INF2010 - Structures de données et algorithmes

Automne 2021 Travail Pratique 2

TP3 : Arbres

Objectifs du laboratoire :

1. Apprendre le fonctionnement d'un arbre binaire de recherche
2. Comprendre la complexité asymptotique d'un arbre binaire de recherche
3. Utiliser les concepts associés aux arbres binaires dans un problème complexe

Ce laboratoire aura pour objectif principal de vous familiariser avec les arbres binaires de recherche et de leurs complexités algorithmiques. Votre travail est de compléter les TODO's manquants dans le code et de passer tous les tests unitaires. Le laboratoire sera en majorité évalué en fonction de tests passés (veuillez vous référer à la section « barème de correction » située à la fin de ce document).

Format de la remise :

Remettre dans un folder nommé : tp3_MatriculeX_MatriculeY.zip avec tous les fichiers de laboratoire.

Important : Les travaux mal nommés ou avec des fichiers supplémentaires auront une pénalité de 20% sur le travail. Les travaux en retard seront pénalisés de 20% par jour de retard.

Révision : Arbre binaire de recherche

Un arbre binaire de recherche est un arbre qui respecte les deux propriétés suivantes :

- La valeur du fils gauche d'un nœud donné est inférieure ou égale à la valeur du nœud courant.
- La valeur du fils droit d'un nœud donné est supérieure à la valeur du nœud courant.

Les arbres binaires de recherche sont souvent composés de clés uniques, mais pour ce problème, nous acceptons les doublons. Comme mentionné, les doublons sont placés aux nœuds gauches.

Travail à effectuer :

Partie 1 : Implémentation d'un arbre binaire de recherche

Laissez-vous guider par les « TODO » et les commentaires dans les fichiers BinaryNode.java et BinarySearchTree.java. Vos algorithmes seront testés sur la complexité du temps, ainsi ils doivent avoir une complexité du temps optimale pour chaque méthode, indiquée dans le « TODO » de chaque méthode.

BinaryNode :

- Data : represente la donnée du problème, elle doit implémenter l'interface Comparable pour pouvoir placer les items aux bons endroits.
- Insert : on place le nouvel item à gauche s'il est plus petit ou égal à l'item courant, sinon à droite.
- Contains : on veut savoir si l'item est égal à l'item courant ou à un de ses enfants
- GetHeight : on va savoir la hauteur d'un certain nœud de bas en haut. La racine seule a 0 de hauteur par défaut.
- FillListInOrder : on remplit une liste déjà créée en rajoutant les éléments en ordre « InOrder » donc : gauche, centre et droite.

BinarySearchTree :

- Insert : l'insertion se fait à partir de la racine de l'arbre.
- Contains : est-ce que l'arbre contient l'élément.
- GetHeight : quelle est la hauteur à partir de la racine.
- GetItemsInOrder : passe une liste à la racine de l'arbre qui ensuite remplirait l'arbre en ordre logique.
- ToStringInOrder : retourne le string des items de l'arbre en ordre logique sous le format : « [1, 2, 3] », on note les [], les virgules et les espaces.

Note : *les arbres binaires pourraient avoir les items positifs et négatifs*

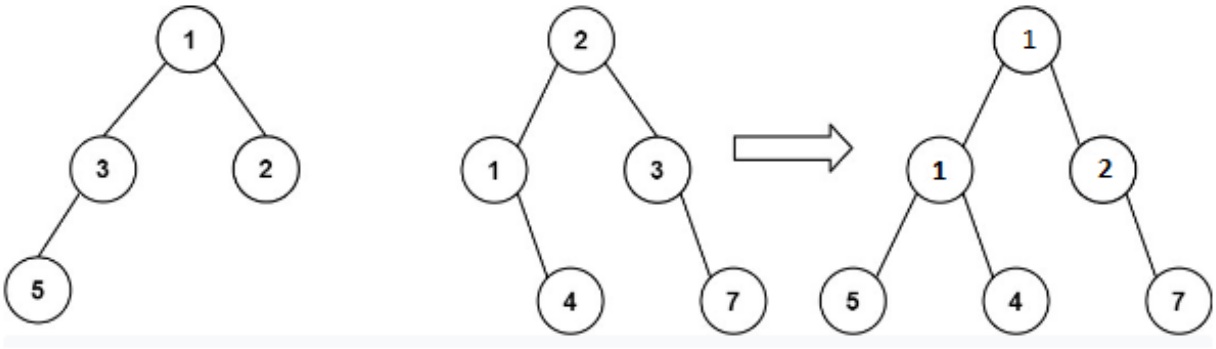
Partie 2 : Fusion de deux arbres binaires

Vous êtes donnés deux arbres binaires : **root1** et **root2**

Imaginez que vous avez une situation ou vous avez deux arbres binaires et vous avez le besoin de les fusionner. Lors du fusionnement, vous avez certains nœuds qui se chevauchent et d'autres qui ne se chevauchent pas. Voici les deux règles que vous devez suivre :

- Si les nœuds se chevauchent : la somme du nouveau nœud est le minimum des deux nœuds précédents.
- Si les nœuds ne se chevauchent pas : le nouveau nœud aura pour valeur le seul nœud qui existe.

En voici un exemple :



En entrée vous recevez les deux arbres qui ont les nœuds de valeurs :

- Root1 : [1, 3, 2, 5]
- Root2 : [2, 1, 3, null, 4, null, 7]

La sortie qui en suit est l'arbre : [1, 1, 2, 5, 4, null, 7].

Il est à noter que la racine des deux arbres est le minimum des deux racines min (1, 2) = 1 de même que tous les nœuds qui se chevauchent. Les nœuds qui ne se chevauchent pas t.q. le nœud de valeur 5 de l'arbre Root1 est gardé dans l'arbre final comme tel.

Consignes : complétez les « TODO's » manquants.

Dates de remise :

Vous avez deux semaines pour compléter le laboratoire. Ainsi les dates de remise sont :

- Pour les groupes 2 et 4 : **31 octobre à 23h59**
- Pour les groupes 1 et 3 : **7 novembre à 23h59**

Barème de correction :

Tests Partie 1 : /5

Complexité du temps Partie 1: /4

Qualité du code Partie 1: /1

- Absence de warnings à la compilation
- Absence de code mort et de chiffres magiques
- Le code respecte les mêmes conventions dans tout le projet
- Variables expliquent l'intention
- La structure du code est fluide

Tests Partie 2 : /5

Complexité du temps Partie 2: /4

Qualité du code Partie 2: /1

Note : /20