Compression RLE d'images en niveaux de gris

Sujet n°1 — Corrigé

Question 1 *Le codage RLE raccourcit-il toujours la liste ?*

Non. Chaque suite de valeurs identiques devient deux nombres. Si aucune valeur n'est répétée deux fois de suite, la liste codée est **deux fois plus longue** : `[1, 2, 3]` donne `[1, 1, 1, 2, 1, 3]`. Le codage n'est efficace que pour des images à grands aplats.

Question 2 *Décodage*

La liste codée alterne compte et valeur : on la parcourt de deux en deux.

```
def decodage_rle(liste_rle):
    liste_octets = []
    for i in range(0, len(liste_rle), 2):
        occ = liste_rle[i]
        valeur = liste_rle[i + 1]
        for _ in range(occ):
            liste_octets.append(valeur)
    return liste_octets
```

Question 3 *Deux images*

`bac_nsi_32.png` (1024 pixels) est retrouvée à l'identique. Pour `bac_nsi_256.png` (65 536 pixels), l'image décodée est **tronquée** : elle ne compte plus que 51 456 pixels. **Cause** : la liste codée est écrite octet par octet dans le fichier `.rle` ; `enregistrer_octets` ramène toute valeur supérieure à 255 à 255. Or la grande image contient une suite de 5842 pixels identiques : le compte est faussé et les pixels correspondants sont perdus au décodage.

Question 4 *Limiter les suites à 255*

Démarche : un compte doit tenir sur un octet. Au codage, on coupe une suite de plus de 255 valeurs identiques en plusieurs couples ; le décodage n'a pas à changer, puisqu'il recopie simplement chaque couple.

```
def codage_rle(liste_octets):
    liste_rle = []
    i = 0
    while i < len(liste_octets):
        valeur = liste_octets[i]
        occ = 1
        while i + occ < len(liste_octets) and liste_octets[i + occ] == valeur \
                and occ < 255:
            occ += 1
        liste_rle.append(occ)
        liste_rle.append(valeur)
        i += occ
    return liste_rle
```

`codage_rle([7] * 300)` vaut `[255, 7, 45, 7]`. La grande image est alors codée sur 9120 octets et retrouvée à l'identique.