Aix*Marseille université
Initiative d'excellence
Année 2019-20

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 5 : TP 1

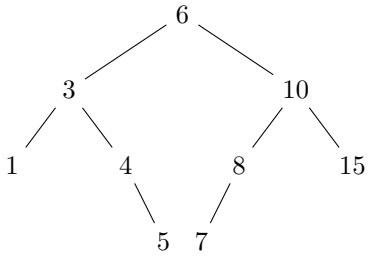
Arbres binaires de recherche

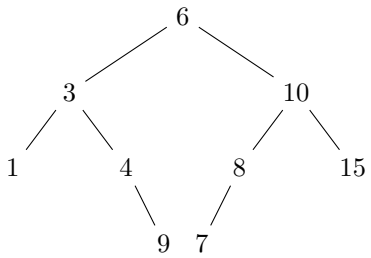
1 Implémentation des arbres binaires de recherche

Repartons de la classe `BinaryTree` d'arbres binaires, dont une correction vous est redonnée sur Ametice. Dans un nouveau fichier, définir une classe `BinarySearchTree` qui hérite de la classe `BinaryTree` : en effet, un arbre binaire de recherche est un arbre binaire particulier dont on a envie de pouvoir utiliser les différentes méthodes (test de feuille, mise en forme d'une chaîne de caractères, calcul de la hauteur et des différents parcours). Cependant, tout arbre binaire n'est pas un arbre binaire de recherche, et on n'aimerait pas stocker dans la classe `BinarySearchTree` un arbre binaire qui ne soit pas un arbre binaire de recherche.

Question 1 Écrire un constructeur permettant de vérifier au moment de sa création que l'arbre est bien un arbre binaire de recherche. On rappelle qu'on peut utiliser l'appel à `instance(x, A)` pour vérifier que l'objet `x` est une instance de la classe `A` ou non. De plus, il n'est pas interdit (et même franchement conseillé) d'ajouter des attributs à la classe `BinarySearchTree` (par contre, on n'a pas le droit de toucher à la classe `BinaryTree` !) pour se simplifier la tâche et éviter d'avoir à *plonger* dans les sous-arbres gauches et droits...

Question 2 Vérifier que ce constructeur permet de créer l'arbre de gauche mais pas l'arbre de droite (pourquoi n'est-il pas légal ?) :





Question 3 Appliquer (à la main d'abord, puis vérifier votre réponse à l'aide de la méthode idoine) un parcours infixe (sans parenthèse) à l'arbre binaire de recherche de gauche. Quelle propriété satisfait-il ? Est-ce toujours le cas ?

Question 4 Proposer une méthode `contains` permettant de tester si l'arbre contient un élément donné en argument.

Question 5 Proposer une méthode `insert` qui insère dans l'arbre un élément donné en argument (s'il ne s'y trouve pas déjà, sinon la méthode ne modifie pas l'arbre).

Question 6 Construire (à la main d'abord, puis vérifier votre réponse à l'aide de la méthode précédente) un arbre binaire de recherche en insérant successivement, à partir de l'arbre vide les entiers de 1 à 7.

Question 7 Dans quel ordre insérer les éléments de 1 à 7 pour obtenir un arbre de hauteur minimale ? Plusieurs ordres sont-ils possibles ?

Question 8 Supprimer (à la main) l'élément 3 de l'arbre binaire de recherche à gauche ci-dessus, puis l'élément 1 et enfin l'élément 6.

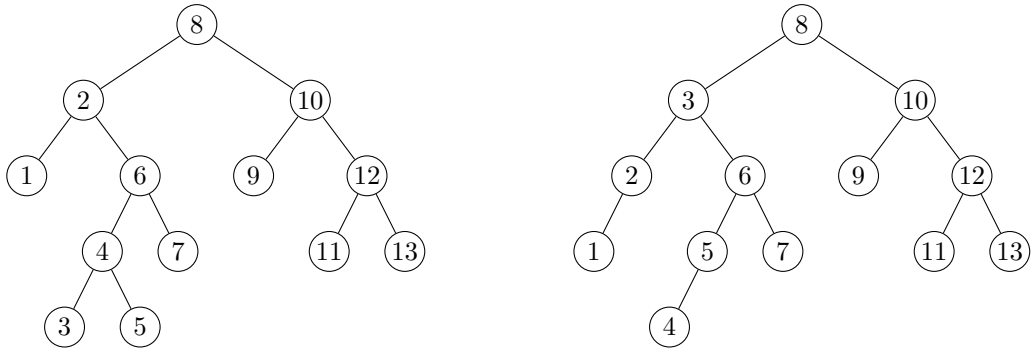
Question 9 Proposer une méthode `suppress` qui supprime de l'arbre un élément donné en argument (s'il s'y trouve, sinon la méthode ne modifie pas l'arbre). On pourra utiliser des méthodes auxiliaires si besoin.

Question 10 D'autres opérations peuvent être utiles sur les arbres binaires de recherche. Par exemple, proposer une méthode `successor` qui renvoie le plus petit élément contenu dans l'arbre qui est strictement supérieur à un élément donné en argument. Par exemple, sur l'arbre de gauche ci-dessus, le successeur de 3 est 4 et le successeur de 5 est 6.

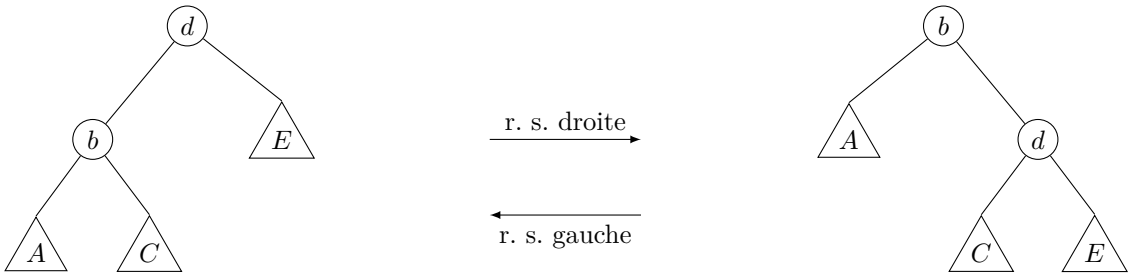
2 Arbres AVL

On poursuit l'étude des arbres binaires de recherche par l'étude des arbres AVL qui permettent de garantir une hauteur logarithmique en le nombre de nœuds dans l'arbre. Un arbre AVL est un ABR tel que chaque nœud x vérifie que la hauteur h_g du sous-arbre gauche de x et la hauteur h_d du sous-arbre droit de x vérifient $|h_g - h_d| \leq 1$.

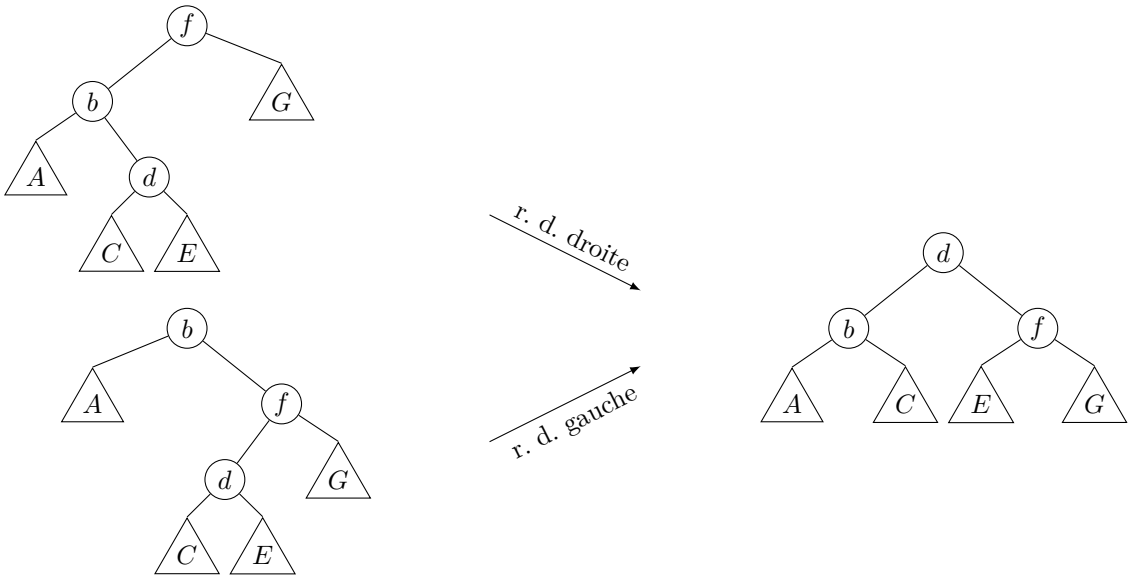
Question 11 Les arbres ci-dessous sont-ils des arbres AVL ?



Au moment de la construction, ou de la destruction, d'un arbre AVL, on doit maintenir la contrainte sur les hauteurs. Pour ce faire, on utilise des *rotations* de deux types : des rotations simples à gauche ou à droite



et des rotations doubles à gauche ou à droite



Pour faire simple (en vrai, il y a évidemment un algorithme précis, que vous pouvez trouver dans les livres

d'algorithmique), lorsqu'on a un déséquilibre au niveau d'un nœud, à gauche par exemple (voulant dire que le sous-arbre gauche a une hauteur supérieure de deux unités à celle du sous-arbre droit), alors on essaie d'appliquer une rotation simple à droite si c'est possible, et sinon on applique une rotation double droite. Une fois que c'est fait, on vérifie s'il reste des déséquilibres puisque cela peut en avoir créé d'autres plus haut !

Question 12 Construire un arbre AVL en partant de l'arbre vide et en ajoutant successivement les éléments 51, 44, 32 , 36, 17, 59, 23, 21, et enfin 24.

Question 13 Supprimer l'élément 3 de l'arbre AVL à droite ci-dessus.

Question 14 Proposer une méthode de la classe `BinarySearchTree` permettant de vérifier si un arbre binaire de recherche donné est un arbre AVL. On pourra ajouter des attributs et des méthodes.