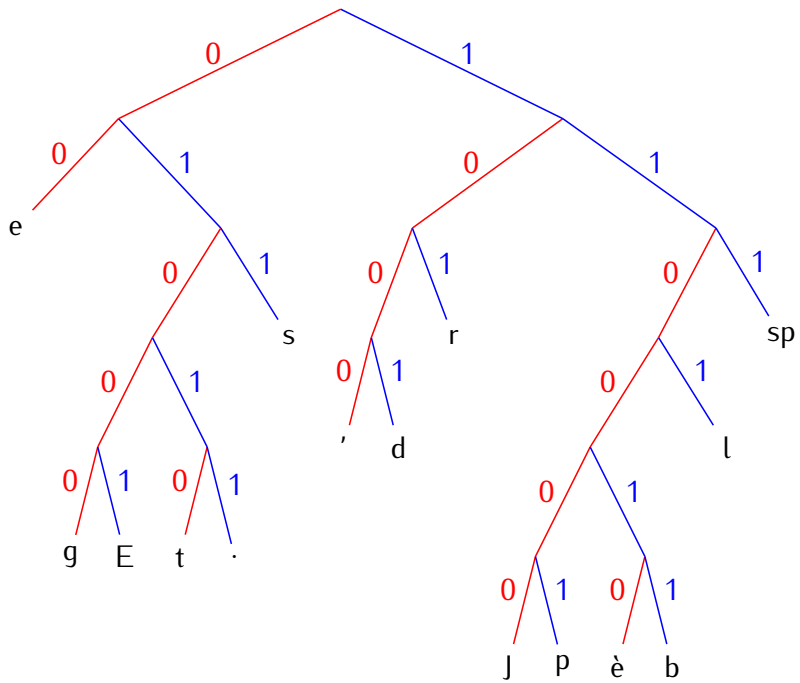
1100001000001011010011111000110111001001111111001000111111100110010101000000111111001001111101100001001110101101010001010101 1

L'algorithme de Huffman en trois étapes

Étape 3

On utilise l'arbre pour coder, la branche qui mène à un symbole donne directement le mot de code pour ce symbole

Exemple



J'erre près des berges de l'Elster.

110000100000101101001111100011011100100111111100100011111111001100101010000001111110010011111010110101000010101011

124 bits pour 35 symboles dont 15 différents

UNIVERSITÉ DE NANTES

Décodage

Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Décodage

Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

Décodage

Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

Décodage

Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111001001111111001000111111100110010101000000011111100100111110000100111010110101000010101011

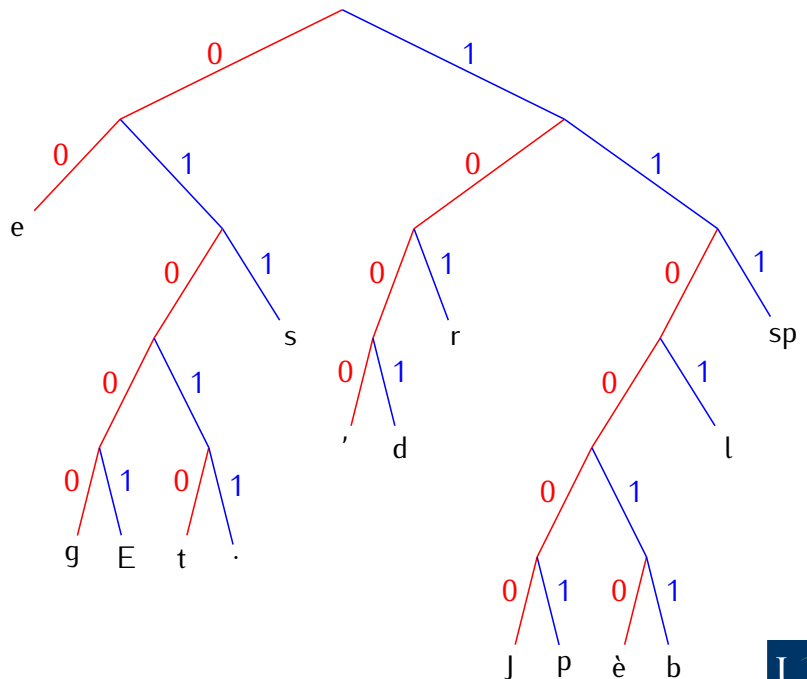
Décodage

Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111001001111111001000111111100100110010101000000111111001001111101100001001110101101010001010101011



UNIVERSITÉ DE NANTES

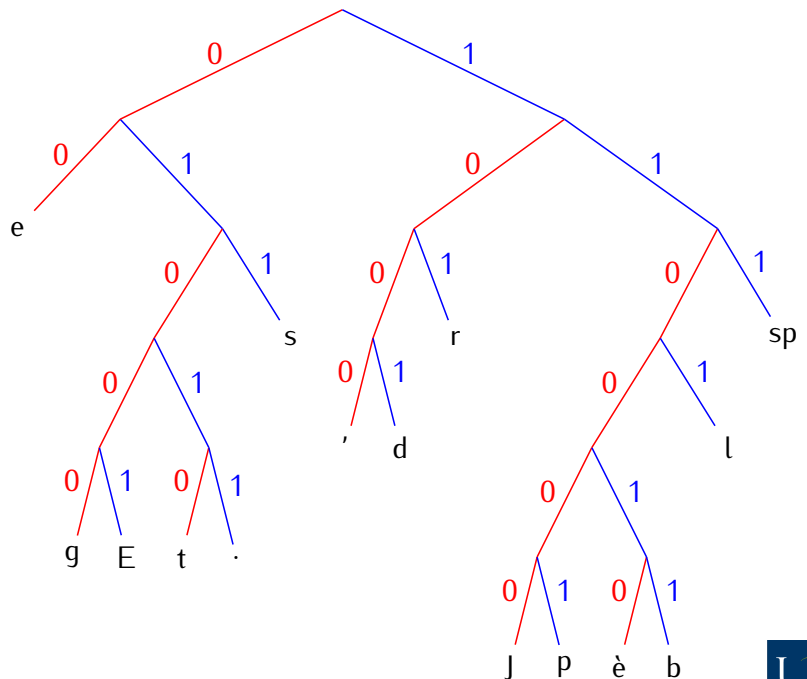
Décodage

Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111001001111111001000111111100100110010101000000111111001001111101100001001110101101010001010101011



UNIVERSITÉ DE NANTES

Décodage

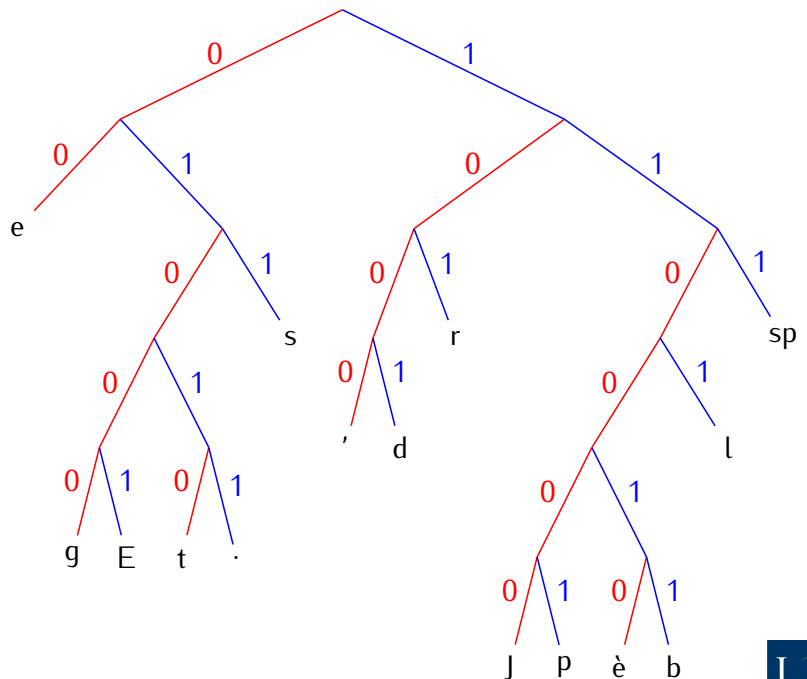
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100101010000001111111001001111101100001001110101101010001010101011

J



Décodage

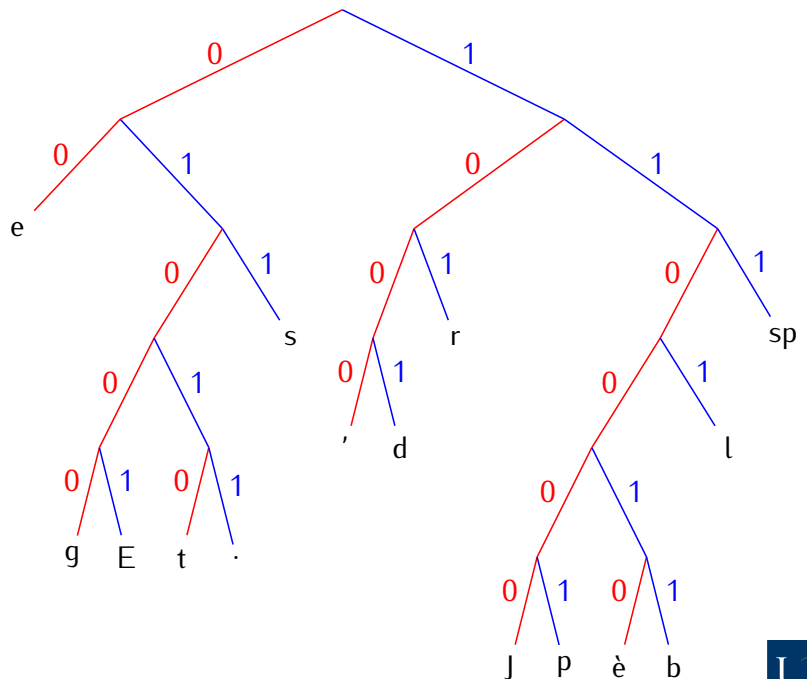
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111001001111111001000111111100110010101000000111111001001111101100001001110101101010001010101011

J



UNIVERSITÉ DE NANTES

Décodage

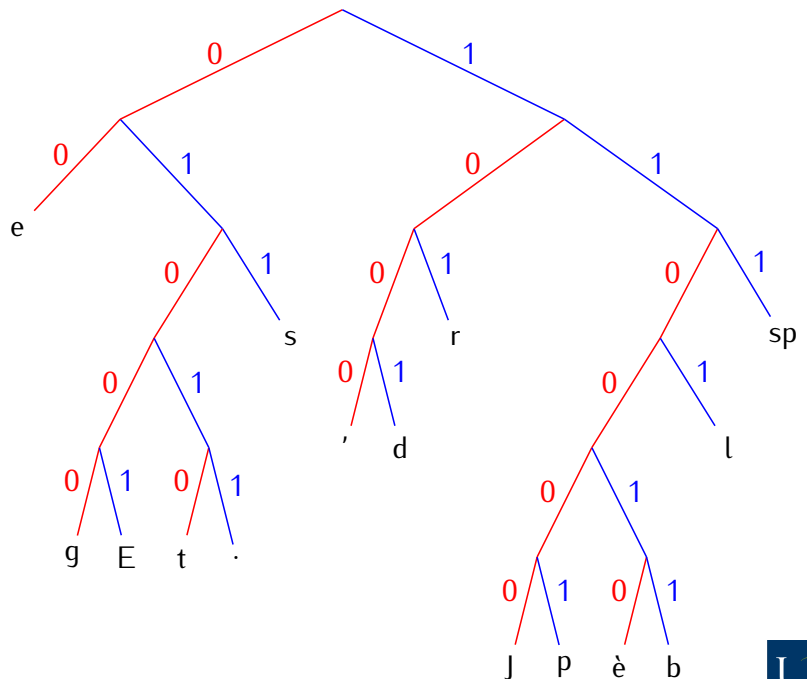
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111001001111111001000111111100100110010101000000111111001001111101100001001110101101010001010101011

J



UNIVERSITÉ DE NANTES

Décodage

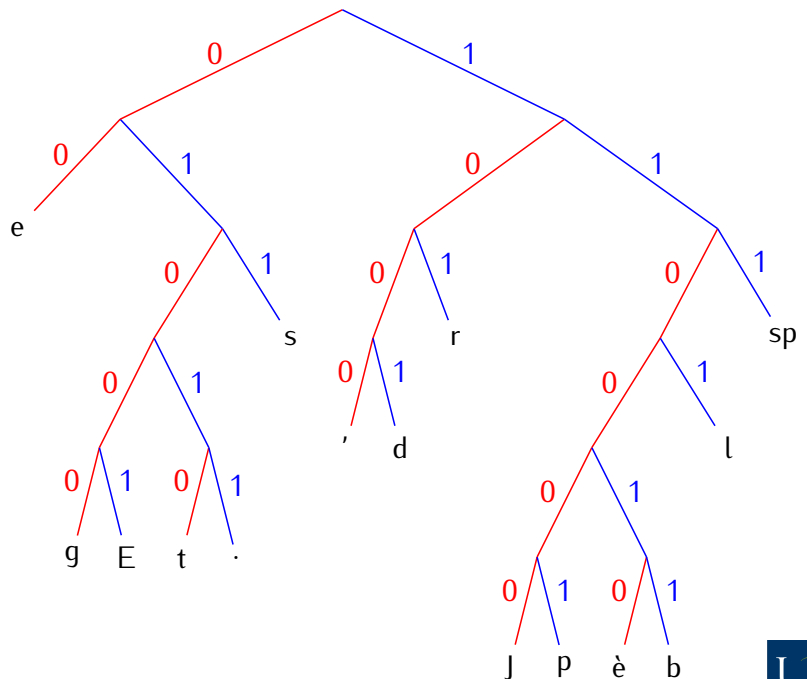
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000010110100111110001
1011100100111111001000111111
0011001010100000011111001001
1111011000010011101011010100
0
10101011

J



UNIVERSITÉ DE NANTES

Décodage

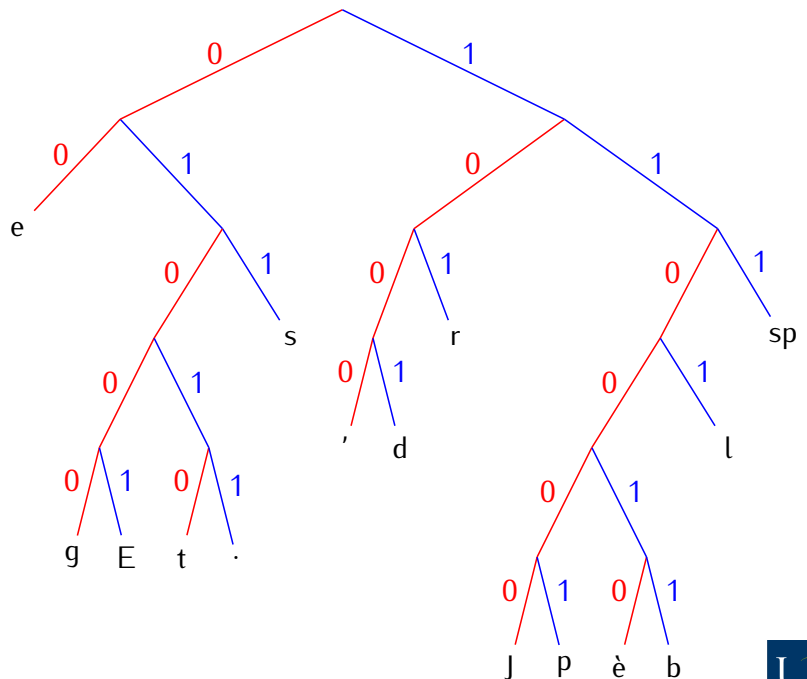
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000010110100111110001
1011100100111111001000111111
0011001010100000011111001001
1111011000010011101011010100
0
10101011

J'



UNIVERSITÉ DE NANTES

Décodage

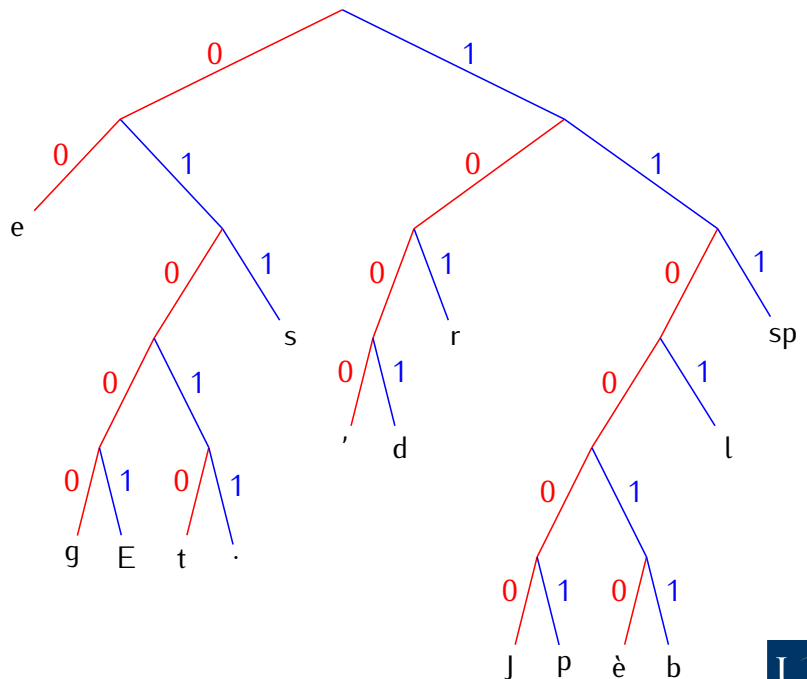
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111001001111111001000111111100100110010101000000111111001001111101100001001110101101010001010101011

J'



UNIVERSITÉ DE NANTES

Décodage

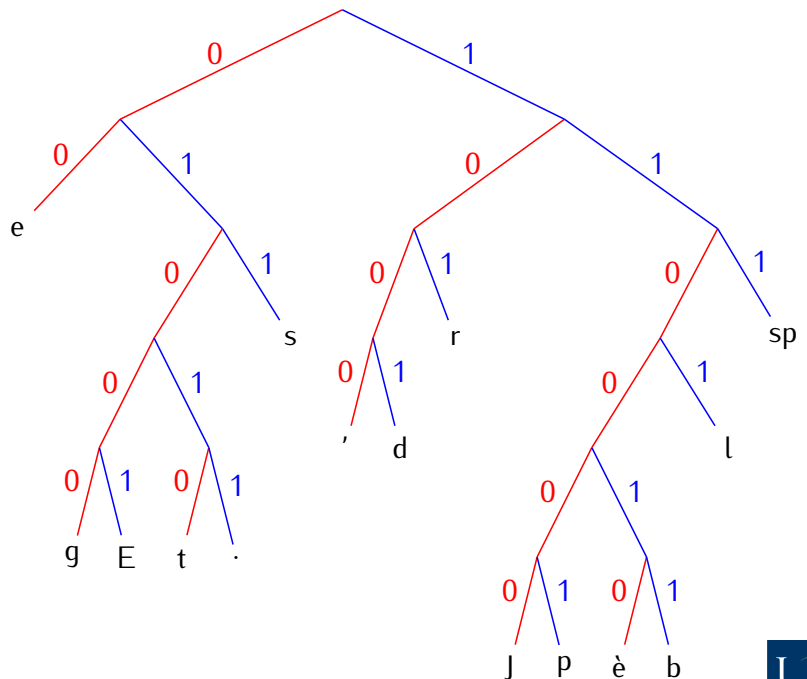
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100101010000001111111001001111101100001001110101101010001010101011

J'e



UNIVERSITÉ DE NANTES

Décodage

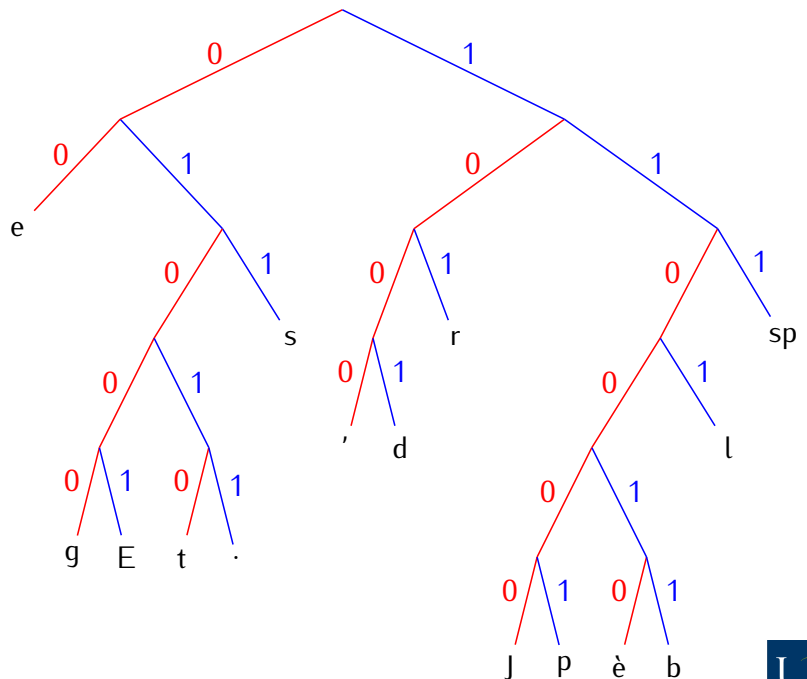
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100101010000001111111001001111101100001001110101101010001010101011

J'e



UNIVERSITÉ DE NANTES

Décodage

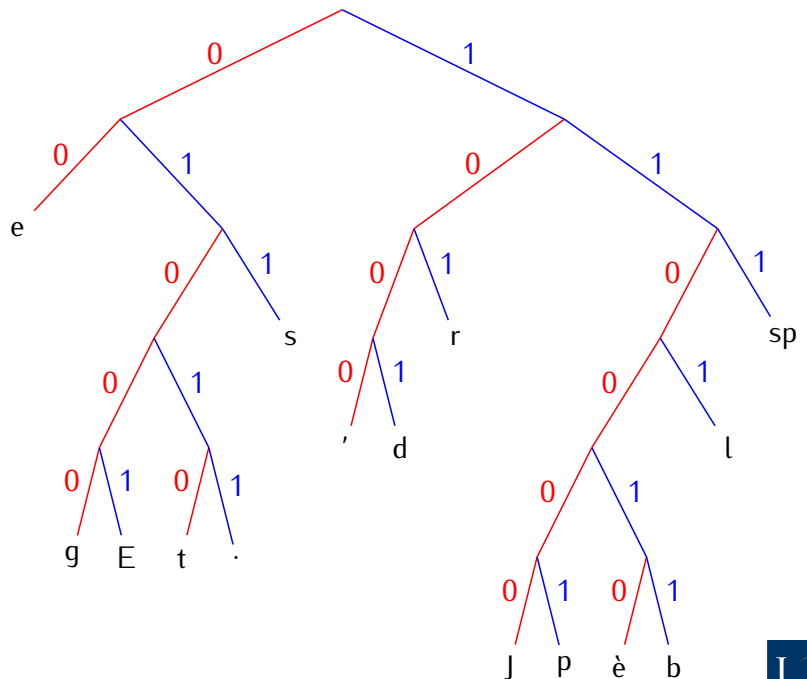
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100101010000001111111001001111101100001001110101101010001010101011

J'e



UNIVERSITÉ DE NANTES

Décodage

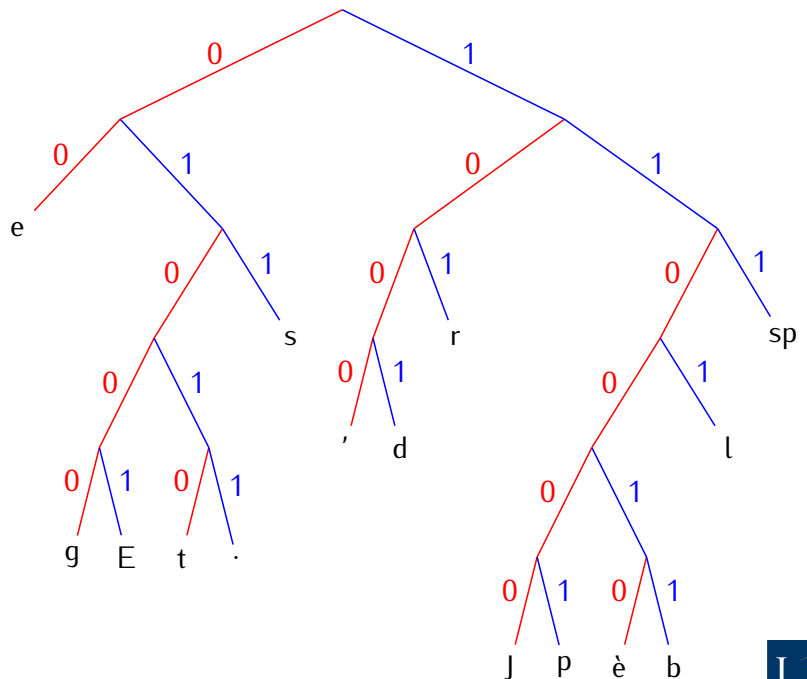
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100101010000001111111001001111101000010011101011010100010101011

J'er



UNIVERSITÉ DE NANTES

Décodage

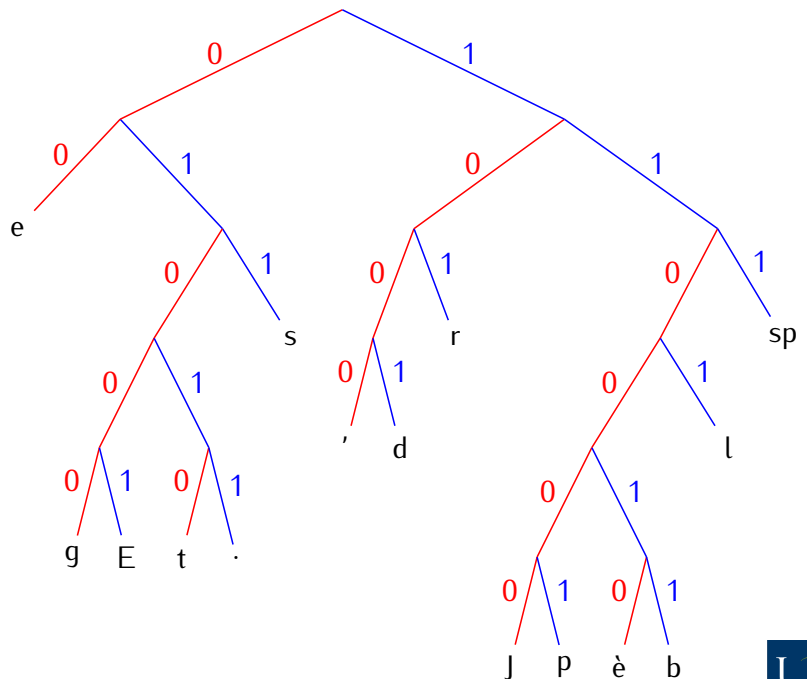
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

1100001000001011010011111000110111100100111111100100011111110011001010100000011111100100111111010110101000010101011

J'er



UNIVERSITÉ DE NANTES

Décodage

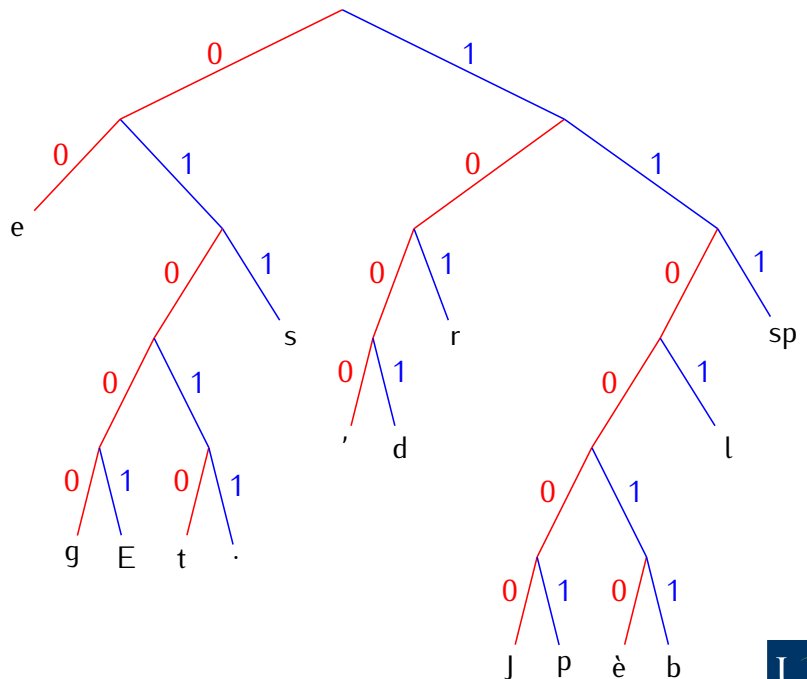
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100100101010000001111111001001111101000010011101011010100010101011

J'er



Décodage

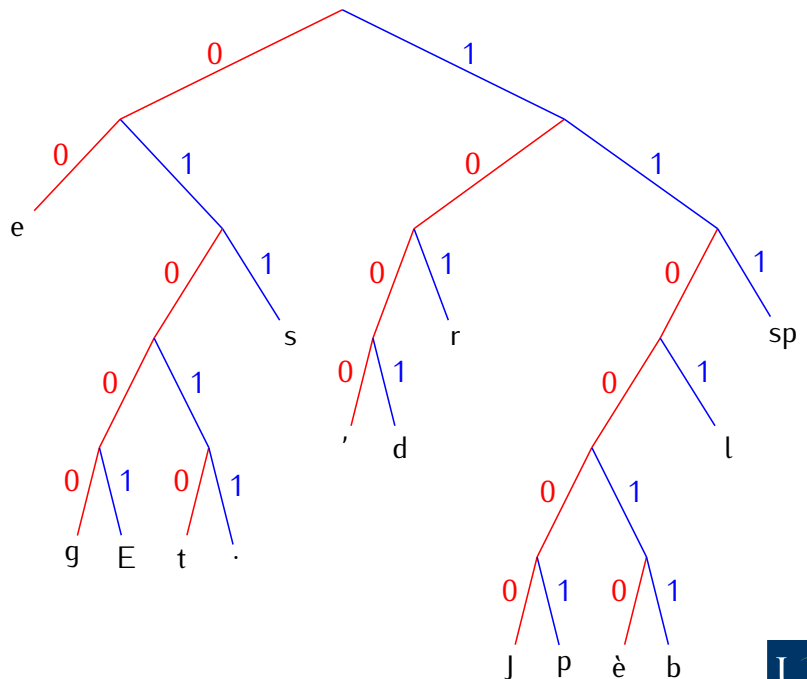
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111110011001010100000011111100100111111010110101000010101011

J'err



Décodage

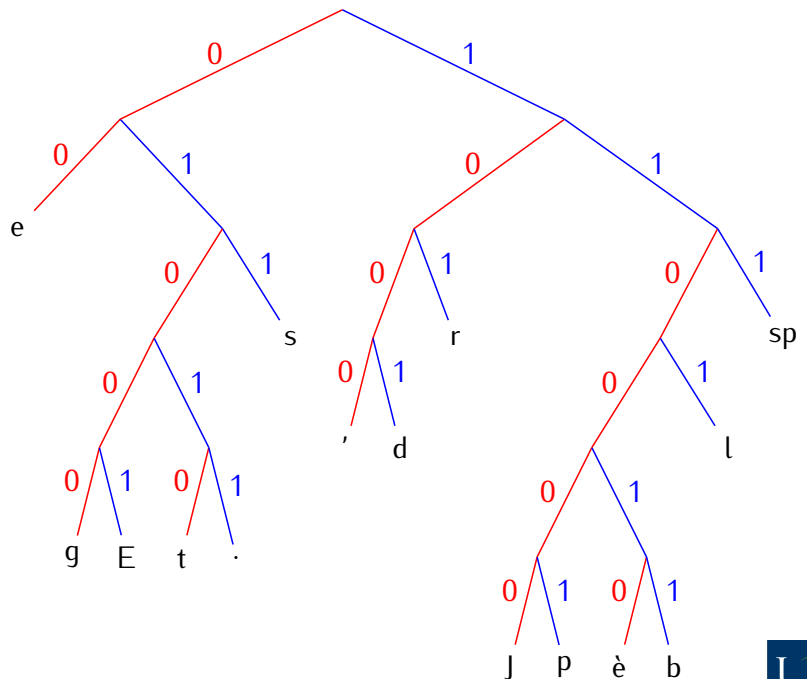
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101110100111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'err



UNIVERSITÉ DE NANTES

Décodage

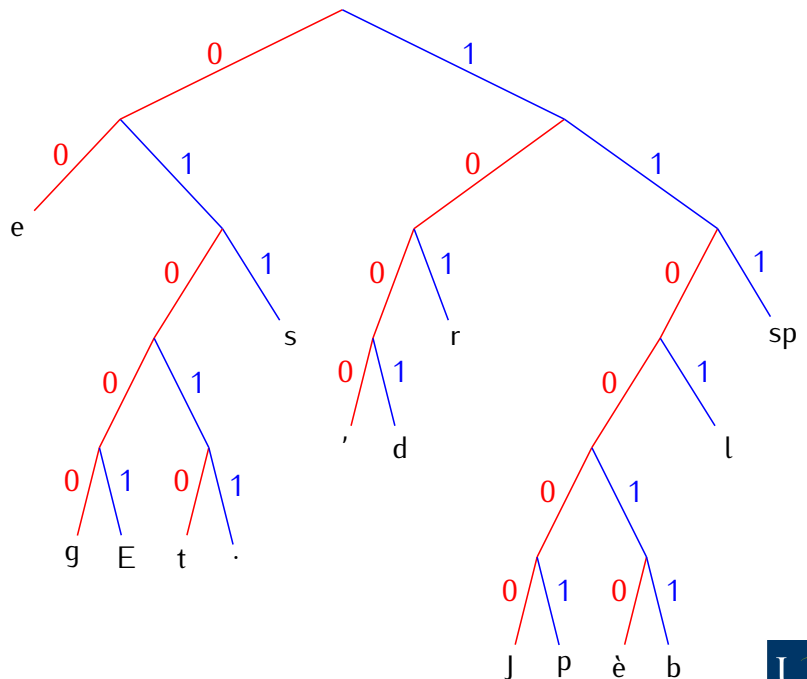
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

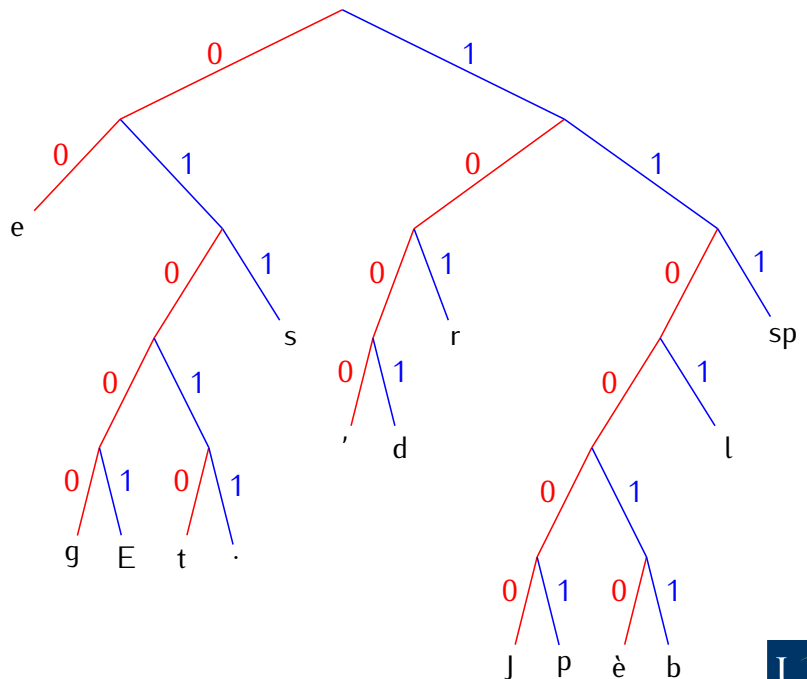
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

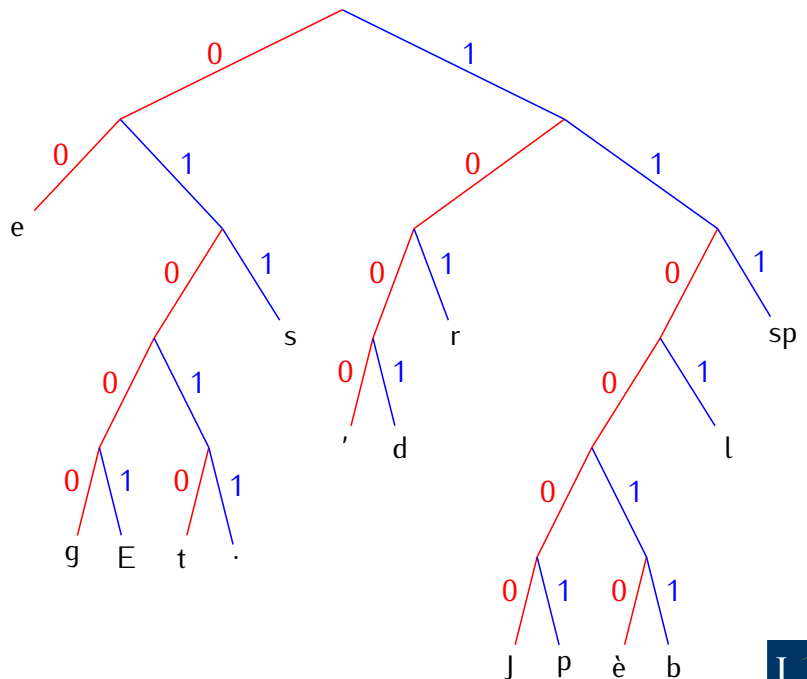
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

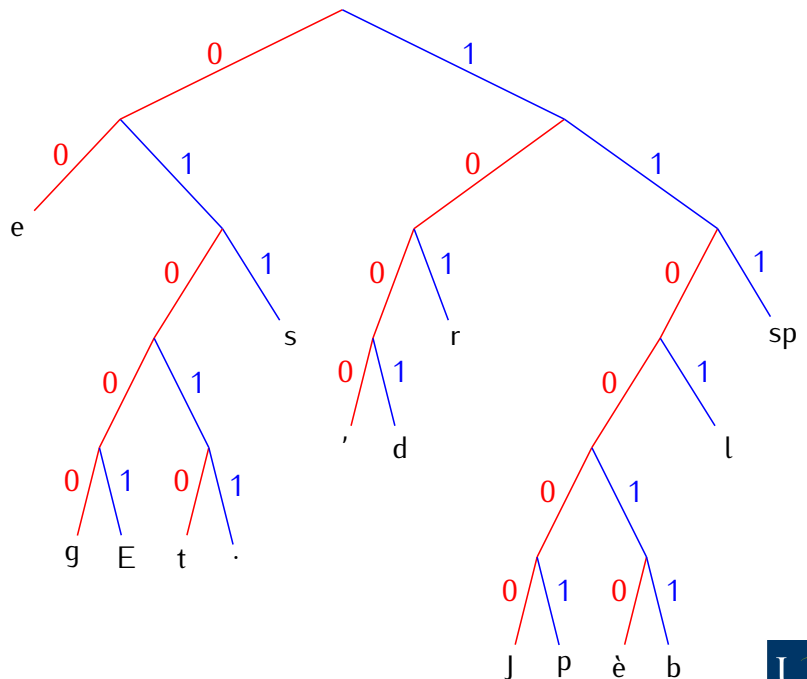
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

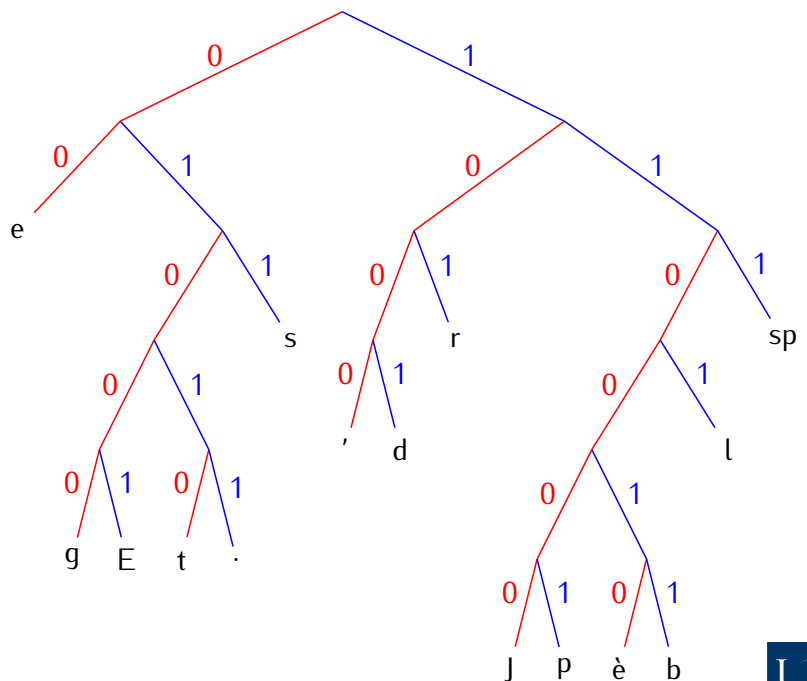
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

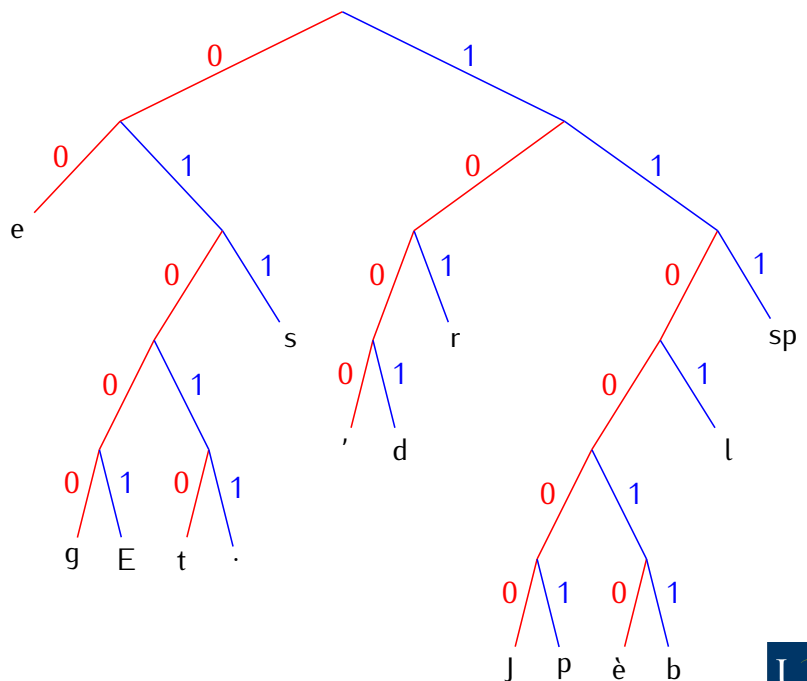
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

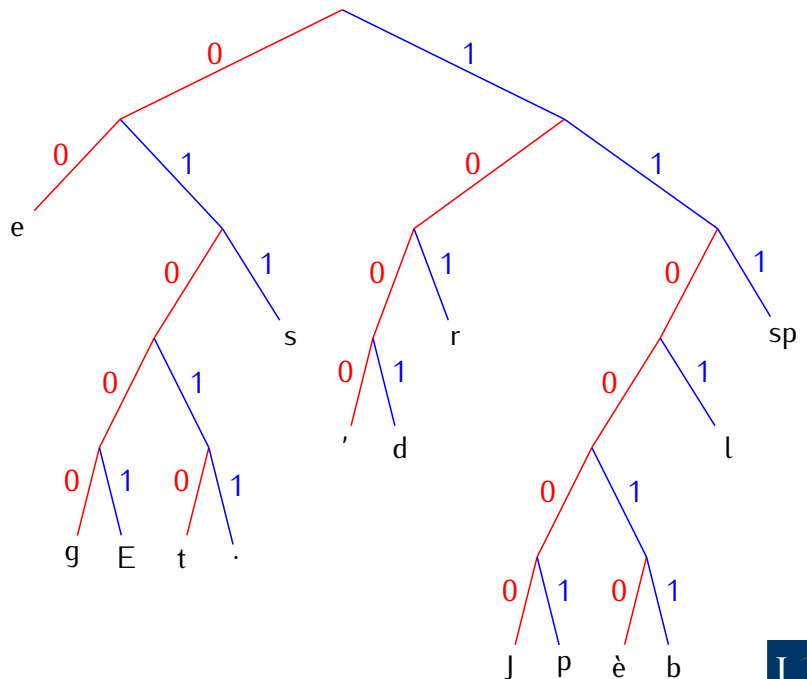
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

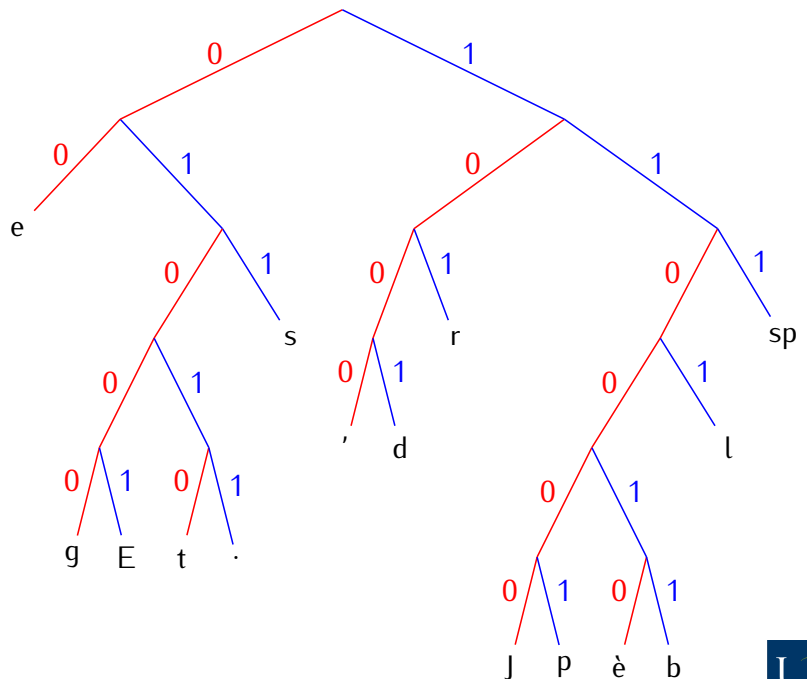
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

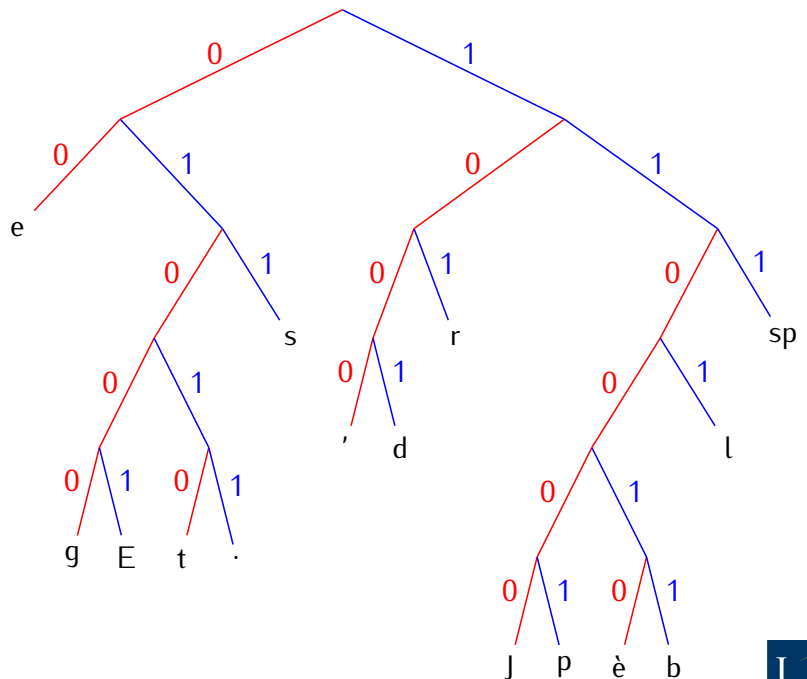
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111110001
101110010011111100100011111111
001100101010000001111111001001
111101100001001110101101010000
10101011

J'erre



UNIVERSITÉ DE NANTES

Décodage

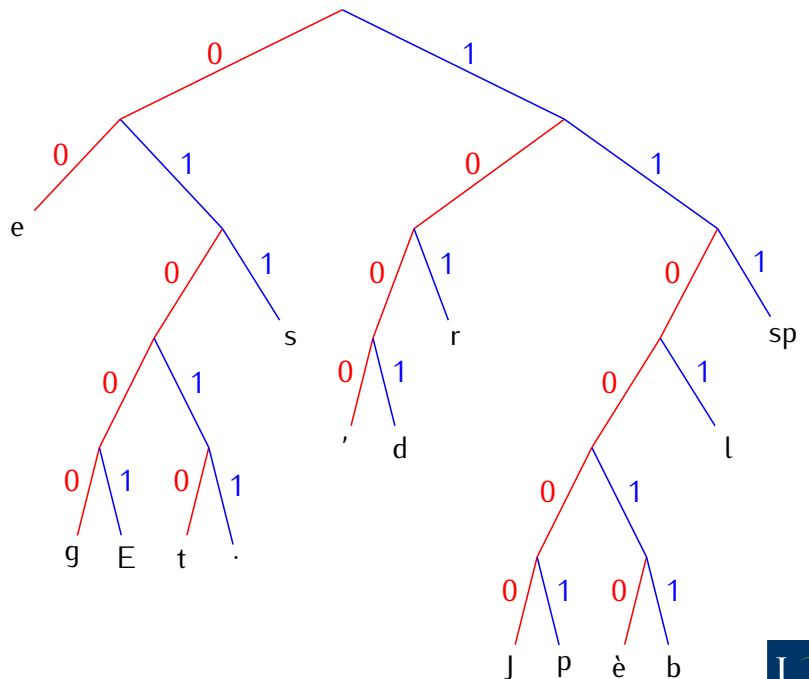
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111110011001010100000011111100100111111010110101000010101011

J'erre p



UNIVERSITÉ DE NANTES

Décodage

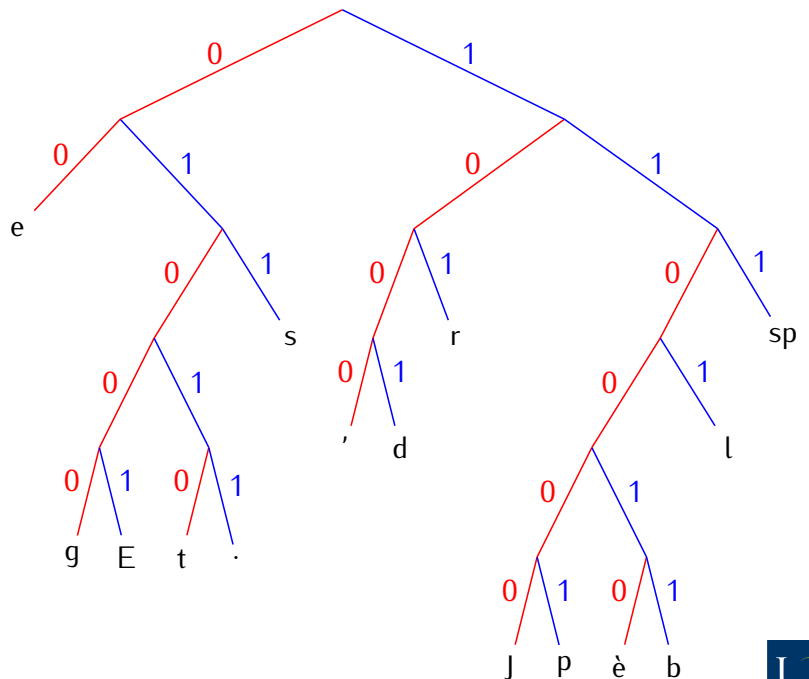
Étape 3b

On utilise l'arbre pour décoder, on suit les branches correspondant aux bits lus, quand on arrive à une feuille on a décodé le symbole correspondant

Exemple

110000100000101101001111100011011100100111111100100011111111001100101010000001111111001001111101000010011101011010100010101011

J'erre près des berges de l'Elster.



UNIVERSITÉ DE NANTES

Utilisation pratique de Huffman

Compression sans perte d'information

Plusieurs méthodes de compression sans perte utilisent l'algorithme de Huffman, souvent en combinaison avec d'autres algorithmes

Utilisation pratique de Huffman

Compression sans perte d'information

Plusieurs méthodes de compression sans perte utilisent l'algorithme de Huffman, souvent en combinaison avec d'autres algorithmes (si Huffman est optimal, pourquoi faire ça ?)

Exemple

Zip : LZ77 suivi de Huffman

Utilisation pratique de Huffman

Compression sans perte d'information

Plusieurs méthodes de compression sans perte utilisent l'algorithme de Huffman, souvent en combinaison avec d'autres algorithmes (si Huffman est optimal, pourquoi faire ça ?)

Exemple

Zip : LZ77 suivi de Huffman

Support à la compression avec perte d'information

Après avoir utilisé des techniques de compression détruisant de l'information on peut toujours ajouter une passe de compression sans perte d'information

Merci à Loïg Jézéquel