TD : Représentation des nombres

Exercice 1. *Représentation binaire entiers positifs*

1. Combien peut-on représenter de nombres entiers positifs sur k bits ?
2. Soit $c = a + b \mod 2^w$ où $0 \leq a, b < 2^w$ sont deux entiers. Comment tester d'après la valeur de c si $c = a + b$ ou s'il y a eu une réduction modulaire ?

On représente un entier sur k bits par une liste de taille k contenant k booléens (`True`, `False`).

3. Quel entier est représenté par `[False, False, True, False, True]` dans les cas suivants ?
   i. `False` représente 1 ; `True` représente 0 ; ordre *gros boutiste*.
   ii. `False` représente 1 ; `True` représente 0 ; ordre *petit boutiste*.
   iii. `False` représente 0 ; `True` représente 1 ; ordre *gros boutiste*.
   iv. `False` représente 0 ; `True` représente 1 ; ordre *petit boutiste*.

On fixe maintenant `False` $= 0$ et `True` $= 1$, et l'ordre petit boutiste.

4. Écrire un algorithme qui prend en entrée deux entiers n et k et renvoie la représentation de n sur k bits, dans la représentation précédente. Que faire si n ne peut pas s'écrire sur k bits ?
5. Écrire un algorithme d'addition binaire avec les spécifications suivantes :
   — Entrée : deux listes de booléens de même taille k, représentant deux entiers a et b ;
   — Sortie : la liste de taille k représentant l'entier $c = a + b \mod 2^k$.
6. *(bonus)* Écrire de même un algorithme de multiplication qui renvoie *deux* listes de taille k, représentant le produit des entrées. *Il peut être intéressant d'écrire une fonction d'addition avec décalage qui prend en entrée deux entiers a et b et un décalage d et calcule $a + b \times 2^d$.*

Exercice 2. *Entiers en complément à 2*

1. Que représente le tableau `[True, False, True, False]` si on est en complément à la base sur quatre bits, en conservant la représentation de l'exercice précédent ?
2. Comment tester si un entier écrit en complément à 2 est positif ou négatif ?
3. Représenter en complément à 2 sur 4 bits les entiers 5 et -5.
4. Appliquer l'algorithme d'addition de l'exercice précédent à ces deux entiers. Qu'obtient-on ?
5. Écrire un algorithme *le plus simple possible* pour passer de la représentation de n à celle de $-n$.
6. Qu'obtient-on si on applique cet algorithme à 0, et -2^{w-1} ? Expliquer.

Exercice 3. *Flottants*

Pour certaines questions, on pourra utiliser `f'{x:.100f}'` qui affiche le flottant `x` avec 100 décimales de précision, et `x.hex()` qui affiche sa mantisse en hexadécimal (et le signe et l'exposant de manière plus *standard*).

1. Combien de réels différents peut-on représenter en flottant sur 32 bits ?
2. Quel est le plus petit réel strictement positif représentable en flottant sur 32 bits ? Et le plus grand ? Donner leurs représentations binaires et leurs valeurs.
3. Écrire une fonction qui calcule le plus petit réel strictement positif représentable par les `float` Python, et une qui calcule le plus grand réel. *La valeur $+\infty$ s'obtient avec* `float('inf')` *en Python.*
4. Utiliser la méthode `hex()` des `float` pour vérifier que les écritures binaires des flottants trouvés est bien celle qu'on attend.
5. Tout rationel admet une écriture ultimement pérodique quelque soit la base utilisée (*exemple :* $1/44 = 0,02\overline{27}$ *en base 10*). Calculer la représentation infinie (ultimement pérodique donc) de $0,1$ écrit en binaire.
6. On définit un type de flottants sur 5 bits : 1 bit de signe, 3 d'exposants et 2 de mantisse. Lister tous les réels positifs représentables avec ce type, et les représenter sur la droite réelle.
7. Comparer $0,3$ et $0,1+0,2$. Expliquer le résultat en observant les valeurs exactes utilisées pour représenter ces réels.

8. Calculer $10^{100} + 1/10^{100} - 10^{100}$. Expliquer le résultat obtenu. Comment s'y prendre pour obtenir le *bon* résultat ?

9. Comparer $(0,7+0,1)+0,3$ et $0,7+(0,1+0,3)$. Expliquer.

10. On souhaite tester si un triangle de côtés a, b et c, avec $c \geq a, b$ est rectangle. On utilise la fonction suivante.

```
def est_rectangle(a, b, c):
    return a**2 + b**2 - c**2 == 0
```

i. Tester si les triplets $(3,4,5)$, $(1,1,\sqrt{2})$ et $(3/10,4/10,5/10)$ définissent des triangles rectangles.

ii. Écrire une nouvelle fonction qui teste si $c^2 = a^2 + b^2$, et retester les triplets.

iii. Que mettre en œuvre pour faire une fonction *robuste* ?

11. Soit $(u_n)_n$ la suite réelle définie par $u_0 = 4$, $u_1 = 4,25$, et $u_n = 108 - \frac{815}{u_{n-1}} + \frac{1500}{u_{n-1}u_{n-2}}$ pour $n > 1$.

i. Programmer cette suite et calculer les 80 premiers termes.

ii. Conjecturer la limite de cette suite.

iii. En admettant que la limite existe, montrer que c'est en fait 5.

iv. *(bonus, plutôt pour les matheux)* Résoudre analytiquement la récurrence et expliquer…