# Métodos Algorítmicos en Resolución de Problemas I

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas

Hoja de ejercicios 2 Curso 2024–2025

---

## Ejercicios de árboles de búsqueda avanzados

**Ejercicio 1** Mostrar con dibujos los resultados de insertar consecutivamente 7, 3, 1, 6, 2, 4, 8 y 5 en un árbol AVL inicialmente vacío.

**Ejercicio 2** Argumentar que en los árboles AVL el número de nodos en el hijo izquierdo puede ser arbitrariamente grande con respecto del número de nodos en el derecho para alturas suficientemente grandes.

**Ejercicio 3** Demostrar, que si se produce un desequilibrio LL (respectivamente, RR) durante la ejecución de *insertar*, entonces nunca se produce la situación $h_{ii} = h_{id}$ (respectivamente, $h_{dd} = h_{di}$). Como corolario, demostrar que a lo sumo una de las llamadas a *equil* en el camino de inserción produce una rotación.

**Ejercicio 4** Idear un árbol AVL y determinar un nodo del mismo tales que, al borrar dicho nodo, el árbol necesite tres rotaciones consecutivas para equilibrarse. Las llamadas a veces "rotaciones dobles" —LR y RL— cuentan como una. Ilustrar con suficientes dibujos la secuencia de borrado, indicando el tipo de rotaciones involucradas.

A partir del ejemplo anterior, indica cómo obtendrías un ejemplo general en el que sean necesarias $\frac{h-1}{2}$ rotaciones, siendo $h$ la altura inicial del árbol.

**Ejercicio 5** Supongamos que en los árboles AVL añadimos a cada nodo un campo $n_{izq}$ con el cardinal del hijo izquierdo de dicho nodo.

1. Modificar la operación *insertar* para que preserve este invariante y siga teniendo coste logarítmico.
2. Modificar la operación *borrar* para que preserve este invariante y siga teniendo coste logarítmico.
3. Programar una operación $buscar(t, k)$ con coste logarítmico que devuelva el $k$-ésimo menor elemento del AVL $t$.
4. Programar una operación $borrar(t, k)$ con coste logarítmico que borre del AVL $t$ el $k$-ésimo menor elemento.

**Ejercicio 6** Los AVL pueden considerarse una implementación de conjuntos de elementos. Programar una función que, dados dos AVL, devuelva un AVL que represente el conjunto unión de los dos. Calcular el coste de la misma.

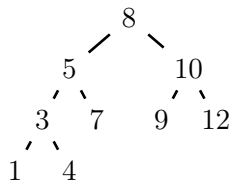
**Ejercicio 7** Idear una representación del tipo de datos *conjunto* de elementos (dotados de la operación $\leq$), que soporte las operaciones habituales y una operación adicional $sig(x, C)$, que dados un $x$ arbitrario y un conjunto $C$, devuelva el menor elemento de $C$ que sea mayor que $x$, o dé un error, si tal elemento no existe. Implementar en detalle dicha operación *sig*. El coste de todas las operaciones para un conjunto de cardinal $n$ ha de estar en $O(\log n)$.

**Ejercicio 8** Suponemos el tipo *conjunto* de elementos —provistos estos últimos de la operación $\leq$ y de dos constantes *minE* y *maxE* que son respectivamente el mínimo y el máximo del dominio— implementado mediante un árbol de búsqueda equilibrado, que puede ser indistintamente AVL, rojinegro, u otra variante igualmente equilibrada.

Se pide implementar una nueva operación *intervalo*$(z, t)$, que dados un $z$ arbitrario y un árbol $t$, devuelva dos elementos $(a, b)$, tales que $a \leq z \leq b$. Si $z$ pertenece al conjunto, ha de devolver $(z, z)$. En caso contrario, $a$ ha de ser el mayor elemento de $t$ tal que $a \leq z$, o bien *minE* si tal elemento no existe. Simétricamente, $b$ ha de ser el menor elemento de $t$ tal que $z \leq b$, o bien *maxE* si tal elemento no existe.

Si $n$ es el cardinal de $t$, el coste de *intervalo*$(z, t)$ ha de ser logarítmico en $n$.

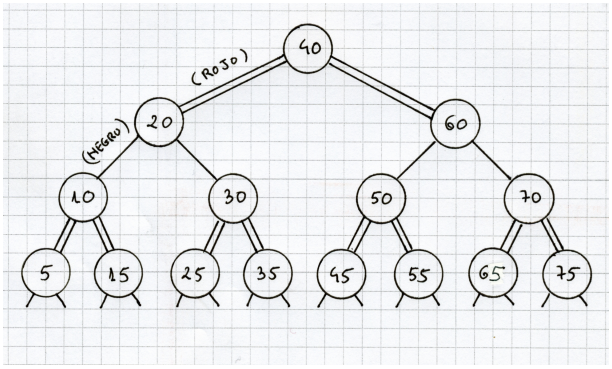
**Ejercicio 9** Dadas dos claves $x$ e $y$ pertenecientes a un árbol de búsqueda equilibrado, llamamos el *ancestro común más cercano* (ACMC) de dichas claves a una clave $z$ cuyo nodo es ancestro de ambas en el árbol y tal que cualquier otro ancestro común está más arriba. Por ejemplo, en el siguiente árbol




el ACMC de las claves 1 y 4 es 3 y el ACMC de 4 y 5 es 5. Se pide escribir un algoritmo que, dados un árbol de búsqueda $t$ y dos claves $x$ e $y$ del mismo, devuelva el ACMC de dichas claves en $t$. Justificar su corrección y dar su coste.

**Ejercicio 10** Mostrar el resultado de insertar 7, 3, 1, 6, 2, 4, 8 y 5 en un árbol rojinegro inclinado a la izquierda (un LLRB de R. Sedgewick) inicialmente vacío.

**Ejercicio 11** Dado el árbol rojinegro inclinado a la izquierda de la siguiente figura:



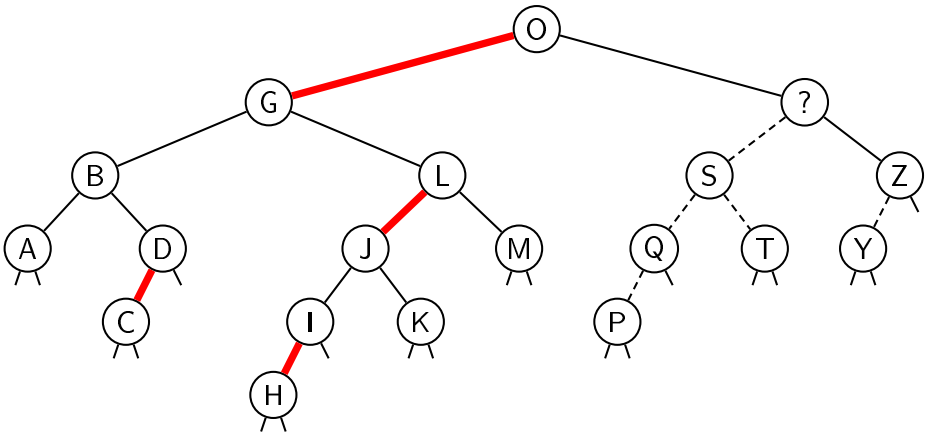

detallar con suficientes dibujos las transformaciones que sufre durante la inserción de la clave 80. Dibuja también el árbol final resultante como árbol 2-3-4.

**Ejercicio 12** Demostrar que la longitud del camino más largo desde un nodo hasta una hoja en un árbol rojinegro estándar es como mucho el doble más uno de la longitud del camino más corto desde ese nodo hasta una hoja.

**Ejercicio 13** ¿Cuál es el mayor número posible de nodos en un árbol rojinegro estándar con altura negra $k$? ¿Cuál es el menor valor posible?

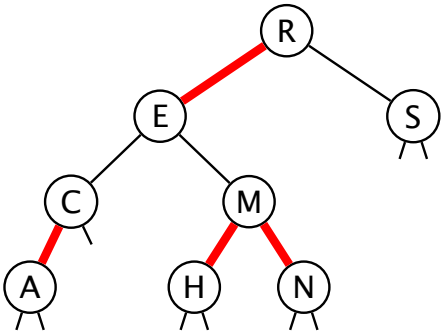
**Ejercicio 14** Dibujar todos los árboles rojinegros estándar estructuralmente diferentes con $n$ valores, para $n$ desde 1 hasta 6.

**Ejercicio 15** Considera el siguiente ABB rojinegro inclinado a la izquierda, donde se ha ocultado una clave y el color de algunas aristas:




1. ¿Cuáles de las siguientes claves podrían ser la clave oculta (marcada con una interrogación)?
   A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z
2. ¿Cuáles de las siguientes claves deben ser rojas (es decir, es roja la arista que la une a su padre)? P Q S T Y
3. Ilustrar con suficientes dibujos los árboles intermedios y finales que resultan de insertar sucesivamente las claves E, N, F en el árbol anterior.

**Ejercicio 16** Considera el siguiente árbol rojinegro inclinado a la izquierda (LLRB):




Explicar con suficientes dibujos el proceso de inserción de la clave P, seguida de la inserción de la clave O.

**Ejercicio 17** Idear un árbol LLRB tal que, al ser insertada una nueva clave, el árbol sufra —en ese orden— una transformación *color-flip*, una *rotate-left* y otra *rotate-right*. Ilustrar con suficientes dibujos la secuencia de inserción y justificar cada paso con una breve frase.

**Ejercicio 18** Supongamos que tenemos un tipo de datos *conjunto* implementado como un árbol de búsqueda equilibrado de cualquiera de las variedades vistas (AVL, rojinegros, 2-3, etc.). Se desea implementar un procedimiento *diferencia*$(S, T)$ que tome el conjunto $S$ y elimine todos los elementos de $T$ del mismo. Suponiendo que $s = |S|$ y $t = |T|$, sean $M = \max(s, t)$ y $m = \min(s, t)$. El tiempo de ejecución del procedimiento *diferencia*$(S, T)$ tiene que estar en $O(m \log M)$. Suponemos que el cardinal de un conjunto se puede conseguir con una operación de coste en $O(1)$.