TP08 TNSI	Algorithmique – 3 – Méthode « Divide and Conquer » Rotation d'image

I - Rotation d'un quart de tour d'une image

L'objectif de ce problème est d'implémenter des algorithmes permettant de faire tourner une image d'un quart de tour dans le sens des aiguilles d'une montre.
Une image est un tableau (bidimensionnel) composé de pixels. Un pixel est, dans le cadre du codage RGB, un triplet d'entiers compris entre 0 et 255 correspondant aux intensités lumineuses respectives du rouge *(red),* du vert *(green)* et du bleu *(blue).* Par exemple, un pixel de valeur *[0, 0, 0*] est noir ; un pixel de valeur *[255, 255, 255]* est blanc ; un pixel de valeur *[0, 255, 0]* est vert et un pixel de valeur *[255, 255, 0]* est jaune.

Dans cet exercice, on exploitera une image carrée comportant 2^n lignes et colonnes.

A - Python et les images

Nous utilisons le module PIL qui permet de gérer aisément les images. La partie qui nous intéresse de ce module peut être importée à l'aide de :

```
from PIL import Image
```

On ouvre alors l'image qui se trouve dans le même répertoire que le fichier à l'aide de :

```
img = Image.open (nom du fichier)
```

Les dimensions de l'image peuvent être récupérées à l'aide de :

```
largeur, hauteur = img.size
```

La valeur du pixel d'abscisse x et d'ordonnée y peut être récupérée à l'aide de

```
img.getpixel((x, y))
```

On rappelle que la première coordonnée correspond à l'abscisse dans l'image et la seconde à l'ordonnée, l'origine du repère étant en haut à gauche de l'image, les axes allant vers la droite et vers le bas.
On peut affecter le triplet de couleur (r, g, b) au pixel de coordonnées (x, y) à l'aide de :

```
img.putpixel((x, y), (r, g, b))
```

L'image peut être sauvegardée à l'aide de :

```
img.save(nom_du_fichier)
```

Elle peut être affichée à l'aide de (deux utilisations peuvent être nécessaires avant d'obtenir le bon fonctionnement sur les windows capricieux) :

```
img.show()
```

Enfin, une nouvelle image (noire par défaut, elle sera modifiée par la suite) peut être créé à l'aide de :

```
im = Image. new( "RGB", (largeur, hauteur))
```

B - Algorithme naïf de rotation d'un quart de tour

1 - Ouvrir l'image mario.png, vérifier sa taille.

2 - Créer une nouvelle image, de même format, destinée à stocker l'image obtenue par rotation.

3 - Pour une image img carrée de taille n x n, l'image img_rot obtenue par rotation d'un quart de tour dans le sens des aiguilles d'une montre a pour pixel (x, y) celui initialement en (y, n - 1 - x) dans img. Écrire une suite d'instructions qui stocke dans la nouvelle image celle obtenue par rotation de la précédente et qui affiche le résultat.

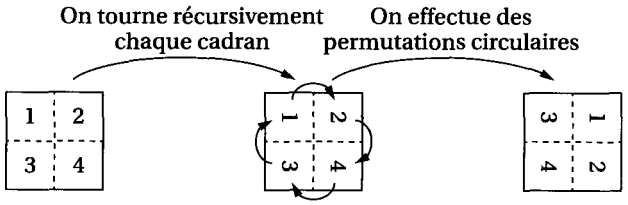
4 - Quelle est la complexité temporelle de cet algorithme ? Et sa complexité spatiale ? (C'est-à-dire l'ordre de grandeur de la taille mémoire utilisée, en fonction de n, la taille du côté de l'image.)

C - Rotation d'un quart de tour d'une image en espace mémoire constant

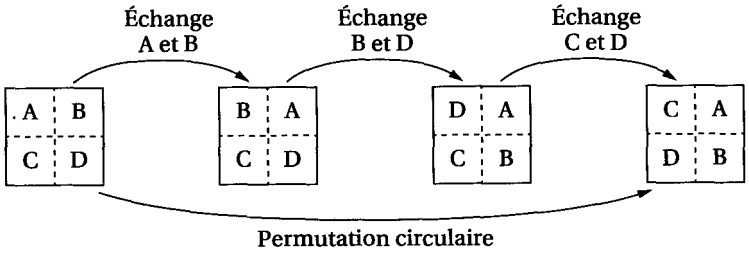
La suite de ce problème a pour objet de réaliser un algorithme permettant d'effectuer une rotation d'un quart de tour d'une image sans utiliser d'espace mémoire supplémentaire. Cet algorithme, lent, n'est pas utilisé en pratique mais illustre un apport possible des méthodes diviser pour régner.

1 - Écrire une procédure echange_pix(image, x0, y0, x1, y1) qui échange les pixels de coordonnées (x0, y0) et (x1, y1) de l'image passée en paramètre.

On adopte dans cet exercice une stratégie diviser pour régner. L'image est divisée en quatre quadrants. Chaque cadran est tourné récursivement puis une permutation circulaire des quadrants est effectuée :



La permutation circulaire est réalisée en enchaînant plusieurs échanges de cadran. Le schéma suivant présente un exemple de stratégie permettant de le faire :



2 - Écrire une procédure echange_quadrant(image, x0, y0, x1, y1, n) qui échange deux quadrants carrés de taille n dont le premier a pour coordonnées de départ (x0, y0) et le second pour coordonnées de départ (x1, y1).

3 - Écrire une procédure récursive tourne_quadrants (image, x0, y0, n) qui prend en argument l'image considérée, une coordonnée de départ (x0, y0), une taille n; qui applique récursivement des rotations à chaque quadrant, puis qui applique les permutations circulaires échangeant les quatre quadrants pour finaliser la rotation.

4 - Écrire une procédure effectuant la rotation du quart de tour de l'image. Enregistrer l'image et l'afficher.

5 - Pouvez-vous évaluer la complexité temporelle de cet algorithme ?

6 - Et sa complexité spatiale ?