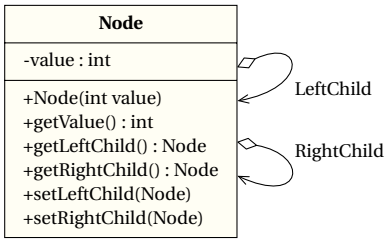
Arbres binaires de recherche.

1 Manipulation d'arbres binaires de recherche.

1.1 Manipulations simples

Au cours de ce TD, nous allons manipuler des *arbres binaires de recherche (ABR)*. Dans un arbre binaire de recherche, tous les nœuds du sous arbre gauche ont une étiquette inférieure à la racine, et tous ceux du sous-arbre droit ont une étiquette supérieure à la racine.
Pour écrire du code, on utilisera (par exemple) la structure de donnée décrite par le schéma ci-contre.



1. Dessinez l'ABR obtenu à partir d'un arbre vide en insérant successivement les valeurs :
   32, 7, 63, 97, 28, 18, 51, 56, 43, 65, 84, 36, 95, 14, 20, 76, 57, 6, 11, 17
2. Dessinez l'ABR obtenu en insérant les mêmes valeurs, mais dans l'ordre inverse.
3. Comment afficher les valeurs d'un ABR dans l'ordre croissant ?
4. Écrivez une fonction qui indique si un élément est présent dans un ABR.
5. Écrivez une fonction (similaire) qui insère un élément dans un ABR (s'il n'y est pas déjà).

1.2 Suppression d'éléments.

On souhaite maintenant retirer un élément de l'arbre. On identifie le nœud à supprimer. S'il s'agit d'une feuille, c'est facile. Sinon, il est à la racine d'un sous-arbre non vide. On analyse dans les sections suivantes deux approches pour traiter ce cas.

1.2.1 Approche naïve

Une première approche consiste à réinsérer chacun des nœuds du sous-arbre supprimé dans l'arbre, comme si c'étaient de nouvelles valeurs. Ce n'est pas très efficace, mais simple à implémenter.

6. Justifiez que toutes les valeurs réinsérées vont à leur tour former un sous-arbre à l'emplacement même où était le sous-arbre précédent.

1.2.2 Approche plus optimisée (*)

Une autre approche plus économique consiste à remplacer le nœud sans changer trop la structure de l'arbre. On se place dans le sous-arbre enraciné au niveau du nœud qui doit être retiré.

7. Où se trouvent les nœuds ayant les valeurs les plus proches de celle de la racine d'un ABR ? Vérifiez votre réponse sur les arbres dessinés plus haut.

Choisissons arbitrairement de prendre comme nouvelle racine la plus petite des valeurs plus grandes que la racine, que l'on appelle R'.

8. Que peut-on dire des sous-arbres gauche et droit de ce nœud ? (Vérifiez à nouveau votre affirmation)
9. Prenons d'abord l'exemple (simple) du retrait du nœud de valeur 7 dans l'arbre de la question 1. Quel changement apporte-t-on à l'arbre pour le remplacer ?
10. Prenons pour deuxième exemple la suppression du nœud de valeur 63 (toujours en le remplaçant par le plus petit des plus grands). Quelle forme a le nouvel arbre ?
11. Expliquez l'algorithme de suppression d'un élément, en détaillant les changements d'emplacement des sous-arbres.

2 Les tas

Les tas sont utilisés pour maintenir un ensemble d'éléments ordonnés dont on veut pouvoir extraire efficacement l'élément le plus petit. Un exemple typique d'utilisation est pour l'algorithme des plus courts chemins, de Dijkstra, que vous avez vu en mathématiques.

Le tas est un arbre binaire qui est stocké dans un tableau. La première case du tableau est la racine, les deux suivantes sont les enfants de la racine, les quatre suivantes les petits-enfants, etc. On rempli toujours les cases du tableau dans l'ordre naturel.

12. Décrivez la forme qu'a l'arbre d'un tas, à partir de la remarque ci-dessus.
13. Quelle est la profondeur d'un tas en fonction du numéro de la dernière case utile du tableau ?
14. À partir du numéro de la case correspondant à un nœud de l'arbre, quel est le numéro des cases de ses enfants ? Quel est le numéro de la case de son parent ?

La propriété principale d'un tas est que chaque nœud est inférieur à chacun de ses enfants. Ainsi, la racine d'un tas est nécessairement le nœud le plus petit de l'arbre.

15. Est-ce qu'un tableau trié correspond à un tas ? Si non, donnez un contre-exemple.
16. Est ce qu'un tas correspond à un tableau trié ? Si non, donnez un contre-exemple.

On s'intéresse à l'insertion d'un élément dans un tas. L'ensemble des cases utilisées du tableau, l'insertion se fait en commençant par placer la nouvelle valeur à la première case libre du tableau. Puis, pour rétablir la structure de tas, des échanges sont faits de proche en proche (un peu comme un tri à bulle, mais dans un arbre). Le but est de rétablir le fait que chaque nœud a une valeur inférieure à celles de ses enfants.

17. Décrivez l'algorithme d'insertion de valeur dans un tas.
18. Dessinez la forme du tas obtenu par l'insertion successive des valeurs
    32, 7, 63, 97, 28, 18, 51, 56, 43, 65, 84, 36, 95, 14, 20, 76, 57, 6, 11, 17.
19. Est-ce que la forme du tas dépend de l'ordre d'insertion des valeurs ? Est-ce que la position des valeurs dépend de l'ordre d'insertion des valeurs ?
20. Une autre opération classique du tas est l'extraction de la racine. Comment procéder pour maintenir la forme du tas ?
21. Déduire un algorithme de tri utilisant un tas. Quel est sa complexité en nombre d'insertion, d'extraction ?
22. Quelle est la complexité des opérations d'insertion et d'extraction dans un tas, en nombre d'opération unitaire dans le tableau ? Déduire la complexité de l'algorithme de tri vu précédemment.