EA4 – Éléments d'algorithmique
TD n° 9 : arbres binaires de recherche (suite)

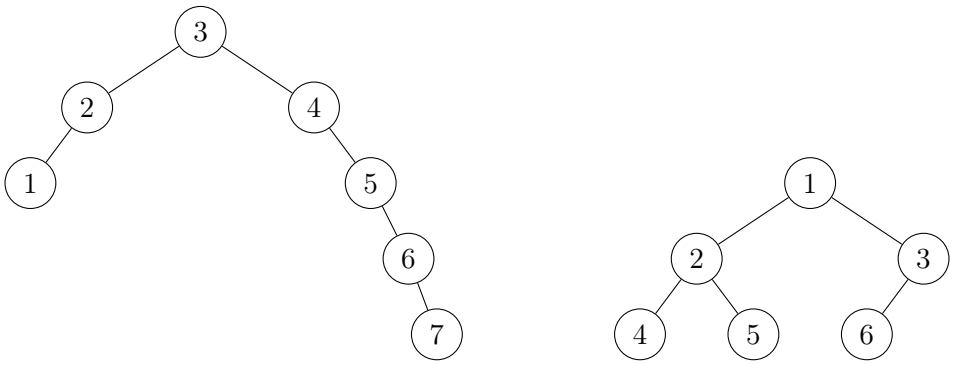
Exercice 1 : tri par ABR

Pour cet exercice, on supposera disposer d'une fonction `creerArbre()` renvoyant un arbre vide, `ajouterSommet(A,x)` qui ajoute le sommet d'étiquette `x` dans `A` et `infixe(A)` qui renvoie une liste représentant le parcours infixe de `A`. La complexité de `ajouterSommet(A,x)` est linéaire en la hauteur de l'arbre et la complexité de `infixe(A)` est linéaire en la taille de `A`.

1. Écrire un algorithme permettant de trier une liste en utilisant un ABR.
2. Quelle est sa complexité dans le pire cas ? dans le meilleur cas ? en moyenne ? Justifier.
3. Comparer cet algorithme à QuickSort (fonctionnement et complexités).

Exercice 2 : ABR presque-parfait et opérations ensemblistes

Un arbre binaire est dit *presque-parfait* si tous ses niveaux sont complètement remplis, excepté éventuellement le dernier. Par exemple, l'arbre (b) est presque-parfait, tandis que l'arbre (a) ne l'est pas.



(a) (b)

1. Donner un algorithme transformant une liste triée en ABR presque-parfait.
2. En déduire un algorithme qui transforme un ABR en ABR presque-parfait.
3. Quelle est sa complexité ?
4. En utilisant les questions précédentes, proposer un algorithme qui, étant donnés deux ABR (quelconques) A et B, renvoie un nouvel ABR presque-parfait représentant l'union des ensembles représentés par A et par B.
5. Que pensez-vous d'autres opérations ensemblistes binaires (intersection, différence, ...) ?

Exercice 3 : sélection efficace dans les ABR

On s'intéresse ici au problème `selection(E, k)`, consistant à déterminer le k^{e} plus petit élément d'un ensemble d'entiers représenté par la structure `E`.
(l'entier k est compris entre 1 et le cardinal de l'ensemble représenté par `E`)

1. Donner des algorithmes simples pour résoudre ce problème dans les cas suivants, ainsi que leurs complexités :
   – `E` est un tableau trié,

- `E` est un tableau non trié,
- `E` est un ABR.

2. Appliquer la sélection rapide aux données ci-dessous :

1. $([13, 16, 7, 10, 4, 8, 12, 3], 4)$.
2. $([8, 12, 16, 10, 4, 7, 3, 13], 2)$.

On souhaite enrichir la structure d'ABR pour effectuer la sélection de manière plus efficace (d'une manière analogue à QuickSelect). Pour cela, on stocke dans chaque nœud, en plus des données usuelles, la `taille` du sous-arbre correspondant (c'est-à-dire le nombre de nœuds du sous-arbre, y compris le nœud lui-même).

3. Réécrire l'algorithme d'insertion `insertionABR(A, x)` pour que le champ `taille` soit correctement tenu à jour. Que devient la complexité ?
(on supposera pour simplifier, et sans le vérifier, que tous les éléments sont distincts)

4. Si les tailles des deux sous-arbres d'un ABR `A` sont respectivement égales à g et d, où le k^{e} plus petit élément de `A` se trouve-t-il, en fonction de k ?

5. Décrire un algorithme, aussi efficace que possible, répondant au problème de la sélection pour ces ABR enrichis à l'aide du champ `taille`.

6. Analyser la complexité de cet algorithme, au pire et en moyenne.