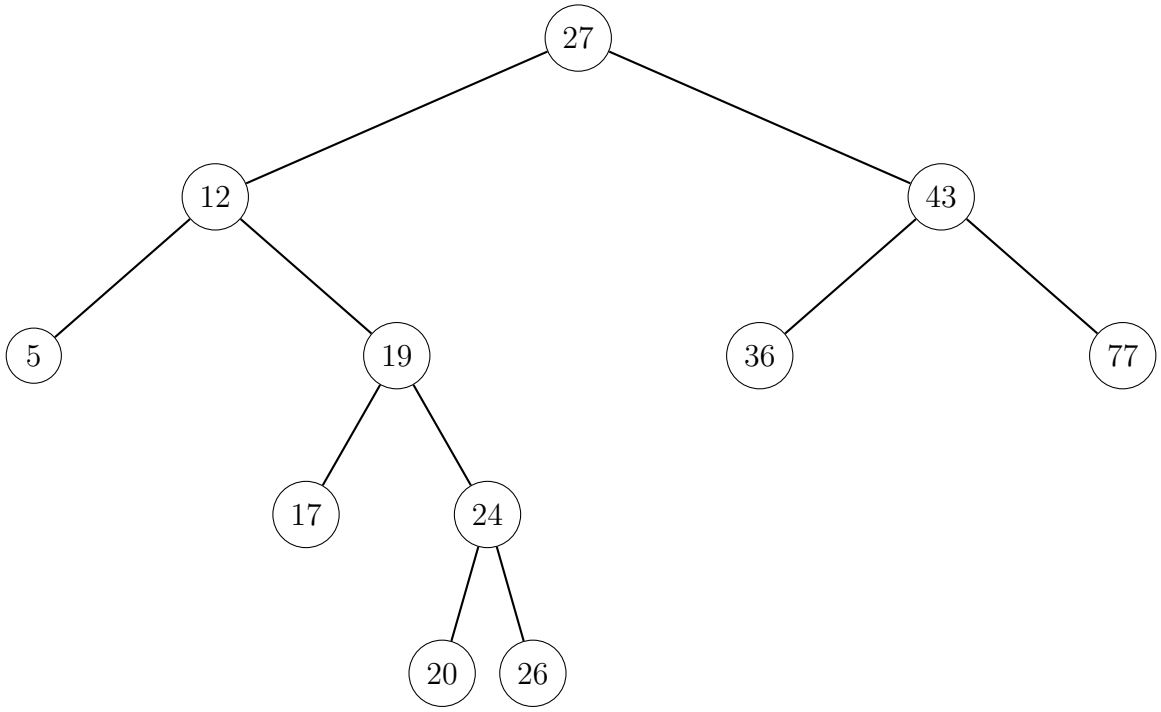
Fiche 4 bis

Arbres binaires de recherche

Nous reprendrons la construction de l'arbre binaire vu en cours.
Un arbre binaire non vide étiqueté est appelé arbre binaire de recherche si les étiquettes appartiennent à un ensemble totalement ordonné et s'il vérifie l'une des deux conditions équivalentes suivantes :

- La liste des étiquettes en ordre infixe est croissante au sans large.
- Pour tout nœud x d'étiquette e, les étiquettes des éléments de la branche gauche de x sont inférieures ou égales à e et les étiquettes des éléments de la branche droite de x sont supérieures ou égales à e.

1. Définir l'arbre binaire de recherche suivant :`ex_1` :



2. Écrire la fonction `recherche` qui, pour un x donné, vérifie si x est l'étiquette d'un arbre de recherche donné.
   Estimer la complexité de la fonction recherche.

3. Écrire la fonction `min_gauche_max_droit` qui retourne l'étiquette du nœud le plus à gauche, ainsi que l'étiquette du nœud le plus à droite.
   En déduire la fonction `verif_arb_rech` qui retourne vrai si l'arbre est un arbre de recherche, et faux sinon.

4. La deuxième méthode consiste à donner la liste des étiquettes dans l'ordre infixe et à vérifier que la liste est triée.

   (a) Écrire une fonction `verif_tri` qui vérifie si une liste donnée est triée par une relation d'ordre donné.

   (b) En retrouvant la fonction infixe, en déduire la fonction `verif_arb_rech2`.

On désire insérer un nouvel élément dans un arbre binaire de recherche sans changer sa caractéristique d'arbre de recherche.

5. **Insertion aux feuilles :**
   Écrire une fonction qui place un nouvelle élément sur une branche vide de l'arbre ou renvoie cet arbre si l'élément était déjà présent.

6. **Insertion à la racine :**
   On souhaite écrire une fonction qui place un nouvelle élément à la racine du nouvel arbre, ou renvoie cet arbre si l'élément était déjà présent.
   L'idée est de découper l'arbre en deux, et de le recoller avec la nouvelle racine.

   (a) Écrire la fonction `decoupe` qui prend un arbre de recherche *abre_rech* et un entier x. La fonction renvoie deux arbres de recherche extrait de *abre_rech* , le premier constitué des éléments inférieur à x, le deuxième constitué des éléments supérieur à x.

   (b) En déduire la fonction `insere_racine`.

7. Écrire une fonction `construction_feuille` qui utilise la fonction `insere_feuille` pour construire un arbre de recherche composé des éléments d'une liste donnée.

   Écrire une fonction `construction_racine` qui utilise la fonction `insere_racine` pour construire un arbre de recherche composé des éléments d'une liste donnée.

   Comparer les arbres obtenus avec les listes :

```
liste_ex_1 = [1,2,3,4,5,6,7,8,9]
liste_ex_2 = [1,3,5,7 ,9 ,8,6,4,2]
```