Compte rendu réunion 06/03/2018

1) Travail accompli

Le but de la première partie du projet consistait à générer un graphe à partir de la base de données. Ce que nous avons donc réalisé. Le programme dans son état actuel récupère les informations nécessaires à la génération de graphe dans la base de données. Pour ce faire nous avons installé MySQL Workbench pour avoir la BDD en local, ainsi que MySQL Connector C++ pour s'y connecter directement depuis notre programme C++. Nous avons fait le choix de nous abstenir d'utiliser du PHP pour récupérer la base de données pour simplifier l'exécution du programme (tout se passe dans le même programme) ainsi que pour découvrir l'utilisation de MySQL en C++, même si ce fût légèrement fastidieux au départ.

Une fois la connexion à la BDD faite et les données sur les cartes récupérées, nous créons dans notre programme une map<int, Card> pour stocker ces dernières. Le premier int correspond à l'id de la carte et l'objet Card contient les différentes informations de chaque carte. Cette structure nous permettra de faire tous nos traitements tels que la génération de graphe, ou la création des arêtes.

Notre objet graphe contient un vector<Edge> pour stocker les arêtes ainsi qu'une matrice d'adjacences int** dont nous nous servirons pour l'algorithme de construction de deck. A ce stade, le programme n'est pas en mesure de stocker toutes les cartes contenues dans la base de données du fait du nombre d'arêtes : 17 847 cartes donc possiblement 318 515 409 arêtes (pour un graphe complet). Pour le moment nous pouvons créer des arêtes pour entre 10 000 et 15 000 cartes.

Nous avons également mis en place une première façon de calculer la valeur d'une arête en fonction de la caractéristique couleur de deux cartes. La formule est la suivante :

```
Si (nbCommuns == 0)
        colorValue = 0
Sinon
        colorValue = 5 – (max(nbColors1, nbColors2) – nbCommuns)
```

Avec en paramètres :

- nbCommuns : nombre de couleurs en communs entre les deux cartes
- nbColors1 et nbColors2 : nombre de couleurs de chaque carte
- colorValue : la valeur de l'arête pour la caractéristique couleur
- 5 n'a pas été choisi par hasard non plus, il correspond au nombre de couleurs différentes

Cette formule nous permet de donner une valeur comprise en 0 et 5 qui nous parait cohérente. Exemples (En prenant B, G, R, U et W comme lettres pour les couleurs) :

Carte 1 : BGR	Carte 1 : BGR	Carte 1 : BRRWU
Carte 2 : BGR	Carte 2 : BGRU	Carte 2 : W
Valeur de l'arête : 5	Valeur de l'arête : 4	Valeur de l'arête : 1

Enfin, point que nous avons oublié d'évoquer lors de la réunion, nous avons mis notre projet sur GitHub pour faciliter le travail collaboratif, notamment avec VisualStudio, ce qui fût légèrement fastidieux également, notamment dû au fait que le projet sous VisualStudio nécessite une configuration particulière pour l'utilisation de MySQL Connector C++. Projet que vous pouvez retrouver à l'adresse suivante : https://github.com/guillaumebourgeois/Projet-M2

2) Sujets abordés

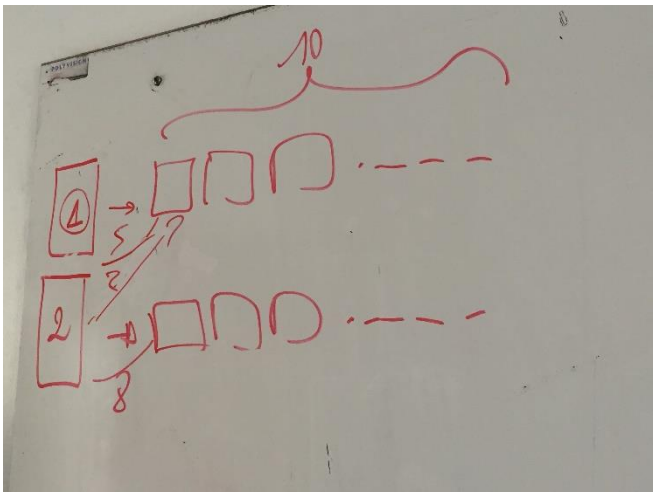
Lors de la réunion nous avons d'abord présenté le travail accompli (voir la partie 1). Nous avons discuté principalement de l'implémentation et des différentes méthodes pour palier au problème de mémoire que nous rencontrons. Nos professeurs et Julien nous ont d'ailleurs présenté plusieurs solutions et nous ont montré différents points que nous devrons revoir (voir partie 3). Julien nous a expliqué la méthode CRS (Compress Row Storage). En réalité le problème de mémoire réside dans le fait que pour chaque arête nous avons 3 entiers (idArete, idCarte1, idCarte2). Il faut donc que nous diminuions la quantité d'informations que nous stockons dans chaque arête et que nous changions l'implémentation de la structure qui stock ces arêtes.

Nous avons également parlé des autres critères pour lier 2 cartes et pour donner une valeur totale à une arête, notamment avec la caractéristique de l'édition qui apparaît comme un facteur important dans la constitution d'un deck. En effet, lors d'un concours les joueurs constituent leurs decks en choisissant des cartes parmi les 3 derniers blocs (un bloc contenant 2 éditions et une édition 340 à 350 cartes). Ils jouent donc sur les 3 derniers blocs, les 6 dernières éditions, soit 1500 cartes seulement.

Nous avons aussi présenté la méthode de calcul pour donner une valeur à la caractéristique couleur entre 2 cartes et M. Rabat nous a indiqué qu'il faudra rajouter un lien avec les cartes sans couleur et leur donner une valeur neutre (3 par exemple) et exclure toutes les cartes de type LAND et de sous-type BASIC de l'importation de la BDD.

Un autre point important de la réunion a été de démarrer une réflexion sur le problème d'optimisation que nous allons devoir traiter ainsi que des algos que nous pourrions utiliser pour proposer des cartes à l'utilisateur. Dans un premier temps l'algo du plus court chemin a été évoqué mais c'est en réalité un algo nous donnant les plus proches voisins qui sera mis en place. Lors de la réunion 3 premières solutions ont été présentées brièvement (avec N le nombre de cartes que nous voulons proposer à l'utilisateur, hypothétiquement 10) :

- La première consiste à prendre les N premières cartes les plus proches communes à deux premières cartes de départ (voir photo du tableau ci-dessous)
- La deuxième consiste à donner une valeur entre chaque carte de départ et leurs N cartes les plus proches, puis de prendre les N cartes ayant les valeurs les plus élevées parmi les N*(nombre de carte de départ) proposées
- La dernière consiste en un graphe dynamique dont les cartes sélectionnées forment un unique sommet à partir duquel de nouvelles valeurs sont établies vers toutes les autres cartes.



Nous devons donc réfléchir à ces trois solutions à l'implémentation de ces 3 solutions, en tester au minimum une et, si possible, en trouver d'autres.

Si nous allons jusqu'à l'implémentation d'une ou plusieurs de ces solutions, le programme pourra prendre entrée un fichier contenant les id des cartes de départ lors de la proposition de cartes et devra rendre en sortie un fichier du même type avec les id des cartes composant le deck final (40 cartes car sans les terrains).

Nous avons enfin parlé de la nécessité de programmer des réunions plus souvent, de faire des comptes-rendus de réunion pour mieux visualiser le suivi des travaux et des tâches à réaliser pour la prochaine fois, ce qui nous amène au dernier point.

3) Travail à faire

Voici donc la liste des travaux à réaliser pour la prochaine réunion :

- Extraire les types LAND et sous-types BASIC de l'importation de la BDD
- Changer l'implémentation
    - Matrice 2D ➔ matrice 1D (pour une allocation mémoire contiguë)
    - Supprimer des paramètres de l'objet arête
- Inclure les autres critères dans le calcul de la valeur d'une arête : type, capacité, coût en mana, édition)
    - Valeur totale d'une arête = SOMME(valeur*coefficient)
- Réfléchir aux problèmes d'optimisation proposés, si possible les implémenter et les tester
- Mettre en place le système pour les fichiers d'entrée/sortie
- Faire ce rapport =)