EjercicioAVL

Estructuras de Datos

Tema 4: árboles

1º Grado en Ingeniería de la Computación
© Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)
URJC (www.urjc.es)

Programa en C++ la clase “AVL” (no admite elementos repetidos) con las siguientes características, aprovechando todo lo que puedas de la actividad EjercicioABB2:

- La estructura Nodo será la misma que en EjercicioABB2, pero añadiendo un atributo para guardar la altura del nodo. Recuerda que la altura de una hoja es 0. Consideremos que la altura de un árbol vacío es -1. Para facilitar la programación, puedes hacer que Nodo sea una clase con varios métodos (en vez de un struct), como por ejemplo calcular el factor actual de equilibrio de un nodo. En el caso de que Nodo sea una clase en vez de un struct, habrá que añadir métodos públicos para obtener y modificar los diferentes atributos de la clase.

- La clase AVL tendrá al menos lo siguiente:

    - Atributos: el tamaño del árbol (“n”) y el puntero al nodo raíz (que será NULL si el árbol está vacío).
    - Tendremos un constructor sin parámetros para crear un árbol AVL vacío.
    - Tendremos un método privado llamado “insertarHoja()”, que será muy parecido al método público “insertar()” de EjercicioABB2. La única diferencia es que devolverá el puntero al nodo hoja añadido en el último nivel del árbol.
    - Tendremos un método privado llamado “reequilibrar()”, al que habrá que llamar después de "insertarHoja()". Recibirá como único parámetro el puntero a un nodo hoja. A partir de dicho nodo, irá ascendiendo por el árbol, actualizando la altura de cada nodo y averiguando su factor de equilibrio, y rotando en caso necesario. Seguiremos ascendiendo y haciendo esto hasta llegar a la raíz del árbol.
    - Para que el método “reequilibrar()” pueda llamarlos, tendremos también que programar privadamente los cuatro métodos que implementan cada una de las cuatro rotaciones. Cada uno de esos métodos recibirá como único parámetro el puntero al nodo raíz del subárbol que queremos rotar, y devolverá el puntero a la nueva raíz del subárbol que ha quedado después de la rotación. Cada uno de esos cuatro métodos deberá también actualizar la altura de todos los nodos relevantes del subárbol que se está rotando. Los nodos que pueden cambiar su altura después de hacer la rotación fíjate en que son “x”, “y” y “z” (este último sólo existe en las rotaciones compuestas).
    - Para facilitar la programación interna de las cuatro rotaciones, programa un método privado llamado "cambiarHijo()" que reciba dos parámetros: el puntero al nodo que contiene el antiguo hijo, y el puntero al nodo que contiene el nuevo hijo. Hará que el padre de antiguoHijo apunte a nuevoHijo en vez de a antiguoHijo. Es decir, se cambia el antiguoHijo por el nuevoHijo.

- Tendremos un método público llamado “insertar()”, que, al igual que el método “insertar()” de EjercicioABB2, recibirá como único parámetro el elemento a insertar, y no devolverá nada. La precondición es que el nuevo elemento no exista previamente en el árbol. Este método por dentro llamará al método “insertarHoja” y luego reequilibrará el árbol para que éste vuelva a ser AVL, llamando al método “reequilibrar()”.
- Tendremos también los métodos (público y privado) para imprimir el árbol en forma de esquema sangrado con tabulaciones. Estos métodos son iguales que en EjercicioABB2.
- Tendremos también los dos métodos necesarios para buscar un elemento en el árbol: el público y el privado recursivo. Serán los mismos que en EjercicioABB2. El privado recursivo, además, lo llamaremos dentro de "insertarHoja()", con el fin de localizar el hueco en donde meter dicha hoja.
- Tendremos también el mismo destructor (y su respectivo método privado recursivo) que en EjercicioABB2.

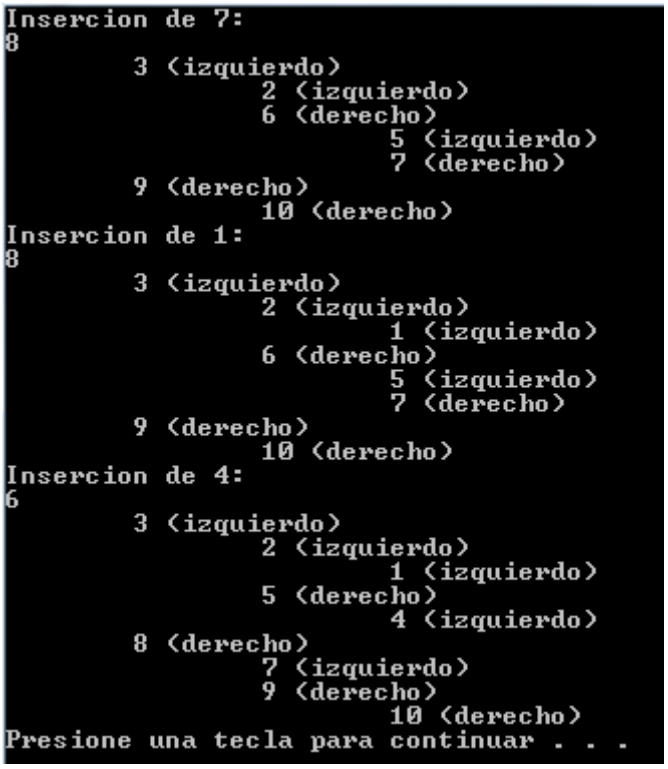
- Idea un main que pruebe suficientemente el programa con las cuatro posibles rotaciones, insertando nodo a nodo e imprimiendo justo después de cada una de las inserciones. Un posible main podría tener el siguiente resultado:

```
Probamos la rotacion simple a la derecha
Insercion de 6:
6
Insercion de 5:
6
        5 (izquierdo)
Insercion de 4:
5
        4 (izquierdo)
        6 (derecho)
Insercion de 3:
5
        4 (izquierdo)
                3 (izquierdo)
        6 (derecho)
Insercion de 2:
5
        3 (izquierdo)
                2 (izquierdo)
                4 (derecho)
        6 (derecho)
Insercion de 1:
3
        2 (izquierdo)
                1 (izquierdo)
        5 (derecho)
                4 (izquierdo)
                6 (derecho)

Probamos la rotacion compuesta derecha-izquierda
Insercion de 2:
2
Insercion de 5:
2
        5 (derecho)
Insercion de 3:
3
        2 (izquierdo)
        5 (derecho)
Insercion de 9:
3
        2 (izquierdo)
        5 (derecho)
                9 (derecho)
Insercion de 8:
3
        2 (izquierdo)
        8 (derecho)
                5 (izquierdo)
                9 (derecho)
Insercion de 1:
3
        2 (izquierdo)
                1 (izquierdo)
        8 (derecho)
                5 (izquierdo)
                9 (derecho)
Insercion de 10:
3
        2 (izquierdo)
                1 (izquierdo)
        8 (derecho)
                5 (izquierdo)
                9 (derecho)
                        10 (derecho)
```

```
Insercion de 4:
3
        2 (izquierdo)
                1 (izquierdo)
        8 (derecho)
                5 (izquierdo)
                        4 (izquierdo)
                9 (derecho)
                        10 (derecho)
Insercion de 6:
3
        2 (izquierdo)
                1 (izquierdo)
        8 (derecho)
                5 (izquierdo)
                        4 (izquierdo)
                        6 (derecho)
                9 (derecho)
                        10 (derecho)
Insercion de 7:
5
        3 (izquierdo)
                2 (izquierdo)
                        1 (izquierdo)
                4 (derecho)
        8 (derecho)
                6 (izquierdo)
                        7 (derecho)
                9 (derecho)
                        10 (derecho)

Probamos la rotacion compuesta izquierda-derecha
Insercion de 9:
9
Insercion de 6:
9
        6 (izquierdo)
Insercion de 8:
8
        6 (izquierdo)
        9 (derecho)
Insercion de 2:
8
        6 (izquierdo)
                2 (izquierdo)
        9 (derecho)
Insercion de 3:
8
        3 (izquierdo)
                2 (izquierdo)
                6 (derecho)
        9 (derecho)
Insercion de 10:
8
        3 (izquierdo)
                2 (izquierdo)
                6 (derecho)
        9 (derecho)
                10 (derecho)
Insercion de 5:
8
        3 (izquierdo)
                2 (izquierdo)
                6 (derecho)
                        5 (izquierdo)
        9 (derecho)
                10 (derecho)
```


Insercion de 7:
8
3 (izquierdo)
2 (izquierdo)
6 (derecho)
5 (izquierdo)
7 (derecho)
9 (derecho)
10 (derecho)
Insercion de 1:
8
3 (izquierdo)
2 (izquierdo)
1 (izquierdo)
6 (derecho)
5 (izquierdo)
7 (derecho)
9 (derecho)
10 (derecho)
Insercion de 4:
6
3 (izquierdo)
2 (izquierdo)
1 (izquierdo)
5 (derecho)
4 (izquierdo)
8 (derecho)
7 (izquierdo)
9 (derecho)
10 (derecho)
Presione una tecla para continuar . . .