Feuille de TD n° 9 - Représentation des entiers

Proposition : Pour tout entier $n \in \mathbb{N}$, il existe $r \geq 0$ et $a_0, \ldots, a_r \in \{0,1\}$ tels que $n = a_0 + a_1.2 + a_2.2^2 + \ldots + a_r.2^r = \sum_{k=0}^{r} a_k 2^k$.

Exercice 1.

1. Ecrire une fonction `base2` qui prend en entrée un entier positif n et qui renvoie une chaîne de caractères qui contient l'écriture en base 2 de n (de gauche à droite).
2. Déterminer `base2(10)`, `base2(100)`, `base2(65535)`.
3. Pour m la plus grande puissance de 2 qui apparaît dans l'écriture en binaire de n, exprimer m en fonction de n.
   Montrer que `len(base2(n))` est un entier équivalent à $\frac{\ln(n)}{\ln(2)}$.

Exercice 2 (Addition en binaire)**.**

1. Écrire une fonction `plus(b1,b2)` qui prend en entrée deux bits (valant 0 ou 1) et qui renvoie un tuple de deux entiers `(s, r)` : `s` représente le dernier bit de la somme de `b1` et `b2` et `r` l'éventuelle retenue qui devra se propager (elle vaut 0 s'il n'y a pas de retenue et 1 sinon).
   Par exemple, `plus(1,1)` renverra `(0, 1)` et `plus(1,0)` renverra `(1, 0)`.
2. Écrire une nouvelle version de cette fonction `plusr(b1,b2,r)` qui prend en plus en entrée une retenue (valant 0 ou 1) et qui renvoie le couple `(s,t)` où `s` est le dernier bit de la somme `b1 + b2 + r` et `t` la retenue suivante.
   Par exemple, `plusr(1,0,1)` renverra `(0,1)`.
3. Écrire une fonction `ajoute(S1,S2)` prenant en entrée deux chaînes de caractères composées de 0 et de 1 et de même longueur (à vérifier avec un assert) et qui renvoie la chaîne de caractères représentant la somme des deux nombres en binaire correspondant.
   Par exemple, `ajoute("011", "101"])` renverra `"1000"`.
4. Écrire une fonction `ajoute2(S1,S2)` prenant en entrée deux chaînes de caractères composées de 0 et de 1 de longueurs quelconques, et qui renvoie la chaîne de caractères représentant la somme des deux nombres en binaire correspondant.
   On utilisera de la fonction `ajoute`.

Exercice 3.

1. Ecrire une fonction `base10` qui prend en entrée un nombre écrit en binaire sous forme de chaîne de caractères c, et qui renvoie le nombre entier `n` en base 10.
2. Ecrire une fonction `Inv` qui transforme une écriture en binaire en son complémentaire (les 0 deviennent des 1 et les 1 deviennent des 0).
3. Tester `Ajoute(c,Inv(c))` pour plusieurs valeurs de `c`. Que remarque-t-on ?

Exercice 4 (Entiers relatifs en binaire, avec complément à 2)**.**

1. Écrire une fonction `conv_binaire_r(n,b)` qui prend en entrée un **entier relatif** `n` ainsi qu'un entier naturel `b` indiquant le nombre de bits sur lequel on code les entiers en machine, et qui renvoie une chaîne de caractères codant la représentation machine de l'entier n sur b bits en utilisant la méthode du complément à 2. On vérifiera, à l'aide d'un assert, que l'entier `n` est dans le bon intervalle pour être codé sur `b` bits.
   Par exemple, `conv_binaire_r(-1,8)` renverra la chaîne de caractères `"11111111"`, et `conv_binaire_r(1,8)` renverra la chaîne `"00000001"` (on veut toujours une chaîne de caractères de longueur `b`).
2. Quel est le plus grand nombre entier que l'on peut représenter avec `b` bits ?
   Écrire la fonction réciproque `conv_decimal_r(S,b)`.
3. En utilisant les fonctios `Inv` et `Ajoute` dans les exercices précédents, écrire une fonction `oppose(n,b)` prenant en entrée un entier positif `n` et qui renvoie la chaîne de caractères correspondant à la représentation binaire de l'entier `-n` en machine sur `b` bits. Par exemple, `oppose(1,8)` renverra `"11111111"`.

Python dispose de fonctions déjà intégrées (et beaucoup plus rapides) pour effectuer ces conversions :

- `bin(x)` (fonction native en Python) convertit un nombre entier en binaire dans une chaîne avec le préfixe `0b`.
  La tester avec `bin(3)` et `bin(-10)`. Qu'observe-t-on ?
- `binary_repr(x)` (du module numpy), quant à elle, renvoie le même résultat que l'exercice précédent.
  Par exemple, `np.binary_repr(-3, width=5)` renverra `"11101"`.

Exercice 5.

1. Ecrire une fonction `DecimB2` qui renvoie les n premiers chiffres après la virgule, en base 2, d'un nombre réel x dans $[0,1[$.
   Le résultat sera sous forme d'une chaîne de caractères.
2. Ecrire une fonction `ConvReel` qui convertit une écriture décimale en base 2, sous forme de chaîne de caractères, en un nombre réel $x \in [0,1[$.
3. Déterminer l'écriture en binaire de $\frac{3}{4}, \frac{1}{3}, \frac{1}{5}, \frac{3}{7}, \frac{15}{256}$.
   Pour quels nombres cette écriture est-elle en réalité finie ?

4. Vérifier que l'écart entre $x \in [0,1[$ et son approximation en binaire à n chiffres après la virgule est majoré par 2^{-n}.

Le binaire est utilisé entre autres pour stocker les images. Une image est un tableau rectangulaire dont chaque case (pixel) contient une couleur spécifique. Une couleur est codée par trois nuances de couleurs fondamentales (rouge, vert, bleu, abrégé en RVB), et chaque nuance est un nombre entier entre 0 et 255 (0 : absence de couleur, 255 : intensité maximale). Une couleur en informatique se représente donc par 3 octets (ou 6 caractères hexadécimaux).

Cacher une image dans une autre

Contrairement à la cryptographie qui rend les données incompréhensibles, la stéganographie est l'art de dissimuler des données (écrire un texte dont on ne lit qu'une ligne sur deux, écrire à l'encre sympathique, etc).
Avec deux images M, N de même taille, on peut modifier légèrement la couleur de chaque pixel de M pour obtenir une image $\tilde{M}$ qui ressemble à M, mais à partir de laquelle on peut retrouver l'image N (modulo une perte de qualité).

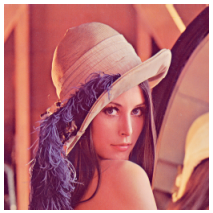
On utilise le fait que dans un nombre écrit sur un octet (8 bits), les 4 bits les plus significatifs sont ceux de gauche (ceux associés à $16, 32, 64, 128$, on parle de bits de poids fort). Les 4 autres bits (associés à $1, 2, 4, 8$) ne changent pas énormément la valeur du nombre (bits de poids faible). On va ainsi conserver les 4 bits significatifs pour chaque pixel de M, et utiliser les 4 autres bits pour stocker les bits significatifs de N.

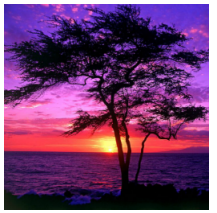
Exercice 6.

1. Écrire une fonction `stegano(S1,S2)` qui prend en entrée deux chaînes de caractères correspondant à la représentation sur 8 bits de deux entiers `n1,n2`$\in \{0,\dots,255\}$, et qui renvoie la chaîne de caractères obtenue en conservant les 4 bits de poids fort de `S1` suivis des 4 bits de poids faible de `S2`.
   Il faut que `stegano("10100000","11110010")` renvoie `"10101111"`.

2. Écrire une fonction `recup(S3)` qui prend en entrée une chaîne de caractère associée à un entier sur 8 bits et qui renvoie une chaîne de caractères correspondant à l'entier `S2` sur 8 bits dont les 4 bits de poids fort sont les 4 bits de poids faible de `S3`. Il faut que `recup("10101111")` renvoie `"11110000"`.

3. Ecrire une fonction qui cache l'image `TP15_lena.png` dans l'image `TP15_coucher.png` (disponibles sur leroy.cpge.free.fr).





Pour importer une image sous la forme d'un tableau numpy (listes de listes), on utilisera la syntaxe :

```
import matplotlib.pyplot as plt #Charger la commande pyplot
M = plt.imread("TP15lena.png") #Lire l'image
M = (M * 255).astype(np.uint8) #Obtenir un tableau de listes de 3 entiers.

# Pour afficher une image :
plt.figure(figsize=(4, 4)) #taille de la figure totale
plt.imshow(P) #Charger l'image P dans la figure
plt.axis("off") #Ne pas afficher les axes sur la figure
plt.show() #Afficher la figure
```