MASQUAGE D'UN TEXTE DANS UNE IMAGE

PRINCIPE

Dans une image couleur (tableau 3D), chaque pixel est représenté par un tableau 1D de 3 entiers compris entre 0 et 255 (niveaux de rouge, vert et bleu). Or, la modification du bit de poids faible (le dernier) pour un niveau de couleur change très peu cette couleur. On peut donc modifier tous ces bits dans une image, sans trop changer son apparence, pour y cacher de l'information comme un texte.

Plus précisément, on utilisera un niveau de couleur (trois disponibles par pixel dans une image couleur) pour cacher un bit suivant le principe de codage suivant :

- Si le niveau de couleur est pair (dernier bit égal à 0), alors
  - On conserve sa valeur pour y cacher le bit 0
  - On lui ajoute 1 pour y cacher le bit 1
- Si le niveau de couleur est impair (dernier bit égal à 1), alors
  - On conserve sa valeur pour y cacher le bit 1
  - On lui ajoute 1 pour y cacher le bit 0

Ainsi, un niveau de couleur codé fournit le bit 0 s'il est pair et le bit 1 s'il est impair. Le décodage (récupération de l'information cachée) est alors assuré.

Remarque : si le niveau de couleur est égal à 255, on ne peut lui ajouter 1. On réalisera donc un prétraitement de l'image pour transformer tous les 255 en 254.

Exemple : le tableau suivant réalise le codage et le décodage du 100^{ième} caractère dans le code ASCII à savoir la lettre « d » : 01100100. Huit niveaux de couleurs sont alors nécessaires pour cacher les **huit bits** en question.

Codage								
Niveau de couleur initial	255	255	253	145	0	65	35	255
Niveau de couleur après prétraitement	254	254	253	145	0	65	35	254
Bit à cacher	0	1	1	0	0	1	0	0
Niveau de couleur codé	254	255	253	146	0	65	36	254
Décodage								
Test de parité	pair	impair	impair	pair	pair	impair	pair	pair
Bit récupéré	0	1	1	0	0	1	0	0

OBJECTIF

Construire :

- une fonction de codage qui permet de cacher un texte dans une image couleur suffisamment grande
- une fonction de décodage qui, étant donnée une image codée, permet de récupérer le texte caché connaissant sa longueur.

On utilisera :

- L'image couleur « SuperHEI.png »
- Le texte « fab.txt »

COMPLEMENTS PYTHON

```
#%% Passage du caractère au code ASCII (en base 10)
ord('d')
#%% Passage de la base 10 à la base 2
bin(100)
#%% Passage de la base 2 à la base 10
int('0b1100100',2)
#%% Passage du code ASCII (en base 10) au caractère
chr(100)
#%% Ajout d'un élément dans un tableau
import numpy as np
tab = np.array([],dtype='int8')
np.append(tab,[1,2])
#%% "Aplatir" un tableau multidimensionnel
tab = np.array([[[5,5,3],[5,0,5],[2,5,0]],[[5,5,3],[5,0,5],[2,5,0]]])
tab.flatten()
```

MARCHE À SUIVRE

1. Déterminer le nombre maximal de caractères qu'il est possible de cacher dans l'image « SuperHEI.png ».
2. Déterminer le nombre de caractères (à cacher) contenus dans le fichier « fab.txt ».
3. Déclarer une fonction **pretraitAplat** définie de la manière suivante :
   - **Entrée** : une image couleur (tableau 3D)
   - **Sortie** : l'image aplatie et prétraitée (tableau 1D sans 255)
4. Déclarer une fonction **codageCar** définie de la manière suivante :
   - **Entrées** : un tableau 1D de 8 niveaux de couleurs, et un caractère
   - **Sortie** : le tableau 1D des 8 niveaux de couleurs codés
5. Déclarer une fonction **decodageCar** définie de la manière suivante :
   - **Entrée** : un tableau 1D de 8 niveaux de couleurs codés
   - **Sortie** : le caractère caché
6. Déclarer une fonction **codage** définie de la manière suivante :
   - **Entrées** : un tableau 1D de niveaux de couleurs, et une chaîne de caractères (pas trop grande)
   - **Sortie** : le tableau 1D des niveaux de couleurs codés
7. Déclarer une fonction **decodage** définie de la manière suivante :
   - **Entrées** : un tableau 1D de niveaux de couleurs codés, et un entier (longueur de la chaîne cachée)
   - **Sortie** : la chaîne de caractères cachée
8. Représenter, dans une seule fenêtre, l'image initiale « superHEI.png » et l'image codée (contenant le texte de « fab.txt »).
9. Récupérer le texte caché dans l'image codée puis l'écrire dans un nouveau fichier « txtRecupere.txt » enregistré sur le bureau.

Rappel : chaque fonction doit disposer d'une ligne de test délimitée par #%%