Estructuras de Datos
Grado en Ingeniería Informática en Sistemas de Información

Enseñanzas Prácticas y de Desarrollo
EPD 7: Árboles Binarios de Búsqueda y AVL

Objetivos

- Conocer el concepto de árbol.
- Implementar y utilizar árboles.

Conceptos

1. Árboles binarios ordenados o de búsqueda

En este tipo de árboles cada nodo tiene asociado un valor, siendo éste mayor o igual que los valores de los nodos de su subárbol izquierdo y menor o igual que todos los de su subárbol derecho. Según el tipo de aplicación puede no permitirse valores iguales en uno o ambos subárboles. Si se realiza un recorrido en ***inorden*** por el árbol accederemos a los valores en orden ascendente. Al definir el tipo de datos que representa el valor de un nodo dentro de un árbol binario de búsqueda es necesario que en dicho tipo se pueda establecer una relación de orden. Una ventaja fundamental de los árboles binarios ordenados, también llamados de búsqueda, es que son en general mucho más rápidos para localizar un elemento que una lista enlazada. Por tanto, son más rápidos para insertar y borrar elementos.

2. Árboles AVL

Básicamente un árbol AVL es un Árbol binario de búsqueda al que se le añade una condición de equilibrio. Esta condición es que para todo nodo la altura de sus subárboles izquierdo y derecho pueden diferir a lo sumo en 1.
La propiedad de equilibrio que debe cumplir un árbol para ser AVL asegura que la profundidad del árbol sea O(log(n)), por lo que las operaciones sobre estas estructuras no deberán recorrer mucho para hallar el elemento deseado. Como se verá, el tiempo de ejecución de las operaciones sobre estos árboles es, a lo sumo O(log(n)) en el peor caso, donde n es la cantidad de elementos del árbol.
Sin embargo, y como era de esperarse, esta misma propiedad de equilibrio de los árboles AVL implica una dificultad a la hora de insertar o eliminar elementos: estas operaciones pueden no conservar dicha propiedad.

Para insertar o eliminar un elemento, se sigue una estrategia de busca y repara. Primero se inserta/elimina un nodo de la misma forma que en un ABB, y luego se ejecutan los cambios necesarios para restablecer la condición de equilibrio. Básicamente se trata de ejecutar una o varias rotaciones. Para los detalles puede consultar el material de EB.

Bibliografía Básica

Documentación de la API de Java:

- *Introduction to Algorithms*. Thomas H. Cormen. Capítulo 12, Página 253.

Experimentos

E1. (5 mins.) Analice la siguiente interfaz correspondiente a un árbol binario de búsqueda, donde la interfaz BinaryTree es la EPD anterior:

```
public interface BinarySearchTree extends BinaryTree {
       void addElement(Object element);
       Object removeElement(Object element);
       void removeAllOcurrences(Object element);
       Object removeMin();
```

```
        Object removeMax();
        Object findMin();
        Object findMax();
        boolean find(Object elementToFind);
    }
```

Ejercicios

EJ1. (90 mins.) Implementar un árbol binario de búsqueda (ABB) que implemente la interfaz `BinarySearchTree`. Tenga en cuenta que un ABB es una extensión del árbol binario desarrollado en el EPD anterior. Se recomienda que la clase implementada tenga un comparador como atributo, inicializándolo en el constructor. Este comparador se utilizará para poder ordenar los nodos y nos permitirá que contengan cualquier tipo de elementos.

Problemas

P1. (180 mins.) Implementar un árbol AVL. Tenga en cuenta que un árbol AVL es una extensión del árbol binario de búsqueda desarrollado en el Ejercicio 1.

Los árboles **AVL** fueron creados en 1962 por los matemáticos *Georgi **A**delson-**V**elskii* y *Yevgeni **L**andis* del *Instituto de Física Teórica y Experimental de Moscú* y representan un tipo especial de árbol binario ordenado.

Los árboles AVL tiene la peculiaridad de que siempre están equilibrados, por lo que se garantiza que cualquier búsqueda que se realice sobre ellos tendrá una complejidad de O(log n), siendo *n* el número de nodos del árbol. Para conseguir que en todo momento se respete el equilibrio del árbol AVL, se almacenará en cada uno de los nodos un valor entero llamado *factor de equilibrio*, que es la diferencia entre la altura de su subárbol derecho menos la altura de su subárbol izquierdo. Si en un determinado nodo el *factor de equilibrio* es inferior a -1 o superior a 1, entonces el subárbol que tiene a ese nodo como raíz necesitará ser balanceado para seguir garantizando la propiedad básica de equilibrio de los árboles AVL.

Un árbol AVL solo puede desequilibrarse cuando se inserta un nuevo elemento en el árbol o se elimina un nodo existente. Así, cada vez que se realiza una de estas operaciones, es necesario actualizar los *factores de equilibrio* y comprobar el equilibro del árbol, comenzando por el punto de inserción o eliminación de un nodo y subiendo hacia la raíz del árbol. En el caso de encontrar algún nodo cuyo *factor de equilibrio* supere los umbrales descritos anteriormente, habrá que balancear el subárbol cuya raíz es el nodo donde se ha detectado que el *factor de equilibrio* no cumple con lo establecido para este tipo de árboles. A continuación se describen las posibles rotaciones al balancear un árbol para recuperar el equilibrio perdido:

Rotación a la derecha de un árbol AVL:
Si el *factor de equilibrio* de un nodo es menor que -1, entonces su subárbol izquierdo presenta una ruta demasiado larga. En este caso se comprobará el *factor de equilibrio* del hijo izquierdo del nodo original. Si dicho valor es -1, entonces la ruta de gran longitud se encuentra en el subárbol izquierdo del hijo izquierdo y que, por tanto, una simple rotación a la derecha del hijo izquierdo alrededor del nodo original permitirá reequilibrar el árbol.

Rotación a la izquierda de un árbol AVL:
Si el *factor de equilibrio* de un nodo es mayor que 1, entonces su subárbol derecho presenta una ruta demasiado larga. En este caso se comprobará el *factor de equilibrio* del hijo derecho del nodo original. Si dicho valor es 1, entonces la ruta de gran longitud se encuentra en el subárbol derecho del hijo derecho y que, por tanto, una simple rotación a la izquierda del hijo derecho alrededor del nodo original permitirá reequilibrar el árbol.

Rotación a la derecha-izquierda de un árbol AVL:
Si el *factor de equilibrio* de un nodo es mayor que 1, entonces su subárbol derecho presenta una ruta demasiado larga. En este caso se comprobará el *factor de equilibrio* del hijo derecho del nodo original. Si dicho valor es -1, entonces la ruta de gran longitud se encuentra en el subárbol izquierdo del hijo derecho y que, por tanto, una rotación doble derecha-izquierda permitirá reequilibrar el árbol. Esto se lleva a cabo realizando primero una rotación a la derecha del hijo izquierdo del hijo derecho del nodo original, alrededor del hijo derecho del nodo original, y luego realizando una rotación a la izquierda del hijo derecho del nodo original, alrededor del nodo original.

Rotación a la izquierda-derecha de un árbol AVL:
Si el *factor de equilibrio* de un nodo es menor que -1, entonces su subárbol izquierdo presenta una ruta demasiado larga. En este caso se comprobará el *factor de equilibrio* del hijo izquierdo del nodo original. Si dicho valor es 1, entonces la ruta de gran longitud se encuentra en el subárbol derecho del hijo izquierdo y que, por tanto, una rotación doble izquierda-derecha permitirá reequilibrar el árbol. Esto se lleva a cabo realizando primero una rotación a la izquierda del hijo derecho del hijo izquierdo del nodo original, alrededor del hijo izquierdo del nodo original, y luego realizando una rotación a la derecha del hijo izquierdo del nodo original, alrededor del nodo original.

Ejemplos:

A continuación se describe una operación de inserción que requiere una rotación a la derecha para reequilibrar el árbol. En la figura 1 podemos ver el árbol AVL inicial de números enteros con los *factores de equilibrio* de cada nodo entre paréntesis. En la figura 2 se ve la inserción de un nuevo nodo de valor 1 en el árbol AVL con los *factores de equilibrio* de cada nodo actualizados. Se ve claramente que el *factor de equilibrio* del nodo raíz está por debajo del valor umbral permitido, por lo que habrá que realizar una rotación a la derecha para reequilibrar el árbol AVL, tal como se ve en la figura 3.

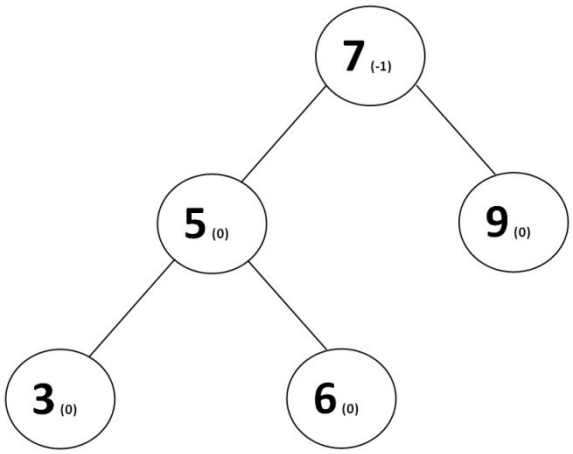

Figura 1. Árbol AVL inicial

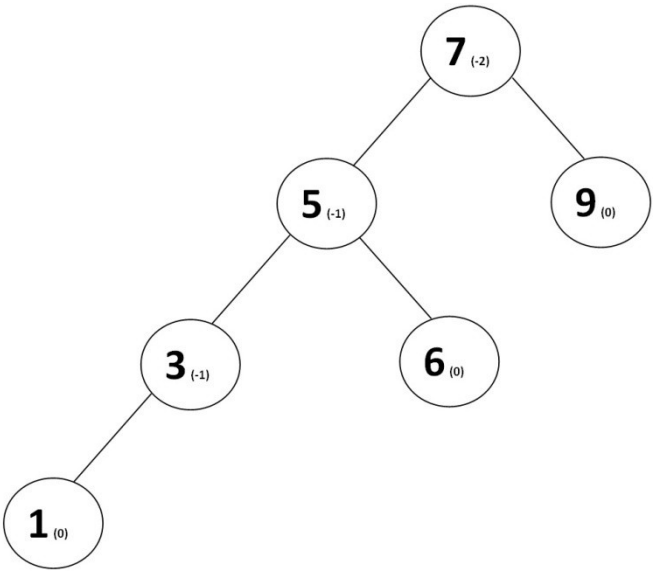

Figura 2. Inserción del nodo 1 con factores de equilibrio actualizados

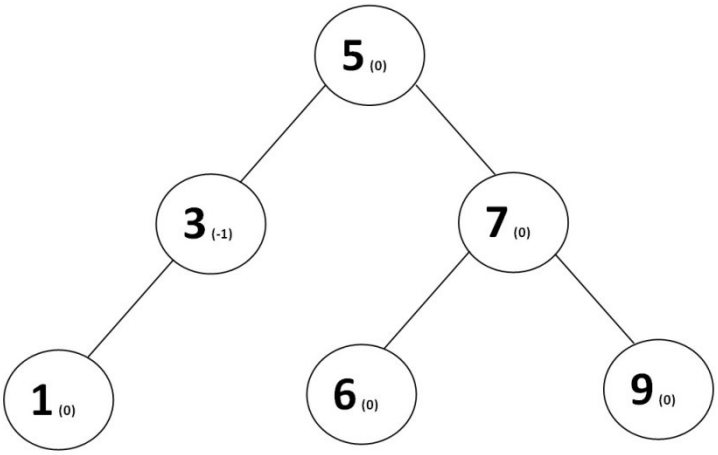

Figura 3. Rotación a la derecha con factores de equilibrio actualizados

A continuación se describe una operación de borrado que requiere una rotación derecha-izquierda para reequilibrar el árbol. En la figura 4 podemos ver el árbol AVL inicial de números enteros con los *factores de equilibrio* de cada nodo entre paréntesis. En la figura 5 se ve la eliminación del nodo de valor 1 del árbol AVL con los *factores de equilibrio* de cada nodo actualizados. Se ve claramente que el *factor de equilibrio* del nodo raíz supera el valor umbral permitido, por lo que habrá que realizar una rotación derecha-izquierda para reequilibrar el árbol AVL, tal como se ve en las figura 6 y 7.

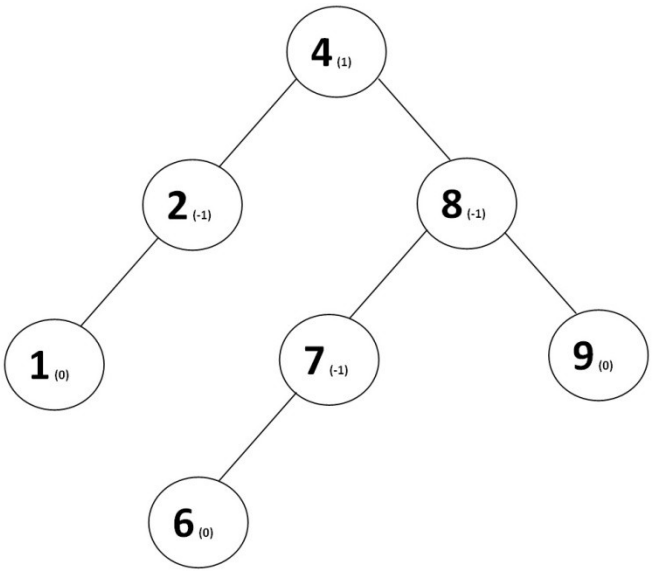

Figura 4. Árbol AVL inicial

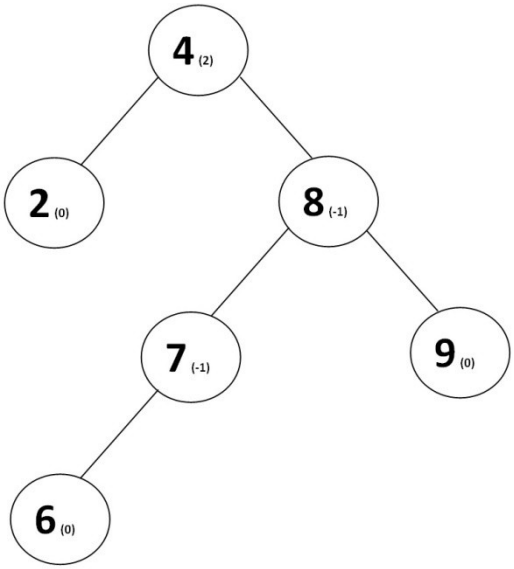

Figura 5. Eliminación del nodo 1 con factores de equilibrio actualizados

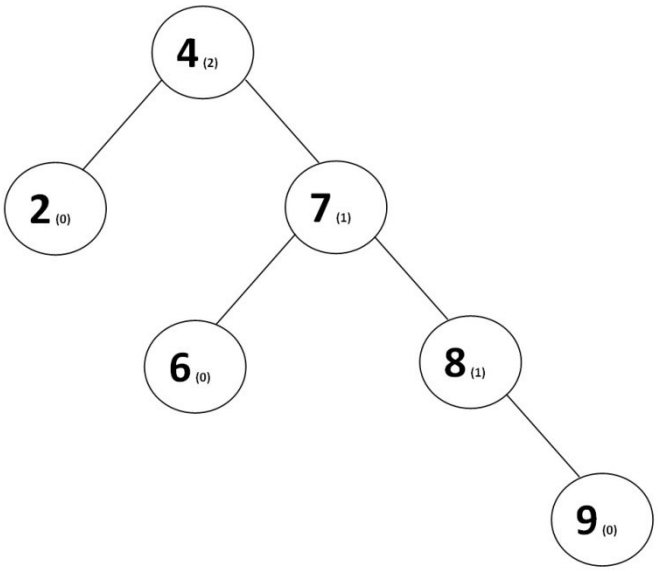

Figura 6. Rotación a la derecha con factores de equilibrio actualizados

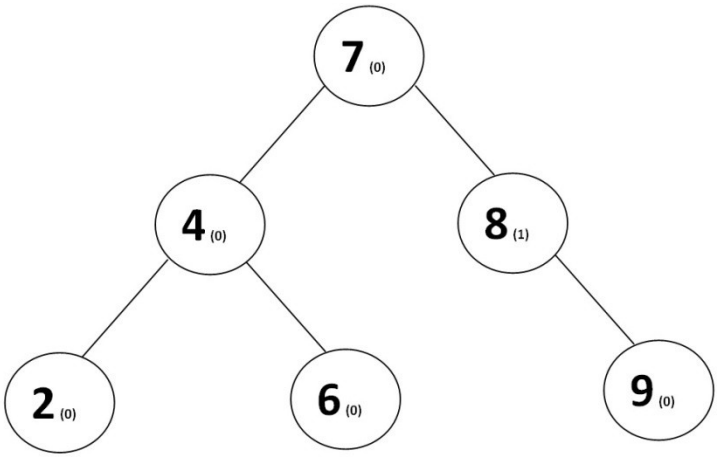

Figura 7. Rotación a la izquierda con factores de equilibrio actualizados

La rotación simple a la izquierda y la rotación doble izquierda-derecha son totalmente simétricas a las explicadas en estos dos ejemplos.