Algorithmique avancée - TD6

—o000o——o000o—

Introduction aux arbres binaires

1 Génération d'un arbre aléatoire

On s'intéresse tout d'abord à la création aléatoire d'un arbre binaire de type `ABInt` contenant des valeurs entières qui étend `AB<Integer>`.

1. Écrire une méthode `addRand(int val)` qui permet d'ajouter la valeur `val` passée en argument en suivant aléatoirement les branches de l'arbre binaire. Par construction du modèle objet, l'arbre binaire ne peut être vide. En partant du nœud initial, un tirage alétoire doit être réalisé pour savoir si la valeur doit être ajoutée au sous-arbre gauche ou au sous-arbre droit. L'algorithme doit vérifier si les sous-arbres existent avant de les modifier en y ajoutant récursivement la valeur.

2. À partir de la méthode précédente, proposer une méthode `addAlea(int size, int valMax)` qui ajoute aléatoirement `size` valeurs générées aléatoirement entre 0 et $valMax - 1$ à l'arbre binaire.

3. Écrire enfin une méthode `addOrder(int val)` qui cette fois n'ajoute pas la valeur en choisissant aléatoirement entre l'arbre gauche et l'arbre droit, mais en suivant les règles suivantes :

   - si la valeur est égale à l'étiquette de l'arbre, on ne l'insère pas et on laisse l'arbre tel qu'il est
   - si la valeur est strictement inférieure à l'étiquette de l'arbre, on l'insère dans le sous-arbre gauche
   - sinon, on l'insère dans le sous-arbre droit

2 Plus court chemin dans un arbre binaire

On souhaite compléter la classe précédente en calculant le chemin le plus court entre la racine de l'arbre et une feuille.

1. Écrire la méthode `shortestPath()` qui retourne la longueur du chemin le plus court entre la racine de l'arbre et une feuille.

2. Écrire la méthode `shortestPathNodes()` qui retourne la liste chaînée des valeurs contenues dans le chemin le plus court entre la racine de l'arbre et une feuille. La liste chaînées prendra la forme d'un nœud de liste de type `Node<Integer>`. Si le chemin gauche et le chemin droit font la même longueur, on préférera le chemin gauche.

3 Liste infixe, préfixe et suffixe

Le parcours d'un arbre binaire peut donner lieu à 3 listes de valeurs différentes selon l'ordre dans lequel on considère la racine, l'arbre gauche et l'arbre droit.

1. À partir de votre cours, et sans regarder les implémentations Java proposées, écrire les méthodes `infixe()`, `prefixe(A)` et `postfixe(A)` qui s'appliquent sur tout arbre binaire et retournent une liste chaînée représentée par une structure de type `Node<T>`.

4 Dénombrement à un certain niveau

1. On s'intéresse tout d'abord à l'écriture d'une méthode pour déterminer le nombre de nœuds à un niveau k (entier positif) dans un arbre binaire B. On considérera que la racine d'un arbre binaire non vide est au niveau 1 et que de manière général, le niveau d'un nœud est égal à celui de son père augmenté de 1.

2. Écrire maintenant une méthode qui permet de déterminer le nombre de feuilles à un niveau donné k (entier positif) en suivant les mêmes contraintes que précédemment.

5 Égalité entre deux arbres binaires

On s'intéresse maintenant à la définition d'une méthode pour déterminer si deux arbres binaires sont égaux. On considère ici que 2 arbres binaires sont égaux s'ils possèdent les mêmes valeurs d'étiquettes et la même structure : si l'un possède (resp. ne possède pas) un fil (gauche ou droit) l'autre doit (resp. ne doit pas) en avoir un. On pose que deux feuilles sont égales si elles possèdent la même étiquette.

6 Miroir d'un arbre binaire

L'arbre miroir d'un arbre binaire B est défini récursivement de la manière suivante :

- il possède la même racine que l'arbre binaire B,
- son fil gauche est l'arbre miroir du fil droit de B,
- son fil droit est l'arbre miroir du fil gauche de B.

Le miroir d'une feuille est elle-même et on ne gère pas le cas des arbres vides.

1. Écrire une méthode qui prend en argument un arbre binaire et retourne son arbre miroir. Vous proposerez différents tests pour vérifier votre méthode.

2. Écrire une méthode prenant en argument 2 arbres binaires quelconques A et B et retourne `true` si et seulement si B est l'arbre miroir de A et `false` sinon.

3. Que vaut l'arbre miroir de l'arbre miroir d'un arbre binaire ?