IFT 2015 : Devoir 3

Professeure : Esma Aïmeur
Démonstrateurs : Dorsaf Sallami et Hugo Rocha

Hiver 2024

1 Consignes générales

- Date de rendu : jeudi 28 mars à 23h59.
- Ce travail est à faire seul(e) ou à deux.
- Aucun plagiat ne sera accepté.
- Il est possible que des précisions ou des modifications de l'énoncé soient envoyées par courriel et mises sur Studium.
- Chaque jour de retard pour la remise de votre travail entraînera une pénalité de 5 points.
- Tout code est à faire avec Java JDK 17
- Vous devez soumettre les fichiers Java demandés et un unique PDF avec les réponses écrites.
- Le PDF peut être des photos/scans de réponses écrites mains. Attention à la lisibilité!
- **Si vous êtes en équipe :** l'un soumet les fichiers demandés et l'autre soumet uniquement un PDF avec le nom de son partenaire.
- L'évaluation du code peut être automatisée, assurez-vous d'avoir **les même signatures de fonctions**.
- Vous avez le droit de créer autant de fonctions intermédiaires que vous voulez.
- **N'oubliez pas d'ajouter des docstrings et des commentaires à vos implémentations. Suivez également les principes de CamelCase. Le code non documenté sera pénalisé.**
- Pour toute question sur le devoir 3, nous vous demandons de la publier sur le forum "TP3 Q&A" sur Studium.
- Bon travail!

Pour chaque exercice, créez un dossier pour les fichiers Java correspondants.

2 Arbres binaires de recherche (5 points)

Créer une nouvelle version de l'arbre binaire de recherche vue en classe (BinarySearchTree). Cette mise en œuvre devrait inclure les opérations standard d'un arbre binaire de recherche : size(), isEmpty(), insert(int key), getMax(), search(int key) and remove(int key).
Ajoutez la liste des fonctions suivante:

1. (1.5pt) Implémentez la fonction void updateBST(Node root) qui permet de modifier un arbre de recherche binaire, de telle sorte que la clé de chaque nœud soit mis à jour pour contenir la somme de toutes les clés supérieures à lui présentes dans le BST.

2. (2pts) Implémentez la fonction boolean areSameBST(int[] array1, int[] array2) qui vérifie si deux tableaux d'entiers, qui sont des séquences de clés pour un arbre binaire de recherche (BST), définissent le même BST sans nécessiter la construction de l'arbre lui-même. On part du principe que l'insertion des clés dans le BST suit l'ordre présenté dans les tableaux.

3. (1.5pt) Déterminez la complexité temporelle de chaque fonction.

Soumettre: Le fichier BinarySearchTree.java et un fichier Main.java avec une petite démonstration du fonctionnement de votre implémentation.

3 Réseau Social (10 points)

Un réseau social stocke des informations sur ses utilisateurs dans un arbre binaire de recherche. Chaque noeud de l'arbre contient un numéro d'utilisateur unique (la valeur clé), le nom de l'utilisateur et une liste des noms de chacun des amis de l'utilisateur.

1. (1.5pt) Implémentez la définition de la classe ReseauBST et son constructeur.

2. (1.5pt) Implémentez une fonction compte−amis qui prend comme input un arbre binaire de recherche et un numéro d'utilisateur et retourne le nombre d'amis associé à la personne ayant ce numéro. Votre fonction doit être conçue pour minimiser le nombre de comparaisons effectuées dans l'arbre.

3. (1.5pt) Implémentez une fonction ami−tout−le−monde. La fonction prend comme input un arbre binaire de recherche et le nom d'une personne et retourne l'arbre binaire de recherche de telle sorte que la personne ait été ajoutée à la liste d'amis de chaque utilisateur.

4. (1.5pt) Implémentez une fonction liste −noms−dans−ordre. La fonction prend comme input un arbre binaire de recherche et retourne une liste des noms d'utilisateurs dans l'arbre, de sorte que la liste des noms soit dans un ordre numérique ascendant par numéro d'utilisateur.

5. (2pts) Implémentez une fonction ajoute−nouvel−utilisateur. La fonction prend comme input un arbre binaire de recherche, un numéro d'utilisateur et un nom d'utilisateur, et ajoute un nouvel utilisateur avec les informations données à l'arbre binaire de recherche. La liste d'amis de l'utilisateur devrait être vide. Assurez-vous que l'arbre produit est un arbre binaire de recherche. Vous pouvez supposer que le numéro d'utilisateur n'existe pas déjà dans l'arbre donné. Les nouveaux enregistrements sont toujours ajoutés à l'extrémité (feuille) de l'arbre. Les enregistrements ne sont jamais insérés dans les couches intermédiaires d'un arbre de recherche binaire.

6. (2pts) Implémentez une fonction recommender qui prend en entrée un arbre binaire de recherche et un numéro d'utilisateur, et qui renvoie une liste de recommandations d'amis : les utilisateurs dans l'arbre binaire de recherche qui ne sont pas déjà dans la liste d'amis de cet utilisateur mais qui ont des amis communs.

Soumettre: Le fichier ReseauBST.java et un fichier Main.java avec une petite démonstration du fonctionnement de votre implémentation.

4 Monceaux (5 points)

Contexte: Dans un jeu de société, au début de chaque tour, les participants reçoivent des cartes qui fixent la séquence de leur passage. Ces cartes possèdent divers niveaux de priorité.

Tous les joueurs dévoilent simultanément leurs cartes. La séquence d'action pour le tour est établie selon la priorité des cartes exposées. Dans cet ordre, les joueurs procèdent à une action spécifique : lancer un dé. Ce dé n'est pas ordinaire, car il peut afficher des valeurs négatives (2, 1, 0, 0, -1, -2). Le résultat du lancer est additionné au total de points du joueur.

La partie prend fin après un nombre déterminé de manches ou dès qu'un joueur parvient à un score cible spécifique (par exemple, 10 points).

Pour organiser la séquence des tours, nous utiliserons un monceau Jeu.

1. (1pt) Implémentez la définition de la classe Jeu et son constructeur de manière à répondre aux exigences du jeu, en tenant compte que la priorité d'un nœud correspond à la priorité la plus élevée parmi la liste des priorités du participant.
2. (1pt) Implémentez la fonction ajout qui permet d'ajouter un joueur.
3. (2pts) Implémentez également la fonction tour qui ajuste le tas après chaque tour (tous les participants ont joué).
4. (1pt) Lors d'un tournoi plusieurs jeux qui doivent être combinés. Implémenter la fonction fusion permettrait de fusionner deux monceaux Jeu en un seul.

Soumettre: Le fichier Jeu.java et un fichier Main.java avec une petite démonstration du fonctionnement de votre implémentation.