Devoir de AMD5 n° 1 : Segmentation d'image

Ce TP est à faire à la maison (ou en salle de TP...) de façon individuelle, et à rendre sur DIDEL (cours AMD5) pour le dimanche 5 octobre à 23h59.

I) Définition du problème

On considère des images en noir et blanc, rectangulaires. Étant donné un nombre *entier* R, deux points de coordonnées (x, y) et (x', y') sont **R-voisins** si

$$(x - x')^2 + (y - y')^2 \leq R$$

Ainsi un pixel est 0-voisin avec lui-même seulement, 1-voisin avec quatre pixels (situés immadiatement en haut, en bas, à gauche et à droite) ; 2-voisin avec au plus huit pixels (on rajoute ceux qui le touchent en diagonale) ; etc.

Étant donné un nombre entier R, une R**-composante** (que l'on appellera simplement *composante* quand R est sous-entendu) est un ensemble de pixels **noirs** répondant aux deux critères suivants :

- **connexité** : pour tout choix de deux pixels P et P' de la composante, il existe une suite $P_0 = P, P_1, P_2 ... P_{k-1}, P_k = P'$ de pixels telle que pour tout i entre 0 et k, P_i et P_{i+1} sont R-voisins
- **maximalité** : aucun autre pixel noir de l'image n'est R-voisin d'un pixel de la composante.

Étant donné un nombre entier N, une composante est N**-négligeable** (que l'on abregera en *négligeable* quand N est sous-entendu) si le nombre de pixels qu'elle contient est inférieur ou égal à N.

Le problème à résoudre est, étant donnés une image en noir et blanc et deux entier R et N, **compter** les R-composantes de l'image non négligeables.

II) Algorithmique

Vous **devez** utiliser un algorithme de calcul avec **Union-Find**. L'idée est de partir d'une collection où chaque pixel noir est isolé dans un ensemble, puis de fusionner les ensembles de deux R-voisins, tant que de telles fusions sont possibles. Le plus simple est d'essayer toute paire de pixels (voir pseudo-code page suivante). Ensuite, il faut parcourir la liste des composantes (éléments racines dans l'Union-Find) afin de compter les non-triviales.

On rappelle que la structure dynamique pour maintenir une liste d'ensemble disjoints d'Union-Find est basée sur des arbres. Un nœud de l'arbre est un élément des ensembles considérés. Ici les ensembles sont les composantes, et les éléments les pixels. Chaque nœud a deux champs :

- pere : pointeur sur le nœud père. Une racine R se pointe elle-même : R.pere = R

– taille : si le nœud est une racine, alors le nombre de nœuds de son arbre (sinon indéfini)

Algorithme général :
pour *tout pixel P* **faire**
  pour *tout pixel P'* **faire**
    si *P et P' sont R-voisins* **alors**
      Union(P,P')

Racine(Element X) : **début**
  tant que *X.pere !=X* **faire**
    X.pere := Racine(X.pere)
  retourne X.pere

Find(Element X, Element Y) : **début**
  retourne Racine(X) == Racine(Y)

Union(Element X, Element Y) : **début**
  si *non Find(X,Y)* **alors**
    RacX = Racine(X)
    RacY = Racine(Y)
    si *X.taille < Y.taille* **alors**
      X.pere=Y
      Y.taille += X.taille
    sinon
      Y.pere=X
      X.taille += Y.taille

III) Implémentation

Le programme sera **non interactif !** Il attend trois paramètres, dans l'ordre :
– Le nom du fichier image à traiter, en **format PNG**
– R (entier)
– N (entier)

En sortie, il se contente d'afficher un nombre entier, le nombre de R-composantes non N-négligeables. **Il n'affiche rien d'autre !** Toutefois en cas d'erreur (ouverture d'un fichier image inexistant, mauvais codage de l'image...) il doit afficher un message d'erreur.

Les langages autorisés sont : Python 3 et Java. Il peut avoir plusieurs fichiers sources. Dans ce cas, ils doivent être réunis dans une archive `tar` nommée `archive.tar`, qui ne contient **pas de répertoire**. Nous la désarchiverons par la ligne de commande :

```
tar -xf archive.tar
```

Le nom du fichier principal doit être respectivement `Main.py` et `Main.java`.

Votre programme doit prendre en argument le fichier image. Pour Python, nous exécuterons la ligne de commande suivante :

```
python3 Main.py fichier.png R N
```

Pour java :

```
javac *.java
java Main fichier.png R N
```

Il ne doit y avoir **aucune erreur** lors de la compilation sur les machines de l'ufr (pensez à vérifier que c'est bien le cas).

Lors de l'exécution (sur les machines de l'ufr), votre programme doit afficher **uniquement un entier**, qui correspond à la réponse demandée. Tout autre format (comme par exemple "La réponse est...") donnera la note 0. Voici un exemple d'exécution (où `$` symbolise l'invite du shell) :

```
$ java Main fichier.png 2 15
13
$
```

si votre programme retourne 13.

Vous pouvez bien sûr utiliser des bibliothèques **standard** (c'est-à-dire présentes dans l'installation par défaut) pour charger les images PNG, par exemple PIL en python ou ImageIO en Java.

Vous trouverez sur DidEL un jeu d'images noir et blanc pour lesquelles les bonnes réponses sont :

fichier	R	N	nb de composantes
charpetit.png	1	18	25
cells-inv.png	1	1	116
RBC_bw.png	2	15	63