Structures de données II

Enoncé du projet

Université de Mons – Année académique 2017 – 2018

Professeur V. Bruyère – Assistant G. Devillez

Le projet comporte les étapes suivantes:

1. Lecture et compréhension du sujet et de l'énoncé du problème;
2. Analyse et implémentation;
3. Remise du code, d'un rapport et entretien.

Certaines ressources sont disponibles sur la page cours de la plate-forme e-learning[1].

1 Sujet et référence

Le sujet du projet concerne le "map overlay" (superposition de cartes). Une copie du Chapitre 2 *Line Segment Intersection - Thematic Map Overlay* du livre *Computational Geometry*[2] est disponible sur la plate-forme e-learning.

Cette référence constitue votre ressource principale sur le sujet. Les pages à lire sont celles numérotées 19-29 (Introduction et Section 2.1). Elles concernent le "map overlay" dans sa forme la plus simple où les cartes sont représentées par des segments de droite. Le problème se ramène au calcul des intersections des segments de chaque carte.

Dans la Section 2.1, on y explique comment calculer ces intersections de manière efficace. Les algorithmes proposés se basent sur deux structures de données qui sont des arbres binaires de recherche équilibrés utilisant deux ordres différents.

2 Enoncé du problème

Il est demandé d'écrire un programme qui permette de résoudre le problème du "map overlay" pour des cartes représentées par des segments de droite.

- selon la méthode décrite dans la Section 2.1;
- en respect des complexités annoncées.

Une interface graphique permettra d'afficher les différentes cartes, ainsi que la carte résultant du "map overlay". Les cartes en entrée pourront être chargées à partir de fichiers, où construites segment par segment par l'utilisateur. En plus de visualiser la carte résultant du "map overlay", l'utilisateur pourra demander de voir comment la "sweep line" (droite de balayage) balaye l'écran et repère une à une les intersections de segments.

Comme dans le livre de référence, on fait l'hypothèse qu'*il n'y a pas deux segments qui se superposent* (c'est-à-dire que toute paire de segments a au plus un point d'intersection).

[1] `https://moodle.umons.ac.be/course/view.php?id=184`

[2] de Berg, M. , van Kreveld, M., Overmars, M. and Schwarzkopf, O., *Computational Geometry – Algorithms and Applications*, Second Ed., Springer, 2000.

3 Etapes de votre travail

3.1 Lecture et compréhension du sujet

Votre travail commence par la lecture de la Section 2.1 de *Computational Geometry* (*cf.* section 1) et la compréhension approfondie de celle-ci, en particulier :

- les deux structures de données $\mathcal{Q}$ et $\mathcal{T}$ d'arbre de recherche équilibré et leur ordre sous-jacent;
- les algorithmes FindIntersections(S), HandleEventPoint(p) et FindNewEvent(s_l, s_r, p);
- les algorithmes d'insertion d'une donnée, de suppression d'une donnée, de recherche du minimum, de recherche d'un successeur, de recherche d'un prédécesseur dans la structure d'arbre binaire de recherche équilibré;
- les cas particuliers des segments horizontaux et des points d'intersection appartenant à au moins trois segments.

Pour vous aider dans votre compréhension, tentez de répondre aux questions suivantes:

- Avez-vous bien compris la structure de données $\mathcal{Q}$, son ordre sous-jacent? Quelle est sa taille au pire? Quelles sont les opérations de base que doit supporter la structure $\mathcal{Q}$?
- Même question pour $\mathcal{T}$.
- Voyez-vous comment implémenter la structure de données $\mathcal{Q}$ de telle sorte que ses opérations de base vérifient les complexités annoncées (voir page 28)?[3]
- Même question pour $\mathcal{T}$.
- Pour $\mathcal{T}$, voyez-vous comment repérer facilement les ensembles $L(p)$, $U(p)$ et $C(p)$, et effectuer les instructions 5. et 6. de l'algorithme HandleEventPoint(p), cela en un temps en $O(m(p)\log_2 n))$ où $m(p)$ est la taille de $L(p) \cup U(p) \cup C(p)$ et n est la taille de $\mathcal{T}$?
- Voyez-vous comment gérer les cas particuliers, en particulier le cas des segments horizontaux?

3.2 Analyse et implémentation

3.2.1 Analyse

Avant de commencer l'implémentation de votre programme, il conviendra de réaliser une première analyse du sujet de ce projet. Le but de cette phase est d'apporter une première réflexion sur les différentes parties du projet afin de partir sur de bonnes bases pour son implémentation.
Vous remettrez un rapport d'analyse qui contiendra :

- une description détaillée (accompagnée d'un exemple) des deux structures de données utilisées ($\mathcal{Q}$ et $\mathcal{T}$) ainsi qu'une explication de leur ordre sous-jacent;
- une explication concernant les algorithmes présentés dans le chapitre de référence;
- une première intuition quant à votre gestion des cas particuliers des segments horizontaux et des points d'intersection appartenant à au moins 3 segments.

[3] Elles doivent être en $O(\log_2 m)$ où m est la taille de $\mathcal{Q}$

- une description complète, mais succincte, des différentes étapes de votre futur programme, en y mentionnant bien l'utilisation des différentes structures.

Vous devez écrire votre rapport de manière à ce qu'il soit compréhensible pour une personne n'ayant pas connaissance du projet, ni des structures utilisées.

3.2.2 Implémentation

L'implémentation se fera en *Java* (voir par exemple le livre *Java Concepts*[4]). Votre code devra être documenté au format *javadoc*. La classe contenant la méthode `main` permettant de lancer votre programme sera clairement identifiée par un nom commençant par `Test`.

3.2.3 Format de fichiers

Pour ce projet, une carte peut-être vue comme un ensemble de segments. Plusieurs cartes vous sont proposées sur la plate-forme Moodle et sont stockées dans des fichiers formatés de la façon suivante:

- Pour chaque segment, une ligne du fichier contient quatre nombres réels : $x\ y\ x'\ y'$, qui sont les coordonnées (x, y), (x', y') des deux points définissant le segment;

D'autres fichiers de ce type seront utilisés pour tester votre programme.

3.2.4 Opérations supportées

Votre programme doit pouvoir supporter les opérations suivantes :

a) charger une carte, c'est-à-dire lire un fichier au format adéquat;

b) créer une nouvelle carte saisie segment par segment, et stocker celle-ci dans un fichier au format adéquat;

c) afficher une carte;

d) modifier une carte déjà chargée et permettre de l'enregistrer dans un fichier;

e) afficher le résultat du "map overlay" sur les cartes choisies par l'utilisateur;

f) afficher le balayage de l'écran par la "sweep line" et la façon dont elle repère une à une les intersections de segments.

3.3 Rapport final

Votre rapport final ne doit pas faire référence au rapport d'analyse. Vous pouvez cependant reprendre des parties de celui-ci. Votre rapport final contiendra:

- les explications en français de chaque structure de données *importante* utilisée;
- les explications en français du principe et du fonctionnement de chacun de vos algorithmes *importants*;
- une note sur la complexité de chacun de vos algorithmes *importants*;

[4] Horstmann, C., *Java Concepts*, 4th Ed., Wiley, 2005

- un diagramme de classes de votre implémentation;
- une conclusion comprenant une réflexion sur le projet (apports, difficultés, comparaison de vos résultats avec la théorie, ...).

Votre rapport répondra donc aux questions énoncées plus haut dans la Section 3.1.
Le but de ce rapport final est de pouvoir comprendre grâce à sa lecture :

- votre raisonnement;
- votre résolution du problème;
- et votre implémentation.

Encore une fois, vous devez écrire votre rapport de manière à ce qu'il soit compréhensible pour une personne n'ayant pas connaissance du projet, ni des structures utilisées.

4 Calendrier

- **Vendredi 10 novembre 2017 : composition des groupes**
  Pour cette date, vous aurez communiqué la composition de votre groupe : une feuille d'inscription est disponible sur la porte du bureau de G. Devillez (2E14). Le projet se fait par groupe de **deux étudiants**.
- **dimanche 11 février 2018 à 12h : remise du rapport d'analyse**
  Vous déposerez via la plate-forme e-learning votre rapport d'analyse (conforme aux informations demandées dans la section 3.2.1), au format PDF.
- **Au second quadrimestre : remise du code et du rapport final**

  a) *Code*
  Vous déposerez via la plate-forme e-learning une archive (zip, tgz, ...) *ne contenant que les fichiers .java* de votre programme (pas de fichiers .class) et les fichiers nécessaires à la compilation du programme s'il y en a.

  b) *Rapport final*
  Vous déposerez via la plate-forme e-learning votre rapport final (conforme aux informations demandées dans la section 3.3), au format PDF.

  La date finale sera déterminée après la remise du rapport d'analyse en tenant compte des échéances des projets dans les autres cours.

  Attention, un rapport ou un code de trop mauvaise qualité peut mener à une seconde session, peu importe la qualité globale du projet. Veillez au moins à mettre en pratique les consignes données sur moodle.
- **Après la remise du projet : entretien**
  Une date et un horaire de passage pour une défense seront communiqués à chaque groupe. Lors de cet entretien, vous présenterez brièvement la manière dont vous avez abordé le projet (15 minutes), et des questions seront ensuite posées **à tous les étudiants** du groupe.

5 Remarques supplémentaires

1. Les dates des dépôts sont strictes. Le site n'acceptera plus de dépôt après l'heure imposée.
2. Si vous avez des questions – pendant toute la durée du projet – vous prendrez rendez-vous par e-mail avec G. Devillez (`gauvain.devillez@umons.ac.be`).

Bon travail!