EJERCICIOS

15.1. Dibujar el árbol binario de búsqueda equilibrado que resulta de aplicar la operación de insertar con las claves: `14, 6, 24, 35, 59, 17, 21, 32, 4, 7, 15` y `22`.

15.2. Dada la secuencia de claves enteras: `100, 29, 71, 82, 48 39, 101, 22, 46, 17, 3, 20, 25, 10`. Dibujar el árbol AVL correspondiente. Eliminar claves consecutivamente hasta encontrar un nodo en el que se *viole* la condición de equilibrio cuya restauración sea con una rotación doble.

15.3. En el árbol construido en el Ejercicio 15.1, eliminar el nodo raíz. Hacerlo tantas veces como sea necesario hasta que se desequilibre un nodo y haya que aplicar una rotación simple.

15.4. Encontrar una secuencia de n claves que al ser insertadas en un árbol binario de búsqueda vacío se apliquen las cuatro rutinas de rotación: `II, ID, DD, DI`.

15.5. Dibujar el árbol equilibrado después de insertar en orden creciente 31 ($2^5 - 1$) elementos del 11 al 46.

15.6. ¿Cuál es el número mínimo de nodos de un árbol binario de búsqueda equilibrado de altura 10?

15.7. Escribir la función recursiva `buscarMin()` que devuelva la clave mínima de un árbol de búsqueda equilibrado.

15.8. Dibujar un árbol AVL de altura 6 con el criterio del *peor de los casos*; es decir, cada nodo tiene como factor de equilibrio ±1.

15.9. En el árbol equilibrado formado en el Ejercicio 15.8 eliminar una de las hojas menos profundas. Representar las operaciones necesarias para restablecer el equilibrio.

15.10. Escribir la función recursiva `buscarMax()` que devuelva la clave máxima de un árbol de búsqueda equilibrado.

15.11. Escribir la función `buscarMin()` y `buscarMax()` en un árbol de búsqueda equilibrado de manera iterativa.

PROBLEMAS

15.1. Leer un archivo de texto y almacenar en memoria, todas las palabras de dicho texto y su frecuencia. La estructura de datos debe ser tal que permita realizar una búsqueda en un tiempo `O(logn)`, sin que dependa de la entrada de datos, por lo que se requiere utilizar un árbol AVL. El archivo de texto se llama *carta.dat*, se quiere almacenar las palabras del archivo en memoria utilizando la estructura de árbol indicada.

15.2. Añadir al programa escrito en el Problema 15.1 una función con el fin de que dada una palabra se obtenga el número de veces que aparece en el texto.

15.3. En el archivo *alumnos.txt* se encuentran los nombres completos de los alumnos de las escuelas de taller de la Comunidad Alcarreña. Escribir un programa para leer el archivo y formar inicialmente un árbol de búsqueda con respecto a la clave *apellido*. Una vez formado el árbol, construir con sus nodos un árbol de Fibonacci.

15.4. La implementación de la operación *insertar* en un árbol equilibrado se realiza de manera natural en forma recursiva. Escribir de nuevo la codificación aplicando una estrategia iterativa.