Représentation binaire d'un entier relatif

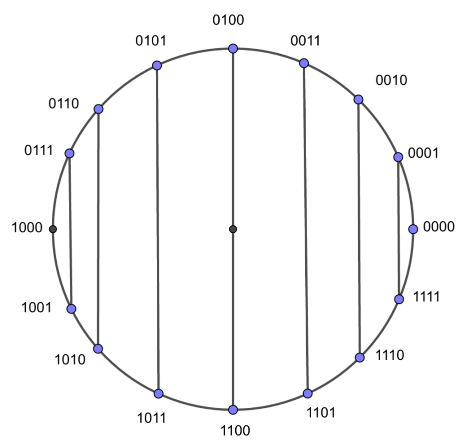
Comment représenter les entiers négatifs ?

1. *Une première approche* : Par exemple sur un octet on va coder 4 par 0000 0100 et pour coder -4 on va utiliser **un bit significatif** celui qui est le "plus à gauche" dit **bit de poids fort** que l'on va mettre à 1 pour dire que ce nombre est négatif et on ne change pas les autres bits ce qui donne
   $-4 = (10000100)_2$
   Calculons $-4 + 4 = (10001000)_2$ qui vaut -8 avec notre convention au lieu de 0 ce qui est un problème !
2. *Une deuxième approche* Supposons que les entiers sont codés uniquement sur 4 bits et plaçons les mots de 4 bits dans l'ordre croissant de 0000 jusqu'à 1111 sur un cercle de la manière suivante : on apparie ensemble les mots de telle sorte que la somme des deux mots donne toujours 1 0000 que l'on va assimiler à 0

   On a donc apparié les entiers positifs et leur opposé on choisit pour positifs les mots de 4 bits **dont le bit de poids fort est 0** et les négatifs ceux dont le bit de poids fort est 1

   Ainsi 2 est codé par 0010 et -2 par 1110.

   1000 n'a plus de sens dans cette représentation. Ce mot n'est pas utilisé.



Théorème 1. *(Complément à 2) Si les entiers relatifs sont codés sur n bits alors :*

1. *Il y a* $2^{n-1} - 1$ *mots ayant un bit de poids fort égale à 0, ils servent à coder les entiers positifs (On exclut le mot* $1\ \underbrace{0...0}_{2^{n-1}\textit{zéros}}$ *)*
2. *Les mots précédents ont un opposé obtenu en remplaçant chaque chiffre par son complément et en ajoutant 1*

Preuve

Soit $m = c_n c_{n-1} .... c_0$ *avec* $c_i \in \{0, 1\}$ *et* $c_n = 0$ *et où* $\overline{c_i}$ *le complémentaire de* c_i

Considérons $\overline{m} = \overline{c_n c_{n-1}} .... \overline{c_0}$ *donc* $m + \overline{m} = 11....11$ *(que des 1)*

donc $m + \overline{m} + 1 = m + (\overline{m} + 1) = 1\ 00...00$ *que l'on assimile à 0, et l'opposé de m et* $\overline{m} + 1$

Exemples

1. Sur 16 bits, il y a $2^{15}-1=32767$ nombres positifs (ou négatifs).
   Le plus grand entier positif que l'on peut coder est $2^{15}-1=32767$.
   Ensuite $2^{15}=1\underbrace{0..0}_{15\_\text{zéros}}$ est un codage non significatif.
   L'entier 2024 qui est strictement inférieur à 32767 peut donc être codé (ainsi que -2024)
   2024 est codé par 0000 0111 1110 1000 donc pour trouver le code de son opposé on prend le complémentaire 1111 1000 0001 0111 puis on ajoute 1
   ce qui donne pour le codage de -2024 –> 1111 1000 0001 1000
2. Quel est l'entier relatif codé par 1111 1000 0001 1000 sur 2 octets ?
   Déjà on remarque que le bit de poids fort est 1 donc c'est un nombre négatif.
   Pour trouver de quel entier il s'agit, dans un premier temps on retranche 1 à 1111 1000 0001 1000
   On obtient 1111 1000 0001 0111
   Ensuite on prend le complémentaire du mot ci-dessus
   On obtient 0000 0111 1110 1000 et on retrouve 2024, donc 1111 1000 0001 1000 code -2024
3. Peut on coder 10^{15} sur 4 octets ?
   Non, car le plus grand entier naturel que l'on peut coder sur 32 bits est :
   $2^{31}-1=2147483647<10^{10}$
4. Peut on coder 10^{15} sur 8 octets ?
   Oui, car le plus grand entier que l'on peut coder sur 64 bits est :
   $2^{63}-1=9223372036854775807>9\times 10^{18}$
5. Sur 8 octets 2^{15} est codé en binaire par
   0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1000 0000 0000 0000
   En hexadécimal :
   00 00 00 00 00 00 80 00
   On prend le complémentaire :
   ff ff ff ff ff ff 7f ff
   On ajoute 1 et on a la représentation de -2^{15} sur 64 bits (en hexadécimal)
   ff ff ff ff ff ff 80 00
6. Par contre on peut lire cette succession d'octets dans deux sens :
   (a) De la gauche vers la droite ('big endian') :
       ff ff ff ff ff ff 80 00
   (b) De la droite vers la gauche ('little endian')
       00 80 ff ff ff ff ff ff

```
>>> (-2**15).to_bytes(8,byteorder='big',signed=True)
b'\xff\xff\xff\xff\xff\xff\x80\x00'

>>> (-2**15).to_bytes(8,byteorder='little',signed=True)
b'\x00\x80\xff\xff\xff\xff\xff\xff'
```

Exercices

Ex 1

Sur 8 bits coder -1 et -4

Combien de nombres négatifs peut on coder sur 8 bits ?

Ex 2

1. Quel est le plus grand entier positif que l'on peut coder sur 16 bits ?
   Peut on coder 10^9 sur 16 bits ?
2. Sur 16 bits coder 15,16,31 et 32 puis leur opposé

Ex 3

1. Quel est le plus grand entier positif que l'on peut coder sur 4 octets ?
2. Sur 4 octets coder 2048 et -2048

Ex 4

1. Les systèmes d'exploitation actuels codent les entiers sur 64 bits. Quel est le plus grand entier naturel représentable en machine ?
2. Combien peut-on coder de nombres négatifs sur 64 bits ?
3. Coder 2^{12} sur 64 bits en hexadécimal
4. Coder -2^{12} sur 64 bits en hexadécimal

Ex 5

Quel entier relatif est codé par 1111 1000 sur 8 bits ?

Ex 6

Quel entier relatif est codé par 1111 1111 1111 1000 sur 16 bits ?

Ex 7

Quel entier relatif est codé par ff ff ff ff ff ff ca fe sur 8 octets ?

Ex 8

Expliquer les messages d'erreur

```
>>> (1023).to_bytes(1,byteorder='big',signed=False)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
OverflowError: int too big to convert
```

```
>>> (10**15).to_bytes(4,byteorder='big',signed=False)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
OverflowError: int too big to convert
```

Ex 9

A partir des informations suivantes trouver la représentation en machine de 10^{15} et -10^{15} sur 64 bits (8 octets)

```
>>> hex(10**15)
'0x38d7ea4c68000'

>>> (10**15).to_bytes(8,byteorder='big',signed=False)
b'\x00\x03\x8d~\xa4\xc6\x80\x00'

>>> (-10**15).to_bytes(8,byteorder='big',signed=True)
b'\xff\xfcr\x81[9\x80\x00'
```