Session d'exercices - Chaînes de caractères

Conversion binaire → entier positif

Introduction théorique

La conversion d'une représentation binaire vers un entier positif se fait en additionnant chaque puissance de 2 multipliée par la valeur du bit correspondant:

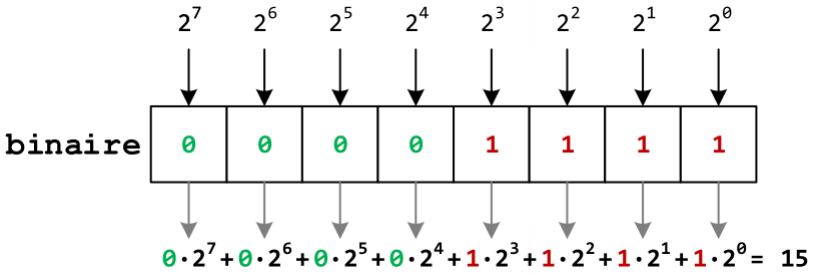


Figure 1: Conversion binaire vers entier. La puissance la plus faible de la base 2 apparaît à droite.

On peut décrire cette conversion succinctement de la manière suivante:

$$d = \sum_{i=0}^{l-1} 2^i \cdot \mathbf{b}[l-i-1].$$

Notation $\mathbf{b}$ est la représentation binaire de l'entier positif d, l sa longueur, et $\mathbf{b}[k]$ avec $0 \leq k < l$ le bit à l'index k. Par exemple, si $\mathbf{b} =$ `"110"`, alors $d = 6$, $l = 3$, $\mathbf{b}[0] = 1, \mathbf{b}[1] = 1$ et $\mathbf{b}[2] = 0$.

Implémentation en Python

La bibliothèque standard de Python contient une fonction `int()` qui permet de convertir une chaîne de caractère vers un entier. Elle prend deux paramètres: l'entier à convertir et, optionnellement, la base dans laquelle il est représenté (base 10 par default).

Pour convertir un nombre `b` représenté en base 2 (binaire), il suffit donc d'exécuter le code suivant: `int(b, 2)`. Par exemple `int("110", 2)` retourne l'entier `6`.

Exercice 1 [Difficulté: *] Écrivez un script `bin2int_simple.py` de deux lignes en Python qui demande à l'utilisateur d'entrer la représentation binaire d'un nombre et qui affiche sa représentation décimale, en utilisant la fonction `int()` de la bibliothèque standard.

Exemple de déroulement:

```
Please enter the binary representation of a number: 1101
The decimal representation of your number is 13
```

Exercice 2 [Difficulté: **] Écrivez un script `bin2int_loop.py` qui fait la même chose que celui de l'exercice 1, mais cette fois-ci en implémentant la conversion manuellement, sans utiliser le deuxième paramètre de la fonction `int()` (vous pouvez utiliser `int(s)` pour convertir un caractère donné en entier, mais pas `int(s, 2)`). Vous pouvez tester que votre script fonctionne correctement en comparant ses résultats à ceux du premier script.

Exercice 3 [Difficulté: **] Écrivez un troisième script `bin2int_listcomp.py` qui fait la même chose comme Exercise 2, mais qui performe la conversion en une seule ligne de code.

Exercice 4 [Difficulté: ***] Pour finir cette partie de l'exercice, écrivez un quatrième script `bin2int_rec.py` et ajoutez-y une fonction récursive `bin2int` qui, pour une chaîne de caractères passée en argument, retourne la valeur entière correspondante.

Conversion entier positif → binaire

Introduction théorique

La conversion d'un entier positif d vers sa représentation binaire $\mathbf{b}$ est moins immédiate; elle est obtenue en utilisant l'algorithme suivant:

$\mathbf{b} \leftarrow$ ""
tant que $d \neq 0$
    $\mathbf{b} \leftarrow \text{str}(d \mod 2) + \mathbf{b}$ # "$d \mod 2$" calcule le reste de la division entière de d par 2
                                  # str(·) convertit un nombre en une chaîne de caractères
    $d \leftarrow d/2$ # division entière de d par 2

Ici, $\mathbf{b}[k]$ est associé à la puissance 2^k. L'algorithme calcule la valeur de chaque bit de droite à gauche. Il commence donc par calculer le *dernier* bit $\mathbf{b}[l-1]$, puis le suivant, et ainsi de suite jusqu'au premier: $\mathbf{b}[0]$.

Par exemple, si d est initialisé à 13, la première itération calcule $\mathbf{b}[3] = 1$ et d devient 6, ensuite $\mathbf{b}[2] = 0$ et d devient 3, ensuite $\mathbf{b}[1] = 1$ et d devient 1, et finalement $\mathbf{b}[0] = 1$ et $d = 0$. Alors, le valeur binaire équivalente à 13 est $\mathbf{b} =$ `"1101"`.

Implémentation en Python

Exercice 5 [Difficulté: **] Écrivez un script `int2bin.py` en Python qui demande à l'utilisateur d'entrer la représentation décimale d'un nombre et qui affiche sa représentation binaire, en utilisant l'algorithme présenté ci-dessus.

Exemple de déroulement:

```
Please enter the decimal representation of a number: 172
The binary representation of your number is 10101100
```

Exercice 6 [Difficulté: ***] Écrivez un script `int2bin_rec.py` et ajoutez-y une fonction récursive `int2bin_rec` qui, pour un entier passé en argument, retourne sa représentation binaire sous la forme d'une chaîne de caractères.

Exercice 7 [Difficulté: ***] L'algorithme présenté ci-dessus calcule la valeur de chaque bit de *droite à gauche* (c'est-à-dire qu'il commence par la valeur du bit correspondant à la plus petite puissance, puis continue avec les bits suivants). Concevez le pseudo-code d'un algorithme produisant le même résultat, mais qui calcule la valeur de chaque bit de *gauche à droite*. Implémentez-le ensuite dans un script `int2bin_alternative.py`.

Indice Le nombre de bits nécessaires pour représenter un nombre d est donné par $\lfloor \log_2(d) \rfloor + 1$ (il est également possible d'utiliser l'attribut `int.bit_length` en Python).