Árboles binarios:

* Un árbol es una estructura de datos jerárquica, es decir la forma en la que se conectan los elementos, permite identificar niveles de importancia.
* Los árboles se componen de nodos. Cada nodo contiene un dato y una o más referencias a otros nodos.
* En un árbol binario, cada nodo tiene a lo sumo dos nodos.
* Existen árboles n-ários, en los que los nodos pueden tener más de dos nodos.
* Estructura básica de un árbol binario.
* Un árbol binario de búsqueda (BST) cumple la condición:
* El hijo derecho es mayor que la raíz.
* El hijo izquierdo es menor que la raíz.
* Estas condiciones se cumplen en cualquier parte del árbol.
* Un BST provee búsquedas rápidas y eliminaciones/inserciones eficientes.
* ¿Cómo se define un nodo en java?

Class Nodo{

Int elemento;

Nodo right;

Nodo left;

}

* ¿Cómo se define el árbol?

Class BST{

Nodo root = null;

Boolean isEmpty(){

Return root == null;

}

}

* ¿Cómo se busca en el árbol?

Boolean contains(int e){

Return this.contains(e, root);

}

Private boolean contains(int element, Nodo current){

If(current == null){

Return false;

} else if (element < current.element){

Return contains(e, current.left);

} else if (element > current.element){

Return contains(e, current.right);

} else {

Return true;

}

}

* ¿Cómo encontrar el mayor/menor?

Public Nodo findMin(){

Return this.findMin(this.root);

}

Private Nodo findMin(Nodo current){

If(current == null){

Return null;

} else if (current.left == null){

Return current;

} else {

Return findMin(current.left);

}

}

* ¿Cómo se inserta en un árbol?

Public void insert(int e) {

Root = this.insert(e, this.root);

}

Private Nodo insert (int e, Nodo current){

If(current == null){

Return new Node(e);

} else if(e < current.element){

Current.left = insert(e, current.left);

} else {

Current.right = insert(e, current-right);

}

Return current;

}