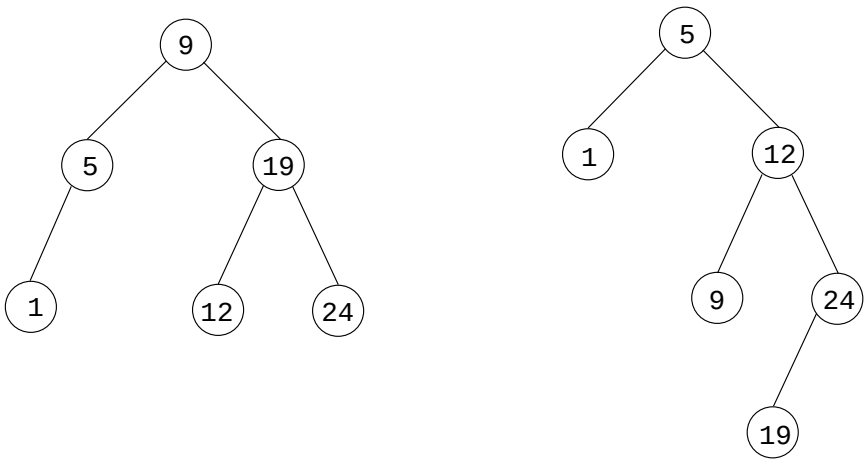

Pour tester l'appartenance d'un élément à un ensemble représenté par un arbre de recherche, on commence par tester si l'arbre est vide. Si c'est le cas on renvoie le booléen `false`. Sinon, on compare l'élément recherché à l'élément contenu dans la racine de l'arbre. Si ces deux éléments sont identiques on renvoie la valeur `true`. Si l'élément recherché est inférieur à l'élément contenu dans la racine, on peut ignorer le sous-arbre droit et appliquer récursivement la même méthode au sous-arbre gauche. Et on procède de manière symétrique quand l'élément recherché est supérieur à l'élément contenu dans la racine. Le temps nécessaire pour tester si un élément appartient à l'ensemble ou non est, au pire, proportionnel à la hauteur de l'arbre.

Pour programmer cette recherche en Java, nous pouvons choisir d'utiliser une méthode statique, ce qui nous permettra de l'utiliser même quand l'arbre est vide.

```
static boolean recherche (final int x, final Arbre a) {
 if (a == null) return false;
 if (x == a.val) return true;
 if (x < a.val) return recherche (x,a.gauche);
 return recherche (x,a.droit);}
```

De même, l'insertion d'un nouvel élément dans l'arbre demande un temps proportionnel à la hauteur de l'arbre. Nous choisissons d'utiliser une méthode qui modifie son argument et non une méthode qui le recopie. Cette méthode renvoie un arbre qui est le résultat de cette insertion. Remarquons que, quand on exécute l'instruction `insert(a) ;`, la valeur de `a` après que la méthode a été exécutée est également le résultat de cette insertion, sauf dans le cas où `a` est l'arbre vide.

```
static Arbre insert (final int x, final Arbre a) {
```