TP2: Structures de données arborescentes

Le but de ce TP est d'implanter les algorithmes de recherche et de modification dans les ABR et tas vus en cours.

Partie 1 : Arbres binaires de recherche

Consignes
Télécharger le fichier TP2Exo1ABR.py. Le fichier contient deux classes, une pour gérer nœuds et une autre pour les arbres binaires :

- la classe `ArbreBinaire` contient un attribut `rac`, qui est un pointeur vers un `Noeud` ;
- la structure `Noeud` contient quatre attributs : trois pointeurs vers les fils et le père, et une valeur entière ;
- Les constructeurs permettent de construire des objets de chacune de ces classes.
- diverses méthodes sont proposées pour chaque classe, dont `DessinArbre()` qui permet d'afficher un arbre binaire à l'aide de `mathplotlib`.

▷ Notez la définition de classes, attributs et méthodes en Python...

- Compilez le fichier, exécutez le et notez bien la construction de l'exemple `arb` proposé. Vous devez obtenir l'affichage suivant :

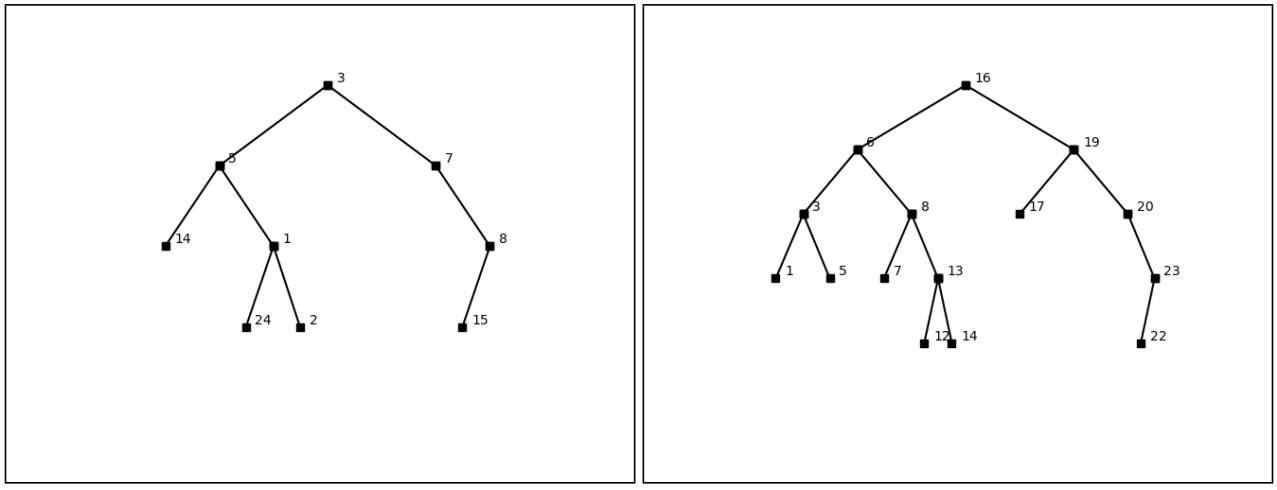

FIGURE 1 – Affichage de l'exemple d'arbre binaire `arb` et d'arbre binaire de recherche proposés.

- Bien entendu, il sera **impératif** de tester chaque fonction que vous écrirez !

Arbres binaires de recherche : à de vous de jouer !

1. Compléter la fonction `Maximum(x)` qui renvoie la valeur maximale d'un nœud dans la sous-arborescence de racine de `x` (on pourra supposer que tous les nœuds ont une valeur positive).
   Vous devriez trouver 24...
2. Ecrire la fonction `ParcoursInfixe(x)` qui affiche un parcours infixe de la sous-arborescence de racine `x`. On lancera le parcours par l'appel `ParcoursInfixe(Arb.rac)`.
   L'affichage de votre parcours devrait être : 14 5 24 1 2 3 7 15 8
   De même, programmer un algorithme de parcours préfixe et postfixe (voir la fiche de TD 3 si besoin).
   Attention ! Les questions suivantes traitent d'arbres binaires de recherche !
3. Compléter la fonction `Inserer(ABR,z)` qui insère le noeuds z dans l'arbre binaire de recherche *ABR*.
4. Compléter la fonction `CreerABR(ABR,L)` qui crée l'arbre binaire de recherche *ABR* dont les clés des nœuds sont données dans la liste L. Vous devriez obtenir la figure de droite précédente sur l'exemple fourni.

5. Compléter la fonction `RechercheCle(ABR,k)` qui cherche si un l'arbre binaire de recherche *ABR* contient un nœud de valeur k et renvoie un message précisant si un tel nœud est trouvé ou non.
6. *(Bonus)* Ecrire les fonctions `Successeur(ABR,z)` et `Supression(ABR,z)` qui respectivement donnent le successeur de z dans l'arbre binaire de recherche *ABR* et supprime le nœud z de l'arbre binaire de recherche *ABR*.

Partie 2 : Tas

Consignes
Télécharger le fichier TP2Exo2Tas.py. Le fichier contient les mêmes classes que le précédent, une pour gérer nœuds et une autre pour les arbres binaires. Ces classes seront utilisées pour gérer des tas ici. Les tas seront encodés et manipulés sous forme de tableaux et la fonction `DessineTasArbre` permet de dessiner un tas sous forme d'arbre, ce qui est plus pratique à visualiser.

▷ Notez le fait qu'on puisse stocker n'importe quel objet dans un tableau en Python, ici des nœuds. Remarquez l'opérateur `//` qui permet de faire une division entère.

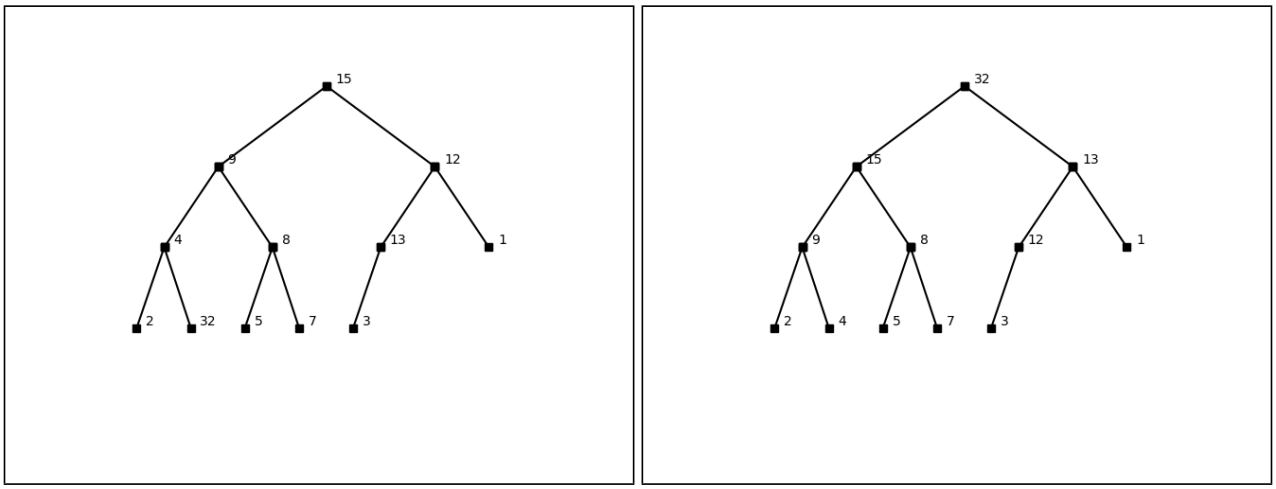

FIGURE 2 – Affichage du tableau donné en exemple sous forme d'arbre binaire quasi-complet et du même tableau transformé en tas.

Tas et tri par tas
L'objectif du premier exercice est d'implanter les fonctions sur les tas nécessaires au tri par tas, puis d'implanter celui-ci.

1. Compléter la fonction `TableauHasard(n)` qui retourne un tableau de n nombres choisis au hasard entre 1 et 99.
2. Compléter la fonction `void Remonter(L,i)` qui entasse le nœud i dans le tableau L.
3. Utiliser la fonction `Remonter` pour compléter la `CreerTas(L)` qui retourne un tas contenant les mêmes valeurs que le tableau L.
   Tester votre fonction `CreerTas` sur l'exemple fourni. Tester là aussi sur des tableaux aléatoires.
4. *(Bonus)* Écrire les fonctions `Entasser` et `Supprimer` données dans le cours.
5. (*(Bonus)* Écrire la fonction `TriParTas` permettant de trier un tableau à l'aide d'une structure de tas.