Tarea 9 - Programación y Algoritmos 1

Juan Luis Baldelomar Cabrera y Abdiel Ulises Beltrán Aranda

October 19, 2020

Empecemos por enumerar las propiedades de un árbol rojo-negro:

1. Cada nodo es rojo o negro
2. La raíz es negra
3. Cada hoja (NIL) es negra
4. Si un nodo es rojo, entonces ambos de sus hijos son negros
5. Para cada nodo, todos los caminos simples hasta las hojas descendientes tiene la misma cantidad de nodos negros

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

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

Demostración. *Por la propiedad 4 podemos ver que al menos la mitad de los nodos de cualquier camino simple hasta las hojas son negros, si tomamos el camino que define la altura del árbol implica que la cantidad de nodos negros en ese camino son al menos* $\frac{h}{2}$*. Por la propiedad 5, todos los caminos tienen la misma cantidad de nodos negros, por lo tanto*

$$h_b \geq \frac{h}{2}$$

■

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

Demostración. *Es importante considerar que para estos árboles los nodos hoja son los nodos NIL.*

Procederemos a mostrarlo por inducción sobre la altura h_b*. Supongamos que tenemos un subárbol de altura 0. Por lo tanto estamos en un nodo hoja (NIL) que tiene una altura de* $h_b = 0$ *y así tenemos*

$$2^0 - 1 = 0$$

nodos internos, por lo cual podemos ver que cumple para el caso base.

Ahora supongamos que tenemos un subárbol con raíz en v con altura positiva y con dos hijos. Cada hijo tiene una altura negra de $h_b(v)$ o $h_b(v) - 1$ dependiendo de si el hijo es rojo o negro respectivamente. Debido a que la altura de los hijos es menor que la altura del nodo v, aplicando la hipótesis inductiva tenemos que cada hijo tiene al menos $h_b(v) - 1$ nodos internos, de manera que el nodo padre v tiene (contandose a él mismo) al menos

$$\begin{aligned} &(2^{h_b(v)-1} - 1) + (2^{h_b(v)-1} - 1) + 1 \\ =&2^{h_b(v)} + 1 \end{aligned}$$

■

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

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

Demostración. *Por la pregunta 1 y la pregunta 2, tenemos que*

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

■

4. Definimos la inserción de un nuevo dato en un arbol 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 raız (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 arbol rojo-negro es cuando el padre de w es **rojo**.

   Demostración. *Si el nodo a insertar es el primero, entonces tenemos que estamos insertando la raíz, por lo tanto se colorea de negro y las propiedades de un árbol rojo-negro se siguen cumpliendo.*

   Supongamos ahora que el nodo a insertar no es la raíz y lo insertamos en el subárbol con raíz v. Claramente este subárbol cumple las propiedades de un árbol rojo-negro. Si el nodo v es negro, al insertar el nuevo nodo este nodo es coloreado de color rojo y tiene dos hijos (NIL) que son de color negro, por lo tanto el subárbol con raíz en v sigue manteniendo la misma altura h_b y sigue cumpliendo las demás propiedades.

   Por otro lado, si el nodo raíz v es rojo, al insertar el nuevo nodo este es coloreado de color rojo y por lo tanto ahí ocurre una violación de la propiedad 4.

   ■

5. Mostrar que en el otro caso (si el tıo es negro), se puede usar las mismas rotaciones que vimos en el caso de árboles AVL para corregir el arbol rojo-negro. Cómo cambia la altura negra de los nodos del árbol con esta correción? Cuál es la complejidad de esta correción?

Demostración. *Si se genera una violación debido a que el nodo padre v es rojo entonces sabemos que el nodo abuelo v_p existe, porque debido a que las propiedades de un árbol rojo-negro se cumplian antes de la inserción, el nodo raíz debe ser negro, por lo tanto el nodo padre v no puede ser el nodo raíz y así el nodo abuelo v_p existe. Si el nodo tío v_t también es rojo, entonces al cambiar los colores del nodo abuelo y los hijos del abuelo, las propiedades de un árbol rojo-negro cumplen para el subárbol con raíz en v_t.*

Notemos que al cambiar el color del nodo abuelo, si este es la raíz, genera una violación de la propiedad 2. Si por otro lado este no es la raíz y su padre (el bisabuelo) es de color rojo, entonces nuevamente tenemos otro caso en el cual se genera una violación de las propiedades. Por lo tanto es necesario repetir este procedimiento con el nodo abuelo.

Es fácil convencernos de que la altura negra para el subárbol del nodo abuelo aumenta en uno en este escenario, pues al cambiar de colores el abuelo no tiene ningún efecto para su altura negra ya que el no cuenta, pero cambiar el color de sus hijos le aumenta en 1 la altura negra. Sin embargo, la altura negra de todo el árbol no cambia, ya que con el intercambio de color del nodo abuelo y sus hijos aumenta y disminuye en 1 la altura negra del subárbol que contiene al nodo abuelo.

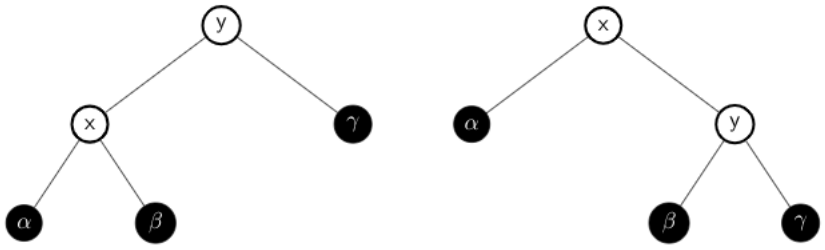
La complejidad total de esta corrección depende de la altura del árbol ya que en cada iteración va subiendo dos niveles, por lo tanto es de

$$O(\log(n+1))$$

■

6. Mostrar que en el otro caso (si el tıo es negro), se puede usar las mismas rotaciones que vimos en el caso de árboles AVL para corregir el arbol rojo-negro. Cómo cambia la altura negra de los nodos del árbol con esta correción? Cuál es la complejidad de esta correción?

Demostración. *En el caso de rotación derecha, la idea es pasar el hijo izquierdo de y, el cual llamamos x, a la posiciń de y y a la vez cambiar el hijo derecho de x en la posición del hijo izquierdo de y para mantener la estructura de árbol de búsqueda. En el caso de rotación izquierda, la idea es pasar el hijo derecho del nodo x, el cual llamamos y, a la posiciń de x y a la vez movemos el hijo izquierdo de y a la posición del hijo derecho de x para mantener la estructura de árbol de búsqueda. Lo anterior lo podemos visualizar de la siguiente manera.*

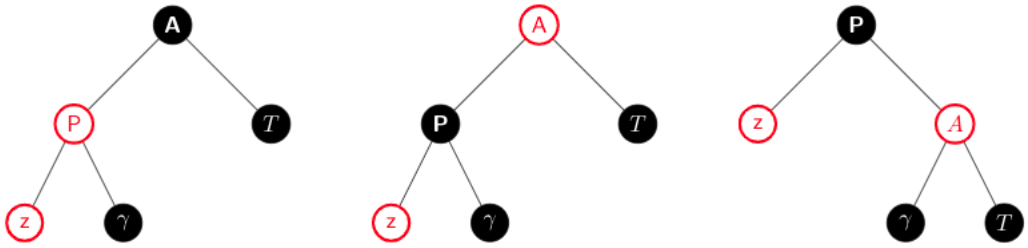


En la figura anterior a la estructura izquierda se le aplica una rotación derecha, resultando en la estructura derecha. A su vez, si a la estructura derecha se le aplica una rotación izquierda, se obtiene la estructura izquierda. Los nodos en relleno blanco indican los nodos sobre los que se hace la rotación.

Ahora, en el caso de arboles rojo-negro, hacemos rotaciones cuando agregamos un nuevo nodo, que denotamos por z, tal que el nodo padre de z es rojo y el tío de z es negro. Si antes de insertar el nodo z el árbol estaba en un estado válido, esto obliga a que el abuelo de z sea un nodo negro. Denotamos al abuelo de z como A y al nodo padre de z como P. Tenemos entonces 4 posibles casos: dependiendo de la posición entre P y A, así como la posición entre P y z

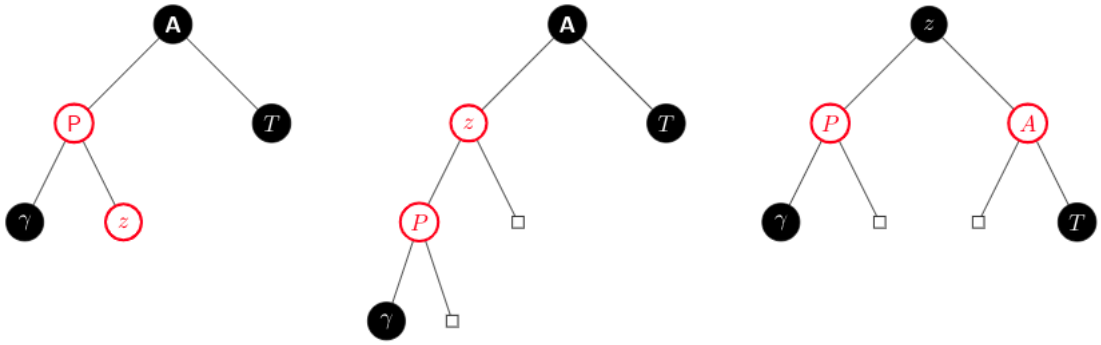
P a la izquierda de A

El caso sencillo es cuando z es el hijo izquierdo de P. Ya A es de color negro, cambiamos el color de P a negro y cambiamos el color de A a rojo antes de hacer la rotación hacia la derecha sobre el nodo A y P. Esto asegura que el nodo que queda en la posición de abuelo será negro, por lo cual evitamos repetir el problema. Ya que antes de insertar z el árbol está en un estado válido, significa que el nodo P tiene un hijo derecho negro o NIL, en ambos casos el hijo derecho de P es de color negro. Denotando por T al tío de z, el proceso es el siguiente



Después de la rotación, la altura negra en la posición original del nodo P cambia a cero ya que el sub-árbol γ se cambia al nodo A al final. La altura negra del nodo P original será la altura negra del nodo A final si tomamos el camino de su hijo izquierdo. A lo largo de la rama que contiene al nodo T en la configuración final, la altura negra no cambia con respecto a la posición final de P, ya que en ambos casos esa rama conserva el mismo sub-árbol T.

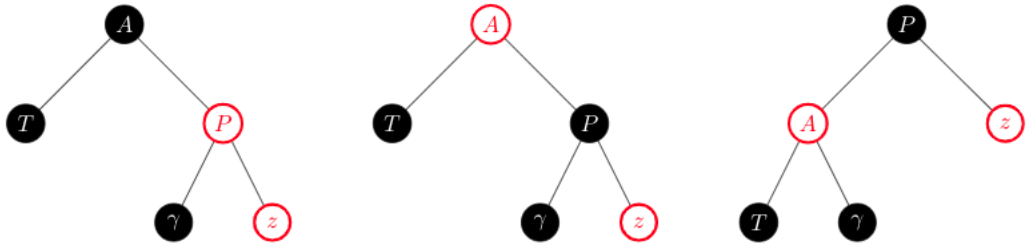
El segundo subcaso es cuando el nodo z se agrega como hijo derecho de P. En este caso se debe hacer una rotación extra a la izquierda sobre el nodo P para llegar al caso descrito arriba.



En este segundo caso se representan con cuadrados los nodos NIL de z y como cambian de lugar. En ambos subcasos el nodo γ puede representar otro sub-árbol o un nodo NIL. En este subcaso la altura negra con respecto al nodo A original, que ahora es el nodo z, no cambia en ninguno de los dos sub-arboles. Ya que una rotación solo implica cambiar apuntadores entre los nodos, en el pero caso en que son necesarias dos rotaciones la complejidad será constante $O(8) = O(1)$.

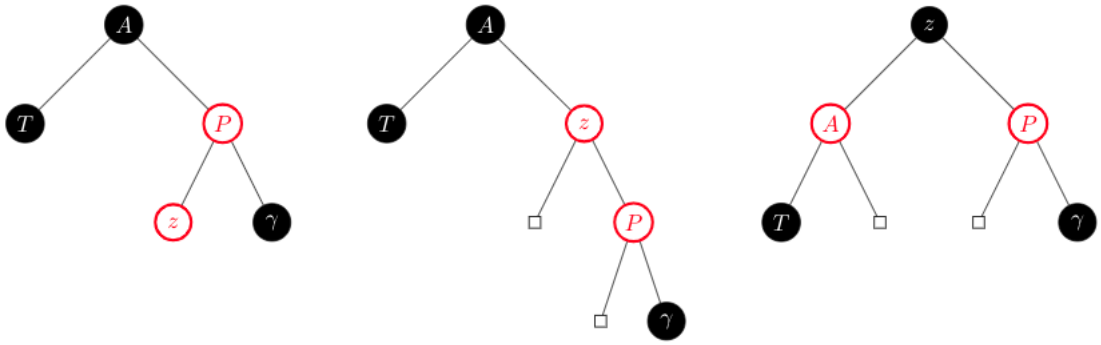
P a la derecha de A

El caso sencillo, en el que se necesita solo una rotación, tenemos a z como hijo derecho de P. Mantenemos las etiquetas T para el tío de z, A para su abuelo y γ para el posible sub-árbol restante de P. Nuevamente antes de hacer la rotación cambiamos el color de A y P para que el nodo que queda en la posición de abuelo sea negro como el original y no tengamos que corregir hacia arriba nuevamente.



En este subcaso, la altura negra desde el nodo A original (que ahora es P) a lo largo del sub-árbol T no cambia. La altura negra a lo largo del camino que pasa por z tampoco cambia con respecto a la posición final P. La altura negra desde la posición original de P (y por ende también con respecto a la posición original de A), que pasa por posible sub-árbol γ se hace cero. Esta misma altura negra $h_b(\gamma)$, será la altura negra con respecto a la posición final de P (y por lo tanto también de A) a lo largo del camino que contiene al posible sub-árbol γ.

Para el segundo subcaso, insertamos el nuevo nodo z como hijo izquierdo del nodo P. Por lo tanto hay que hacer una rotacón hacia la derecha con respecto al nodo P para obtener la misma estructura del subcaso anterior. Finalmente, posterior al cambio de color de A y su hijo derecho, hacemos una rotación izquierda con respecto al nodo A.



En este subcaso la altura negra no cambia con respecto a la posición original de A (donde queda z al final), ya que a lo largo de la rama izquierda tenemos $h_b(T)$. A lo largo del camino derecho, en donde esá originalmente el posible sub-árbol γ, la altura negra sigue siendo $h_b(\gamma)$. En el peor caso se tienen que hacer dos rotaciones para llegar al estado final, ya que las rotaciones toman un número constante de operaciones para reasignar los apuntadores, la complejidad es constante $O(1)$.

7. Implementación árbol Rojo-Negro:

   Repositorio Git: https://github.com/Juan-Baldelomar/PA1-T9