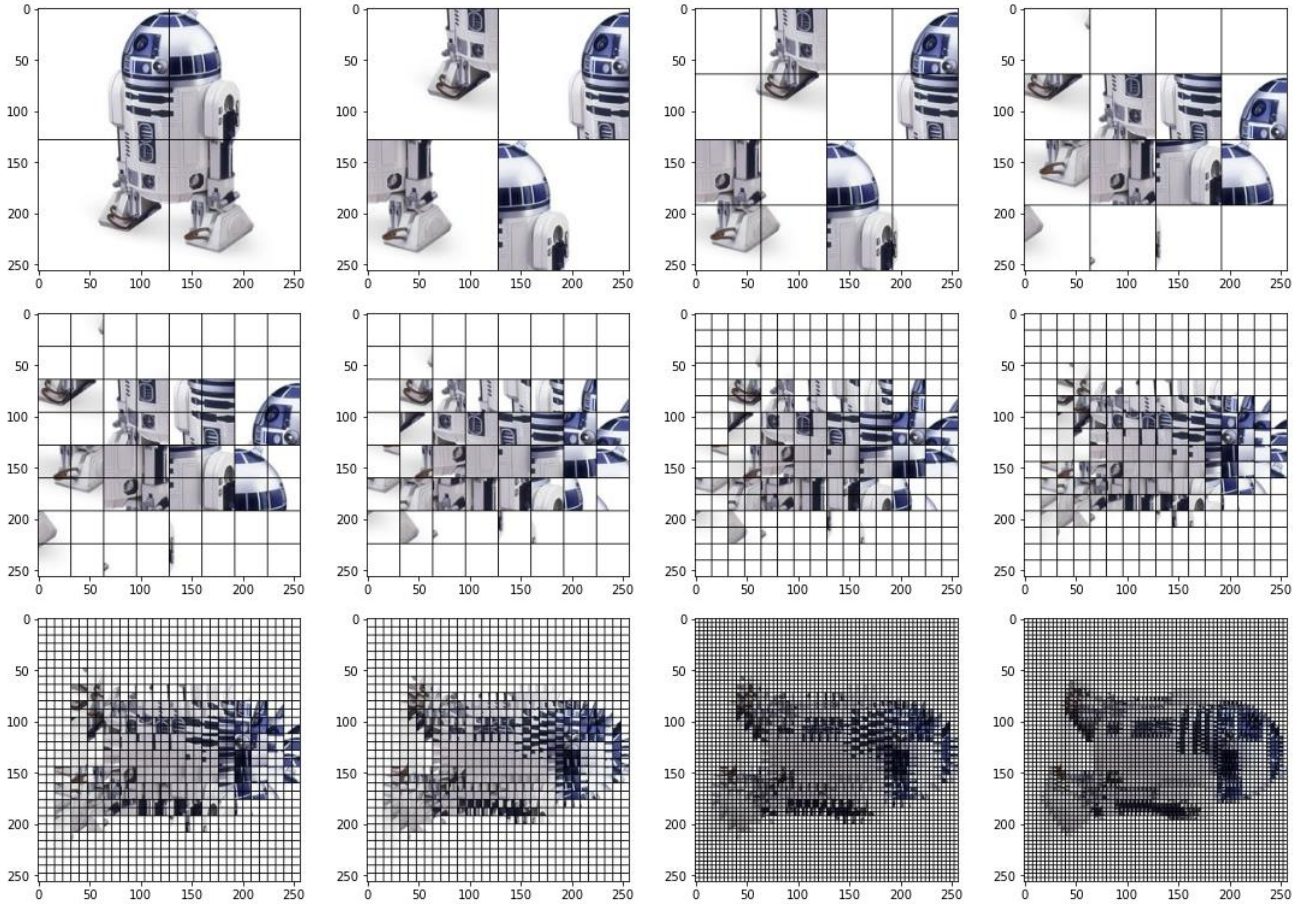
Méthode « diviser pour régner »
Rotation d'une image carrée d'un quart de tour

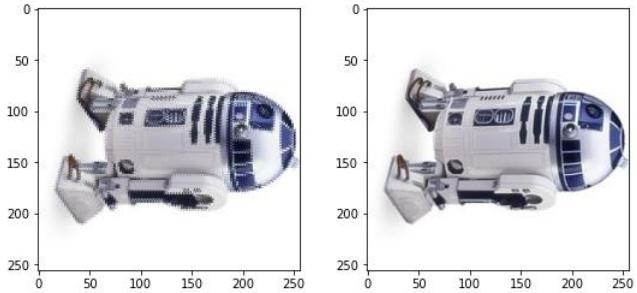
Présentation de la méthode « diviser pour régner »
La méthode « diviser pour régner » (*divide-and-conquer* en Anglais) consiste à **découper un problème en sous-problèmes similaires** (d'où l'algorithme récursif résultant) afin de réduire la difficulté du problème initial. On espère casser récursivement le problème en sous-problèmes de plus en plus petits jusqu'à obtenir des cas simples permettant une résolution directe.
L'expression provient du latin « *divide ut imperes* » et désignait une stratégie militaire (ou politique).
La **recherche dichotomique** dans une liste triée fournit un premier d'utilisation de cette méthode.

Application de la méthode « diviser pour régner » pour effectuer une rotation d'un quart de tour
Méthode valable sur les images carrées dont le côté est une puissance de 2.

Pour effectuer une rotation d'un quart de tour, on peut diviser l'image en 4, effectuer une permutation circulaire des quatre quadrants, puis appliquer récursivement le même processus à chaque quadrant, jusqu'à obtenir des quadrants contenant un seul pixel.



On poursuit sans représenter le quadrillage devenu trop fin pour les deux dernières étapes.



Implémentation en Python
Le code suivant applique ce procédé, mais pas dans le même ordre que sur les images précédentes…

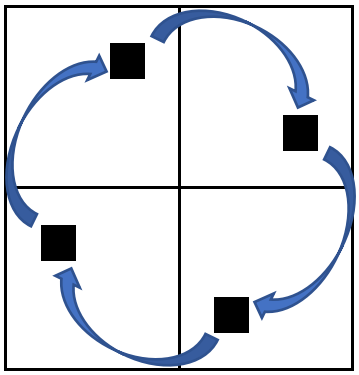
```
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

def rotate(image):
    n = len(image)
    if n > 1:
        temp = np.copy(image[n//2:, :n//2])
        image[n//2:, :n//2] = rotate(image[n//2:, n//2:])
        image[n//2:, n//2:] = rotate(image[:n//2, n//2:])
        image[:n//2, n//2:] = rotate(image[:n//2, :n//2])
        image[:n//2, :n//2] = rotate(temp)
    return image

image = np.array(Image.open('R2D2.jpg'))
image = rotate(image)
plt.imshow(image)
```

Cet algorithme est très joli, mais peu efficace.
Sur l'exemple de taille 256x256, chaque pixel est déplacé 8 fois, alors qu'un seul déplacement est suffisant.

Rotation d'un quart de tour d'une image carrée, positionnement direct des pixels
Méthode valable sur toutes les images carrées.
Lors du déplacement de chaque pixel dans le quadrant suivant, on peut le placer directement à la bonne position et éviter les appels récursifs de la méthode précédente.



Recopier et compléter le code ci-dessous pour appliquer une rotation d'un quart de tour à l'image représentée par le tableau de nombres *image*.

```
def rotate(image):
    n = len(image)
    for i in range(n//2):
        for j in range(n//2):
            # partie à compléter pour déplacer
            # les pixels des quatre quadrants
    return image
```

Tester votre code avec l'image vador.jpg.

Rotation d'un quart de tour d'une image quelconque

Dans le cas d'une image quelconque (rectangulaire), on ne peut plus déplacer les pixels sur la même image.
On doit créer un tableau aux nouvelles dimensions pour accueillir la nouvelle image.
Le programme est alors plus simple, mais nécessite de dupliquer l'image en mémoire.

```
def rotate(image):
    dim = list(image.shape)
    dim[0], dim[1] = dim[1], dim[0]
    nimage = np.empty(dim, np.uint8)
    n, m = nimage.shape[0:2]
    for i in range(n):
        for j in range(m):
            nimage[i, j] = image[m - 1 - j, i]
    return nimage
```

On peut également utiliser scipy.ndimage.rotate (après import scipy).