**Algorithme sur les arbres binaires de recherche**

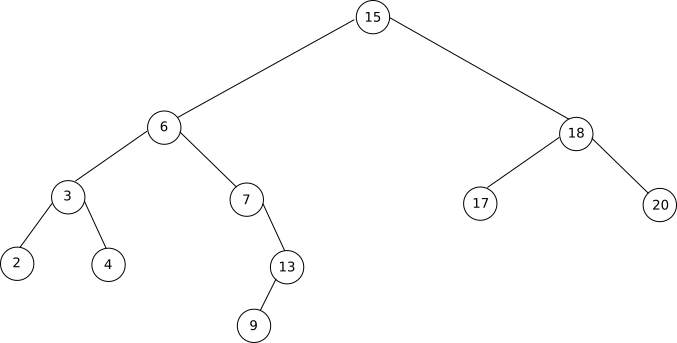
1. **Arbre binaire de recherche**

**Arbre binaire de recherche**

Un arbre binaire de recherche est un cas particulier d'arbre binaire. Pour avoir un arbre binaire de recherche :

* il faut avoir un arbre binaire !
* il faut que les clés de nœuds composant l'arbre soient ordonnables (on doit pouvoir classer les nœuds, par exemple, de la plus petite clé à la plus grande)
* soit x un nœud d'un arbre binaire de recherche. Si y est un nœud du sous-arbre gauche de x, alors il faut que y.clé ⩽ x.clé. Si y est un nœud du sous-arbre droit de x, il faut alors que x.clé ⩽ y.clé

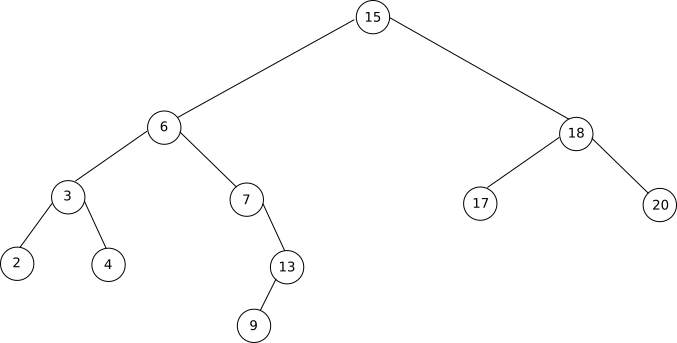
**À faire vous-même 7**



Vérifiez que l'arbre ci-dessus est bien un arbre binaire de recherche.

**À faire vous-même 8**

Appliquez l'algorithme de parcours infixe sur l'arbre ci-dessous :



Que remarquez-vous

1. **Recherche d’une clé dans un arbre binaire de recherche**

**Recherche d'une clé dans un arbre binaire de recherche**

Nous allons maintenant étudier un algorithme permettant de rechercher une clé de valeur k dans un arbre binaire de recherche. Si k est bien présent dans l'arbre binaire de recherche, l'algorithme renvoie vrai, dans le cas contraire, il renvoie faux.

**À faire vous-même 9**

Étudiez l'algorithme suivant:

VARIABLE

T : arbre

x : nœud

k : entier

DEBUT

ARBRE-RECHERCHE(T,k) :

si T == NIL :

renvoyer faux

fin si

x ← T.racine

si k == x.clé :

renvoyer vrai

fin si

si k < x.clé :

ARBRE-RECHERCHE(x.gauche,k)

sinon :

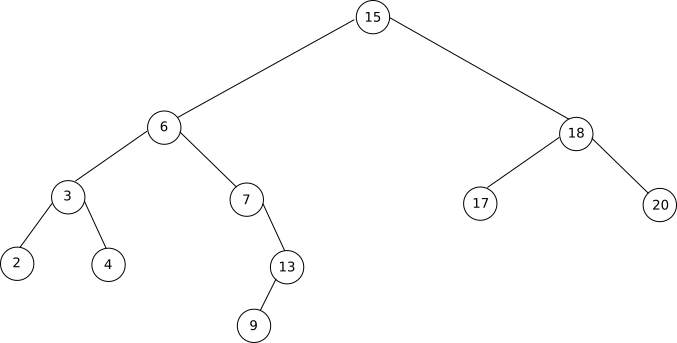
ARBRE-RECHERCHE(x.droit,k)

fin si

FIN

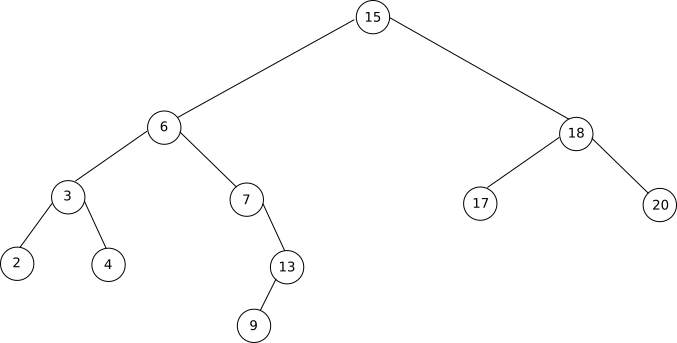
**À faire vous-même 10**

Appliquez l'algorithme de recherche d'une clé dans un arbre binaire de recherche sur l'arbre ci-dessous. On prendra k = 13.



**À faire vous-même 11**

Appliquez l'algorithme de recherche d'une clé dans un arbre binaire de recherche sur l'arbre ci-dessous. On prendra k = 16.



Cet algorithme de recherche d'une clé dans un arbre binaire de recherche ressemble beaucoup à la recherche dichotomique vue en première dans le cas où l'arbre binaire de recherche traité est équilibré. La complexité en temps dans le pire des cas de l'algorithme de recherche d'une clé dans un arbre binaire de recherche équilibré est donc O(log2(n))O(log2(n)). Dans le cas où l'arbre est filiforme, la complexité est O(n)O(n). Rappelons qu'un algorithme en O(log2(n))O(log2(n)) est plus "efficace" qu'un algorithme en O(n)O(n).

À noter qu'il existe une version dite "itérative" (qui n'est pas récursive) de cet algorithme de recherche :

**À faire vous-même 12**

Étudiez l'algorithme suivant:

VARIABLE

T : arbre

x : nœud

k : entier

DEBUT

ARBRE-RECHERCHE\_ITE(T,k) :

x ← T.racine

tant que T ≠ NIL et k ≠ x.clé :

x ← T.racine

si k < x.clé :

T ← x.gauche

sinon :

T ← x.droit

fin si

fin tant que

si k == x.clé :

renvoyer vrai

sinon :

renvoyer faux

fin si

FIN

1. **Insertion d’une clé dans un arbre binaire de recherche**

**Insertion d'une clé dans un arbre binaire de recherche**

Il est tout à fait possible d'insérer un noeud y dans un arbre binaire de recherche (non vide) :

**À faire vous-même 13**

Étudiez l'algorithme suivant:

VARIABLE

T : arbre

x : nœud

y : nœud

DEBUT

ARBRE-INSERTION(T,y) :

x ← T.racine

tant que T ≠ NIL :

x ← T.racine

si y.clé < x.clé :

T ← x.gauche

sinon :

T ← x.droit

fin si

fin tant que

si y.clé < x.clé :

insérer y à gauche de x

sinon :

insérer y à droite de x

fin si

FIN

**À faire vous-même 14**

Appliquez l'algorithme d'insertion d'un noeud y dans un arbre binaire de recherche sur l'arbre ci-dessous. On prendra y.clé = 16.

