Compte-rendu réunion 13/03/2018

1. Travail accompli

* Changement de l’implémentation
  + Création d’une bimap<int, int> pour stocker les ids des cartes correspondant aux ids de la matrice d’adjacence et pouvoir y accéder en O(1) dans les deux sens
  + Vector<Card> : inchangé
  + Vector<int> : pour les cartes qui vont être extraites du fichier
  + Vector<Edge> : pour le stockage des arêtes
* Changements et améliorations :
  + Class Edge :
    - Il n’y a plus que la variable pour le total
    - Idée proposée pour le stockage des valeurs des autres critères
  + Class Card :
    - Stockage des couleurs avec une chaine de bits
    - Vector<int> pour les blocs, éditions, types, sous-types, capacités
  + Class Graph :
    - Le calcul du critère color a été déplacé dans la fonction de création de matrice d’adjacence (Grosse amélioration en temps de calcul)
    - Ajout du calcul des critères édition, bloc, types, sous-types, capacités, total
    - Histogramme du poids des arêtes
* Calcul des critères :
  + Edition : 0 si bloc et éditions différents, 1 si même bloc, 2 si même édition
  + Type et sous-type différenciés (si jamais nous voulions donner des coefficients différents à ces deux critères) :
    - Type : +1 par type commun aux deux cartes
    - Sous-type : +1 par type commun aux deux cartes
  + Capacités : +1 par capacité commune aux deux cartes
  + Total : SOMME (critère\*coefficient\_du\_critère)
    - Pour le moment tous les coefficients sont à 1, ils seront ajustés en fonction des résultats obtenus
* Requêtes SQL :
  + Cartes
  + Blocs + éditions
  + Types
  + Sous-types
  + Super-types (éventuellement à supprimer plus tard si non-nécessaire)
  + Capacités
  + Ids : Pour avoir la correspondance multiverseid ⬄ ids des cartes
* Problème d’optimisation : 2 algos ont été implémentés indépendamment (1 par membre du binôme), ils seront présentés lors de la réunion
* Ce début de rapport =)

1. Sujets abordés
2. Travail à faire