ÁRBOLES BINARIOS DE BÚSQUEDA

Jesús Pérez Melero

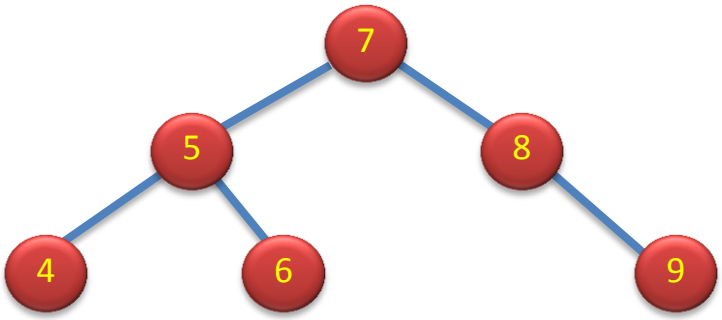
Índice

AVISO: El código se puede optimizar mucho más. Está escrito así para que sea más comprensible.

Introducción a los BST

Los árboles binarios de búsqueda (AAB – BST) son árboles binarios con características especiales. Concretamente, a la **izquierda** de un nodo encontraremos otros nodos cuyo valor será siempre **menor.** También, a la **derecha** de un nodo encontraremos otros nodos cuyo valor será siempre **mayor.**

Ejemplo de BST:



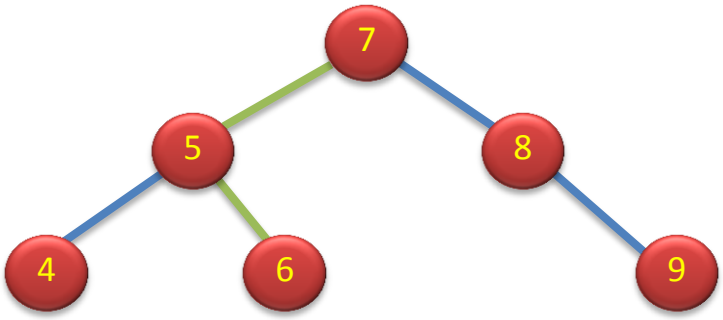
Es interesante ver cómo un **recorrido en inorden** devuelve las claves (valores) de los nodos de **menor a mayor**.

Búsqueda en BST

Para encontrar un elemento en un BST debemos realizar un recorrido **descendente comenzando desde la raíz del árbol.**

Iremos comparando la clave que deseamos obtener con la que hay en el nodo que estemos visitando. Si la clave que buscamos es **menor** que la que estamos viendo, entonces iremos a la **izquierda** del nodo en el que estamos. En el caso contrario, iremos a la derecha.

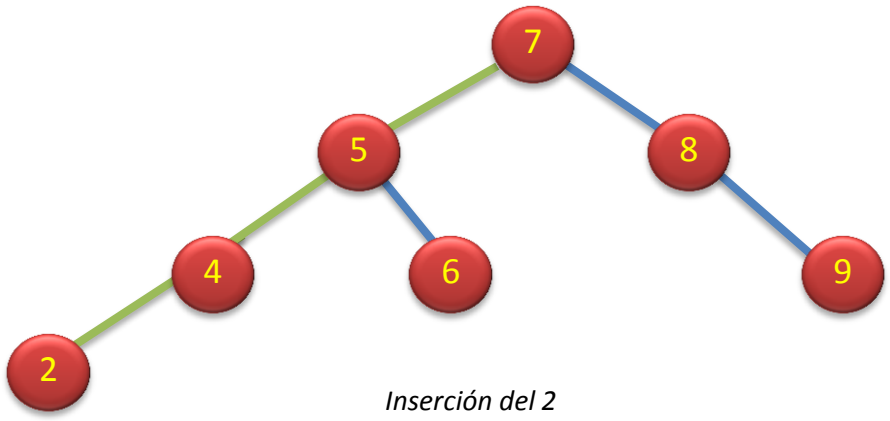
Si llegamos a un nodo hoja cuyo valor no sea el que buscamos, **se termina la búsqueda indicando que no se encontró el elemento.**



Búsqueda del elemento 6

Inserción en BST

Para insertar, **utilizaremos primero la búsqueda** definida anteriormente. Esta búsqueda nos situará en un nodo hoja, y situaremos el elemento como hijo (izquierdo o derecho **según su tamaño**) del nodo encontrado.

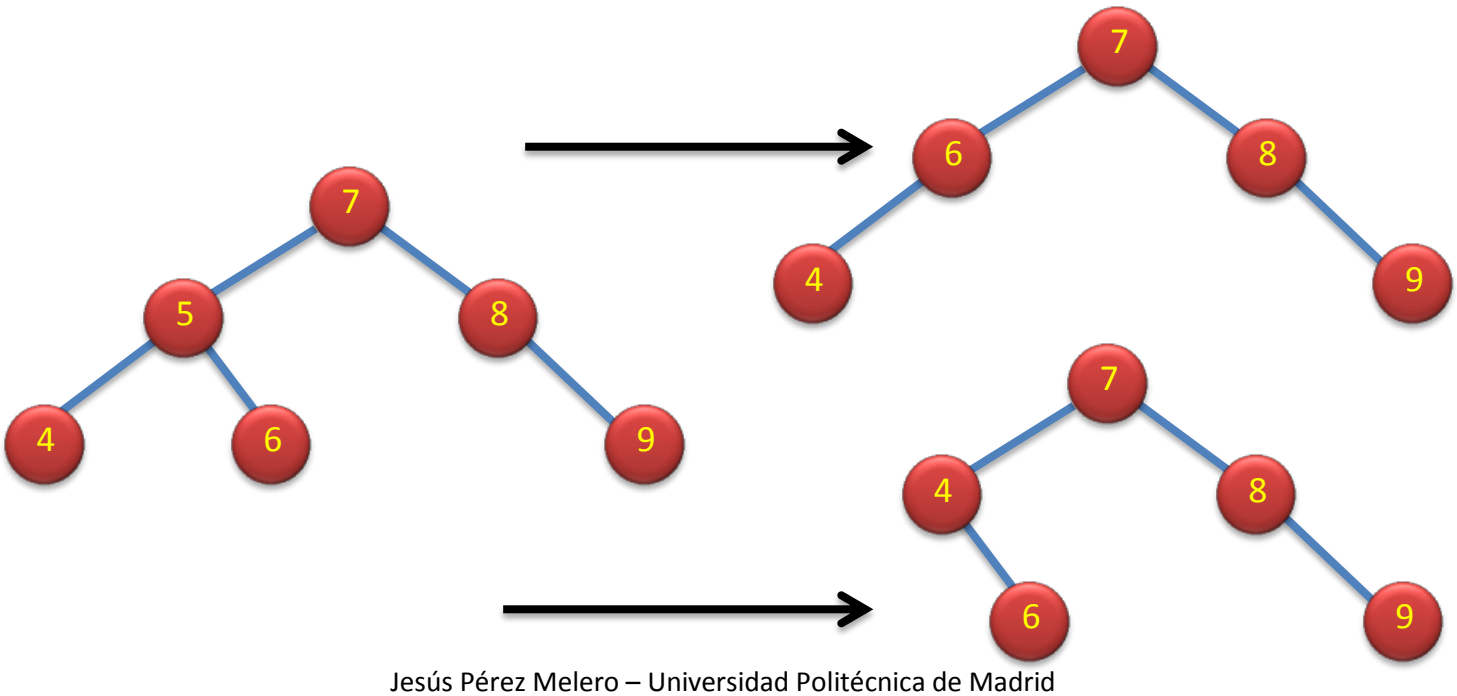


Inserción del 2

Borrado en BST

Para borrar hemos de contemplar varios casos, **tras haber buscado el elemento utilizando la búsqueda.**

- **El nodo no tiene hijos**: Simplemente se borra el nodo y se pone a null la referencia a su padre
- **El nodo tiene un hijo**: Se borra el nodo y se establece como hijo de su padre al hijo del nodo borrado.
- **El nodo tiene dos hijos**: Se reemplaza el valor del nodo por el de su sucesor del recorrido en inorden. (o su predecesor, también es válido) Ejemplo de este último caso, borrando el predecesor y el sucesor en inorden del nodo **5**:



Complejidad y rendimiento

- **Para la búsqueda**, si el árbol binario está en una forma adecuada, la complejidad en el mejor caso puede ser O (log n), sin embargo en el peor caso (árbol estirado totalmente), la similitud entre el árbol y una lista es prácticamente total, por lo que el coste de la búsqueda es lineal respecto de la altura, que coincide con el número de elementos en el peor caso. **Por tanto, O (n).**
- **Para la inserción,** se ha explicado que es necesario buscar la posición en la que irá el nodo antes de insertarlo, por lo que al requerir realizar una búsqueda, la complejidad asciende a orden lineal como mínimo. La acción de crear un nuevo nodo y enlazarlo correctamente conlleva tiempo constante, por lo que la complejidad es **O (n).**
- **Para el borrado,** se realiza un análisis análogo al de la inserción. La complejidad es **O (n).**