SAÉ 2.02 – Exploration algorithmique d'un problème
SAÉ 2.01 – Développement d'une application

David Auger – Isabelle Robba

1 SAÉ 2.02 – Exploration algorithmique d'un problème

1.1 Le pitch

L'Association Publique des Pokémonistes Libres et Indépendants ou APPLI regroupe des collectionneurs de cartes Pokémon. Chaque année, les membres décident entre eux de certaines ventes de cartes entre joueurs. Les membres ne faisant pas confiance aux services professionnels pour délivrer leurs précieuses raretés, le président de l'APPLI, qui habite à Vélizy, a décidé de partir avec sa camionnette faire le tour de la France et effectuer toutes les livraisons de cartes entre membres, les transactions financières associées ayant déjà été effectuées.

En tant que membre de l'APPLI et éminent informaticien, on vous demande de réaliser une appli pour l'APPLI, qui permette d'étudier les différents itinéraires pour effectuer ces livraisons dans un but économique et écologique.

1.2 Les données

Deux fichiers à utiliser pour tous les scénarios sont donnés :

- la liste des membres de l'APPLI, avec un pseudonyme et une ville
- un fichier qui contient toutes les distances deux à deux entre les grandes villes de France

1.3 Les scénarios

Trois fichiers correspondant à trois scénarios distincts sont également donnés :

- le scénario 0 est volontairement de taille très réduite afin d'effectuer vos premiers tests facilement
- le scénario 1 est de taille plus importante : il contient une trentaine de ventes
- le scénario 2 est volontairement de grande taille afin d'éprouver les limites de calcul des algorithmes mal conçus

Un scénario consiste en une liste de ventes du type **membre1 −> membre2** qui signifie qu'il va falloir passer prendre des cartes chez **membre1** et les déposer chez **membre2**, où **membre1** et **membre2** sont deux membres de l'APPLI ; **membre1** est le vendeur et **membre2** est l'acheteur.

1.4 Les solutions

Une solution du problème pour un scénario donné consiste à donner un itinéraire sous la forme d'une liste de villes, commençant et finissant à Vélizy, qui soit tel que pour chaque vente du scénario, on passe obligatoirement dans la ville du vendeur avant de passer dans la ville de l'acheteur. La longueur de cette solution est le nombre total de kilomètres effectués lors de cet itinéraire.

1.5 Le travail algorithmique attendu

Les algorithmes à réaliser sont les suivants :

- ≻ l'algorithme présenté en cours de graphes, qui calcule une seule solution (pas obligatoirement la meilleure)
- ≻ un algorithme fondé sur une solution heuristique, c'est-à-dire une solution qui soit basée sur des optimisations locales, qui ne donne pas forcément le résultat optimal mais qui soit rapide à effectuer
- ≻ un algorithme qui calcule les k meilleures solutions, en partant de la meilleure et par kilométrage croissant, k est un paramètre réglable

1.6 Remarques sur l'algorithmique du projet

Trouver les meilleures solutions consiste ici à énumérer toutes les solutions possibles de sorte à assurer toutes les ventes. C'est un principe d'énumération en "force brute" mais on peut essayer de limiter le temps de calcul en raisonnant.

Notez que pour la solution optimale, il peut être nécessaire de passer deux fois dans une ville mais c'est le maximum. En effet, il peut y avoir des ventes de **A** à **B** et de **B** à **A**, ce qui force à passer deux fois en **A** ou en **B**. Passer une troisième fois dans une ville **A** est inutile : au premier passage en **A** on prend toutes les cartes à vendre au départ de **A**, et quitte à repasser plusieurs fois en **A** ensuite, on pourrait supprimer les passages intermédiaires et faire toutes les ventes à destination de **A** en dernier, gagnant ainsi des kilomètres.

Il est possible de modéliser le problème comme un problème de graphe : pour chaque ville **A**, on introduit un sommet **A+** s'il y a des ventes au départ de la ville, et **A-** s'il y a des ventes à destination de la ville. On modélise chaque vente **A−>B** comme un arc de **A+** vers **B-**. On cherche alors à énumérer tous les sommets de ce graphe de sorte à ce que pour chaque arc **(A+,B-)**, **A+** figure avant **B-** dans l'ordre, et à minimiser le nombre de kilomètres total d'une ville à l'autre. Bien entendu, il est souvent préférable d'enchaîner directement les sommets **C+** et **C-** pour une même ville **C**, car la distance entre ces deux sommets est de 0.

2 SAÉ 2.01 – Développement d'une application

L'objectif de la SAÉ 2.01 est donc de réaliser une application qui résoudra le problème algorithmique posé dans la SAÉ 2.02, problème de parcours de graphe, posé par votre client : l'Association Publique des Poké-monistes Libres, ou APPLI.

L'application sera écrite en java et possèdera une interface homme-machine pour :

- ≻ gérer les scénario : choisir d'un scénario, créer un nouveau scénario, modifier un scénario existant
- ≻ visualiser les solutions obtenues : villes parcourues et kilométrage
- ≻ ...

Pour le bon déroulement de votre travail, il est très important que vous réfléchissiez aux algorithmes et au choix des différentes classes de l'application et de ses différentes structures de données (tableaux, collections, dictionnaires) avant de vous lancer dans la programmation.

3 Calendrier

- **Le mercredi 4 juin à 10 heures**, un rapport commun aux deux SAÉ devra être déposé sur ecampus, ainsi qu'un dossier compressé contenant le code de votre projet, les classes de tests du modèle et la documentation sous la forme javadoc. Le contenu du rapport est détaillé dans la section suivante.

  Tout retard dans la remise du rapport sera pénalisé.

- Une soutenance commune aux deux SAÉ sera organisée **le mercredi 4 juin à partir de 13 heures**, elle aura lieu sur les machines du département, vous présenterez votre application et répondrez individuellement aux questions des enseignants.

- Les évaluations finales des ressources R2.01, R2.02 auront lieu la semaine du 9 juin, elles comporteront des questions sur les SAÉ 2.01 et 2.02

4 Rapport

Le rapport doit contenir les sections ci-dessous :

1. Une introduction qui présente l'application réalisée (merci de ne pas recopier l'énoncé)
2. La conception
   - ➢ Une présentation des différents composant de votre application : modèle, vue, contôleur
   - ➢ Pour chacun des composants, une présentation détaillée de ses classes. Pour les classes du modèle vous présenterez en détail les structures de données utilisées et expliquerez vos choix
   - ➢ Une présentation des algorithmes utilisés pour résoudre les problèmes
3. Une conclusion contenant un bilan de votre travail, ainsi que les tâches non réalisées et les perspectives d'évolution ou d'amélioration
4. Les annexes
   - ➢ Le dossier de tests unitaires concernant les classes du composant modèle
   - ➢ Le lien vers le dépôt git contenant votre code, ses tests et sa documentation ; la qualité de votre code sera prise en compte dans la notation