

Métodos Algorítmicos en Resolución de Problemas I

Grado en Ingeniería Informática

Hoja de ejercicios 2

Curso 2020-2021

EJERCICIOS DE ÁRBOLES DE BÚSQUEDA AVANZADOS

Ejercicio 1 Escribir un algoritmo recursivo de coste lineal en el cardinal del árbol que compruebe si un árbol binario está equilibrado en altura.

Ejercicio 2 Se dice que un árbol binario es *zurdo* si o bien (i) es el árbol vacío o es una hoja, o bien (ii) sus hijos izquierdo y derecho son ambos zurdos y más de la mitad de sus descendientes son descendientes izquierdos. Desarrollar un algoritmo recursivo lineal para decidir si un árbol binario es o no zurdo.

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

Ejercicio 4 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 5 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 6 Encontrar un ejemplo en el que al borrar un nodo se produzca más de una rotación en cascada en el camino de borrado (*ayuda*: investigar árboles AVL con pocos elementos).

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 7 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 8 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 9 Programar una función *partir*(x, t) que, dado un AVL t y una clave x , devuelva dos AVL, uno conteniendo las claves de t menores o iguales que x , y otro conteniendo las mayores estrictas. El coste en espacio y en tiempo ha de ser menor que $O(n)$ (*ayuda*: hacer primero un algoritmo recursivo que consiga un árbol de búsqueda, y luego otro que lo equilibre).

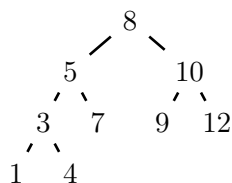
Ejercicio 10 Idear una representación del tipo de datos *conjunto* de elementos (dotados de la operación \leq), que soporte las operaciones habituales y una operación adicional $\text{sig}(x, C)$, que dados un x arbitrario y un conjunto C , devuelva el menor elemento de C que sea mayor que x , o dé un 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 11 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 $\text{intervalo}(z, t)$, 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 $\text{intervalo}(z, t)$ ha de ser logarítmico en n .

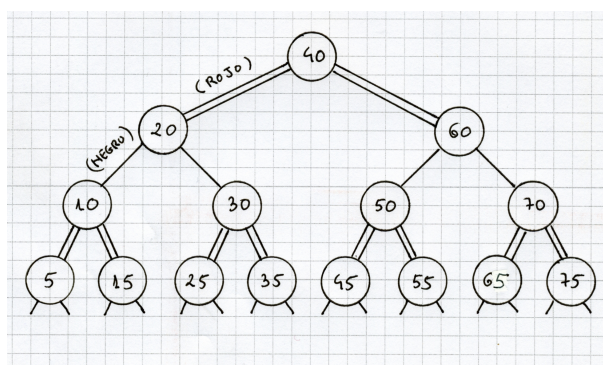
Ejercicio 12 Dadas dos claves x e y pertenecientes a un árbol de búsqueda equilibrado, llamamos el *ancestro común más cercano* (ACMC) de dichas claves a una clave z cuyo nodo es ancestro de ambas en el árbol y tal que cualquier otro ancestro común está más arriba. Por ejemplo, en el siguiente árbol



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 13 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. Sedgwick) inicialmente vacío.

Ejercicio 14 Dado el árbol rojinegro inclinado a la izquierda de la siguiente figura:



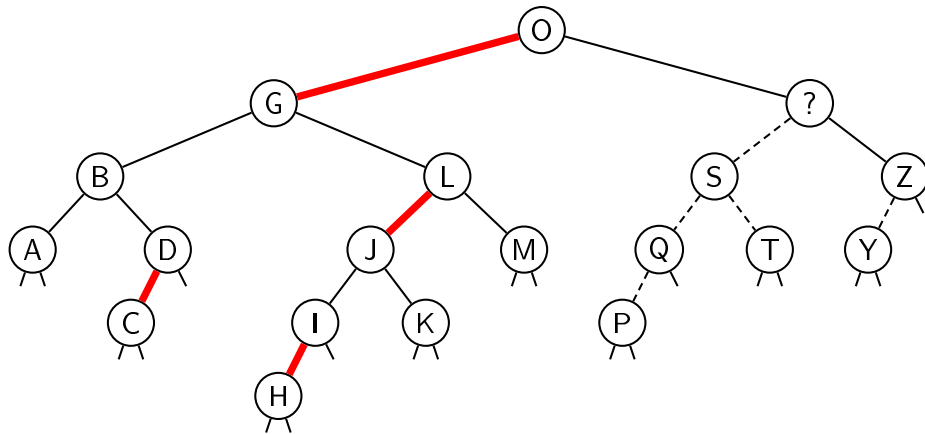
detallar con suficientes dibujos las transformaciones que sufre durante la inserción de la clave 80. Dibuja también el árbol final resultante como árbol 2-3-4.

Ejercicio 15 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 16 ¿Cuál es el mayor número posible de nodos en un árbol rojinegro estándar con altura negra k ? ¿Cuál es el menor valor posible?

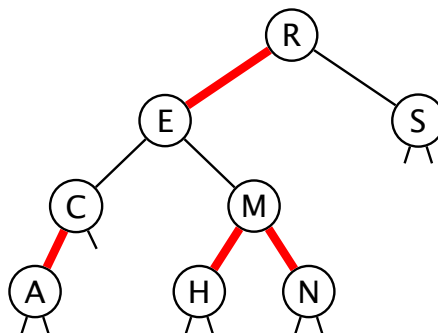
Ejercicio 17 Dibujar todos los árboles rojinegros estándar estructuralmente diferentes con n valores, para n desde 1 hasta 6.

Ejercicio 18 Considera el siguiente ABB rojinegro inclinado a la izquierda, donde se ha ocultado una clave y el color de algunas aristas:



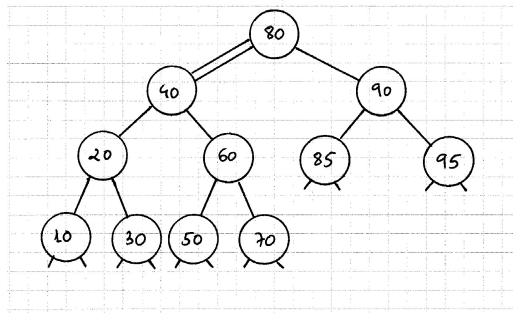
1. ¿Cuáles de las siguientes claves podrían ser la clave oculta (marcada con una interrogación)?
A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z
2. ¿Cuáles de las siguientes claves deben ser rojas (es decir, es roja la arista que la une a su padre)? P Q S T Y
3. Ilustrar con suficientes dibujos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

Ejercicio 19 Considera el siguiente árbol rojinegro inclinado a la izquierda:



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

Ejercicio 20 Dado el siguiente árbol rojinegro inclinado a la izquierda, ilustrar con suficientes dibujos la ejecución de la operación *deleteMax*. Los dobles enlaces representan enlaces rojos.



Ejercicio 21 Un árbol *blaurrojinegro* se define como un árbol rojinegro en el que se puede utilizar también el color azul para pintar los nodos, con la restricción adicional de que un nodo azul solo puede ser hijo de un nodo rojo. Justificar que estos árboles son equivalentes a los 2-3-4-5-6-7-8 árboles.

Ejercicio 22 Sobre el siguiente árbol autoajutable, también llamados *splay trees*, se ejecuta la operación de inserción de la clave 45. Ilustra con suficientes dibujos la transformación que sufre el árbol durante dicha inserción

