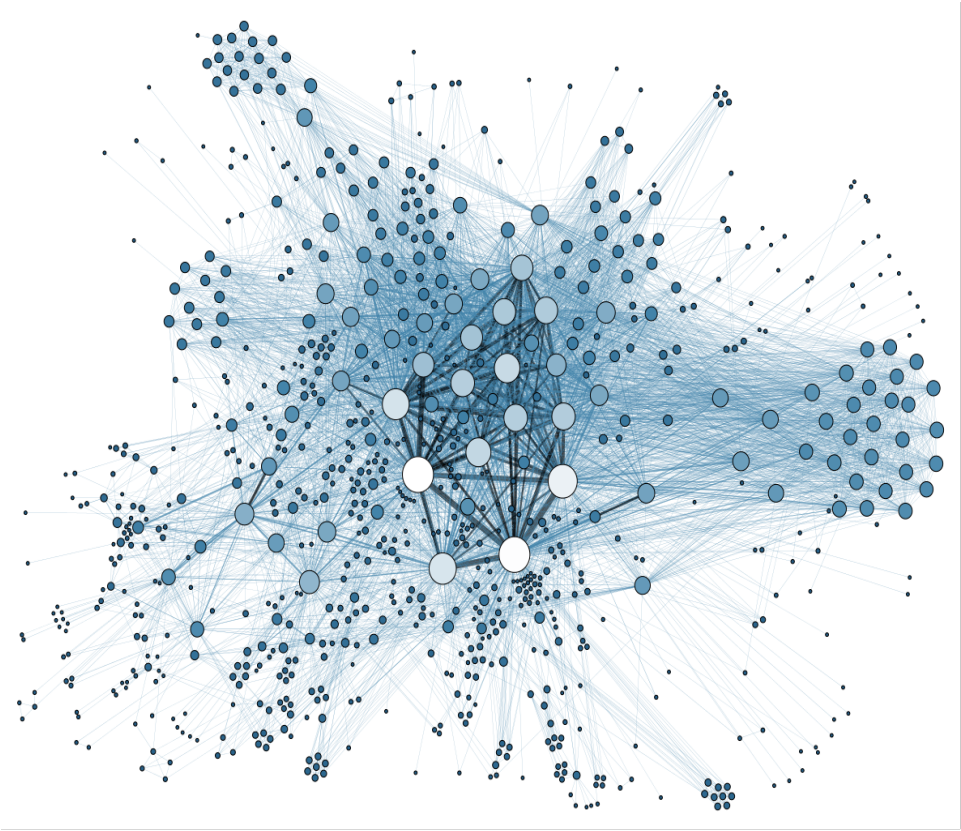
Rapport de projet : Graph Maker



Prof tuteur : David Hébert

Etudiants :

Myrvete Hatoum - Nassim Ahmed Mahmoud - Allaoua Nait-Hamouche - Jean-Eudes Llas

Avril 2015

IUT Villetaneuse
Université Paris 13

Sommaire

1.Introduction

1.1. Présentation du projet

Dans le cadre du dernier semestre de notre deuxième année de DUT Informatique à l'IUT de Villetaneuse, nous sommes amenés à réaliser un projet permettant de valider les acquis de notre formation universitaire.

Ce projet mêle les mathématiques à l'informatique et, parmi la longue liste de projets, allant de la cryptographie à la théorie des graphes, le Graph Maker nous a été confié. En effet, ce dernier consistait à développer une application permettant à l'utilisateur de tracer des graphes avec le nombre de sommets qu'il souhaite, les relier avec des arcs ou des arêtes et faire plusieurs opérations afin d'obtenir le graphe voulu.

L'objectif principal est de pouvoir ensuite faire tourner des algorithmes de la théorie des graphes, sur les graphes créés.

1.2. Cahier des charges

Lors de nos entretiens hebdomadaires avec notre client, Monsieur HEBERT nous a fait part de ses attentes. Il souhaitait que l'on développe une application simple, ergonomique qui puisse permettre à l'utilisateur de créer les graphes et de les manipuler.
Certaines fonctionnalités allaient donc de soi comme la création de sommets, d'arêtes mais aussi le fait de pouvoir changer ces dernières en arcs donc orienter le graphe. Il fallait également arriver à pouvoir changer l'orientation d'une seule arête, en effacer qu'une seule, donner un nom au graphe mais aussi à chacun de ses sommets et une valeur aux arcs.

De plus, nous devions pouvoir indiquer à l'utilisateur si son graphe non orienté est connexe, c'est-à-dire si deux sommets quelconques peuvent être reliés par une chaîne et si le graphe est un arbre ou pas.

Quant aux algorithmes de la théorie des graphes, nous avons choisi d'intégrer dans notre application, les algorithmes de coloration, dit de Brelaz et du chemin le plus court, dit aussi de Dijkstra.
Mais des choses plus techniques encore étaient demandées, comme le fait de pouvoir courber les arcs mais surtout d'intégrer l'outil qui serait bien utile à notre professeur de mathématiques, notamment pour faire ses exercices : le LaTeX.

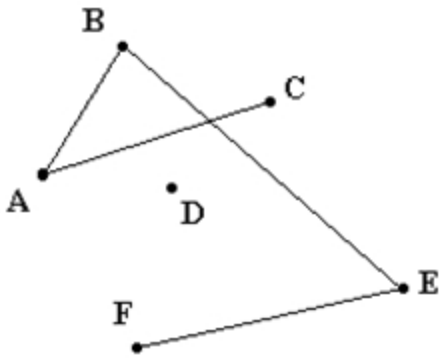
2. Développements mathématiques/informatiques

2.1. Graphe

Quelques définitions générales :

- Graphe, sommet, arête

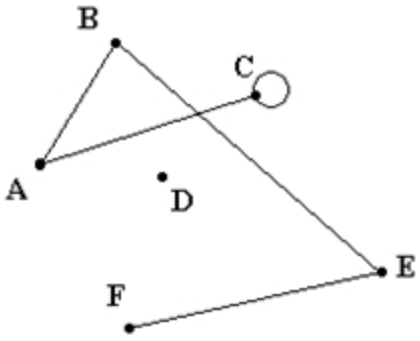
Un graphe est un ensemble de points nommés sommets, reliés ou non par des segments appelés arêtes. Voici le Graphe 1 :



A est un sommet, le segment [AB] est une arête reliant A et B (ou B à A).
D est un sommet isolé, non relié à un autre sommet.

- Boucle

Une boucle est une arête reliant deux fois le même sommet.
Voici le Graphe 2 :



Ici, C est muni d'une boucle.

- Sommets adjacents

Deux sommets sont adjacents lorsqu'ils sont reliés par une arête.
Sur le Graphe 2, A et B sont adjacents, A et E ne le sont pas.

- Graphe simple

Un graphe est simple s'il ne comporte aucune boucle et que deux arêtes ne relient jamais la même paire de sommets.
Ici, le Graphe 1 est simple, le Graphe 2 ne l'est pas.

- Degré d'un sommet

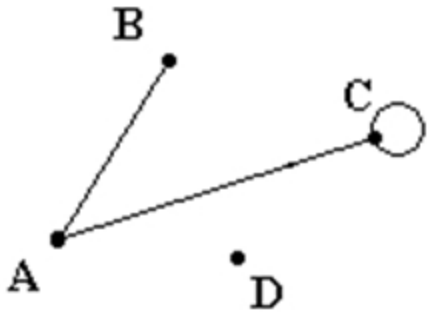
Le degré d'un sommet est égal au nombre d'arêtes qui le relient aux autres sommets.
Dans l'exemple précédent, A est de degré 2, B est de degré 2, D est de degré 0.

- Ordre d'un graphe

C'est le nombre total de sommets.
Les graphes 1 et 2 sont d'ordre 6.

- Sous graphe

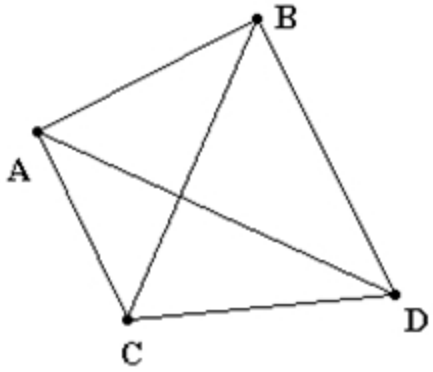
Le sous graphe d'un graphe est un graphe composé de certains de ses sommets avec toutes les arêtes qui les relient.
Voici le Graphe 3 :



Le Graphe 3 est un sous graphe du Graphe 2.

- Graphe complet

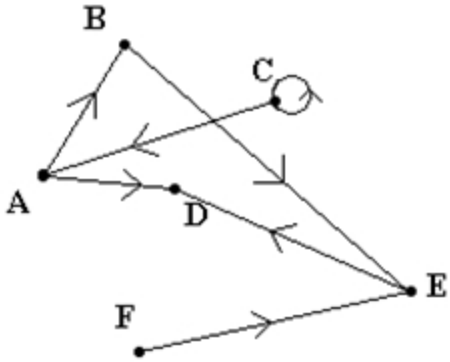
Un graphe est complet si tous les sommets sont adjacents les uns avec les autres.
Voici le Graphe 4 :



Ci-contre, le Graphe 4 est un graphe complet d'ordre 4.
Le Graphe 1 n'est pas complet car par exemple B et C ne sont pas reliés.

- Graphe orienté

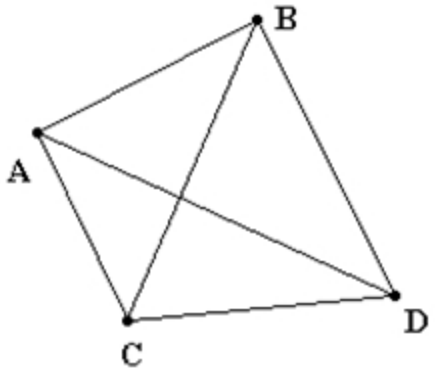
Un graphe orienté est un graphe dont les arêtes sont orientées. Elles ont une origine et une extrémité. Elles ne peuvent être parcourues que dans un sens et deviennent donc des arcs.
Voici le Graphe 5 :



- Matrice associée à un graphe d'ordre n

La matrice associée de n lignes et n colonnes est la matrice où le terme à l'intersection de la ième ligne et de la jème colonne est égal au nombre d'arêtes reliant les sommets i et j.
Voici la matrice du Graphe 4 :

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$



Pour le Graphe 4, on numérote les sommets dans l'ordre alphabétique, 1 pour A, 2 pour B, 3 pour C et 4 pour D.
Pour la première ligne, A n'est pas en relation avec lui–même (pas de boucle), donc dans la case correspondant à la première ligne et la première colonne, on met 0.
Pour les colonnes suivantes (toujours en première ligne), le graphe est simple, complet et A est adjacent à chaque autre sommet une seule fois. D'où les 1 qui complètent la ligne.

- Chaîne simple

Dans un graphe non orienté, une chaîne simple (ou chemin simple dans un graphe orienté) est une chaîne ne passant pas deux fois par une même arête, c'est-à-dire dont toutes les arêtes sont distinctes.

- Chaîne élémentaire

Dans un graphe non orienté, une chaîne élémentaire (ou chemin élémentaire dans un graphe orienté) est une chaîne ne passant pas deux fois par un même sommet, c'est-à-dire dont tous les sommets sont distincts.

- Cycle et circuit

Un cycle est une chaîne simple dont les deux extrémités sont identiques dans un graphe non orienté. L'équivalent dans un graphe orienté est le circuit.

Mais concrètement, à quoi servent les graphes?

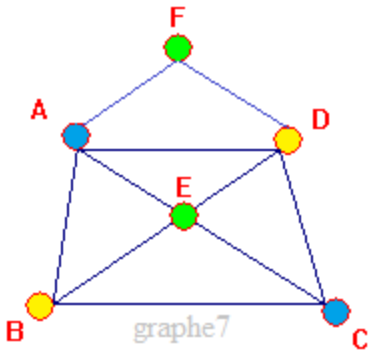
Il y a plusieurs types d'applications. Le premier se rencontre principalement en informatique. Les graphes sont une structure mathématique particulièrement bien adaptée à l'ordinateur, ils servent de « structure de données », c'est-à-dire qu'ils permettent d'organiser des ensembles d'objets, des noms, des nombres, des suites d'opérations et plus encore, de façon simple et pratique à exploiter. Par exemple, la méthode de compression utilisée pour créer un fichier au format « zip » est un algorithme inventé par l'américain David Huffman en 1952, qui consiste en un codage des éléments du fichier à partir d'un arbre.

Dans notre projet, pour modéliser un graphe, nous avons tout simplement fait une classe Graphe. Chaque graphe est constitué d'une liste de sommets et chaque sommet a lui-même une liste contenant ses arcs sortants, ceci est une manière de récupérer son degré afin de l'utiliser, notamment pour l'algorithme de coloration.
Concernant l'orientation du graphe, nous avons choisi de prendre deux variables de classe booléennes. Si la variable est à "true", le graphe est orienté, si elle est à "false", le graphe ne l'est pas.
Dans cette classe, on trouve donc toutes les méthodes relatives au graphe comme celles qui permettent de savoir si le graphe est métrique, s'il est connexe, si c'est un arbre etc ou pas, mais aussi celles qui concernent, par exemple, les matrices des graphes.

2 .2. Algorithme de Brelaz

L'algorithme de coloration de graphes DSAT (Degree SATuration) ou DSATUR, dit de Brelaz, est un algorithme de coloration séquentiel par heuristique, c'est-à-dire que la solution est réalisable mais pas forcément optimale ou exacte.
Inventé par le mathématicien Daniel Brelaz en 1979, il consiste à colorier tous les sommets d'un graphe de façon à ce que deux sommets adjacents n'aient jamais la même couleur.

Prenons un exemple :



1- On détermine les degrés de chacun des sommets du graphe G :

Som(G)	A	B	C	D	E	F
DSAT(x,G)	4	3	3	4	4	2

2 - On trie ces degrés par ordre décroissant
3 - Pour la première itération, on prend le plus grand degré en partant de la gauche et on lui attribue une couleur quelconque.
4 - Ensuite, on détermine le nombre de voisins coloriés de chaque sommet, s'il n'en a pas, on met la valeur de la ligne au dessus.
5 - On revient à l'étape 3 en tâchant de ne pas attribuer la même couleur à deux sommets voisins.

Som(G)	A	D	E	B	C	F
$DSAT_1(x,G)$	4	4	4	3	3	2
$DSAT_2(x,G)$	●	1	1	1	3	1
$DSAT_3(x,G)$	●	2	2	2	●	1
$DSAT_4(x,G)$	●	●	3	2	●	2
$DSAT_5(x,G)$	●	●	●	3	●	2
$DSAT_6(x,G)$	●	●	●	●	●	2
Couleur	1	2	3	2	1	3

Au niveau de la programmation, l'algorithme de Brelaz est assez simple, il suffit de suivre les étapes données plus haut. En pseudo-langage cela donne :

1. Initialisation

```
Pour chaque sommet x
        DSAT[x] = degré de x
Fin pour
```

2. Itérations

```
Tant que il existe des sommets non coloriés
        Pour chaque sommet x non colorié
                Si aucun voisin de x n'est colorié
                        DSAT(x) = degré de x = nombre de voisin de x ;
                Sinon
                        DSAT(x) = nombre de sommet adjacent de x qui sont coloriés ;
                Fin si
        Fin pour

        On choisit un sommet x non colorié tel que DSAT(x) soit le plus grand;
        Si il y a un conflit, on choisit le sommet qui a en plus le degré maximal ;

        On colorie le sommet avec la plus petite couleur possible ;
Fin tant que
```

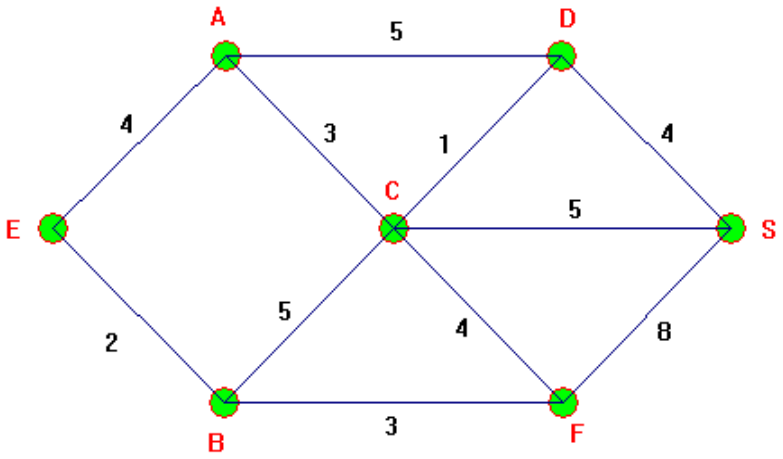
Durant le projet, nous avons rencontré quelques difficultés pour cet algorithme. En effet, d'abord pour trier les degrés à chaque itération, nous nous sommes demandé comment faire pour ne pas compter le sommet de plus grand degré et les sommets déjà coloriés. Il fallait trouver un moyen afin de pouvoir travailler uniquement sur ceux qui n'ont pas encore de couleur. Nous avons finalement décidé de faire une autre liste où l'on a ajouté le sommet de plus grand degré que l'on a supprimé de la liste initiale.

Ensuite, nous avons également eu un peu de mal à définir la couleur en fonction de celles des sommets voisins mais finalement nous avons décidé de faire un tableau de couleurs, on parcoure les couleurs des sommets voisins et dès qu'un sommet n'a pas de couleur, on lui attribue la première couleur qu'il ne trouve pas dans le tableau.

2.2. Algorithme de Dijkstra

L'algorithme de Dijkstra date de 1959 et permet de déterminer le plus court chemin entre deux sommets d'un graphe connexe (orienté ou non).
Pour pouvoir appliquer cet algorithme, il faut que le graphe soit métrique(=avec des valeurs).

Prenons un exemple :



On cherche à optimiser un chemin d'origine E (appelée entrée) et d'extrémité S (sortie).
Chaque arête a une étiquette(toujours positive), représentant un poids.
1 - On affecte d'abord le point d'entrée E de poids 0.
2 - Si un sommet est adjacent, on l'affecte de sa distance à E en indiquant à droite la provenance.
3 -On note les sommets non adjacents symboliquement à l'infini : ∞ .
4 - On "fixe" E et on ferme la colonne E.

E	A	B	C	D	F	S	Fixés
0	4E	2E	∞	∞	∞	∞	E
	4E		7B	∞	5B	∞	B
			(7A) 7B	9A	5B	∞	A
			7B	9A		13F	F
				8C		12C	C
						(12D) 12C	D
						0	S

5 - On repère le poids *P* le plus faible : c'est ici 2E correspondant au sommet B.
6 - On fixe ce point (signalé ici en jaune), on ferme sa colonne (grisée en dessous) et on inscrit en ligne suivante les distances de ce point aux autres sommets adjacents du graphe à l'exception des points déjà fixés (colonnes fermées) *augmentées du poids P*.
Tous les points non adjacents sont mis à l'infini.
Si le poids est identique ou supérieur, on conserve l'ancien et sa provenance.

Ainsi, dans cette exemple, il y a 2 solutions équivalentes de poids 12 : E A C S et E B C S.

En pseudo-langage, l'algorithme de Dijkstra se présente comme suit :

1. Initialisation

```
d_min(x) = 0 ;
sommet_proche(x) = {} ;
Pour tous les sommets différents de x
        d_min(y) = + ∞ ;
        sommet_proche(y)={} ;
Fin pour
```

2. Itérations

```
Tant qu'il existe des sommets non marqués
        On marque un sommet y non marqué tel que d_min(y) soit le plus petit
        possible
        Pour tout sommet z non marqué et successeur/voisin à y
                Si d_min(z) > d_min(y) + λ({y, z})
                 d_min(b) = d_min(y) + λ({y, z}) ;
                 sommet_proche(z) = y ;
                Fin si
        Fin pour
Fin tant que
```

Pour cet algorithme, dans notre classe Dijkstra, nous avons fait un tableau pour les distances minimales, un autre pour les sommets adjacents et un pour les sommets fixés. Nous avons utilisé l'algorithme du marquage afin de pouvoir sélectionner le sommet voisin ayant la plus petite distance.
Mais le principal bug rencontré était dû au simple fait que l'on avait négligé une condition très importante : le fait que les poids ne peuvent être que positifs. Du coup, les calculs de distances étaient faussés et donc le résultat également.

2.4. Autres algorithmes

Nous avons également ajouter d'autres algorithmes pour indiquer à l'utilisateur si, par exemple, le graphe est connexe ou non, c'est-à-dire si deux sommets quelconques et distincts du graphe admettent au moins une chaîne qui les relie.
Ou ci celui-ci est un arbre, c'est-à-dire s'il a un nombre d'arêtes qui équivaut au nombre de sommets -1. Nous avons aussi ajouté des algorithmes permettant de dessiner une clique, un cycle et une chaîne.

Pour l'algorithme de la connexité, nous partions du principe qu'un graphe est connexe si le nombre d'arêtes est supérieur ou égal au nombre de sommets - 1, or cela est faux. Notre solution était donc, à partir d'un sommet de départ, de marquer ses sommets voisins et si tous les sommets ne sont pas marqués, le graphe n'est pas connexe parce qu'un graphe est connexe seulement si tous les points sont reliés entre eux.

2.5. LaTeX

Le LaTeX, prononcé « latec » est un langage créé pour séparer le fond de la forme lors de la création d'un document. Plus clairement, l'utilisateur tape des instructions dans une sorte d'éditeur et structure son texte grâce à des mots, des balises et des commandes propres à LaTeX.
Par exemple, l'utilisateur peut indiquer à LaTeX de placer la première partie de son texte en gras, et une autre en italique. En somme, il décrit comment il veut hiérarchiser l'information. Ensuite, son code est traitée par un logiciel : LaTeX choisit alors les meilleurs agencements et la disposition optimale pour chacun des éléments du document.
Aussi, LaTeX est un des rares logiciels de traitement de texte orienté vers la production de documents scientifiques car outre les équations et autres formules, LaTeX possède un grand nombre de fonctionnalités axées autour de la rédaction d'articles.

Pour notre application, notre professeur voulait pouvoir récupérer le code des graphes créés pour pouvoir les intégrer directement dans LaTeX et ainsi les utiliser dans ses pdf de cours et d'exercices.
Pour l'algorithme, l'enjeu était donc de trouver un moyen de récupérer les positions de chaque sommet, ce que l'on a fait en stockant toutes les lignes dans un tableau et toutes les colonnes dans un autre et en les comparant.
Pour les arcs, une fois que nous avons obtenu la matrice avec les lignes et les colonnes, nous avons pris un sommet et nous l'avons comparé avec son voisin. Par exemple, si la ligne du premier est inférieure à celle du deuxième, alors le premier sommet sera placé en dessous dans le graphe.

3. Présentation des différents outils informatiques

3.1. Langage de programmation

Pour le développement de notre application, aucun langage de programmation n'était imposé. C'est donc tout naturellement que nous avons choisi Java.
En effet, Java est un langage de programmation moderne développé par James Gosling et Patrick Naughton, employés de Sun MicroSystems (aujourd'hui racheté par Oracle) et qui a été présenté officiellement en 1995.

Aussi, on peut faire de nombreuses sortes de programmes avec Java, notamment des applications sous forme de fenêtre, ce que nous avons utilisé dans le cadre de notre projet et il faut noter que cela s'y prêtait particulièrement bien puisqu'en plus d'avoir appris ce langage de programmation durant notre formation à l'IUT, nous avons également appris à faire des interfaces graphiques (Interface Homme Machine), d'autant plus que plusieurs outils permettant de faire des graphes sont fournis dans Java.

De plus, Java est un langage orienté objet, ce qui facilite la conception du programme et contrairement aux langages compilés traditionnels, pour lesquels le compilateur crée un fichier directement exécutable, le code source Java est compilé en un langage intermédiaire dans un fichier portant le même nom que le fichier source à l'exception de son extension.
Cette caractéristique est majeure, car c'est elle qui fait qu'un programme écrit en Java est portable, c'est-à-dire qu'il ne dépend pas d'une plate-forme car une fois le programme en Java créé, il fonctionnera automatiquement sous Windows, Mac, Linux etc. En réalité, le code intermédiaire n'est exécutable sur aucune plate-forme sans la présence d'une machine virtuelle.

3.2. Environnement de travail

La question du logiciel à utiliser pour développer notre programme en Java ne s'est pas posée, puisque l'IUT de Villetaneuse utilise Eclipse version 3.8.

De plus, la caractéristique essentielle d'Eclipse est l'extensibilité de l'environnement. Plus que de se focaliser sur un environnement de développement Java, les concepteurs d'Eclipse se sont efforcés avant tout de créer un socle applicatif sur lequel viennent se greffer des modules.

Pour le développement Java, Eclipse est très complet car il possède : un explorateur de package, une console, la fenêtre de debug, des tâches, l'outline (une autre façon d'explorer les classes), ainsi qu'une barre de repérage interactive des erreurs et des "warnings" présents dans un fichier.
Point important, Eclipse gère ce qu'on appelle un compilateur java incrémental. Une sorte de pré-compilation permanente. Ainsi toute faute de syntaxe ou autre est détectée et signalée au moment de l'écriture du code !
Des corrections sont alors proposées si l'on le désire.

4. Commentaires des résultats obtenus/personnels sur le projet

4.1. Résultats obtenus

Dans l'ensemble, nous sommes assez satisfaits de notre application, tout ce que nous avons fait fonctionne : que ce soit par rapport à la création des graphes, aux différents algorithmes ou au LaTeX.

Cependant, nous regrettons de ne pas avoir pu faire mieux. En effet, il nous manquait du temps pour pouvoir ajouter d'autres options, soigner l'interface graphique et la rendre plus attrayante et ergonomique ou encore pour pouvoir courber les arcs.
Mais nous avons préféré nous focaliser sur les fonctionnalités principales et indispensables à l'application.

4.2. Organisation du travail

Concernant l'organisation de notre travail en groupe, Nassim était notre chef. Il nous a très vite motivé et a eu, dès le départ, des idées intéressantes sur lesquelles nous avons pris appuie.
Aussi, nous avons essayé dans un premier temps de nous répartir les tâches, de façon à ce que nous avancions le plus vite possible et que chacun puisse faire au moins un algorithme.

Mais devant les difficultés que nous avons rencontrés, nous nous sommes rendus compte qu'il était préférable de s'entraider pour une tâche précise au lieu que chacun essaye de s'en sortir dans son coin car cela nous aurait fait perdre encore plus de temps.
Ainsi, pour chaque algorithme, chacun a participé et a donné ses idées afin d'avancer et d'essayer de débloquer la situation.

Cependant, nous avons dû, à un certain moment, nous répartir les tâches car nous arrivions à un stade du projet où nous allions prendre beaucoup de retard si nous faisions autrement. Donc Nassim et Jean-Eudes étaient chargés de faire les plus grands algorithmes, comme le LaTeX par exemple et Allaoua et Myrvete des algorithmes plus petits tel que la connexité et de la rédaction du rapport.
Ainsi, la plus grande difficulté a été de gérer son temps entre les cours, la recherche de stage et le projet.

5. Bibliographie

- http://www.maxicours.com/se/fiche/4/7/417374.html/tes

- http://openclassrooms.com/courses/apprenez-a-programmer-en-java

- http://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_profondeur

- http://stackoverflow.com/questions/20792947/move-an-oval-in-java

- http://docs.oracle.com/javase/7/docs/api/

- http://yallouz.arie.free.fr/terminale_cours/graphes/dijkstra.php

- http://serge.mehl.free.fr/base/index_concr3.html#graphe

- http://arxiv.org/pdf/1207.4616.pdf

6. Annexes

Compte rendu de la semaine du 9/02/15

Projet : Graphe Maker

Date et heure : 13 Février 2015, 13h

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- Maquette
- Code
- Fiche

Tâches effectuées	Par qui ?	Quand ?	Durée
Maquette	Myrvete, J-Eudes, Nassim, Allaoua	12/02/15	2h/1h30/1h/1h
Code(initiation JavaScript)	Nassim, Allaoua, J-Eudes, Myrvete		11h/2h/2h/2h
Fiche	Myrvete	12/02/15	45min

Prochaine réunion **Date et heure :** - 18/02/15, 13h/13h30

Ordre du jour : - fonctions nécessaires au code/interface

Compte rendu de la semaine du 16/02/15

Projet : Graphe Maker

Date et heure : 18 Février 2015, 13h30

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- Editeur : début du travail avec Java (révision + code)
- Détermination des fonctions nécessaires + interface

Tâches effectuées	Par qui ?	Quand ?	Durée
Maquette (amélioration)	Myrvete, J-Eudes, Nassim, Allaoua	15/02/15	1h30
Code	Nassim, Allaoua, J-Eudes, Myrvete		15h/5h/6h/4h
Fiche	Myrvete	15/02/15	10min

Prochaine réunion **Date et heure :** - 5/03/15, 13h/13h30

Ordre du jour : - Editeur : fonctions graphe

Compte rendu de la semaine du 23/02/15

Projet : Graphe Maker

Date et heure : 25 Février 2015, 13h

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- Fonctions graphes(sommets, arcs, gomme, arcs <-> arêtes)

Tâches effectuées	Par qui ?	Quand ?	Durée
Maquette	Myrvete, J-Eudes, Nassim, Allaoua		1h30
Code	Nassim, Allaoua, J-Eudes, Myrvete		8h/3h/5h/4h
Fiche	Myrvete		10min

Prochaine réunion **Date et heure :** - 5/03/15, 13h/13h30

Ordre du jour : - Création classes métiers, dialogue

Compte rendu de la semaine du 02/03/15

Projet : Graphe Maker

Date et heure : 5 Mars 2015, 13h

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- Création des classes métiers, dialogue

Tâches effectuées	Par qui ?	Quand ?	Durée
Code	Nassim, Allaoua, J-Eudes, Myrvete		20h/3h/3h/3h
Fiche	Myrvete	4/03/15	10min

Prochaine réunion **Date et heure :** - 11/03/15, 13h/13h30

Ordre du jour : - Sélection des arcs/Résolution des bugs

Compte rendu de la semaine du 09/03/15

Projet : Graphe Maker

Date et heure : 11 Mars 2015, 13h30

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAIT-AMOUCHE
Myrvete HATOUM

Ordre du jour :

- Sélection des arcs/Résolution des bugs

Tâches effectuées	Par qui ?	Quand ?	Durée
Code	Nassim, Allaoua, J-Eudes, Myrvete		5h/2h/4h/2h
Fiche	Myrvete	10/03/15	10min

Prochaine réunion **Date et heure :** - 18/03/15, 13h/13h30

Ordre du jour : - Courbure des arcs/Matrice

Compte rendu de la semaine du 16/03/15

Projet : Graphe Maker

Date et heure : 18 Mars 2015, 13h

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- courbure des arcs/Matrice
- méthode pour trouver les voisins
- étude des algorithmes Brelaz et Djikstra

Tâches effectuées	Par qui ?	Quand ?	Durée
Code et algorithmes	Nassim, Allaoua, J-Eudes, Myrvete		22h/20h/15h/12h
Fiche	Myrvete		10min

Prochaine réunion **Date et heure :** - 25/03/15, 13h/13h30

Ordre du jour : - Algorithme de coloration

Compte rendu de la semaine du 23/03/15

Projet : Graphe Maker

Date et heure : 25 Mars 2015, 13h

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- Fin de coloration/Dijkstra/clique/cycle/chaine/connexité/arbre

Tâches effectuées	Par qui ?	Quand ?	Durée
Code	Nassim, Allaoua, J-Eudes, Myrvete		60h/18h/60h/13h
Fiche/Début du rapport de projet	Myrvete		5h

Prochaine réunion **Date et heure :** - 30/03/15, 13h/13h30

Ordre du jour : - Fin Dijkstra/Résolution bugs

Compte rendu de la semaine du 30/03/15

Projet : Graphe Maker

Date et heure : 30 Mars/3 Avril 2015

Team : Nassim AHMED-MAHMOUD
Jean-Eudes LLAS
Allaoua NAITAMOUCHE
Myrvete HATOUM

Ordre du jour :

- Fin Dijkstra/Résolution bugs/LaTeX
- Amélioration interface/Méthode isClique()/Chargement-Sauvegarde
- Fin du rapport de projet/Préparation de la soutenance

Tâches effectuées	Par qui ?	Quand ?	Durée
Rapport/Présentation de soutenance	Myrvete, J-Eudes, Nassim, Allaoua		35h/3h/3h/3h
Dijkstra/LaTeX/ Bugs	Nassim/ J-Eudes		60h/60h
Méthode isClique()/Sauvegarde-Chargement	Allaoua	5/04/15	25h