Práctica 7 Árboles Binarios de Búsqueda

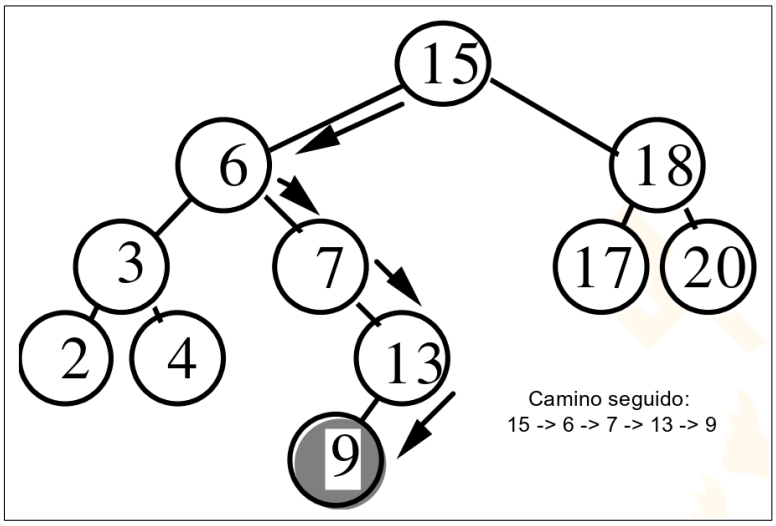
La práctica 7 será necesario realizar un árbol binario de búsqueda.

Para ello, se define una clase Nodo que tendrá los siguientes atributos(las variables en minúsculas):

- **Clave:** Almacena el número.
- **Valor** : Un dato cualquiera asociado a la Clave (por ejemplo, la clave expresada como texto).
- **Padre.-** Indica quién es el nodo padre.
- **Izq.-** Referencia al nodo izquierdo.
- **Der.-** Referencia al nodo izquierdo.

Así mismo, realizar una clase ÁrbolBinario que tendrá un atributo de la clase Nodo, llamado raíz que hará referencia al nodo raíz del árbol

Trabajaremos sober un árbol como el siguiente:



Secuencia sugerida para la práctica:[1]

1. Tras la definición de las clases escribe el ćodigo python que construye una árbol como el indicado en la figura, a base de crear los nodos y conectarlos de forma manual (punteros padres e hijos). Asigna al árbol como nodo raiz el nodo con clave 15 , como se ve en la figura.

2. Escribir un método "**mostrar"** para el árbol binario. Este debe mostrar por pantalla el árbol binario **de la manera más simple posible** tal que se puedan

1 Omite en una primera pasada lo marcado como opcional. No es necesario para la entrega de la práctica, pero se incluye como ejercicio adicional.

identificar perfectamente a los padres y a los hijos de cada nodo. Ejecútalo y **Comprueba tu árbol.** Añade al método la funcionalidad adicional de mostrar la altura del árbol y su equilibrio (Skew).

3. Añade a la clase ArbolBinario un método "**secuenciar**". Éste debe retornar la secuencia (lista) central de claves, op tuplas (clave, valor) almacenadas en árbol. Ejecútalo y comprueba que el resultado es el esperado

   - **OPCIONAL:** añade los métodos **MaxKey** y **MinKey** que toman un nodo como argumento y que buscan la máxima y la mínima clave dentro del sub-arbol definido por el nodo dado y todos sus descendientes.

   - OPCIONAL: Construye lo métodos **next(clave)** y **prev(clave),** según el orden central de secuenciación.

4. Añade un método de "**buscarClave(key)".** Este método ha de buscar una clave dada (key) en el arbol binario, retornado la pareja (clave,valor) en caso de encontrarse en árbol, o **null** en caso contrario. Comprueba el método buscando la clave 13, para obtener **(13, 'trece')** o bien con clave 14 para obtener **null**

5. Añade ahora al árbol binario el método "**insertar(clave, valor)**"- Este método debe crear un nodo e insertarlo en el lugar correcto del nodo, con el método descrito en clase. Añade la pareja (8,"ocho") invocando este nuevo método.

   - Muestra de nuevo el árbol resultante, constatando la ubicación del nuevo nodo, y los nuevos valores de balance y altura. Imprime la nueva secuencia del árbol y comprueba como éste se comporta como una colección ordenada.
   - Comprueba la correcta inserción, invocando el método **buscarClave(8)**
   - Qué pasa si la clave ya existe y el Valor es diferente ?

6. Finalmente añade el metodo **borrar(clave)**.- Este método debe eliminar el primer nodo que encuentre con la clave dada, utilizando la lógica mostrada en clase, retornando True si el borrado es correcto, y False si no lo es. Valida elminando varios tipos de nodos, mostrando el árbol tras cada operación de borrado, para verificar el correcto borrado del nodo con la clave indicada.

   - Nodo hoja (clave=4)
   - Nodo con un solo hijo (clave = 13)
   - Nodo con dos hijos (clave = 6)
   - nodo raíz

Informe de la práctica

1. A través de Moodle de la Asignatura de Algorítmicos y Estructura de Datos. Se deberá entregar el martes 11/04/2023. Es necesario adjuntar a la entrega el notebook o el fichero py.