Université de Bordeaux

Informatique 1

NOM, Prénom:

2024/2025

TP noté

Durée : 1 heure 15 minutes.

Vous rendrez ce devoir par mail sous forme d'une unique fichier .py à envoyer à l'adresse *nacim.oijid@u-bordeaux.fr*.

Les notes papiers ainsi que les fichiers .py des précédents TPs sont autorisés. Les ressources trouvées sur internet sont interdites. Vous êtes cependant autorisés à utiliser un navigateur pour afficher des graphes si vous le souhaitez.

La propreté et la clarté du code sont évaluées (3 points). Vous devez donc utiliser des noms de variables explicites, ne pas effectuer de calculs inutiles. Il est recommandé de commenter le code.

Exercice 1 (5 points)

On considère L une liste d'entier. Le but de cet exercice est de trier L du plus grand au plus petit élément. Par exemple, la liste $L = [3, 7, 14, 2, 8]$ doit devenir $[14, 8, 7, 3, 2]$

Question 1.1. Écrire une fonction indiceMinimum(L : list, a : int, b : int) -> int qui prend en entrée une liste L et renvoie l'indice du plus petit élément de L qui se trouve entre les indices a et b (inclus). Par exemple, indiceMinimum(L,0,2) doit renvoyer 0, car le plus petit élément entre 3, 7 et 14 est 3, et il se trouve en position 0.

Question 1.2. Écrire une fonction triMinimum(L : list) qui trie la liste L en paramètre en sélectionnant plusieurs fois le plus petit élément de L qui n'est pas encore bien placé, puis en échangeant sa place avec celui qui est à sa bonne position. Le plus petit élément va donc à la fin, le 2e plus petit en avant-dernière position, etc. Sur la liste L, les étapes successives donnent donc $[3, 7, 14, 2, 8]$, $[3, 7, 14, 8, 2]$, $[8, 7, 14, 3, 2]$, $[8, 14, 7, 3, 2]$ et $[14, 8, 7, 3, 2]$.

Exercice 2 (5 points)

Vous disposiez de 3 copies d'une image, mais vos neveux, Riri, Fifi et Loulou sont tombés dessus et ont dessiné dessus avec leurs fluo rouges ($(255, 0, 0)$). Vous aimeriez retrouver votre image de départ. Pour ce faire, vous pensez à l'algorithme suivant :

Question 2.1. Écrire une fonction restaureImage(img1 : image, img2 : image, img3 : image) qui va restaurer votre image en remplaçant chaque Pixel de img1 par un pixel pas rouge d'une des 3 images (et laissé le pixel rouge si les trois images ont un pixel rouge à l'emplacement considéré)

Il se peut que cette méthode ne suffise pas, car il peut y avoir certains pixels qui restent rouges, c'est le cas si les trois neveux ont dessiné dessus. Pour compléter cette méthode, on se propose de remplacer chaque pixel rouge restant par la moyenne de ses quatre pixels adjacents.

Question 2.2. Écrire une fonction MoyenneRouge(img : image) qui effectue cette opération. On pourra supposer qu'il n'y a pas de pixels rouges sur les bords de l'image.

Exercice 3 (7 points)

Soit G un graphe simple. G est dit biparti s'il est possible de colorier les sommets de G en deux couleurs, rouge ("red") et bleu ("blue") de sorte que deux sommets adjacents soient toujours de couleur différente. Le but de cet exercice est de déterminer si un graphe est biparti on non.

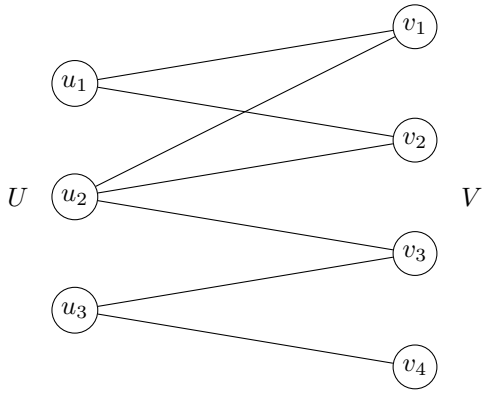

Figure 1: Exemple de graphe biparti : il est possible de colorier en rouge tous les sommets de U et en bleu tous les sommets de V.

Question 3.1. Écrire une fonction Initialise(G : graphe) qui prend en entrée un graphe G et qui colorie en blanc tous ses sommets.

Question 3.2. Écrire une fonction sommetRougeBleu(s : sommet) qui prend en entrée un sommet s d'un graphe G et qui le colorie en rouge s'il n'a pas de voisin rouge, en bleu si tous ses voisins sont rouges ou blancs, et en vert ("green") s'il a au moins un voisin rouge et un voisin bleu.

Question 3.3. Écrire une fonction sommetAdjacentNonBlanc(G : graphe) -> sommet qui prend en entrée un graphe G et renvoie un sommet blanc de G dont au moins un voisin n'est pas blanc. Si tous les sommets de G sont blancs, on renverra n'importe quel sommet. Si G n'a plus aucun sommet blanc, on renverra -1 pour l'indiquer[1].

Notez qu'il est important de commencer par colorier les sommets dont un voisin a déjà été colorié. Dans l'exemple ci-dessus, si vous commencez par colorier v_1 et u_3 en rouge, il ne sera pas possible de colorier u_2 et v_3 sans que deux sommets adjacents aient la même couleur.

question 3.4. Écrire une fonction estBiparti(G : graphe) -> bool qui prend en entrée un graphe G et qui renvoie True si G et biparti et False sinon. Pour ce faire, on coloriera un sommet de G, puis, tant qu'il restera des sommets non coloriés dans G, on coloriera un sommet adjacent à un sommet déjà colorié si possible en utilisant les fonctions précédentes. Le graphe est biparti si on parvient à colorier tous les sommets en rouge ou en bleu, et il ne l'est pas si on est obligé de colorier un sommet un vert.

[1] -1 est souvent utilisé en programmation comme code de sortie des programmes qui cherchent quelque chose pour indiquer que l'élément recherché n'a pas été trouvé