

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

VARIABLE

T : arbre

x : noeud

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)

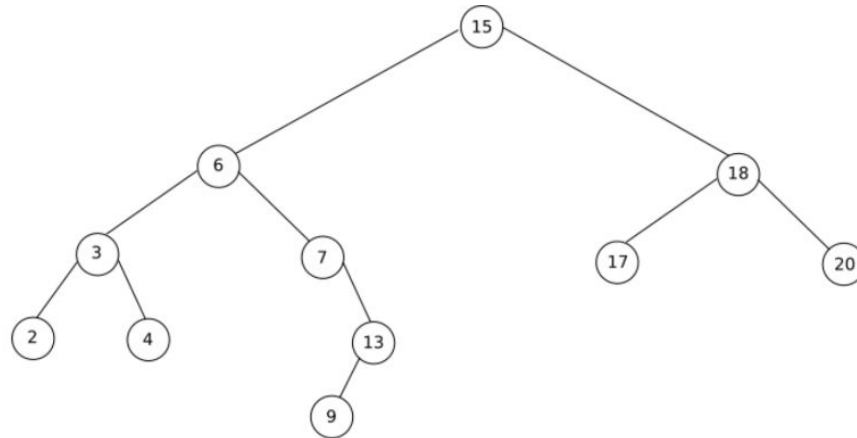
sinon :

ARBRE-RECHERCHE(x.droit,k)

fin si

FIN

T	T == NIL	x	h	h == clé	h < x.clé	si non
	faux	x = T.racine x = 15	13	13 == 15 faux	6 Vrai	
	faux	x = T.racine x = 6	13	13 == 6 faux	faux	7
	faux	x = 7	13	13 == 7 faux	Faux	13
	faux	x = 13	13	13 == 13 vrai		
Tableaux 1 ↑						



k = 13

VARIABLE

T : arbre

x : noeud

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)

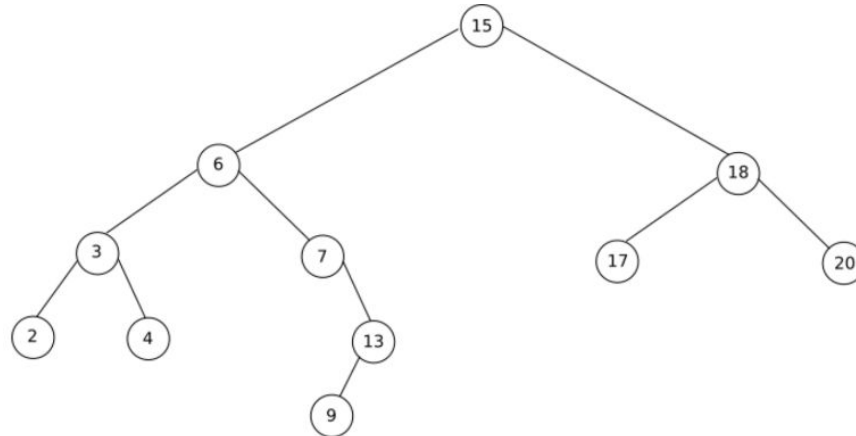
sinon :

ARBRE-RECHERCHE(x.droit,k)

fin si

FIN

T	T==NIL	rac	h	h==rac.clé	h < rac.clé	sinon
	faux	rac = T.garçine rac = 15	16	16 == 15 faux	faux	18
	faux	rac = 18	16	16 == 18 faux	Vrai 17	
	faux	rac = 17	16	16 == 17 faux	Vrai	
	vrai					



k = 16

Étudiez l'algorithme suivant avec l'arbre précédent :

VARIABLE

T : arbre

x : noeud

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

Tableau 3

T	se	h	T ≠ NIL	k ≠ x.clé	h < x.clé	si h == x.clé	sinon
<div> <div>15</div> <div> 16 18 </div> </div>	15	20	vrai	vrai 20 ≠ 15	20 < 15 faux sinon: T = 18		
<div> <div>18</div> <div> 17 20 </div> </div>	18	20	vrai	vrai 20 ≠ 18	20 < 18 faux sinon: T = 20		
<div> <div>20</div> </div>	20	20	vrai	vrai 20 = 20	faux		
			faux			h == x.clé 20 == 20 vrai	