Tarea Nueve
Programación y Algoritmos I

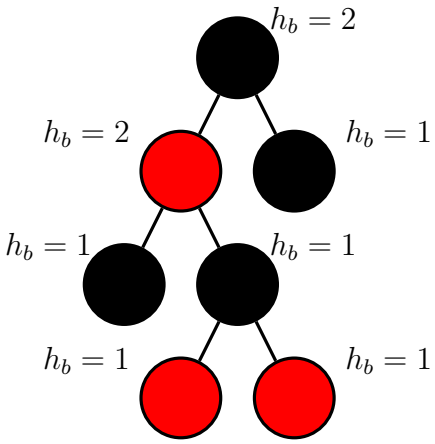
Andrea Quintanilla Carranza Esteban Reyes Saldaña

20 de octubre de 2020

Un árbol **rojo-negro** (RB,por sus siglas en inglés) es un ABB que tiene las siguientes restricciones

1. Cada nodo tiene un solo elemento (**label**) llamado color que es o rojo o negro.
2. La raíz del ABB es siempre negra.
3. Los hijos de un modo rojo **tiene que ser ambos negros** (o sea, no puede haber dos rojos consecutivos en un camino de la raíz a una hoja).
4. Las ligas vacías (apuntadores NULL en los nodos termianles) cuentan como negro.
5. Cualquier camino de un nodo v del árbol hacia un NULL tienen el mismo número de nodos negros (sin contar v). Ese número se llama **altura negra** de v y lo notaremos $h_b(v)$.

En la figura siguiente, se ilustra un árbol rojo-negro con los valores $h_b(v)$ correspondientes. No se ha representado los apuntadores NULL.



Pregunta 1. Demostrar que si r es la raíz de un árbol rojo-negro de altura h, tenemos

$$h_b(r) \geq \frac{h}{2}. \tag{1}$$

Demostración. Sea r la raíz de un árbol RB y h su altura. Observemos primero que por la propiedad 5 de los RB (cualquier camino de un nodo hacia una hoja tiene el mismo número de nodos negros), $h_b(r)$ está bien definida. Sabemos que si existe un nodo rojo, la propiedad 3 asegura que sus dos hijos son negros. Por lo que para cualquier camino de la raíz a un hoja, al menos la mitad de los nodos son negros, de lo contrario, utlizando el principio del palomar, existiría un nodo rojo con un hijo rojo. Por lo tanto:

$$h_b(r) \geq \frac{h}{2}.$$

□

Pregunta 2. Demostrar por inducción sobre la altura de los nodos que un subárbol enraizado en un nodo v tiene al menos $2^{h_b(v)} - 1$ nodos internos.

Demostración.

(i) Si $h = 0$, ya que los nodos internos con aquellos no nulos, el subárbol tiene 0 nodos internos y un solo elemento v y corresponde a un nodo terminal (NULL), entonces $h_b(v) = 0$. Así que

$$2^{h_b(v)} - 1 = 2^0 - 1 = 0.$$

De donde se cumple que v tiene al menos $2^{h_b(v)} - 1$ nodos internos.

(ii) Tomemos ahora un árbol enraizado en un nodo v y altura $h > 0$. Sabemos que v puede tener un hijo o dos, pero observemos que si sólo tiene un hijo, necesariamente su hijo es rojo, de lo contrario tendría altura nergra igual a 1 por un lado y mayor a 1 por el otro. Así que el árbol sólo consta de una raíz negra y un hijo rojo, en cuyo caso $2^{h_b(v)} - 1 = 1$ y la afirmación se cumple. Para el caso en el que v tiene dos hijos vc_1 y vc_2. Notemos que la altura negra de sus hijos es $h_b(v)$ si el hijo es rojo $h_b(v) - 1$ si el hijo es negro. Así, en el caso de mínima altura, la altura negra de los hijos es $h_b(v) - 1$, cuando ambos hijos son negros y el padre es rojo.
Supongamos entonces que la propiedad se cumple para todo subárbol de altura menor que h. Particularmente, sabemos que la altura de los subárboles enraizados en vc_1 y vc_2 es menor a la altura del subárbol enraizado en v, entonces

- el subárbol enraizado en vc_1 tiene al menos $2^{h_b(v)-1} - 1$ nodos y
- el subárbol enraizado en vc_2 tiene al menos $2^{h_b(v)-1} - 1$ nodos.

así que el el subárbol enraízado en el padre v tendrá al menos

$$2^{h_b(v)-1} - 1 + 2^{h_b(v)-1} - 1 + 1 = 2\left(2^{h_b(v)-1}\right) - 1 = 2^{h_b(v)} - 1$$

nodos. Se suma un 1, pues ambos hijos son negros. Así que la propiedad se cumple también para el subárbol enraizado en v, por el principio de inducción matemática, tenemos que la propiedad se cumple para toda $h_b(v) \in \mathbb{N}$.

□

Pregunta 3. Deducir de lo anterior que, si n es el número total de nodos, la altura h del árbol satisface:

$$h \leq 2\log_2(n+1)$$

Solución. Sea h la altura de un árbol RB y r su raíz. Por el ejercicio anterior sabemos que el árbol tiene al menos $2^{h_b(r)} - 1$ nodos internos, es decir, si denotamos por n al total de estos últimos, tenemos que:

$$n \geq 2^{h_b(r)} - 1.$$

Por la pregunta 1 tenemos que:

$$\begin{aligned} h_b(r) &\geq h/2 \\ 2^{h_b(r)} &\geq 2^{h/2} \end{aligned}$$

Por lo que poedemos concluir:

$$\begin{aligned} n &\geq 2^{h/2} - 1 \\ n+1 &\geq 2^{h/2} \\ \log_2(n+1) &\geq h/2 \\ 2\log_2(n+1) &\geq h. \end{aligned}$$

Esta propiedad demuestra que los árboles RB, efectivamente están balanceados, y por lo tanto las operaciones en ellos que dependen de la altura son de complejidad logarítmica. 😊

Pregunta 4. Definimos la inserción de un nuevo dato en un árbol rojo-negro como sigue: insertamos el nuevo nodo w como en un ABB normal (bajando hacia su lugar por búsqueda) y lo coloreamos como **rojo**. Si ese nodo es la raíz (w fue el primer nodo), lo coloreamos como negro. Mostrar que el único caso en que se puede generar una violación de las reglas de árbol rojo-negro es cuando el padre de w es rojo.

Solución. Sea z un nuevo nodo insertado a un árbol RB. Enlistando las posibiles fallas de las propiedades de un árbol rojo-negro que se pueden crear al hacer una inserción como la descrita:

1. Cada nodo tiene un solo elemento (**label**) llamado color que es o rojo o negro: siempre se cumple.

2. La raíz del ABB es siempre negra: si después de hacer inserción z es la raíz, esta es roja, así que se recolorea a negro. En otro caso, la raíz es la misma que la original.

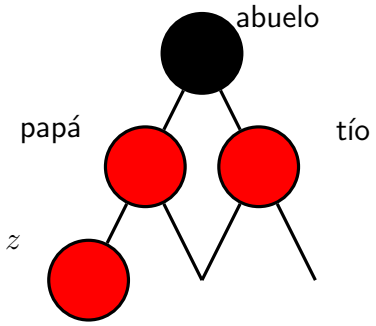
3. Los hijos de un nodo rojo **tiene que ser ambos negros**: esta propiedad falla si el padre de z es rojo.

4. Las ligas vacías (apuntadores NULL en los nodos terminales) cuentan como negro: se sigue cumpliendo, ya que los únicos nodos terminales agregados son los hijos de z y en la inserción normal estos se inicializan como NULL y NULL se cuenta como negro.

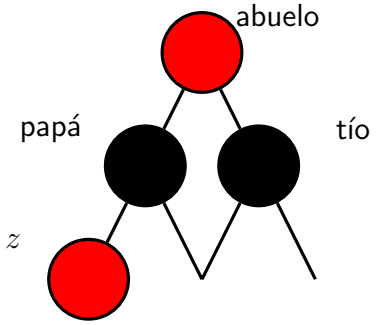
5. Cualquier camino de un nodo v del árbol hacia un NULL tienen el mismo número de nodos negros (sin contar v): se cumple, dado que antes de insertar z el futuro padre de z apunta a NULL y al reemplazarlo por z se le resta 1 a la altura negra de los árboles que estén en ese camino. Pero ahora z es rojo y apunta a NULL, así que se suma 1 a la altura negra de los nodos en esa rama. Lo anterior aplica también en el caso en el que el padre de z sea NULL, es decir, cuando z es la raíz. Por otro lado, si el nuevo dato se acompaña de una llave ya existente, en la inserción usual, se "aplasta.[el] dato, es decir, el árbol no se modifica salvo en el dato correspondiente a la llave, por lo que las alturas permanecen siendo la mismas ya que no se agregan nodos.

Concluimos que la única propiedad que se puede violar al insertar de esa manera un nodo es cuando el padre de z es rojo.

Pregunta 5. Mostrar que, en el caso anterior de violación, si el nodo tío de w (es decir, el otro hijo de su abuelo) es también rojo, hay una corrección muy simple que se puede hacer al *cambiar de colores el abuelo, el papá y el tío.* ¿Cómo cambia la altura negra de los nodos del árbol con esta corrección? Mostrar que la corrección puede provocar una violación al nivel el abuelo. ¿Cuál es la complejidad de esta corrección?
Solución. Supongamos que se inserta un nodo z con color rojo y que su padre también es rojo. Supongamos además que el tío de z también es rojo. Notemos que, dado que el árbol es RB, el abuelo de z no puede ser rojo (porque entonces sus hijos serían rojos y eso viola la propiedad 4). Así que el abuelo de z es negro.



Si cambiamos los colores del abuelo, el papá y el tío ahora tendríamos la configuración:

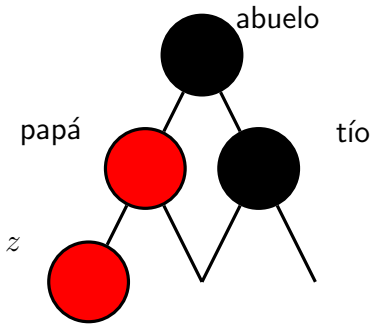


Notemos que si la nueva configuración es un árbol rojo-negro, la altura negra de los nodos papá, tío y abuelo aumentará en 1, mientras que para los demás nodos permanecerá igual.
Si el padre del abuelo es negro, entonces el árbol sigue siendo rojo-negro, porque, como se muestra en la figura, las alturas negras en el subárbol que tiene por raíz al abuelo respetan la propiedad correpondiente, y las del resto del árbol no se modifican.
Por otro lado, si el padre del abuelo es rojo, y suponemos que su tío es rojo, volvemos al escenario anterior, y la correción se puede repetir mientras se repita la misma violación.
En el mejor caso, solo se hace una correción y ésta se hace en $O(1)$ pues el cambio de tres colores es una operación con valor constante. En el peor caso, esta corrección se hace hasta llegar a la raíz, en este caso, la complejidad será proporcional a la altura del árbol, y por una pregunta anterior, sabemos que esa altura es de $O(log(n))$, donde n es el número de llaves.

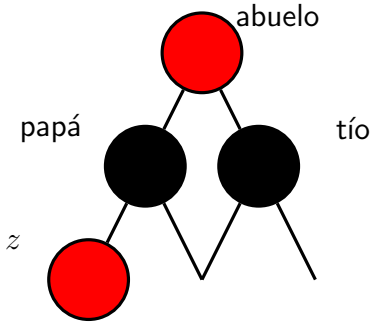
Pregunta 6. Mostrar que en el otro caso (si el tío es negro), se pueden usar las mismas **rotaciones** que vimos en el caso de árboles AVL para corregir el

árbol rojo-negro. ¿Cómo cambia la altura negra de los nodos del árbol con esta corrección? ¿Cuál es la complejidad de esta corrección?

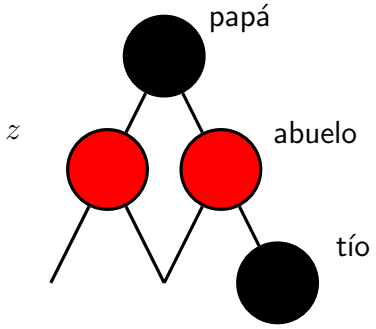
Solución. Podemos asegurar que el subárbol con raíz abuelo, tiene, salvo reflexiones (derecha-izquierda), la siguiente estructura:



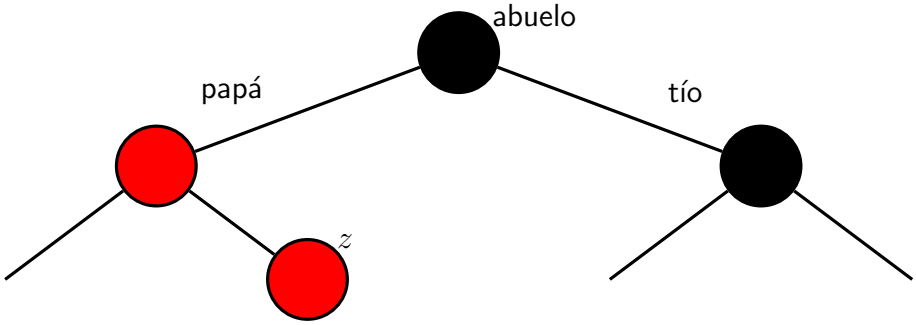
de lo contrario el papá también sería negro y no habría violación, los hijos restantes del papá y del tío pueden ser subárboles no necesariamente nulos. En el caso en el que el papá y el abuelo sean hijos izquierdos, como se muestra en la imagen primero se invierten los colores del papá y del abuelo:



Después se aplica una rotación hacia la derecha:



Este subárbol es rojo-negro porque el abuelo era originalmente negro y el nuevo nodo de raíz también lo es, además las alturas negras de los predecesores al subárbol no se ven afectadas, y la configuración del subárbol de la última imagen se mantiene equilibrado si se reacomodan los subárboles como se vio en clase (si el papá hubiera tenido un hijo/subárbol derecho, ese se convierte en hijo/subárbol izquierdo del abuelo). Observemos que existe otro caso en el que el papá se mantiene siendo hijo izquierdo del abuelo pero el z es hijo derecho del papá:



pero haciendo una rotación a la izquierda del subárbol del papá volvemos al caso de la primera imagen. Los casos restantes son análogos a los anteriores salvo reflexiones (derecha-izquierda).

Finalmente, la complejidad de una rotación es constante pues sólo involucra hacer cambios de apuntadores y en el peor caso haremos dos rotaciones y una coloración, así que la complejidad de la correción en este caso es $O(1)$.

Pregunta 7. Adjuntamos el zip en la entrega, añadimos un makefile y el ejecutable se llama RUN. Este es el enlace de nuestro repositorio: https://github.com/AndreaQuinCa/pai-tarea-09.git. En los issues escribimos cómo repartimos las actividades y algunos otros comentarios.