Árboles ordenados balanceados

Árboles Rojinegros

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

28 de julio de 2021

Definiciones

Árboles Rojinegros

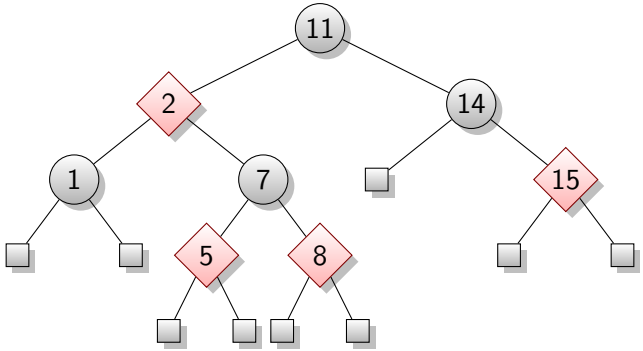
Los árboles rojinegros cumplen con las propiedades siguientes:

1. Cada nodo es *rojo* o ***negro***.
2. La raíz es **negra**.
3. Cada hoja `null` es **negra**.
4. *Si un nodo es rojo, sus dos hijos son **negros**.
5. Para cada nodo, todos los caminos del nodo a sus hojas descendientes contienen el mismo número de nodos **negros**.

Definición

Al máximo número de nodos negros desde un nodo hasta sus hojas se le conoce como *altura negra*.

Ejemplo

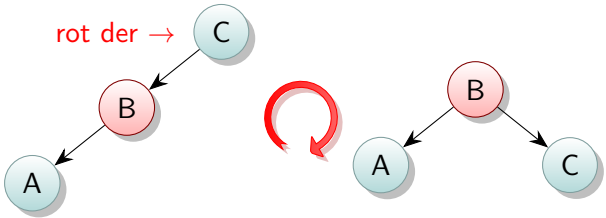


Rotaciones

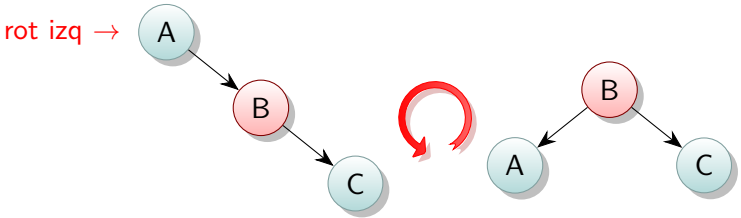
- Las operaciones siguientes se utilizarán para recuperar el balance del árbol cuando este se rompe.

Temas

Rotación a la derecha

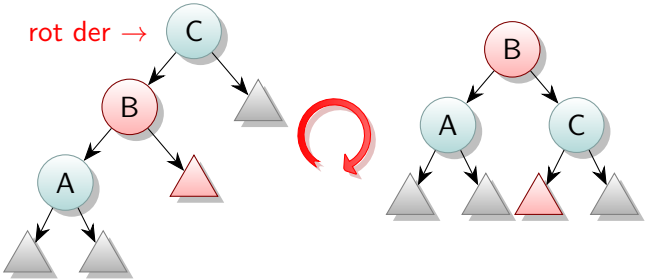


Rotación a la izquierda



Temas

Rotación a la derecha (LL)



Inserción

Insertar

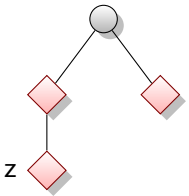
- El dato se inserta como en cualquier árbol binario ordenado (en una hoja).
- Al insertar el dato, el nuevo nodo siempre se pinta de *rojo* y tendrá como hijos dos nodos vacíos (negros).
- Sólo hay problemas si el nodo insertado queda como hijo de un nodo rojo, lo cual viola la propiedad 4. [1]

[1] Si un nodo es rojo, sus dos hijos son negros.

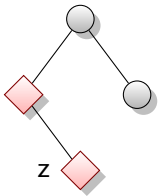
Tres casos

- Se dice que hay siete casos, pero realmente en uno no se hace nada y tres son simétricos de los otros tres.

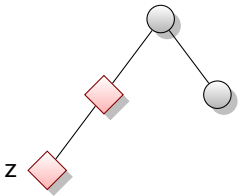
Caso 1



Caso 2
(Rotación doble)



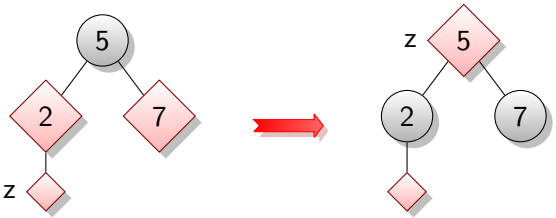
Caso 3



Algoritmo de inserción (reparar propiedad 4)

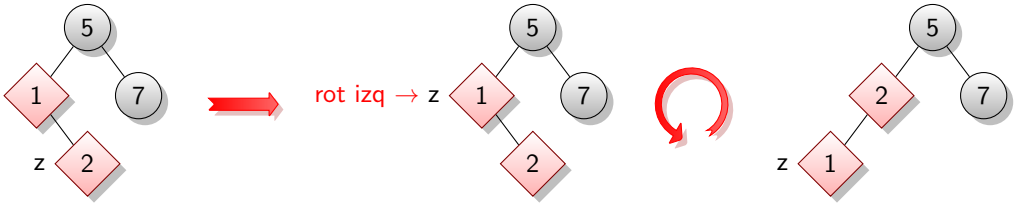
1: $z \leftarrow$ Nodo agregado (rojo).
2: **while** El padre de z no sea negro **do**
3: **if** *Caso 1:* El padre y el tío del nodo nuevo son rojos. **then**
4: Pintar al padre y al tío de negro.
5: Pintar al abuelo de rojo.
6: $z \leftarrow \mathtt{abuelo}(z)$.
7: **else**
8: **if** *Caso 2:* Padre rojo, tío negro y z hijo derecho. **then**
9: $z \leftarrow \mathtt{padre}(z)$
10: Realizar rotación izquierda sobre z.
11: *Caso 3:* Padre rojo, tío negro y z hijo izquierdo.
12: Pintar al padre de negro.
13: Pintar al abuelo de rojo.
14: Realizar rotación a la derecha sobre el abuelo de z.
15: Pintar de negro la raíz.

Caso 1: El padre y el tío del nodo nuevo son rojos.



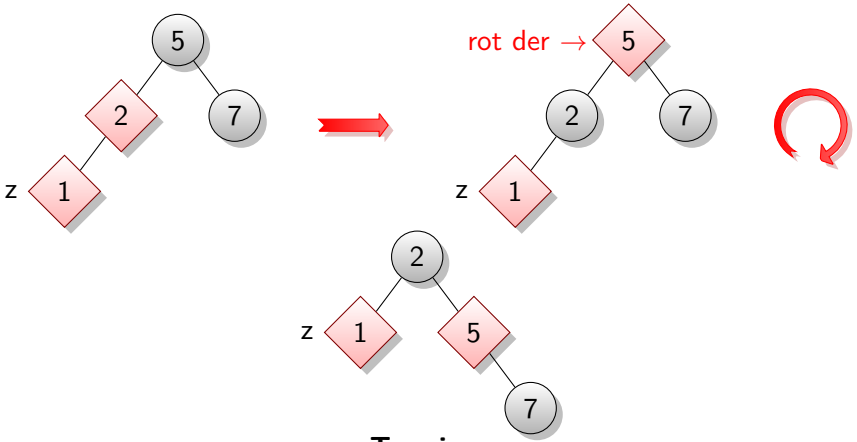
Caso 2: Padre rojo, tío negro y z hijo derecho.

Caso 2
(Rotación doble)



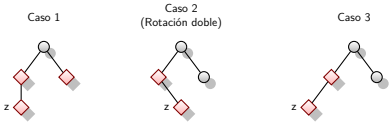
Continúa como el caso siguiente.

Caso 3: Padre rojo, tío negro y z hijo izquierdo.



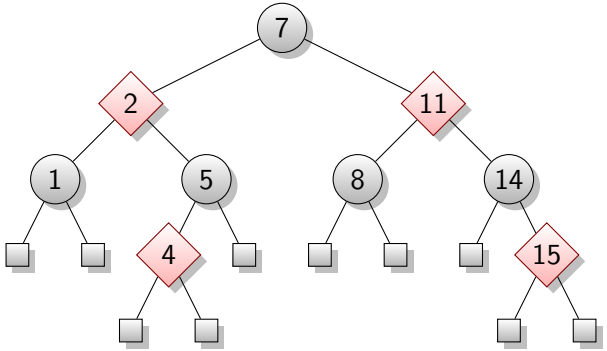
Ejercicio de inserción

Insertar en este orden: 11, 2, 14, 1, 7, 15, 5, 8, 4.



Resultado

- Insertar en este orden: 11, 2, 14, 1, 7, 15, 5, 8, 4.



Pseudocódigo

```
1   while(color(z.padre()) == ROJO) {
2     if(z.padre() == z.padre().padre().hijol()){
3       y = z.padre().padre().hijoD();
4       if(color(y) == ROJO) {
5         z.padre().color(NEGRO);                    // Caso 1
6         y.color(NEGRO);
7         z.padre().padre().color(ROJO);
8         z = z.padre().padre();
9       } else {
10        if(z == z.padre().hijoD()) {
11          z = z.padre();                           // Caso 2
12          z.rotalzquierda();
13        }
14        z.padre().color(NEGRO);                    // Caso 3
15        z.padre().padre().color(ROJO);
16        z.padre().padre().rotaDerecha();
17      }
18    } else {
19      // Igual pero intercambiando "izquierdo" y "derecho".
20    }
21  }
22  if(z != null) { while(z.padre() != null) z = z.padre();raíz = z; }
23  raíz.color(NEGRO);
```

Remoción

Remover

- El dato se remueve como en cualquier árbol binario ordenado.
- Necesitaremos la referencia a la hoja removida y a su padre.
- Sólo hay problemas si el nodo removido era negro, lo cual viola la propiedad 5. [2]
- Sean:

  p: el padre del nodo que causa el problema (inicialmente el padre del nodo removido).

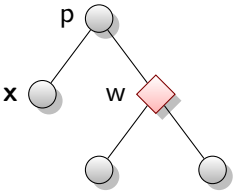
  x: el hijo de p que recibe el color $negro$ del nodo removido (nodo *doble negro*).

  w: el hermano de x.

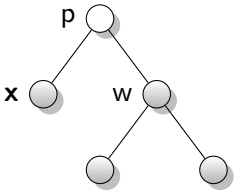
[2] Para cada nodo, todos los caminos del nodo a sus hijos descendientes contienen el mismo número de nodos **negros**.

4 Casos

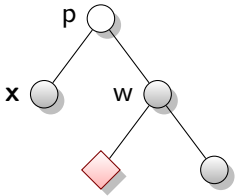
Caso 1



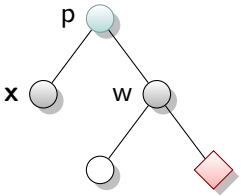
Caso 2



Caso 3

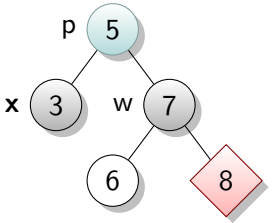


Caso 4 (resuelve)



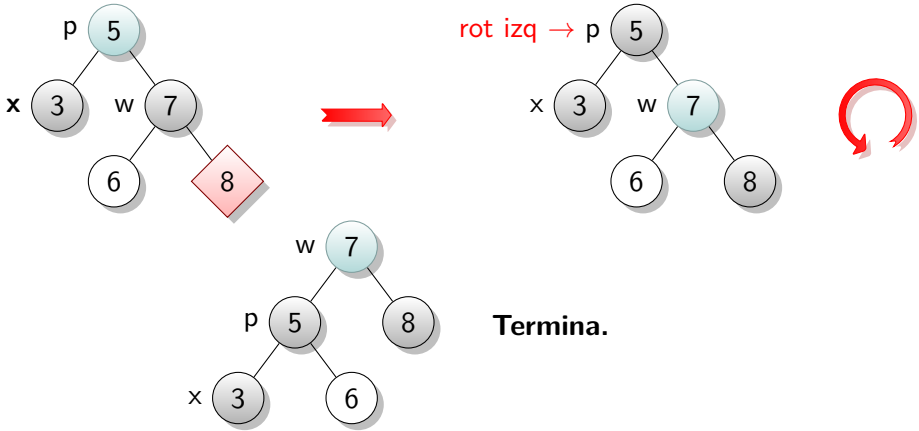
Caso 4

x	Negro
hermano	Negro
sobrino derecho	Rojo

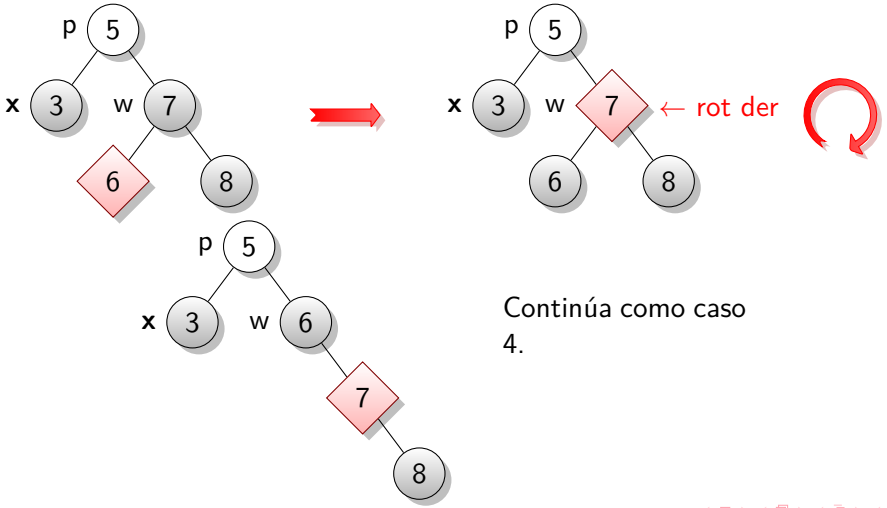


1. Intercambiar colores de p y w
2. Rotar a la izquierda sobre p.
3. Pintar al sobrino de negro.

Caso 4

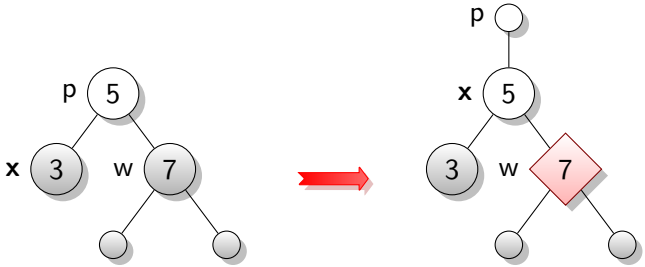


Caso 3

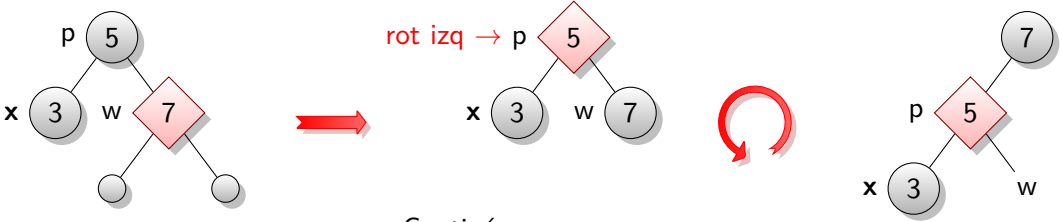


Continúa como caso 4.

Caso 2



Caso 1

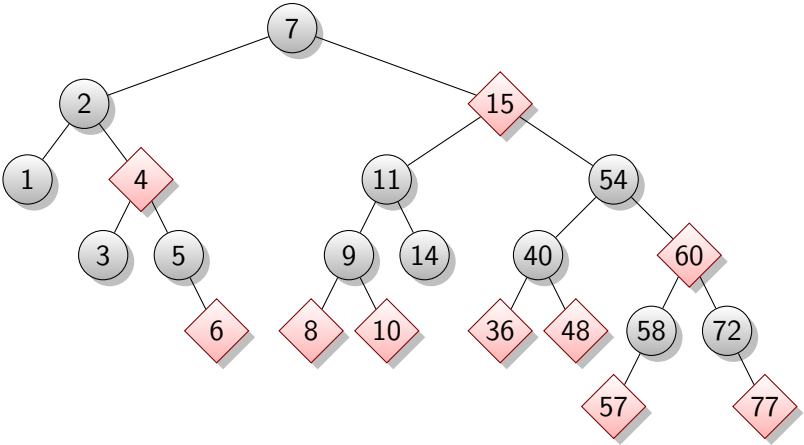


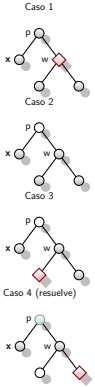
Continúa como caso 2 o 3.

Pseudocódigo

```
1  while(x != raíz && color(x) == NEGRO) {
2    if(x == p.getHijoI()) {
3      w = p.getHijoD();
4      if(color(w) == ROJO) {                                    // Caso 1
5        w.setColor(NEGRO);  p.setColor(ROJO);
6        p.rotalzquierda(); if(p == raíz) { raíz = p.getPadre(); }
7        w = p.getHijoD();
8      }
9      if(color(w.getHijoI()) == NEGRO && color(w.getHijoD()) == NEGRO) { // Caso 2
10       w.setColor(ROJO);
11       x = p;
12       p = x.getPadre();
13     } else {
14       if (color(w.getHijoD()) == NEGRO) {                     // Caso 3
15         w.getHijoI().setColor(NEGRO);  w.setColor(ROJO);
16         w.rotaDerecha();  //if(w == raíz) {raíz = w.getPadre(); }
17         w = p.getHijoD();
18       }
19       w.setColor(color(p));                                   // Caso 4
20       p.setColor(NEGRO);  w.getHijoD().setColor(NEGRO);
21       p.rotalzquierda();  if(p == raíz) {raíz = p.getPadre(); }
22       break;
23     }
24   } else { // Igual pero intercambiando "izquierdo" y "derecho". }
25 }
26 if(x != null) x.setColor(NEGRO);
```

Ejercicio: remover nodos





Bibliografía I


Cormen, Thomas H. y col. (2009). *Introduction to Algorithms*. 3rd. The MIT Press.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual