Structures de données II

Énoncé du projet

Université de Mons — Année académique 2022–2023

Professeur V. Bruyère — Assistant P. Hauweele

1 Sujet et référence

Le sujet du projet concerne le « *windowing* ». Une copie du Chapitre 10 « *More Geometric Data Structures, Windowing* » du livre « *Computational Geometry* » [1] est disponible sur la plateforme Moodle [2]. En complément, la section 5.5 de ce même livre est également disponible.

Cette référence constitue votre ressource principale sur le sujet. Les pages à lire sont celles numérotées 110–111 (Section 5.5), 219–221 et 226–230 (principalement Section 10.2).

Le début du Chapitre 10 présente la technique de *windowing*. Dans le cadre de ce projet, elle sera utilisée sur un ensemble de segments horizontaux et verticaux (voir page 212).

La section 10.2 du Chapitre 10 traite des arbres de recherche à priorité (appelés « *priority search trees* » dans le livre). Cette structure de données T permet de stocker un ensemble de points de $\mathbb{R}^2$. En gros, elle est à la fois une file de priorité [3] sur les coordonnées en x et un arbre binaire de recherche sur les coordonnées en y. De plus, elle est d'une certaine façon équilibrée. Cette structure permet d'obtenir efficacement l'ensemble des points de T qui sont contenus dans une fenêtre donnée sous la forme de deux intervalles (p.ex. $(-\infty : x'] \times [y : y']$).

Le Chapitre 10 s'appuie sur l'hypothèse que deux points ont leurs coordonnées en x distinctes et leurs coordonnées en y distinctes. La page 111 montre comment on peut se passer de cette hypothèse.

2 Énoncé du problème

Il est demandé d'écrire un programme qui permette d'appliquer le technique de *windowing* à un ensemble de segments de droites horizontaux et verticaux de $\mathbb{R}^2$. La fenêtre pourra être

- bornée selon les deux composantes, c'est-à-dire de la forme $[x : x'] \times [y : y']$;
- partiellement bornée en la composante en x, c'est-à-dire de la forme $(-\infty : x'] \times [y : y']$ ou $[x : +\infty) \times [y : y']$;
- partiellement bornée en la composante en y.

Une interface graphique permettra d'afficher l'ensemble des segments, ainsi que le résultat de l'application du *windowing* sur base d'une fenêtre personnalisable.

Pour ce faire, il est demandé d'utiliser la structure de données d'arbre de recherche à priorité. Une telle structure sera construite à partir d'un fichier contenant l'ensemble des segments à traiter.

1. de Berg, M. , van Kreveld, M., Overmars, M. and Schwarzkopf, O., *Computational Geometry – Algorithms and Applications*, Second Ed., Springer, 2000.
2. `https://moodle.umons.ac.be/course/view.php?id=184`
3. Une file de priorité est un arbre binaire tel que pour tout nœud, la donnée qui s'y trouve est supérieure ou égale (respectivement inférieure ou égale) à celles de ses fils.

3 Étapes de votre travail

Le projet comporte les étapes suivantes :

1. lecture et compréhension du sujet ;
2. résolution de l'exercice préliminaire ;
3. analyse et implémentation ;
4. remise du code, d'un rapport et défense orale.

3.1 Lecture et compréhension du sujet

Votre travail commence par la lecture du Chapitre 10 du livre « Computational Geometry » (cf. Section 1 du présent document) et la compréhension approfondie de celui-ci, en particulier :

— la structure de données d'arbre de recherche à priorité ;
— la construction d'une telle structure à partir d'un ensemble donné P de points de $\mathbb{R}^2$;
— l'algorithme de recherche des points de P qui sont contenus dans une fenêtre W donnée.

3.2 Exercice préliminaire

Avant de commencer l'implémentation, un exercice préliminaire est prévu. Celui-ci consiste en l'écriture d'un rapport contenant vos réponses aux questions données ci-dessous. Vos réponses devront être justifiées et accompagnées d'une courte explication.

Une fois celui-ci rendu, il vous sera possible de demander un feedback sur vos réponses afin d'être certain que vous avez bien compris le problème et les différentes structures de données et algorithmes avant de commencer l'implémentation.

Questions :

1. Si on ne considère que les coordonnées en x, voyez-vous que celles-ci sont organisées selon une file à priorité ? Où est la coordonnée minimum (maximum) ? Cette file à priorité est-elle un tas ? Justifiez.
2. Si on ne considère que les coordonnées en y, voyez-vous que celles-ci sont organisées selon un arbre binaire de recherche ? Où est la coordonnée minimum (maximum) ? Justifiez.
3. De quelle façon est équilibré un arbre de recherche à priorité ?
4. La construction d'un arbre de recherche à priorité peut se faire en $O(n \log_2 n)$ si n est le nombre de points de $\mathbb{R}^2$ contenus dans l'arbre. Expliquez comment on peut y parvenir et comment le prouver. Il faut sans doute utiliser une autre structure de données qui permet de calculer efficacement la médiane.
5. La structure d'arbre de recherche à priorité permet d'obtenir efficacement l'ensemble des points de T qui sont contenus dans une fenêtre donnée de la forme $(-\infty : x'] \times [y : y']$. Expliquez comment adapter l'algorithme proposé pour traiter une fenêtre de la forme $[x : +\infty) \times [y : y']$, $[x : x'] \times (-\infty : y']$ ou $[x : x'] \times [y : y']$.
6. Les auteurs du Chapitre 10 font l'hypothèse que tous les points ont des coordonnées bien distinctes en x et en y. Expliquez pourquoi.
7. Une technique est présentée à la page 111. Celle-ci permet de simuler des coordonnées distinctes pour un ensemble de points quelconques et une requête. Expliquez comment.
8. Dans le chapitre, la technique présentée est adaptée à des points. Quelles différences importantes aurait-on avec des segments de droite ? Comment peut-on adapter la technique ?

3.3 Analyse et implémentation

3.3.1 Langage de programmation

L'implémentation doit être en *Java* (voir par exemple le livre *Java Concepts*[4]) et doit utiliser au mieux les concepts orienté objet de ce langage (héritage, interface, etc.). Votre code doit être documenté au format *javadoc*. La classe contenant la méthode `main`, permettant de lancer votre programme, doit être clairement identifiée par un nom commençant par `Test`.

3.3.2 Format de fichiers

Un ensemble de segments est stocké dans un fichier sous la forme 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 ;

— ces lignes sont précédées d'une première ligne contenant quatre nombres réels : $x_0\ x'_0\ y_0\ y'_0$, qui décrivent une fenêtre $[x_0 : x'_0] \times [y_0 : y'_0]$ contenant l'ensemble de tous les segments.

Des exemples de fichiers seront déposés sur Moodle.

3.3.3 Opérations supportées

Votre programme doit pouvoir supporter les opérations suivantes :

a) charger un ensemble de segments, c'est-à-dire lire un fichier au format adéquat et stocker les données dans un arbre de recherche à priorité T ;

b) afficher cet ensemble de segments ;

c) saisir une fenêtre (bornée complètement ou partiellement) ;

d) afficher le résultat de l'application du *windowing* sur base de la fenêtre proposée en utilisant les structures de données proposées par l'ouvrage de référence.

3.4 Rapport final

Votre rapport final doit contenir :

— un diagramme UML – reflétant la structure de votre programme – accompagné d'une description des différentes classes et interfaces utilisées ;

— 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 dans le pire des cas de chacun de vos algorithmes *importants* ;

— un court mode d'emploi de votre programme (comment compiler, exécuter et utiliser votre programme)

— une conclusion comprenant une réflexion sur le projet (apports, difficultés, comparaison de vos résultats avec la théorie...).

Le but de ce rapport final est de pouvoir comprendre grâce à sa lecture (par une personne ne connaissant pas le projet) :

— votre raisonnement ;

— votre implémentation ;

— et votre résolution du problème.

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

4 Calendrier

vendredi 18 novembre 2022 : composition des groupes

Pour cette date, vous aurez communiqué la composition de votre groupe à l'endroit prévu à cet effet sur Moodle. Le projet se fait par **groupe de deux**.

vendredi 23 décembre 2022 à 12h00 : remise du rapport de l'exercice préliminaire

Vous déposerez sur la plateforme e-learning votre rapport au format PDF comprenant vos réponses aux questions. Si un groupe désire un feedback sur son rapport, il peut prendre rendez-vous par e-mail avec Pierre Hauweele (pierre.hauweele@umons.ac.be).

lundi 17 avril 2023 à 12h00 : remise du code et du rapport final

Code Vous déposerez sur la plateforme Moodle une archive (zip, tar.gz, tar.xz, 7z) *ne contenant que les fichiers .java* de votre programme et les fichiers utiles à la compilation.

Rapport final Vous déposerez sur la plateforme Moodle votre rapport final (conforme aux informations demandées dans la section 3.4), au format PDF.

après le lundi 17 avril 2023 : défense orale

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 chaque membre** du groupe.

5 Remarques supplémentaires

1. Les dates des dépôts sont strictes. Le site n'acceptera plus de dépôt après midi.
2. Si vous avez des questions – pendant toute la durée du projet – vous prendrez rendez-vous par e-mail avec P. Hauweele (pierre.hauweele@umons.ac.be).

Bon travail !