Information

Codage de l'Information

Massih-Reza Amini

Université Joseph Fourier
Laboratoire d'Informatique de Grenoble
Massih-Reza.Amini@imag.fr

UNIVERSITE JOSEPH FOURIER
SCIENCES.TECHNOLOGIE.SANTÉ

Table des matières

Programme

1. Le codage numérqiue de l'information
   - nombres entiers, réels
   - chaînes de caractères (texte) - ASCII, Unicode, UTF8
2. La représentation de l'information textuelle
   - Texte plein, HTML, XML
3. La théorie de l'information
   - Notion probabiliste de l'information
   - Codage de source

Nécessité de coder l'information

Un ordinateur manipule des données de différente nature :

- Nombres (entiers, relatifs, réels)
- Chaînes de caractères
- Objets complexes (dates, images, son, ...)

de taille différente qu'il est nécessaire de coder de façon uniforme. Nous nous intéressons dans un premier temps aux nombres.

Codage des nombres

Il y a bien sûr beaucoup de façons de coder des nombres (et nous utilisons toujours plusieurs bases différentes). La présence ou l'absence du charge électrique, régit la représentation de l'information dans un ordinateur.

La base binaire (1 ou 0), et ses dérivées, base octale et hexadécimale, sont privilégiées en informatique.

Base binaire

Symboles utilisés : 0 et 1

Exemple

$$7_{10} = 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 111_2$$

Addition et soustraction (et multiplication) standard, avec la table suivante :

$0 + 0 = 0$	$0 + 1 = 1$	$1 + 0 = 1$	$1 + 1 = 0$ avec 1 retenue
$0 - 0 = 0$	$0 - 1 = 1$ avec 1 retenue	$1 - 0 = 1$	$1 - 1 = 0$

Exemples

1001 + 11 =
1101 + 1010 =
1000 - 1 =
1101 $\times$ 10 =

Bases octale et hexadécimale

Symboles utilisés : $\overbrace{\underbrace{01234567}_{\text{octal}}89ABCDEF}^{\text{hexadécimal}}$

Exemples

$$171_{10} = 2 * 8^2 + 5 * 8^1 + 3 * 8^0 = 253_8$$

$$171_{10} = \underbrace{10}_{A} * 16^1 + \underbrace{11}_{B} * 16^0 = AB_{16}$$

Conversions entre bases (1)

→ D'une base b vers la base décimale ; $N = \sum_{i=0}^{p} a_i b^i$

- Multiplications et additions successives ...
- ... par la méthode de Horner :

On considère le polynôme de degré p :
$P(x) = a_p x^p + a_{p-1} x^{p-1} + \cdots + a_1 x + a_0$
Alors, pour tout x_0, la (pseudo-)suite :

$$u_p = a_p, \forall n, p \geq n > 0, u_{n-1} = a_{n-1} + u_n x_0$$

est telle que : $u_0 = P(x_0)$. D'où l'algorithme :

Algorithm 1 Algorithme de Horner

```
1: Initialisation: n ← p
2: Res ← a_n
3: repeat
4:     Res ← a_{n-1} + x_0 * Res
5:     n ← n − 1
6: until n = 0
```

Conversions entre bases (2)

→ De la base décimale vers une base b

Méthode des divisions (entières) successives :

1. (q_0, r_0) quotient et reste de la division euclidienne de N par b
2. Pour $i > 0$, (q_i, r_i) quotient et reste de la division euclidienne de q_{i-1} par b
3. q_k premier quotient strictement inférieur à b obtenu de la sorte

→ $N_b = q_k r_k \cdots r_0$

Conversions entre bases (3)

→ Conversions entre binaire et octal/hexadécimal

- Un chiffre octal correspond à 3 chiffres binaires
- Un chiffre hexadécimal correspond à 4 chiffres binaires

☞ Du binaire à l'octal/hexadécimal

1. On regroupe, en partant de la droite, les chiffres par paquets de 3/4, en ajoutant évtlt des 0 en tête
2. On remplace chaque paquet par son chiffre correspondant en octal/hexadécimal

☞ De l'octal/hexadécimal au binaire : on remplace chaque chiffre par un paquet de 3/4 chiffres binaires que l'on concatène

Illustration

10110001101 en octal et hexadécimal ?

254_8 en binaire, $D46C_{16}$ en binaire ?

Le cas des nombres réels (1)

→ Conversion base b vers la base décimale

Notation à virgule, avec puissances négatives :

$$\begin{aligned} N_b &= a_n a_{n-1} \cdots a_1 a_0, a'_1 \cdots a'_m \\ &= (a_n b^n + \cdots + a_0 + a'_1 b^{-1} + \cdots + a'_m b^{-m})_{10} \end{aligned}$$

Le cas des nombres réels (2)

→ Conversion décimal vers une base *b*

- Méthode des divisions successives pour la partie entière
- Pour la partie fractionnaire :
  1. Multiplier la partie fractionnaire par *b* et noter la partie entière obtenue
  2. Recommencer cette opération avec la partie fractionnaire du résultat
  3. Arrêter quand la précision souhaitée est obtenue (ou quand la partie fractionnaire est nulle)
  4. Concaténer les parties entières obtenues dans l'ordre après la virgule

Illustration

12,6875 en binaire ?

171,3046875 en binaire ?

Le cas des nombres réels (3)

Conversion entre binaire et octal/hexadécimal : comme précédemment, mais depuis la gauche vers la droite après la virgule

$1948,B6_{16} = 0001100101001000,10110110_2$

$14510,554_8 = x_2$?

Les entiers positifs (naturels)

Utilisation du codage binaire, avec taille fixe

- Chaque chiffre est un *bit* (atome)
- La taille correspond aux nombres de bits utilisés pour coder les entiers naturels
  - Codage sur n bits : nombres de 0 à $2^n - 1$
  - Codage sur 1 octet (8 bits) : nombres de 0 à 255
  - Codage sur 2 octets (16 bits) : nombres de 0 à 65535
  - Codage sur 3 octets (32 bits) : nombres de 0 à 4 294 967 295
- Les bits sont rangés selon leur poids (complément à gauche par des 0)

Remarque entiers non signés

Problème de la taille fixe ou de la précision finie

Remarque : L'algèbre des nombres en précision finie est différente de l'algèbre des nombres normale :

$$a + (b - c) = (a + b) - c$$

Que se passe-t-il si l'on utilise un codage sur 3 bits avec $a = 100, b = 111, c = 101$?

Les entiers relatifs (1)

Problème non trivial, si on réserve le bit de poids fort pour le signe (0 pour les nombres positifs, 1 pour les nombres négatifs). Les autres bits codent la valeur absolue du nombre

$\rightarrow$ Binaire signé ou entiers signés

Exemple

Codage en binaire signé sur un octet des nombres décimaux 24, -24, 128 et -128 ?

- Étendue du codage : avec n bits on peut coder tous les nombres N tels que $-(2^{n-1}-1) \leq N \leq 2^{n-1}-1$ (avec 4 bits, tous les nombres entre -7 et 7)
- Mais ...
  - 0 est codé deux fois
  - Addition et soustraction moins évidente car l'opération du passage au négatif n'est pas réversible

Les entiers relatifs (2)

Deuxième solution Le complément à 2 ou complément arithmétique

1. Les nombres positifs sont codés de la même façon qu'en binaire signé
2. Un nombre négatif est codé
   - en prenant son opposé,
   - en le représentant en base 2 sur $n-1$ bits,
   - en complémentant chaque bit, et
   - en ajoutant 1

Exemples et remarque

- Codage en complément à 2, sur un octet, des nombres décimaux 24, -24, 128 et -128 ?
- Sur 4 bits, à quel nombre décimal correspond le nombre codé en complément à 2 par 1111 ?
- Addition, la retenue engendrée par les bits les plus à gauche est simplement ignorée (ex. sur 10-3)

Quelques exemples de taille

Illustration sur la norme ANSI C

Dénomination	Taille min. (octets)	Domaine min.
Entier court signé	2	-32768 à 32767
Entier court non signé	2	0 à 65535
Entier signé	2	-32768 à 32767
Entier non signé	2	0 à 65535
Entier court signé	4	-2 147 483 648 à 2 147 483 647
Entier court non signé	4	0 à 4 294 967 295

Codage des réels

- Il s'agit de représenter un nombre binaire à virgule (par exemple, $101,01$ qui ne se lit pas *cent un virgule zéro un* puisque c'est un nombre binaire mas $5,25$ en décimale) sous la forme $1,... \times 2^n$ (c'est à dire dans notre exemple $1,0101 \times 2^2$)
- La norme IEEE définit la façon de coder un nombre réel sur 32 bits et définit trois composante
  - *Signe* sur 1 bit
  - *Mantisse* sur 23 bits
  - *Exposan*t sur 8 bits (biais $= 2^7 - 1 = 127$)

• La valeur d'un nombre est donnée par :

$$(-1)^s \times (1 + \sum_{i=1}^{23} m_i 2^{-i}) \times 2^e$$

Attention, e représente l'exposant réel et non la valeur, biaisée, stockée dans le champ de l'exposant (e = champ exposant - biais)

Codage des réels (2)

Ainisi le codage d'un réel se fait sous la forme

$$s \underbrace{eeeeeeee}_{8bits} \underbrace{mmmmmmmmmmmmmmmmmmmmmmm}_{23bits}$$

Codage sur la valeur 525,5

- 525,5 est positif donc le 1er bit sera 0
- Sa représentation en base 2 est: 1000001101,1
- En normalisant on trouve $1,0000011011 \times 2^9$
- On ajoute 127 à l'exposant qui vaut 9 ce qui donne 136, soit en base 2: 10001000
- La mentisse est composée de la partie décimale de $525,5$ en base 2 normalisée, c.à.d 0000011011
- on complète la mentisse en ajoutant des 0 à droite pour qu'il occupe 23 bits, soit 0 10001000 00000110110000000000000

Quelques exercices

1. Quelle est l'étendue de codage sur 8 et 16 bits en :
   - Binaire pur
   - Binaire signé
   - Complément à 2
2. Coder les nombres 20 et -15 sur 8 et 16 bits en binaire signé, complément à 1 et complément à 2 ?
3. Calculer 20 - 15 sur 1 et 2 octets en complément à 2 ?
4. Décoder en décimal 11001001_{c2} et 01101101_{c2} (codage sur 1 octet)
5. Coder -9,5 et 6,25 avec la norme IEEE.

L'ASCII (1)

Un codage historique : ASCII (*American Standard Coding for Information Interchange*)

- 128 caractères (jeu minimal) : 33 non imprimables, 94 imprimables, et l'espace (considéré comme invisible)
- Des limitations claires (US-ASCII), liées à son histoire
- Jusque fin 2007, l'encodage le plus répandu (UTF-8 depuis)

L'ASCII (2)

- 32 premiers caractères de contrôle (pour des machines comme les imprimantes)
  - Caractère 10 : LF (*Line Feed*), permet de faire avancer le papier dans une imprimante)
  - Caractère 8 : BS (*backspace*, retour arrière)
- Pas de description de la structure d'une page (language de balisage)

L'ASCII (3)

- Beaucoup d'extensions et de propositions nationales pour utiliser le huitème bit et coder les caractères suivants
- Certaines extensions non compatibles avec le jeu minimal !

UTF-8 (1)

UCS Transformation Format 8 bits

- Permet de représenter les (milliers de) de caractères du répertoire universel (UCS), commun à la norme IS/CEI 10646 et au standard Unicode
- Compatible avec US-ASCII
- UTF-8 est approuvé par l'IETF (*Internet Engineering Task Force*) et le W3C (*World Wide Web Consortium*)
- Inventé en 1992 par K. Thompson (Unix)

UTF-8 (2)

Remarque On parle de points de code (*code points*) plutôt que de caractères dans UCS

Principe

1. Les points de code ayant une valeur (scalaire, en base 10) comprise entre 0 et 127 sont codés sur un seul octet dont le bit de poids fort est nul
2. Les points de code de valeur supérieure à 127 sont codés sur plusieurs octets :
   - les bits de poids forts du premier octet forment une suite de 1 de longueur égale au nombre d'octets utilisés pour coder le caractère
   - les octets suivant ont 10 comme bits de poids forts

UTF-8 (3)

Exemple

é - 233 - 11000011 10101001

2048 ?

UTF-8 (4)

Propriétés

- Unicité du codage (séquence minimale)
- Lecture possible à partir de n'importe quelle position
- Perte d'efficacité en lecture car taille variable