Projet de programmation

Coloration de graphe

Paris, Léopold, Daniel, Alexandre, Jade

Notre projet porte sur le thème du Théorème des 4 couleurs. Le but de notre projet est de manipuler des graphes planaires et de pouvoir les colorier avec 4 ou 5 couleurs et de transformer des cartes en graphes planaires pour pouvoir par la suite les colorier. Le tout à travers une interface graphique.

Introduction

1. Le projet devra être facilement récupérable sur git et exécutable sur toute machine.
2. Proposer un guide d'utilisation clair.
3. Un rapport qui présente l'essentiel des axes du projet.
4. Un code bien structuré, muni d'un diagramme explicatif.
5. Une trace continue de notre méthodologie.

Les membres de notre projet travaillent soit sur windows ou sur windows avec un sous système ubuntu ou encore directement sur un linux natif. Cela nous permet d'être sûr que notre projet puisse être exécutable sur toutes sortes de machines.

Par la suite, notre projet se trouve déjà sur gitlab ce qui le rend facilement récupérable. De plus, c'est un projet maven ce qui nous permettra à la fin du projet grâce à la commande "mvn package" de produire un fichier .JAR qui permettra de lancer la version finale de notre appli.

Pour le guide d'utilisation clair il vous sera disponible dans le fichier README.md et nous mettrons un accent sur la GUI pour qu'elle soit la plus adaptée pour chaque utilisateur et donc facile d'utilisation.

Nous avons **4 axes principaux** pour notre projet : Création des Cartes, Création des algorithmes de coloriage de graphe, Création des graphes, GUI (Interface Graphique).

Création des cartes

Choix de différentes cartes et possibilité de les colorier manuellement et d'enregistrer les noms des différentes sections d'une carte en cliquant directement sur celle-ci.

Création des algorithmes de coloriage de graphe

Cet axe du projet consiste à colorier des graphes planaires avec différents algorithmes que nous allons vous présenter ci-dessous :

Welsh Powell

1) Repérer le degré de chaque sommet.
2) Ranger les sommets par ordre de degrés décroissants (dans certains cas plusieurs possibilités).
3) Attribuer au premier sommet (A) de la liste une couleur.
4) Suivre la liste en attribuant la même couleur au premier sommet (B) qui ne soit pas adjacent à (A).
5) Suivre (si possible) la liste jusqu'au prochain sommet (D) non encore coloré de la liste.
6) Continuer jusqu'à ce que la liste soit finie.
7) Prendre une deuxième couleur pour le premier sommet (D) non encore coloré de la liste.
8) Répéter les opérations 4 à 7.
9) Continuer jusqu'à avoir coloré tous les sommets.

Dsatur

1) Ordonner les sommets par ordre décroissant de degrés.
2) Colorer un sommet de degré maximum avec couleur 1.

3) Choisir un sommet avec DSAT maximum. En cas d'égalité, choisir un sommet de degré maximal.
4) Colorer ce sommet avec la plus petite couleur possible.
5) Si tous les sommets sont colorés alors stop, sinon aller en 3).

Greedy

Avec Greedy, on va colorier le premier sommet avec la couleur 1 puis on regarde le deuxième : s' il n'y a pas la couleur 1 parmi les couleurs de ses voisins on le colorie avec la couleur 1 sinon on essaie de le colorier avec la couleur 2 et ainsi de suite.

Si on ne peut pas attribuer de couleur (car toutes les couleurs disponibles sont parmis les voisins) alors on revient en arrière : on enléve la derniére couleur que l'on a mise et on mets la couleur suivante si possible et on repars.

Si ça ne marche pas ou si il n'y a pas de couleur suivante, alors on revient en arrière eventuellement plusieurs fois jusqu'à trouver une solution. Si on ne trouve aucune solution alors on renvoie le booléen faux, le graphe n'a pas pu être coloré.

Remarques

- La première solution trouvée sera adoptée même si ce n'est pas forcément la meilleure possible.

- La coloration peut aussi dépendre du nombre de couleurs autorisées:

    - Par exemple, si on utilise Greedy sur le graphe des USA, il peut être colorié avec 4 couleurs quand on en autorise 4 et avec 5 si

on en autorise 5. Ce n'est pas forcément le cas mais c'est possible. L'ordre dans lequel les sommets sont triés et donc dans lequel ils vont être coloré à une importance, parfois avec un ordre différent le graphe peut être coloré avec moins de couleurs ou au contraire la coloration peut échouer s'il n'y a pas suffisamment de couleurs.

- Il est possible de tester toutes les combinaisons possibles et de sélectionner la meilleure mais cela nécessite beaucoup de comparaisons (comme en témoigne la fonction "bestGreedy" qui ne fonctionne que sur des graphes de très petite taille (moins de 12 sommets)) et donc l'algorithme est peu efficace.

Kempe

L'algorithme commence par trier les sommets du graphe en les plaçant dans une pile en commençant par le sommet ayant le moins de voisins jusqu'à celui qui en a le plus.

Il va ensuite coloré avec des appels récursifs chaque sommet en commençant donc par le sommet avec le plus de voisins.

Si un sommet à 5 voisins ou plus et que les 5 couleurs autorisés sont parmis ses voisins alors il est impossible de le colorer.L'algorithme sélectionne alors deux de ses voisins qui ne sont pas voisins entre eux et ayant une couleur différente.

On va alors construire un graphe similaire au graphe de départ mais en gardant uniquement les sommets ayant la même couleur qu'un de ses 2 sommets et en partant du premier voisin sélectionné puis en regardant ses voisins puis leurs voisins et ainsi de suite.

On construit la chaîne de sommets constituée de ces 2 couleurs, passant par le premier voisin, la plus longue possible. Si cette chaine ne passe pas par le second voisin sélectionné alors on inverse chaque couleur de la chaîne deux à deux, on va ainsi libérer une couleur pour le sommet de départ.

Sinon si la chaîne passe par le second voisin alors on sélectionne de la même manière deux autres voisins du sommet initial et on recrée une chaîne passant par le premier des deux. (Cette chaîne ne passe pas par le second voisin, il n'est pas nécessaire de le vérifier.) On inverse les couleurs de cette chaîne, on libère donc une couleur et on colore le sommet initial. De cette manière n'importe quel graphe peut être coloré avec 5 couleurs ou moins.

Création des graphes

Les outils et fonctions du module *utils* permettent la création de graphiques. Les fichiers GraphCreator, Converter et GraphUpdater sont chargés de convertir l'ensemble des images associées aux fichiers csv en graphique. Ainsi, toute image disposant d'un fichier csv décrivant les régions et leurs voisins respectifs peut être utilisée dans notre projet pour être coloriée avec les algorithmes de coloration.

Concernant la création de graphes aléatoires, nous avons décidé d'utiliser une approche basée sur la création de diagrammes de Voronoi, de cette façon chaque graphe créé sera planaire car il est dérivé d'une figure planaire.

Le diagramme de Voronoi est un processus de partition d'un plan en régions, chacune plus proche d'un point sur le plan. En répartissant aléatoirement des points dans l'espace et en respectant l'algorithme de

création de régions basé sur un simple calcul de la distance “euclidienne” ou “Manhattan” entre chaque pixel du plan et une cellule de Voronoi (point du plan) on peut dessiner des régions. Pour chaque ensemble de points aléatoires, nous aurons une configuration différente.

Après avoir créé le diagramme de Voronoi, nous convertissons le diagramme en une structure de graphe. On passe par chaque point du plan et pour chacun des points on cherche ses régions les plus proches, analogues aux “voisins” sur une carte. Maintenant, nous pouvons générer des graphes avec des cartes, mais nous pouvons également générer des graphes aléatoires planaires et appliquer les algorithmes de coloration.

Nous avons également décidé de permettre à l’utilisateur d’analyser la complexité des algorithmes de coloration. Pour chaque algorithme, pour chaque graphe et carte, une analyse de runtime, de la complexité en temps et du nombre de couleurs est à la disposition de l’utilisateur, afin de comparer les différentes performances parmi les algorithmes sur des cartes et des graphes aléatoires de l’application.

GUI (Interface Graphique)

Pour afficher une interface graphique, nous avons utilisé la librairie Java Swing. Tout d’abord, la classe qui permet d’afficher une fenêtre pour l’utilisateur est GUI. Elle hérite de la classe JFrame. Nous avons ensuite plusieurs sous classes héritant de JPanel que nous pouvons inclure dans notre **GUI** : *HomeView, GraphCreatorView, RandomGraphView, MapChooser ou encore GraphPlayView*. Ces classes représentent les différentes pages qu’on veut pouvoir afficher à l'utilisateur dans le logiciel. On précise que ces classes ont un attribut “gui” initialisé dans leur constructeur. Grâce à celui-ci, on peut alors interagir facilement avec la JFrame principale et changer la page actuellement affichée à l’écran grâce aux méthodes qui permettent de switcher entre les pages

(setHomeView, setRandomGraphView, setCreatorPage, etc…) définies dans GUI.java.

HomeView

Cette classe permet d'afficher le menu principal. On peut alors cliquer sur un des boutons affiché à l'écran pour aller dans le mode qui nous intéresse (**Graph Creator** [GraphCreatorView.java qui hérite de GraphPlayView.java] , **Map Chooser** [MapChooser.java], **Random Graph** [RandomGraphView.java qui hérite également de GraphPlayView.java]).

MapChooser

Cette page est assez simple à comprendre. Il s'agit d'un sélectionneur de cartes. On peut passer d'une carte à l'autre avec les boutons ou les flèches du clavier puis choisir une carte (avec un clic sur la carte ou avec la touche ENTREE). Un **GraphPlayView** va alors s'afficher à l'écran et on pourra interagir avec la carte sélectionnée (et son graphe).

GraphPlayView

Cette classe est la classe la plus importante du projet au niveau de l'interface graphique. En effet, elle permet d'afficher un graphe et/ou une carte coloriable à l'écran. Elle a également sur le côté gauche un menu pour interagir facilement avec un graphe et/ou une carte. Pour pouvoir afficher un graphe, elle utilise la classe GraphView. Pour les cartes, il s'agit de MapView. La plupart du temps, on a donc un MapView et un GraphView, tous les deux reliés à un même graphe (Graph.java). Lorsqu'on applique des algorithmes, on modifie donc le Graph (du model) et "il suffit" donc ensuite de rafraîchir le GraphView et le MapView avec les nouvelles valeurs de leur Graph. Ensuite, cette classe dispose d'un menu avec différentes options (qui peuvent varier si la classe est un *RandomGraphView, GraphCreatorView* ou alors vraiment un **GraphPlayView** [et donc pas une

classe qui hérite de GraphPlayView]). Les principaux boutons, toujours présents, sont le bouton “Play” qui permet de changer les couleurs des sommets d’un graph et le bouton “Algorithms”, avec lequel on peut appliquer nos algorithmes sur nos graphes/cartes. Il y a également un bouton “Run simulation” qui va nous permettre de comparer la complexité des algorithmes appliqués sur un graphique spécifique. Le bouton “switch to Map/Graph” est aussi parfois présent et nous permet d’afficher soit la carte, soit son graphe selon nos envies. Voici une description des différentes pages basées sur un **GraphPlayView** :

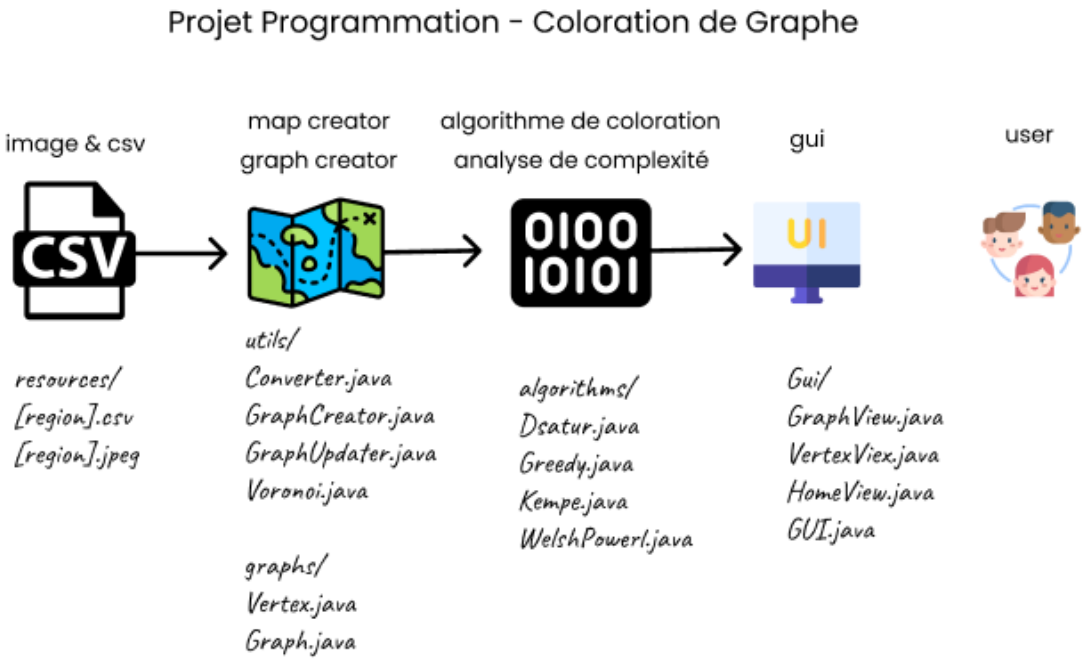
Graph Creator (GraphCreatorView.java)

Cette page permet à l’utilisateur de créer son propre graphique. Il peut alors créer des sommets puis les relier entre eux très facilement en quelques clics. Ensuite, l'intérêt de ce mode est que l’utilisateur peut appliquer nos différents algorithmes sur son graphique personnalisé. Finalement, il peut l’enregistrer grâce au bouton “save” ou alors en rouvrir un autre avec le bouton “open”.

Random Graph (RandomGraphView.java)

Sur cette page, l’utilisateur peut générer un graphe aléatoirement grâce à Voronoï (décrit dans l’axe *Création des graphes*). On peut aussi changer les couleurs et appliquer des algorithmes sur le graphe.

Structure du Projet



Rapport de l'avancé du projet

(méthodologie, problèmes rencontrés, idées) :

18 janvier 2022 : Création du groupe, choix du projet et début des recherches sur le sujet de notre projet pour comprendre ce qu'on devait faire. Création du projet sur gitlab et implémentation de Maven dans le projet.

23 janvier 2022 : Première réunion avec la professeure chargée de notre sujet. Début de la structure des classes.Léopold et Paris se mettent en binôme pour commencer l'élaboration des premiers graphes. Daniel, Jade et Alexandre se mettent ensemble pour débuter la confection des algorithmes de coloriage de graphe.

27 janvier 2022 : Création de notre discord et première réunion en distanciel entre membres du projet pour mettre les idées claires sur la

démarche à suivre : Création de graphes, d'algorithmes de coloriage de graphe, création de cartes et mise en relation de ces 3 axes dans la gui pour tester l'affichage. Par la suite nous essayerons de faire un outil dans la gui qui nous permettra de comparer la complexité (temps, mémoire, …) de chaque algorithmes avec différentes cartes.

5 février 2022 : Paris crée l'outil pour transformer une carte dans un graphe et Léopold a créé l'outil pour pouvoir colorier sur une carte.

8 février 2022 : La professeur nous explique comment fonctionne l'un des algorithmes de coloriage de graphes qui s'appelle Kempe et Alexandre et Daniel commencent à réfléchir à la manière de faire pour l'élaborer.

11 février 2022 : Paris crée le backup de notre projet sur drive.

17 février 2022 : Léopold crée un outil mapChooser qui permet soit en mode "false" de glisser une image et un fichier csv vide puis de sélectionner la carte directement pour faire des tests de coloriage pour voir si les frontières sont bien distinctes. Soit en mode "true" de pouvoir sélectionner une carte et dès que l'on clique sur une section de la carte il nous est demandé d'écrire son nom. Cela permet d'enregistrer à des données précises le nom du sommet qui sera après un point sur le graphe.

18 février 2022 : Jade crée la carte France avec son fichier .csv et enregistre chaque section avec son nom.

22 février 2022 : Debrief entre membres de l'équipe avant la réunion avec la professeur pour débugger ou parler de certains bugg et réfléchir aux éventuels solutions. Problème de test, les tests ne sont pas automatisés pour le moment et tout le monde ne sait pas comment tester chaque partie du projet.

24 février 2022 : Léopold crée un outil pour switcher entre une carte et son graphe.

14 mars 2022 : Veille de la réunion entre l'équipe et la professeur chargée de notre projet. Création de ce fichier par Jade listant et décrivant les différents points du sommaire (voir au début du fichier). Alexandre et Daniel en charge de Kempe, ils leur restent quelques questions à éclaircir avec la professeur. Léopold quant à lui à rajouter une page dans la gui pour créer à la main des graphiques. Clic gauche pour poser un vertex et clic droit en maintien pour relier 2 vertex ensemble. Deux boutons sont également là pour ceux qui n'ont pas de souris. Si on veut déplacer la des sommets il faut remettre le mode déplacement avec la quadruple flèche. Également un bouton save (pour sauvegarder le graphe) et open pour ré-ouvrir le graphe. Bug avec la toolbar (décale le Y de 50px) à régler mais rien de dramatique. Jade doit faire la carte des États-Unis et de l'Europe (les fichiers .csv sont prêts il faut juste trouver des cartes qui s'affichent bien dans la GUI et enregistrer chaques données). Paris va créer l'outil qui permettra de calculer la complexité d'un algorithme (nombre d'opérations car le temps est relatif à la machine) et de voir si un graphe est planaire. Suggestion d'amélioration du design de la GUI.

15 mars 2022 : Alexandre et Daniel ont quasiment fini l'algorithme de Kempe mais ils souhaitent tester avec un graphe avec énormément de sommets pour tester si l'algo fait bien des chaînes de Kempe. Un graphe partiellement colorier pour imposer l'utilisation de chaînes de Kempe. Paris va essayer de faire l'algorithme de Voronoi pour créer directement des graphes planaires et créer l'outil permettant de calculer la complexité d'un algorithme (nombres d'opérations). Léopold a réglé le bug de la toolbar, va faire avec Paris Voronoi et peut-être la triangulation. Alexandre va optimiser les algos.

29 mars 2022 : (résumé des 2 dernières semaines) Paris a fait Voronoi et les calculs de complexité, soit tout le dossier outils et avec Léopold ils ont

écrit les classes Vertex et Graph. Alexandre a fait un prototype de gui. Daniel a fini Kempe et a fait également un graphe partiellement colorier pour imposer l'utilisation de chaînes de Kempe. Daniel a également remarqué qu'en utilisant la GUI pour les cartes, quand on alterne Kempe et Greedy la coloration fonctionne bien puis quand on utilise WelshPowell après Greedy ne colorie pas de la même manière qu'auparavant. Petit bug qui sera sûrement réglé rapidement. Jade a ajouté la carte de l'Amérique du Sud, la Chine, l'Ukraine, Paris avec ses arrondissements et l'Afrique mais les fichiers .csv ne sont pas encore finis. Léopold a fait en sorte que les cartes et les graphes sont reliés (quand on colorie la carte ça colorie le graphe) puis des boutons pour appliquer les algos sur les cartes et graphes puis on peut utiliser les flèches gauche et droite pour se déplacer dans le mapchooser et on peut sélectionner une carte avec la touche entrée. Il y a également une nouvelle esthétique pour le menu.

Pour ce qui est des choses à faire, Daniel va optimiser le code des algos de coloriage de graph et peut-être faire une fonction pour vérifier si un graphe est planaire et faire un descriptif de l'algo de Greedy et Kempe. Paris va optimiser Voronoi et les calculs de complexité et avec Léopold ils vont faire un descriptif de l'axe de création des graphiques. Alexandre va travailler sur la GUI au niveau du design et faire un descriptif de l'algo Welsh Powell. Jade va faire les fichiers csv qui ne sont pas encore complets, enregistrer les cartes et envoyer une carte du Japon à Alexandre et en attendant je vais faire également la carte de l'Italie puis faire un descriptif de l'algo de Dsatur.

5 avril 2002 : Alexandre a fait le descriptif de Welsh Powell et compte travailler sur la GUI. Léopold a fait en sorte de faire une mise à l'échelle pour que les cartes soient toujours à la bonne taille de l'écran. Et pareil pour les graphes. Daniel a fait le descriptif de Kempe et Greedy et à modifier le code pour éviter des appels redondant et a remarqué un problème dans Kempe qui est que certains cas précis ne fonctionne pas (demander à Yan Jurski pour Kempe pour voir s'il a une idée pour régler ce

problème). Jade a fait un descriptif de l'algo de Dsatur et a fait la carte de l'Afrique, de l'Amérique du Sud, de la Chine et de Paris. Jade compte faire par la suite la carte de l'Italie, de l'Ukraine et réfléchir à un personnage que l'on pourrait découper en plusieurs morceaux pour faire en sorte sous forme de graph qu'on le reconnaisse.

12 avril 2002 : Alexandre a fait le design de la gui avec Léopold. Jade a fait la carte de l'Italie et de l'Ukraine. Paris a mis à jour les graphiques pour les coordonnées et aussi pour les cartes faire une sauvegarde pour l'enregistrement des coordonnées pour indiquer ce qui a déjà été enregistré. Il a également fait un texte pour expliquer Voronoi et a connecté Voronoi à la création de graphe aléatoire. Léopold a modifié le design de chaque page de la GUI avec Alexandre. Léopold va corriger certains aspects de la GUI (boutons trop gros ou déformé ou mal placés). Daniel a réussi à régler le problème qu'il avait remarqué avec l'algorithme de Kempe.