# Arbres binaires

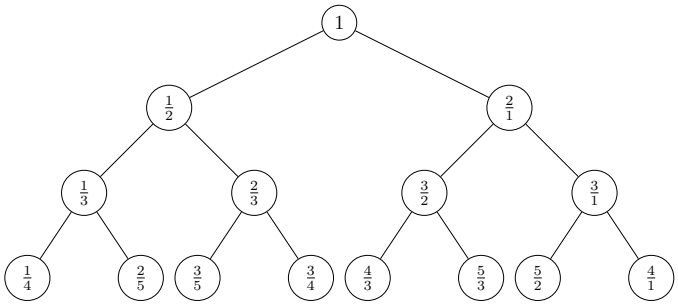
Étienne Miquey
etienne.miquey@ens-lyon.fr

Le but de ce TP est principalement de vous faire manipuler une structure de données arborescente. Pour cela, nous allons travailler dans un premier temps avec des arbres binaires de recherches usuels avant de s'intéresser dans un second temps aux AVL, une version équilibrée. L'un des objectifs étant de voir à quel point vous pouvez être autonomes pour créer des types/objets, je ne vous donnerai rien d'autre que des dessins, à vous de coder de la manière qui vous semble la plus naturelle. Néanmoins, n'hésitez pas à me solliciter en cas de doute !

## 1 Arbre binaire de recherche

Comme vous le savez, un arbre binaire est un arbre où tout noeud a au plus deux fils, et où si $\mathcal{A}_g$ et $\mathcal{A}_d$ sont les fils gauches et droits d'un noeud étiquetté par $n$, tout noeud de $\mathcal{A}_g$ (resp. $\mathcal{A}_d$) est étiquetté par une valeur inférieure (resp. supérieure) à $n$.

**Question 1.** Définir un type `arbre`, et représenter un exemple de votre choix avec, par exemple celui-ci :




**Question 2.** Écrire une fonction de recherche d'un élément au sein d'un ABR. Complexité ?

**Question 3.** Écrire une fonction d'insertion d'un élément dans un ABR. Complexité ?

**Question 4.** Écrire de même une fonction de suppression d'un élément au sein d'un ABR. Complexité ?

**Question 5.** Écrire une fonction de parcours préfixe d'un ABR, qui prendra en argument une fonction de type `int` $\rightarrow$ `unit`. En déduire un algorithme de tri d'une liste par un ABR.

Jusqu'ici, tout va bien [1]. Sauf que l'on s'aperçoit vite que si l'on crée un arbre en insérant les éléments dans l'ordre où ils nous arrivent, on a tôt fait de créer des arbres déséquilibrés. Pour remédier à cela, nous allons donc étudier les AVL, un modèle d'ABR équilibré.

## 2 Adelson-Velsky/Landis

### 2.1 La base

On définit naturellement la notion de *hauteur* d'un arbre comme la profondeur maximale d'un chemin de la racine vers une feuilles. On appelle *taille* d'un arbre le nombre de noeuds le composant. On définit pour tout noeud $n$ ayant pour fils $\mathcal{A}_g$ et $\mathcal{A}_d$ son facteur d'équilibrage $\delta(n) = h(\mathcal{A}_d) - h(\mathcal{A}_g)$. Un noeud $n$ est dit équilibré si $\delta(n) \in \{-1, 0, 1\}$, et un AVL est un arbre binaire de recherche qui a tous ses noeuds équilibrés.

**Question 6.** Pour une hauteur $h$ donnée, quelle est la taille minimale d'un AVL ?

**Question 7.** Créer un type de données adapté (à savoir avec en plus de l'étiquette une information sur la hauteur), et écrire une fonction renvoyant la hauteur d'un arbre.

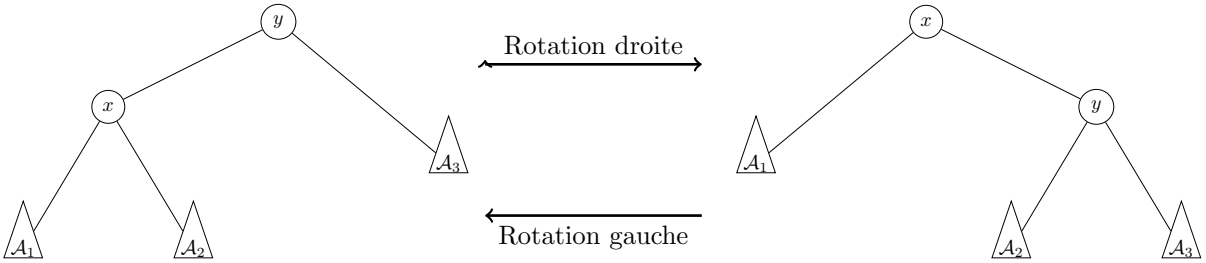
**Question 8.** Écrire une fonction qui, étant donné un nouveau sommet et deux sous-abres, crée un nouvel l'arbre correspondant.

**Question 9.** Que se passe-t-il si l'on effectue une insertion (ou une suppression) simple dans l'AVL ?

1. *Référence ?*

## 2.2 Rotations

On définit deux nouvelles opération sur les arbres, les rotations (à gauche et à droite). Un petit dessin valant mieux qu'un long discours [2], je vous laisse observer :




Attention cependant, si après ces transformations on garde des arbres binaires de recherche, ceci ne préservent pas en revanche l'invariant de l'AVL.

**Question 10.** Écrire des fonctions réalisant les rotations gauche et droite.

Pour rééquilibrer un arbre, il ne suffit pas forcément toujours de faire une rotation.

**Question 11.** En supposant avoir créé un déséquilibre en insérant un élément (disons, par symétrie, dans le fils gauche), dessinez les différents cas possibles, et cherchez au besoin à corriger avec une double rotation.

**Question 12.** À l'aide de la question précédente, écrire une fonction de rééquilibrage post-insertion et en déduire une fonction d'insertion dans un AVL qui conserve donc l'équilibre.

Faites des tests sur quelques petits exemples afin de vous assurer de la correction de votre algorithme.

**Question 13.** Programmenr une fonction qui vérifie si un arbre est bien un AVL. Générer ensuite une grande suite d'entiers aléatoires, et créer l'AVL correspondant en vérifiant à chaque étape sa validité.

## 2.3 Bonus trackz

S'il vous reste du temps en trop, vous pouvez au choix vous penchez sur l'une des questions suivantes :

**Question 14.** Implémenter une fonction de tri d'une liste à l'aide d'un AVL.

**Question 15.** Implémenter la suppression d'un élément dans un AVL tout en maintenant l'équilibrage.

**Question 16.** Écrire une (jolie) fonction d'impression des arbres.

2. *Référence ?*