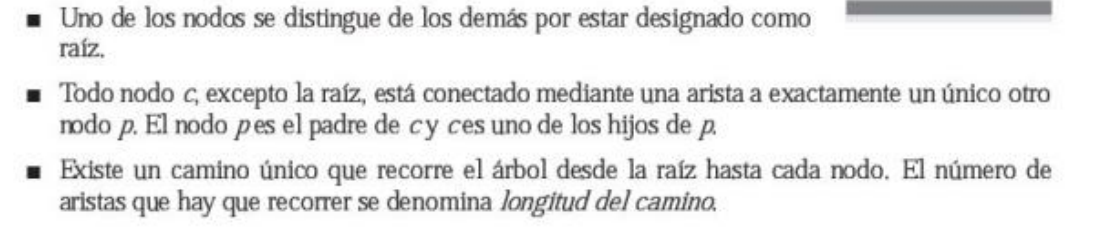
# Arboles Binarios

Un árbol con raiz cumple con las siguientes propiedades:  


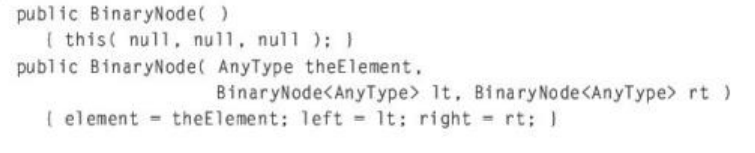
Un árbol con N nodos debe tener N-1 aristas, porque todo nodo salvo la raiz, tiene una arista entrante.

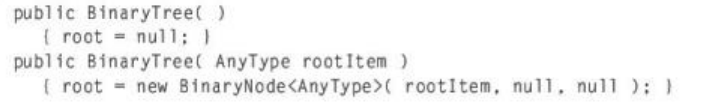
La **profundidad** de un nodo de un árbol es la longitud del camino que va desde la raiz hasta ese nodo. Por lo que la profundidad de la raíz siempre es 0 y la profundidad de cualquier nodo es 1 unidad más que la profundidad de su padre.

La **altura** de un nodo es la longitud del camino que va desde el nodo hasta la hoja más profunda. La altura de cualquier nodo es 1 más que la de su hijo de máxima altura. La altura de un árbol es la altura de su raíz.

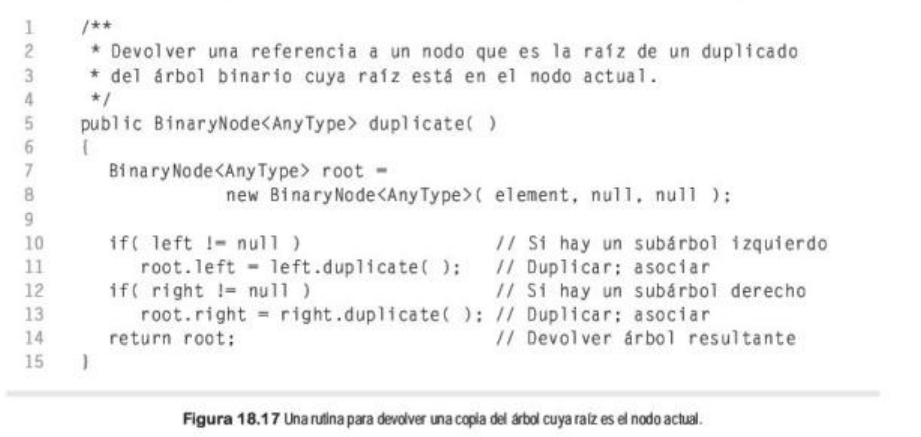
El tamaño de un nodo es el numero de descendientes que tiene el nodo (incluyendo el propio nodo).

Un arbol binario es un arbol en el que ningun nodo puede tener mas de 2 hijos. Recursivamente, un arbol binario o esta vacio o esta compuesto de una razi, un arbol izquierdo y un arbol derecho.

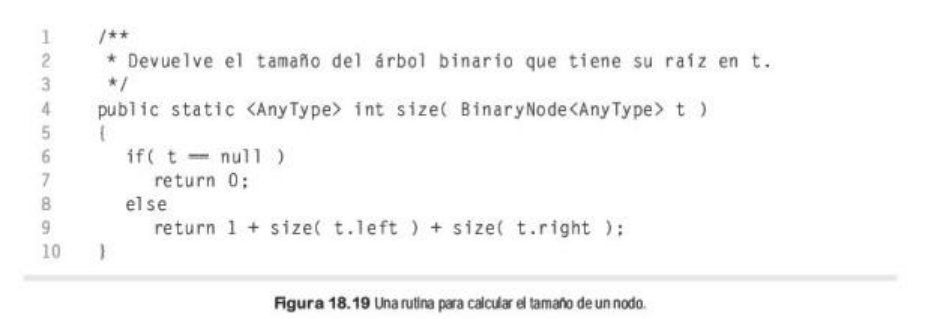




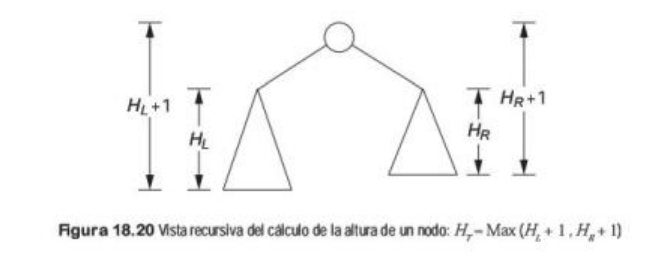
Implementación de metodo de duplicar en un arbol:

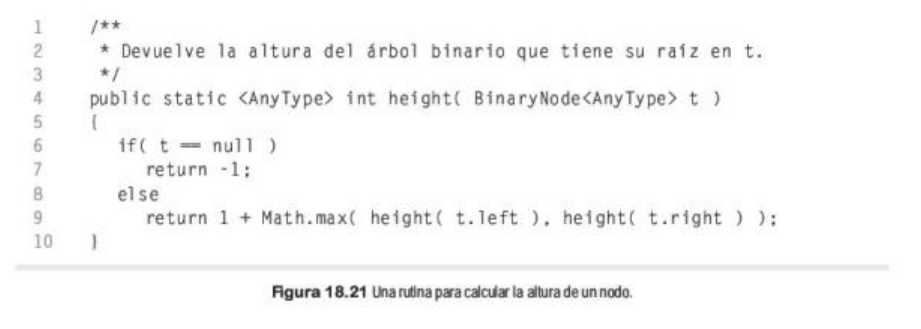


Calcular el tamaño de un árbol:



Calcular la altura de forma recursiva:

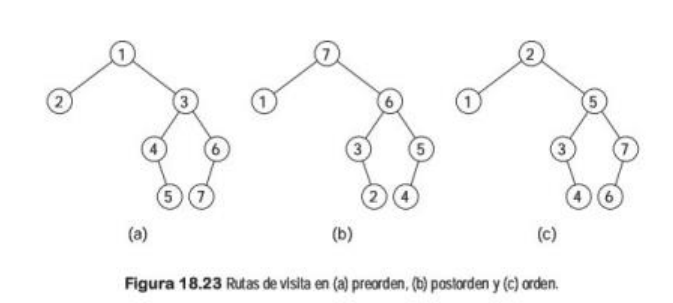




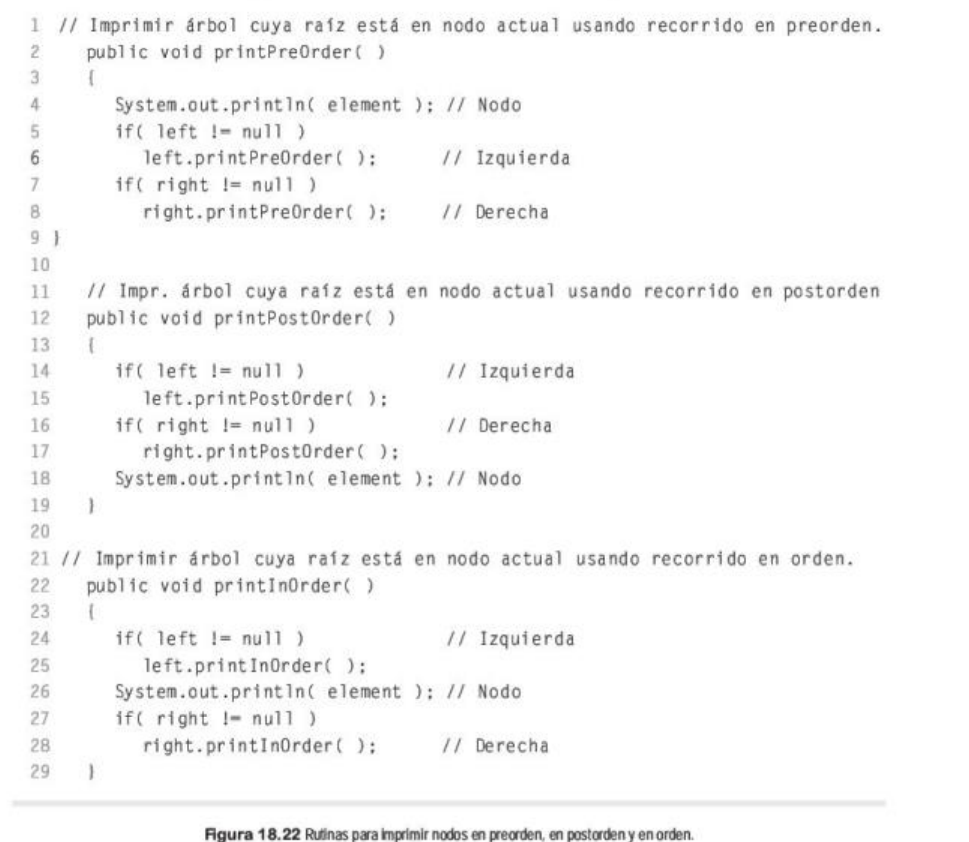
### Tipos de Recorridos en Recursión

**Preorden:** primero se procesa el nodo y luego se procesan recursivamente sus hijos (ej: duplicate).

**Postorden:** el nodo se procesa después de haber procesado recursivamente ambos hijos (ej: size y height)

**Recorrido en Orden:** se procesa recursivamente el hijo izquierdo, luego se procesa el nodo actual y después el hijo derecho. Si el arbol esta ordenado y se recorre en inorden, entonces devuelve los valores del árbol ordenados.

* En el recorrido inorden (orden) , para elegir el primer hijo izquierdo se empieza por el primero que no tenga mas hijos izquierdos.
* En el postorden, el último nodo es el nodo raíz del árbol.

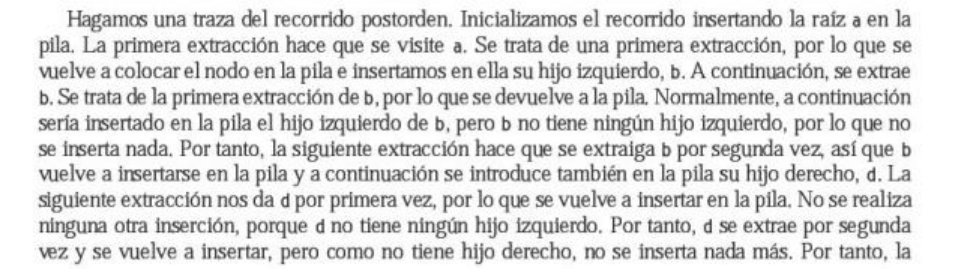


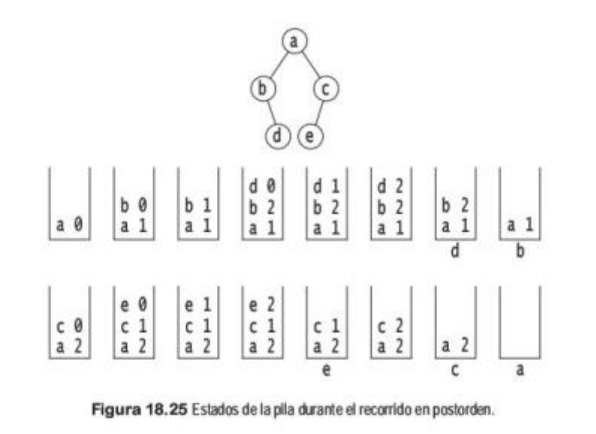
**Recorrido en postorden:**

Se implementa una pila para guardar el estado actual. La cima representa el nodo que estamos visitando en algún instante dentro del recorrido. Podemos estar en uno de tres lugares dentro del algoritmo: 1. A punto de hacer una llamada recursiva al árbol izquierdo. 2. A punto de hacer una llamada recursiva al árbol derecho. 3. A punto de procesar el nodo actual.

Es decir que el recorrido en postorden mantiene una pila que almacena los nodos que han sido visitados pero cuyas llamadas recursivas no han sido completadas todavía.

Cada nodo se inserta tres veces en la pila y la tercera vez que se extrae, el nodo se declara como visitado. Las veces anteriores simulamos una llamada recursiva.





**Recorrido en orden:**

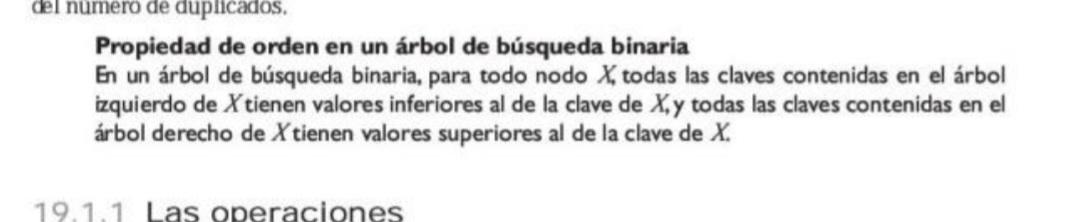
Es igual al recorrido en postorden pero un nodo se declara visitado despues de haber sido extraido una segunda vez. Despues de volver, el iterador inserta en la pila el hijo derecho para que la siguiente llamada pueda continuar recorriendo el hijo derecho.

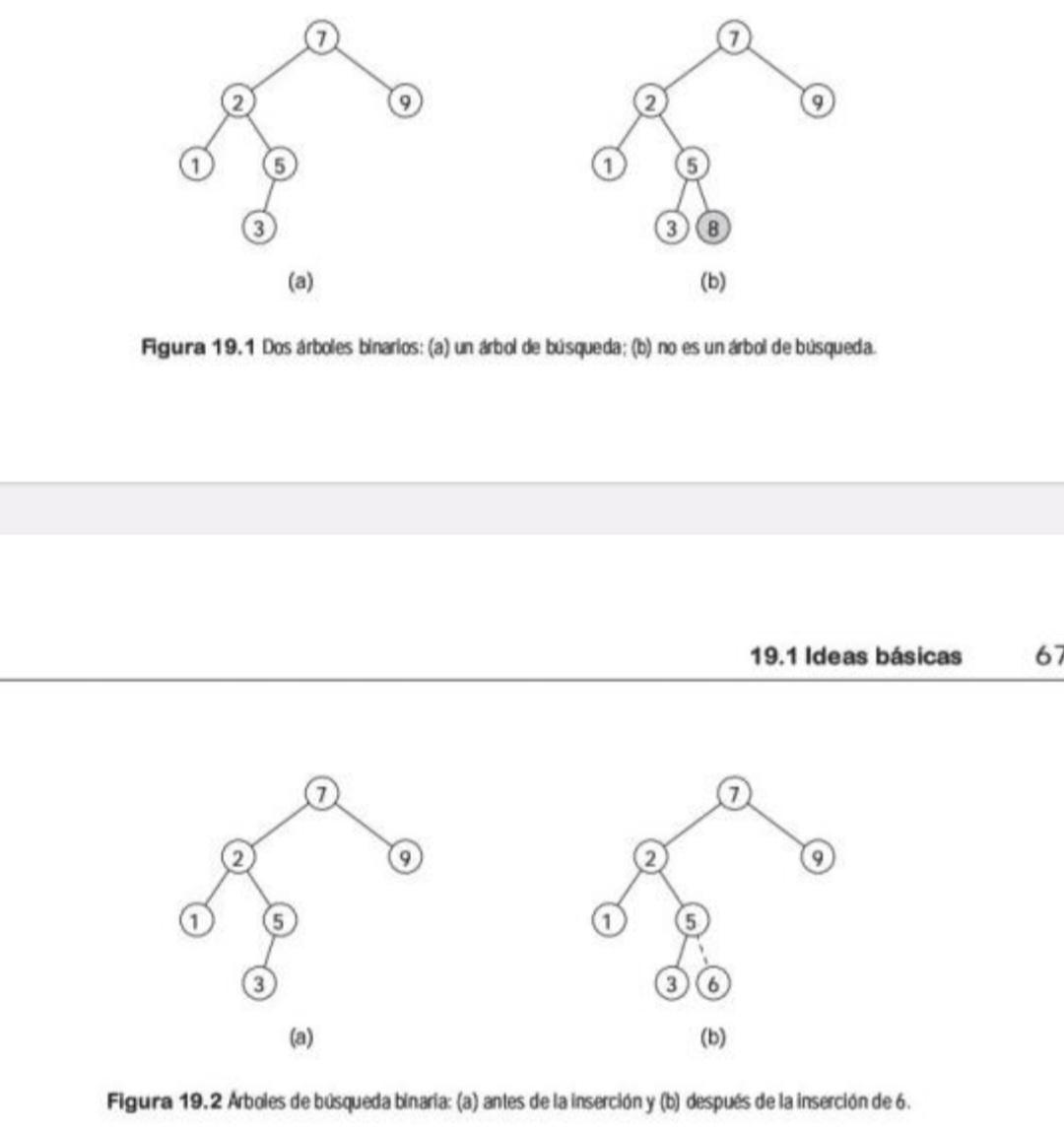
**Recorrido en preorden:**

Un nodo se declara como visitado al haber sido extraído por primera vez. El hijo derecho y luego el izquierdo se insertan en la pila antes de volver. Con esto el hijo izquierdo se procesa antes que el derecho.

### Árboles Binarios de Búsqueda

Para grandes cantidades de datos, el tiempo de acceso lineal en las listas enlazadas es muy grande. Como alternativa para inserción y borrado se usan los Árboles de Búsqueda **Binaria donde el tiempo de ejecución para la mayoría de los casos es O(logN) y el tiempo para el peor caso es O(N).**



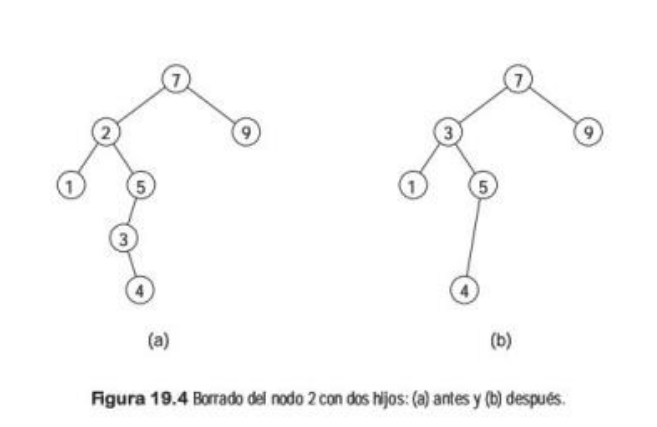


Para hacer una operación para **encontrar el elemento más pequeño**, comenzamos en la raíz y vamos bifurcando repetidamente a la izquierda hasta que deje de haber un hijo izquierdo. El punto en que nos detenemos es el elemento más pequeño. Para hacer una operación para **encontrar el elemento más grande** es similar, ya que que bifurca hacia la derecha. El coste de las operaciones es proporcional al número de nodos en el camino de la búsqueda. El coste tiende a ser logarítmico.

La **eliminación** de elementos es la más costosa. Tiene el problema de que la eliminación de un nodo puede dejar desconectadas parte del árbol y en ese caso se tiene que volver a asociar siguiendo la propiedad del orden de un árbol de busqueda binaria.

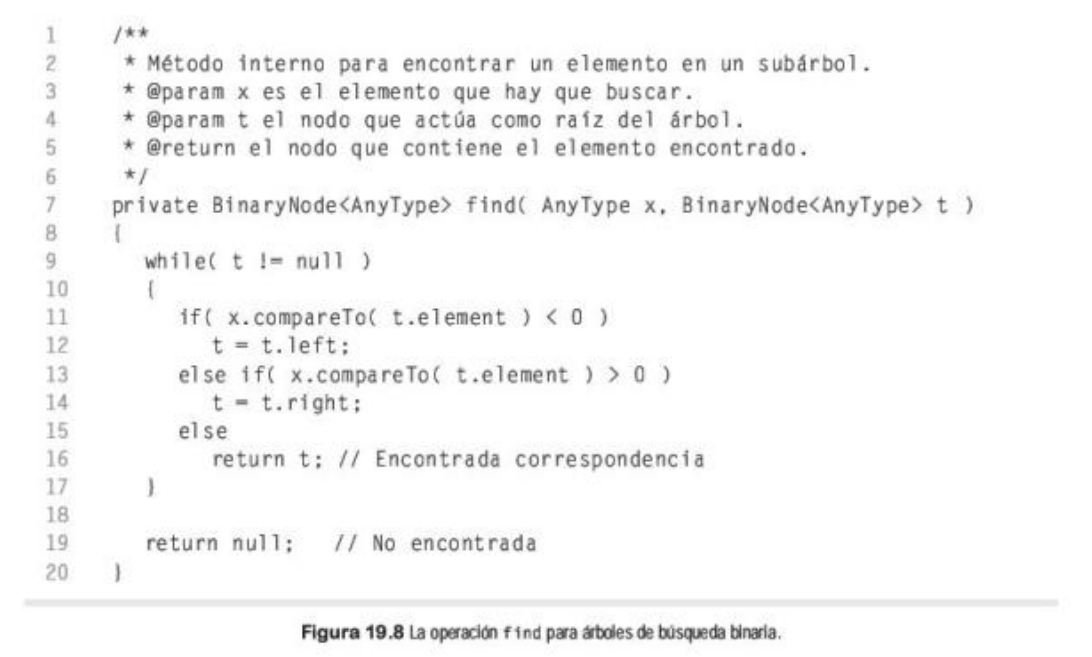
Eliminación de elementos (Descripción en lenguaje natural):

* Si el nodo es una hoja se puede eliminar directamente ya que no deja partes del árbol desconectadas
* Si el nodo tiene un solo hijo, se ajusta el enlace para apuntar desde el padre del nodo hacia el hijo del nodo, eliminando el nodo
* Si el nodo es la raíz, se elimina directamente
* Si un nodo tiene dos hijos, se reemplaza el elemento del nodo por el elemento más pequeño del subárbol derecho. Esto es posible porque el nodo mínimo de un árbol no tiene un hijo izquierdo. Sino, se puede tomar el elemento mas grande del subarbol izquierdo.

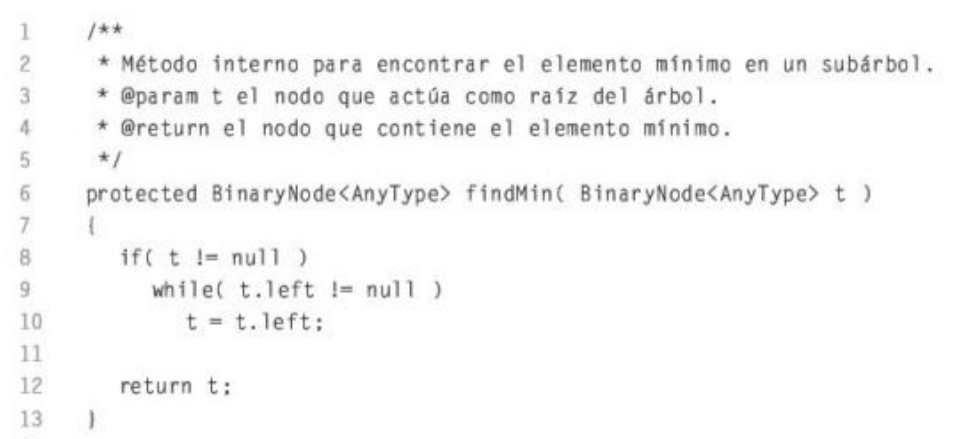


Ejemplos de Implementación en Java:

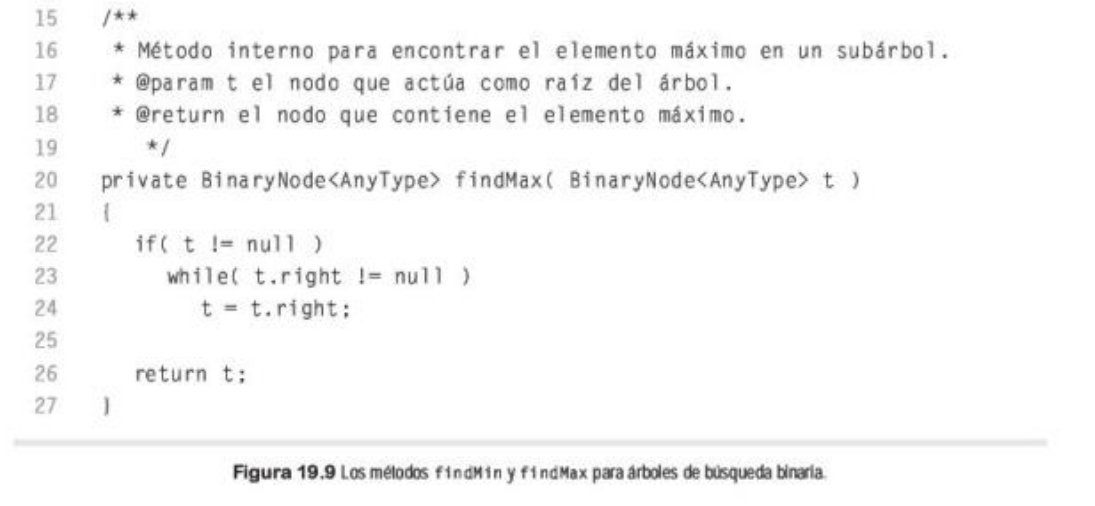
Buscar un nodo:



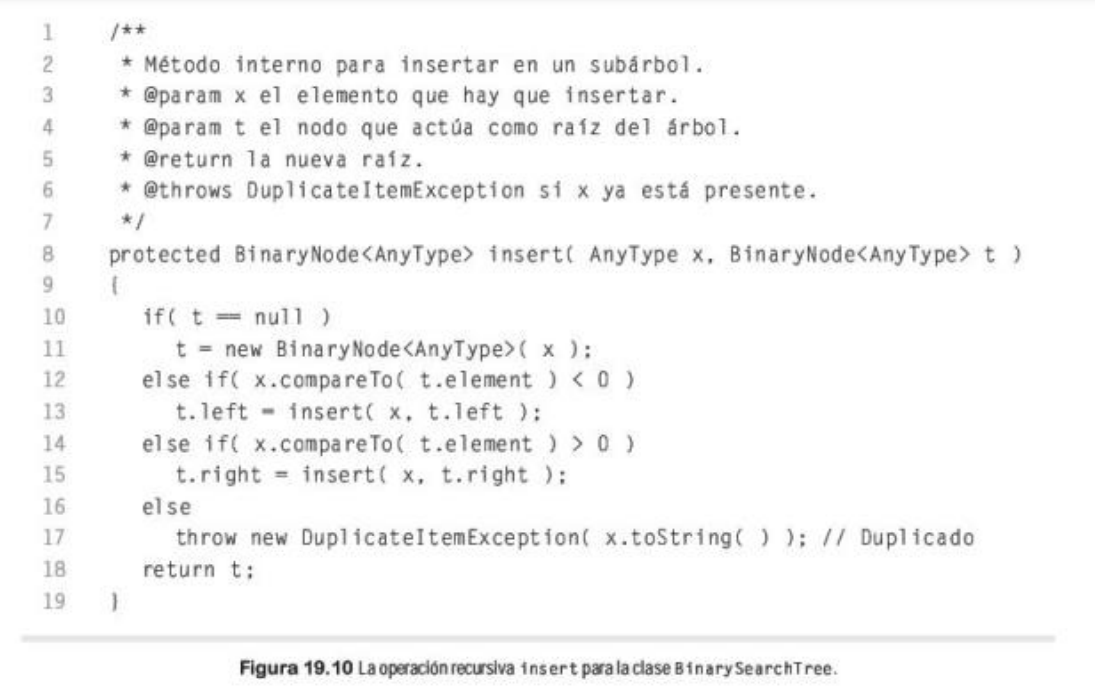
findMin:



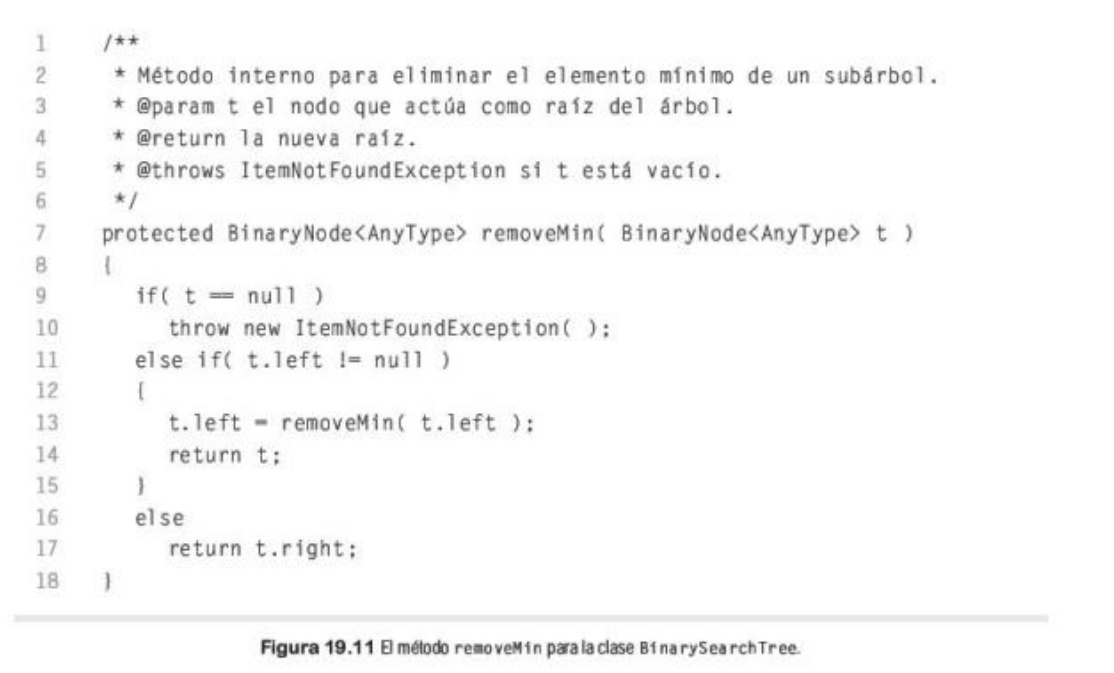
findMax:



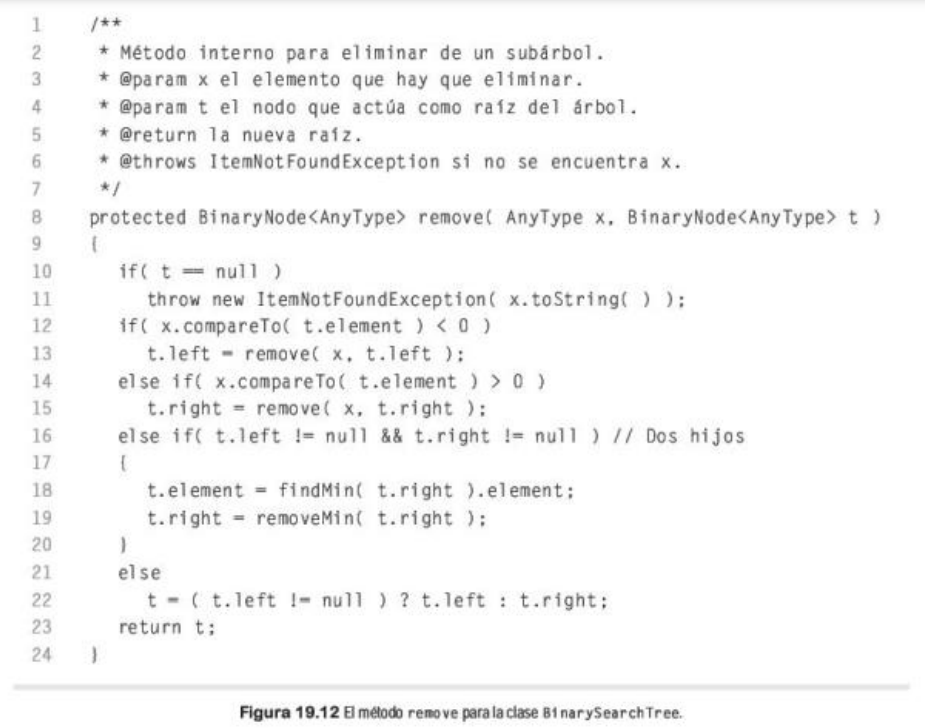
Inserción:



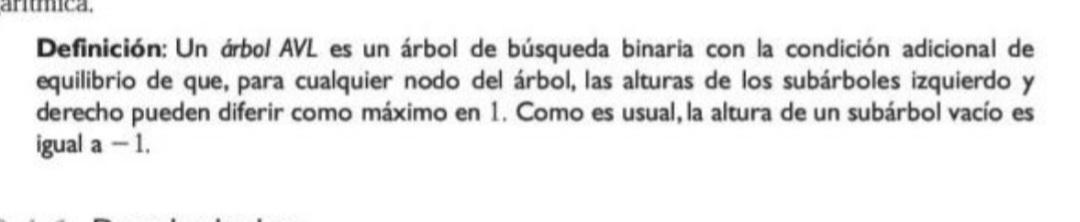
removeMin:



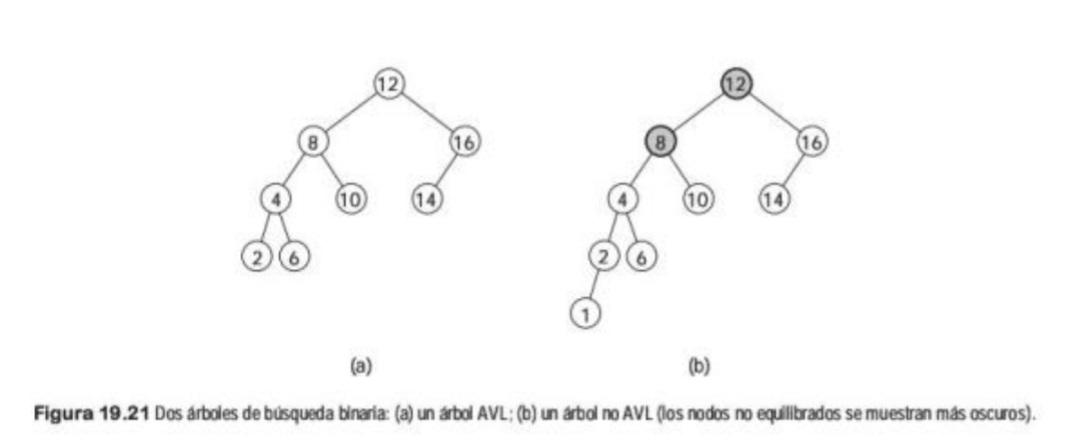
remove:



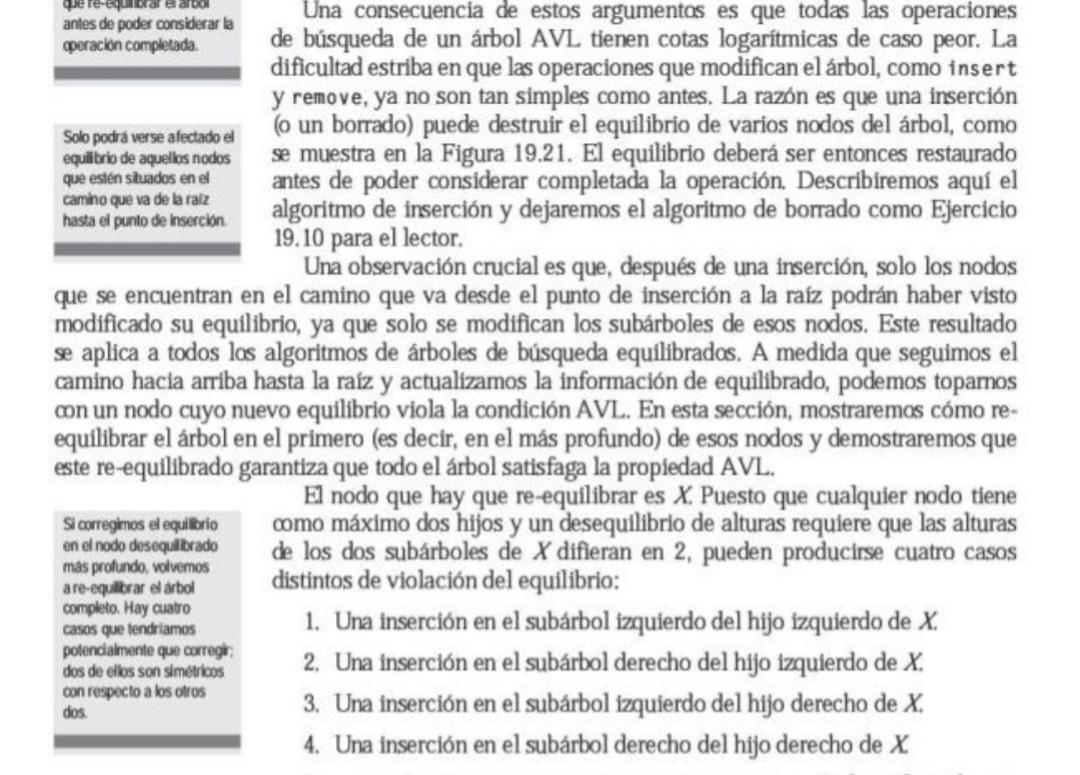
### Arboles AVL



**Un árbol balanceado asegura un orden logaritmo siempre para todas sus operaciones (insercion, eliminacion y búsqueda).**

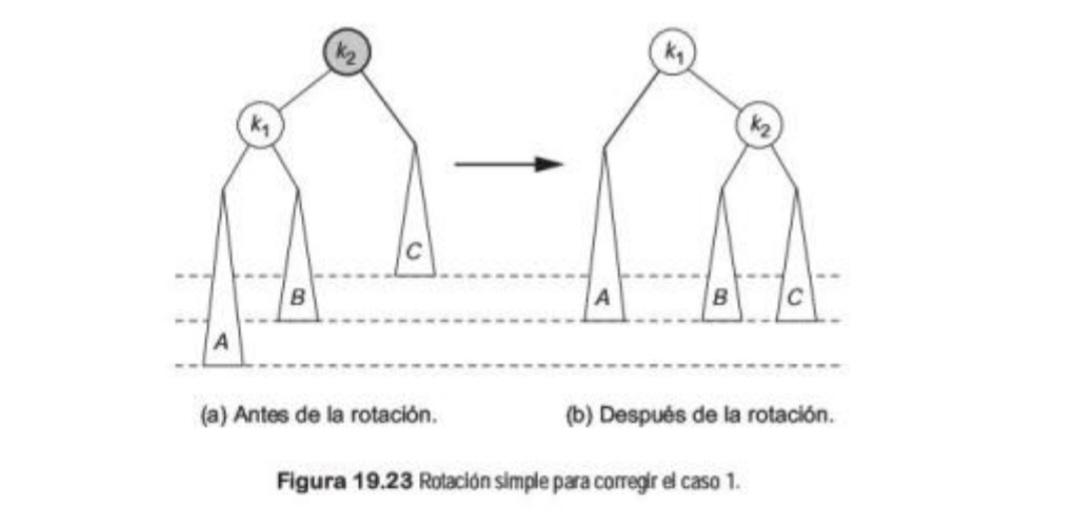


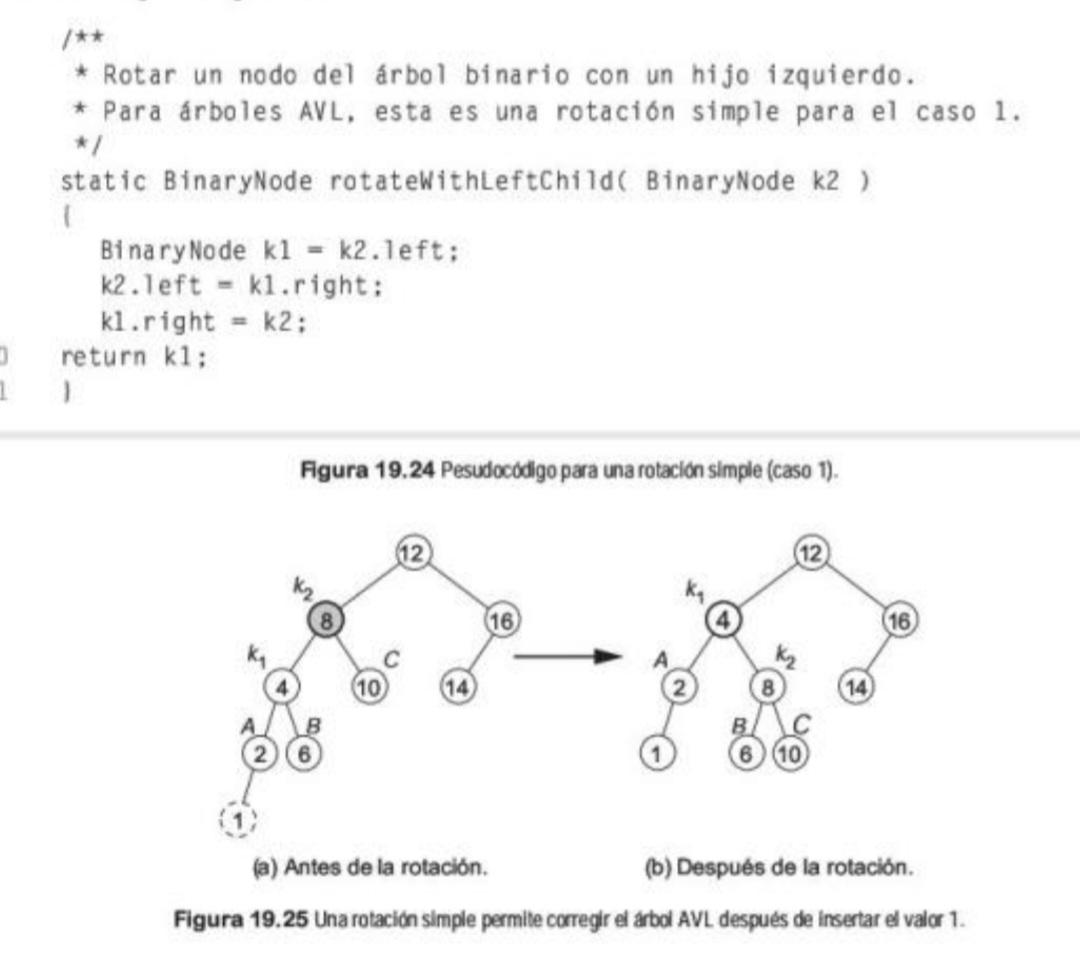
Las operaciones que modifican un árbol, como insert y remove, pueden destruir el equilibrio de varios nodos del árbol. El equilibrio deberá ser restaurado antes de poder considerar completa la operación. Después de una inserción, solo los nodos que se encuentran en el camino que va desde el punto de inserción a la raíz podrán haber visto modificado su equilibrio, ya que solo se modifican los subárboles de esos nodos.

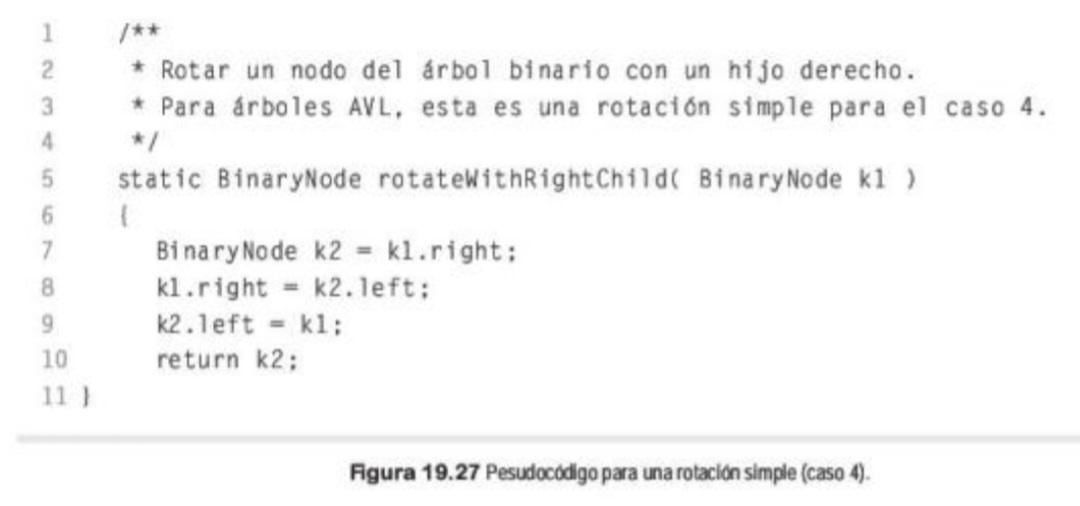


Los casos en que la inserción tiene lugar en el exterior (izq-izq o der-der) se puede corregir con una única **rotación**. Una rotación simple intercambia los papeles del padre y el hijo y mantiene el orden de búsqueda. En el caso de la inserción interior (izq-der o der-izq) se corrige con una **rotación doble**.

#### Rotación Simple







#### Rotación Doble

La rotación simple no funciona para casos de inserción interior, para eso usamos la rotación doble.

