I. Présentation

Définition :
La stéganographie est l'art de la dissimulation : son objet est de faire passer inaperçu un message dans un autre message.

L'idée est de prendre un message et de modifier de manière aussi discrète que possible une image afin d'y dissimuler l'information à transmettre. La technique de base — dite LSB pour least significant bit — consiste à modifier le bit de poids faible des pixels codant l'image. On rappelle qu'une image numérique est une suite de pixels codés sur un octet pour les images en niveaux de gris. Passer d'un niveau n au niveau immédiatement supérieur (n+1) ou inférieur (n-1) ne modifie que peu la teinte du pixel, or c'est ce que l'on fait en modifiant le bit de poids faible de l'octet. On peut donc coder une information en recueillant le dernier bit pour chaque pixel.

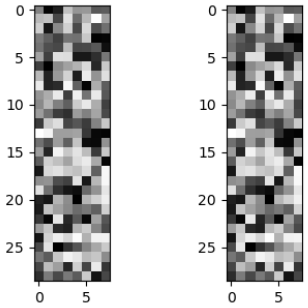

FIGURE 1 – Image originelle à gauche et modifiée à droite

Cette technique de stéganographie très basique s'applique tout particulièrement au format d'image BMP, format sans compression destructive, avec codage des pixels entrelacé sur 1 à 3 octets.

II. Décodage

Pour décoder une image, il est nécéssaire de récupérer le dernier bit du codage du niveau de gris. La fonction `bin` permet de transformer un nombre entier en une chaine de caractère précisant le codage binaire du nombre. Ainsi l'instruction `bin(10)` renvoie `"0b1010"`. Cela signifie qu'en base 10,

$$10 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

Le bit de poids faible est ici 0.

Q 1 - Proposer une fonction `int2bin(nbre)` qui renvoie une liste d'**entiers** codant en binaire le nombre désiré.

On attend les vérifications suivantes :

```
>>> int2bin(10)
[1,0,1,0]
>>> int2bin(8)
[1, 0, 0, 0]
```

Le tableau `img` est un tableau de 8 colonnes pour 29 lignes. Ainsi, en utilisant le dernier bit de chaque pixel, on peut coder 8 bits par colonnes donc 29 octets.

Q 2 - Proposer une fonction `decodage(img)`, utilisant la fonction `int2bin()` qui en parcourant le tableau `img` récupére les valeurs des derniers bits pour chaque pixel. Les données seront regroupées par listes de 8 valeurs. La valeur de retour sera une liste de `len(img)` listes contenant 8 valeurs binaires.

```
>>> decodage(img)
[[0, 1, 0, 0, 1, 1, 0, 0], [0, 1, 1, 0, 0, 1, 0, 1], ... ]
```

Q 3 - Définir une fonction `bin2dec(L)` qui prend en paramètre une liste de 0 et de 1 codant un nombre en binaire et qui renvoie la valeur du nombre correspondant en base 10. On attend le résultat suivant :

```
>>>bin2dec([1,0,1])
5
```

Définition :
La fonction `chr(int)` associe à un entier `int` compris entre 0 et 255 le caractère dont le codage ASCII est `int` :

```
>>>chr(42)
'*'
```

Q 4 - Ecrire une fonction `phrase_ascii(L)` qui à partir d'une liste de nombres, renvoie une chaine de caractère.

```
>>>phrase_ascii([77, 80, 83, 73])
'MPSI'
```

Q 5 - Proposer une fonction `phrase(img)` qui retourne la phrase secrète cachée dans l'image.

III. Codage entropique - Algorithme de Rice

Le codage de Rice est particulièrement adapté à la compression de données entières lorsque :

- les entiers sont codées sur un octet ;
- les valeurs les plus faibles codées sont celles qui ont la probabilité d'apparition la plus élevée.

Il est également avantageux lorsque la vitesse d'exécution de l'algorithme est un paramètre important.

Le principe du codage d'un entier N avec l'algorithme de Rice de paramètre `p` est le suivant :

- l'entier N à coder est divisé par 2^p (division entière) ;
- le quotient q de la division entière est codé avec un codage unaire (q occurences de 1 suivies d'un 0, voir tableau), ce qui constitue la première partie du code de Rice ;
- le reste r de la division entière est codé en binaire sur p bits, ce qui constitue la seconde partie du code de Rice.

Le tableau de la page suivante illustre le codage des entiers 0 à 7 avec un codage de Rice de paramètre p = 2.

Q 6 - Calculer les valeurs associées à 1,4 et 10 par le code de Rice avec le paramètre $p = 3$.

Q 7 - Définir la suite d'instructions de la fonction `codage1(nbre,p)` permettant d'obtenir une chaine de caractère `code1` associée à la première partie du code de Rice (codage unaire du quotient).

```
>>>codage1(11,2)
'110'
```

Q 8 - Définir la suite d'instructions de la fonction `codage2(nbre,p)` permettant d'obtenir une chaine `code2` associée à la seconde partie du code de Rice (codage binaire du reste). Pour tester le code, essayer de reproduire la première partie codage de Rice présente dans le tableau ci-dessous.

Q 9 - Proposer une fonction `rice(nbre,p)` qui renvoie le code correspondant. Pour tester le code, on essayera de reproduire le deuxième tableau du codage de Rice présenté ci-dessous.

Décimal	Code unaire
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	1111110
7	11111110

Décimal	Rice $p = 2$
0	0 00
1	0 01
2	0 10
3	0 11
4	10 00
5	10 01
6	10 10
7	10 11

FIGURE 2 – Codage unaire des entiers 0 à 7 Décimal, Codage de Rice de paramètre $p = 2$