Structures de données II

Enoncé du projet

Université de Mons – Année académique 2024 – 2025

Professeur V. Bruyère – Assistant C. Grandmont

Avis important : Ce projet sera réalisé de manière **individuelle**. Pour rappel, tout plagiat ou fraude est punissable par le règlement des études.

Celui-ci 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 le site du cours de la plate-forme e-learning [1].

1 Sujet et référence

Le sujet du projet concerne le "Orthogonal Range Searching". Une copie du Chapitre 5 *Orthogonal Range Searching* 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 99-105 (Section 5.2) et les pages 110 et 111 (Section 5.5). Elles traitent d'une structure de donnée sous forme d'arbre appelée *Kd-Tree*, permettant d'obtenir efficacement l'ensemble des points contenus dans une région rectangulaire du plan, utile notamment lors d'une requête basique à une base de données concernant deux critères numériques.

2 Enoncé du problème

Il est demandé d'écrire un programme qui, étant donné un *échantillon* P de données numériques $(x, y) \in \mathbb{R}^2$, permette de répondre à des requêtes SQL de base selon des critères numériques.

Illustrons par un exemple. Prenons un *échantillon* P d'étudiants de la Faculté des sciences ayant chacun les critères suivants : C_1 : *year* l'année de l'étudiant et C_2 : *courses* le nombre de cours suivis. Alors, votre programme doit être capable de traiter la requête suivante

```
SELECT year FROM P WHERE courses >= 5
```

et de renvoyer la liste des années dans lesquelles se trouve au moins un étudiant ayant 5 cours ou plus. Une requête peut aussi être de la forme

```
SELECT year, courses FROM P WHERE courses >= 5 AND year in [2,3]
```

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.

auquel cas le programme doit renvoyer la liste de toutes les paires "année, nombre de cours" correspondant à un étudiant de deuxième ou troisième année ayant 5 cours ou plus.

Pour ce faire, il est demandé d'utiliser la structure de données *Kd-tree* pour stocker l'échantillon P.

De manière générale, une requête sera toujours dans l'une des formes suivantes

```
SELECT critère1 FROM P WHERE condition1
SELECT critère1 FROM P WHERE condition1 AND condition2
SELECT critère1, critère2 FROM P WHERE condition1
SELECT critère1, critère2 FROM P WHERE condition1 AND condition2
```

où une condition dépend d'un critère et est toujours dans l'une des formes suivante, pour $\mathtt{x}, \mathtt{y} \in \mathbb{R}$

```
critère >= x
critère <= x
critère in [x,y]
```

3 Étapes de votre travail

3.1 Lecture et compréhension du sujet

La section 5.2 de la référence traite des *Kd-Trees*. Un *Kd-tree* T (à 2 dimensions) est un arbre binaire qui permet de stocker un ensemble de points $(x, y) \in \mathbb{R}^2$ d'une manière particulière. Cette structure permet d'obtenir efficacement l'ensemble des points de P étant contenus dans une région donnée du plan. La section 5.5 allège quant à elle les hypothèses sur l'ensemble des points.

Votre travail commence par la lecture de la référence et la compréhension des sections traitant le sujet, en particulier :

— la structure de données *Kd-tree*,
— l'algorithme de construction d'un *Kd-tree* T à partir d'un ensemble P de points donnés,
— l'algorithme de recherche des points de P qui sont contenus dans une région R donnée,
— l'intérêt de la section 5.5 de la référence et comment l'appliquer dans l'implémentation.

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

— Comprenez-vous d'où vient la complexité annoncée de l'algorithme BuildKdTree ?
— Comprenez-vous l'ordre des données pour cet algorithme ? Avez-vous un exemple d'échantillon "dégénéré" que la section 5.5 permet de traiter ?
— Voyez-vous ce que représente la région d'un nœud de l'arbre et à quoi elle sert dans l'algorithme SearchKdTree ?
— Et lors d'un parcours de l'arbre, comment évoluent les régions des noeuds visités ?
— Voyez-vous comment adapter une requête à un *Kd-tree* qui ne traite qu'une seule condition sur un seul critère ?
— La référence présente les requêtes d'un critère comme des intervalles $[x : x']$, mais voyez-vous comment adapter cette approche dans le code pour avoir une requête du type "$\leq x$" ou "$\geq x$" ?

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) de la structure de donnée *Kd-tree* ;
- une explication à haut niveau de la construction d'un *Kd-tree* à 2 dimensions ainsi que la complexité attendue pour celle-ci ;
- une intuition concernant une requête à un *Kd-tree* à 2 dimensions ;
- une première intuition quant à votre gestion des cas particuliers de points $p, p' \in P$ ayant la même abscisse ou la même ordonnée ;
- 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*[3]). 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`.

Votre programme maintient un échantillon P en mémoire (stocké dans un *Kd-tree* T à deux dimensions). La structure T est au départ vide mais pourra être construite de deux manières :

- en lisant un fichier (texte) contenant les informations d'un tel échantillon ;
- ou en entrant interactivement des éléments dans l'échantillon.

Il n'est pas nécessaire d'avoir une interface graphique mais essayez de rendre votre programme facile à utiliser (pensez aux arguments en ligne de commande par exemple).

3.2.3 Format de fichiers

Pour pouvoir sauver et charger un échantillon P à partir de votre programme, vous utiliserez un format de *fichier texte* listant les critères, puis listant tous les éléments $p \in P$ de l'échantillon.

Vous ne serez pas nécessairement limités à deux critères. En pratique, on aimerait aussi stocker des chaînes de caractères, comme le nom des étudiants ou le nom de leur département. Ainsi, vous pourrez avoir plus de deux critères dans un échantillon, mais seuls les deux premiers seront numériques et utilisés par le *Kd-tree*, ceux qui seront être utilisés dans les conditions des requêtes pour rechercher des données.

Le format de fichier texte est ainsi le suivant

```
Nombre M de critères (entier)
Nom du critère 1 (chaîne de caractères)
...
Nom du critère M (chaîne de caractères)
Nombre N d'éléments dans l'échantillon (entier)
Élément 1: critère1 (nombre) critère2 (nombre)
           critère3 (chaîne de caractères) ... critèreM (chaîne de caractères)
...
Élément N: critère1 (nombre) critère2 (nombre)
           critère3 (chaîne de caractères) ... critèreM (chaîne de caractères)
```

Exemple de fichier :

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

```
3
year
height
department
6
1  175.20  mathématique
3  182.00  informatique
2  167.45  informatique
4  176.33  physique
4  181.13  biologie
1  164.63  chimie
```

Nous pourrions alors faire la requête

```
SELECT department FROM P WHERE year = 3 AND height in [170,180]
```

qui nous liste tous les départements où un étudiant en troisième année mesure entre 170 et 180cm. Ici, les deux premiers critères (année et taille) sont les critères utilisés pour les conditions. 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 un échantillon P, c'est-à-dire lire un fichier au format adéquat et stocker les points de cet échantillon dans T, un *Kd-tree* à 2 dimensions.

b) sauver un échantillon en créant un fichier au format décrit plus haut à partir de T ;

c) ajouter un point à la base de donnée pour ensuite recréer T ;

d) afficher les valeurs de T dans la console. Il est plus facile d'afficher un arbre binaire en mettant la racine à gauche plutôt qu'en haut (comme s'il avait subi une rotation de 90 degrés vers la gauche). Par exemple, l'arbre

```
   1
  / \
 2   3
```

peut être affiché comme suit :

```
    3                    ---3
   /                     |
 1                   OU  1
   \                     |
    2                    ---2
```

En outre, l'utilisateur doit être capable d'obtenir des réponses aux questions suivantes sur l'échantillon en utilisant un *Kd-tree* :

f) quelle est la valeur minimale ou maximale pour le critère C_j, où $j = 1$ ou 2 ?

g) étant donné une requête comme décrite dans les sections 2 et 3.2.3, quelle est la liste des critères d'éléments de l'échantillon satisfaisant les conditions de la requête.

3.3 Généralisation à 3 dimensions

Il vous sera également demandé de lire la fin de la section 5.2 de la référence, décrivant la manière dont on peut généraliser un *Kd-tree* à trois dimensions.

Vous ne devez pas implémenter les algorithmes pour les *Kd-trees* à 3 dimensions, mais seulement comprendre la manière dont on peut en construire un. Voici un exemple avec trois critères numériques (année, taille et le nombre de cours suivis) ainsi qu'un quatrième critère purement utilitaire contenant le nom de l'étudiant.

```
4
year
height
courses
name
15
3  161.50  7    Alexandra
5  175.00  11   Bob
2  163.75  9    Charlotte
4  178.20  13   David
1  168.90  6    Emma
3  172.30  8    Frank
4  166.70  10   Grace
2  183.40  12   Henry
5  162.80  5    Irene
1  170.20  13   Jack
4  172.60  8    Kate
3  175.10  9    Liam
2  165.00  10   Mathieu
5  180.90  4    Noah
1  168.30  10   Olivia
```

Il vous est demandé d'inclure dans votre rapport la construction et l'explication de la construction d'un *Kd-tree* à trois dimensions sur cet exemple.

3.4 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 explication sur la complexité de chacun de vos algorithmes *importants* ;
- l'explication de la construction d'un *Kd-tree* à 3 dimensions et les différences avec 2 dimensions, ainsi qu'un exemple "à la main" de son fonctionnement sur base de l'exemple de la section 3.3 (à partir des critères numériques *year*, *height* et *courses*) ;
- un diagramme de classes de votre implémentation ;
- un mini guide utilisateur expliquant comment compiler, exécuter et utiliser votre programme ;
- 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 20 décembre 2024 à 23h59 : 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.

— **dimanche 20 avril 2025 à 23h59 : 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 (fichier `build.xml`, `Makefile`, ...).

b) *Rapport final*
Vous déposerez via la plate-forme e-learning votre rapport final (conforme aux informations demandées dans la section 3.4), au format PDF.

— **Après la remise du projet : entretien pendant la session de juin 2024**
Une date et un horaire de passage pour une défense seront communiqués à chaque étudiant. Lors de cet entretien, vous présenterez brièvement la manière dont vous avez abordé le projet, et des questions seront ensuite posées.

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.

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 C. Grandmont (`christophe.grandmont@umons.ac.be`).

Bon travail !