Algorithmique

Diviser pour régner

01

Exercice 1 — Trouver le nombre manquant

Principe

On dispose d'une liste **triée** contenant tous les entiers de 1 à n sauf un. L'objectif est de retrouver le nombre manquant en appliquant la méthode **diviser pour régner**.
Propriété clé : pour un indice i, si `liste[i] == i + 1` alors aucun nombre ne manque dans la portion `liste[0..i]`. Sinon, le nombre manquant se trouve dans la moitié gauche.

Question 1 — `trouver_manquant(liste, debut, fin)`

Écrire une fonction récursive `trouver_manquant(liste, debut, fin)` qui retourne le nombre manquant dans la portion `liste[debut..fin]`.

Python

```
>>> liste = [1, 2, 4, 5, 6, 7, 8]
>>> trouver_manquant(liste, 0, len(liste) - 1)
3

>>> liste = [2, 3, 4, 5, 6]
>>> trouver_manquant(liste, 0, len(liste) - 1)
1
```

Réponse

.....

.....

.....

.....

.....

.....

. . . . . .

. . . . . .

Exercice 2 — Tri fusion

Principe — Diviser pour régner

Le tri fusion suit les trois phases de la méthode :

1. **Diviser** : découper le tableau en deux sous-tableaux de taille (environ) égale, jusqu'à obtenir des tableaux de taille 1.
2. **Régner** : un tableau d'un seul élément est déjà trié.
3. **Combiner** : fusionner deux sous-tableaux triés pour produire un tableau trié.

Slicing

En Python, le **slicing** permet d'extraire une portion d'une séquence. Si `seq` est une séquence, `seq[a:b]` contient les éléments d'indice $\geq a$ et $< b$. On peut omettre une borne : `seq[:k]` donne les k premiers éléments, `seq[k:]` donne tous les éléments à partir de l'indice k.

Question 2 — `diviser(tab)`

Écrire une fonction `diviser(tab)` qui prend en paramètre un tableau `tab` et renvoie deux tableaux : le premier contient les éléments d'indice strictement inférieur au quotient de la division euclidienne de `len(tab)` par 2, le second contient le reste.

Exemples

```
>>> T1, T2 = diviser([1, 2, 3, 4, 5])
>>> T1
[1, 2]
>>> T2
[3, 4, 5]

>>> T1, T2 = diviser([1, 2, 3, 4, 5, 6])
>>> T1
[1, 2, 3]
>>> T2
[4, 5, 6]
```

Réponse

. . . . . .

. . . . . .

. . . . . .

. . . . . .

pop()

La méthode `pop` retire et renvoie un élément d'une liste. Sans paramètre, elle retire le **dernier** élément. Avec un entier n, elle retire l'élément à l'indice n.

Exemple

```
tab = [1, 2, 3]
elmt = tab.pop(0)    # tab vaut [2, 3], elmt vaut 1
```

Question 3 — `fusion(tab1, tab2)`

Écrire une fonction `fusion(tab1, tab2)` qui prend deux tableaux **triés** en paramètres et renvoie un troisième tableau trié contenant tous leurs éléments.

Exemples

```
>>> fusion([1, 8, 14], [2, 3, 9, 17])
[1, 2, 3, 8, 9, 14, 17]

>>> fusion([133, 145, 255], [72, 124, 169, 213])
[72, 124, 133, 145, 169, 213, 255]
```

Réponse

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Question 4 — `tri_fusion(tab)`

À l'aide des fonctions **`diviser`** et **`fusion`** définies précédemment, écrire une fonction récursive **`tri_fusion(tab)`** qui prend en paramètre un tableau **`tab`** et retourne un tableau trié contenant l'ensemble de ses éléments.

Exemples

```
>>> tri_fusion([1, 8, 14, 2, 9, 3, 17])
[1, 2, 3, 8, 9, 14, 17]

>>> tri_fusion([498, 487, 127, 199, 332])
[127, 199, 332, 487, 498]
```

Réponse

Exercice 3 — Rotation d'image

Représentation d'une image

Une image carrée de taille $n \times n$ est représentée par une **liste de n listes**, chaque sous-liste contenant n entiers (pixels).

Exemple — image 4×4

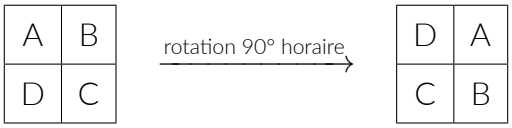
```
image = [[ 1,  2,  3,  4],
         [ 5,  6,  7,  8],
         [ 9, 10, 11, 12],
         [13, 14, 15, 16]]
```

Pour extraire un quadrant (exemple : haut-gauche de taille $k \times k$) :

```
k = len(image) // 2
A = [ligne[:k] for ligne in image[:k]]
```

Question 5 — `rotation(image)`

On souhaite faire tourner une image carrée de taille $2^n \times 2^n$ de **90° dans le sens horaire** en appliquant la méthode **diviser pour régner**.
L'algorithme divise l'image en 4 quadrants A, B, C, D :



Chaque quadrant est lui-même tourné récursivement avant d'être replacé. Le cas de base est une image 1×1 (déjà tournée).
Écrire la fonction `rotation(image)`.

Exemples

```
>>> rotation([[1, 2], [3, 4]])
[[3, 1], [4, 2]]

>>> rotation([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])
[[13, 9, 5, 1], [14, 10, 6, 2], [15, 11, 7, 3], [16, 12, 8, 4]]
```

Réponse

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .