

# 1 Problématique

La rotation d'une image est une fonctionnalité proposée par n'importe quel logiciel de retouche tel *Gimp*. L'opération n'est cependant pas triviale et peut demander une durée non négligeable.

Construire un algorithme de rotation d'une image en appliquant le principe de « diviser pour régner ».

## 2 Principe

Un algorithme de type « diviser pour régner » se décompose en trois parties :

- *diviser* : Le problème est partagé en plusieurs petits problèmes identiques.
- *traitement* : Chaque petit problème est résolu.
- *recombinaison* : Les petits problèmes résolus sont assemblés pour remonter au problème principal.

**Activité 1 : Réflexion commune** : Considérons une image aux dimensions connues. Quelles étapes pourrions-nous imaginer pour répondre à notre problématique ?

## 3 Algorithme de rotation

### 3.1 Chargement de l'image

*PIL* (*Python Image Library*) -anciennement *pillow*- est une bibliothèque de traitement d'image. Le code ci-après charge et affiche une image :

```
1 from PIL import Image
2
3 im = Image.open("image.png")
4 im.show()
```

Afin de travailler sur l'image nous pouvons récupérer des informations :

```
1 largeur, hauteur = im.size
2 px = im.load()
```

La variable *px* est une matrice représentative des pixels de l'image. La couleur du pixel de coordonnées  $(x,y)$  est donnée par l'instruction `px[x,y]`. Il est également possible d'affecter une nouvelle couleur *c* à un pixel : `px[x,y] = c`.

**Activité 2** : Charger et afficher une image carrée. Il est possible de récupérer cette image sur <https://www.freepng.fr/>.

### 3.2 Application de l'algorithme *Diviser pour régner*

#### Activité 3 :

1. Écrire une double boucle qui effectue la rotation de quatre pixels dans le sens anti-horaire.
2. Écrire une fonction `rotation_auxiliaire(px : object, x : int, y : int, t : int) → None` qui applique l'algorithme récursif exprimé dans l'activité 1. Inclure la boucle de la question précédente dans cette fonction.
3. Écrire la fonction `rotation(px : object, taille : int) → None` qui effectuera l'appel principal de la fonction précédente.

Plutôt qu'une seconde fonction nous pourrions donner des valeurs par défaut à x et y.  
`im.save("fichier.png")` pour sauvegarder le fichier.

## 4 Créer une bibliothèque

Afin de pouvoir réutiliser la fonctionnalité de rotation développée, il peut être intéressant de l'intégrer dans une classe.

### Activité 4 :

1. Créer une classe **Image\_lib** et son constructeur qui demande un argument : le chemin du fichier source.
2. Créer les méthodes **montrer()**  $\rightarrow$  **None** et **sauvegarder(nom : str)**  $\rightarrow$  **None**.
3. Adapter les fonctions *rotation* et *rotation\_auxiliaire* pour en faire des méthodes. La fonction *rotation\_auxiliaire* peut être vue comme une fonctionnalité interne à la classe, invisible pour l'utilisateur.
4. Adapter la fonction pour pouvoir proposer une rotation horaire ou anti-horaire.

fonction **filtrer** pour les plus avancés