TP : Quart de tour d'une image

L'objectif est d'écrire un programme en Python qui réalise la rotation d'une image carrée d'un quart de tour.

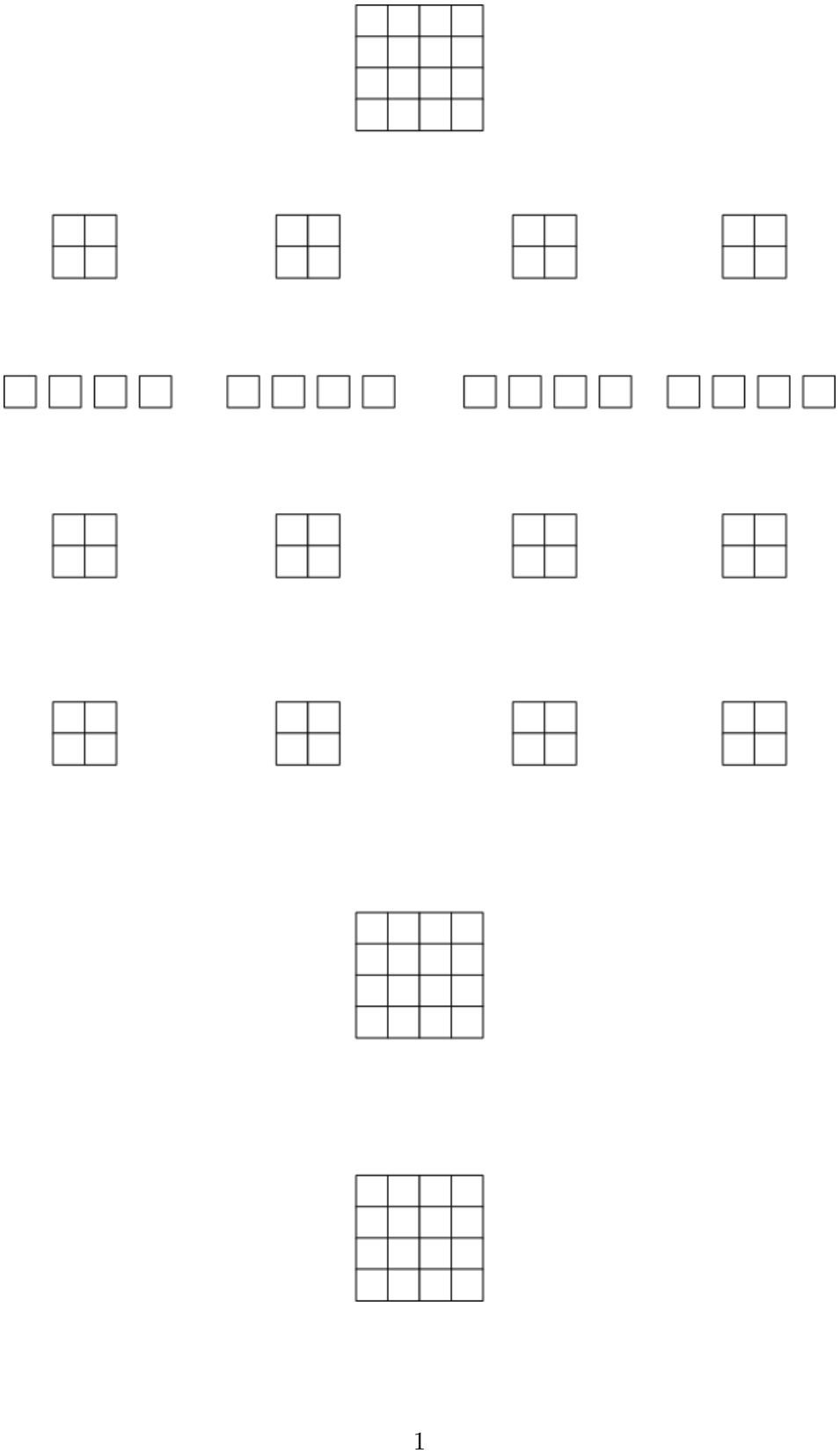
Notre image carrée doit avoir une des dimensions qui sont des puissances de 2. Par exemple 2, 4, 8, 16, 32, ... pixel de côté.

L'algorithme applique le principe de **diviser pour régner**.

Commençons par dessiner

On se restreint à une image carrée de 4 pixel de côté pour comprendre le principe.

On va compléter la figure ci-dessous et faire tourner d'un quart de tour notre image.



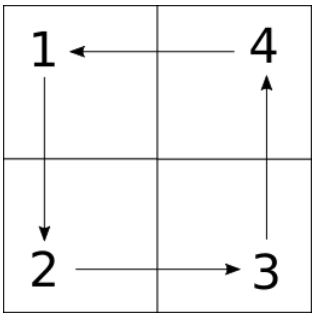
Algorithme diviser pour régner

L'algorithme pour tourner une image carrée revient à :

1) Diviser l'image en 4 images carrées dont les dimensions sont obtenues en divisant la dimension de l'image initiale par 2 ;

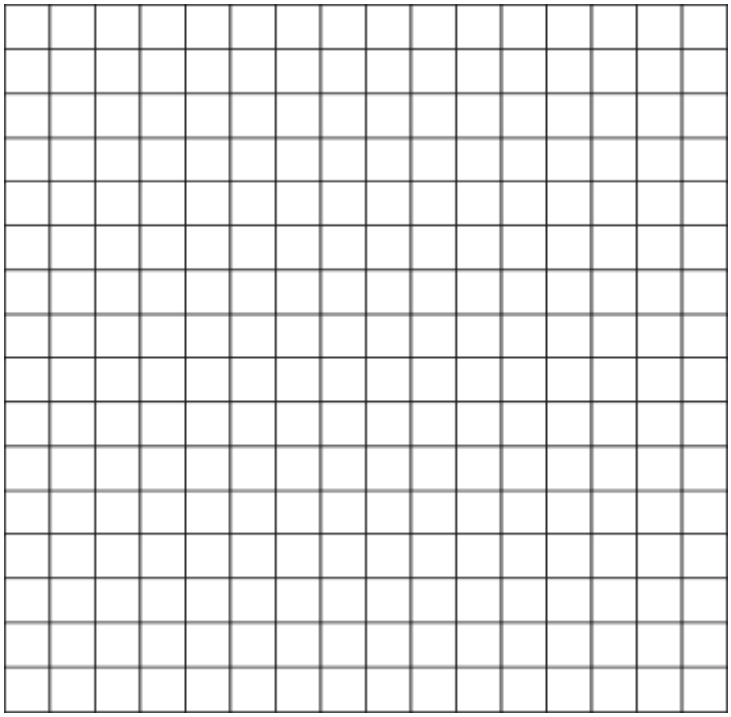
2) Quand on arrive à une image de 1 pixel de côté, la division s'arrête ;

3) On reconstruit l'image, carré par carré en les décalant (permutation circulaire) comme le montre la figure ci-dessous.



Appliquons ce principe sur un pixel d'une image carrée de 16 pixel de côté.

- Choisir un pixel et le colorier en bleu ;
- Diviser l'image en carrés en convergeant vers le pixel bleu ;
- Recomposer l'image en décalant à chaque étape le carré contenant le pixel bleu.



Code en Python

Python dispose d'un module PIL qui permet la manipulation d'image. Ce module permet de récupérer sous forme de tableau chaque pixel de l'image. Les coordonnées (-1) d'un pixel dans l'image correspondent aux indices de position dans le tableau.

Créer un fichier nommé quart_tour.py et télécharger l'image tux256.png dans le même dossier.

1) On donne ci-dessous les instructions pour créer le tableau contenant chaque pixel de l'image.

```
1  from PIL import Image
2
3  # création de la variable img associée à une image
4  # img est un objet de type PIL
5  img = Image.open("../img/tux256.png")
6
7  # on récupère les dimensions de l'image avec l'attribut size de img
8  largeur,hauteur = img.size
9
10 # on crée une variable px avec la méthode load()
11 # la variable px est un tableau contenant chaque pixel de l'image
12 # chaque pixel est un triplet de couleurs RGB
```

Ajouter ces instructions au début de votre programme. Attention au chemin du fichier image !

2) Dans la console ou interpréteur Python :

a) Afficher les dimensions de l'image.

b) Afficher le premier pixel de votre image. Interpréter la valeur affichée.

3) Le quart de tour d'une image revient à :

- découper l'image en 4 carrés de même dimension t ;
- décaler chaque pixel d'un carré dans un carré adjacent.

Soit i et j les indices d'un pixel de l'image positionné dans le carré 1. On remplace ce pixel par un pixel du carré 4 adjacent en effectuant un décalage égal à la dimension d'un carré.

a) Quels sont les indices du pixel du carré 4 qui va remplacer le pixel du carré 1 ?

b) Quelle est l'instruction Python à écrire pour effectuer le décalage du pixel du carré 4 vers le pixel du carré 1 ?

c) Quelles sont les instructions en Python pour décaler les autres pixel des autres carrés ?

d) Quelle est l'instruction Python qui décale tous les pixel de l'image dans les 4 carrés ?

4) La deuxième étape consiste à diviser l'image en 4 images carrées plus petites et à effectuer les décalages des pixel de chaque carré.

Cette étape revient à effectuer 4 appels récursifs avec chaque carré de l'image.

On donne un extrait du code en Python :

```
def quart_de_tour(px,x,y,t):
    """
    - px : tableau contenant les pixel de l'image
    - x,y : indices du premier pixel d'une image carrée
    - t : côté d'une image carrée
    """
    if t>1:
        # on divise la taille image par 2
        t=t//2
        # on tourne le quart haut gauche de l'image

        quart_de_tour(px,.....................)
        # on tourne le quart haut droit de l'image

        quart_de_tour(.......................)
        # on tourne le quart bas droit de l'image

        quart_de_tour(.......................)
        # on tourne le quart bas gauche de l'image

        quart_de_tour(.......................)
```

a) Compléter les appels récursifs de la fonction correspondant à la division du problème

b) Compléter la fonction quart_de_tour avec le code avec les décalages des pixel correspondant à la solution de chaque sous-problème.

c) Exécuter votre code en ajoutant un appel à la fonction quart_de_tour.

d) Ajouter à la fin de votre programme l'instruction img.show() pour afficher votre image et ainsi vérifier le quart de tour de celle-ci.