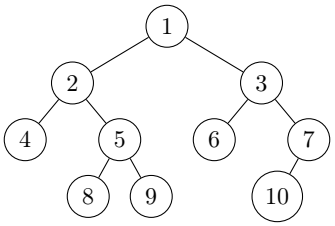
EA4 – Éléments d'algorithmique
TD n° 8 : arbres binaires de recherche
(Correction)

Exercice 1 : parcours d'arbres binaires

1. Vérifier que les sommets de l'arbre ci-dessous sont étiquetés dans l'ordre d'un parcours en largeur.

2. Lister les sommets de l'arbre binaire ci-dessous en ordres préfixe, infixe et suffixe.



Correction :

1. *Le resultat du parcours en largeur est : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.*

2. *Les algorithms à utiliser sont definies le transparent 24 dans ce PDF. Ici va rappeller le pseudo-code des algorithmes, ainsi que donner les resultats des parcours.*

- ***Parcours préfixe :***

```
VisiterPréfixe(Arbre A) :
  Si NonVide(A) :
    Afficher(A)
    VisiterPréfixe(gauche(A))
    VisiterPréfixe(droite(A))
```

Resultat de l'exécution : 1, 2, 4, 5, 8, 9, 3, 6, 7, 10.

- ***Parcours postfixe :***

```
VisiterPostfixe(Arbre A) :
  Si NonVide(A) :
    VisiterPostfixe(gauche(A))
    VisiterPostfixe(droite(A))
    Afficher(A)
```

Resultat de l'exécution : 4, 8, 9, 5, 2, 6, 10, 7, 3, 1

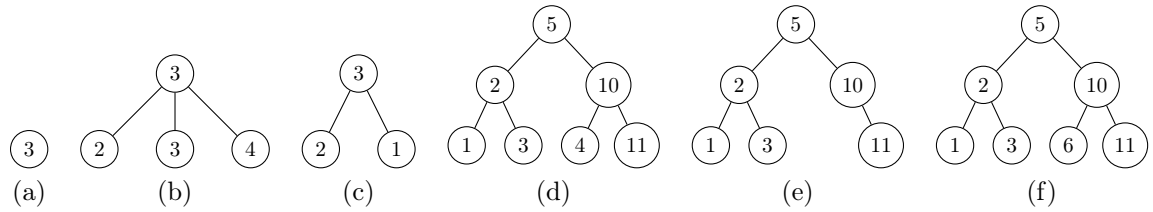
- ***Parcours infixe :***

```
VisiterInfixe(Arbre A) :
  Si NonVide(A) :
    VisiterInfixe(gauche(A))
    Afficher(A)
    VisiterInfixe(droite(A))
```

Resultat de l'exécution : 4, 2, 8, 5, 9, 1, 6, 3, 10, 7

Exercice 2 : arbres binaires de recherche

1. Parmi les arbres ci-dessous, lesquels sont des ABR ? (justifiez votre réponse).



2. Dessiner des ABR de toutes les hauteurs possibles pour l'ensemble de clés $\{1, 2, 3, 4, 5, 6, 7\}$.

3. Combien y a-t-il d'ABR d'une forme donnée pour un ensemble de n valeurs fixées ? Appuyez-vous sur un parcours d'arbre pour justifier votre réponse.

4. Donner un algorithme *de complexité linéaire en la taille de l'arbre* qui teste si un arbre binaire est un arbre binaire de recherche.

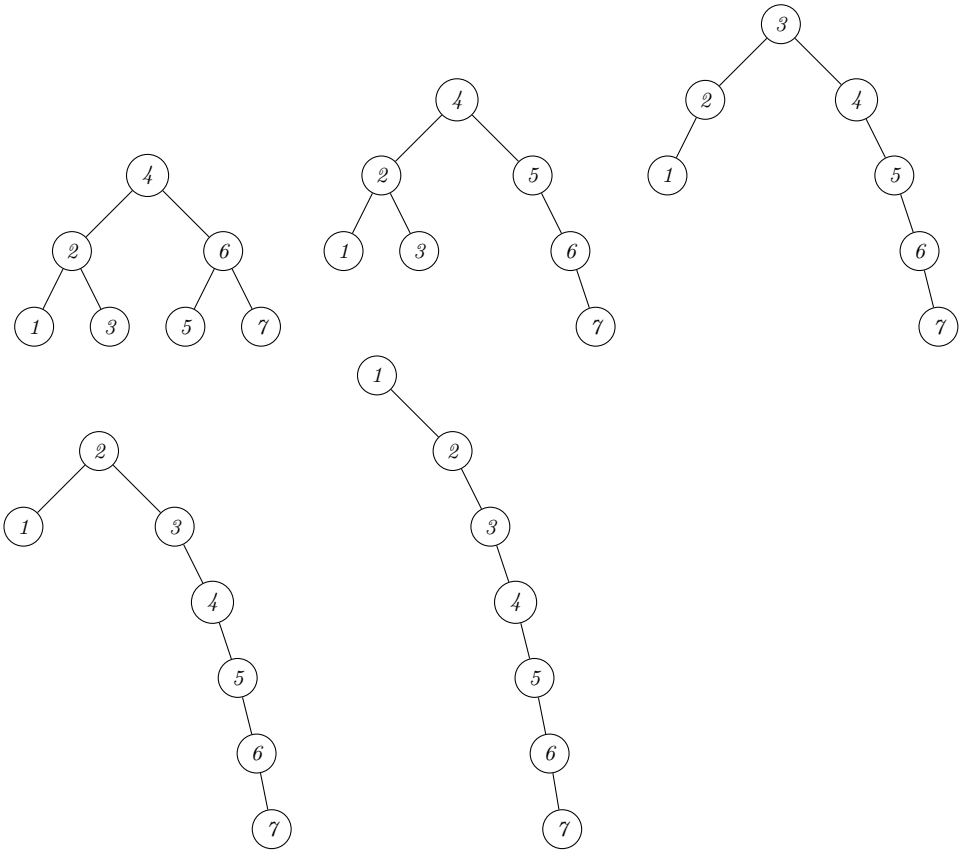
5. Soit A l'arbre vide. Insérer successivement dans A les noeuds d'étiquette, $5, 9, 4, 2, 7, 1, 6, 3, 8$ en appliquant l'algorithme d'insertion dans un ABR vu en cours.

6. Supprimer alors les noeuds d'étiquette 1, puis 2, puis 5, puis 6 de A en appliquant l'algorithme de suppression d'un noeud dans un ABR vu en cours.

Correction :

1. *(a) oui, il s'agit d'une feuille ;*
(b) non, l'arbre n'est pas binaire ;
(c) non, les clés ne respectent pas la definition de ABR ;
(d) non, car la clé 4 est mineur de 5 et pourtant elle est à droite de 5 ;
(e) oui ;
(f) oui.

2. *Voici les possibles ABR,*



3. *Un seul puisqu'il faut que le parcours infixe (que est unique) corresponde à l'ordre croissant des éléments.*

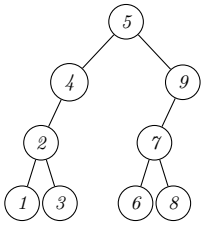
4. *Nous pouvons utiliser deux méthodes.*

(a) La première méthode est de vérifier si un arbre A satisfait les conditions de la définition d'un ABR. L'algorithme doit vérifier que toutes les clefs à gauche de la racine sont plus petites (ou égales à) celle de la racine, que toutes les clefs à droite sont plus grandes, et (recursivement) que les sous-arbres gauche et droite de A sont aussi des ABR.

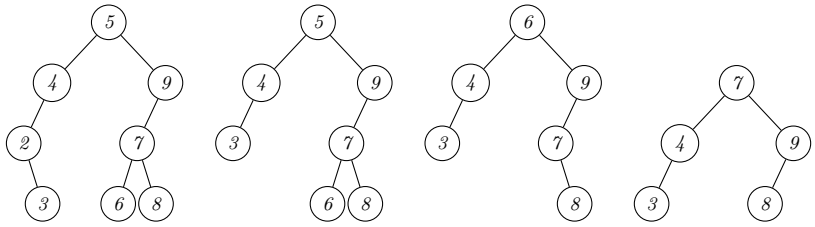
(b) Tranformer l'arbre en une liste et vérifier si la liste est triée.

Attention ! *Il y a un algorithme naïf qui ne marche pas ! Cet algorithme vérifie récursivement que la clef d'un noeud est plus petite que la racine de son sous-arbre droit et plus grande que la racine de son sous-arbre gauche. Par exemple, si on exécute cet algorithme sur l'arbre (d) de l'exercice 2.1, on obtient* `True`*, tandis que il ne s'agit pas d'un ABR.*

5. *En exécutant l'algorithme de insertion defini dans les transparents, on obtient comme resultat final l'arbre suivant :*



6. *En exécutant l'algorithme de suppression defini dans les transparents, on obtient les arbres suivants :*



Exercice 3 : manipulation d'insertions

1. Dessiner les trois ABR obtenus par insertion successive pour les différents ordres suivants :
- `1, 9, 8, 2, 3, 7, 6, 4, 5`
- `4, 2, 1, 3, 6, 5, 8, 7, 9`
- `4, 1, 2, 3, 8, 6, 5, 7, 9`

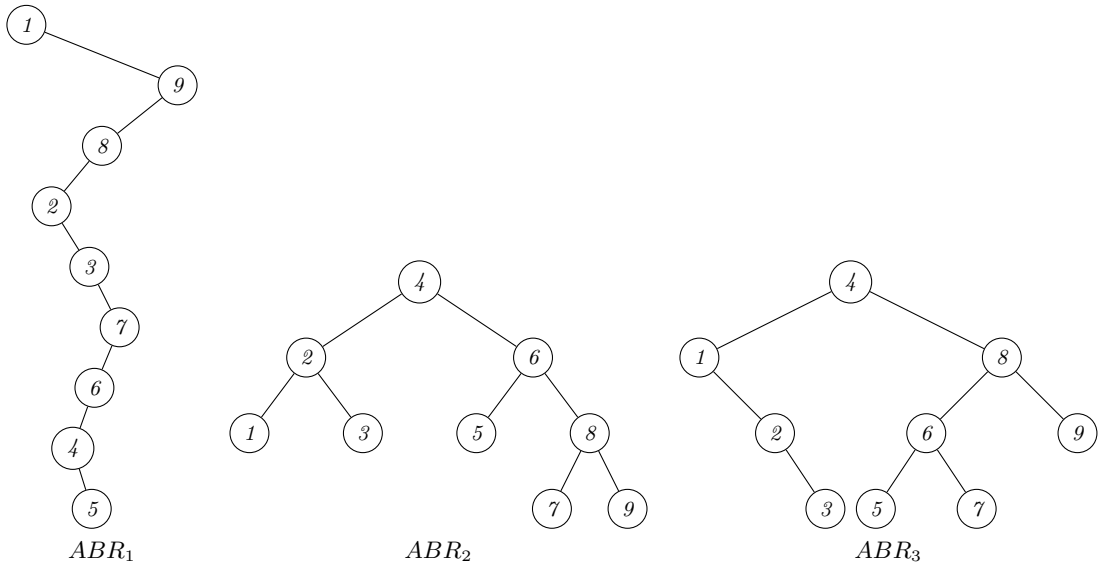
2. Proposer, si possible, d'autres ordres menant aux mêmes ABR.

3. Pour chacun des ABR, dénombrer les ordres possibles.

4. On appelle arbre binaire *parfait* un arbre binaire dont toutes les feuilles sont à la profondeur maximale – autrement dit, tous ses niveaux sont entièrement remplis. Donner une équation de récurrence pour $N(h)$, le nombre d'ordres possibles pour l'ABR parfait de hauteur h.

Correction :

1. *Voici les tris ABR,*



ABR_1 ABR_2 ABR_3

2.
- *Premier arbre : Il n'y a pas d'alternative.*
- *Deuxième arbre, par exemple :* `4, 6, 5, 8, 2, 7, 9, 1, 3`.
- *Troisième arbre, par exemple :* `4, 8, 9, 6, 5, 1, 2, 3, 7`.

3. *On remarque que la seule contrainte que l'on a sur l'ordre d'insertion des éléments est que pour tout ABR (y compris ses sous-arbres), la racine doit être insérée avant les sommets de ses arbres-fils.*

Soit $A = (r, A_0, A_1)$ un ABR à n sommets, k le nombre de sommets de A_0, et $P(A)$ le nombre d'ordres possibles pour A.

La racine de A doit être insérée en premier. Les sommets de A_0 peuvent être insérés à n'importe quels k moments parmi les $n-1$ insertions restantes, soit $\binom{n-1}{k}$ choix possibles. Les sommets de A_0 ont $P(A_0)$ ordres possibles différents entre eux, et les sommets de A_1 en ont $P(A_1)$.

Ceci donne la relation de récurrence : $P(A) = \binom{n-1}{k} P(A_0)P(A_1)$, ce qui donne pour les arbres ci-dessus :
- $P(ABR_1) = 1$
- $P(ABR_2) = 896 = \binom{8}{3} \cdot \binom{2}{1} \cdot \left(\binom{4}{1} \cdot 1 \cdot \binom{2}{1}\right)$
- $P(ABR_3) = 448 = \binom{8}{3} \cdot 1 \cdot \left(\binom{4}{1} \cdot \binom{2}{1} \cdot 1\right)$

4. *Un ABR parfait de hauteur h a $n = 2^{h+1} - 1$ sommets, et ses sous-arbres gauche et droit sont des ABR parfaits de hauteur $h-1$ donc de $k = 2^h - 1$. D'où $N(0) = 1$ et $N(h) = N(h-1)^2 \cdot \binom{2^{h+1}-2}{2^h-1}$*