Herzig Melvyn

SIO – 25.03.2021

Heuristique   
largest first

Laboratoire 1



Table des matières

[Introduction 1](#_Toc67156432)

[Architecture 1](#_Toc67156433)

[Complexité 2](#_Toc67156434)

[Vérification des implémentations 3](#_Toc67156435)

[Newest 4](#_Toc67156436)

[Oldest 5](#_Toc67156437)

[Least 6](#_Toc67156438)

[Most 7](#_Toc67156439)

[Analyse des performances 8](#_Toc67156440)

# Introduction

Le but de ce laboratoire est d’implémenter 4 versions de l’heuristique Largest First de Welsh et Powel.

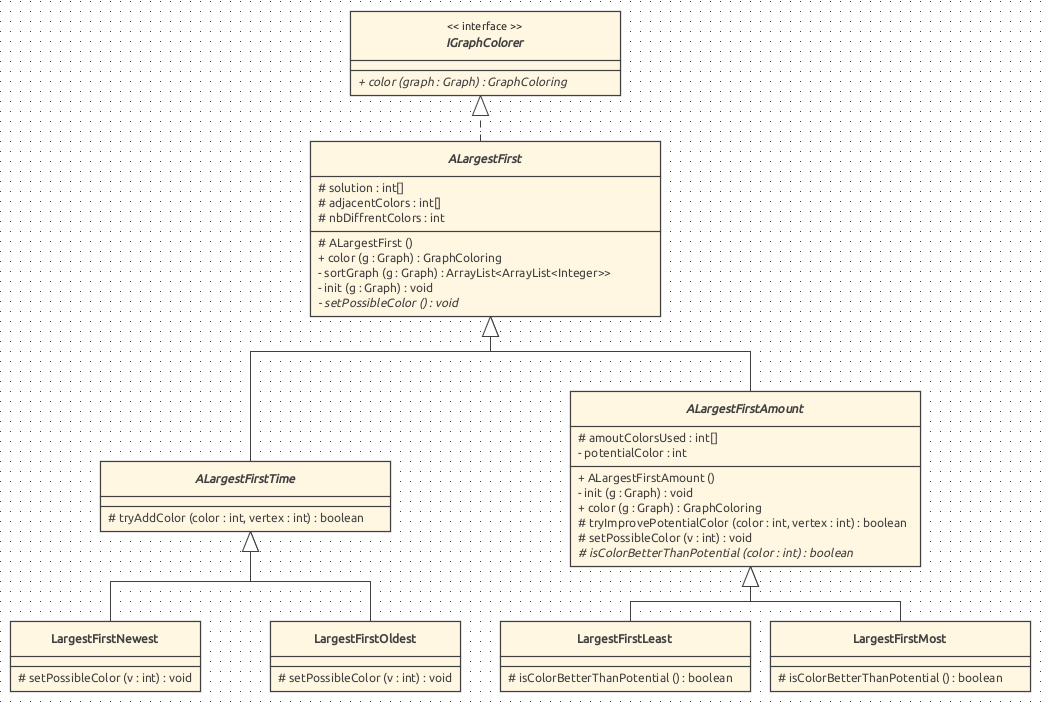
Les 4 versions sont :

* Oldest : La couleur choisie en priorité est la plus vieilles des couleurs.
* Newest : La couleur choisie en priorité est la plus récente des couleurs.
* Least : La couleur choisie en priorité est la moins utilisée des couleurs.
* Most : La couleur choisie en priorité est la plus utilisée des couleurs.

Ce document décrit l’implémentation ainsi que les résultats observés.

# Architecture et algorithme

Le diagramme de classes du programme est le suivant :



La classe *ALargestFirst* est le noyau des 4 algorithmes. Elle trie les sommets et elle vérifie quelles sont les couleurs adjacentes à un sommet donné.

Nous avons ensuite deux variantes :

## Oldest/Newest

Pour chaque sommet, la fonction *colo*r, de ALargestFirst, appelle *setPossibleColor* qui parcourt *adjacentColors* dans un sens spécifique à la version de LF. Pour chaque couleur visitée, on tente de l’ajouter au sommet donné avec *TryAddColor* dans *ALargestFirstTime*.

## Least/Most

La classe *ALargestFirstAmount* offre une structure supplémentaire. *AmountColorUsed* stocke le nombre d’utilisations de chaque couleurs. Le parcours de *adjacentColors* est semblable dans la version Least et Most de LF. De ce fait *setPossibleColor* est implémenté dans *ALargestFirstAmout*.

Dans tous les cas, on parcourt le tableau des couleurs utilisées à la recherche de la plus/moins affectée. On conserve la couleur candidate dans *potentialColor*. Les couleurs sont comparées à *potentialColor* dans *isColorBetterThanPotentialColor*. Une fois que toutes les couleurs sont parcourues, il est possible d’affecter une couleur au sommet courant.

Dans les 4 variantes, une nouvelle couleurs est ajoutée et affectée au sommet courant si aucune couleurs déjà employée ne peut être affectée.

# Complexité

Soit N le nombre de sommets d’un graphe et M son nom d’arêtes. On pose aussi C le nombre maximal de couleurs de ce graphe dans le pire des cas. On a donc C <= N.

Le tri des sommets est effectué à l’aide d’un tri bucket, donc en O(M+N)

Dans le cas des versions Oldest et Newest.

Dans ces deux versions, nous parcourons les couleurs utilisées et nous nous arrêtons sur la première candidate affectable. La recherche d’une couleur est effectuée en O(C).  
On en déduit donc que **les versions Oldest et Newest sont amorties en O(M + N).**

Dans le cas des versions Least et Most.

Dans ces deux versions, l’ensemble des couleurs affectées est parcouru à la recherche de la meilleure candidate. Cette opération est faite en O(C).

Ainsi la recherche et le repositionnement d’une couleur, dans le bon bucket, s’effectue en O(2C).  
On en déduit donc que **les versions Least et Most sont amorties en O(M + N).**

# Vérification des implémentations

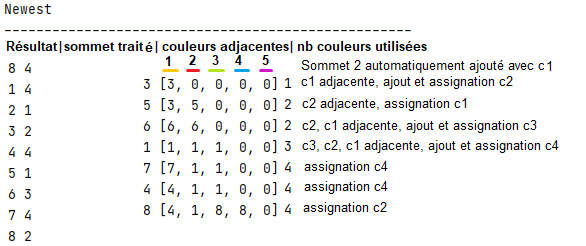
Nous avons reçu trois graphes de tests : *EX\_SERIE1\_4\_a.txt, EX\_SERIE1\_4\_b.txt*, *EX\_SERIE1\_4\_c.txt*. Les 4 versions de LF ont été testées sur les trois graphes. Afin d’alléger ce document, nous ne présenterons que les résultats du dernier test sur le graph *EX\_SERIE1\_4\_c.txt*.

Les sorties des 4 algorithmes ont été comparées avec les solutions mises à disposition. Toutes correspondent.

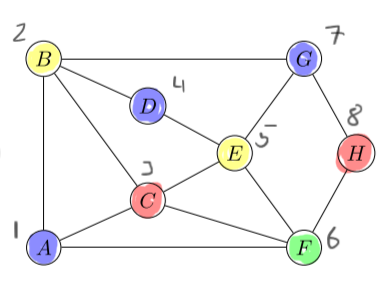
## Newest

Voici une sortie console commentée, les deux colonnes de gauches sont le graphes colorié, la colonne à gauche de la matrice correspond au sommet traité, la matrice représente les couleurs adjacentes au sommet traité et la colonne à droite de la matrice est le nombre de couleurs utilisées.

Comme indiquée la première couleur est allouée au plus petit sommet de plus grand degré avant le l’affichage des logs.



**Représentation**

****

Comme nous pouvons le voir, les sommets sont traités dans le bon ordre et la couleur assignée est toujours la plus récente ou une nouvelle.

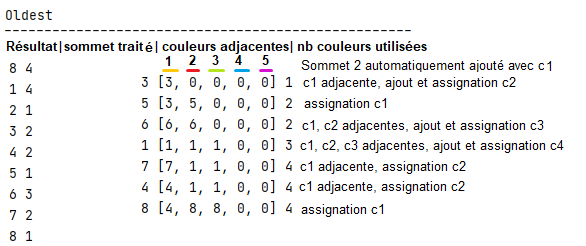
Aucun sommet ne se voit assigner une couleur adjacente.

Au terme des trois testes, en première approche, l’algorithme LF version Newest est fonctionnel.

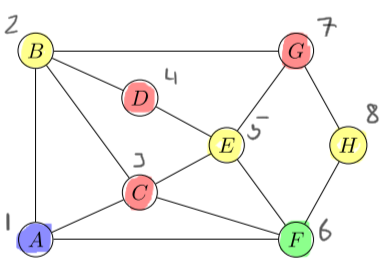
## Oldest

Voici une sortie console commentée, les deux colonnes de gauches sont le graphes colorié, la colonne à gauche de la matrice correspond au sommet traité, la matrice représente les couleurs adjacentes au sommet traité et la colonne à droite de la matrice est le nombre de couleurs utilisées.

Comme indiquée la première couleur est allouée au plus petit sommet de plus grand degré avant le l’affichage des logs.



**Représentation**

****

Comme nous pouvons le voir, les sommets sont traités dans le bon ordre et la couleur assignée est toujours la plus ancienne ou une nouvelle.

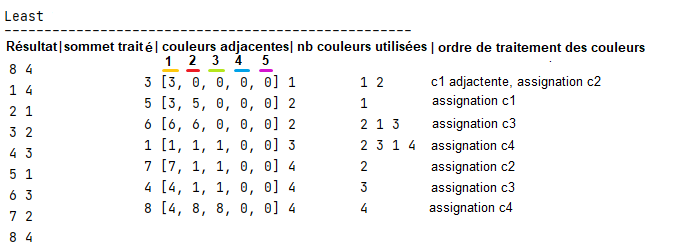
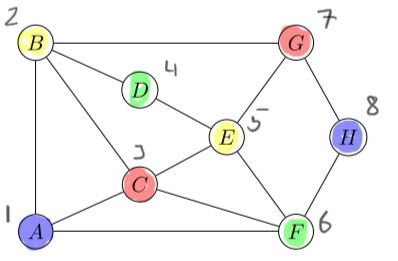
Aucun sommet ne se voit assigner une couleur adjacente.

Au terme des trois testes, en première approche, l’algorithme LF version Oldest est fonctionnel.

## Least

Voici une sortie console commentée, les deux colonnes de gauches sont le graphes colorié, la colonne à gauche de la matrice correspond au sommet traité, la matrice représente les couleurs adjacentes au sommet traité et la colonne à droite de la matrice est le nombre de couleurs utilisées, puis l’ordre de traitement des couleurs.

Comme indiquée la première couleur est allouée au plus petit sommet de plus grand degré avant le l’affichage des logs.

 **Représentation  
**

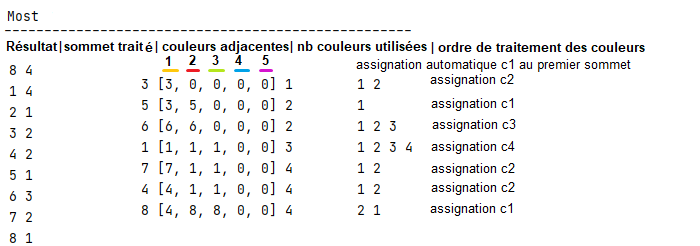
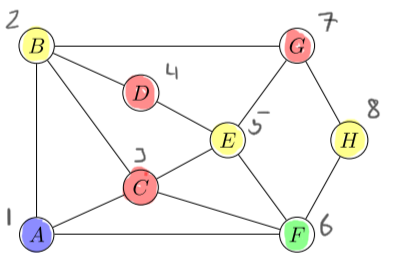
|  |  |  |  |
| --- | --- | --- | --- |
| Sommet traité | Couleurs 0 utilisation | Couleurs 1 utilisation | Couleurs 2 utilisation |
| 2 | 1, 2, 3, 4, 5 | / | / |
| 3 | 2,3,4,5 | 1 | / |
| 5 | 3,4,5 | 1,2 | / |
| 6 | 3,4,5 | 2 | 1 |
| 1 | 4,5 | 2,3 | 1 |
| 7 | 5 | 2,3,4 | 1 |
| 4 | 5 | 3,4 | 1,2 |
| 8 | 5 | 4 | 1,2,3 |

Comme nous pouvons le voir, les sommets sont traités dans le bon ordre et la couleur assignée est toujours la première la moins utilisée non adjacente. Les couleurs existantes sont privilégiées aux couleurs avec 0 utilisation. En cas d’égalité, la plus vieille couleur vient avant.  
  
Au terme des trois testes, en première approche, l’algorithme LF version Least est fonctionnel.

## Most

Voici une sortie console commentée, les deux colonnes de gauches sont le graphes colorié, la colonne à gauche de la matrice correspond au sommet traité, la matrice représente les couleurs adjacentes au sommet traité et la colonne à droite de la matrice est le nombre de couleurs utilisées, puis l’ordre de traitement des couleurs.

Comme indiquée la première couleur est allouée au plus petit sommet de plus grand degré avant le l’affichage des logs.

**  
Représentation  
**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sommet traité | Couleurs 0 utilisation | Couleurs 1 utilisation | Couleurs 2 utilisations | Couleurs 3 utilisations |
| 2 | 1, 2, 3, 4, 5 | / | / |  |
| 3 | 2,3,4,5 | 1 | / |  |
| 5 | 3,4,5 | 1,2 | / |  |
| 6 | 3,4,5 | 2 | 1 |  |
| 1 | 4,5 | 2,3 | 1 |  |
| 7 | 5 | 2,3,4 | 1 |  |
| 4 | 5 | 3,4 | 1,2 |  |
| 8 | 5 | 4 | 1,3 | 2 |

Comme nous pouvons le voir, les sommets sont traités dans le bon ordre et la couleur assignée est toujours la première la plus utilisée non adjacente. Les couleurs existantes sont privilégiées aux couleurs avec 0 utilisation. En cas d’égalité, la plus vieille couleur vient avant.  
  
Au terme des trois testes, en première approche, l’algorithme LF version Least est fonctionnel.

# Analyse des résultats

L’analyse des résultats est axée sur le nombre de couleurs et le temps d’exécution. Les graphes ont été séparés en 4 familles : peu dense (0.1), moyennement dense (0.5), dense (0.9) et VLSI.

Le temps d’exécution est une moyenne de 5 mesures. Nous nous basons principalement sur les grand graphes (4000 et 8000 sommets), où l’ordonnancer semble avoir moins d’impact et les tendances mieux dessinées.

## Densité faible

Dans un graphe peu dense, l’algorithme LF Most est le plus efficace et le pire est LF Least. La différence, entre le pire et le moins bon varie jusqu’à 10%. LF Oldest et LF Newest sont dans la moyenne mais LF Oldest est sensiblement plus performant que LF Newest. De plus, il est souvent proche de LF Most.

Avec 500 sommet, les trois algorithmes sont autant rapides. Pour le graphe à 2000 sommet, étonnamment, LF Most est le plus rapide. Au-delà de 2000 sommets, LF Most est les plus rapide.

Ainsi, dans un graphe peu dense et en termes d’optimisation des couleurs, il faut préférer LF Most. Concernant un rapport couleur/temps, LF Oldest

## Densité moyenne

Dans le cas de graphes moyennement denses, LF Newest est le plus optimal, suivi de près par LF Most et, loin derrière, LF Oldest et LF Least.

Difficile de déterminer un algorithme plus rapide. Avec 2000 sommets : LF Least, avec 4000 : LF Most et avec 8000 : LF Oldest. A ce stade, on en déduit que nous ne pouvons définir une algorithme plus rapide sans investigation plus poussée.

Dans le cas d’un graphe moyen, nous préférerons donc utilise LF Lest pour son faible nombre de couleurs.

## Densité élevée

Dans le cas de graphes denses, LF Most, LF Least et LF Oldest sortent du lot. Ils ont toujours le même nombre de couleurs à l’exception d’une fois. Avec 2000 sommets, LF Most a été meilleure d’une couleur par rapport à ses deux rivaux. Contrairement aux graphes moyennement denses, LF Newest passe de premier à dernier.

De façon semblable aux couleurs, LF Most, LF Least et LF Oldest sont au coude à coude et LF Newest est en retard. Cependant LF Oldest semble tirer son épingle du jeu et être sensiblement plus rapide que LF Least et LF Most.

Dans le cas de graphes denses LF Most, LF Least et LF Oldest sont trois bonnes alternatives optimales en termes de couleurs. Toutefois LF Oldest est le plus rapide ce qui le rend plus aventageux.

## VLSI

Je ne savais pas trop comment traiter les graphes VLSI. Ils ont une densité qui s’étend de 0.9 à 0.16. De ce fait j’ai décidé de ne pas les inclure avec les graphes peu denses par manque d’homogénéité. Je les considère.

Difficile de déterminer un meilleure algorithme : dans VLSI 1 LF Least est meilleur, dans VLSI2 LS Most et dans VLSI3 LF Newest. On ne peut pas déduire de meilleure algorithme mais un moins bon : LF Oldest.

Contrairement aux couleurs, LF Oldest est le plus rapide.

## Analyse globale

Il est difficile de déterminer un meilleure algorithme.

Le temps d’exécution est très proche, entre les quatre versions et ce, quel que soit le type de graphe. Comme aucune valeur aberrante n’est apparue, il n’est donc pas un critère pertinent dans le choix d’une version.

Du côté des couleurs, il est également difficile de choisir une meilleure version. Même pour une densité identique, il n’est pas aisé de déterminer une version qui écrase les autres. Parfois c’est l’une qui est meilleur que l’autre et inversement.

Les graphes peuvent prendre moultes formes et chacune de ses formes influence sur les algorithmes. A moins de connaître la nature exacte de son graphe, il n’est pas possible de déterminer un meilleur algorithme. Du moins, ce n’est pas possible avec l’étude faite au cours de ce laboratoire.

A mon avis, si chacune de ces variantes existes, c’est parce que, dans un sens, elles sont toutes valables.

# Fonctionnement du programme

Pour utiliser et tester mes implémentation, il est possible de passer des arguments à mon programme.

Le premier argument peut prendre 5 formes :

* *oldest*/*newest*/*least*/*most* : exécute l’algorithme concerné et affiche la coloration dans le flux de sortie.
* Benchmark : Lance un benchmark qui teste 5 fois chaque algorithme pour une série de graphes.

Le second argument peut prendre dans forme qui dépendent du premier :

* Dans le cas de *oldest*/*newest*/*least*/*most*, passer un chemin vers un fichier contenant un graphe.
* Dans le cas de benchmark, passer un chemin contenant une série de fichier graphes. Il ne doit rien contenir d’autre que des graphes car le benchmark tente de lire tous les fichiers.