4TPU148U Informatique 2 - 1ère année

TP noté 12 janvier 2023 groupe B

Durée 1h30 – Tous documents autorisés – Votre voisin(e) n'est pas un document

Téléchargez l'archive TP20230112.zip, décompressez là et travaillez dans le dossier. Sélectionnez le fichier p.blasi.py et renommez le sous le format **j-dupont.py** (où vous remplacerez **j-dupont** par votre initiale et nom de famille). Faites attention de sauvegarder régulièrement ce fichier dans un dossier où vous avez l'habitude de travailler pour pouvoir l'envoyer par mail à la fin de l'épreuve.

A la fin de l'épreuve, envoyez votre fichier par mail à l'adresse: philippe.blasi@u-bordeaux.fr Tout fichier non envoyé sera noté 0/200. Tout fichier envoyé après la fin de l'épreuve sera noté 0/200. Ne quittez pas la salle avant d'avoir vérifié que votre mail a bien été reçu par votre enseignant.

Si une fonction que vous avez écrite ne fonctionne pas, mettez là en commentaire mais ne l'effacez pas, je pourrai tout de même vous accorder des points si l'idée algorithmique est la bonne. Inutile d'expliquer les fonctions qui fonctionnent.

Ce TP noté utilise la bibliothèque PIL et la biblithèque bibgraphes que nous avons étudié en Cours/TD/TM.

Après chacune de vos fonctions, vous avez en commentaire des instructions pour tester vos fonctions. Vous pouvez en rajouter.

Exercice 1 : Stéganographie dans les images

La stéganographie est un domaine où l'on cherche à dissimuler discrètement de l'information dans un media de couverture (typiquement un signal de type texte, son, image, vidéo, etc.). Elle se distingue de la cryptographie qui cherche à rendre un contenu inintelligible à autre que qui-de-droit. Lorsqu'un acteur extérieur regarde un contenu cryptographié il peut deviner la nature sensible de l'information qui lui est cachée. L'intérêt de la stéganographie réside précisément dans la possibilité de communiquer en échangeant des contenus d'apparence anodines de telle sorte à ne pas éveiller de soupçons (https://fr.wikipedia.org/wiki/Stéganographie).

Pour prendre une métaphore, la stéganographie consisterait à enterrer son argent dans son jardin là où la cryptographie consisterait à l'enfermer dans un coffre-fort — cela dit, rien n'empêche de combiner les deux techniques, de même que l'on peut enterrer un coffre dans son jardin.

Exercice 1.1

On a caché de manière grossière, cela se voit, une image mystère dans l'image 'hidden_tiger.png', en remplaçant les pixels de coordonnées multiples de 5 par ceux de l'image mystère. Par exemple, un pixel de coordonnée (8,5) sera donc caché donc caché dans le pixel de coordonnée (40,25) de hidden_tiger. Écrire la fonction **`recupereImage`** qui récupère une image cachée dans **`img`**, dans les pixels dont les coordonnées sont des multiples de **`ligne`** et **`colonne`** et renvoie l'image obtenue.

Exercice 1.2

Maintenant que vous savez récupérer une image cachée, vous pouvez faire l'opération inverse, à savoir cacher une image dans une autre toutes les c colonnes et l lignes. Écrire la fonction **`cacheImage`** qui cache dans l'image **`img`** l'image **`img_a_cacher`**, dans les pixels dont les

coordonnées sont des multiples de **ligne** et **colonne**. Si **img** n'est pas assez grande, on ne cachera qu'une partie l'image.

Exercice 1.3

La façon précédente de cacher les images n'est pas très satisfaisante. On voit à l'œil nu que quelque chose ne va pas dans l'image. Nous allons donc procéder de manière plus fine en cachant une image monochrome. C'est pratique pour cacher du texte par exemple. Pour ce faire, nous allons utiliser la parité de la composante bleue d'un pixel : pair signifiera un pixel noir, impair signifiera un pixel blanc. Ces pixels cachés sont répartis tous les c colonnes et l lignes comme précédemment.

Écrire la fonction **recupereBienImage** récupère une image cachée noir et blanc dans **img**, dans les pixels dont les coordonnées sont des multiples de **ligne** et **colonne**. La couleur du pixel de l'image cachée dépend de la parité de la composante bleue du pixel de l'image source (0 pour noir (0,0,0), 1 pour blanc (255,255,255)). Elle renvoie l'image obtenue. Appliquer cette fonction à l'image 'hidden_tiger2.png' pour trouver le message caché.

Exercice 1.4

Maintenant que vous savez récupérer une image monochrome cachée dans la composante bleue, vous pouvez faire l'opération inverse, à savoir cacher une image monochrome dans une autre toutes les c colonnes et l lignes. Écrire la fonction **cacheBienImage** cache dans l'image **img** l'image **img_a_cacher**, dans les pixels dont les coordonnées sont des multiples de **ligne** et **colonne**. Si **img** n'est pas assez grande, on ne cachera qu'une partie l'image. La couleur du pixel de l'image à cacher sera stockée dans la composante bleu du pixel de l'image cible en modifiant sa parité, pair pour noir et impair pour blanc.

Exercice 2 : Graphes

Exercice 2.1

Écrire une fonction **toutDemarquer** qui demarque tous les sommet du graphe **G**

Exercice 2.2

Écrire une fonction **trouveNonMarque** qui renvoie un sommet non marqué du graphe **G** s'il en existe un, et None sinon.

Exercice 2.3

Écrire une fonction **nombreComposanteConnexe** qui renvoie le nombre de composantes connexes d'un graphe **G**.