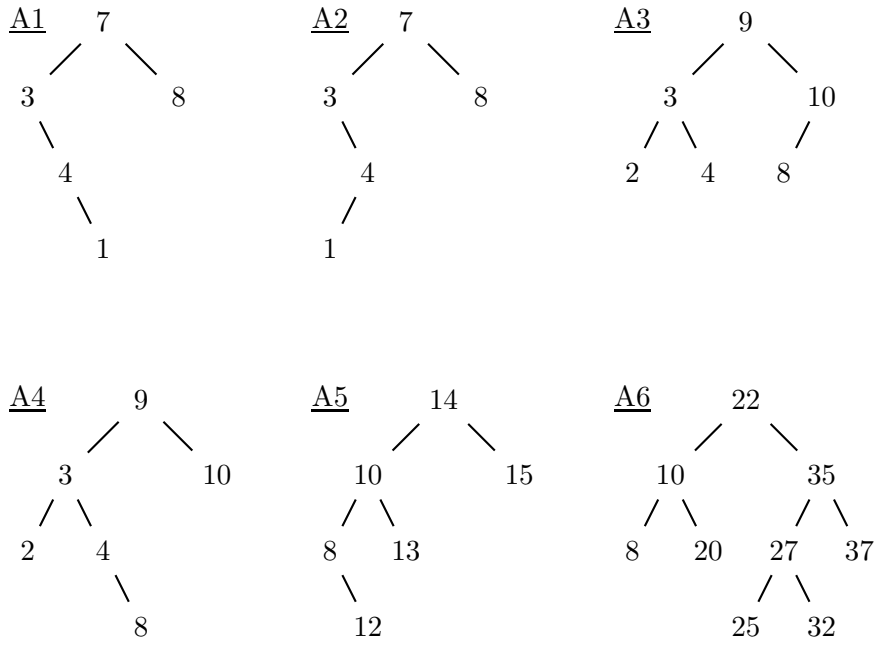
TD 11 - Arbres binaires de recherche

Objectif : Savoir manipuler les arbres binaires de recherche.

Exercice 1 - Déterminer parmi les arbres suivant ceux qui sont des arbres binaires de recherche.



Exercice 2 - Soit l'ensemble $\{25, 60, 35, 10, 20, 30, 5\}$. Construire un arbre binaire de recherche en ajoutant un par un les éléments de cet ensemble, selon leur ordre d'apparition :

— par ajout de feuilles ;

— par insertion de racines.

Dans chaque cas, vous déroulerez pas à pas l'algorithme vu en cours.

Exercice 3 - Ecrire un algorithme **récursif** qui prend en entrée un **arbre binaire de recherche** A, et qui renvoit le plus petit élément de A. Vous pourrez vous inspirer de l'algorithme *supMax* vu en cours.

Exercice 4 - Ecrire une version itérative de l'algorithme *supMax* vu en cours, qui renvoie le maximum d'un arbre et cet arbre privé du noeud correspondant.

Exercice 5 - Ecrire un algorithme récursif fusionnant deux arbres binaires de recherche en un seul. Vous pourrez faire appel à la fonction *coupe* définie dans le cours.

Exercice 6 : Suppression d'un sommet dans un ABR (*examen 2019*)

On rappelle ci-dessous l'algorithme *supMax* (algorithme 1) qui renvoie le sommet maximal d'un arbre binaire de recherche (ABR) et cet arbre privé de ce sommet (mais toujours ABR). Cet algorithme est utilisé par l'algorithme 2, qui supprime un sommet d'un ABR.

Écrivez en pseudo-code un algorithme *supMin* qui renvoie le sommet minimal d'un arbre binaire de recherche (ABR) et cet arbre privé de ce sommet (mais toujours ABR). Indiquez le ou les numéro(s) de ligne à changer dans l'algorithme 2 pour qu'il utilise *supMin* au lieu de *supMax*, et précisez le nouveau pseudo-code de la / des ligne(s) concernée(s).

Algorithm 1: supMax(A)

```
début
    /* ENTRÉES : un ABR A */
    /* SORTIE : l'élément max, l'ABR A privé de l'élément max*/
    si est_vide(A) alors Retour A
    sinon
        si est_vide(D(A)) alors
            Retour (racine(A), G(A))
        sinon
            (x,B) ← supMax(D(A))
            Retour (x, ⟨racine(A), G(A), B⟩)
fin
```

Algorithm 2: supprime(x,A)

```
début
    /* ENTRÉES : un ABR A, un élément x */
    /* SORTIE : l'ABR A privé de x */
1   si est_vide(A) alors Retour A
    sinon
2       r ← racine(A)
3       si x > r alors B ← supprime(x, D(A)) ; Retour ⟨r, G(A), B⟩
4       sinon si x < r alors B ← supprime(x, G(A)) ; Retour ⟨r, B, D(A)⟩
        sinon
5           si est_vide(G(A)) alors Retour D(A)
6           sinon si est_vide(D(A)) alors Retour G(A)
7           sinon (max, B) ← supMax(G(A)) ; Retour ⟨max, B, D(A)⟩
fin
```