Rapport
Euler goes on a walk

Sujet proposé par Adrienne Lancelot

EL HOUARI Nohaïla 22001594
HAMACHE Doryan 22007196
MAKTOUT Rayana 71805581
SBITRE Chaïmae 22001674
SAVOCA Giovanni Maria 22206691

Le sujet :

Pour ce quatrième semestre de licence, nous avons eu une UE consistant à la réalisation d'un projet. Grâce à nos bases acquises le semestre dernier sur le logiciel GitLab, et nos compétences en Java, nous avons pu implémenter différents algorithmes permettant de trouver un chemin Euler ou hamiltonien dans un graphe.

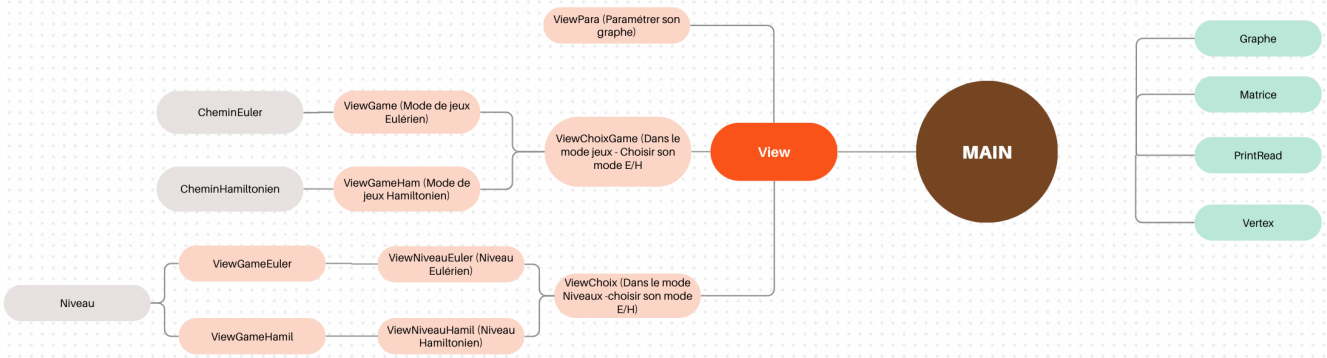
Les règles du jeu :

Le Jeu Euler goes on walk se base sur les chemins eulériens et hamiltonien. Un chemin eulérien consiste à parcourir toutes les arêtes une seule et unique fois alors qu'un chemin hamiltonien consiste à parcourir par tous les sommets une fois et une seule. Notre projet est sous forme de jeu avec des graphes en rapport avec les chemins eulériens et hamiltoniens.

Sommaire :

1. Différents Algorithmes
2. Jeu
3. Éditeur de graphe
4. Niveaux
5. Étude de complexité
6. Erreurs rencontrées

Organigramme du projet



Présentation :

Notre projet se présente sous la forme d'un jeu. Il est divisé en trois parties :

- La première est l'édition de son propre graphe, vous pourrez créer votre propre graphe avec le nombre de sommets et d'arête que vous voulez.

- La deuxième est le mode jeu, celui-ci se divise en deux pour les graphes au chemin Eulérien ou Hamiltonien et il comporte des graphes générés aléatoirement.

- La troisième partie est composée de niveaux, vous avez différents niveaux de difficulté pour des graphes de plus en plus compliqués avec le choix de chemin Eulérien ou Hamiltonien également.

Nous avons implémenté au départ du projet les graphes sous forme de matrices mais aussi de listes d'adjacences, nous les avions implémentés tous les deux mais au final nous nous sommes rendu compte que les matrices étaient plus intéressantes dans notre projet car elle permettait directement de pouvoir relier un sommet avec un autre sommet pour pouvoir ajouter une arête c'est donc pour cela que nous avons abandonné l'idée de garder les listes d'adjacence par la suite.

1) Différents Algorithmes :

Nous avons préféré nous orienter vers une interface jeu plutôt que seulement une partie algorithme car cela nous paraissait plus intéressant et plus intuitif pour nous de faire cela sous forme de jeu avec différent niveaux et autres fonctionnalités comme la création de graphe ou l'extension avec le parcours Hamiltonien.

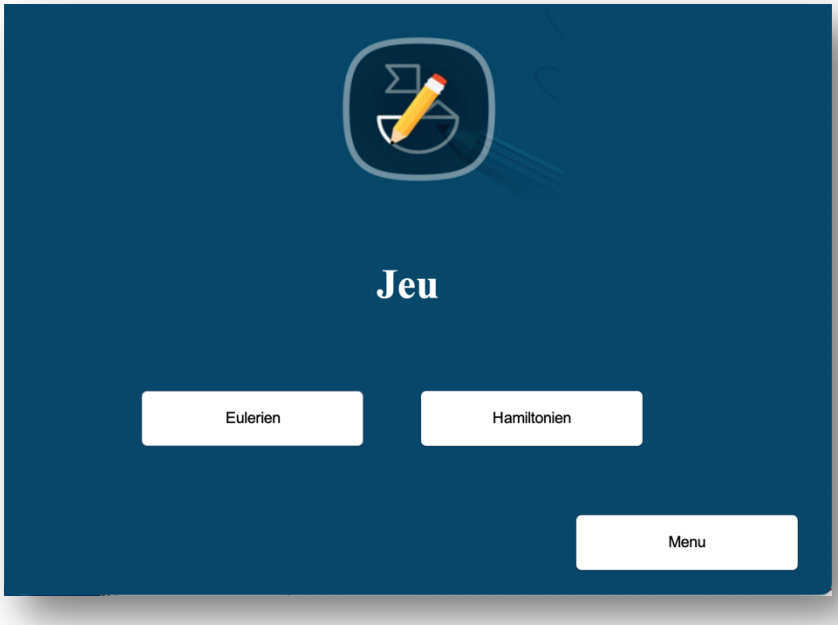
Une partie Jeu sous forme de jeu par défaut qui permet de tester ou de s'entraîner une partie Niveau qui prend différents niveaux avec la difficulté des graphes qui augmentent au fur et à mesure que le niveau augmente qui permet donc de jouer. Et une partie éditeur qui elle permet de tout gérer soi-même et donc de créer notre propre graphe selon nos envies c'est à dire de placer les sommets le nombres, les arêtes et les connecter comme on veut et d'ensuite importer ce graphe créé dans la partie niveau pour pouvoir jouer sur des graphes que l'on a créés nous-même.

Ceci était les grands requis de l'implémentation de notre projet qui était de pouvoir faire un parcours eulérien afin d'avoir un algorithme qui retrace le chemin eulérien ainsi qu'une partie éditeur afin de pouvoir créer nous-même des graphes.

2) Jeu

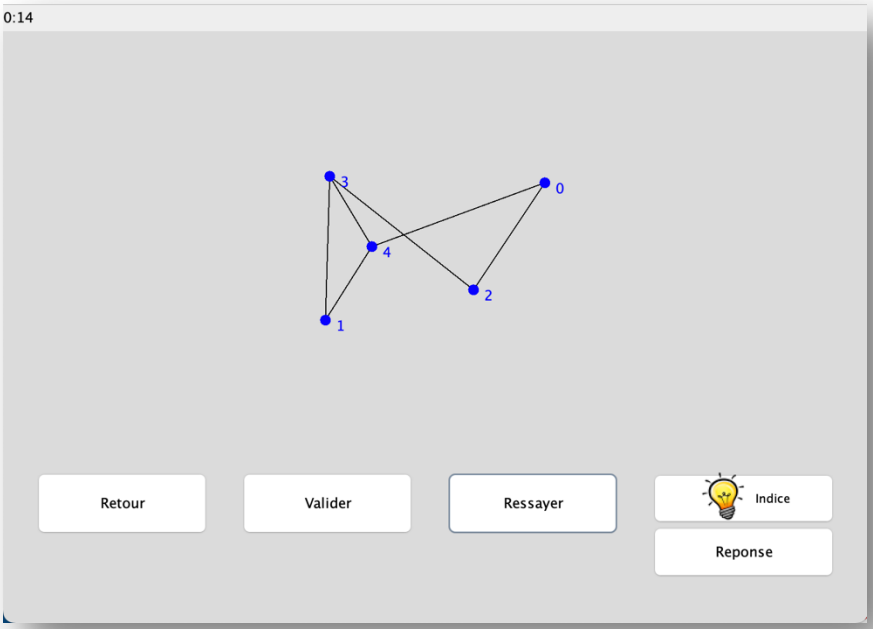
Nous avons dans notre projet deux grandes classes qui font dans l'ensemble la même chose mais se présente différemment. Tout d'abord nous avons la classe pour la partie jeu Eulérien et avec ceci, l'extension que nous avons implémenté qui est la partie Hamiltonien

Les classes sont comme une sorte de jeu par défaut ou jeu aléatoire qui permettent de lancer une partie sur un graphe avec un nombres de sommets et d'arêtes aléatoire et d'y jouer selon le choix qu'on veut c'est à dire eulérien ou hamiltonien. On choisit en fonction que l'on veut jouer en mode eulérien ou hamiltonien.



Cette page se présente avec un bouton timer qui a été implémenté pour faire apparaitre les minutes et les secondes que l'on prend pour terminer le niveau. D'un panneauGraphe qui permet d'avoir l'emplacement de notre graphe ainsi qu'un panneauBoutton qui prend un bouton menu pour revenir au choix de niveaux, un bouton réessayer qui permet de recommencer si l'on a fait une erreur de parcours ou si l'on veut reprendre le parcours depuis le début sur le même graphe, un bouton indice et réponse qui eux deux permettent d'avoir des aides pour jouer. Indice affiche à chaque fois qu'on clique dessus un bout du chemin que l'on doit faire pour terminer la partie. Réponse lui qui affiche directement toute la solution de la partie. Et enfin un bouton valider qui permet de confirmer le résultat du parcours qu'on a effectué et de nous dire si l'on a gagné ou perdu. Si on finit par l'emporter, un message de victoire s'affiche suivi d'un message de score qui te donne ton score réalisé sous forme d'étoiles sur 10 en fonction du temps qu'on met et du fait qu'on ait été aidé par les aides ou non.

Le message affiche aussi le temps qu'on a pris pour terminer la partie. Ce fichier est accordé à un fichier texte ("Score.txt ") qui stock chaque résultat avec le nom d'utilisateur de la personne en question qui permet d'avoir comme une sorte de résumé de chaque partie depuis la création du jeu qui permet de voir la liste des victoires, la liste des personnes qui ont déjà gagné et ainsi que le résultat en fonction de son propre score par rapport aux sommets de chaque partie et du temps mis. A chaque fin de partie lorsque l'on réussit son parcours de graphe nous avons la possibilité de cliquer sur un bouton suivant afin d'avoir un nouveau graphe qui se présente pour qu'on puisse rejouer et cela a l'infini.



3) Éditeur de graphe

Cette partie de notre projet gère la création d'une interface graphique permettant à l'utilisateur d'interagir avec le graphe pour sa création, d'effectuer diverses opérations et d'afficher les résultats.

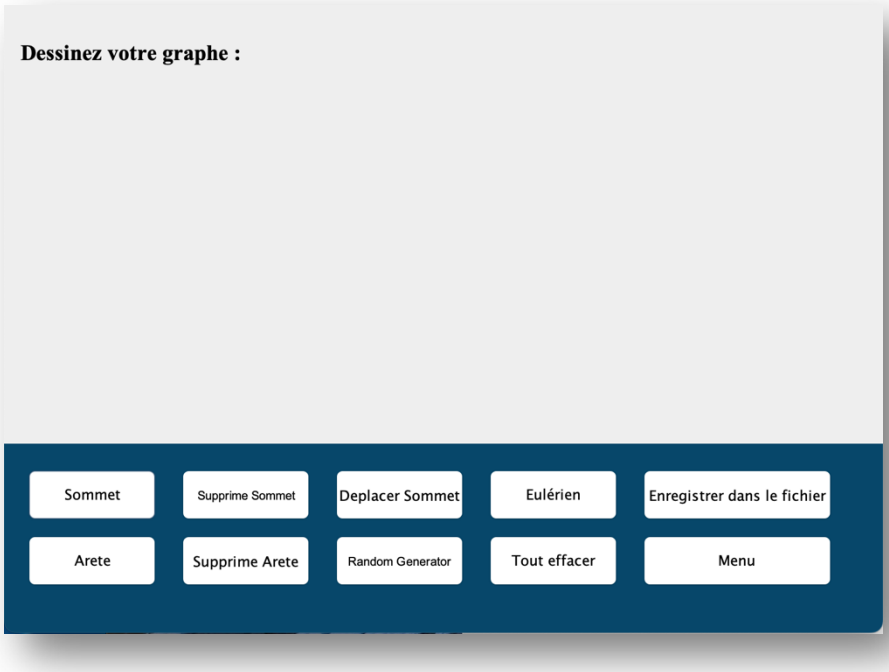
Nous avons plusieurs méthodes auxiliaires afin de gérer l'affichage du graphe créé.

- drawCircle(Graphics g, int x, int y, int radius) : dessine un cercle dont le centre se trouve aux coordonnées (x, y) et dont le rayon est spécifié dans le contexte graphique fourni.
- fillCircle(Graphics g, int x, int y, int radius) : remplit un cercle dont le centre est situé aux coordonnées (x, y) et dont le rayon est spécifié dans le contexte graphique fourni.

- drawLine(Graphics2D g2d, int x1, int y1, int x2, int y2) : dessine une ligne entre les points (x1, y1) et (x2, y2) sur le contexte graphique fourni.
- redrawGraph() : supprime le panneau et redessine le graphique en tenant compte des modifications apportées aux sommets et aux arêtes.

Dans notre interface nous avons plusieurs boutons qui effectuent différentes taches :

- "Menu" retourne à la fenêtre du menu principal.
- "Sommet" qui nous permet de créer des sommets.
- "Arête" qui nous permet de créer une arêtes en sélectionnant deux sommets.
- "Supprime Arête" : supprime une arête en sélectionnant deux sommets.
- "Déplacer Sommet" : déplace un sommet.
- "Euler" : vérifie si le graphique actuel possède un chemin Eulérien et dessine une icône verte ou une icône rouge en fonction du résultat.
- "Tout effacer" : permet de tout supprimer.
- "Random Generator " : génère une matrice aléatoire avec un nombre aléatoire de sommets et d'arêtes.
- "Enregistrer dans le fichier" : Si le graphe est eulérien et connexe il enregistre la matrice dans un fichier et affiche un message de confirmation. Dans le cas contraire, un message d'erreur est affiché.

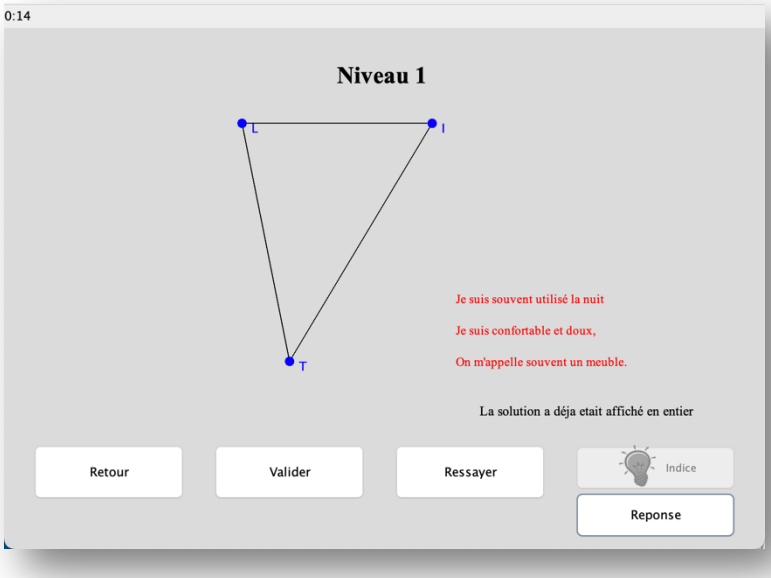


4) Niveaux

Pour la partie Niveau, nous avons deux modes de jeu comme pour la partie Jeu : eulérien et hamiltonien. Après avoir choisi un des deux modes nous avons une interface graphique qui permet de choisir un niveau.

Pour les niveaux eulérien nous avons 6 niveaux avec un niveau ou nous pouvons importer un fichier, ce fichier a été créés grâce à la partie éditeur.

Lorsqu'on appuie sur un niveau nous avons la même interface graphique que la partie jeu, avec un bouton réessayer, réponse, indice, retour et valider qui marche de la même manière que pour la partie jeu. Ici les graphes ne sont pas aléatoires ils ont été initialisés.

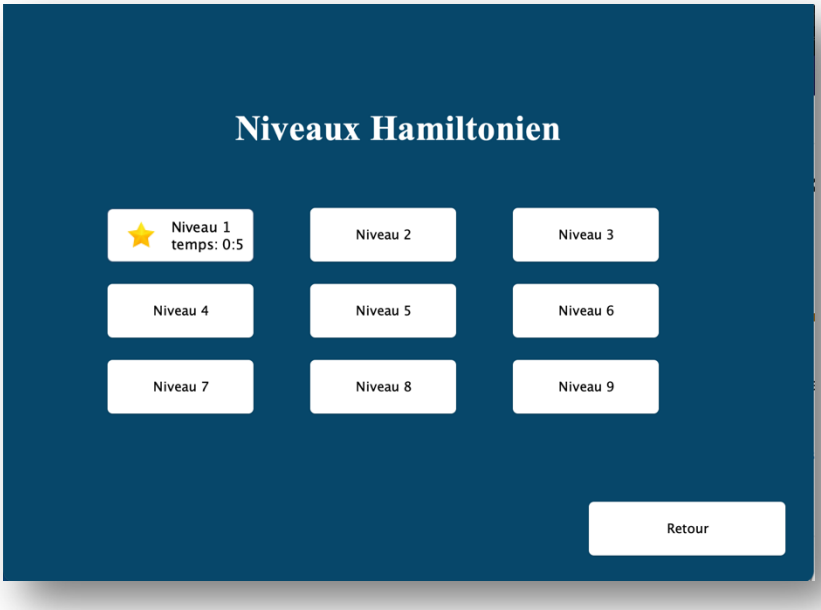


Après la validation du graphe lorsque le chemin sélectionné est correct nous avons une étoile qui s'affiche dans le bouton du niveau sélectionné et également le temps effectué. Pour le bouton importer graphe lorsque le chemin sélectionné est correct il n'y a que le temps qui s'affiche qu'on peut améliorer comme tous les autres niveaux.

Pour les niveaux hamiltonien nous avons 9 niveaux qui ont quasiment la même fonction. Pour les chemins hamiltonien nous avons décidé de modifier un peu le système et ne pas mettre des chiffres pour les sommets mais des lettres. Donc nous avons caché des mots dans le graphe pour aider à l'utilisateur a trouvé le chemin. Les indices sont des devinettes pour trouver le mot. La réponse est directement donnée par nous-même et les graphes ne sont pas aléatoire. Les coordonnées sont fixes contrairement à la partie niveau eulérien.

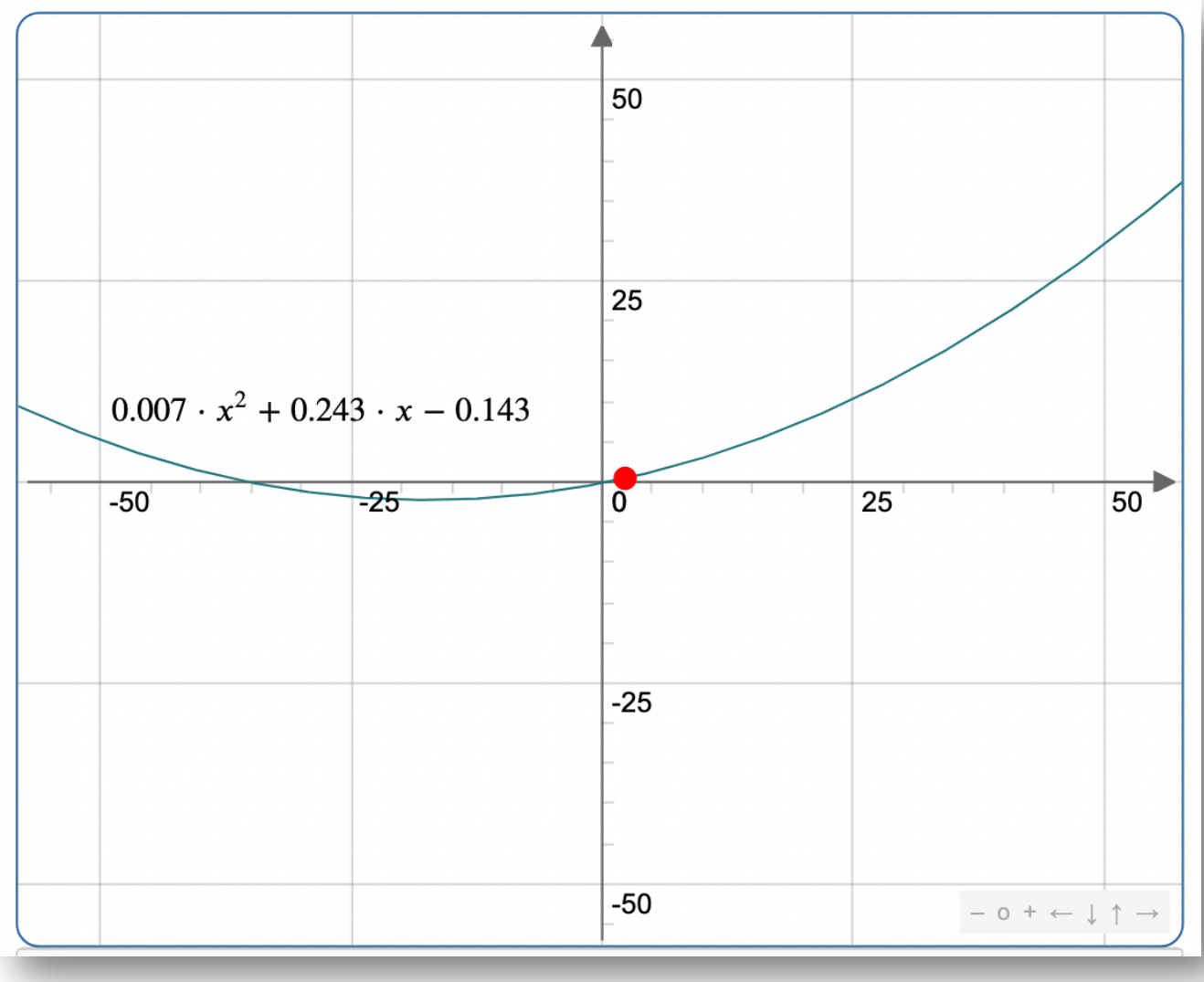
Nous avons le même système d'étoile et de temps que les niveaux eulérien, ainsi que la fonctionnalité des bouton retour valider et réessayer.

Et enfin nous devons entrer un nom lorsqu'on appuie sur un des modes de niveau.



5) Étude de complexité

Nous avons effectué une étude de cas sur la complexité pour les chemins Eulérien et Hamiltonien. Pour les chemins Eulérien, nous avons pu pousser les tests pour plusieurs graphes et nous avons pu faire des analyses plus précises pour savoir ce qui se passait en pratique. Les résultats de ces tests ont pu nous donner une équation quadratique et donc une courbe en fonction du temps (en seconde) et du nombre de sommet au départ.



6) Erreurs rencontrées

On a rencontré un problème au niveau de la sélection des arêtes et sur la récupération de l'arête sélectionnée afin de pouvoir vérifier s'il y a bien une arête entre les deux sommets ou pas par exemple.

Au niveau de l'affichage du graphe cela nous a pris du temps, en effet on a eu du mal à dessiner les arêtes et les sommets et savoir quelle arête a été sélectionnée pour pouvoir la colorier et savoir si on était déjà passée par cette arête ou non. Il nous fallait également vérifier que l'arête sélectionnée après est bien reliée à celle d'avant.

Un autre problème également au niveau de la solution complète, notre solution complète n'était pas très correcte car lorsqu'on ne sélectionnait pas toutes les arêtes mais tous les sommets cela nous montre que c'était correct.

De plus, on a eu un autre problème sur le bouton réessayer car on n'arrivait pas à afficher le même graphe. On a dû créer une matrice origine et méthodes auxiliaire pour cela.

Ensuite nous avons eu aussi un problème sur les indices car on a remarqué tardivement que lorsque nous appelons à voir un chemin eulérien nous renvoyons une seule solution. Nous aurions dû implémenter une fonction pour avoir toutes les combinaisons possibles. Mais le temps ne nous l'a pas permis.

Nous avons eu des difficultés pour pouvoir stocker les scores de chaque partie, au départ nous avions eu juste un système de point qui donnait directement le score après chaque victoire mais nous n'avions aucun moyen de le stocker. Nous avons réglé ce problème à l'aide d'un fichier Score qui stock à chaque fois la partie en question.

Pour la partie niveau, nous voulions faire de même mais cette fois ci garder seulement les 3 meilleurs scores afin que l'utilisateur qui joue puisse sur chaque niveau voir les meilleurs temps des autres et pouvoir essayer de les battre. Nous avons rencontré plusieurs erreurs d'abord pour stocker les 3 meilleurs, ceci a été réglé avec un système de fichier comme pour la partie jeu, mais cette fois-ci nous avions les 3 premières lignes du fichier afin de les retrouver mais donc avec ceci un nouveau problème est survenu sur le fait de récupérer les 3 meilleurs et de stocker leurs résultats car nous n'avons pas réussi à placer en fonction du score le 1er, 2eme ou 3eme meilleurs et de même pour le stockage des résultats car à chaque fois qu'on relançait notre programme, les 3 meilleurs restaient mais leur score disparaissaient. Il aurait fallu peut-être stocker les 3 meilleurs résultats dans un fichier comme pour les meilleurs joueurs mais nous n'avons pas eu le temps de le faire.

- Partie Éditeur :

Nous avons eu des problèmes de détection des clics de souris : nous avons eu des difficultés à détecter correctement les clics de souris et à les associer aux actions correspondantes, telles que l'ajout, la suppression ou la modification de sommets et d'arêtes. Pour cela nous avons créé des variables booléennes. De cette manière, nous avons pu gérer correctement les différentes actions basées sur les clics de souris.

Nous avons eu aussi des problèmes d'affichage du graphique : nous avons rencontré des problèmes pour dessiner correctement le graphique sur le panneau, en particulier lors de la mise à jour du graphique après des modifications.

Nous avons eu également des problèmes liés à la modification de la position d'un sommet : nous avons rencontré des problèmes pour déplacer correctement un sommet dans le graphique. Pour résoudre ce problème, nous avons introduit une variable booléenne qui nous permettait de déterminer si un sommet avait effectivement été sélectionné ou non.

Nous avons rencontré des problèmes pour générer un graphe aléatoire dans lequel les sommets étaient en position centrale et facilement visibles pour l'utilisateur. Pour résoudre ce problème, nous avons travaillé sur la fonction de génération aléatoire, en ajoutant une logique pour placer les sommets plus uniformément dans la zone de dessin, garantissant ainsi qu'ils soient visibles et accessibles à l'utilisateur

- Partie Études de complexité

Nous avons eu un problème sur l’implémentation des fonctions Hamiltonienne et sur son étude de cas. L’étude des fonctions pour un chemin Hamiltonien est théorique car il n’existe pas d’algorithmes efficaces connus pour résoudre un problème np-complet en temps polynomial pour tous les graphes contrairement au chemin eulérien qui peut être possible en temps linéaire.