Rapport TP algorithmique et programmation

Coloriage d’images

# Réponses

## Question 1

Pour vérifier le format du fichier, nous pouvons simplement vérifier que la première ligne correspond bien au « nombre magique » P1 ;

## Question 2

Pour vérifier la structure de données, lors de l’ajout d’une couleur dans la matrice « tab2D\_color » on teste si la case qu’on veut modifier est bien noire.

En effet, dans le constructeur de « WriteFic » on initialise toutes les cases de « tab2D\_color » à noir (« 0 0 0 »). Ainsi, si la case n’est pas noire lors de l’ajout de couleur dans celle-ci cela signifie que la case appartient à deux ensembles.

Pour tester cette fonction nous avons implémenté une fonction testEcriture() dans testWriteFic.cpp. Cette fonction crée un tableau d’ensemble de quatre cases avec un ensemble par case qui contient chacun un pixel, un tableau de couleur contenant quatre couleurs différentes, et un objet WriteFic avec le nombre de ligne du futur fichier à 2 et le nombre de colonne à 2. On lance la fonction ajEnsembleTable pour écrire la couleur de chaque pixel dans un tableau. Puis, avec writeThePPMFic on transcrit ce tableau dans un fichier ppm qui sera nommé result.ppm. Pour vérifier les résultats on a juste à regarder le fichier result.ppm qui doit être constitué de la manière suivante :

* Première ligne : pixel rouge, pixel vert
* Deuxième ligne : pixel bleu, pixel jaune

## Question 4

## Question 5

MakeSet(x) correspond au constructeur de la classe « Ensemble ». La complexité asymptotique relative au nombre d’affectation est de 3 ce qui correspond à Θ(1). Le coût en mémoire est de 3 car il y a trois affectations.

FindSet(x) correspond à la fonction getHead() de la classe Pixel. Sa complexité asymptotique relative au nombre d’affectation est de Θ(1) car il n’y a qu’un « return ». Son coût en mémoire est de zéro.

## Question 6

Soit X et Y deux ensemble avec X le plus grand des deux ensembles. On commence par faire pointer tous les pixels de y vers le représentant de X. C’est-à-dire que chaque pixel de Y aura pour attribut « \_head » un pointeur vers le « \_head » de X, qui est le représentant de l’ensemble X.

Ensuite on stocke la taille du « nouvel » ensemble. On rattache la tête de y à la queue de x. La queue de x devient la queue de y.

Enfin on détruit le contenant y qui ne sert plus, le contenant x étant devenu l’ensemble xy.

Dans le pire des cas, n étant la taille de y la complexité asymptotique dans le pire des cas est de O(n). Le coût en mémoire correspondrait à la taille des deux ensembles.

## Question 7

On définit |S| comme étant le nombre d’élément de l’ensemble S.

La création d’un ensemble de pixel S1 a une complexité asymptotique (en pire cas) de |S1|. Et l’union de S1 et S2 (avec |S1| >= |S2|) a une complexité asymptotique en pire cas de |S2|.

En supposant que les ensembles créés ici contiennent un nombre **m** de pixels, la complexité asymptotique en pire cas de la création de **n** ensembles contenant **m** pixels est

(**n** -1) **m**

## Question 8

## Question 12

Commençons par le cout mémoire :

Lors de la lecture du fichier, l’objet *LecteurImage* alloue n\*m\*16 octets. n et m étant la hauteur et la largeur de l’image. 16 octets représente la taille d’un objet *Pixel*.

Exemple : Pour une image de 2567 de largeur et 1091 de hauteur. Le cout mémoire sera de : 44,8Mo

Complexité asymptotique en pire cas :

* De l’algorithme d’union des ensembles

On applique au n pixels la fonction d’union de deux ensembles addEnsemble\_inTail