Estructura de Datos

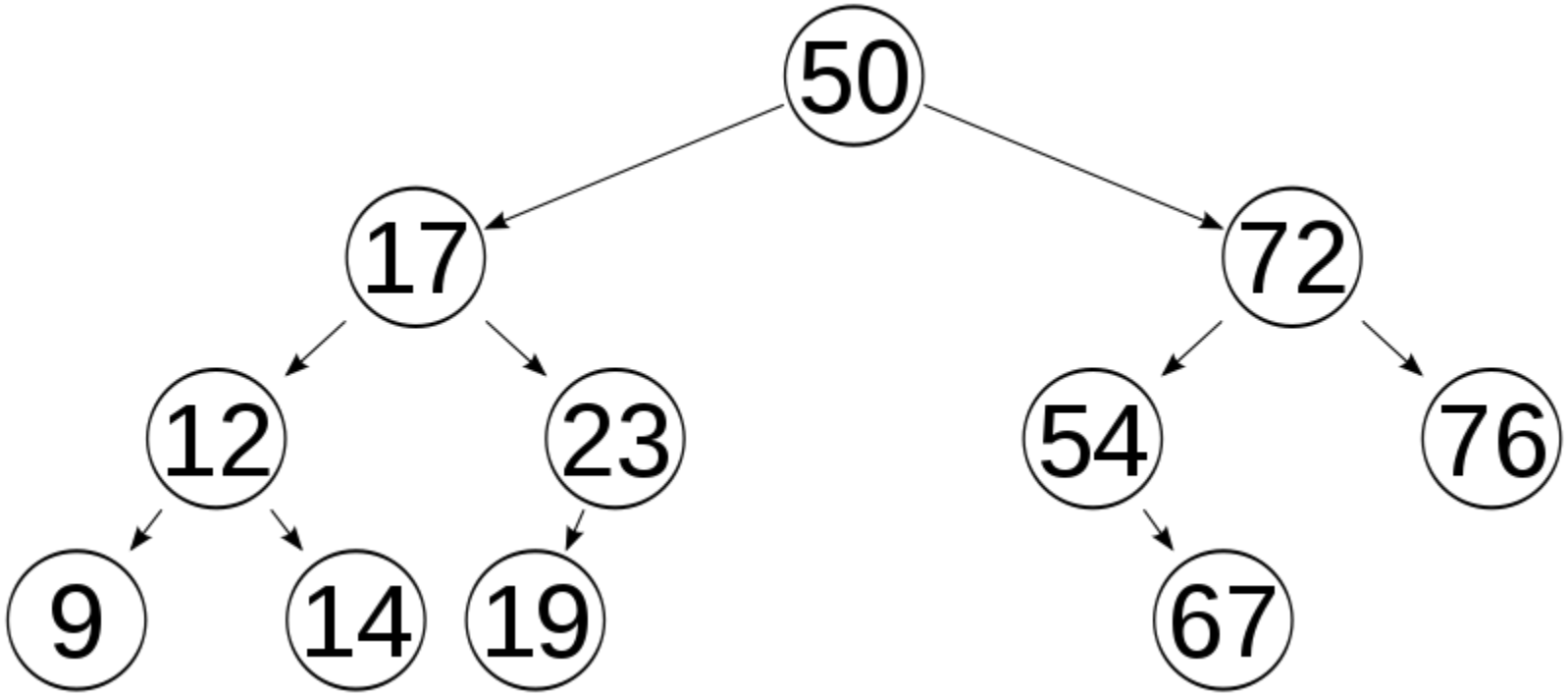
Semana 10

Logro de la sesión

Al finalizar la unidad, el estudiante identifica, analiza y resuelve problemas algorítmicos que usan el tipo abstracto de datos: árboles AVL, desarrollando funciones/métodos que usen esas estructuras.

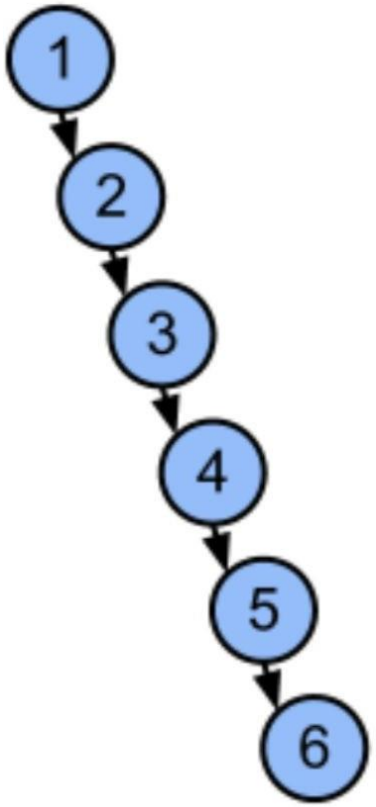
Resumen - ABB

Ventaja: Insert, remove y search ~O(h) =**O($log_2 n$).**



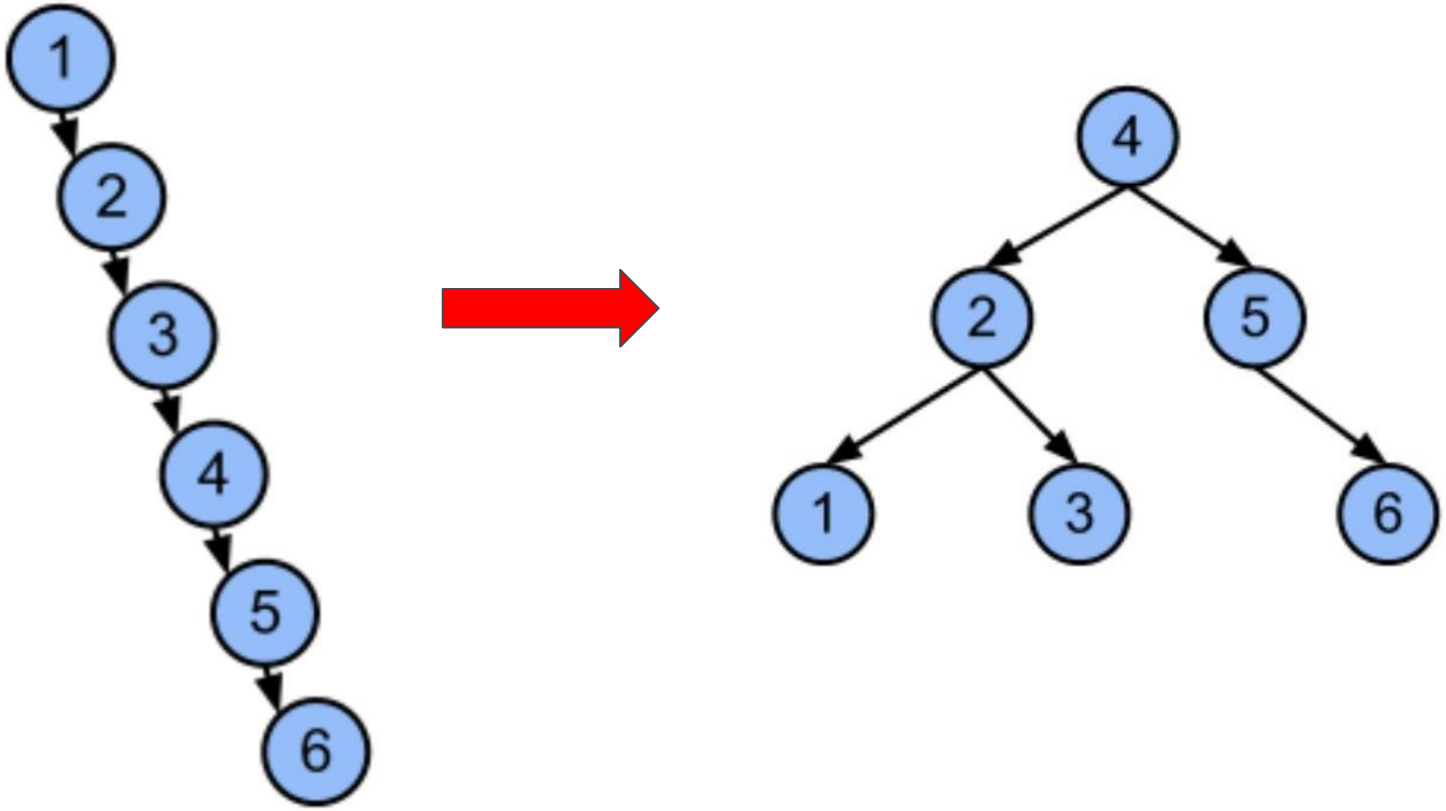
Resumen - ABB

Si h (height) ≃ n ⇒**O(h)=O(n)**



Resumen - ABB

Solución: Equilibrado de un ABB



Arboles equilibrados

- Los **árboles equilibrados** son **árboles BB** que imponen restricciones estructurales para garantizar (o tender a) que su altura sea **logarítmica**.
- Para ello añaden **etapas extra** a las operaciones de inserción y borrado (y a veces al acceso)
- **Árboles AVL**: Imponen que para todo nodo la diferencia de altura entre los subárboles izquierdo y derecho no sea mayor que uno.
- **Árboles Rojo-Negro**: Los nodos se clasifican como rojos o negros, y se cumple:
  - Los hijos de un nodo rojo son negros
  - Todo camino de la raiz a una hoja pasa por el mismo número de nodos negros.
- **Splay Trees**: Cada vez que se accede a un nodo se "eleva" en el árbol pasando a ser la raiz (equilibrado "promedio")

Arboles equilibrados

- Los **árboles AVL** son **árboles BB** donde todo nodo cumple la propiedad de **equilibrado AVL**:

  La altura del subárbol izquierdo y del derecho no se diferencian en más de uno.

- Se define **factor de equilibrio** de un nodo como:

  Fe(nodo) = altura(derecho) – altura(izquierdo)

- En un árbol AVL el factor de equilibrio de todo nodo es -1, 0 ó +1.
- Tras la inserción o borrado de un elemento, sólo los **ascendientes** del nodo pueden sufrir un cambio en su factor de equilibrio, y en todo caso sólo en una unidad.
- Se añade una etapa donde se recorren los ascendientes. Si alguno está desequilibrado (+2 o -2) se vuelve a equilibrar mediante operaciones denominadas **rotaciones**.

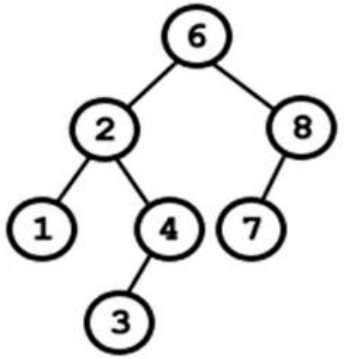
Árboles Binarios de Búsqueda AVL

¿Qué Son? → ABB pero Balanceados

¿Por Qué AVL? → De sus Creadores: Georgii Adelson-Velskii Y Yevgeniy Landis

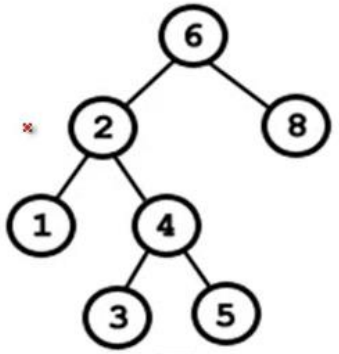
¿Qué significa Balanceado (Equilibrado)? → Para todo nodo la altura de sus subárboles izquierdo y derecho pueden diferir a lo máximo en 1.

Árbol A



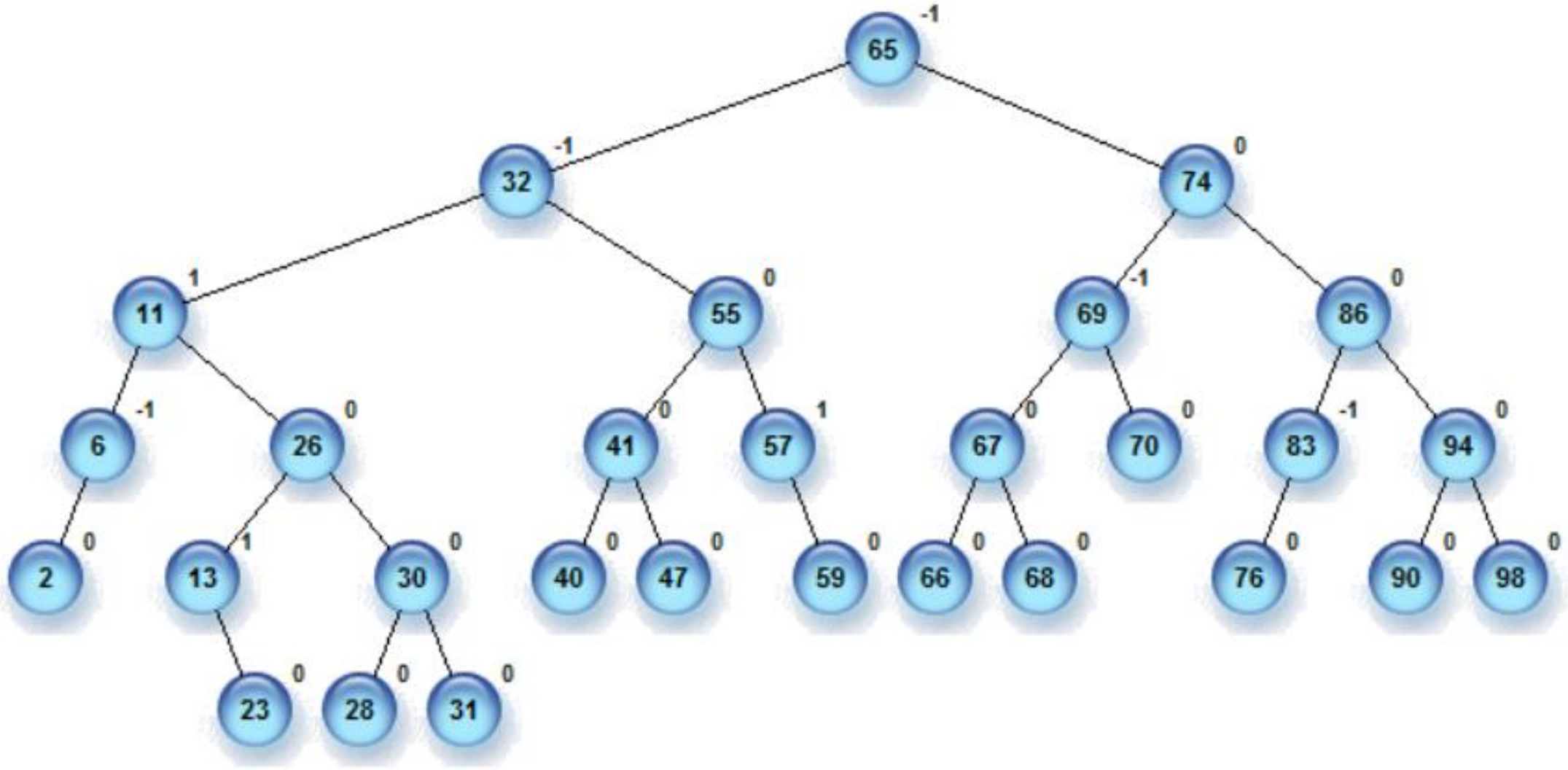
Árbol Equilibrado
(Nodo 6)
Subárbol Izq. Altura 3
Subárbol Der. Altura 2

Árbol B



Árbol No Equilibrado
(Degenerado)
(Nodo 6)
Subárbol Izq. Altura 3
Subárbol Der. Altura 1

Ejemplo de arbol AVL



Árboles Binarios de Búsqueda AVL - Operaciones

1. Insertar.
2. Eliminar.
3. Buscar.

Al Insertar o Eliminar se comprueba si el Árbol está desequilibrado, en caso de estarlo, se realiza el Balanceo

¿Como se Balancea un Árbol?

Mediante Rotaciones:
1. Rotación Simple a la Derecha
2. Rotación Simple a la Izquierda
3. Rotación Doble a la Derecha
4. Rotación Doble a la Izquierda

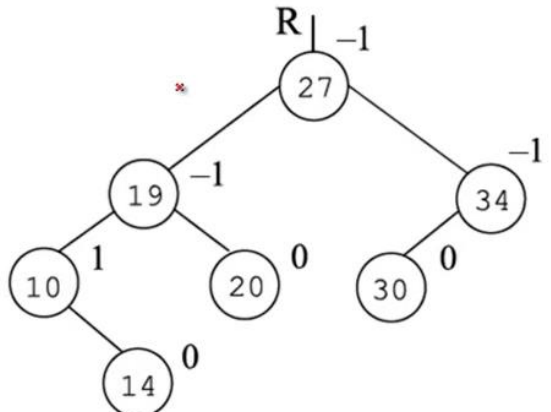
Factor de Equilibrio

Diferencia entre las Alturas del Subárbol Derecho y el Subárbol Izquierdo

FE = Altura Subárbol Derecho - Altura Subárbol Izquierdo

- Un árbol AVL es un **árbol binario de búsqueda** (ABB), ampliado con un campo que indica el **factor de equilibrio** de cada nodo.
- Las operaciones de **acceso** son **idénticas** a las de un ABB.
- Las operaciones de **inserción** y **borrado** se realizan **igual** que en un ABB, salvo que se añade una etapa posterior de **reequilibrado**.
- El reequilibrado recorre los **ascendientes** del nodo que ha sufrido modificación, recalculando sus **factores de equilibrio** y aplicando las **rotaciones** adecuadas cuando es necesario.

Ejemplo FE



Árboles Binarios de Búsqueda AVL - Observaciones

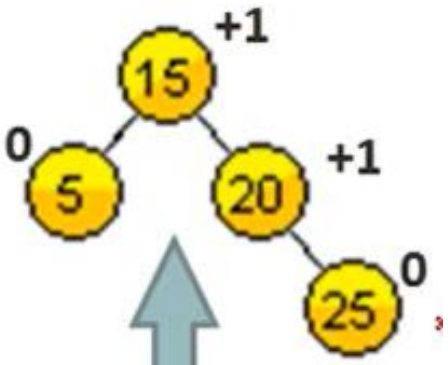
Debo tener en cuenta que...

El FE con:

Signo + Derecha Más Alto

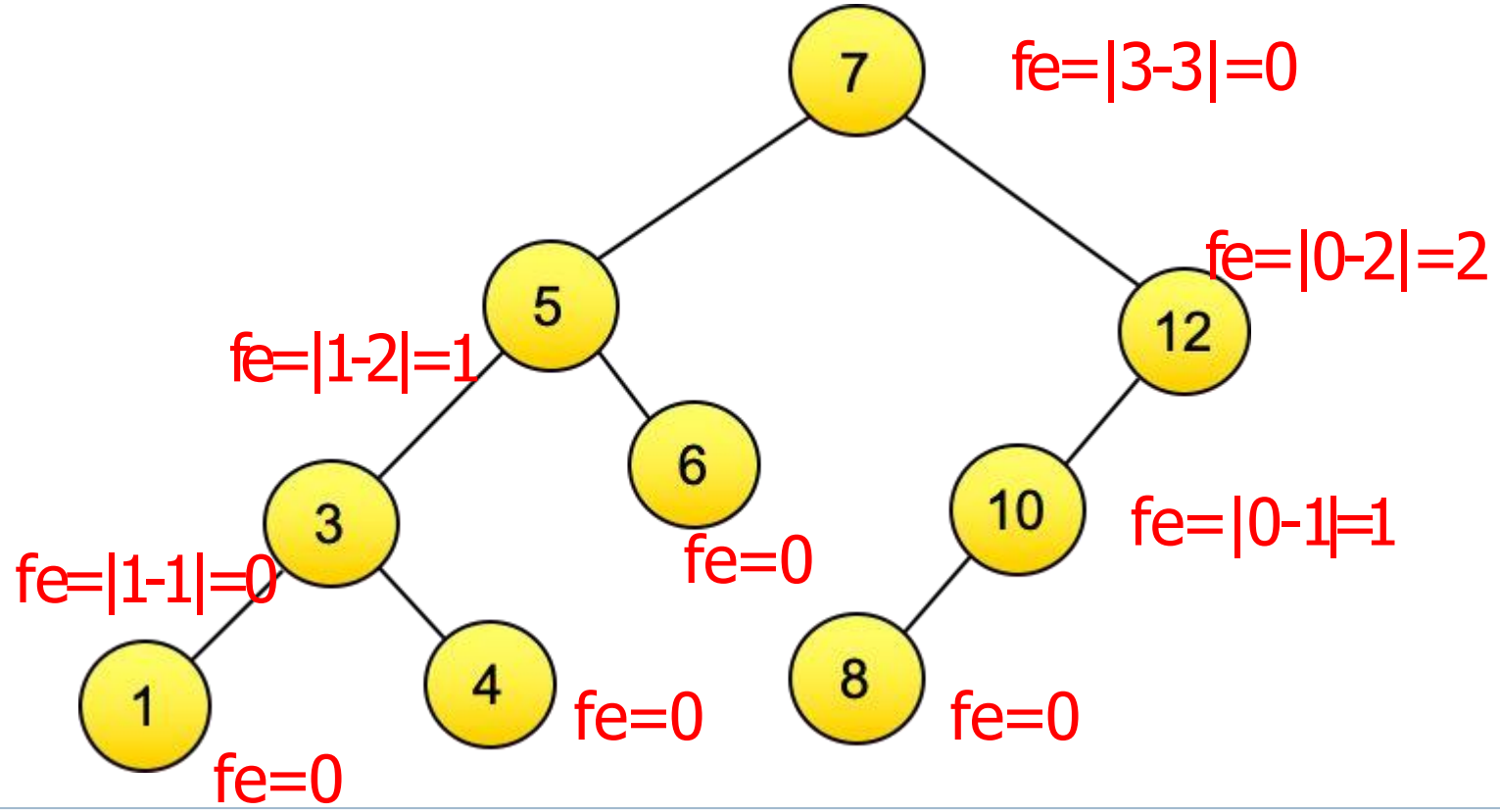
Signo - Izquierda Más Alto

"Un árbol binario es un AVL si y sólo si cada uno de sus nodos tiene un equilibrio de –1, 0, + 1"



Equilibrado basado en altura

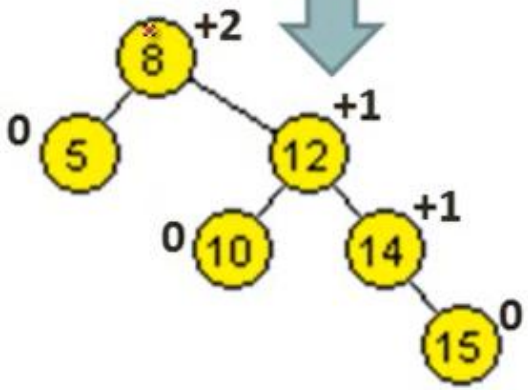
Factor de equilibrio (fe) : valor absoluto de la diferencia entre la altura del subárbol derecho y el subárbol izquierdo. $fe = |h_r - h_l|$



Árboles Binarios de Búsqueda AVL - Observaciones

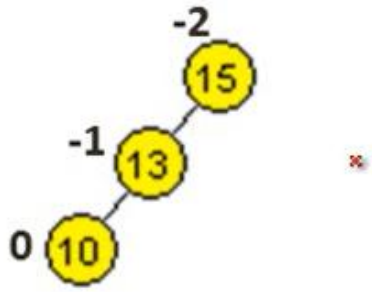
Desequilibrios

Desequilibrio hacia la izquierda (Equilibrio > +1)



Desequilibrios

Desequilibrio hacia la Derecha (Equilibrio < -1)



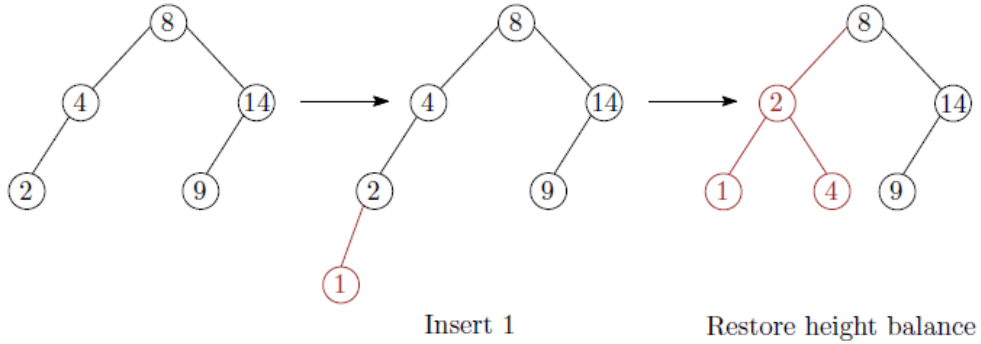
Rotaciones en AVL

- Tras una operación de inserción o borrado, se recorren los ascendientes, recalculando sus factores de equilibrio y teniendo en cuenta el cambio en altura del subárbol.
- Es posible que en el recorrido el factor de equilibrio de algún nodo pasa a valer +2 ó -2 (desequilibrado).
- En ese caso se aplica una determinada rotación que restablece el equilbrio del nodo (aunque es posible que cambie la altura del nodo).
- En un árbol AVL se necesitan 2 tipos de rotaciones (simples y dobles), en un sentido u otro (izquierdas y derechas).

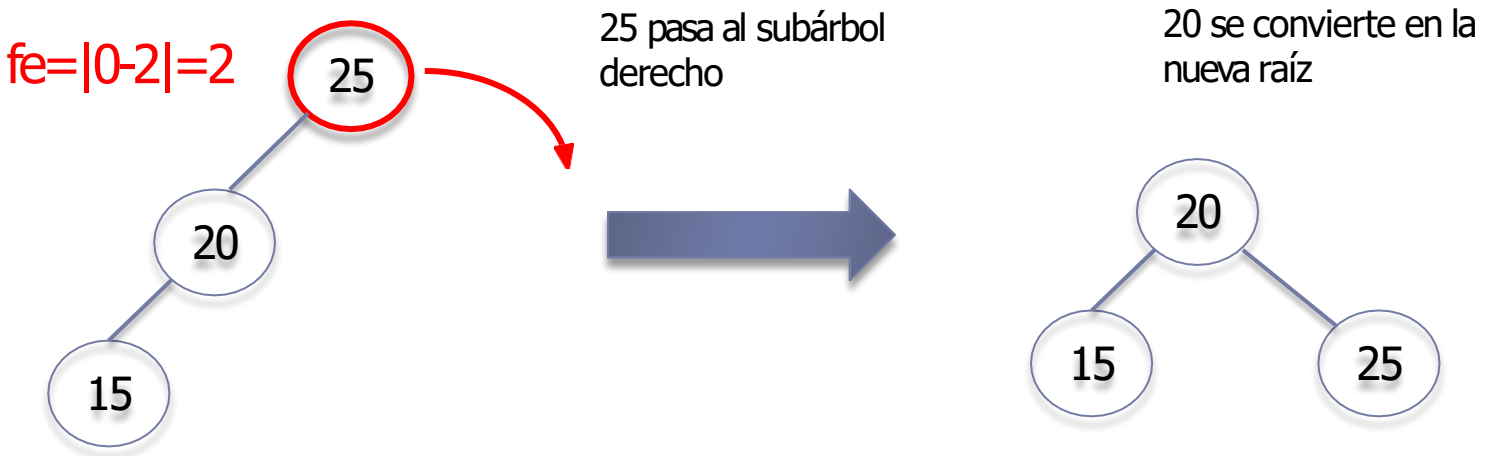
Árboles Binarios de Búsqueda AVL – Rotación simple a la derecha

Ésta rotación se usará cuando el subárbol izquierdo de un nodo sea 2 unidades más alto que el derecho, es decir, cuando su FE sea de -2. Y además, la raíz del subárbol izquierdo tenga una FE de -1, es decir, que esté cargado a la izquierda.

Ejemplo



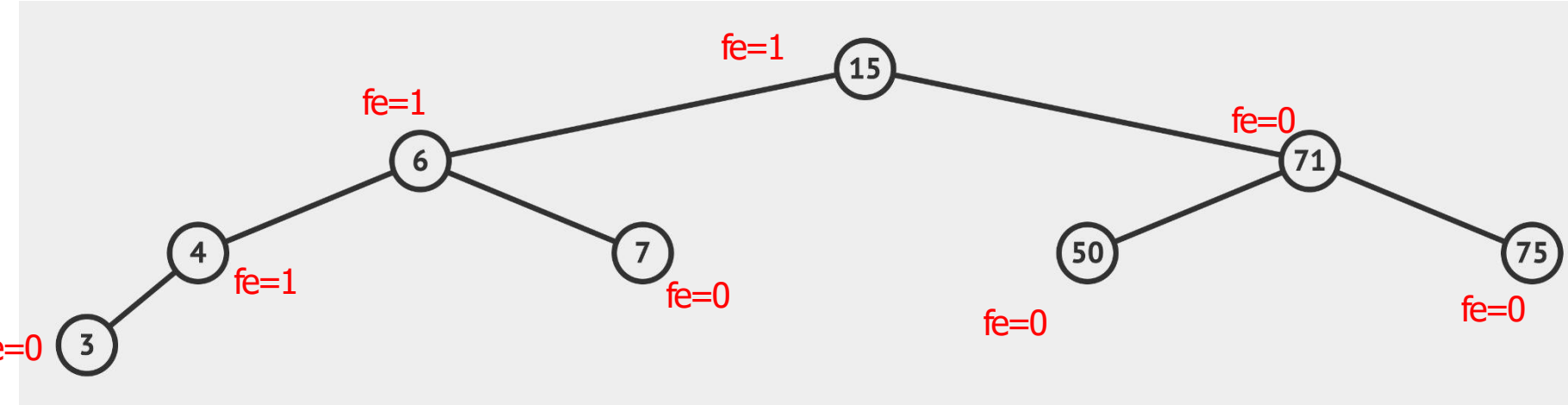
Rotación Simple Derecha



Como la rama más larga es la rama izquierda, rotamos el nodo desbalanceado (25) hacia la derecha. Es decir, el nodo 25 debe pasar al subárbol derecho. La nueva raíz del subárbol será el hijo izquierdo del nodo desbalanceado.

Rotación Simple Derecha

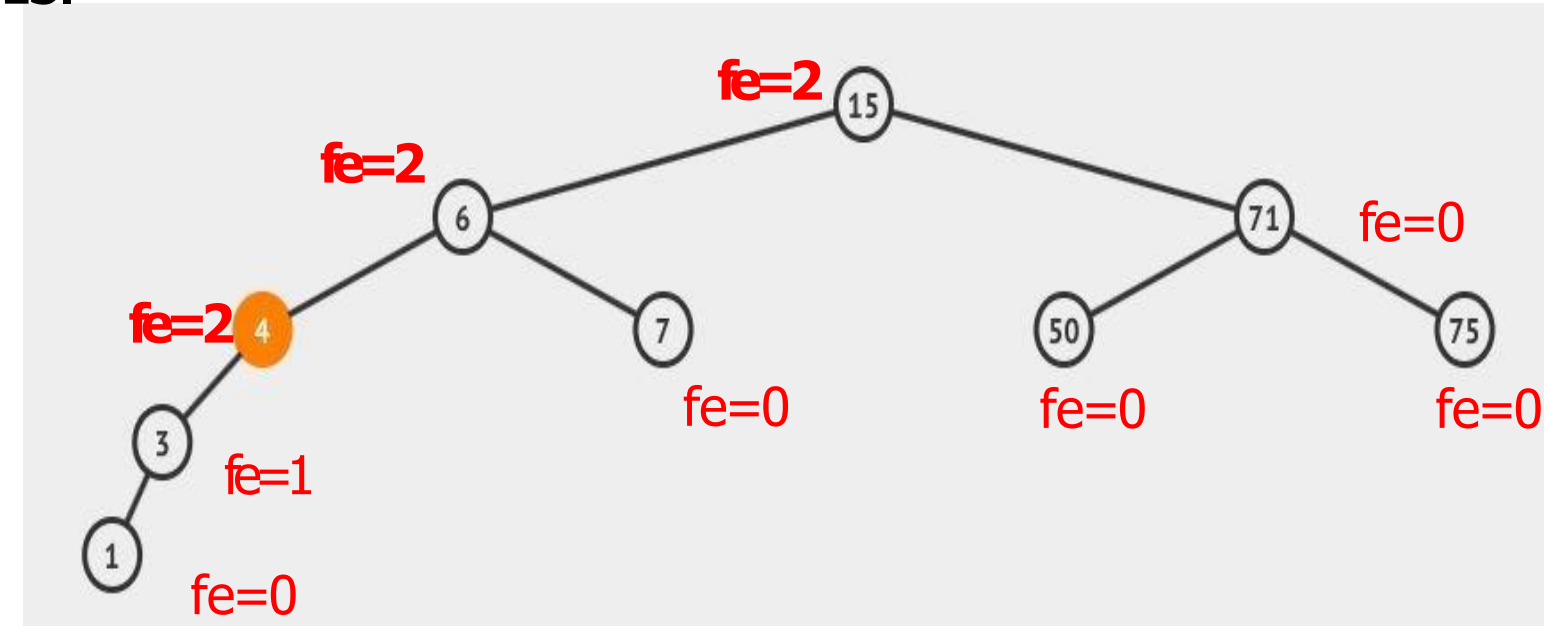
Este árbol está equilibrado.



¿Qué ocurre si insertamos 1?

Rotación Simple Derecha

Después de insertar el 1, hay tres nodos no equilibrados: 4, 6 y 15.

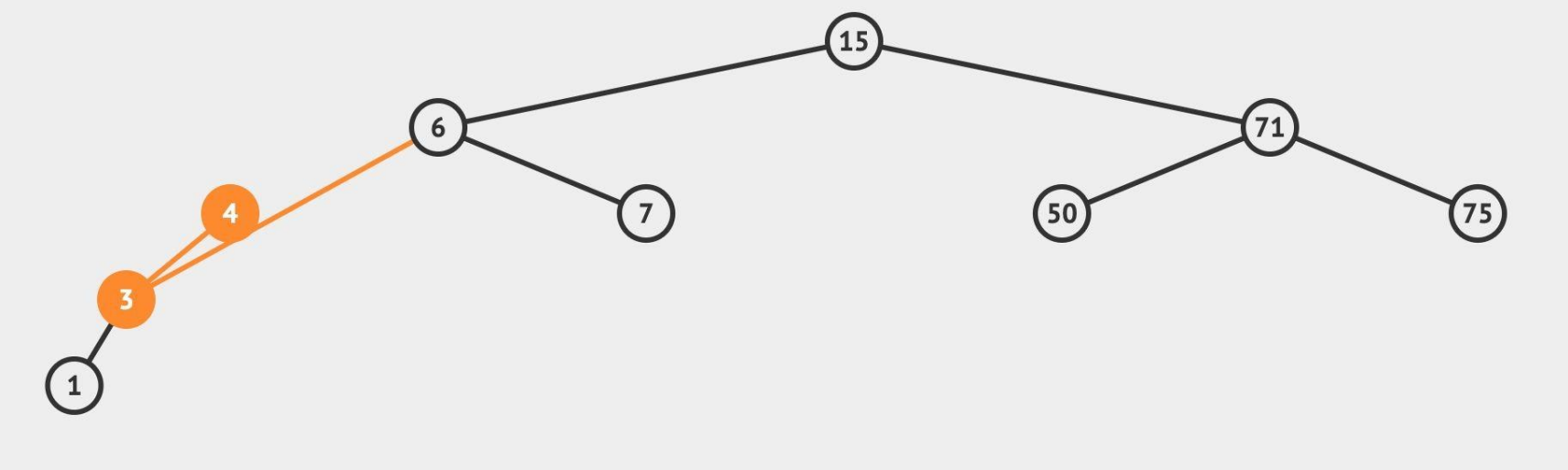


El primer nodo que ha quedado desequilibrado es 4. Al equilibrarlo, conseguiremos que queden también equilibrados sus ancestros.

Rotación Simple Derecha

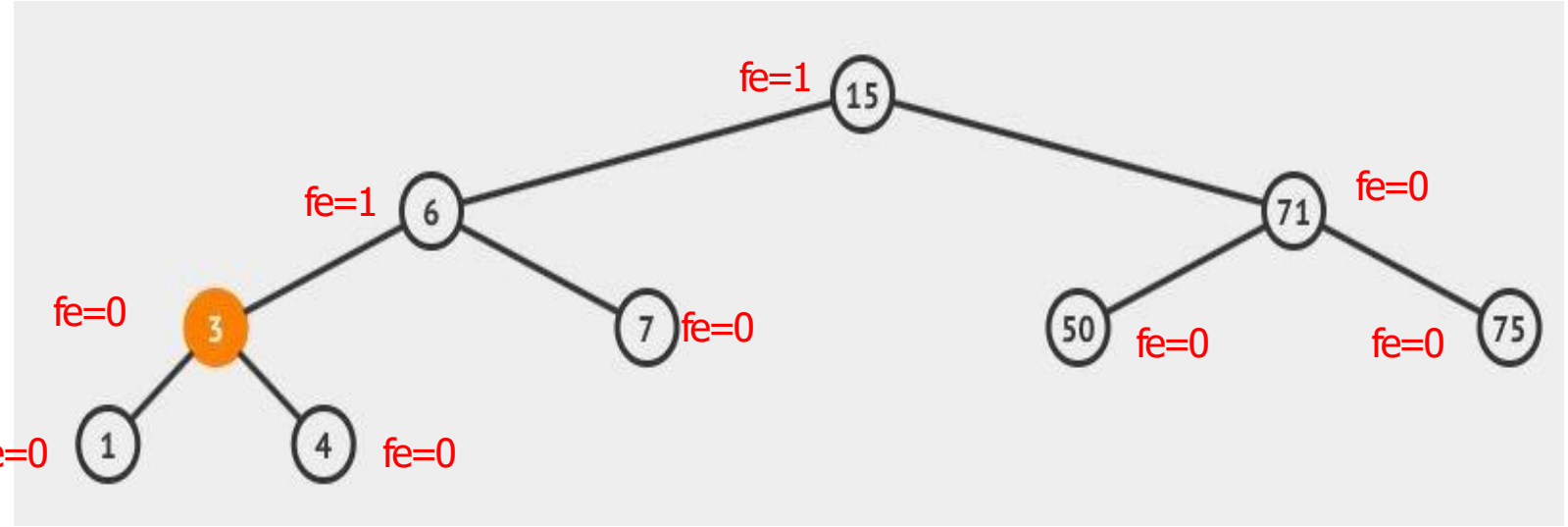
- Considerando el nodo desequilibrado (4), su rama más larga es la izquierda. Para equilibrar demos rotar el nodo (4) a la derecha. Por eso se llama rotación derecha.

- La nueva raíz será el nodo 3 (que es el hijo izquierdo del nodo 4), y el nodo 4, rotará a la derecha, convirtiéndose en el hijo derecho de la nueva raíz.

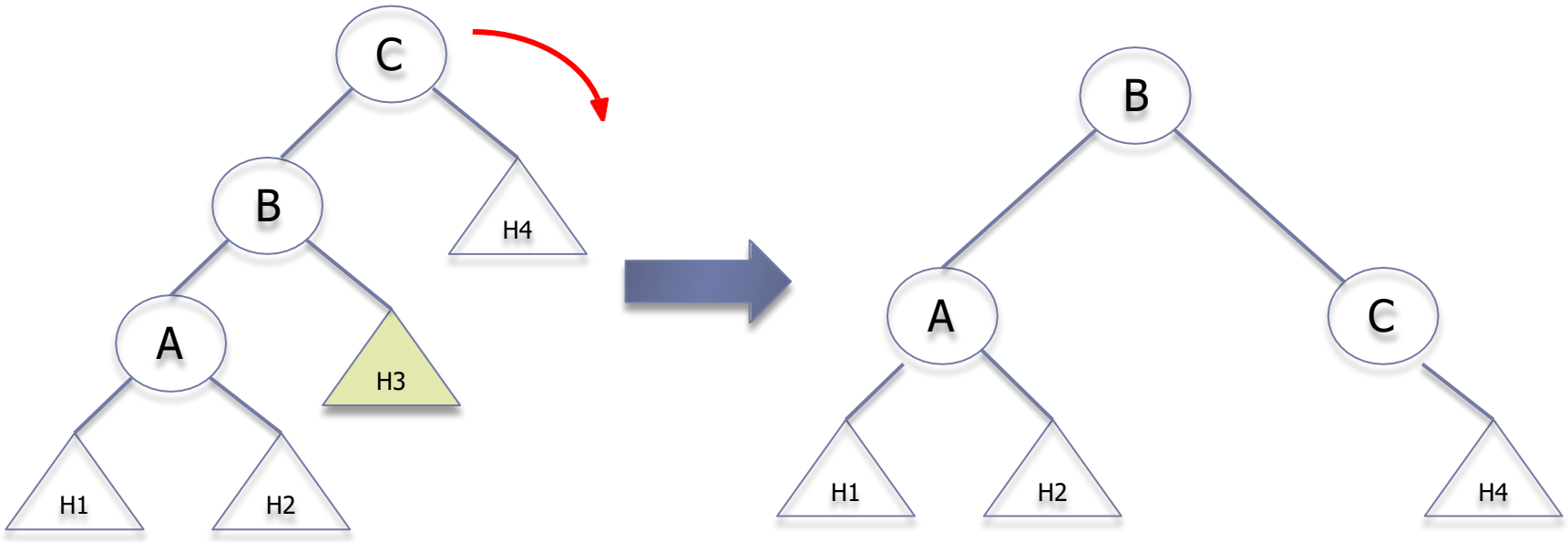


Rotación Simple Derecha

El árbol resultante después de aplicar la rotación derecha es el siguiente (donde todos los nodos están equilibrados):

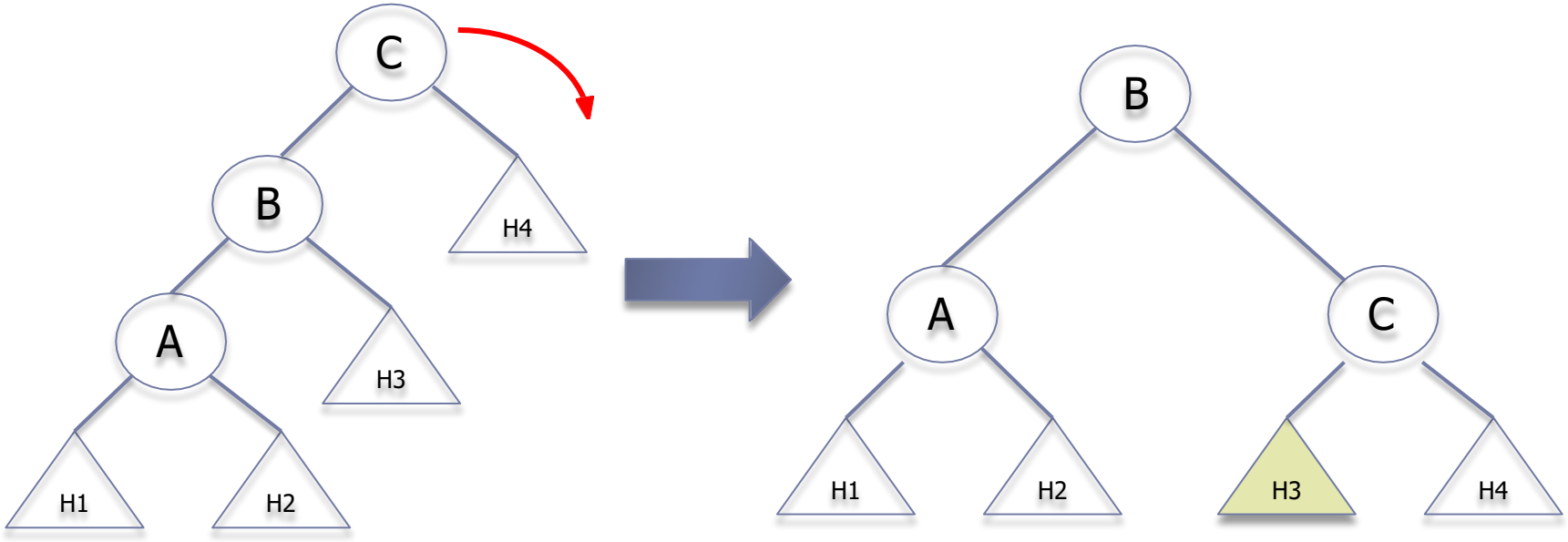


Rotación Simple Derecha



¿Qué pasa si la nueva raíz (B) tenía subárbol derecho?, El nuevo hijo derecho de B es C, ¿qué hacemos con H3?

Rotación Simple Derecha

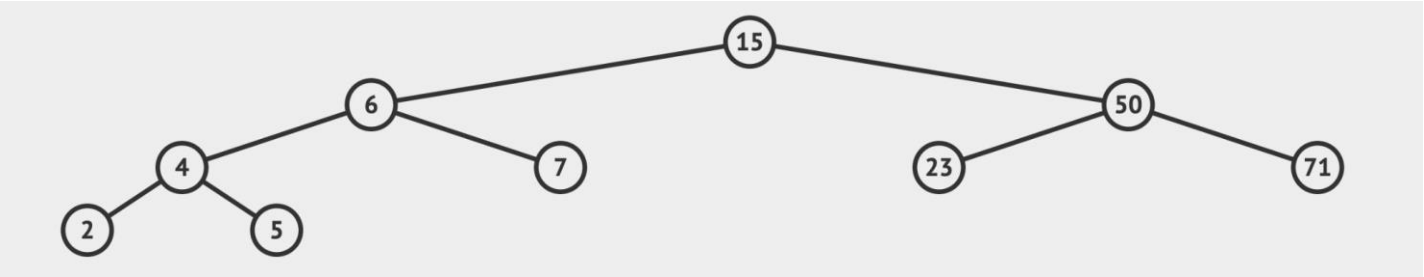


InOrder: $H_1AH_2BH_3CH_4$ = InOrder: $H_1AH_2BH_3CH_4$

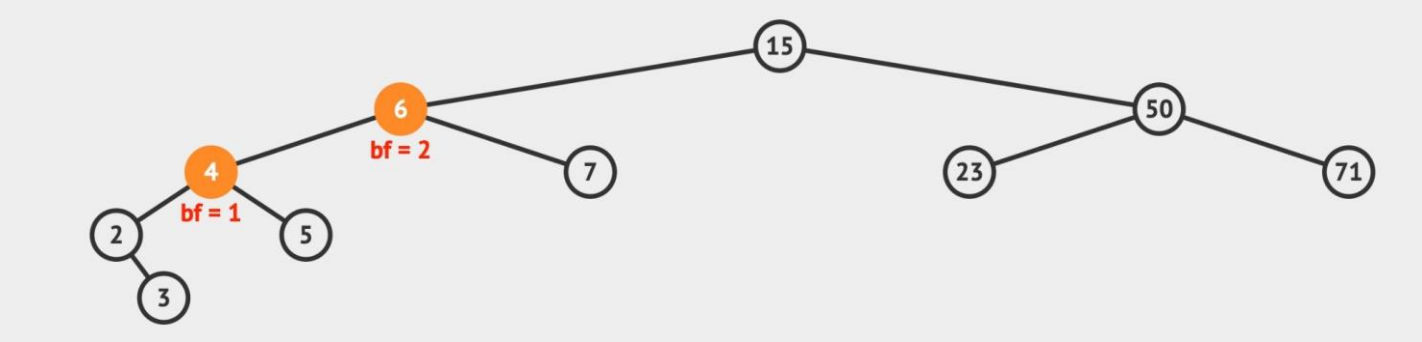
La única opción es que pase a ser hijo izquierdo de C (cualquier otra opción dejaría de ser un ABB)

Rotación Simple Derecha

¿Qué pasa si en el siguiente árbol insertamos el elemento 3?

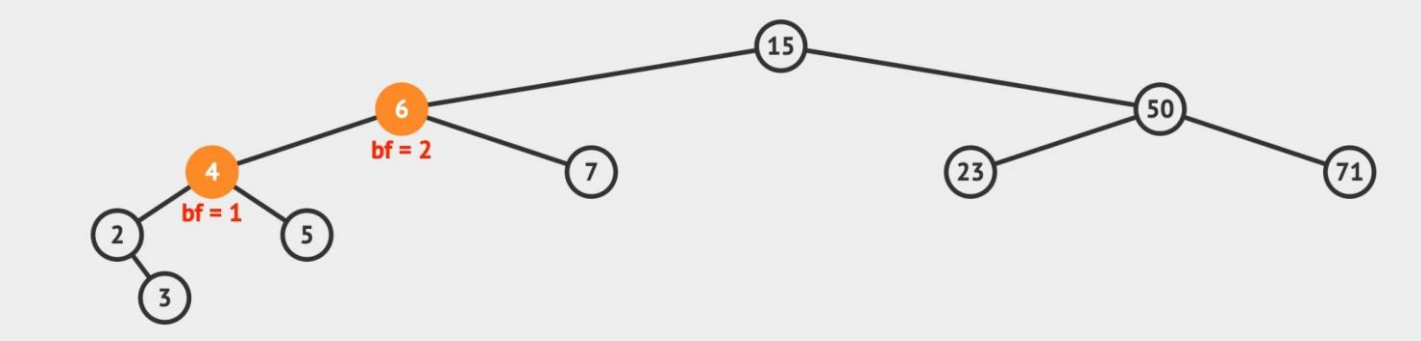


Después de insertar 3, el nodo 6, queda desbalanceado, ¡debemos equilibrarlo!.



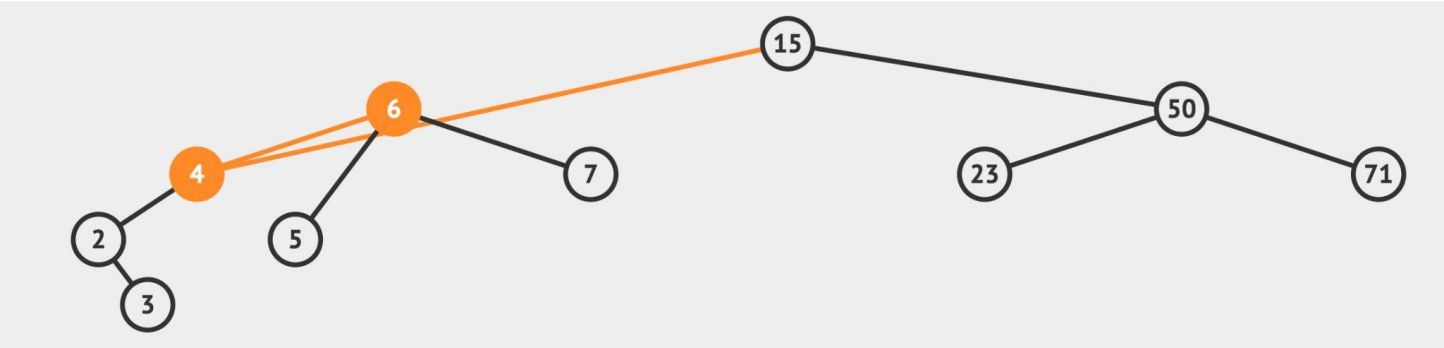
Nota: El nodo 15 también queda desequilibrado (bf=2), pero tras equilibrar 6, quedará equilibrado.

Rotación Simple Derecha

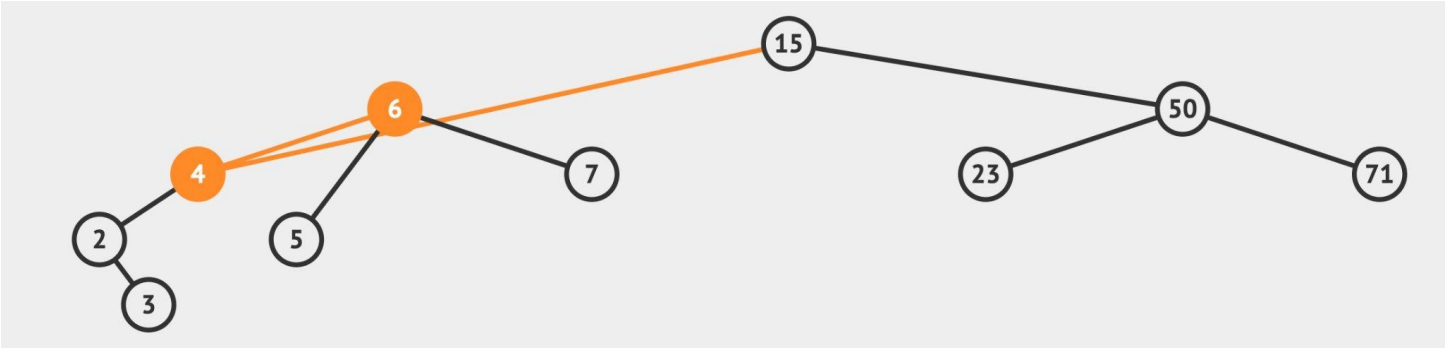


Para equilibrar el nodo 6, como su rama más larga es la izquierda, el nodo 6 deberá rotar a la derecha.

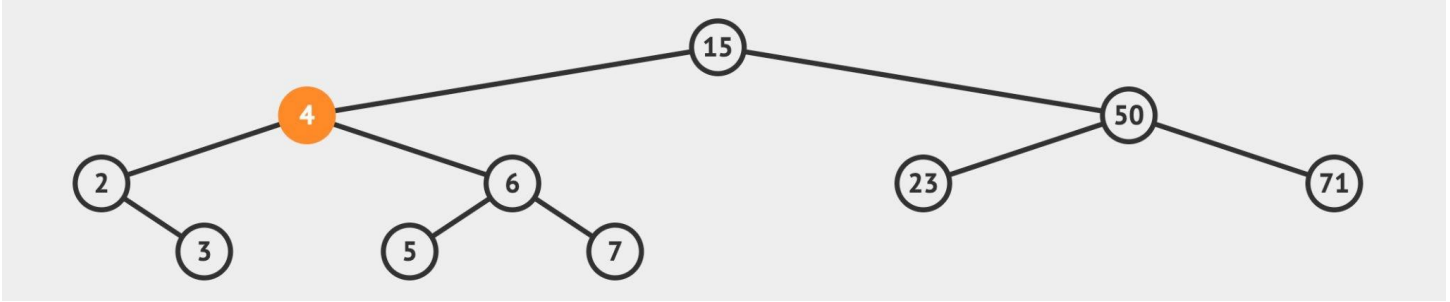
Si aplicamos la rotación simple derecha, el nodo 4, se convertirá en la nueva raíz, y su actual hijo derecho debe pasar a ser el hijo izquierdo de 6



Rotación Simple Derecha



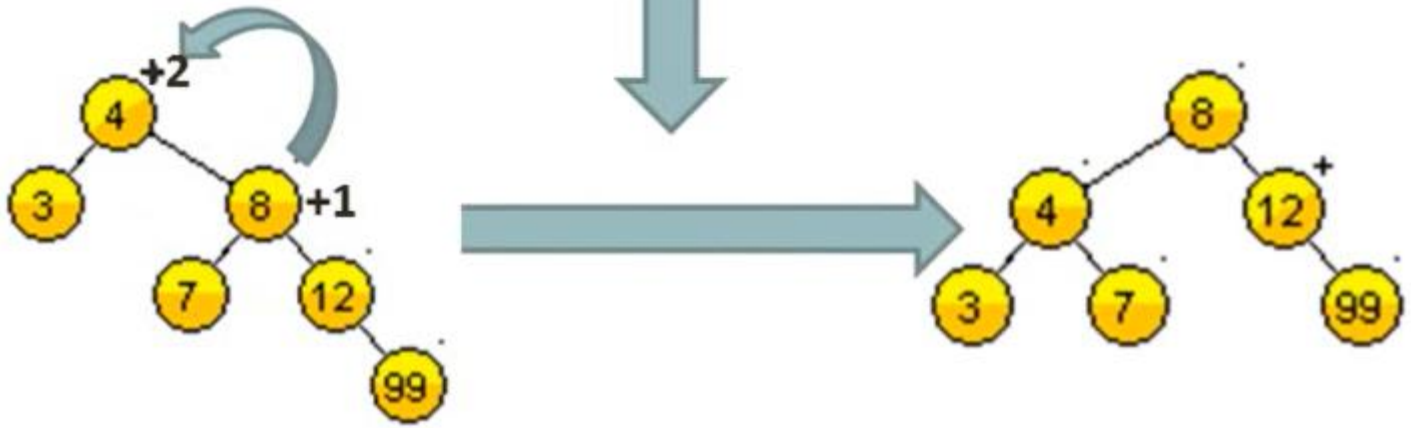
Finalmente, el nodo 6, se convierte en el nuevo hijo derecho, de la nueva raíz del subárbol (nodo 4). En el nuevo árbol resultante, todo los nodos ya están balanceados (también el nodo 15).



Árboles Binarios de Búsqueda AVL – Rotación simple a la izquierda

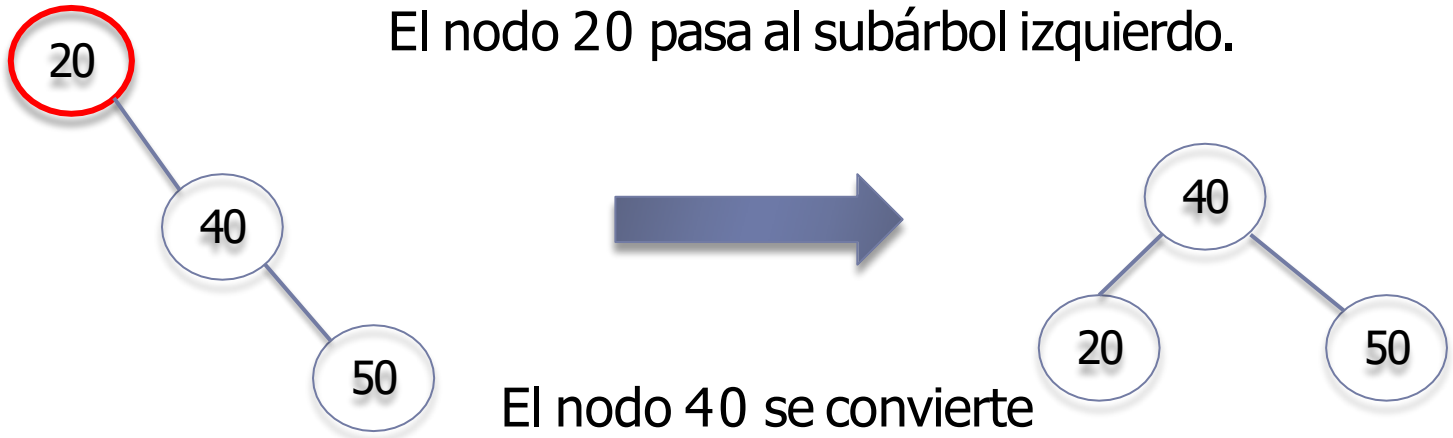
Ésta rotación se usará cuando el subárbol derecho de un nodo sea 2 unidades más alto que el izquierdo, es decir, cuando su FE sea de 2. Y además, la raíz del subárbol derecho tenga un FE de 1, es decir, que esté cargado a la derecha.

Ejemplo



Rotación Simple Izquierda

El nodo 20 está desequilibrado. La rama más larga es la derecha.
El nodo 20 pasa al subárbol izquierdo.

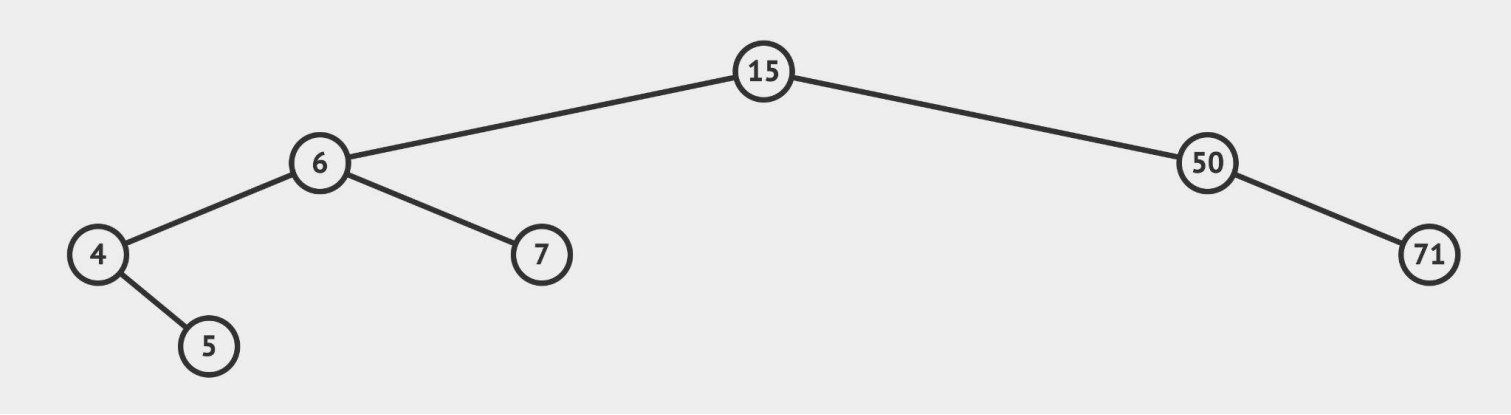


El nodo 40 se convierte en la nueva raíz

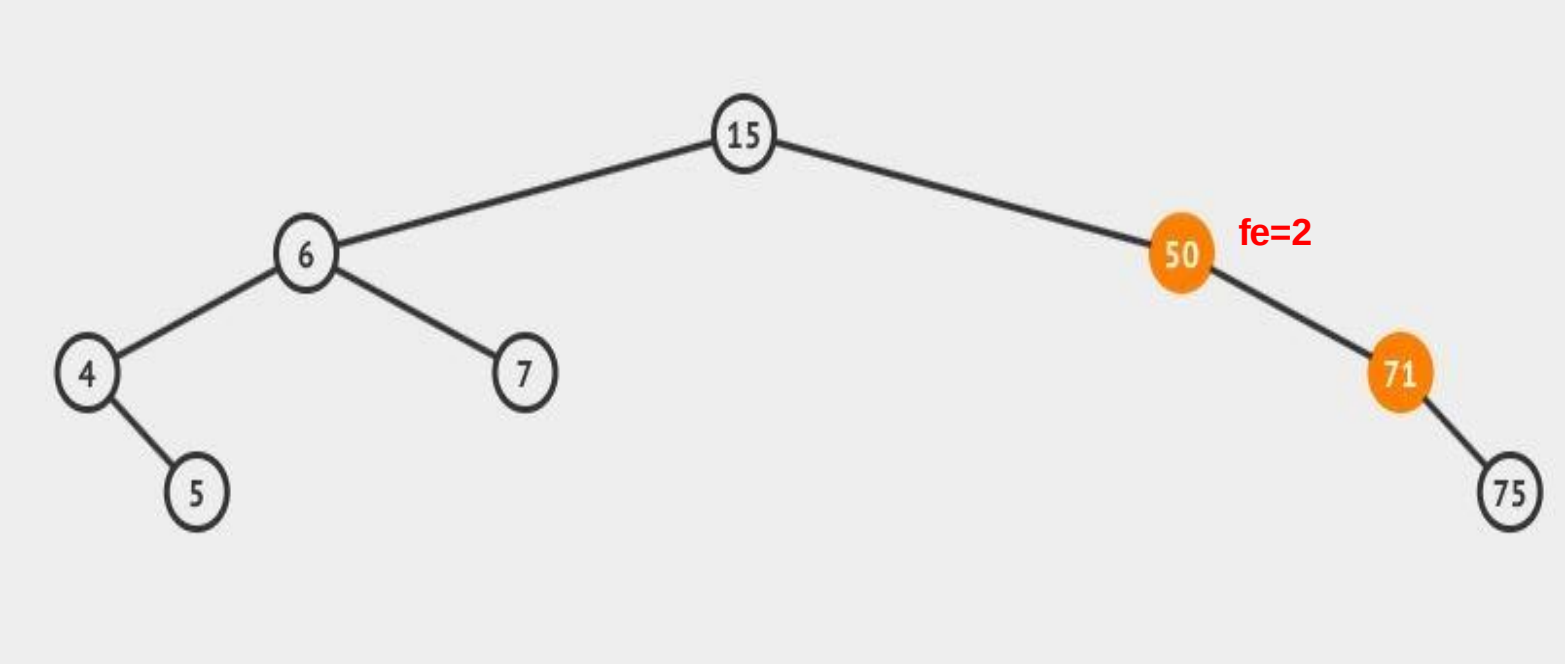
Como la rama más larga es la rama derecha, el nodo desbalanceado (20) debe rotar para pasar al subárbol izquierdo. Por eso se denomina rotación izquierda. La nueva raíz del subárbol será el hijo derecho (nodo 40) del nodo desbalanceado.

Rotación Simple Izquierda

El árbol está equilibrado. ¿Qué ocurre si insertamos 75?

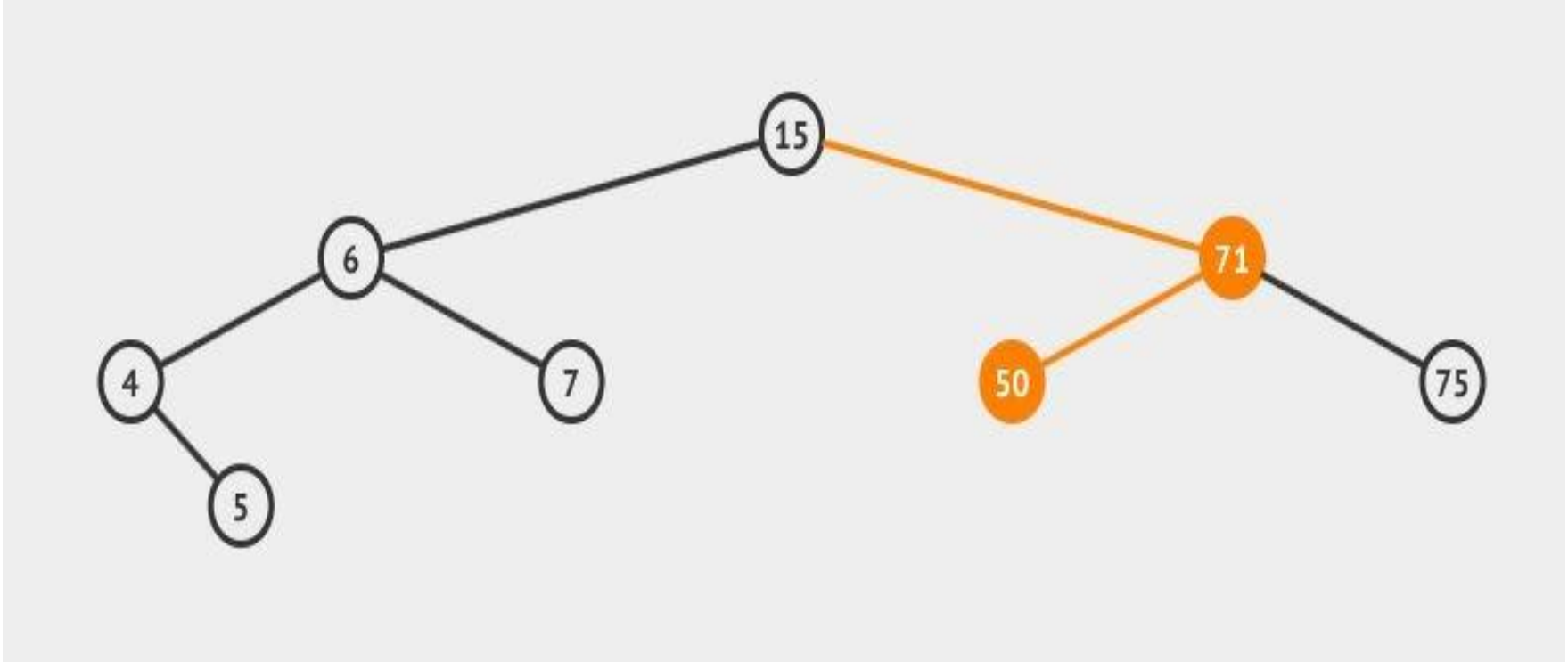


Rotación Simple Izquierda



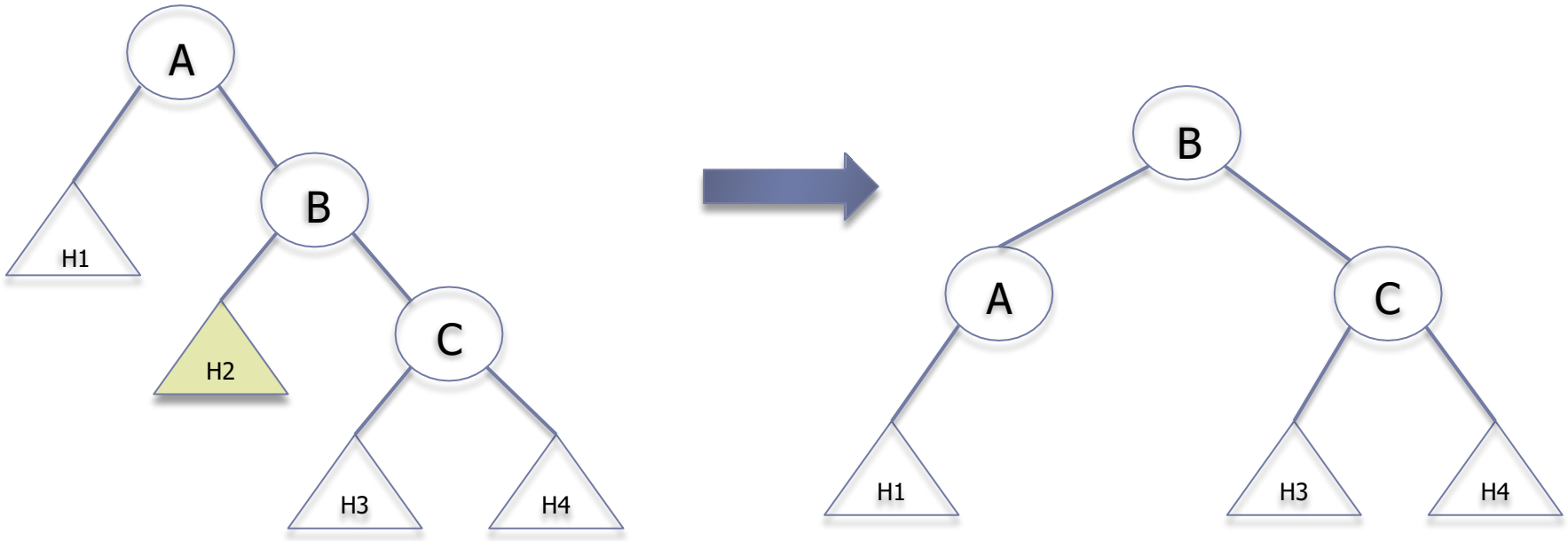
Al insertar 75, 50 queda desequilibrado (fe=2). Debemos aplicar una rotación simple izquierda. 71 será la nueva raíz y 50 rotará a la izquierda, convirtiéndose en el nuevo hijo izquierdo de la nueva raíz (71).

Rotación Simple Izquierda



Todos los nodos ya están equilibrados (fe<=1).

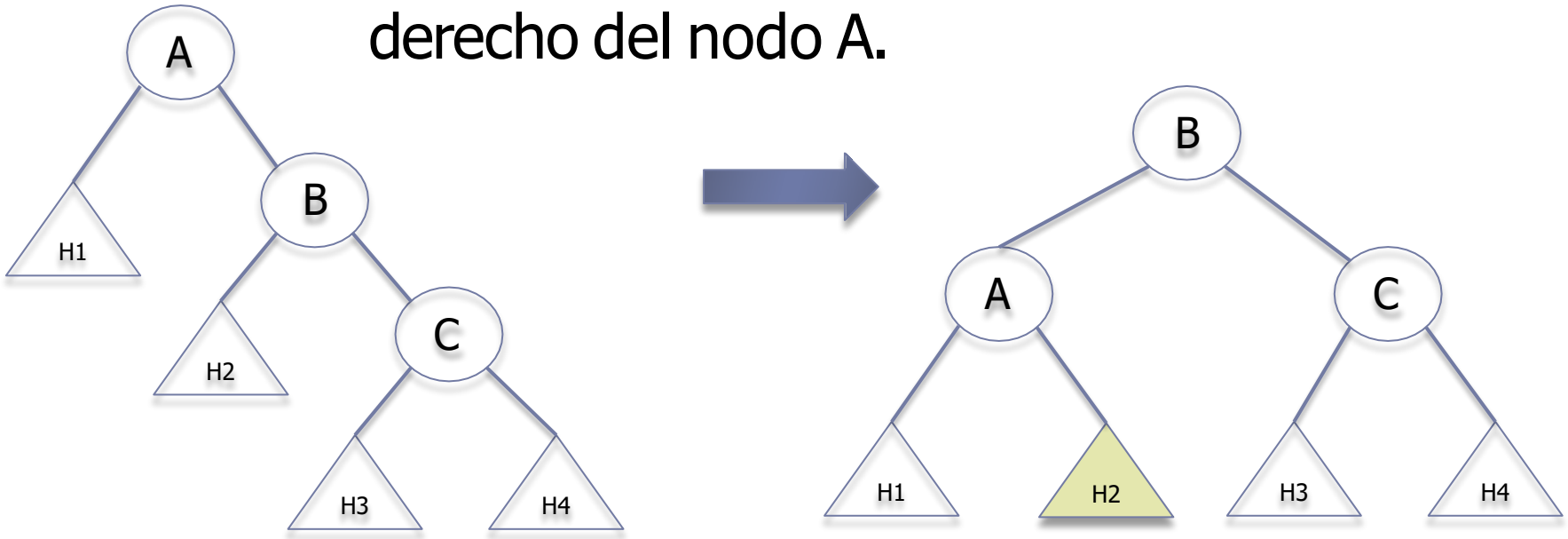
Rotación Simple Izquierda



¿Qué si la nueva raíz (B) ya tenía un hijo izquierdo? El nodo desbalanceado (A) al rotar a la izquierda se convierte en el nuevo hijo de la nueva raíz (B).
¿Qué podemos hacer con el subárbol H2, hijo izquierdo de B?

Rotación Simple Izquierda

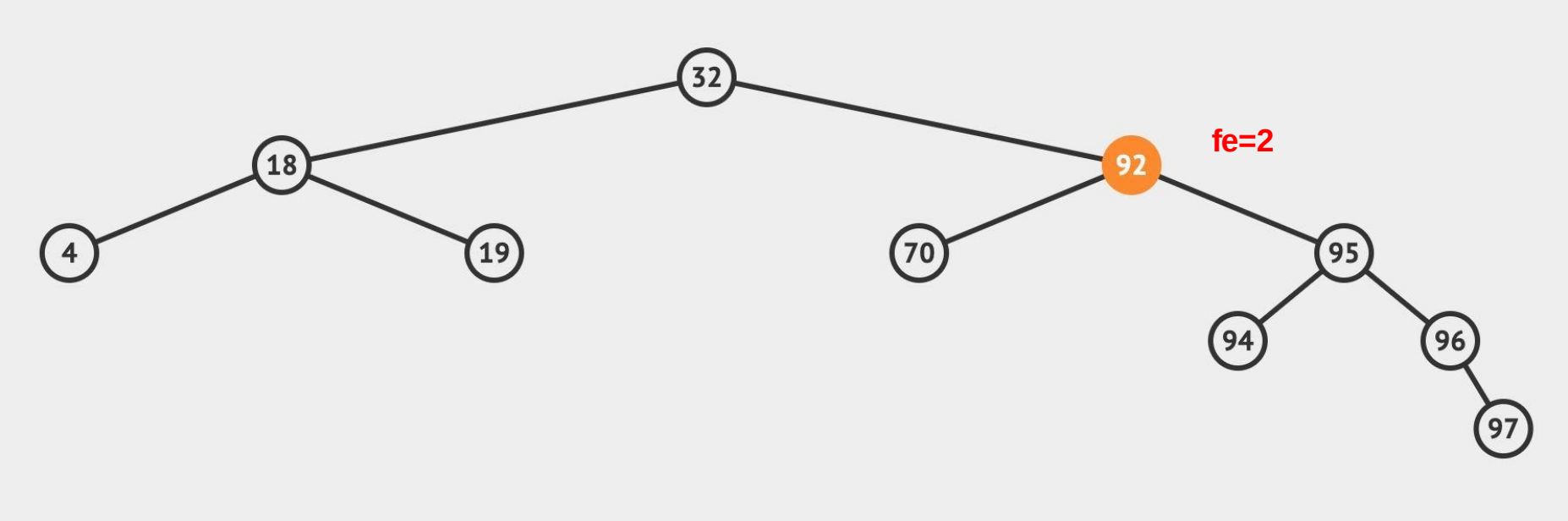
La única opción para que siga siendo un ABB es que H2 se convierta en el hijo derecho del nodo A.



InOrder: $H_1AH_2BH_3CH_4$ = InOrder: $H_1AH_2BH_3CH_4$

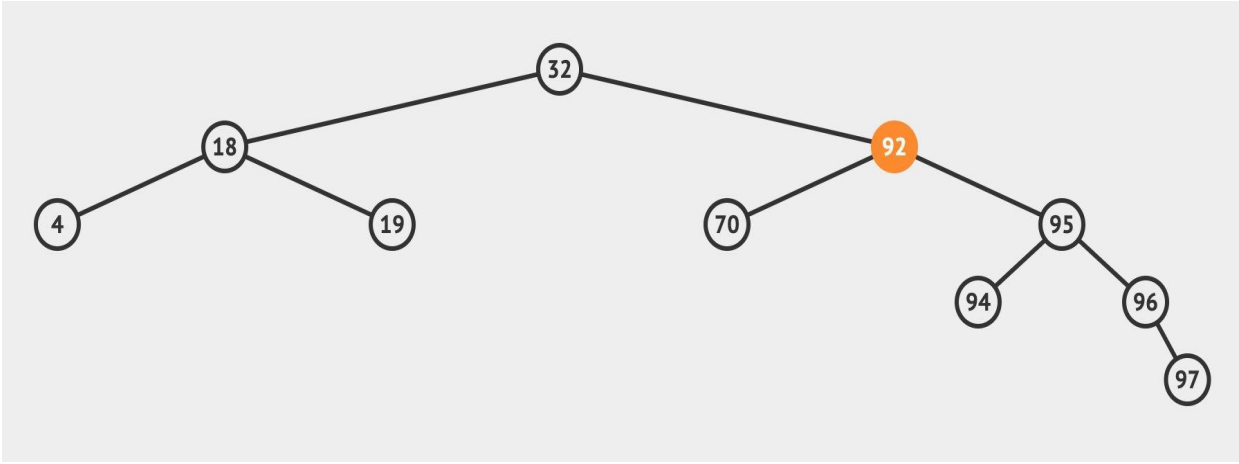
Rotación Simple Izquierda

Supongamos que acabamos de insertar 97. El árbol ha quedado desequilibrado (nodo 92)

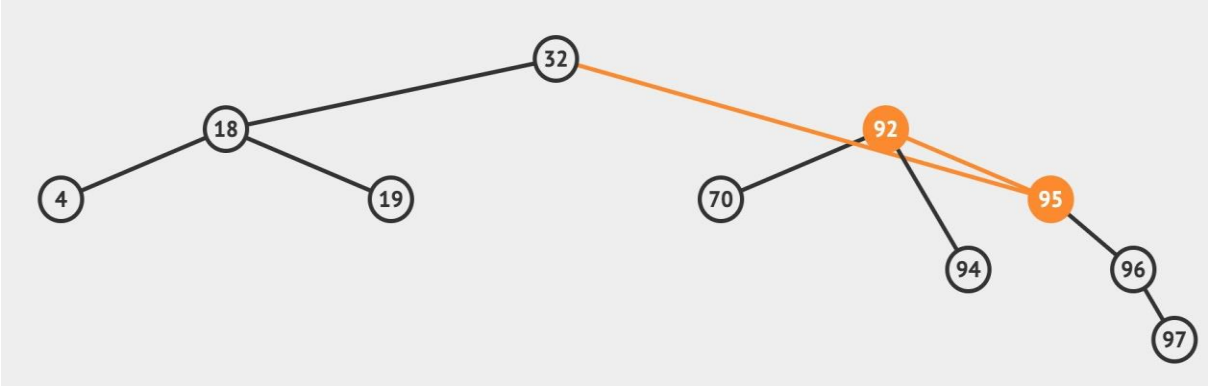


Nota: el nodo raíz (32) también ha quedado desequilibrado (fe=2). Después de equilibrar 92, el nodo raíz también quedará equilibrado.

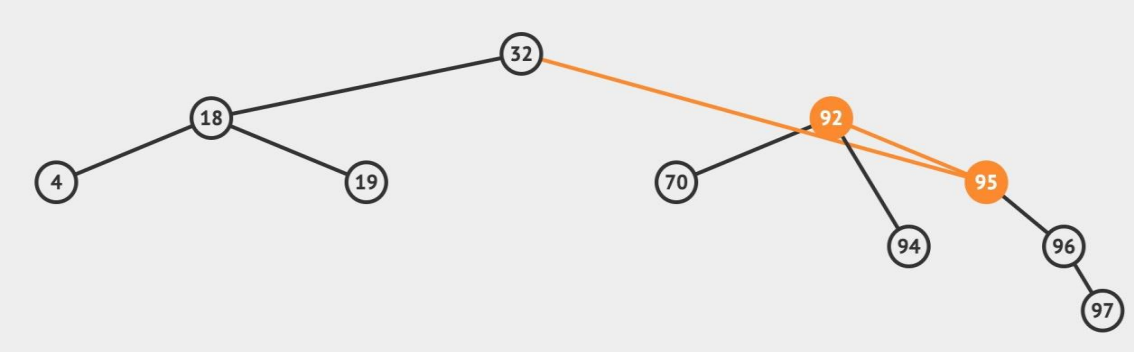
Rotación Simple Izquierda



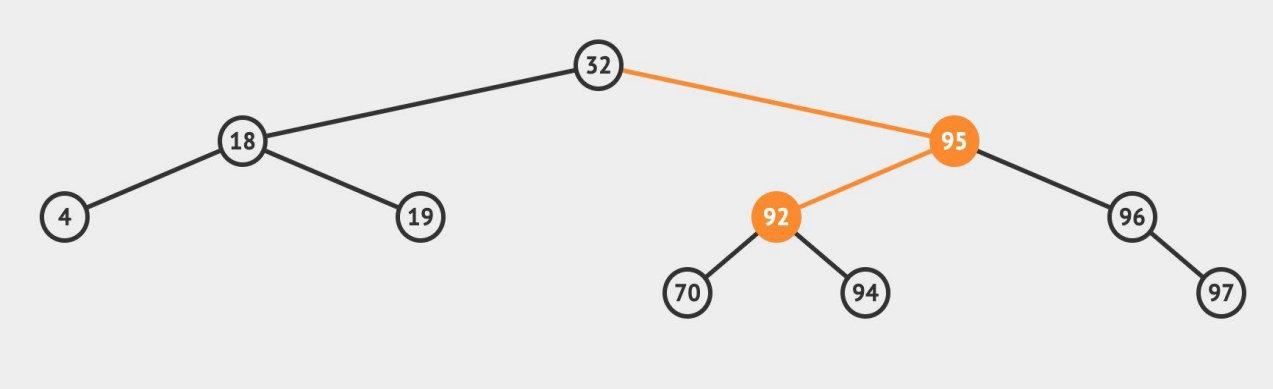
El nodo 92 está desequilibrado. Su rama más larga es la derecha, por tanto, el nodo 92 tiene que rotar a la izquierda. El nuevo nodo raíz del subárbol será su hijo izquierdo: nodo 95. Este nodo, antes de convertirse en la nueva raíz del subárbol, deberá pasar su hijo izquierdo, 94, para que se convierta en el hijo derecho de 92.



Rotación Simple Izquierda



Finalmente, el antiguo nodo desbalanceado 92, se convierte en el nodo izquierdo de la nueva raíz del subárbol, 95. Todos los nodos del árbol ya están balanceados (fe<=1), incluído la raíz del árbol.

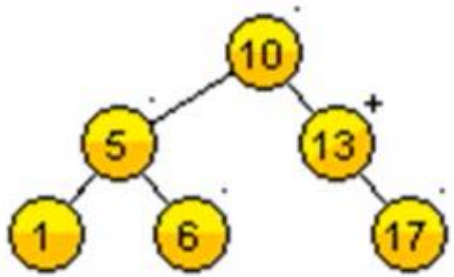


Árboles Binarios de Búsqueda AVL – Rotación doble a la derecha

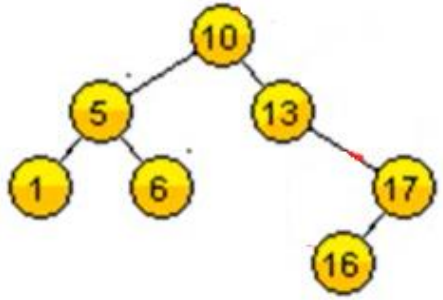
Si esta desequilibrado a la izquierda (FE>+1), y su hijo derecho tiene distinto signo (–) hacemos Rotación Doble Derecha.

Doble Derecha =
Simple Derecha + Simple Izquierda

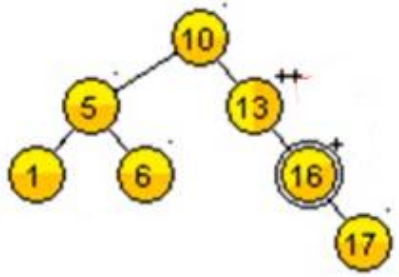
Al Inicio



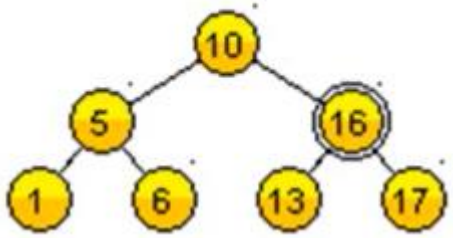
Insertando al 16



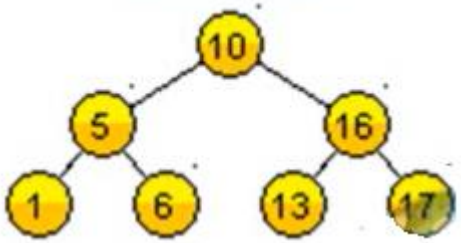
Rotación Der.



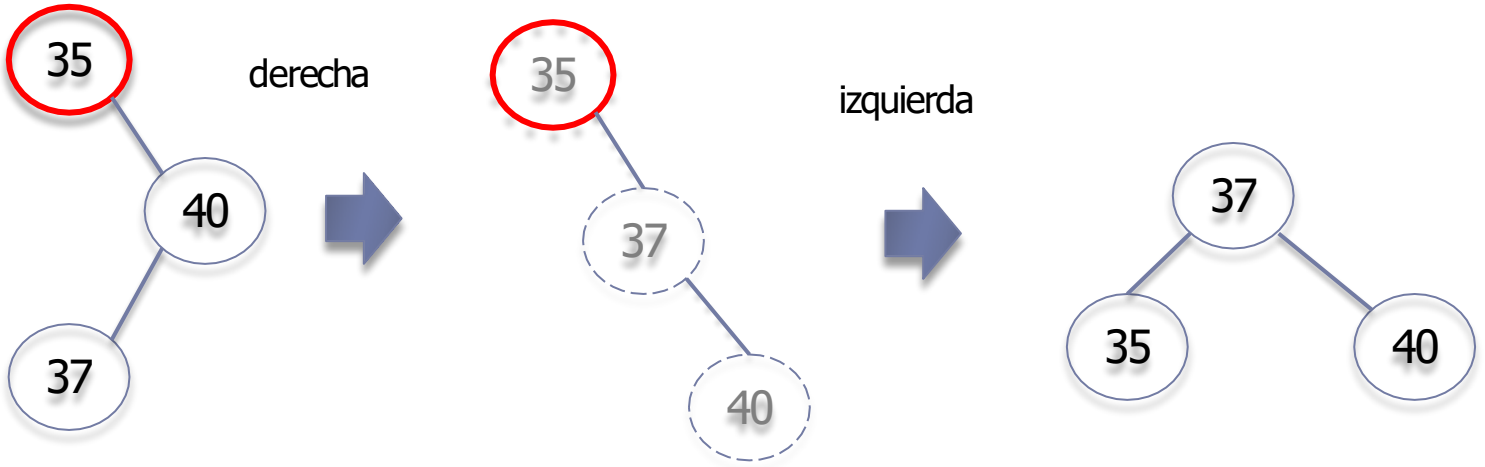
Rotación Izq.



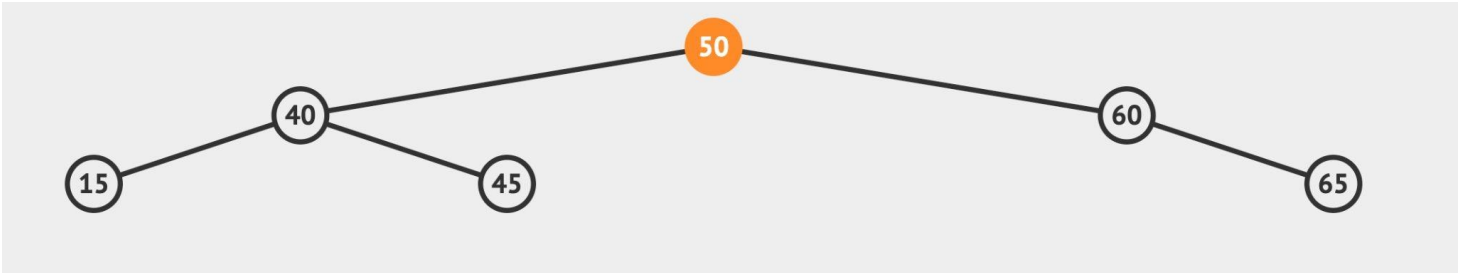
Balanceado



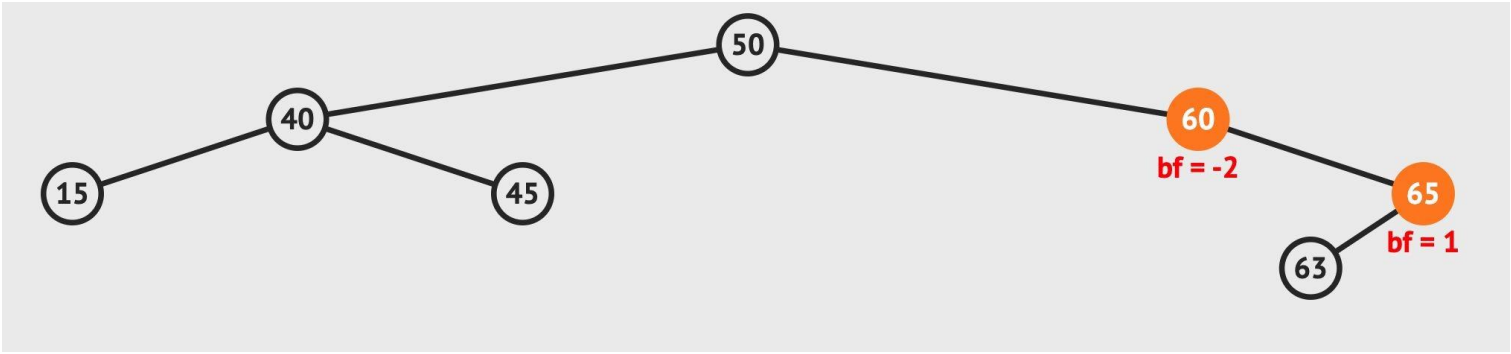
Doble Rotación Derecha Izquierda



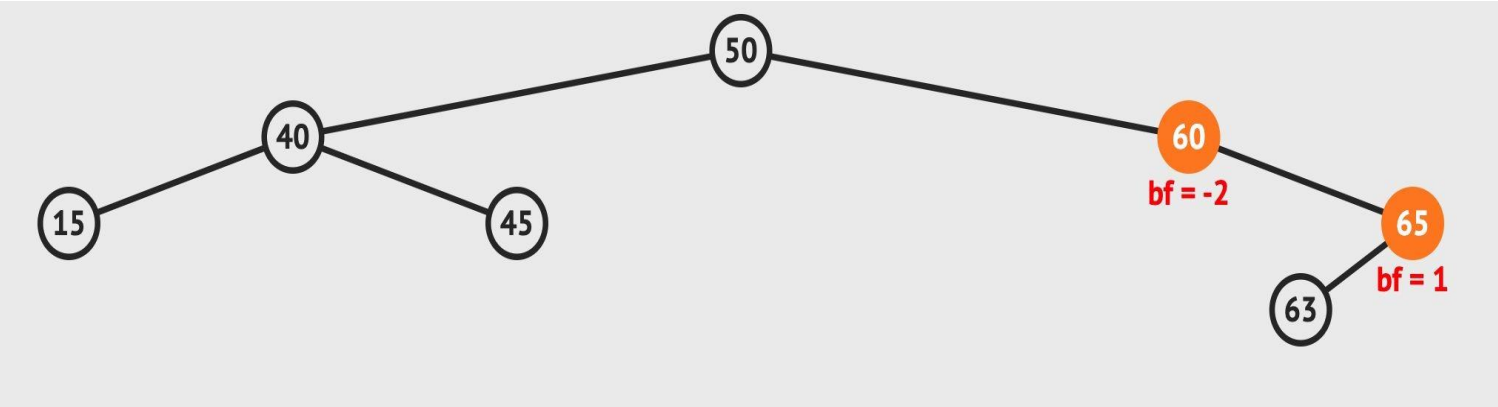
Doble Rotación Derecha Izquierda



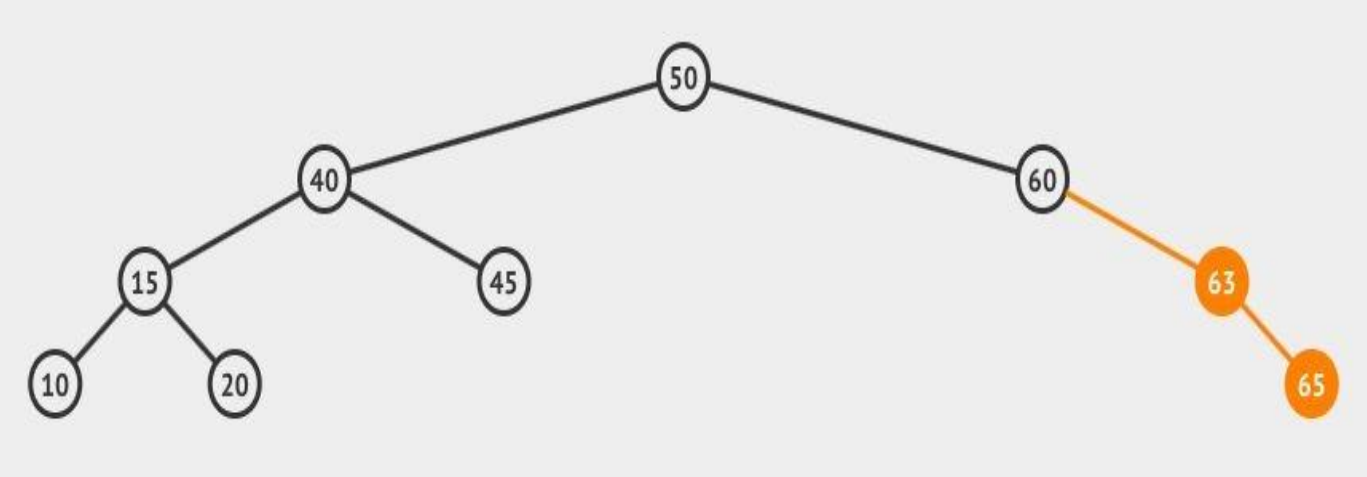
Si insertamos 63, el nodo 60 queda desequilibrado. Debemos aplicar una doble rotación derecha-izquierda



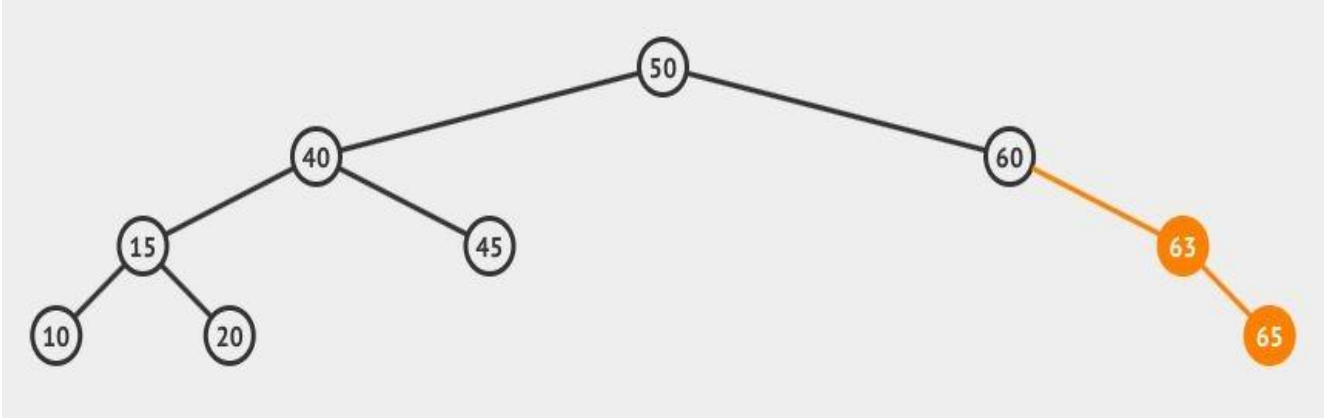
Doble Rotación Derecha Izquierda



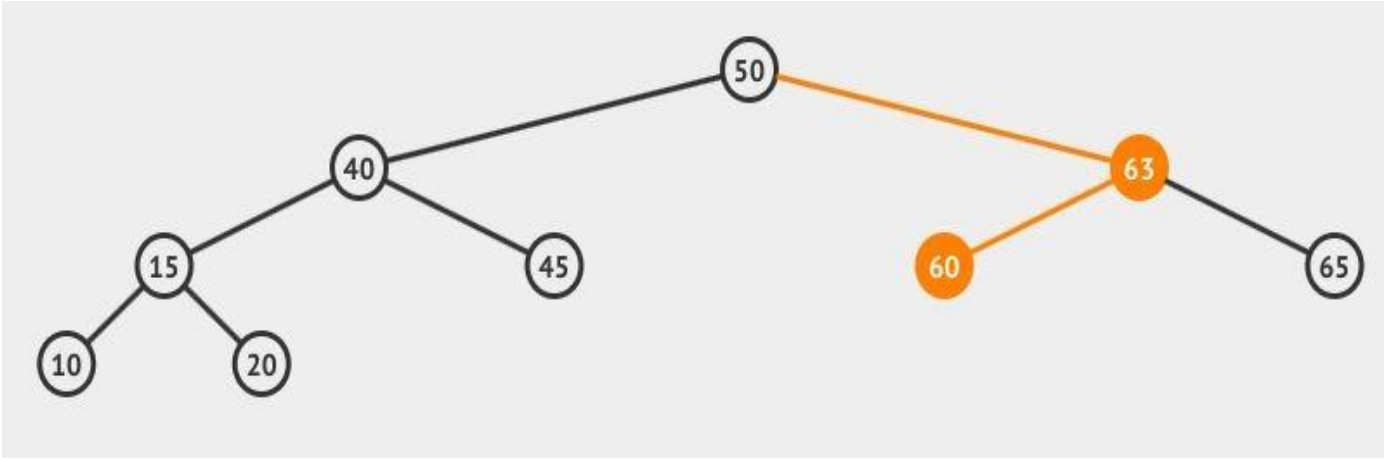
Primera rotación derecha: 63 se convierte en el hijo derecho de 60. Ahora ya podríamos aplicar una rotación simple izquierda sobre 60:



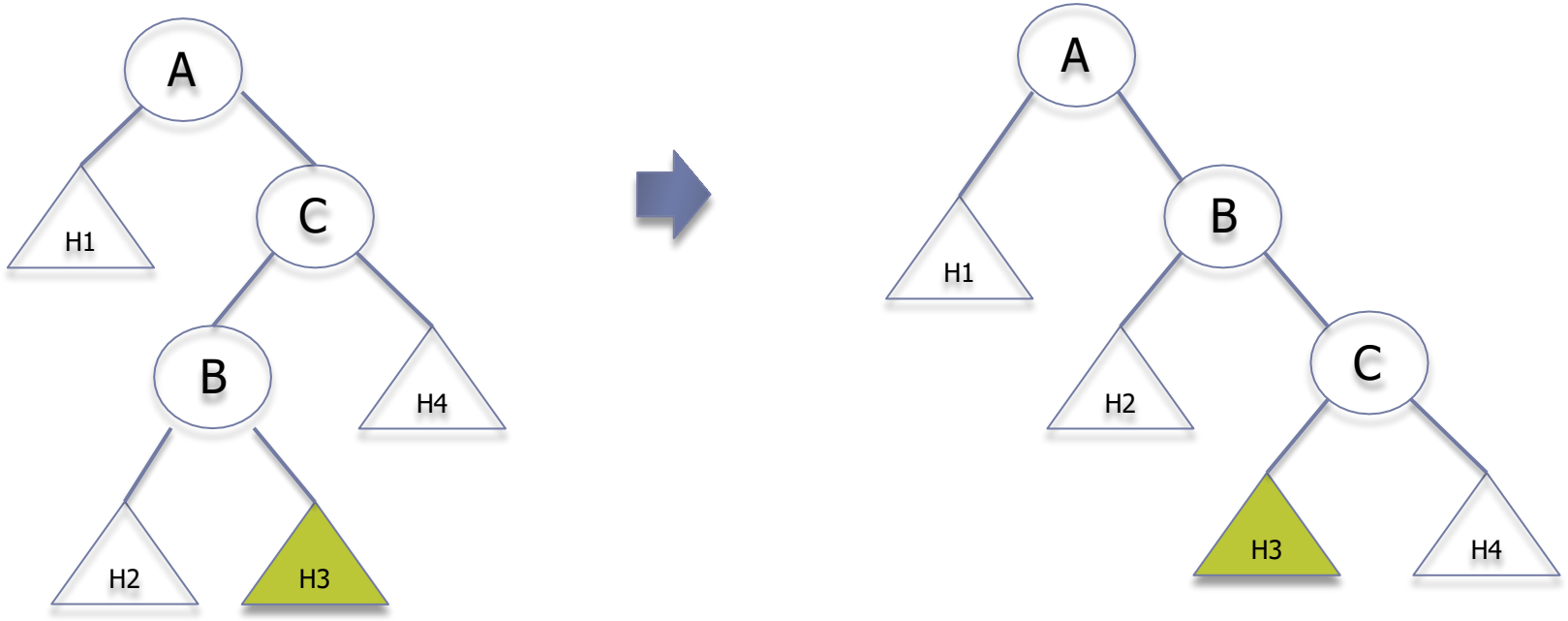
Doble Rotación Derecha Izquierda



Segunda rotación derecha: rotación simple izquierda sobre 60. En el árbol resultante, todos los nodos están equilibrados.

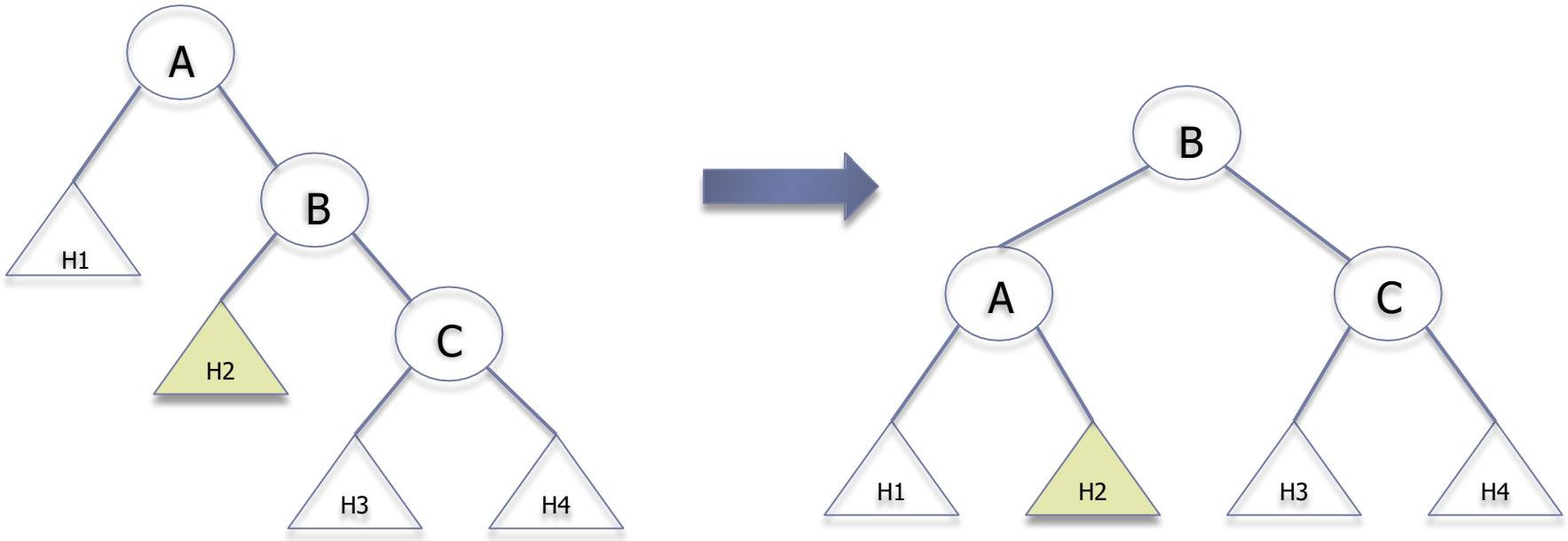


Doble Rotación Derecha Izquierda



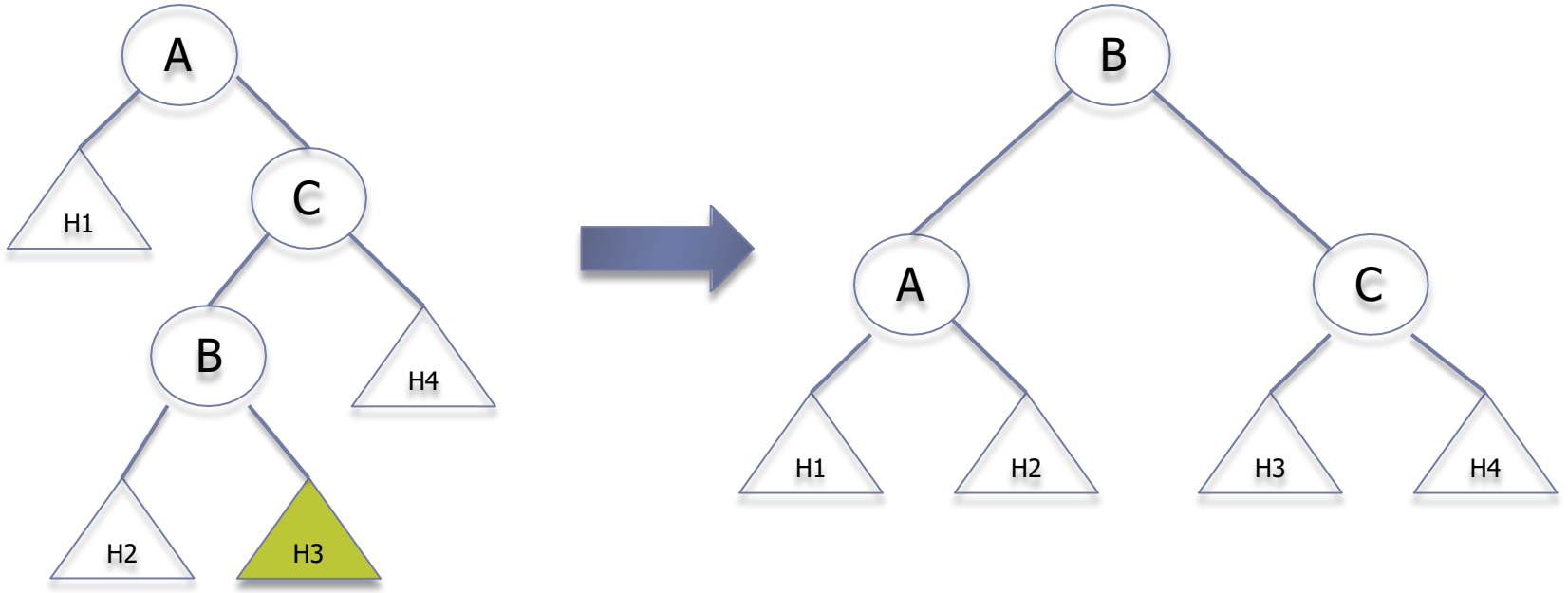
Primera rotación: B como hijo derecho de A y padre de C. El antiguo hijo derecho de B, debe pasar a ser hijo izquierdo de C

Doble Rotación Derecha Izquierda



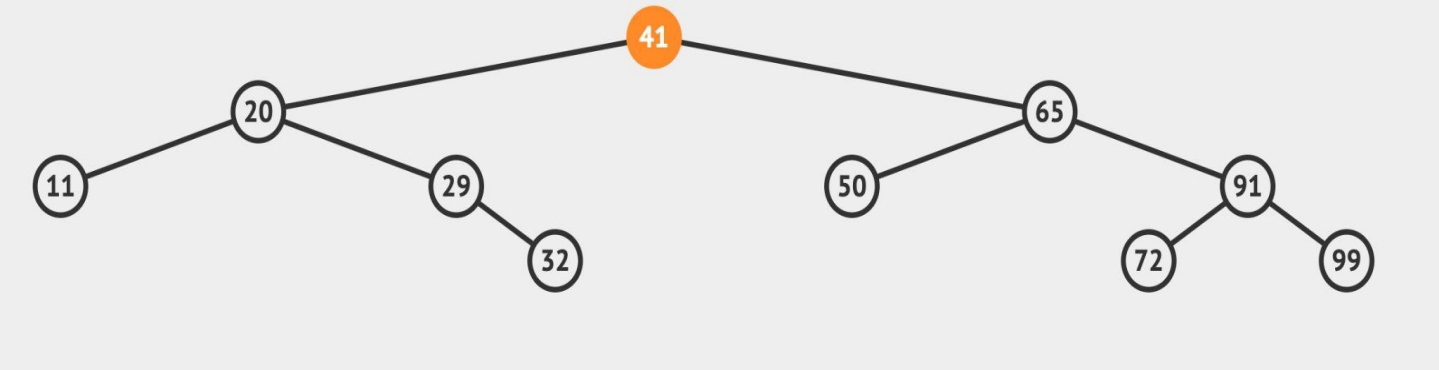
Segunda rotación: A gira a su izquierda y B se convierte en la nueva raíz. El antiguo hijo izquierdo de B debe pasar a ser hijo derecho de A, para que siga siendo un ABB.

Doble Rotación Derecha Izquierda

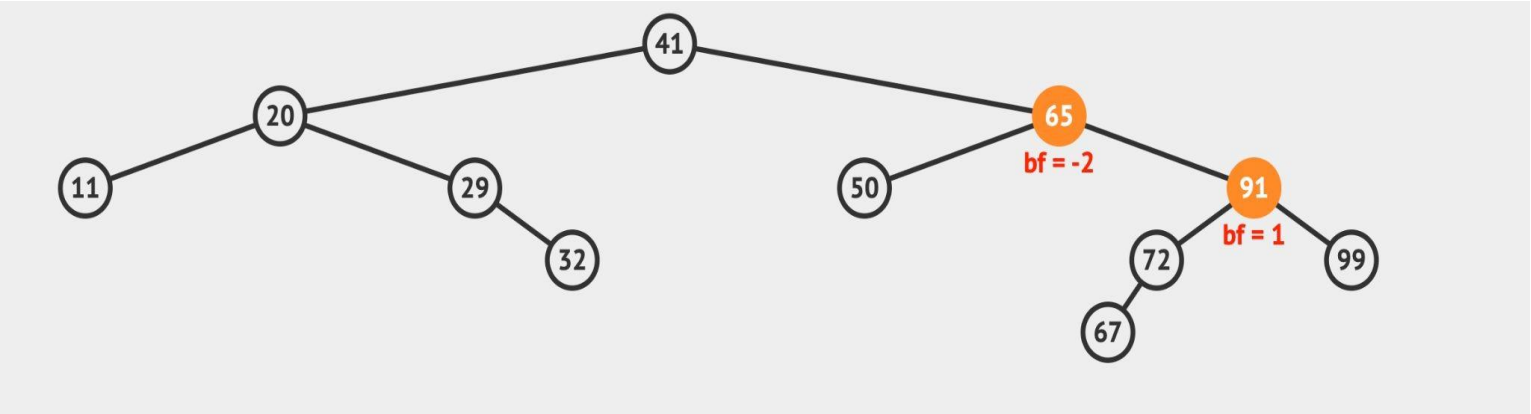


InOrder: $H_1AH_2BH_3CH_4$ = InOrder: $H_1AH_2BH_3CH_4$

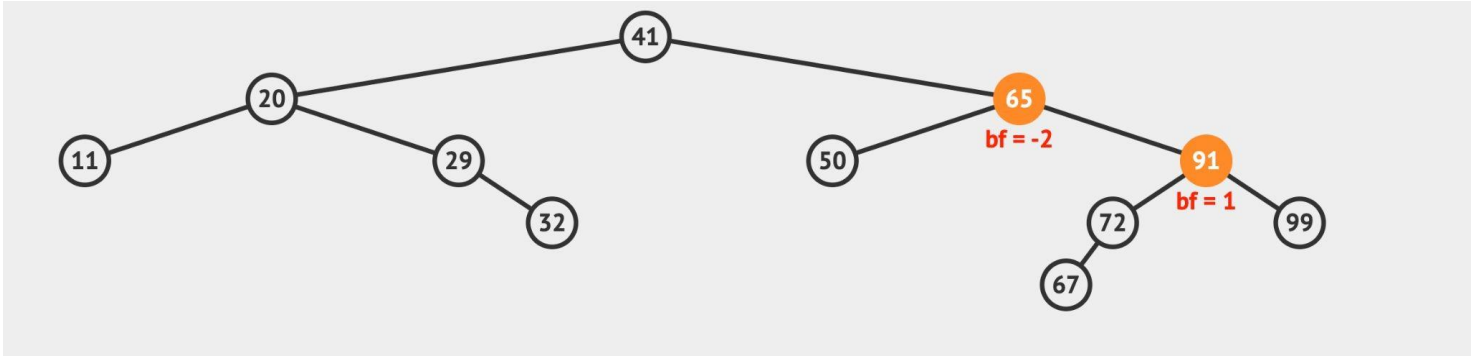
Doble Rotación Derecha Izquierda



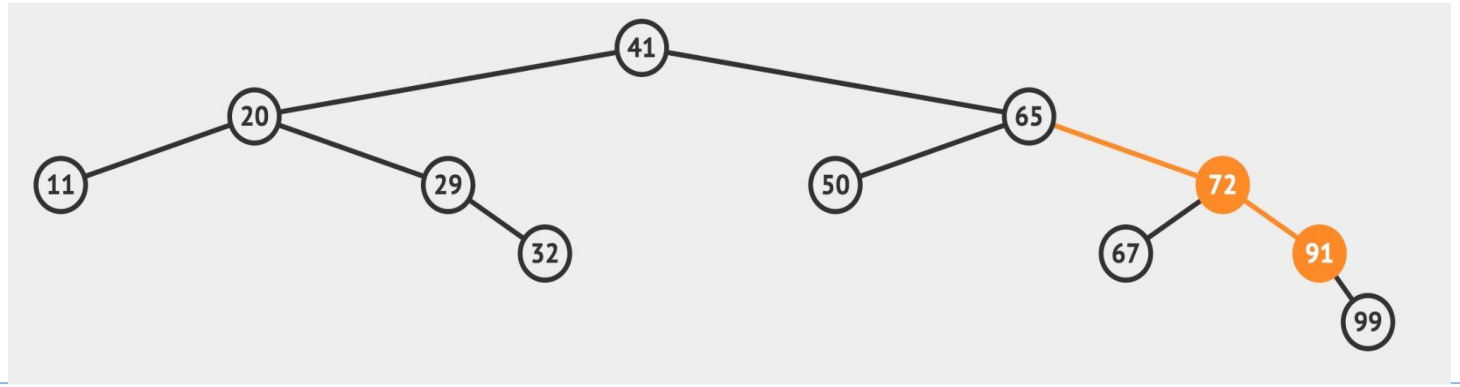
En el anterior árbol equilibrado, insertamos 67. Esto produce que el nodo 65 quede desbalanceado. Su rama más larga viene por la izquierda (91), y luego por la derecha (72). Debemos aplicar una doble rotación derecha, izquierda



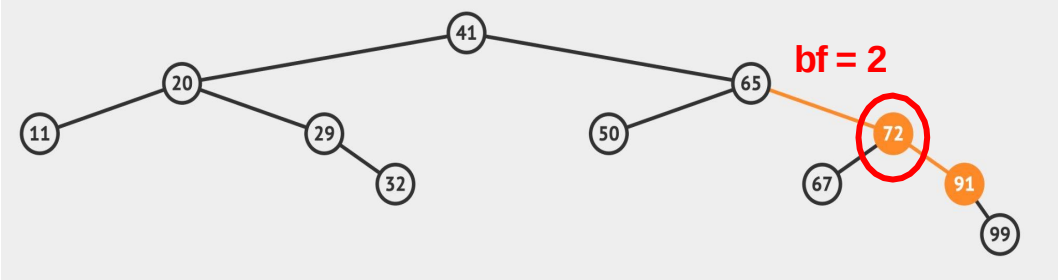
Doble Rotación Derecha Izquierda



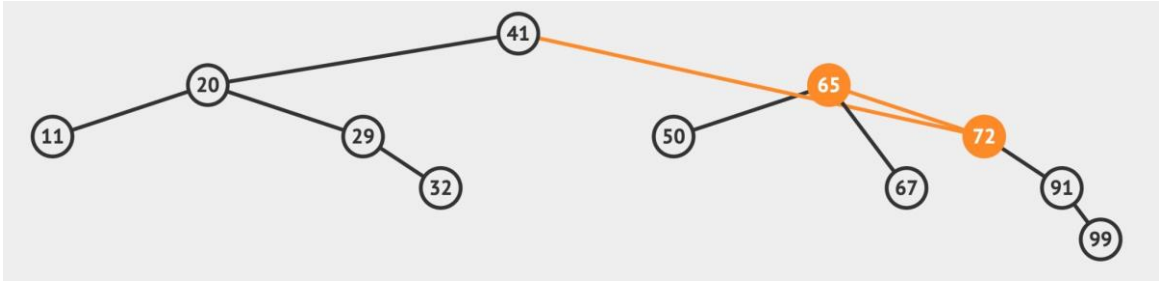
Primera rotación: simple derecha. El nodo 72 debe convertirse en el hijo derecho del nodo desequilibrado:65. El nodo 65 sigue desequilibrado y ahora ya podemos aplicarle la rotación simple izquierda.



Doble Rotación Derecha Izquierda

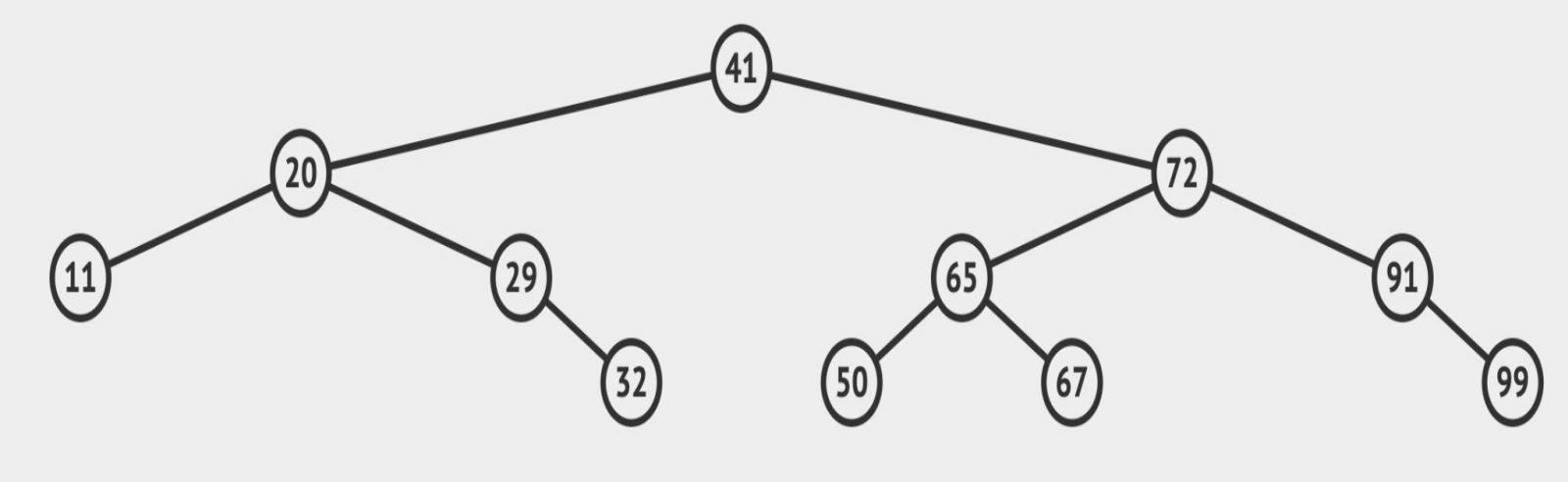


Segunda rotación: simple izquierda. El nodo con elemento 72 deberá convertirse en la nueva raíz del subárbol, y el nodo desbalanceado, nodo con elemento 65, deberá convertirse en su hijo izquierdo. Como el nodo 72 ya tiene hijo izquierdo, nodo con elemento 67, dicho nodo se convertirá en el hijo derecho del nodo con elemento 65.



Doble Rotación Derecha Izquierda

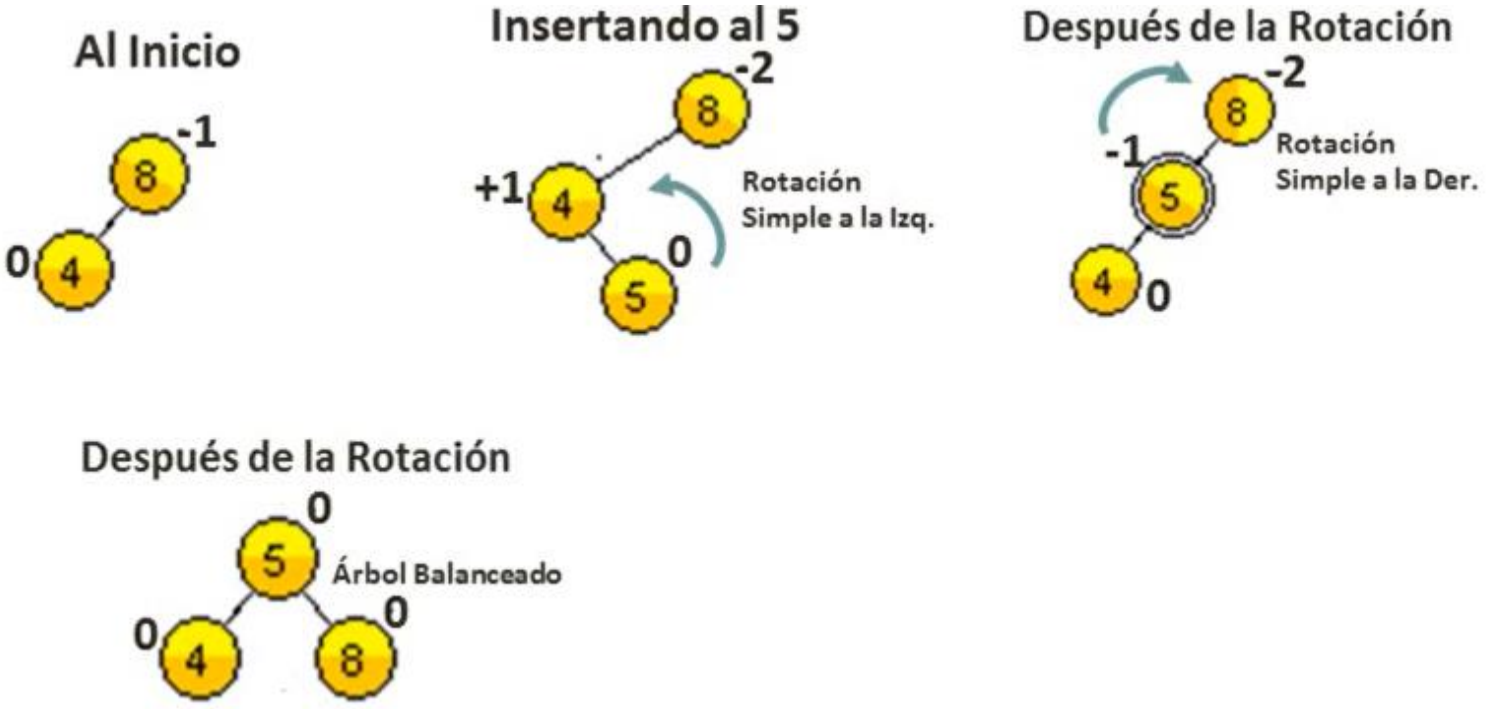
Finalmente, en el árbol resultante, todo los nodos ya están equilibrados



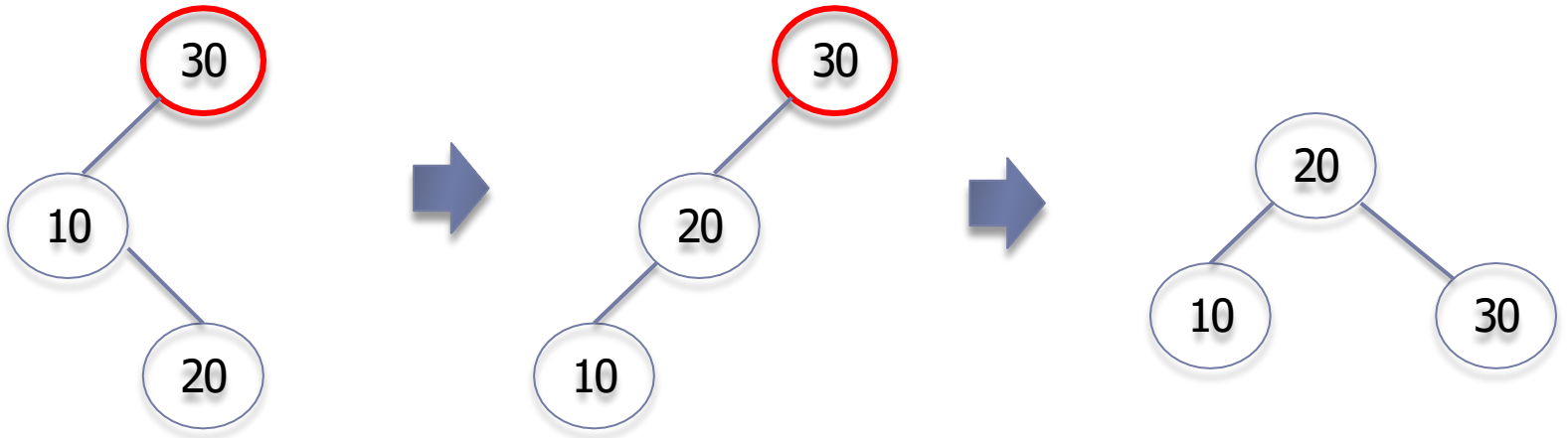
Árboles Binarios de Búsqueda AVL – Rotación doble a la izquierda

Si está desequilibrado a la derecha (FE< –1), y su hijo izquierdo tiene distinto signo (+) hacemos Rotación Doble Izquierda

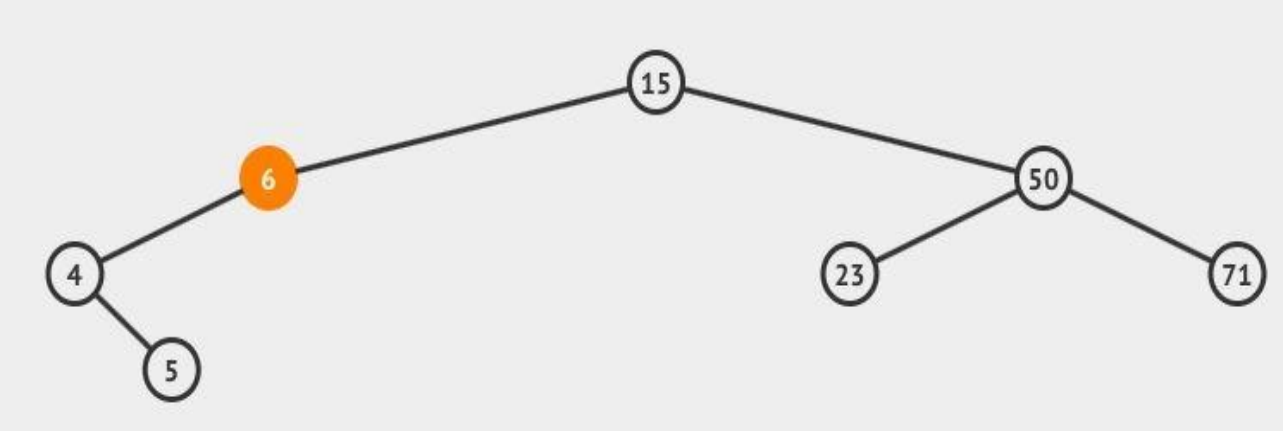
Doble Izquierda =
Simple Izquierda + Simple Derecha



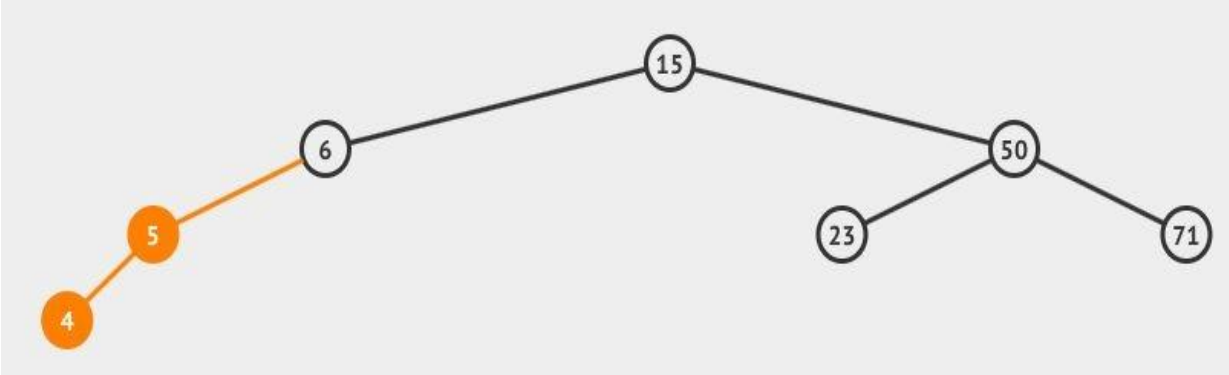
Doble Rotación Izquierda



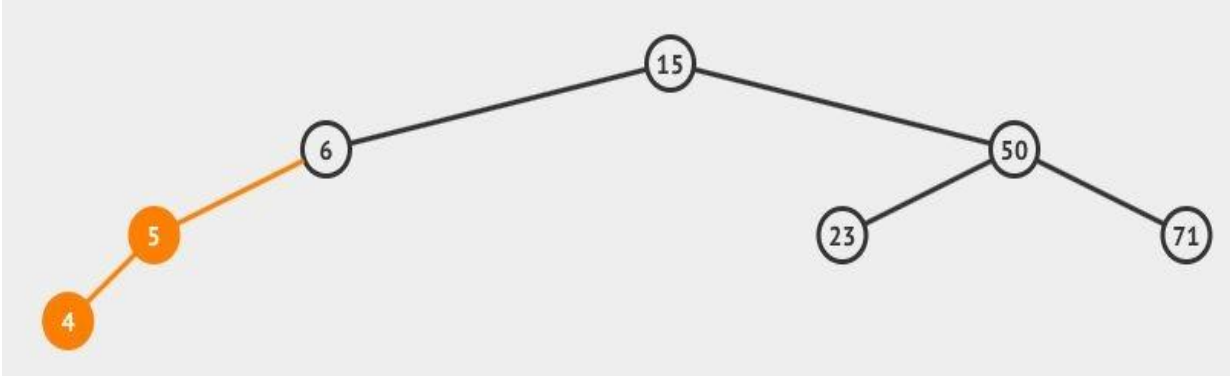
Doble Rotación Izquierda



El nodo 6 no está equilibrado. Primero debemos rotar el 4 (o 5) a la izquierda, y el 5 se convierte en el hijo izquierdo de 6.

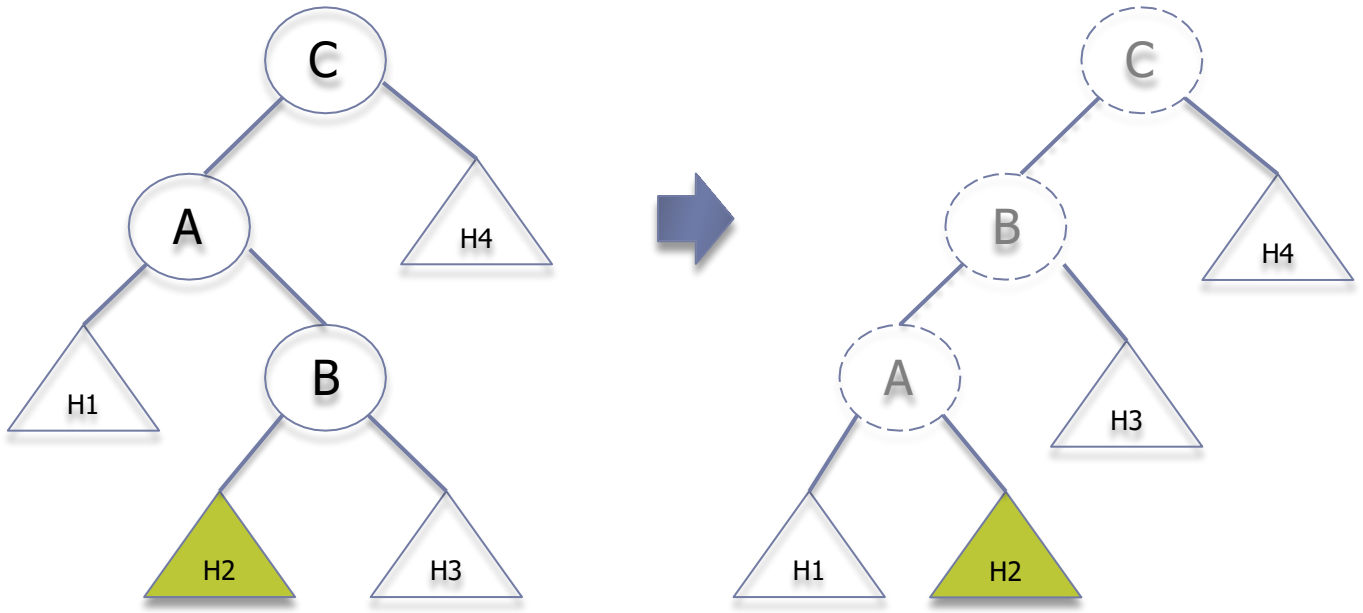


Doble Rotación Izquierda Derecha



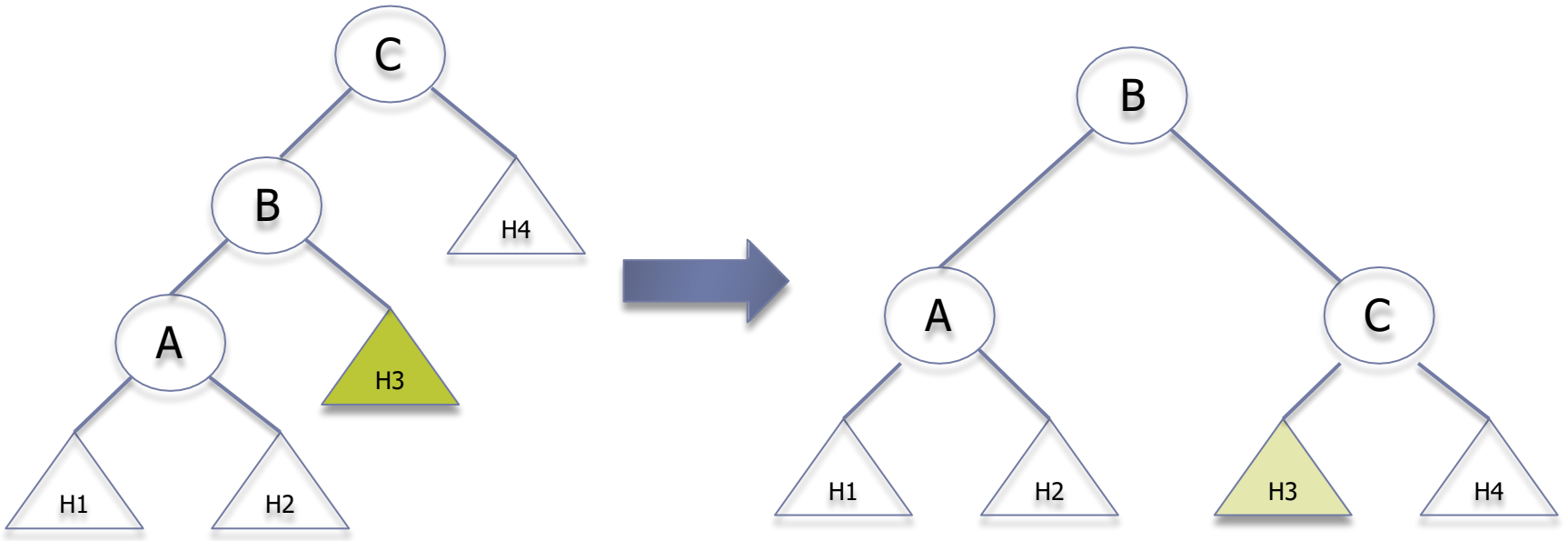
Después de haber realizado esa rotación a la izquierda, el nodo 6, sigue desequilibrado, pero ya simplemente tenemos que aplicar una rotación a la derecha. El árbol resultante habrá quedado equilibrado.

Doble Rotación Izquierda



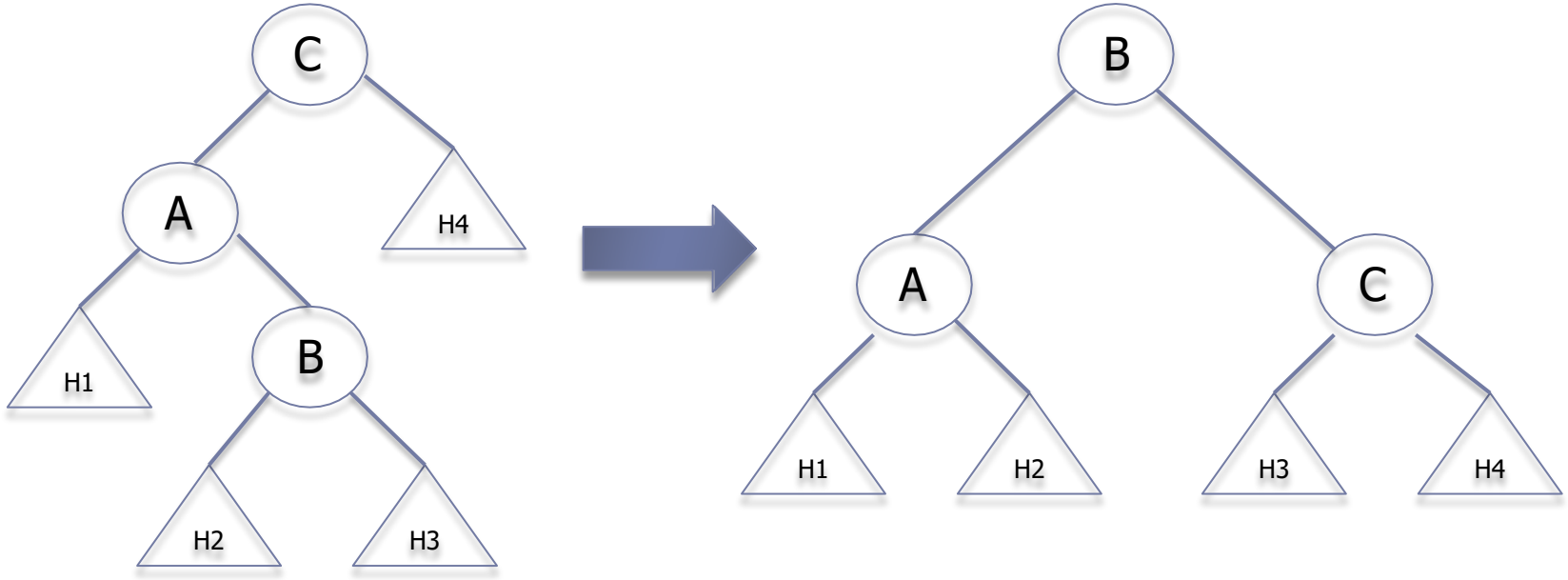
Primer paso, rotación izquierda: rotación de B como hijo izquierdo de C y padre de A. ¿Qué hacemos con sus subárboles? Su subárbol derecho no hay problema, pero su subárbol izquierdo tiene que pasar a ser hijo derecho de A

Doble Rotación Izquierda



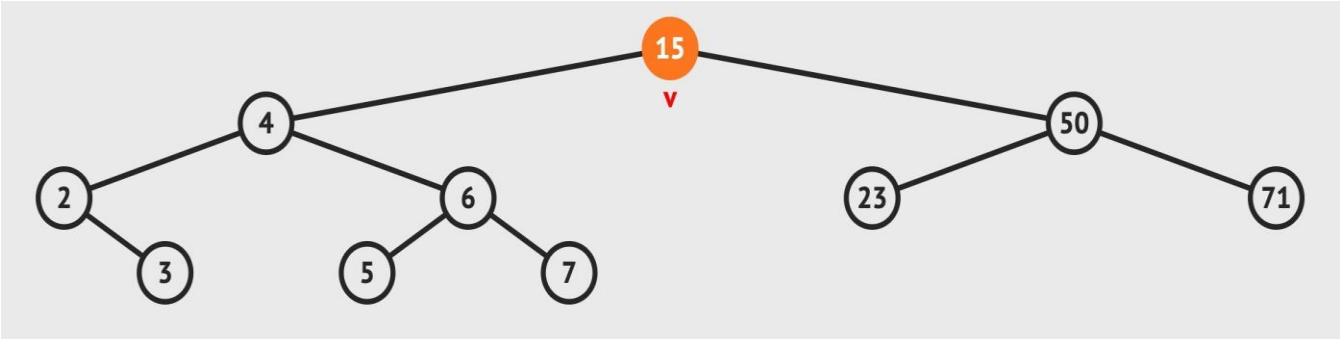
Segunda rotación, rotación derecha: B se convierte a ser la nueva raíz, tomando como nuevo hijo derecho a C. El antiguo hijo derecho de B tiene que pasar a ser hijo izquierdo de C.

Doble Rotación Izquierda

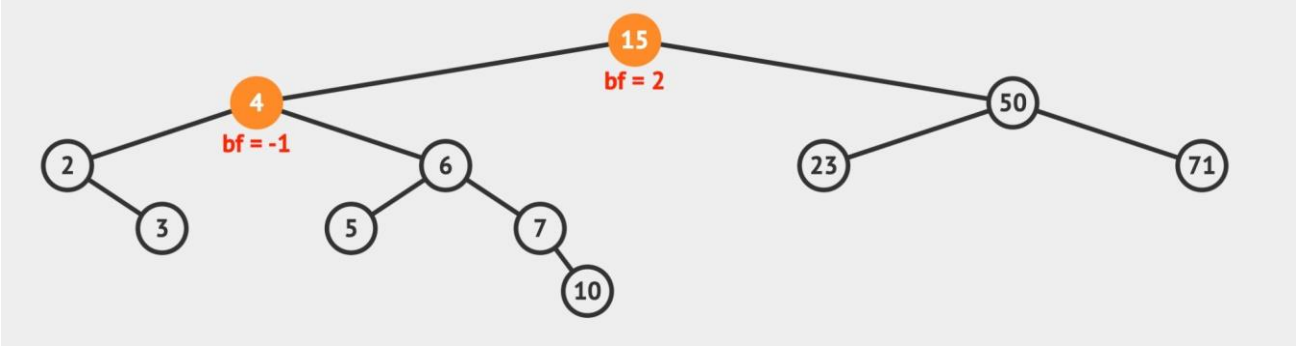


InOrder: $H_1AH_2BH_3CH_4$ = InOrder: $H_1AH_2BH_3CH_4$

Doble Rotación Izquierda

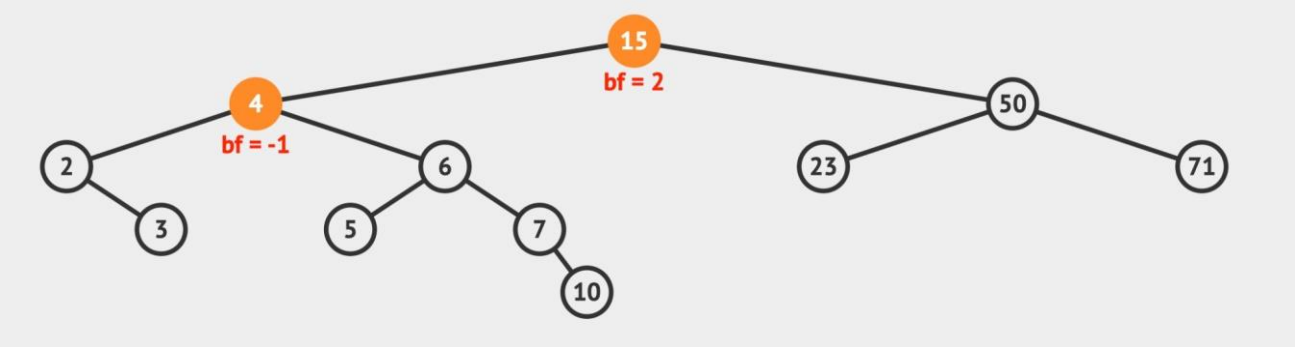


Si insertamos el 10, la raíz no está balanceada:

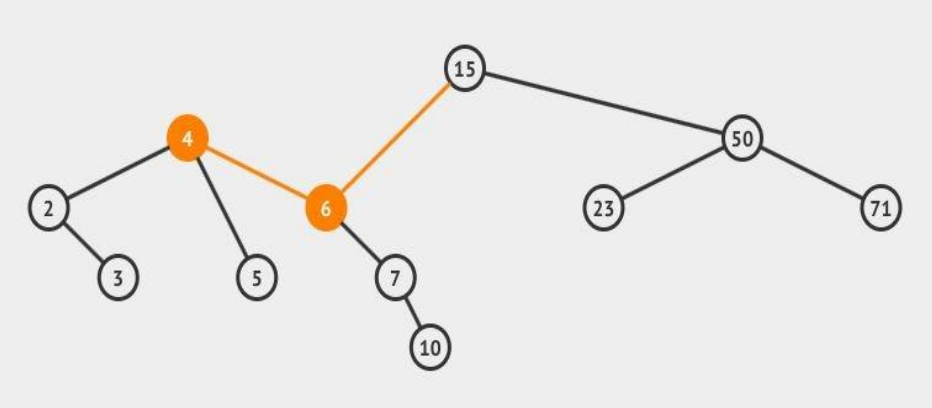


Su rama más larga viene por la izquierda (hasta el 4), y luego por la derecha (nodo 6). Por tanto, deberemos aplicar una rotación izquierda-derecha.

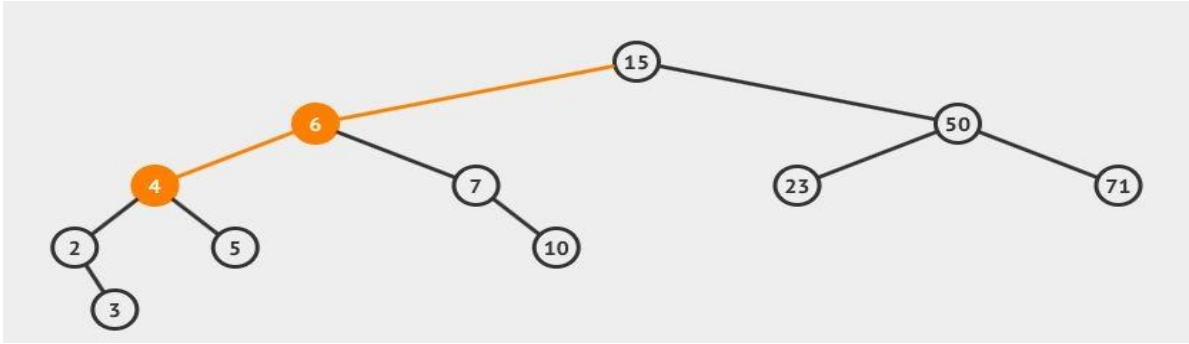
Doble Rotación Izquierda Derecha



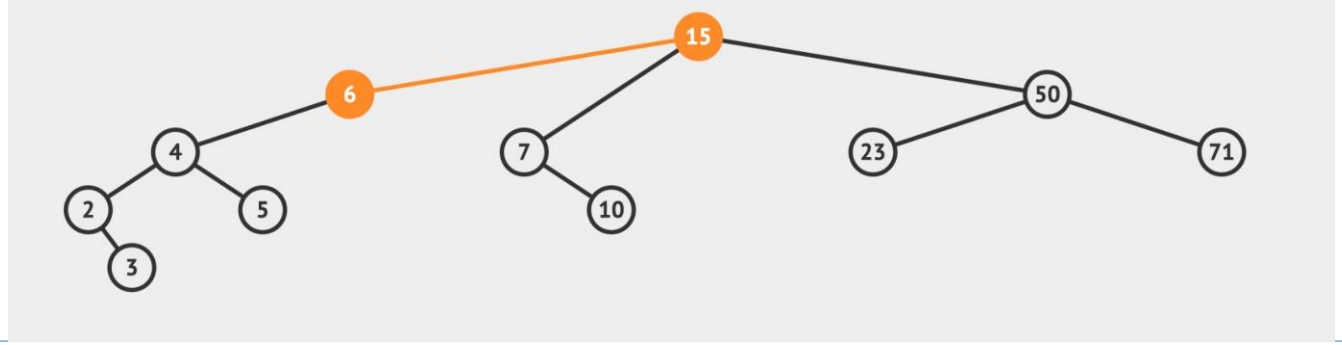
Primera rotación: rotación izquierda del nodo 4. El nodo 6 pasa a ser temporalmente la nueva raíz. Su nuevo hijo izquierdo será el nodo 4. Como el nodo 6 ya tiene hijo izquierdo, este debe pasar a ser hijo derecho del nodo 4.



Doble Rotación Izquierda

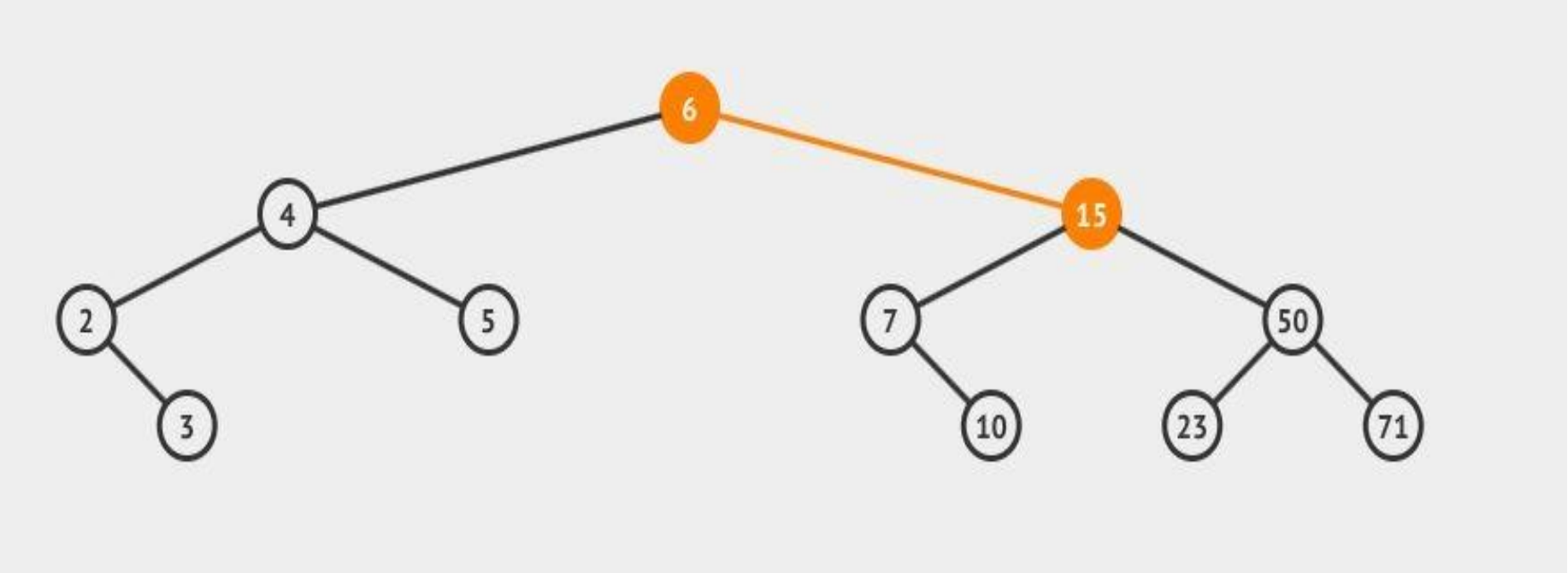


Segunda rotación: rotación derecha. El nodo desequilibrado (sigue siendo el nodo 15), deberá rotar a su subárbol derecho, y su hijo izquierdo se convertirá en el nueva raíz.
Pero antes de hacer eso, el nodo que será la nueva raíz, 6, tiene hijo derecho, el nodo 7. Ese hijo derecho deberá pasar a ser hijo izquierdo de 15. Date cuenta que 15 no puede tener 3 hijos!!!



Doble Rotación Izquierda Derecha

Finalmente, el nodo 6, ya se convierte en la nueva raíz, y el nodo 15, pasa a ser su hijo derecho. Este sería el árbol resultante donde todos los nodos están equilibrados (fe<=1), después de haber aplicado un doble rotación izquierda-derecha.



Clase NodoArbolAVL y ArbolAVL

Clase NodoArbolAVL

```
3   public class NodoArbolAVL {
4       int dato, fe;
5       NodoArbolAVL hijoIzquierdo, hijoDerecho;
6
7       public NodoArbolAVL(int d){
8           this.dato=d;
9           this.fe=0;
10          this.hijoIzquierdo=null;
11          this.hijoDerecho=null;
12      }
13  }
```

Clase ArbolAVL y algunas operaciones

```
2   public class ArbolAVL {
3       private NodoArbolAVL raiz;
4       public ArbolAVL(){
5           raiz=null;
6       }
7       public NodoArbolAVL obtenerRaiz(){
8           return raiz;
9       }
10      //Buscar
11      public NodoArbolAVL buscar( int d, NodoArbolAVL r){
12          if(raiz==null){
13              return null;
14          }else if(r.dato==d){
15              return r;
16          }else if(r.dato<d){
17              return buscar(d, r.hijoDerecho);
18          }else{
19              return buscar(d,r.hijoIzquierdo);
20          }
21      }
22      //Obtener el Factor de Equilibrio
23      public int obtenerFE(NodoArbolAVL x){
24          if(x==null){
25              return -1;
26          }else{
27              return x.fe;
28          }
29      }
```

Clase ArbolAVL con los métodos de rotación

```
27      //Rotación Simple Izquierda
28      public NodoArbolAVL rotacionIzquierda(NodoArbolAVL c){
29          NodoArbolAVL auxiliar=c.hijoIzquierdo;
30          c.hijoIzquierdo=auxiliar.hijoDerecho;
31          auxiliar.hijoDerecho=c;
32          c.fe=Math.max(obtenerFE(c.hijoIzquierdo),obtenerFE(c.hijoDerecho))+1;
33          auxiliar.fe=Math.max(obtenerFE(auxiliar.hijoIzquierdo),obtenerFE(auxiliar.hijoDerecho))+1;
34          return auxiliar;
35      }
36      //Rotación simple derecha
37      public NodoArbolAVL rotacionDerecha(NodoArbolAVL c){
38          NodoArbolAVL auxiliar=c.hijoDerecho;
39          c.hijoDerecho=auxiliar.hijoIzquierdo;
40          auxiliar.hijoIzquierdo=c;
41          c.fe=Math.max(obtenerFE(c.hijoIzquierdo),obtenerFE(c.hijoDerecho))+1;
42          auxiliar.fe=Math.max(obtenerFE(auxiliar.hijoIzquierdo),obtenerFE(auxiliar.hijoDerecho))+1;
43          return auxiliar;
44      }
45      //Rotación doble a la izquierda
46      public NodoArbolAVL rotacioDobleIzquierda(NodoArbolAVL c){
47          NodoArbolAVL temporal;
48          c.hijoIzquierdo=rotacionDerecha(c.hijoIzquierdo);
49          temporal=rotacionIzquierda(c);
50          return temporal;
51      }
52      //Rotación doble a la derecha
53      public NodoArbolAVL rotacioDobleDerecha(NodoArbolAVL c){
54          NodoArbolAVL temporal;
55          c.hijoDerecho=rotacionIzquierda(c.hijoDerecho);
56          temporal=rotacionDerecha(c);
57          return temporal;
58      }
```

Clase ArbolAVL con el método para insertar un nodo

```java
//Método para insertar
public void insertar(int d) {
    NodoArbolAVL nuevo = new NodoArbolAVL(d);
    if (raiz == null) {
        raiz = nuevo;
    } else {
        raiz = insertarAVL(nuevo, raiz);
    }
}
```

```java
//Metodo para insertar AVL
public NodoArbolAVL insertarAVL(NodoArbolAVL nuevo, NodoArbolAVL subAr) {
    NodoArbolAVL nuevoPadre = subAr;
    if (nuevo.dato < subAr.dato) {
        if (subAr.hijoIzquierdo == null) {
            subAr.hijoIzquierdo = nuevo;
        } else {
            subAr.hijoIzquierdo = insertarAVL(nuevo, subAr.hijoIzquierdo);
            if ((obtenerFE(subAr.hijoIzquierdo) - obtenerFE(subAr.hijoDerecho) == 2)) {
                if (nuevo.dato < subAr.hijoIzquierdo.dato) {
                    nuevoPadre = rotacionIzquierda(subAr);
                } else {
                    nuevoPadre = rotacionDobleIzquierda(subAr);
                }
            }
        }
    } else if (nuevo.dato > subAr.dato) {
        if (subAr.hijoDerecho == null) {
            subAr.hijoDerecho = nuevo;
        } else {
            subAr.hijoDerecho = insertarAVL(nuevo, subAr.hijoDerecho);
            if ((obtenerFE(subAr.hijoDerecho) - obtenerFE(subAr.hijoIzquierdo) == 2)) {
                if (nuevo.dato > subAr.hijoDerecho.dato) {
                    nuevoPadre = rotacionDerecha(subAr);
                } else {
                    nuevoPadre = rotacionDobleDerecha(subAr);
                }
            }
        }
    } else {
        System.out.println("Nodo Duplicado");
    }
    //Actualizando la altura
    if ((subAr.hijoIzquierdo == null) && (subAr.hijoDerecho != null)) {
        subAr.fe = subAr.hijoDerecho.fe + 1;
    } else if ((subAr.hijoDerecho == null) && (subAr.hijoIzquierdo != null)) {
        subAr.fe = subAr.hijoIzquierdo.fe + 1;
    } else {
        subAr.fe = Math.max(obtenerFE(subAr.hijoIzquierdo), obtenerFE(subAr.hijoDerecho)) +
    }
    return nuevoPadre;
}
```

Clase ArbolAVL con los métodos para recorrerlo y uso ejemplo

```
125     //Recorridos
126     public void entreOrden(NodoArbolAVL r) {
127         if (r != null) {
128             entreOrden(r.hijoIzquierdo);
129             System.out.print(r.dato + " ");
130             entreOrden(r.hijoDerecho);
131         }
132     }
133
134     public void preOrden(NodoArbolAVL r) {
135         if (r != null) {
136             System.out.print(r.dato + " ");
137             preOrden(r.hijoIzquierdo);
138             preOrden(r.hijoDerecho);
139         }
140     }
141
142     public void postOrden(NodoArbolAVL r) {
143         if (r != null) {
144             postOrden(r.hijoIzquierdo);
145             postOrden(r.hijoDerecho);
146             System.out.print(r.dato + " ");
147         }
148     }
```

Clase main ejemplo

```
2   public class TestArbolAVL {
3       public static void main(String[] args) {
4           ArbolAVL arbolAVL1 = new ArbolAVL();
5           arbolAVL1.insertar(10);
6           arbolAVL1.insertar(5);
7           arbolAVL1.insertar(13);
8           arbolAVL1.insertar(1);
9           arbolAVL1.insertar(6);
10          arbolAVL1.insertar(17);
11          arbolAVL1.insertar(16);
12          arbolAVL1.preOrden(arbolAVL1.obtenerRaiz());
13      }
14  }
```

Ejercicios

Ejercicios

- Dibujar el árbol AVL resultante al insertar los números 33, 25, 28, 40, 66, 18, 15, 100, 75, 50
- Elaborar el método para eliminar un nodo en un árbol AVL
- Elabore un menú para ejecutar los diferentes métodos del árbol AVL.

¿Preguntas?

FIN