TP12 - Algorithmes de tris

On souhaite écrire plusieurs algorithmes de tris.
Pour cela, on *définit une liste par compréhension* de 20 entiers . Puis on mélange la liste à l'aide de la fonction `shuffle` du module `random`

```
from random import shuffle
tab = [k for k in range(1, 21)]
print(tab)
shuffle(tab)
print(tab)
```

Q1. Ecrire une fonction `est_trie(t)` qui a pour paramètre un tableau et qui renvoie `True` si le tableau est trié et `False` sinon.

I Tri à bulles

On considère un tableau `t`, de taille `n`.

Principe :

- On parcourt une première fois le tableau, et dès que deux éléments consécutifs ne sont pas ordonnés, on les échange.
  Ainsi à la fin du premier passage, le plus grand élément se situe en fin de tableau.
- On recommence un tel passage, en s'arrêtant à l'avant-dernier élément, et ainsi de suite.
- Au i-ème passage on fait remonter le i-ème plus grand élément du tableau à sa position définitive, un peu à la manière de bulles qu'on ferait remonter à la surface d'un liquide, d'où le nom d'algorithme de tri à bulles.

Q2. Ecrire une fonction `tri_bulle(t)` qui a pour paramètre un tableau `t` et qui trie le tableau. Cette fonction ne renvoie rien mais modifie le paramètre `t`.
Tester la fonction.
Quelle est sa complexité ?

II Tri fusion

Le tri fusion a déjà été vu en cours.

Le tri fusion fait partie de la famille des algorithmes « diviser pour régner ».

Principe

a) On divise le tableau en deux sous-tableaux de taille sensiblement égale.

b) On règne sur chacun des deux sous-tableaux en les triant.

c) On combine les deux sous-tableaux triés en les fusionnant pour obtenir un tableau trié.

On rappelle :

```
t = [2, 4, 6, 8, 10]
t1 = t[1 : 3]
print(t1)
```

Il s'affiche :

Q3. Ecrire une fonction (récursive) `fusionner` qui a pour paramètres deux tableaux `t1` et `t2` déjà triés et qui renvoie un tableau trié correspondant à la fusion de `t1` et `t2`.
Tester la fonction avec deux listes triées.

Q4. Ecrire la fonction (récursive) `tri_fusion` qui a pour paramètre un tableau `t` et qui renvoie le tableau trié par la méthode « Diviser pour régner ».
Tester la fonction.
Remarque : si le tableau contient 0 ou 1 élément, on peut dire qu'il est déjà trié.

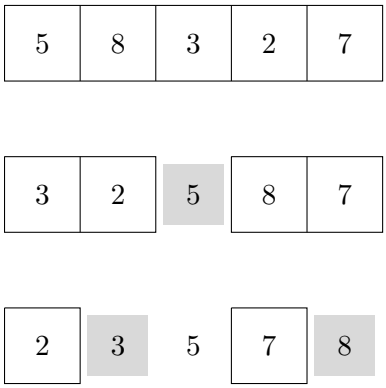
III Tri rapide

Le tri rapide fait partie de la famille des algorithmes « diviser pour régner ».

Principe

a) Le 1er élément du tableau est appelé « pivot ».
   On divise le tableau en deux sous-tableaux :
   — à gauche toutes les valeurs inférieures ou égales au pivot ;
   — à droite toutes les valeurs supérieures au pivot.
b) On règne sur chacun des deux sous-tableaux en les triant.
c) On n'a pas besoin de combiner les deux sous-tableaux.

Exemple



Q5. Ecrire une fonction (non récursive) `partition(t)` qui a pour paramètre un tableau `t` et qui renvoie le pivot et les deux sous-tableaux correspondant à l'algorithme du tri rapide.

Q6. Ecrire une fonction (récursive) `tri_rapide(t)` qui renvoie le tableau trié.
Tester la fonction.

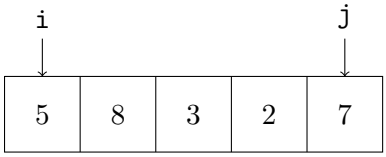
On souhaite réécrire les deux fonctions précédentes mais sans créer de nouveaux tableaux.
Pour cela, on écrit une fonction `partition2` qui a pour paramètre un tableau `t` et `d` et `f` l'indice de début et de fin du tableau qu'on souhaite partitionner. Cette fonction modifie le tableau `t` en

positionnant à gauche du pivot les valeurs inférieures ou égales à celui-ci et à droite celles supérieures au pivot. Cette fonction renvoie l'indice de la nouvelle position du pivot.
Par exemple `partition2([5, 8, 3, 2, 7], 0, 4)` modifie le tableau et renvoie 2.
Pour écrire cette fonction, on parcourt le tableau à l'aide de deux indices :

— l'indice `i` qui démarre à `d` et qui augmente ;

— l'indice `j` qui démarre à `f` et qui diminue.

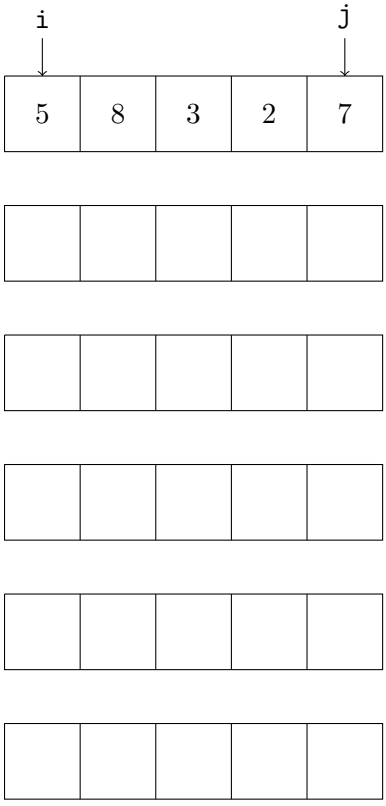
Remarque : l'indice `i` sera toujours l'indice du pivot.



On compare `t[i+1]` et le pivot `t[i]`

- `t[i+1]` doit être à gauche de `t[i]`, alors on échange les 2 et on ...
  (on change i ou j ? ? ?)
- `t[i+1]` doit être à droite de `t[i]`, alors on échange `t[i+1]` avec `t[j]` et on ...
  (on change i ou j ? ? ?)

Quand s'arrête-t-on ? ...



Q7. Ecrire la fonction `partition2(t, d, f)` et la tester.
On fera une version non récursive et une version récursive.

Q8. Ecrire la fonction (récursive) `tri_rapide2` qui a pour paramètres un tableau `t` et `d` et `f` l'indice de début et de fin du tableau qu'on souhaite trier. Cette fonction modifie le tableau `t` et ne renvoie rien.
Tester la fonction.