MPSI - ITC - TD 17 : Flottants

Exercice 1 (Écriture binaire à virgule)

Dans ce TP, on représentera les écritures binaires à virgules par deux listes de zéros et de uns : une liste pour les chiffres avant la virgule, et une liste pour les chiffres après la virgule. Dans chacune des liste, l'ordre d'apparition des chiffres dans la liste sera le même que lors d'une écriture manuscrite.
Par exemple, l'écriture binaire 101.01 qui représente le réel 5.25 sera représentée en python par les listes `[1, 0, 1]` et `[0, 1]`.

1. Écrire une fonction python `decode_bin` qui prend en entrée deux listes représentant une écriture binaire à virgule, et qui renvoie le réel associé.
2. Écriture une fonction python `encode_bin` qui prend en entrée un réel positif et renvoie la paire de listes représentant son écriture binaire à virgule.
3. Écrire une fonction python `decalage` qui prend en entrée deux listes représentant une écriture binaire à virgule non-nulle (au moins l'un des chiffres vaut 1) et qui renvoie un entier correspondant à combien de fois il faut décaler la virgule pour qu'il y aie un et un seul 1 à gauche de la virgule. Une sortie positive représentera un décalage vers la gauche, et une sortie négative représentera un décalage vers la droite.

   Par exemple, le décalage nécessaire pour 101.1001 est de 2 tandis que le décalage nécessaire pour 0.0011 est de -3.

Rappel de cours

Soient $n \in \mathbb{N}^*$ et $m \in \mathbb{N}$ deux entiers positifs. Soit $(c_i)_{i\in[\![-n,m]\!]} \in \{0,1\}^{[\![-n,m]\!]}$ une suite de chiffres binaires indicés par les entiers de $[\![-n,m]\!]$. Dans ce cours (ce n'est pas une notation universelle), on note :

$$\mathrm{Dec_B}((c_i)_{i\in[\![-n,m]\!]}) = \sum_{k=-n}^{m} 2^k \times c_k$$

On appellera une telle suite $(c_i)_{i\in[\![-n,m]\!]}$ une *écriture binaire à virgule*, et on la notera $c_m \dots c_0.c_{-1} \dots c_{-n}$
Dans la suite du sujet, on utilisera cette dernière notation au lieu de poser formellement la suite en question. Par exemple, quand on parlera plus tard de $1.b_3b_2b_1$ cela correspondra à la suite $c_{[\![-3,0]\!]}$ donnée par

$$\{c_0 = 1, c_{-1} = b_3, c_{-2} = b_2, c_{-3} = b_1\}$$

Quand la virgule n'est pas précisée, elle est supposée être tout à droite. Ainsi, quand dans la suite du sujet on parlera de $b_7b_6b_5b_4$ cela correspondra à $b_7b_6b_5b_4.0$

Introduction de vocabulaire : mantisse

On rappelle qu'un flottant sur 8 chiffres binaires $b_8 \dots b_1$ encode le nombre réel donné par :

$$(-1)^{b_8} \times \mathrm{Dec_B}(1.b_3b_2b_1) \times 2^{\mathrm{Dec_B}(b_7b_6b_5b_4)-7}$$

L'écriture binaire à virgule $1.b_3b_2b_1$ s'appelle la **mantisse** du nombre flottant. Mais vous noterez que la mantisse en tant que telle n'est pas encodée, mais seulement ses chiffres après la virgule $b_3b_2b_1$. On appelle ces chiffres la **fraction** du nombre flottant, parce que $0.b_3b_2b_1$ est la partie fractionnaire de $1.b_3b_2b_1$.
Le 7 qui apparaît dans la puissance de 2 du décodage est appelé le **biais de l'exposant**. Pour éviter toute confusion,

— on appelle ici **exposant représenté** le nombre $\mathrm{Dec_B}(b_7b_6b_5b_4)$
— on appelle ici **exposant signifié** le nombre $\mathrm{Dec_B}(b_7b_6b_5b_4) - 7$

Introduction de vocabulaire : bit

Pour parler des chiffres binaires, un anglicisme usuel est d'utiliser le mot *bit*, abréviation de *binary digit* en anglais.

Introduction de vocabulaire : poids fort, poids faible, petit-boutisme, grand-boutisme

Dans une écriture binaire (qu'elle soit à virgule ou pas), les chiffres qui correspondent à des puissances de 2 élevées sont dits des chiffres de **poids fort**, tandis que les chiffres qui correspondent à des puissances de 2 plus petites sont dits des chiffres de **poids faible**.
Quand on représente une écriture binaire en commençant par les chiffres de poids fort (comme on le fait quand on écrit au tableau, ou sur une feuille), cela s'appelle du **grand-boutisme** (on commence par le grand bout). Il est parfois pratique de faire l'inverse et représenter une écriture binaire dans l'autre sens (chiffres de poids faible d'abord), on parle alors de **petit-boutisme**.
(Pour savoir d'où viennent ces noms, lisez les *Voyages de Gulliver* de Jonathan Swift, vous rigolerez).

Exercice 2 (Flottants normalisés sur 8 chiffres binaires)

1. Écrire une fonction `exposant_represente` qui prend en entrée une écriture binaire à virgule non-nulle, et qui renvoie l'*exposant représenté* du flottant 8 bits normalisé correspondant. Vous utiliserez votre fonction `decalage` de l'exercice 1.
2. Écrire une fonction `fraction` qui prend en entrée une écriture binaire à virgule et renvoie la liste des chiffres à droite du premier 1 (sans le premier 1).
3. Écrire une fonction `encode_f8` qui prend en entrée un réel et l'encode sous la forme d'un flottant sur 8 chiffres binaires.
4. Comment se comporte votre fonction quand l'entrée n'est pas représentable comme flottant 8 bits ? Quelles autres possibilités seraient envisageables ?
5. Écrire la fonction `decode_f8` qui prend un flottant 8 chiffres et renvoie le réel qu'il représente.

Exercice 3 (Bonus : opérations arithmétiques)

1. Codez une fonction `addition` qui prend en argument deux flottants normalisés sur 8 chiffres et renvoie le flottant normalisé correspondant à l'addition des entrées. Votre code ne passera pas par un décodage-re-encodage.
2. Codez une fonction `multiplication` qui prend en argument deux flottants normalisés sur 8 chiffres et renvoie le flottant normalisé correspondant à la multiplication des entrées. Votre code ne passera pas par un décodage-re-encodage.