Arboles Rojinegros

Juan Esteban Murcia y David Moreno

November 26, 2018

Arbol Roji-Negro

Un arbol roji-negro es un arbol binario de busqueda equilibrado, fue creado por Rudolf Bayer en 1972,muchas estructuras en geometria computacional se pueden basan en este tipo.

Propiedades

1. Todo nodo es o rojo o negro.
2. La raíz del árbol es negra
3. Un nodo rojo no puede tener padre rojo o hijos rojos
4. Cualquier camino desde la raíz hasta un hijo de una hoja (Nil) tiene la misma cantidad de nodos negros
5. Los hijos de las hojas (Nil) son negros
6. La altura negra de un nodo x, bh(x), es el número de nodos negros que hay en cualquier camino de él a una hoja que desciende de él.

Ejemplo

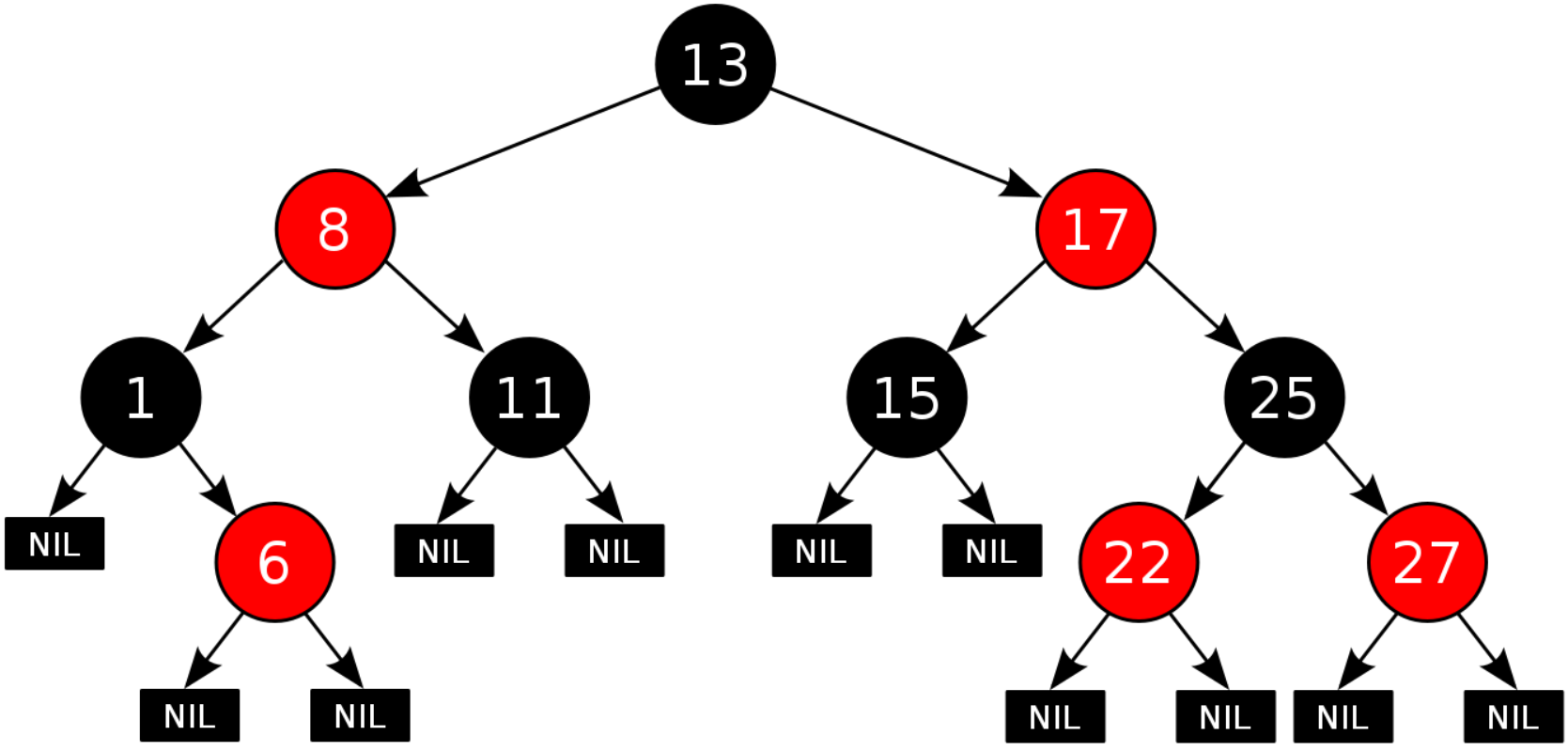


Figure 1:

Demostraciones

- teorema1:un arbol A Roji-Negro de n nodos tiene una altura de a lo sumo $2\lg(n+1)$
- lema:Si x es un nodo en A, entonces, el subárbol con raíz en x tiene a al menos $2^{bh(x)}-1$ nodos internos.

Demostracion:Sea x un nodo en A, por inducción sobre $bh(x)$

- Si $bh(x)=0$, entonces, x es hoja y no hay nodos en el subárbol con raíz en él.
- Por otro lado, $2^{bh(x)}-1=0$
- Supongamos que $bh(x)>0$, si x es interno y tiene dos hijos. Es inmediato que cualquier hijo y de x cumple con lo siguiente:

$C(y)=$ rojo, entonces, $bh(y)=bh(x)$

$C(y)=$ negro, entonces, $bh(y)=bh(x)-1$

Por hipótesis de inducción, el número de nodos descendientes de x es al menos:

- Nodos internos(dere(x)) + 1+Nodos internos(izq(x))
  $<=(2^{bh(x)} - 1) + (2^{bh(x)} - 1) + 1 =$
  $-1 + 2(2^{bh(x)-1})=2^{bh(x)} - 1$

Demostracion Teorema1

Sea h la altura del árbol rojinegro. Al menos la mitad de los nodos en un camino de la raíz a una hoja , sin incluir la raíz, deben ser negros. La altura negra del árbol debe ser al menos h/2 Por lo anterior, $n >= 2^{h/2} - 1$ y de aqui tenemos que $log(n + 1) >= h/2$ luego $h <= 2log(n + 1)$

Analisis de operaciones

- Busqueda la operacion de busqueda toma $O(log(n))$ en el peor de los casos
- insercion tiene una complejidad de $O(logn)$.
- eliminacion

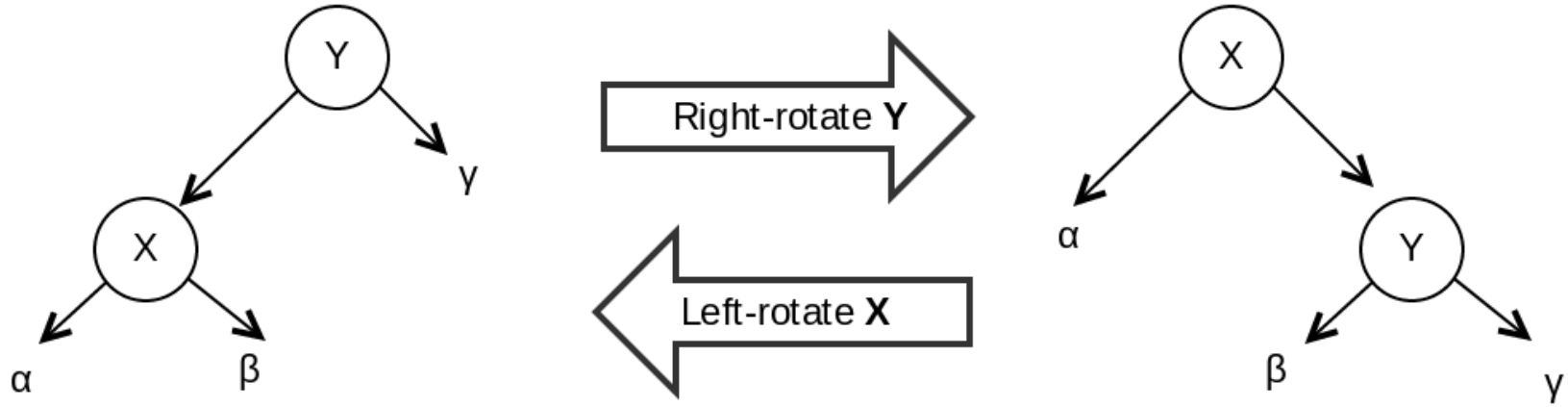
tambien toma $O(log(n))$

Rotaciones

como queremos que el arbol mantenga estas propiedades mencionadas entonces vamos a necesitar dos tipos de Rotaciones,left y Right.

Left Rotation

- rotar alrededor del enlace de x a y
- hacer y la nueva raiz del sub-arbol
- x se convierte en el hijo izquierdo de y
- el hijo izquierdo de y se convierte en el hijo derecho de x



Example: LEFT-ROTATE

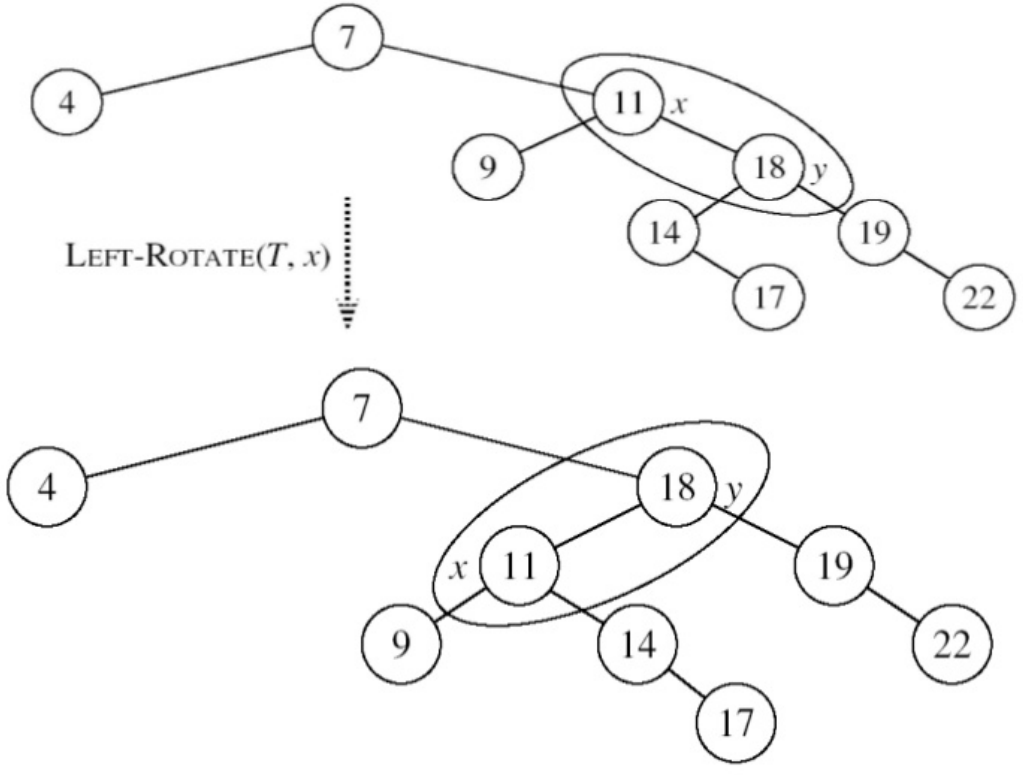


Figure 3:

Right Rotation

- rotar alrededor del enlace de y a x
- hacer x la nueva raiz del sub-arbol
- y se convierte en el hijo derecho de x
- el hijo derecho de x se convierte en el hijo izquierdo de y

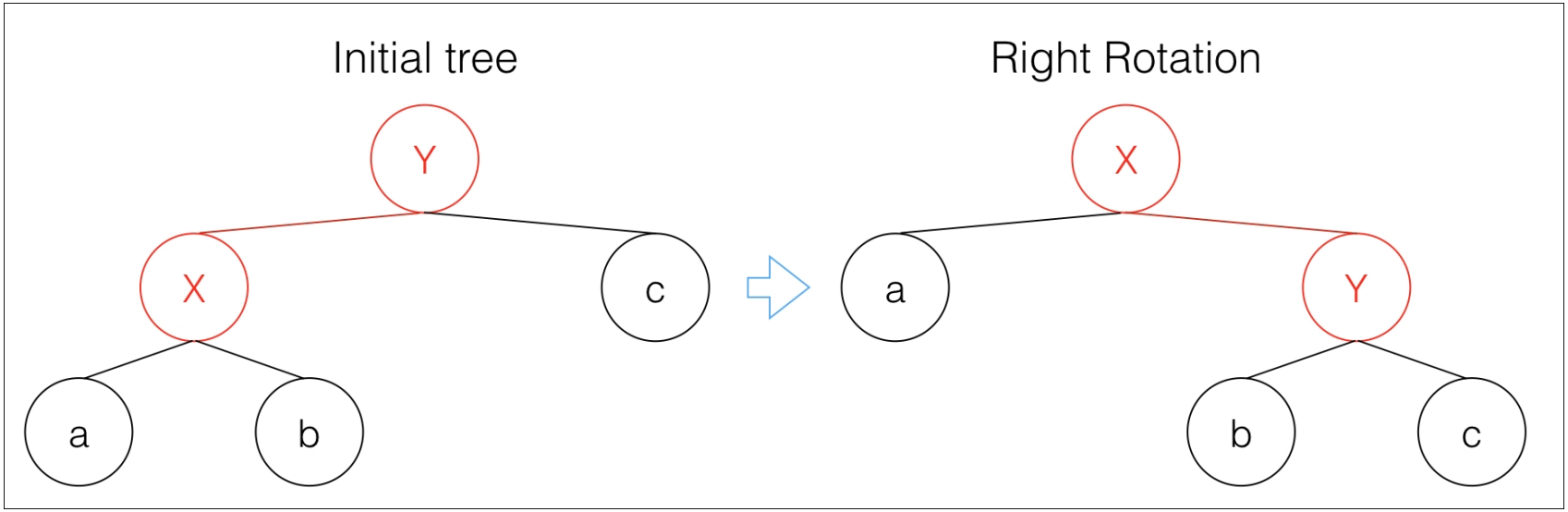


Figure 4:

insercion

- Al insertar nuevos elementos en árboles rojinegros, en tiempo O(log n), se debe garantizar que el árbol resultante continúe siendo rojinegro
- En los árboles rojinegros las hojas son vacías y negras
- Los nuevos elementos se colocan como padres de hojas
- Cada nuevo elemento se coloca como una estructura del árbol binario
- Como las hojas deben ser negras, el nodo que contiene la llave a insertarse se colorea como rojo
- No se aumenta la altura negra del árbol
- La única propiedad que puede violarse la 3

caso 1

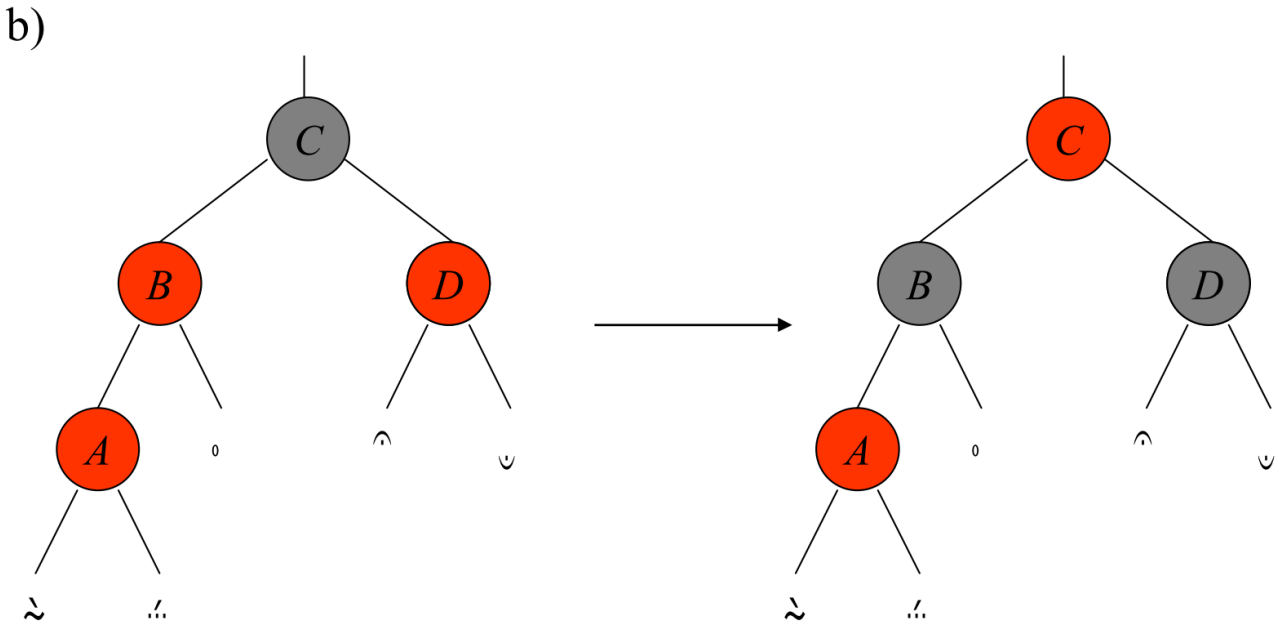


Figure 5:

Si x es rojo y su padre y su tío son rojos, re-coloreamos a estos de negro y al padre de ambos de rojo. Como A es RN, el abuelo era negro. Al hacer los cambios, las alturas negras no se modifican. Trasladamos el problema hacia el abuelo de x y llamamos a recursion sobre este

caso 2 y 3

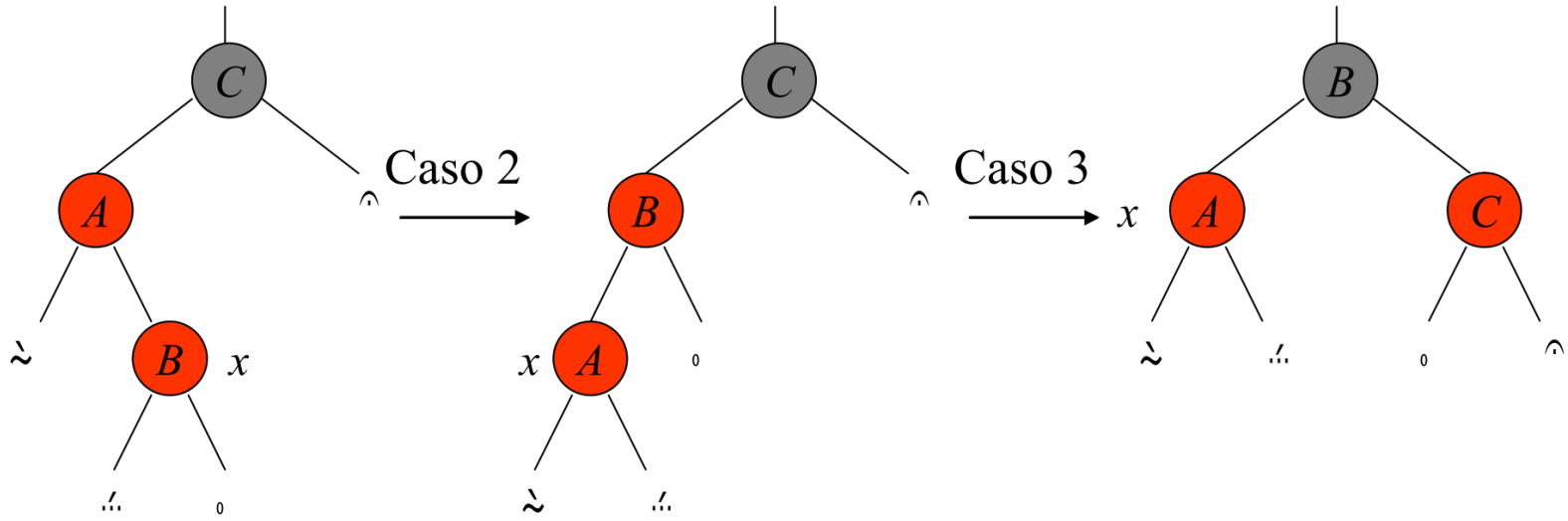


Figure 6:

caso 2

- x y su padre son rojos y su tio es negro, x hijo derecho del padre.

La rotación izquierda sobre el padre reduce el problema al caso 3.

caso 3

- x y su padre son rojos y su tío es negro, x es hijo izquierdo del padre
- Re-coloreamos al padre de negro,y al abuelo de rojo, realizamos rotación a derecha sobre el abuelo.
- Las alturas negras no se modifican y el abuelo quedaría negro, con ambos hijos rojos.
- Termina la ejecucion

Eliminacion

- Si el nodo eliminado fuese rojo, entonces, las alturas negras no cambian y, en tal caso, termina
- Si el nodo eliminado ,y, fuese negro, entonces las ramas que pasen por y tienen un nodo negro menos, lo que viola la propiedad de los árboles rojinegros. En este caso es necesario ajustar colores y realizar rotaciones.

caso 1

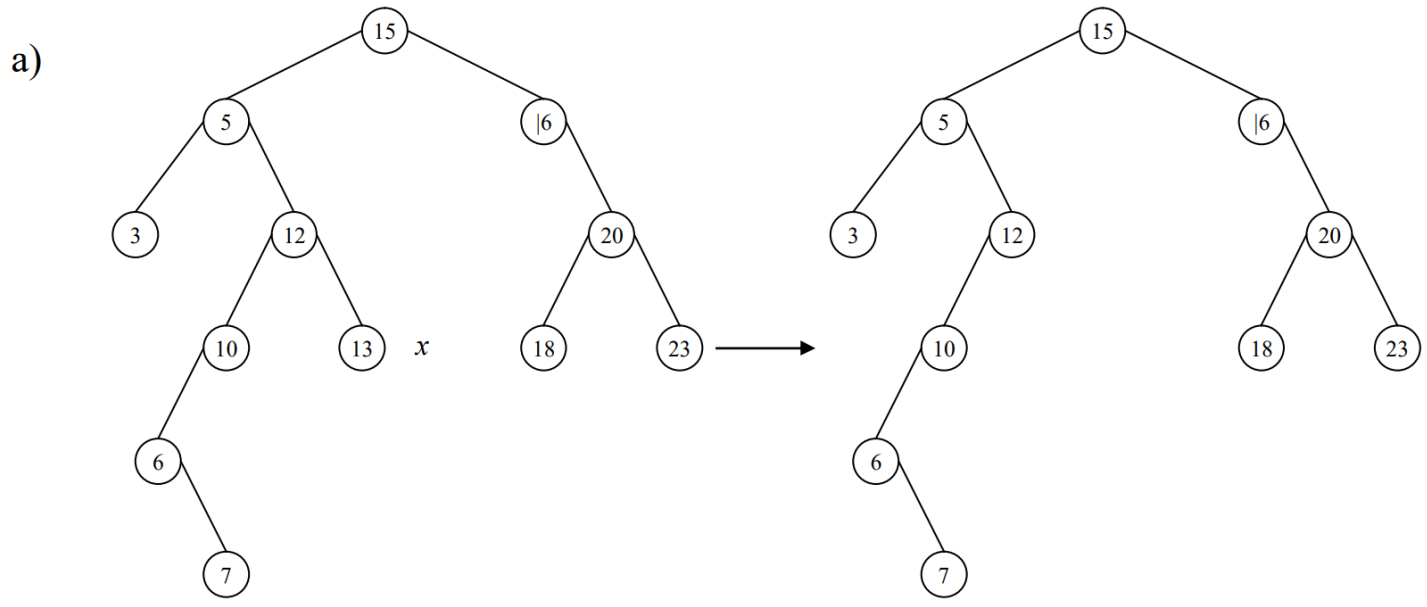


Figure 7:

queremos eliminar el 13

caso 2

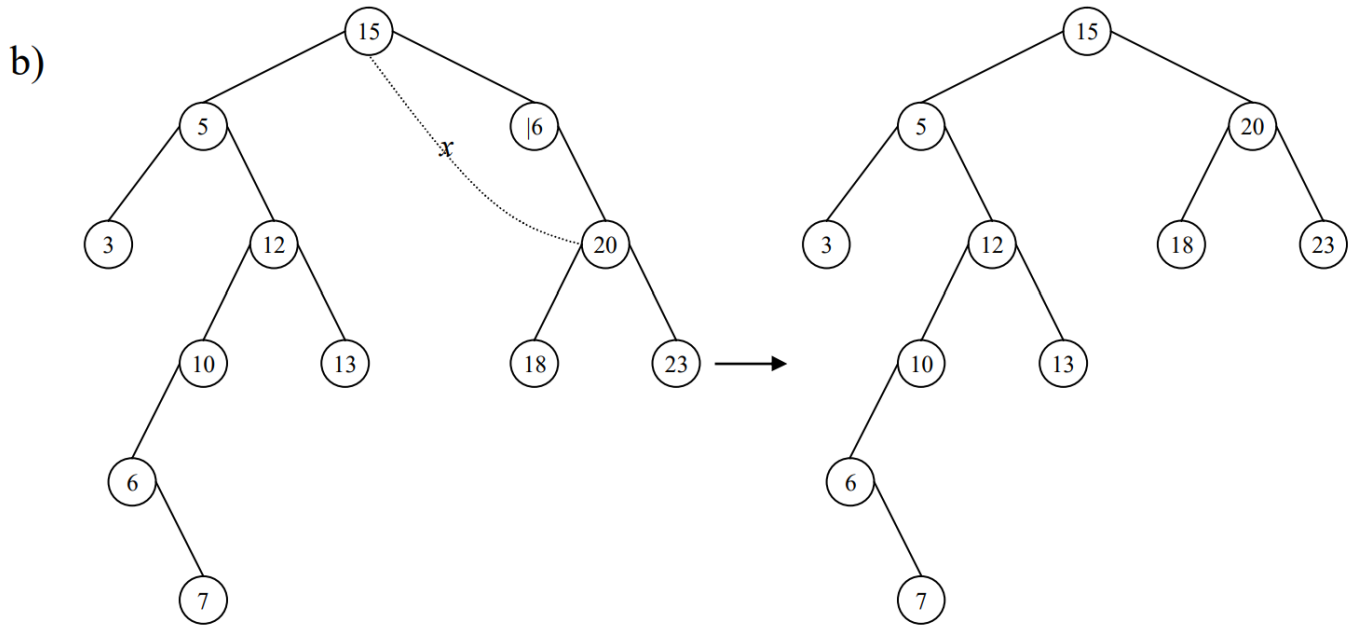


Figure 8:

queremos eliminar el 6

caso3

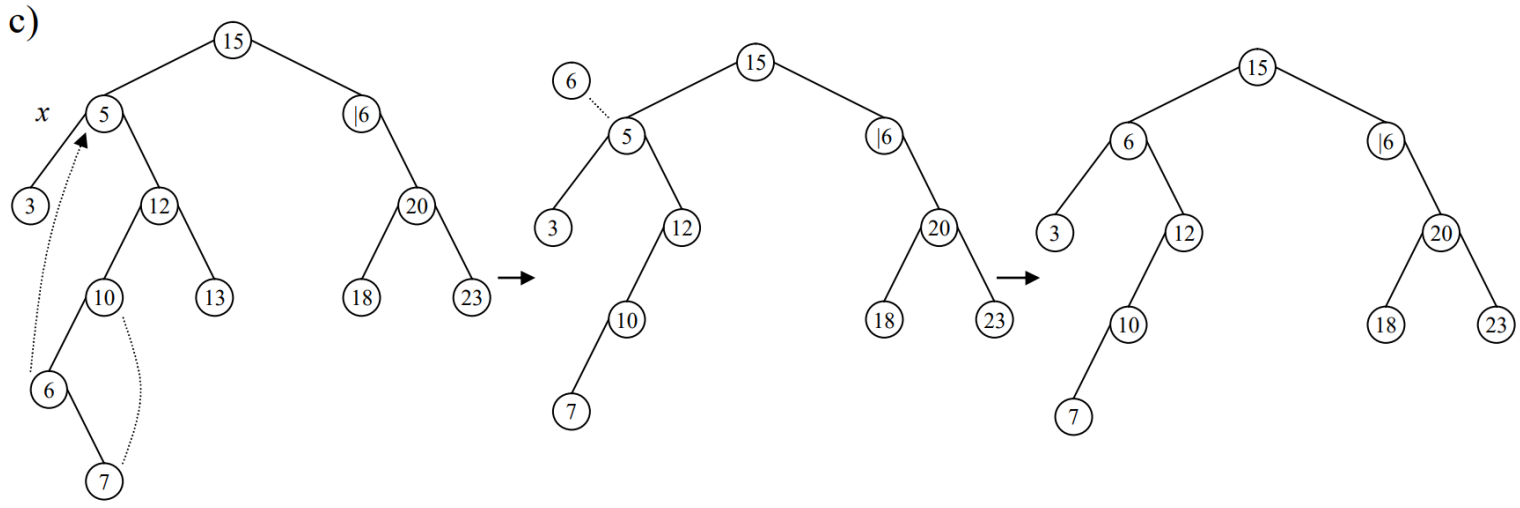
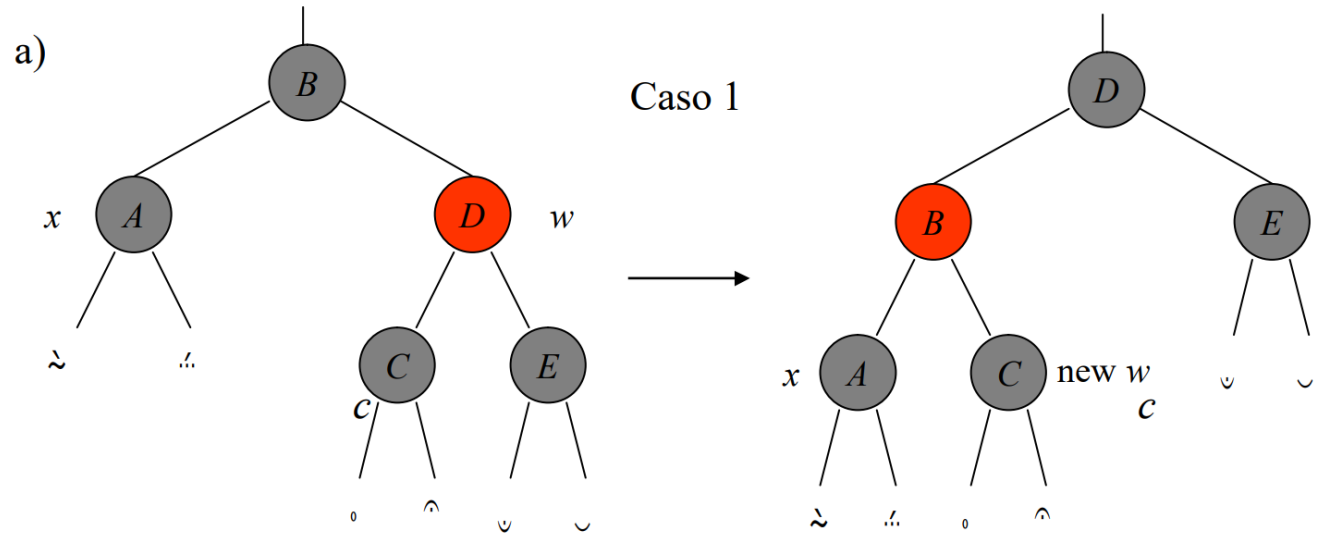


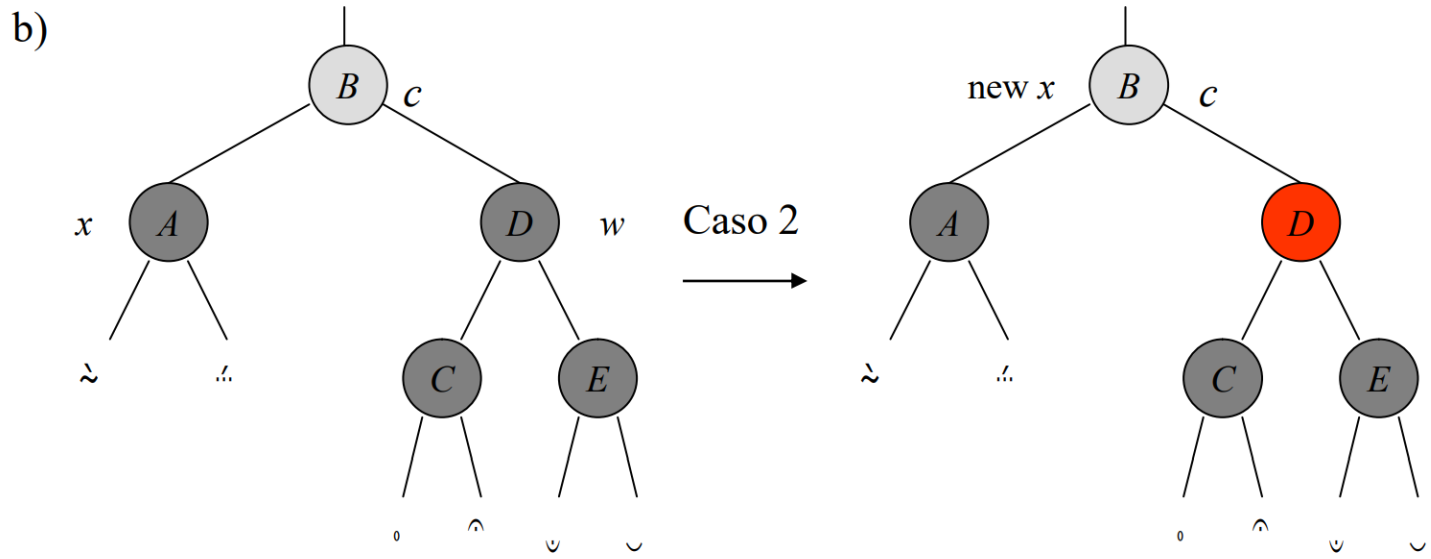
Figure 9:

queremos eliminar el 5

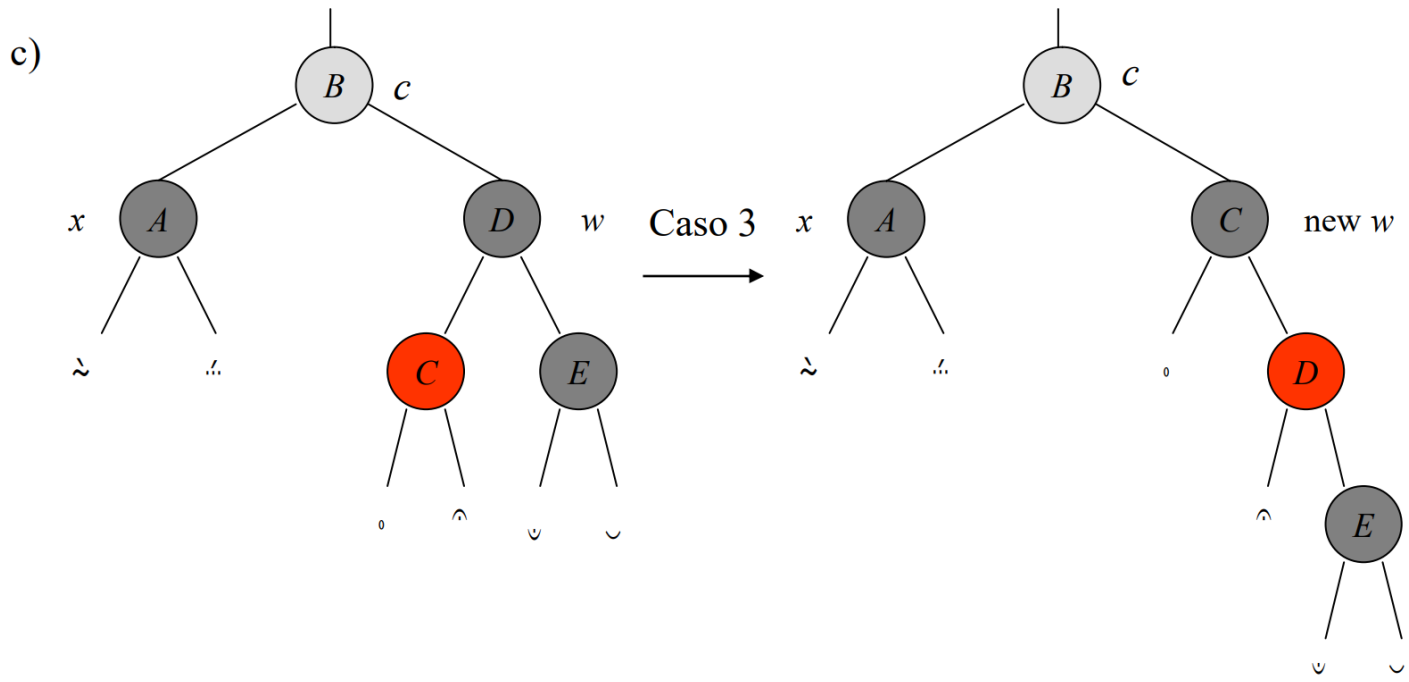
Ajustar Eliminacion



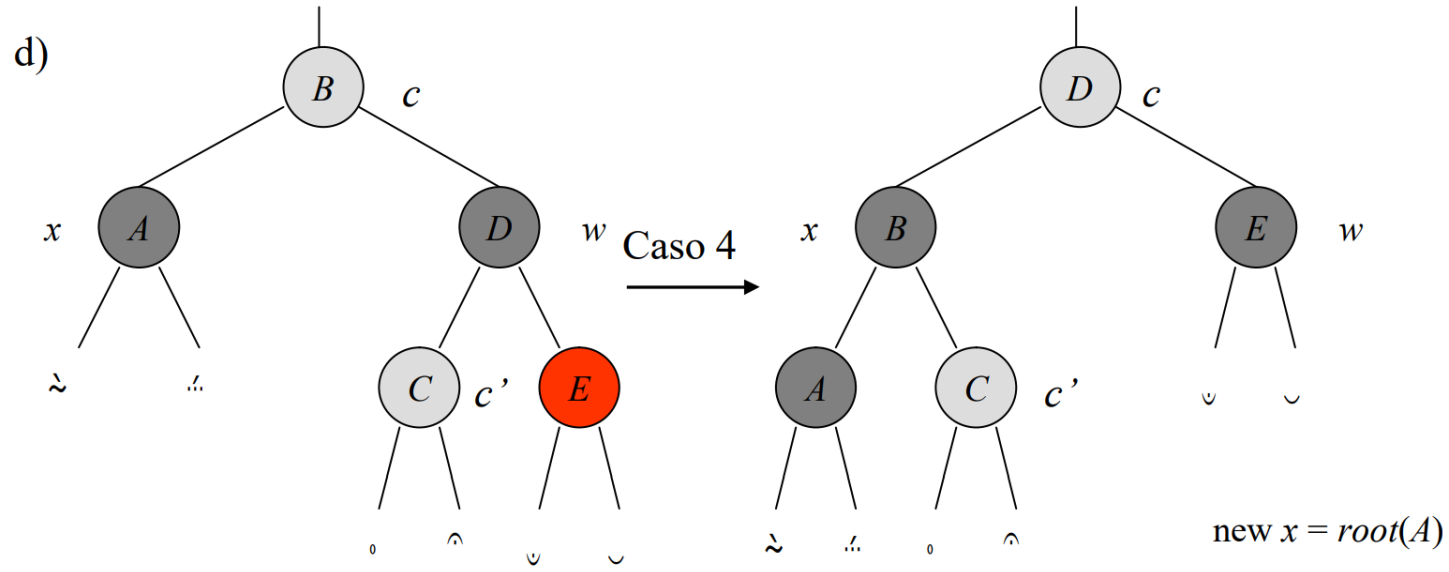
x es negro y su hermano rojo. Al hermano se le pinta de negro, al padre de rojo y rotamos a la izquierda. Se llega a 2). Al ejecutarse 2) se concluirá el ciclo principal pues x es rojo.



x y su hermano son negros, así como ambos hijos del hermano. Al hermano se le pinta de rojo y se fuerza al padre a llevar marca negra de “acarreo”



x y su hermano son negros, el hijo derecho del hermano es negro y el izquierdo rojo. Al hijo izquierdo del hermano se le pinta de negro y al hermano de rojo. Se rota a la derecha para obtener el caso 4).



x y su hermano son negros, el hijo derecho del hermano es rojo y el izquierdo es negro. Al padre se le pinta de negro, al hermano se le pone el color al que tenía el padre, y al hijo derecho del hermano también camba a negro. Se rota a la izquierda para equilibrar alturas negras

Concluciones

la complejidad de insercion y eliminacion en tiempo para el peor caso está acotada por $O(log(n))$

Avl vs Reed Black Tree

Cuando usar cual?:

- Avl genera arboles con un balanceo mas optimo que los que generan los RB,sin embargo,avl presenta mas rotaciones en la insercion y eliminacion que los RBT.
- Concluyendo, si se necesita el arbol optimamente balanceado sin tanta importancia en el costa computacional, se recomienda usar AVL, pero si lo que se necesita es un arbol balanceado, pero que tenga una insercion y una eiminacion mas rapida, se recomienda usar RB trees.