

Représentation des entiers

Fiche de synthèse

① ENTIERS NATURELS

Un **entier naturel** est un entier positif ou nul. Sur n bits, on représente les entiers de 0 à $2^n - 1$: de 0 à 255 sur un octet, de 0 à 65 535 sur deux octets. Par exemple, $9 = 00001001_2$ et $255 = 11111111_2$.

En Python, la taille d'un entier n'est pas limitée : `2 ** 100` se calcule exactement, Python réservant la mémoire nécessaire.

② POURQUOI PAS UN SIMPLE BIT DE SIGNE

Le **bit de poids fort** est le bit de gauche, de plus grand poids ; le **bit de poids faible** est celui de droite. Utiliser le bit de poids fort comme signe, 10000010 pour -2 , pose deux problèmes : zéro a deux écritures, 00000000 et 10000000 , et l'addition binaire donne un résultat faux : $00000011 + 10000110 = 10001001$, soit $3 + (-6) = -9$.

③ LE COMPLÉMENT À DEUX

Définition 1 (Complément à deux) Sur n bits, le **complément à deux** représente un entier positif comme un entier naturel, avec un bit de poids fort à 0, et un entier négatif x par l'écriture binaire de $2^n - |x|$. Le bit de poids fort vaut alors 1. On représente ainsi les entiers de -2^{n-1} à $2^{n-1} - 1$.

Sur huit bits, de -128 à 127 ; sur quatre bits, de -8 à 7 .

☞ Coder un entier négatif sur n bits.

1. Écrire sa valeur absolue en binaire sur n bits : $19 = 00010011_2$.
2. Inverser tous les bits : 11101100 .
3. Ajouter 1 : 11101101 . Donc -19 s'écrit 11101101_2 .

Vérification : $00010011 + 11101101 = 00000000$, la retenue qui sort à gauche étant ignorée, et $19 + (-19) = 0$.

Raccourci : en lisant de droite à gauche, on recopie les bits jusqu'au premier 1 compris, puis on inverse tous les autres. $20 = 00010100_2$ donne $-20 = 11101100_2$.

④ LIRE UN ENTIER RELATIF

- Si le bit de poids fort vaut 0, le nombre est positif : on le lit directement.
- S'il vaut 1, le nombre est négatif : on applique de nouveau le complément à deux pour obtenir sa valeur absolue. 11101101 donne 00010010 , puis $00010011 = 19$: le nombre vaut -19 .

Cas particulier : 10000000_2 vaut -128 sur huit bits, et $+128$ n'est pas représentable. Il y a un nombre négatif de plus que de nombres positifs non nuls.

⑤ L'ADDITION FONCTIONNE

Le même circuit d'addition sert pour les entiers naturels et les entiers relatifs : $00000011 + 11111010 = 11111101$, soit $3 + (-6) = -3$.

⑥ DÉPASSEMENT DE CAPACITÉ

Un résultat hors de l'intervalle de codage provoque un **dépassement de capacité**. Sur huit bits, $127 + 1$ donne $01111111 + 00000001 = 10000000$, qui vaut -128 : on passe du plus grand positif au plus petit négatif. C'est un piège des langages à entiers de taille fixe, comme C ou Java.

📖 À retenir.

- Sur n bits : entiers naturels de 0 à $2^n - 1$; entiers relatifs de -2^{n-1} à $2^{n-1} - 1$.
- Pour coder $-x$: écrire x , inverser les bits, ajouter 1.
- Le bit de poids fort donne le signe : 1 pour un nombre négatif.
- Le complément à deux permet d'additionner avec le circuit habituel.