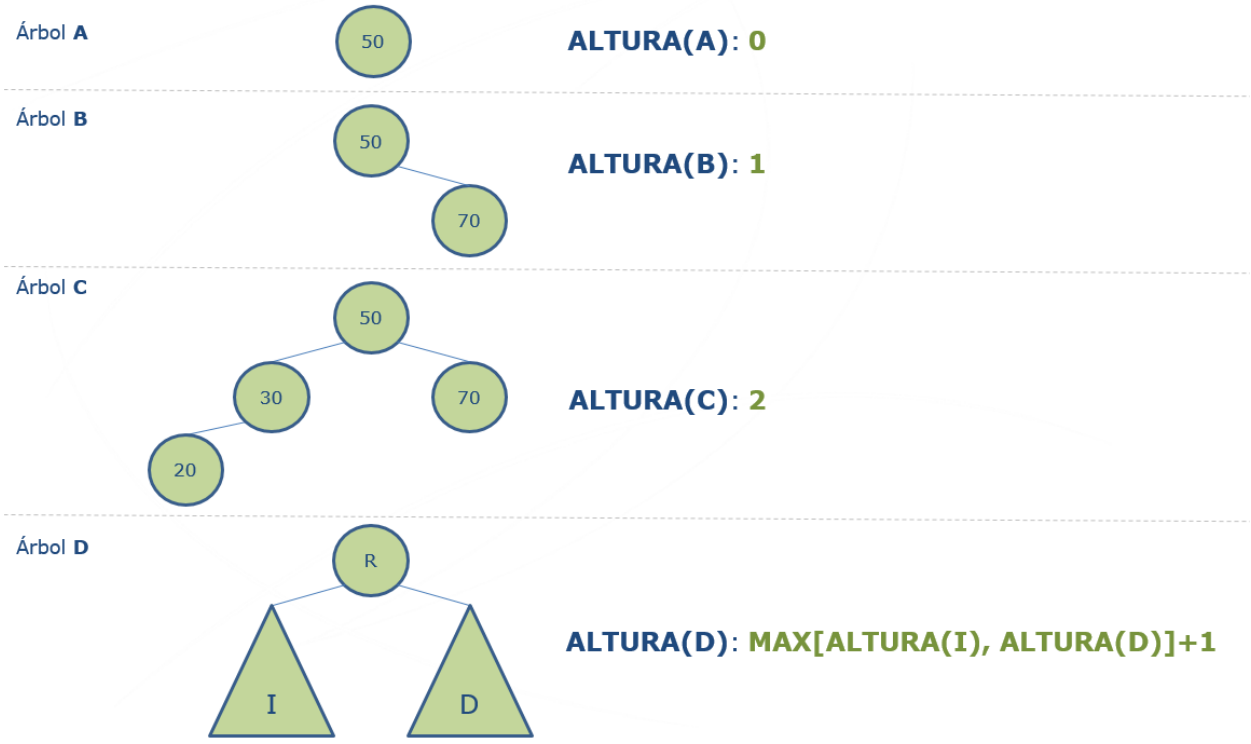
Apuntes

Tema ***Arboles Binarios de Búsqueda Autobalanceados***

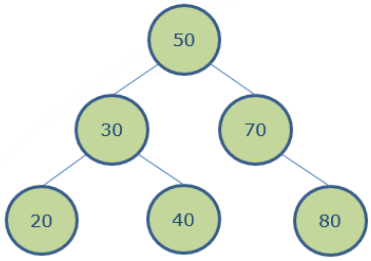
"El científico no es aquella persona que da las respuestas correctas, sino aquél quien hace las preguntas correctas."
Claude Lévi-Strauss

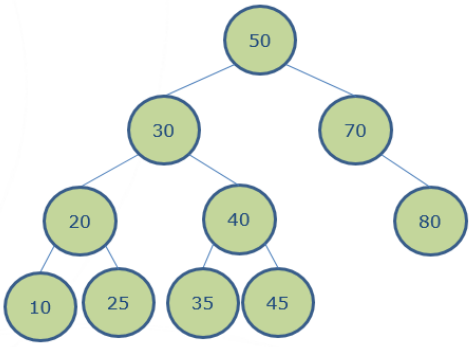
Arboles auto-balanceados

Recordemos conceptos de altura



Criterios de equilibrio (o de balanceo)





El número de nodos del subárbol izquierdo difiere como máximo en 1 con el número de nodos del subárbol derecho	La altura del subárbol izquierdo difiere como máximo en 1 con la altura del subárbol derecho
Árbol Perfectamente Equilibrado	Árbol Equilibrado

Arboles Rojo Negro Historia

Un árbol rojo-negro es un tipo abstracto de datos. Concretamente, es un árbol binario de búsqueda equilibrado. La estructura original fue creada por **Rudolf Bayer** en 1972, que le dio el nombre de *"árboles-B binarios simétricos"*, pero tomó su nombre moderno en un trabajo de Leo J. Guibas y Robert Sedgewick realizado en 1978. Es complejo, pero tiene un buen peor caso de tiempo de ejecución para sus operaciones y es eficiente en la práctica. Puede buscar, insertar y borrar en un tiempo **O(log n)**, donde n es el número de elementos del árbol.


Rudolf Bayer

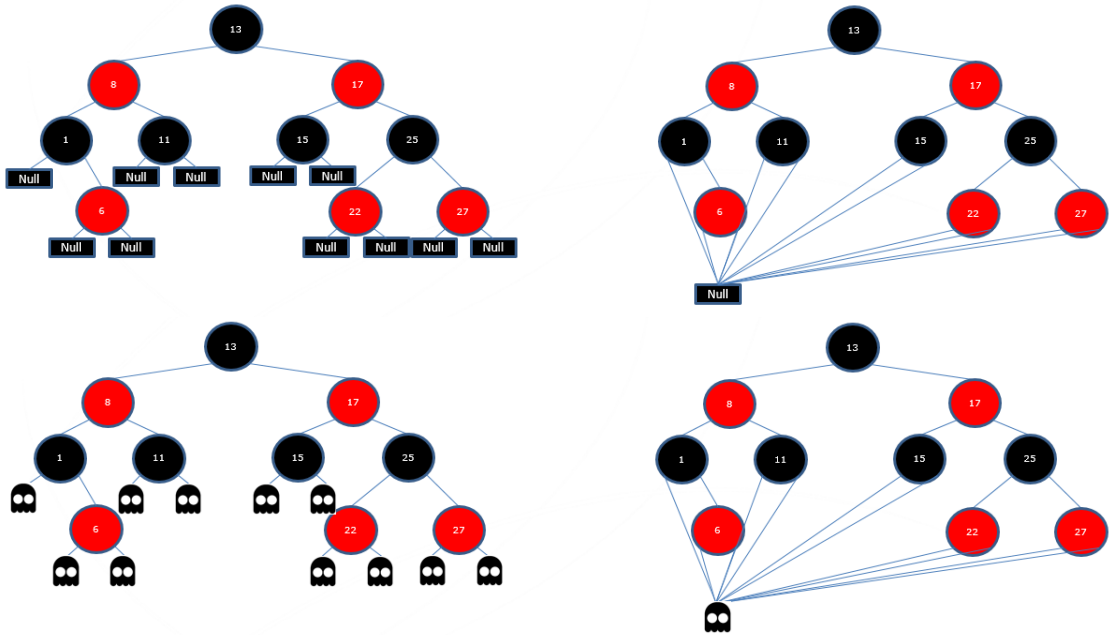
En los árboles rojo-negro las hojas no son relevantes y no contienen datos. Y, como en todos los *árboles binarios de búsqueda*, es posible moverse ordenadamente a través de los elementos de forma eficiente si hay forma de localizar el padre de cualquier nodo. El tiempo de desplazarse desde la raíz hasta una hoja a través de un árbol equilibrado que tiene la mínima altura posible es de **O(log n).**


J. Guibas


Robert Sedgewick

Introducción y definiciones

Al implementar esta estructura es posible utilizar un único **nodo centinela** *(nodo **fantasma**)*. Este cumple la función de hoja para todas las ramas del árbol. Así, todos los nodos internos que finalicen en una hoja tienen referencia a este único nodo centinela. Esto no es necesario, ya que puede hacerse una referencia nula (Null) en el final de cada rama.



Un árbol rojo-negro es un **árbol binario de búsqueda** en el que cada nodo tiene un atributo de **color** cuyo valor es rojo o negro. En adelante, se dice que un nodo es rojo o negro haciendo referencia a dicho atributo.
Además de los requisitos impuestos a los árboles binarios de búsqueda convencionales, se deben satisfacer las siguientes reglas para tener un árbol rojo-negro válido:

1) **Todo nodo es o bien rojo o bien negro.**
2) **La raíz es negra.**
3) **Todas las hojas fantasmas (NULL) son negras.**
4) **Todo nodo rojo debe tener dos nodos hijos negros.**
5) **Cada camino desde un nodo dado a sus hojas descendientes contiene el mismo número de nodos negros.**

Estas reglas producen una regla crucial para los árboles rojo-negro: **el camino más largo desde la raíz hasta una hoja no es más largo que dos veces el camino más corto desde la raíz a una hoja.**
El resultado es que dicho árbol está "aproximadamente" equilibrado.

Dado que las operaciones básicas como insertar, borrar y encontrar valores tienen un peor tiempo de ejecución proporcional a la altura del árbol, esta cota superior de la altura permite a

los árboles rojo-negro ser eficientes en el peor caso, a diferencia de los árboles binarios de búsqueda.
Para comprobarlo basta ver que ningún camino puede tener dos nodos rojos seguidos debido a la propiedad 4. El camino más corto posible tiene todos sus nodos negros, y el más largo alterna entre nodos rojos y negros. Dado que todos los caminos máximos tienen el mismo número de nodos negros por la propiedad 5, no hay ningún camino que pueda tener longitud mayor que el doble de la longitud de otro camino.
En muchas presentaciones de estructuras arbóreas de datos, es posible para un nodo tener solo un hijo y las hojas contienen información. Es posible presentar los árboles rojo-negro en este paradigma, pero cambian algunas de las propiedades y se complican los algoritmos. Por esta razón, utilizamos **"hojas nulas".**

Una variante en un artículo presentado, que se da al árbol rojo-negro es la de tratarlo como un árbol binario de búsqueda cuyas aristas, en lugar de nodos, son coloreadas de color rojo o negro, pero esto no produce ninguna diferencia. El color de cada nodo en la terminología de este artículo corresponde al color de la arista que une el nodo a su padre, excepto la raíz, que es siempre negra (por la propiedad 2) donde la correspondiente arista no existe.
Al número de nodos negros de un camino se le denomina **"altura negra".**

Algunas afirmaciones sobre Arboles Rojo Negro

Al número de nodos negros desde un nodo "m" hasta cualquier hoja se le denomina **"altura negra"** $\mathrm{bh(m)}$, *(no se cuenta le nodo "m" si este es negro)*.

Un subárbol con raíz en el nodo "m" tiene como mínimo $2^{bh(\mathrm{m})} - 1$ nodos internos.

El árbol vacío, no tiene nodos de modo que

$$\mathrm{bh(m)} = 0 \rightarrow 2^{bh(\mathrm{m})} - 1 = 2^0 - 1 = 1 - 1 = 0$$

Un nodo cualquiera "m" presenta un valor de bh(m); este nodo tiene subárboles cuyas alturas negras pueden ser "bh(m)" si es rojo o "bh(m)-1" si es negro. Por lo tanto cada subárbol tiene al menos $2^{bh(\mathrm{m})-1} - 1$ nodos internos

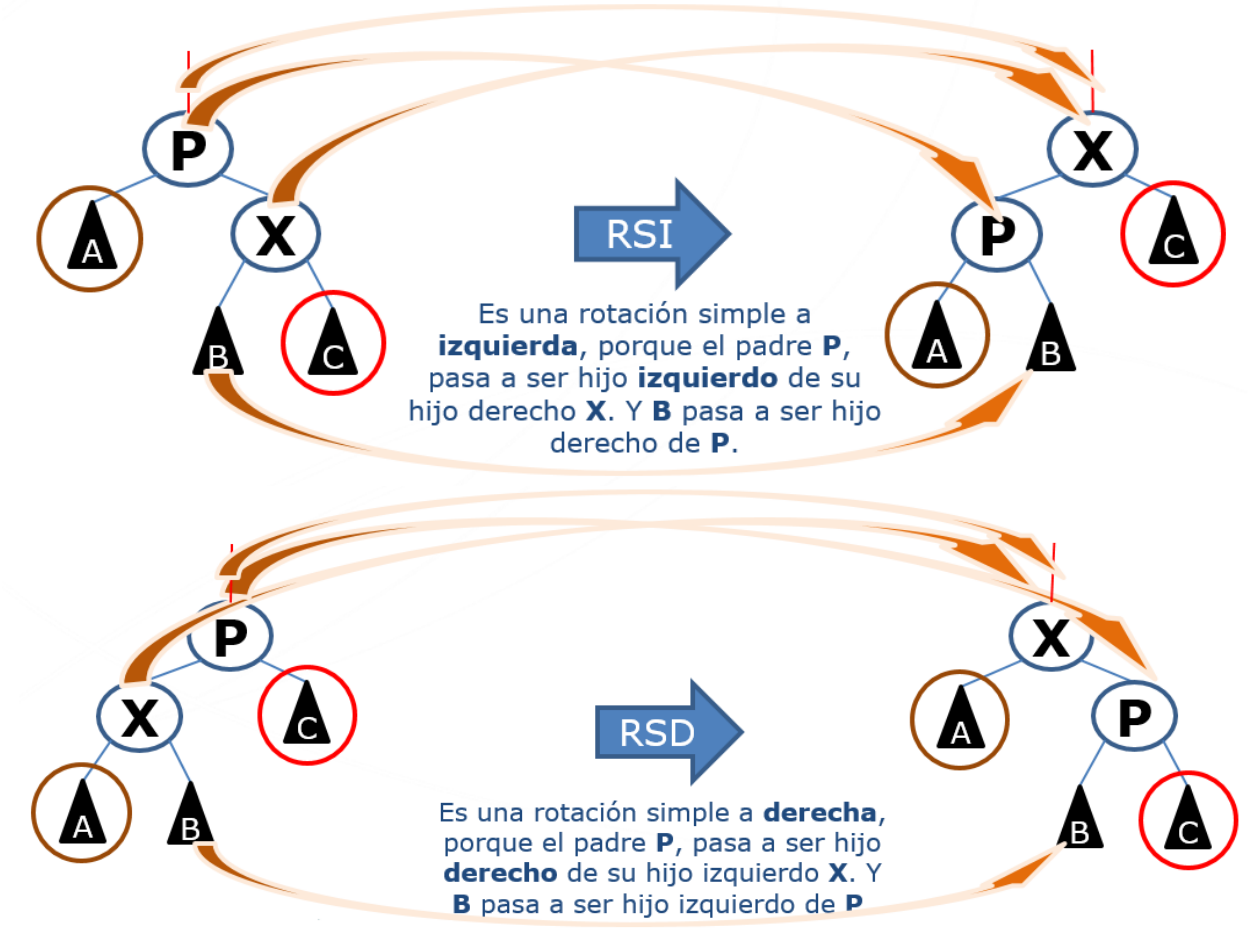
Para cualquier nodo

$$\mathrm{bh(m)} = 1 + \left(2^{bh(\mathrm{m})-1} - 1\right) + \left(2^{bh(\mathrm{m})-1} - 1\right) = 2^{bh(\mathrm{m})} - 1$$

$$\mathrm{bh(raiz)} = \left[\frac{h}{2}\right]; \mathrm{n} \geq 2^{h/2} - 1 \leftrightarrow \log_2(n+1) \geq {}^{h}/_{2} \leftrightarrow h \leq 2\log_2(n+1)$$

Operaciones de Rotación

Para conservar las propiedades que debe cumplir todo árbol rojo-negro, en ciertos casos de la inserción y la eliminación será necesario reestructurar el árbol, si bien no debe perderse la ordenación relativa de los nodos. Para ello, se llevan a cabo una o varias rotaciones, que no son más que reestructuraciones en las relaciones **padre**-**hijo**-**tío**-**nieto**.



La inserción comienza añadiendo el nodo como lo haríamos en un árbol binario de búsqueda convencional y pintándolo de **rojo**. Lo que sucede después depende del color de otros nodos cercanos. El término **tío** nodo será usado para referenciar al hermano del padre de un nodo. Al contrario de lo que sucede en otros árboles como puede ser el Árbol AVL, en cada inserción se realiza un máximo de una rotación, ya sea simple o doble. Por otra parte, se asegura un tiempo de recoloración máximo de $O(\log_2 n)$ por cada inserción.

Nota: *Siempre se inserta como rojo, debido a que si insertamos un nodo negro siempre violaremos la propiedad 5. Y si lo insertamos como un nodo rojo, solo violaremos en algunas oportunidades la propiedad 4.*

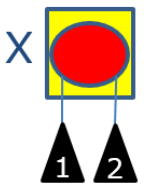
Casos de inserción

1) En un árbol vacío, se inserta un nodo rojo como raíz y se repinta en negro.
2) En un árbol no vacío, se inserta un nodo rojo y si su padre es negro no se hace mas nada, ya que no cambia las propiedades del ARN.

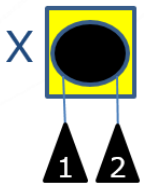
Si en un árbol no vacío se inserta un nodo rojo y si su padre es rojo, se deberá realizar transformaciones para que no viole la propiedad 4.

3) *Inserción con Padre **ROJO** y Tío **ROJO***
4) a) Inserción *con Padre **ROJO** Tío **NEGRO** (X hijo derecho de P)* **izquierda-derecha**
   b) Inserción *con Padre **ROJO** Tío **NEGRO** (X hijo derecho de P)* **derecha-izquierda**
5) a) Inserción *con Padre **ROJO** Tío **NEGRO** (X hijo izquierdo de P))* **izquierda- izquierda**
   b) Inserción *con Padre **ROJO** Tío **NEGRO** (X hijo izquierdo de P))* **derecha-derecha**

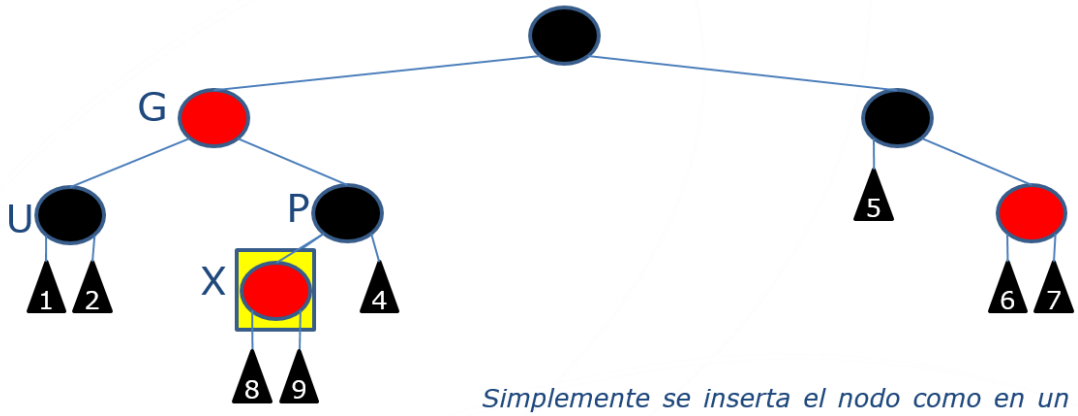
*Inserción – **Caso 1** : Árbol vacío*



*Se recolorea el nodo en **negro***



*Inserción – **Caso 2** :Padre **Negro***



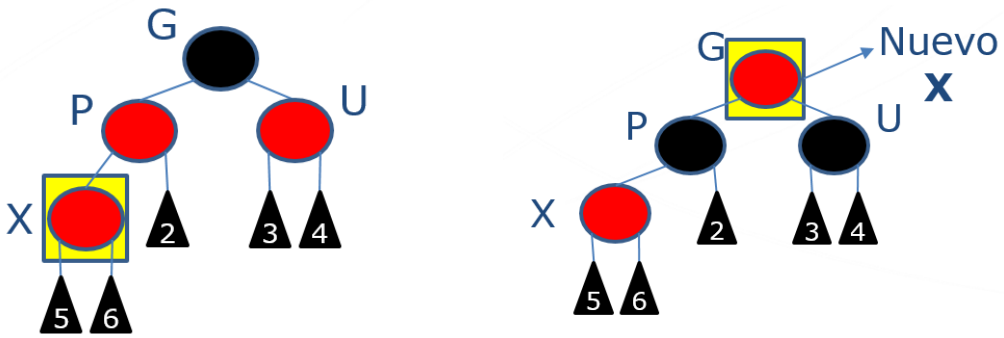
Como había una hoja fantasma que es reemplazado por un nodo rojo se le agregan las hojas fantasmas, para no romper la altura negra.

*Simplemente se inserta el nodo como en un Árbol ABB sin cambiar su color **rojo**.*

Para los siguientes casos siempre debemos tener en mente las 5 propiedades

1) **Todo nodo es o bien rojo o bien negro.**
2) **La raíz es negra.**
3) **Todas las hojas fantasmas (NULL) son negras.**
4) **Todo nodo rojo debe tener dos nodos hijos negros.**
5) **Cada camino desde un nodo dado a sus hojas descendientes contiene el mismo número de nodos negros.**

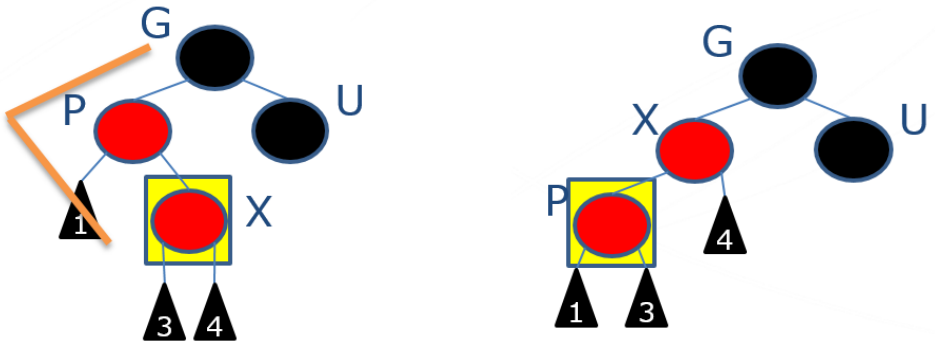
*Inserción – **Caso 3** :Padre **ROJO** y Tío **ROJO***



Corrección: Cambiar de color de G, Y y P

*Si el padre **P** y el tío **U** son rojos, entonces ambos nodos pueden ser repintados de negro y el abuelo G se convierte en rojo para mantener la propiedad 5. Ahora, el nuevo nodo rojo **X** tiene un padre negro. Como cualquier camino a través del padre o el tío debe pasar a través del abuelo, el número de nodos negros en esos caminos no ha cambiado. Sin embargo, el abuelo **G** podría ahora violar la propiedad 2 o la 4; en el caso de la 4 porque **G** podría tener un padre rojo.*
Para solucionar este problema, el procedimiento completo se realizará de forma recursiva hacia arriba hasta alcanzar el caso 1.

Inserción – ***Caso 4*** *:Padre **ROJO** Tío **NEGRO** (X hijo derecho de P)* **izquierda-derecha**

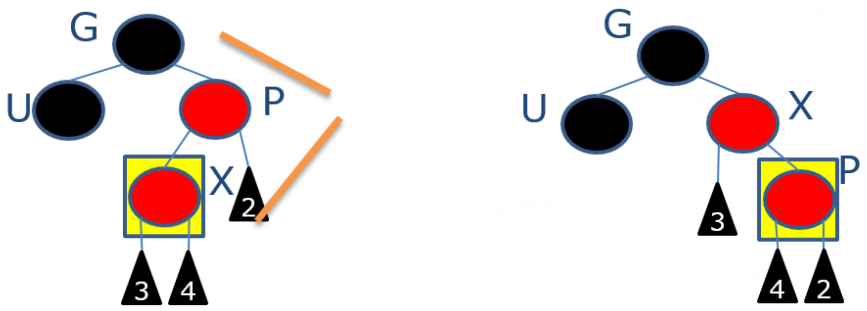


Corrección: Rotar con Padre

*El nodo padre **P** es rojo pero el tío **U** es negro; también, el nuevo nodo **X** es el hijo derecho de **P**, y **P** es el hijo izquierdo de su padre **G**. En este caso, una rotación a la izquierda que cambia los roles del nuevo nodo **X** y su padre **P** puede ser realizada; entonces, el primer nodo padre **P** se ve implicado al usar el **caso 5** de inserción (re etiquetando **N** y **P**) debido a que la propiedad 4 se mantiene aún incumplida. La rotación causa que algunos caminos (en el sub-árbol etiquetado como "1") pasen a través del nuevo nodo donde no lo hacían antes, pero ambos nodos son rojos, así que la propiedad 5 no es violada por la rotación, después de completado este caso, se puede notar que aún se incumple la propiedad 4, esto se resuelve pasando al **caso 5**.*

Nota: *Se asume que el nodo padre **P** es el hijo izquierdo de su padre. Si es el hijo derecho, izquierda y derecha deberían ser invertidas.*

Inserción – ***Caso 4*** *:Padre* ***ROJO*** *Tío* ***NEGRO*** *(X hijo derecho de P)* **derecha-izquierda**



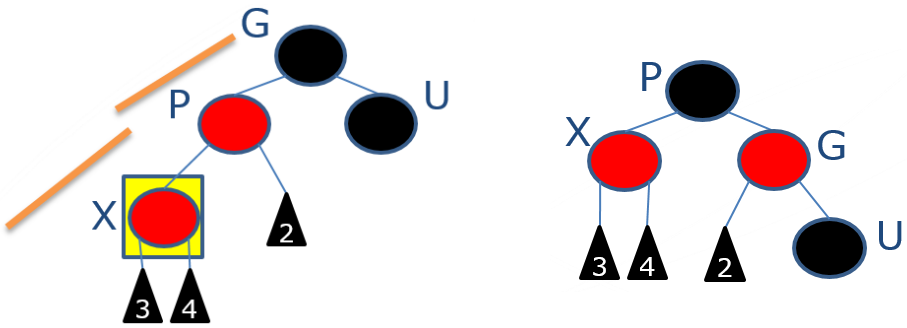
Corrección: Rotar con Padre

Se termina pasando al ***caso 5****. (derecha derecha)*

Espejo del anterior

Nota: *Se asume que el nodo padre* ***P*** *es el hijo derecho de su padre.*

Inserción – ***Caso 5*** *:Padre* ***ROJO*** *Tío* ***NEGRO*** *(X hijo izquierdo de P))* **izquierda- izquierda**

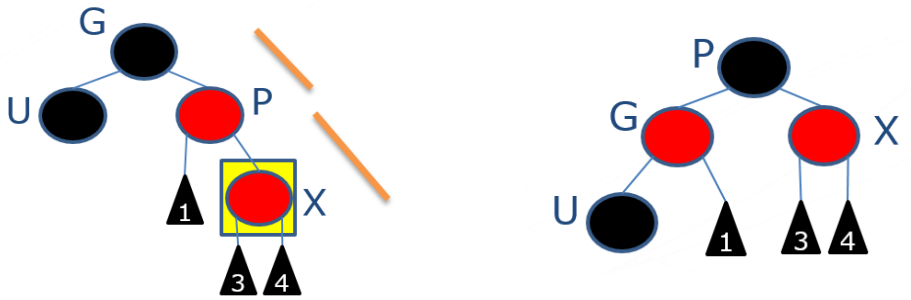


Corrección: Rotar con Padre

*El padre **P** es rojo pero el tío **U** es negro, el nuevo nodo **X** es el hijo izquierdo de **P**, y **P** es el hijo izquierdo de su padre **G**. En este caso, se realiza una rotación a la derecha sobre el padre **P**; el resultado es un árbol donde el padre **P** es ahora el padre del nuevo nodo **X** y del inicial abuelo **G**. Este nodo **G** ha de ser negro, así como su hijo **P** rojo. Se intercambian los colores de ambos y el resultado satisface la propiedad 4. La propiedad 5 también se mantiene satisfecha, ya que todos los caminos que iban a través de esos tres nodos entraban por **G** antes, y ahora entran por **P**. En cada caso, este es el único nodo negro de los tres.*

***Nota:** Se asume que el nodo padre **P** es el hijo izquierdo de su padre. Si es el hijo derecho, izquierda y derecha deberían ser invertidas.*

Inserción – ***Caso 5** :Padre **ROJO** Tío **NEGRO** (X hijo izquierdo de P))* **derecha-derecha**



Corrección: Rotar con Padre

Espejo del anterior

***Nota:** Se asume que el nodo padre **P** es el hijo derecho de su padre.*

Algoritmo de inserción

```
IF (Vacio)  insertar raiz negra
        else
        insertar nodo rojo
        si Padre(insertada)=negro{
```

```
            listo
            }else {
                  si tio es rojo
                        recolor
                        si abuelo no es raiz recheckear
                  }
rotar
si LL    RSD
si LR    RSD RSI
si RL    RSI RSD
si RR    RSI
```

Eliminación de nodos

En un ABB normal, cuando se borra un nodo con **dos nodos** internos como hijos, tomamos el máximo elemento del subárbol izquierdo *(predecesor inorden)* o el mínimo del subárbol derecho *(sucesor inorden)*, y "movemos" su valor al nodo que es borrado ***(es decir esa parte del árbol no cambia estructuralmente)***.
Luego **borramos** entonces el nodo del que copiábamos el valor que ***debe tener menos de dos nodos no hojas por hijos***.
Copiar un valor no viola ninguna de las propiedades rojo-negro y **reduce el problema de borrar en general al de borrar un nodo con como mucho un hijo no hoja**. No importa si este nodo es el nodo que queríamos originalmente borrar o el nodo del que copiamos el valor.

Resumiendo, podemos asumir que borramos un nodo con como mucho un hijo no hoja *(si solo tiene nodos hojas por hijos, tomaremos uno de ellos como su hijo)*. Si borramos un nodo rojo, podemos simplemente reemplazarlo con su hijo, que debe ser negro. Todos los caminos hasta el nodo borrado simplemente pasarán a través de un nodo rojo menos, y ambos nodos, el padre del borrado y el hijo, han de ser negros, así que las propiedades 3 y 4 se mantienen. Otro caso simple es cuando el nodo borrado es negro y su hijo es rojo. Simplemente eliminar un nodo negro podría romper las propiedades 4 y 5, pero si repintamos su hijo de negro, ambas propiedades quedan preservadas.

El caso complicado es cuando el nodo que va a ser borrado y su hijo son negros. Empezamos por reemplazar el nodo que va a ser borrado con su hijo. Llamaremos a este hijo *(en su nueva posición)* **X**, y su hermano *(el otro hijo de su nuevo padre)* **S**. En los diagramas, usaremos **P** para el nuevo padre de **X**, **SL** para el hijo izquierdo de **S**, y **SR** para el nuevo hijo derecho de **S** *(**S** no puede ser una hoja).*

Entre algunos casos cambiamos roles y etiquetas de los nodos, pero en cada caso, toda etiqueta sigue representando al mismo nodo que representaba al comienzo del caso. Cualquier color mostrado en el diagrama es o bien supuesto en su caso o bien implicado por dichas suposiciones. El **blanco** representa un color desconocido *(o bien rojo o bien negro)*.
El cumplimiento de estas reglas en un árbol con n nodos, asegura un máximo de tres rotaciones y hasta $O(\log_2 n)$ recoloraciones.

***Nota**: Con el fin de preservar la buena definición del árbol, necesitamos que toda hoja nula siga siendo una hoja nula tras todas las transformaciones (que toda hoja nula no tendrá ningún hijo). Si el nodo que **estamos** borrando tiene un hijo no hoja N, es fácil ver que la propiedad se satisface. Si, por otra parte **X** fuese una hoja nula, se verifica por los diagramas que para todos los casos la propiedad se satisface también.*

Si **X y su padre original** son negros, entonces borrar este padre original causa caminos que pasan por **X** y tienen un nodo negro menos que los caminos que no *(se genera un doble negro)*. Como esto viola la propiedad 5, el árbol debe ser reequilibrado. Hay varios casos a considerar.

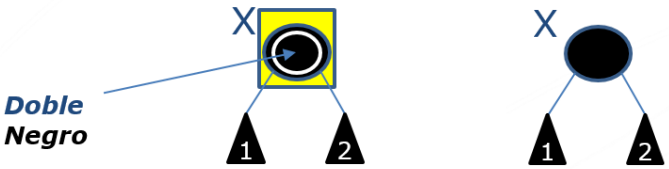
Pasos para el borrado

1) Convertir a caso de 0 o 1 hijo *(Borrado de BST)*.
2) Si el nodo a borrar es rojo o negro y el nodo que reemplaza es rojo y no tiene hijos, se hace el reemplazo directo conservando el color del original.
3) Si el nodo a borrar es negro y el nodo que reemplaza es rojo y tiene un hijo, se hace el reemplazo directo conservando el color negro. Y el hijo del nodo de reemplazo se hace hijo del abuelo del nodo de reemplazo.
4) Si ambos nodos son negros se evalúan los **6 casos**. Y si el nodo de reemplazo es una hoja negra se reemplaza con un nodo nulo **doble negro**.

Para los analizar siguientes casos siempre debemos tener en mente, al igual que para la inserción las 5 propiedades

1) **Todo nodo es o bien rojo o bien negro.**
2) **La raíz es negra.**
3) **Todas las hojas fantasmas (NULL) son negras.**
4) **Todo nodo rojo debe tener dos nodos hijos negros.**
5) **Cada camino desde un nodo dado a sus hojas descendientes contiene el mismo número de nodos negros.**

Eliminación – ***Caso 1** : **X** es la nueva raíz.*



En este caso, hemos acabado. Borramos un nodo negro de cada camino y la nueva raíz es negra, así las propiedades se cumplen.

Eliminación – ***Caso 2*** *: **S** es **ROJO.***
Nota: *Se asume que **X** es el hijo izquierdo de su padre **P**. Si éste fuese el hijo derecho, la izquierda y la derecha deberían ser invertidas en todos estos casos.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando

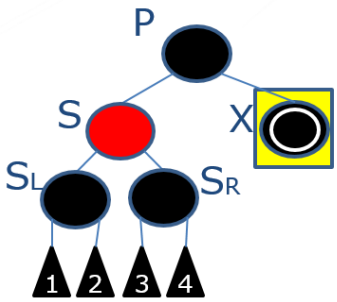
Corrección: Rotar

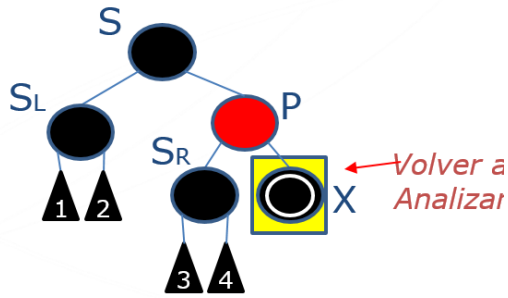
*Condición: Si el nuevo hermano de **X** (negro), **S** es rojo y **P** es negro.*

*En este caso invertimos los colores de **P** y **S**, por lo que rotamos a la izquierda **P**, pasando **S** a ser el abuelo de **X**. Nótese que **P** tiene que ser negro al tener un hijo rojo. Aunque todos los caminos tienen todavía el mismo número de nodos negros, ahora **X** tiene un hermano negro y un padre rojo, así que podemos proceder a al **paso 4, 5 o 6** (este nuevo hermano es negro porque éste era uno de los hijos de **S**, que es rojo). En casos posteriores, reetiquetaremos el nuevo hermano de **N** como **S**.*

Eliminación – ***Caso 2*** *: **S** es **ROJO.***
Nota: *Se asume que **X** es el hijo derecho de su padre **P**.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando



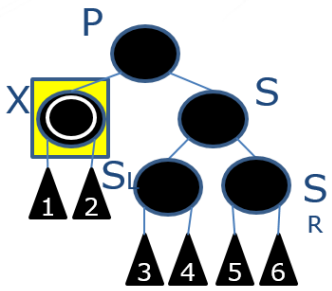


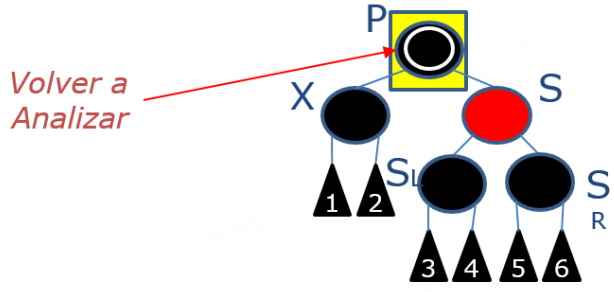
Corrección: Rotar

Espejo del anterior

Eliminación – ***Caso 3*** *: **P**, **S** y los hijos de **S** son **NEGROS**.*
Nota: *Se asume que **X** es el hijo izquierdo de su padre **P**. Si éste fuese el hijo derecho, la izquierda y la derecha deberían ser invertidas en todos estos casos.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando





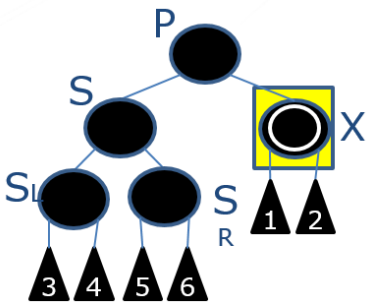
*Corrección: Cambiar color **S***

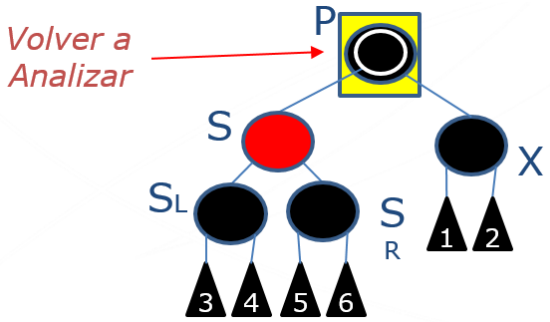
*Condición: **X** es doble negro, **S** es negro y **P** es negro.*

***P**, **S** y los hijos de **S** son negros. En este caso, simplemente cambiamos **S** a rojo. El resultado es que todos los caminos a través de **S**, precisamente aquellos que no pasan por **X**, tienen un nodo negro menos. El hecho de borrar el padre original de **X** haciendo que todos los caminos que pasan por **X** tengan un nodo negro menos nivela el árbol. Sin embargo, todos los caminos a través de **P** tienen ahora un nodo negro menos que los caminos que no pasan por **P**, así que la propiedad 5 aún no se cumple. Para corregir esto, hacemos el proceso de reequilibrio en **P**, empezando en el **caso 1**.*

Eliminación – ***Caso 3** : **P**, **S** y los hijos de **S** son **NEGROS**.*
***Nota:** Se asume que **X** es el hijo derecho de su padre **P**.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando



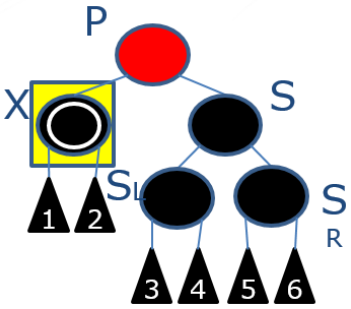


*Corrección: Cambiar color **S***

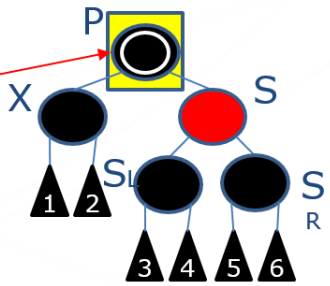
Espejo del anterior

Eliminación – ***Caso 4*** *: **S** y sus hijos son **NEGROS**, pero **P** es **ROJO***
Nota: *Se asume que **X** es el hijo izquierdo de su padre **P**. Si éste fuese el hijo derecho, la izquierda y la derecha deberían ser invertidas en todos estos casos.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando



Volver a Analizar



*Corrección: Cambiar color **S** y **P***

*Condición: **X** es doble negro, **S** es negro y **P** es rojo, **SL** es negro y **SR** es negro.*

***S** y sus hijos de son negros, pero **P** es rojo. En este caso, simplemente intercambiamos los colores de **S** y **P**. Esto no afecta al número de nodos negros en los caminos que no van a través de **S**, pero añade uno al número de nodos negros a los caminos que van a través de **X**, compensando así el borrado del nodo negro en dichos caminos.*

Eliminación – ***Caso 4*** *: **S** y sus hijos son **NEGROS**, pero **P** es **ROJO***
Nota: *Se asume que **X** es el hijo derecho de su padre **P**.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando

*Corrección: Cambiar color **S** y **P***

Espejo del anterior

Eliminación – ***Caso 5*** *: **S** es **NEGRO**, su hijo iz. es **ROJO**, el der. **NEGRO**, y **X** es hijo izquierdo de su padre.*
Nota: *Se asume que **X** es el hijo izquierdo de su padre **P**. Si éste fuese el hijo derecho, la izquierda y la derecha deberían ser invertidas en todos estos casos.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando

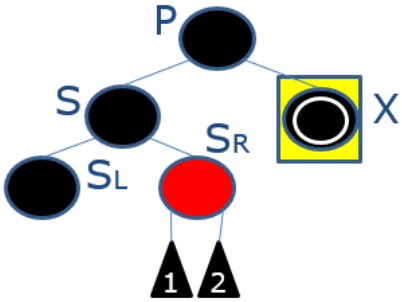
*Corrección: Cambiar color y rotar **SL** y **S***

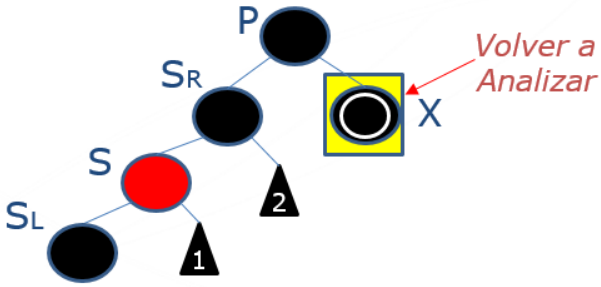
*Condición: **X** es doble negro, **S** es negro y **P** es rojo, **SL** es negro y **SR** es negro.*

***S** es negro, su hijo izquierdo es rojo, el derecho es negro, y **X** es el hijo izquierdo de su padre. En este caso rotamos a la derecha **S**, así su hijo izquierdo se convierte en su padre y en el hermano de **X**. Entonces intercambiamos los colores de **S** y su nuevo padre. Todos los caminos tienen aún el mismo número de nodos negros, pero ahora **X** tiene un hermano negro cuyo hijo derecho es rojo, así que caemos en el **caso 6**. Ni **X** ni su padre son afectados por esta transformación (de nuevo, por el **caso 6**, reetiquetamos el nuevo hermano de **X** como **S**).*

Eliminación – ***Caso 5** : **S** es **NEGRO**, su hijo der. es **ROJO**, el izq. **NEGRO**, y **X** es hijo derecho de su padre.*
***Nota:** Se asume que **X** es el hijo derecho de su padre **P**.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando



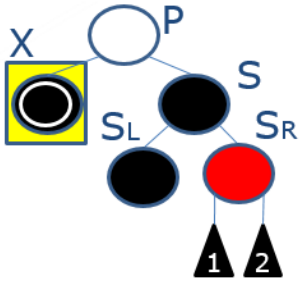


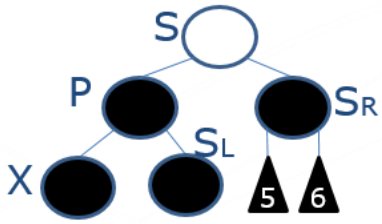
*Corrección: Cambiar color y rotar **SL** y **S***

Espejo del anterior

Eliminación – ***Caso 6** : **S** es **NEGRO**, su hijo der. es **ROJO**, el izq. **NEGRO**, y **X** es hijo izquierdo de su padre.*
***Nota:** Se asume que **X** es el hijo izquierdo de su padre **P**. Si éste fuese el hijo derecho, la izquierda y la derecha deberían ser invertidas en todos estos casos.*

Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando





*Corrección: Rotación y cambio color **S** y **P***

***S** es negro, su hijo derecho es rojo, y **X** es el hijo izquierdo de **P**, su padre. En este caso rotamos a la izquierda **P**, así que **S** se convierte en el padre de **P** y éste en el hijo derecho de **S**. Entonces intercambiamos los colores de **P** y **S**, y ponemos el hijo derecho de **S** en negro. El subárbol aún tiene el mismo color que su raíz, así que las propiedades 4 y 5 se verifican. Sin embargo, **X** tiene ahora un antecesor negro mas: o bien P se ha convertido en negro, o bien era negro y **S** se ha añadido como un abuelo negro. De este modo, los caminos que pasan por **X** pasan por un nodo negro mas. Mientras tanto, si un camino no pasa por **X**, entonces hay dos posibilidades:*

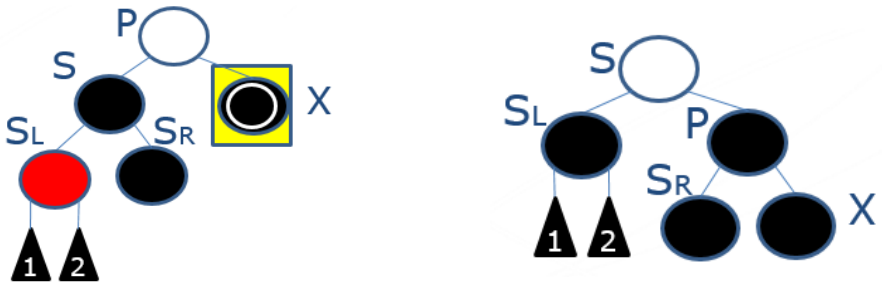
- *Éste pasa a través del nuevo hermano de **X**. Entonces, éste debe pasar por **S** y **P**, al igual que antes, y tienen sólo que intercambiar los colores. Así los caminos contienen el mismo número de nodos negros.*
- *Éste pasa por el nuevo tío de **X**, el hijo derecho de **S**. Éste anteriormente pasaba por **S**, su padre y su hijo derecho, pero ahora sólo pasa por **S**, el cual ha tomado el color de su anterior padre, y por su hijo derecho, el cual ha cambiado de rojo a negro. El efecto final es que este camino va por el mismo número de nodos negros.*

De cualquier forma, el número de nodos negros en dichos caminos no cambia. De este modo, hemos restablecido las propiedades 4 y 5. El nodo blanco en diagrama puede ser rojo o negro, pero debe tener el mismo color tanto antes como después de la transformación.

Eliminación – ***Caso 6** : **S** es **NEGRO**, su hijo iz. es **ROJO**, el der. **NEGRO**, y **X** es hijo derecho de su padre.*

Nota: *Se asume que **X** es el hijo derecho de su padre **P**.*

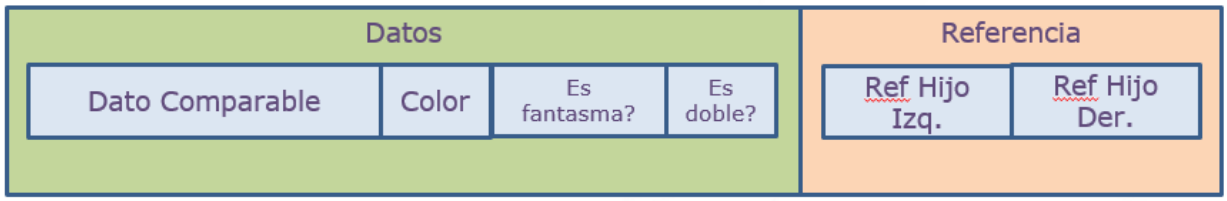
Nota: El doble negro puede ser originado por un borrado directo y en tal caso será un nodo fantasma doble negro(no tendrá hijos). O puede ser el resultado de la resolución de otro caso que hay que seguir analizando



*Corrección: Rotación y cambio color **S** y **P***

Espejo del anterior

Estructura de Nodo del árbol Rojo-Negro



AVL vs. Rojo - Negro

	Operaciones	AVL	Rojo-Negro
Pocos Datos	Búsqueda	Mas rápido, menor profundidad	
	Inserción		Mas rápido, en promedio menos rotaciones.
	Eliminación		Mas rápido, en promedio menos rotaciones.
Muchos Datos	Búsqueda	Mas rápido, menor profundidad	
	Inserción	Mas rápido, el cuello de botella es la búsqueda	
	Eliminación	Mas rápido, en promedio menos rotaciones	El peor caso es más rápido que el AVL