RAPPORT ALGORITHMIQUE & PROGRAMMATION

TP1 – PREMIERE PERIODE BURLAT NILS – PORTE PIERRE-ANTOINE

QUESTIONS

Pour ces questions, l'emploi de « n » signifie la taille du vecteur de pixels utilisé dans le programme.

Question 1 : Lecture d'un fichier conforme au format, précision cas d'erreur, traitement et tests réalisés

Nous identifions les erreurs suivantes :

- Première ligne différente de « P1 »
- Seconde ligne différente de « [INTEGER] [INTEGER] »
- Suites des caractères différent de « 0 », « 1 », espace « » ou retour chariot

Nous avons lancé le programme avec en paramètre les images « non_conforme*.pbm » renvoyant toujours l'exception définie pour ce cas d'usage.

Question 2 : Ecriture d'un fichier, cas d'erreur, traitement et tests réalisés

La seule erreur possible est si un pixel appartenant à l'image n'a pas de représentant (de set), il ne peut donc pas écrire sa couleur. Si c'est le cas, nous renvoyons une erreur que nous avons définit dans le fichier regroupant les différentes exceptions.

Question 3 : Implémentation de la fonction Generate()

La fonction prend en paramètre trois entier : la largeur et la hauteur de l'image à générer et également le pourcentage de pixels noirs que l'image devra contenir (ce paramètre est donc compris entre 0 et 100 inclus).

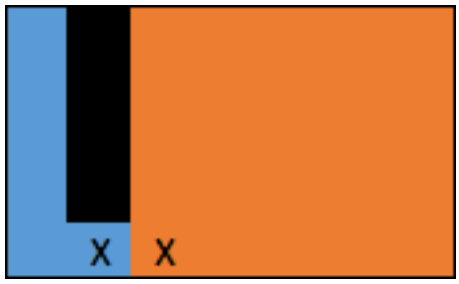
Question 4 et 5 : Implémentation et complexité asymptotique en pire cas et coût mémoire de findSet() et makeSet()

Les fonctions findSet() et makeSet() sont de complexité constante en pire et meilleur cas (donc le coût moyen également) car on retourne un pointeur vers le représentant de son set pour findSet() ou on crée un nouveau set pour makeSet().

On a donc $\theta(1)$ pour le pire des cas.

Question 6 : Union de deux ensemble, complexité pire cas et coût mémoire

L'union est « presque » en place. En effet on ne duplique pas le tableau, mais on va quand même stocker un pointeur vers le représentant du premier set que l'on va parcourir (on occupera donc 32 bits ou 64 bits selon le mode de compilation de l'application).



PIRE CAS 1

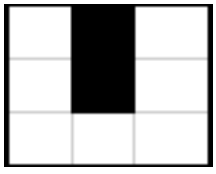
Le pire des cas est une image nous obligeant à joindre deux sets de même taille. Ce cas se produit lorsqu'il faut fusionner un set de taille presque n à un set de petite taille (nécessairement si le premier vaut presque n). Nous devons parcourir le second set afin de lui changer son représentant. On a donc une complexité en pire en cas de $\theta(n)$ pour l'union.

Question 7 : Quelle est la complexité asymptotique d'une séquence de n créations d'ensembles suivie des unions successives de ces ensembles en un seul ensemble final ?

La création d'ensemble est de complexité pour le pire, meilleur et cas moyen de n. On doit parcourir tout le fichier et créer un pixel pour chaque caractère lu dans le fichier.

On doit ensuite parcourir tous ces pixels afin de les assembler. L'assemblage des sets (union) est de complexité asymptotique en pire cas de n (voir question 6). On est donc sur du n^2 en pire cas.

Question 8 : Amélioration de la structure de données permettant de réaliser plus rapidement une séquence de création et union d'ensemble



PIRE CAS 2

On a stocké la taille d'une liste chaînée, ainsi on va préférer parcourir la liste la plus petite pour changer les représentants. On a donc une complexité en

pire cas de (n-1)/2, car on fera une jointure sur deux sets faisant respectivement la moitié de l'image.

Question 10 : Question 7 avec amélioration de la structure de données

On n'a plus qu'à parcourir le set auquel on doit mettre le nouveau représentant pour l'assemblage, et nous sommes certain que ce set sera le plus petit grâce au stockage de la taille. Ainsi on a une complexité en pire cas de (n-1)/2, car au pire des cas, un set composé de la moitié de l'image pour lui changer son représentant.

Si jamais nous faisons plusieurs unions de taille > 1, la somme de leurs coûts ne dépassera jamais (n-1)/2 car on change toujours le set le plus petit passé en paramètre, ainsi nos sets seront de tailles « moyenne » si nous avons plusieurs « grandes » unions à faire.

Donc le pire cas de l'union n'est réalisable qu'une fois. En effet, comme on peut le voir sur l'image « Pire cas », on va effectuer n-1 union de coût de un, et une union finale de coût (n-1)/2. On est donc sur une complexité linéaire 3n/2 en pire cas dans la pratique.

Question 11 : Tests effectués sur l'algorithme de coloriage automatique

Nous avons testé les images disponibles sur Teide. Nous avons effectué un débogage notre application grâce à des images plus petites pour comprendre pourquoi notre algorithme ne marchait pas, comme par exemple l'image conforme.pbm.

Nous avons également réalisé une classe de test lançable avec la commande

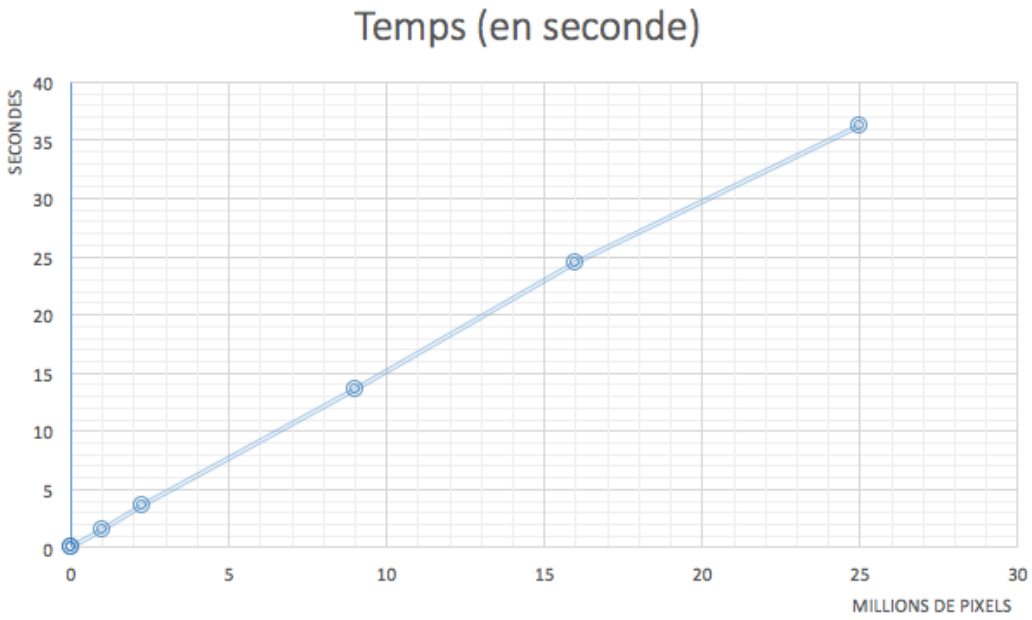
```
./algoprog test
```

Elle teste l'union bonus et classique, et doit renvoyer un ensemble joint de pixel affichant la même couleur.

Question 12 : Quelle est la complexité asymptotique de pire cas et le coût mémoire de l'algorithme

Coût mémoire : Nous devons stocker les n pixels que nous modifierons durant l'algorithme. Ces pixels ont en mémoire 4 *int* (composantes RGB + taille de l'ensemble), ainsi que 2 pointeurs (représentant et suivant du set, la « queue » n'est pas comptée car non utilisée dans la meilleure union). On a donc 7n allocation (6 attributs et le pixel lui même) de mémoire dépendant du système d'exploitation et/ou de la compilation (32 bits/64 bits). On néglige toutes les autres variables dont le place mémoire est constante quelle que soit la taille de l'image à traiter.

Question 13 : Complexité en pire cas théorique et cout réel en temps de l'algorithme



Taille (en pixel)	Temps (en seconde)	Rapport (taille/temps)
100	0	
10000	2,6	3846,153846
1000000	150,2	6657,789614
2250000	357	6302,521008
9000000	1368,8	6575,102279
16000000	2453,7	6520,76456
25000000	3627,8	6891,228844

Nous sommes mathématiquement en $o(n^2)$, cependant dans le pire cas nous avons estimé un $\theta(n)$ car le coût asymptotique est de $3n/2$. Le temps réel montre que le coût est linéaire, ainsi que les rapports entre la taille et le temps.

Les coûts de lecture et d'écriture d'image sont linéaire en $\theta(n)$ car nous écrivons/lisons n pixels, et il n'y a pas de traitement supplémentaire.

Les graphiques représentés prennent en compte la lecture, l'algorithme de coloriage et l'écriture.

Question 14 : Changement à Union supprimant le pointer « queue » gardant les mêmes complexités asymptotiques.

On va « brancher » entre la tête du premier ensemble et le second pixel du premier ensemble l'intégralité de l'autre liste chaînée. Ainsi on n'a plus besoin de stocker queue, car on fait un embranchement des deux listes au début de la 1ere.