ÁRBOLES EQUILIBRADOS 2002

GRUPO # 22

Alumnos:

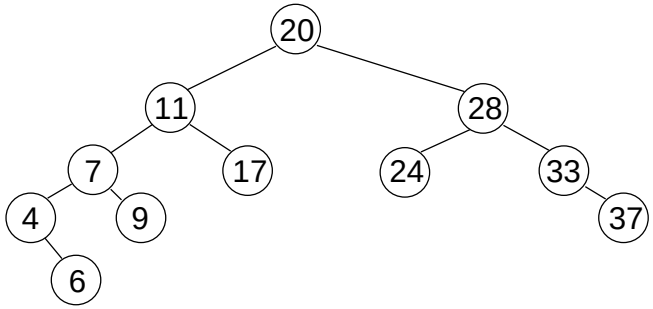
Aguilar Elba
Barrios Miguel
Camacho Yaquelin
Ponce Rodríguez Jhonny

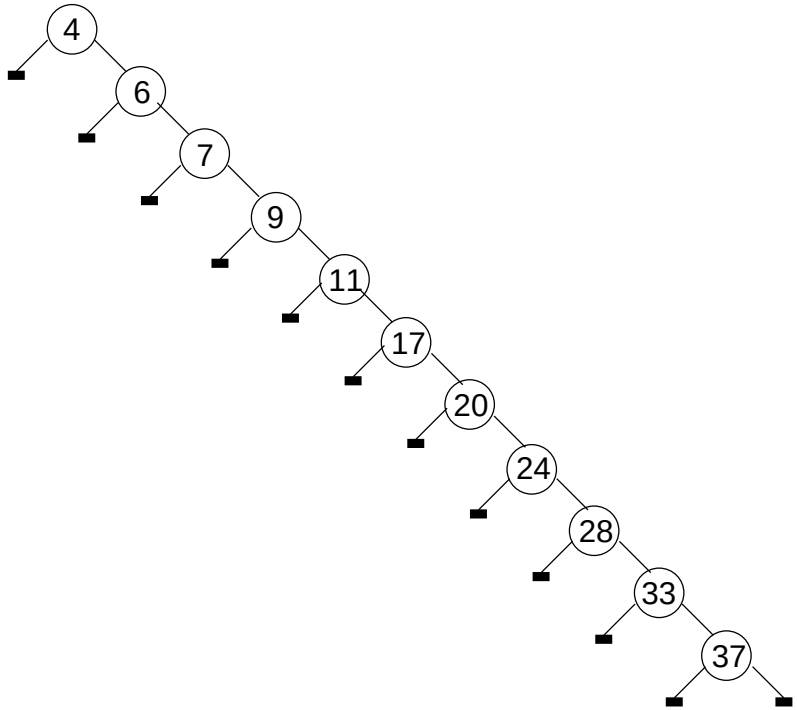
Dept. Sistemas Informáticos y Programación.
Universidad Complutense de Madrid

Árboles hilvanados

Motivación árboles degenerados La geometría de los árboles depende del orden de inserción en los mismos.

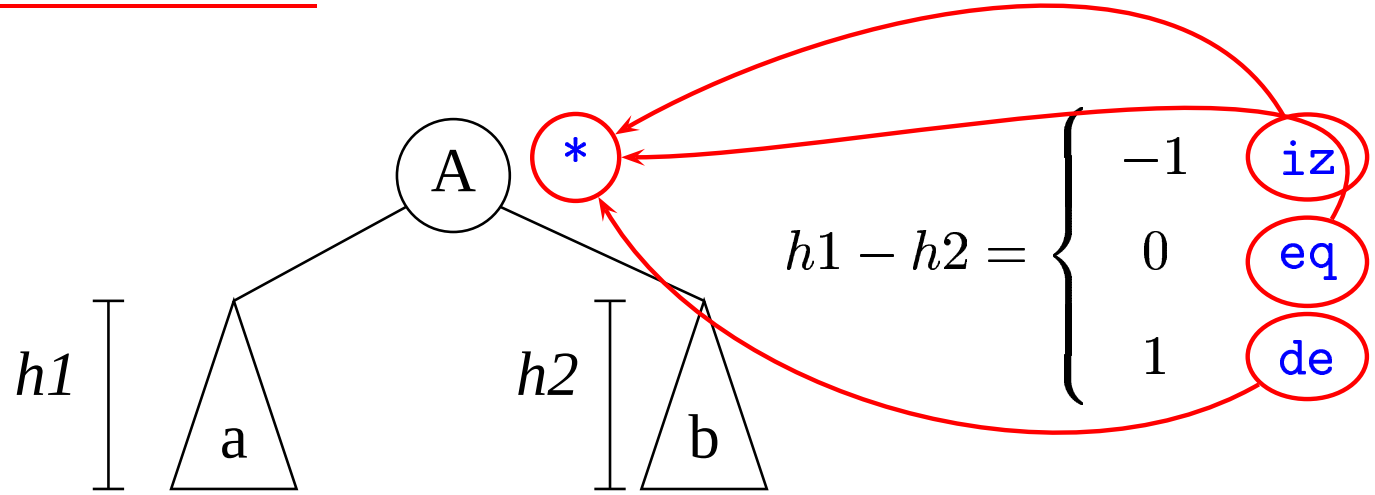
20 28 11 33 7 24 17 37 4 9 6 4 6 7 9 11 17 20 24 28 33 37





Árboles hilvanados

Árbol equilibrado la diferencia de las alturas de los hijos no supera a 1.



Las operaciones de inserción y borrado deben respetar esta propiedad.

```
package estructurasDatos;
import soporte.*;
public class NodoArbolHilvanEquilibrado extends NodoArbolHilvan {
    protected int indiceEquilibrio;
    protected final static int iz=-1;
    protected final static int de=1;
    protected final static int eq=0;

    public NodoArbolHilvanEquilibrado(ObjetoComparable o ) {
        super(o);
        indiceEquilibrio=eq;
    }
    public NodoArbolHilvanEquilibrado(ObjetoComparable o ,  Hilvan iz,  Hilvan der
{
        super(o,iz,der);
        indiceEquilibrio=eq;
    }
    protected String indice(){
        switch(indiceEquilibrio){
        case iz: return "iz";
        case de: return "de";
        case eq: return "eq";
        default: return "Desconocido";
        }
    }

}
```

Árboles hilvanados

Insertar algoritmo recursivo. Si aumenta el tamaño del hijo, puede ser necesario que reequilibrar.

Borrar algoritmo recursivo. Si disminuye el tamaño del hijo, puede ser necesario que reequilibrar.

```
package estructurasDatos;
import soporte.*;

class NodoHilvanCambiado extends NodoHilvan{
    boolean haCambiado;
}


public class ArbolBusquedaHilvanadoEquilibrado extends ArbolBusquedaHilvanado {
    public void insertar( ObjetoComparable x ){
        raiz=insertarEQ( x, raiz , ladoRaiz, null).hilvan;
    }
    private static NodoHilvanCambiado insertarEQ(ObjetoComparable x,
                                                 Hilvan hilvan,
                                                 int lado,
                                                 NodoArbolHilvan padre) {
        if( hilvan.tipo == Hilvan.tipoHilvan ){
            return insertarNodoEQ(x,hilvan,lado,padre);
        }
        else {
            NodoHilvanCambiado resultado;
            if( x.compararA( hilvan.puntero.informacion ) < 0 ){
                resultado=insertarEQ(x,
                                     hilvan.puntero.izquierdo,
                                     ladoIzquierdo,
                                     hilvan.puntero);
                hilvan.puntero.izquierdo=resultado.hilvan;
                resultado=reequilibrarInsercion(hilvan,
                                                resultado.haCambiado,
                                                ladoIzquierdo);
                return resultado;
            } else if( x.compararA( hilvan.puntero.informacion ) > 0 ){
                ..... caso simétrico al anterior .....
                ......................................
```

```
35                  } else {
                        hilvan.puntero.informacion=x;
                        resultado=new NodoHilvanCambiado();
                        resultado.hilvan=hilvan;
                        resultado.haCambiado=false;
40                      return resultado;
                    }
                }
            }

45          private static NodoHilvanCambiado insertarNodoEQ( ObjetoComparable x,
                                                               Hilvan hilvan,
                                                               int lado,
                                                               NodoArbolHilvan padre) {

                Hilvan iz,der;
50              NodoArbolHilvanEquilibrado actual;
                NodoHilvanCambiado resultado=new NodoHilvanCambiado();
                ..... codigo similar al de los árboles hilvanados .....
                resultado.haCambiado=true;
                return resultado;
55          }
60

65
```

```
        public void borrar( ObjetoComparable x ){
70          raiz = borrarEQ( x, raiz, ladoRaiz).hilvan ;
        }
        private NodoHilvanCambiado
            borrarEQ( ObjetoComparable x, Hilvan hilvan , int lado)
            {
75              NodoHilvanCambiado resultado;
                if( hilvan.tipo == Hilvan.tipoHilvan ){
                    resultado=new NodoHilvanCambiado();
                    resultado.haCambiado=false;
                    resultado.hilvan=hilvan;
80              } else {
                    NodoArbolHilvan nodo=hilvan.puntero;
                    if( x.compararA( nodo.informacion ) < 0 ){
                        resultado = borrarEQ( x, nodo.izquierdo , ladoIzquierdo);
                        nodo.izquierdo=resultado.hilvan;
85                      resultado = reequilibrarBorrado(hilvan,
                                                        resultado.haCambiado,
                                                        ladoIzquierdo);
                    }
                    else if( x.compararA( hilvan.puntero.informacion ) > 0 ){
90                      ..... simétrico al anterior ......
                    }
                    else {
                        if (nodo.derecho.tipo==Hilvan.tipoHilvan
                                && nodo.izquierdo.tipo==Hilvan.tipoHilvan){
95                          resultado=new NodoHilvanCambiado();
                                        //borrarNodo de la superclase
                            resultado.hilvan=borrarNodo(hilvan,lado);
                            resultado.haCambiado=true;
                        }
100
```

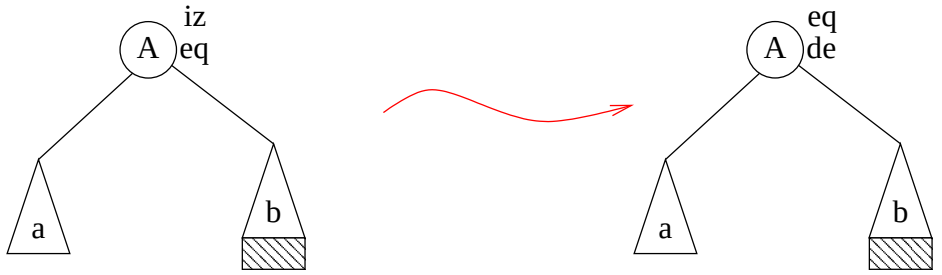
```
                    else if (nodo.derecho.tipo==Hilvan.tipoHilvan){
                                                    //buscarMax de la superclase
105                     NodoArbolHilvan nodoMax=buscarMax(nodo.izquierdo);
                        nodoMax.derecho=nodo.derecho;
                        resultado=new NodoHilvanCambiado();
                        resultado.hilvan=nodo.izquierdo;
                        resultado.haCambiado=true;
110                 }
                    else{
                        resultado = extraerMinEQ(hilvan.puntero.derecho,
                                                 ladoDerecho);
                        ((NodoArbolHilvanEquilibrado)resultado.nodo)
115                                                      .indiceEquilibrio =
                            ((NodoArbolHilvanEquilibrado)hilvan.puntero)
                                                        .indiceEquilibrio;
                        hilvan.puntero=resultado.nodo;
                        hilvan.puntero.izquierdo=nodo.izquierdo;
120                     hilvan.puntero.derecho=resultado.hilvan;
                        if (nodo.izquierdo.tipo==Hilvan.tipoPuntero){
                            NodoArbolHilvan nodoMax=buscarMax(nodo.izquierdo);
                            nodoMax.derecho=new Hilvan(hilvan.puntero,
                                                       Hilvan.tipoHilvan);
125                     }
                        resultado=reequilibrarBorrado(hilvan,
                                                      resultado.haCambiado,
                                                      ladoDerecho);
                    }
130             }
            }
            return resultado;
        }

135
```

```
        private static NodoHilvanCambiado extraerMinEQ(Hilvan hilvan, int lado) {
            NodoHilvanCambiado resultado;
            if (hilvan.puntero.izquierdo.tipo==Hilvan.tipoHilvan){
140             resultado=new NodoHilvanCambiado();
                resultado.nodo=hilvan.puntero;
                if (lado==ladoDerecho)
                    resultado.hilvan=hilvan.puntero.derecho;
                else //lado==ladoIzquierdo
145                 if (hilvan.puntero.derecho.tipo==Hilvan.tipoHilvan){
                        resultado.hilvan=hilvan;
                        resultado.hilvan.tipo=Hilvan.tipoHilvan;
                    }
                    else{
150                     resultado.hilvan=hilvan.puntero.derecho;
                    }
                resultado.haCambiado=true;
            }
            else{ //todavía no hemos alcanzado el mínimo
155             resultado=extraerMinEQ(hilvan.puntero.izquierdo, ladoIzquierdo);
                hilvan.puntero.izquierdo=resultado.hilvan;
                NodoArbolHilvan nodo=resultado.nodo;
                resultado=reequilibrarBorrado(hilvan,resultado.haCambiado,
                                              ladoIzquierdo);
160             resultado.nodo=nodo;
            }
            return resultado;
        }

165

170
```

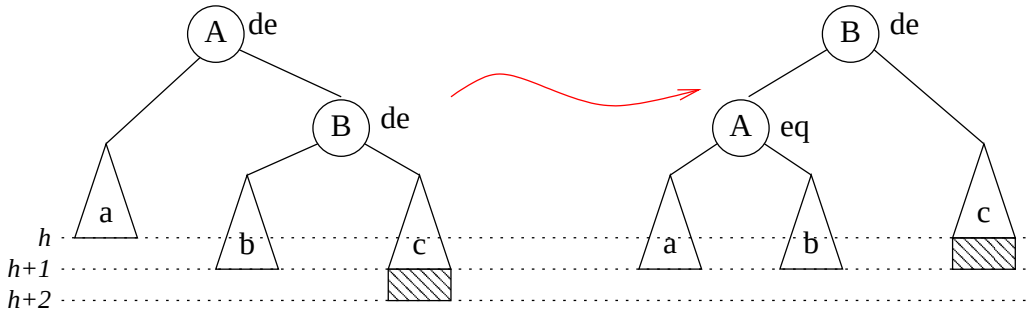
Inserción

Inserción en el subárbol derecho:

Sin rotación:



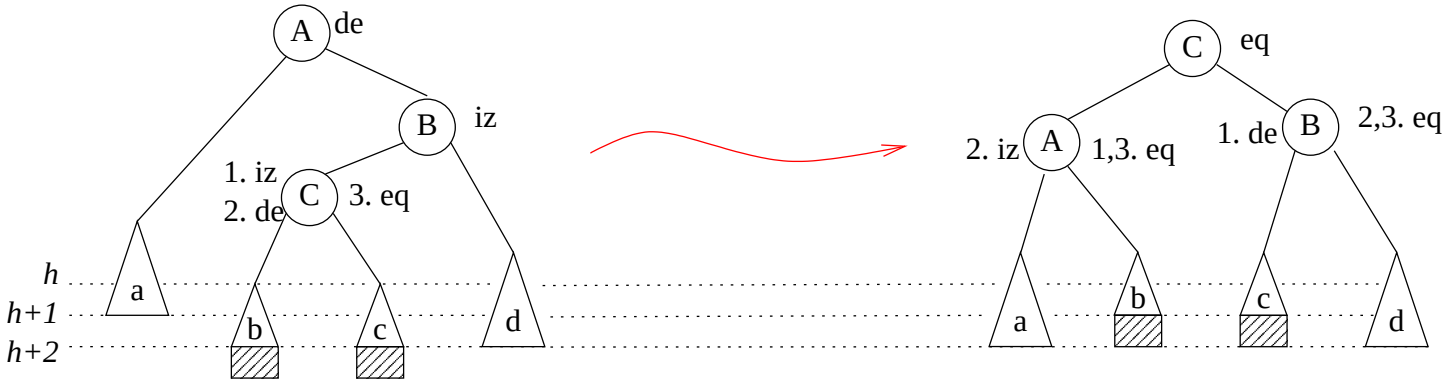
Rotación simple:



```
        private static NodoHilvanCambiado reequilibrarInsercion(Hilvan hilvan,
                                                                 boolean haCambiado,
                                                                 int lado) {
            NodoHilvanCambiado resultado=new NodoHilvanCambiado();
175         if (!haCambiado){
                resultado.hilvan=hilvan;
                resultado.haCambiado=false;
            } else {
                NodoArbolHilvanEquilibrado nodo=
180                 (NodoArbolHilvanEquilibrado)hilvan.puntero;
                switch (lado){
                case ladoDerecho:
                    switch(nodo.indiceEquilibrio){
                    case NodoArbolHilvanEquilibrado.iz:
185                     nodo.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                        resultado.hilvan=hilvan;
                        resultado.haCambiado=false;
                        break;
                    case NodoArbolHilvanEquilibrado.eq:
190                     nodo.indiceEquilibrio=NodoArbolHilvanEquilibrado.de;
                        resultado.hilvan=hilvan;
                        resultado.haCambiado=true;
                        break;
                    case NodoArbolHilvanEquilibrado.de:
195                     resultado.hilvan=rotacionDerechaInsercion(hilvan).hilvan;
                        resultado.haCambiado=false;
                        break;
                    default:
                        throw new ExcepcionInterna(
200                         "Indice de equilibrio incorrecto:"+nodo.indiceEquilibrio);
                    }
                    break;
```

```
205                 case ladoIzquierdo:
                          ..............  simétrico ..................
                        default:
                            throw new ExcepcionInterna(
                                "Indice de equilibrio incorrecto:"+nodo.indiceEquilibrio);
210                     }
                        break;
                    default:
                        throw new ExcepcionInterna("Lado descontrolado: "+lado);
                    }
215             }
                return resultado;
            }

            private static NodoHilvanCambiado rotacionDerechaInsercion(Hilvan hilvan)
220         {
                NodoArbolHilvanEquilibrado nodoA =
                    (NodoArbolHilvanEquilibrado)hilvan.puntero;
                NodoArbolHilvanEquilibrado nodoB =
                    (NodoArbolHilvanEquilibrado)nodoA.derecho.puntero;
225             NodoHilvanCambiado resultado=new NodoHilvanCambiado();
                Hilvan alfa,beta,gamma,delta;
                switch (nodoB.indiceEquilibrio){
                case NodoArbolHilvanEquilibrado.de: //rotación simple
                    beta=nodoB.izquierdo;
230                 if (beta.tipo==Hilvan.tipoHilvan)
                        nodoA.derecho=new Hilvan(nodoB,Hilvan.tipoHilvan);
                    else
                        nodoA.derecho=beta;
                    nodoB.izquierdo=new Hilvan(nodoA,Hilvan.tipoPuntero);
235                 nodoA.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                    nodoB.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                    resultado.hilvan=new Hilvan(nodoB,Hilvan.tipoPuntero);
                    break;
```

Rotación doble:



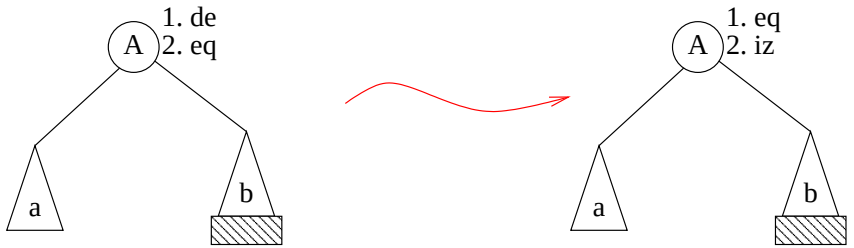
Después de la rotación el tamaño de árbol NO aumenta.

```
            case NodoArbolHilvanEquilibrado.iz: //rotación doble
240             NodoArbolHilvanEquilibrado nodoC =
                    (NodoArbolHilvanEquilibrado)nodoB.izquierdo.puntero;
                beta=nodoC.izquierdo; gamma=nodoC.derecho;
                if (beta.tipo==Hilvan.tipoHilvan)
                    nodoA.derecho=new Hilvan(nodoC,Hilvan.tipoHilvan);
245             else nodoA.derecho=beta;
                if (gamma.tipo==Hilvan.tipoHilvan)
                    nodoB.izquierdo=new Hilvan(nodoC,Hilvan.tipoHilvan);
                else nodoB.izquierdo=gamma;
                nodoC.izquierdo=new Hilvan(nodoA,Hilvan.tipoPuntero);
250             nodoC.derecho=new Hilvan(nodoB,Hilvan.tipoPuntero);
                switch (nodoC.indiceEquilibrio){
                case NodoArbolHilvanEquilibrado.de:
                    nodoA.indiceEquilibrio=NodoArbolHilvanEquilibrado.iz;
                    nodoB.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
255                 break;
                case NodoArbolHilvanEquilibrado.iz:
                    ............... caso similar ........
                case NodoArbolHilvanEquilibrado.eq:
                    ............... caso similar ........
260             default:
                    throw new ExcepcionInterna(
                        "Indice de equilibrio incorrecto del nodoC:"+nodoC.indice());
                }
                resultado.hilvan=new Hilvan(nodoC,Hilvan.tipoPuntero);
265             break;
            default:
                throw new ExcepcionInterna(
                   "Indice de equilibrio incorrecto del nodoB:"+nodoB.indice());
            }
270         resultado.haCambiado=false;
            return resultado;
        }
```

Borrado

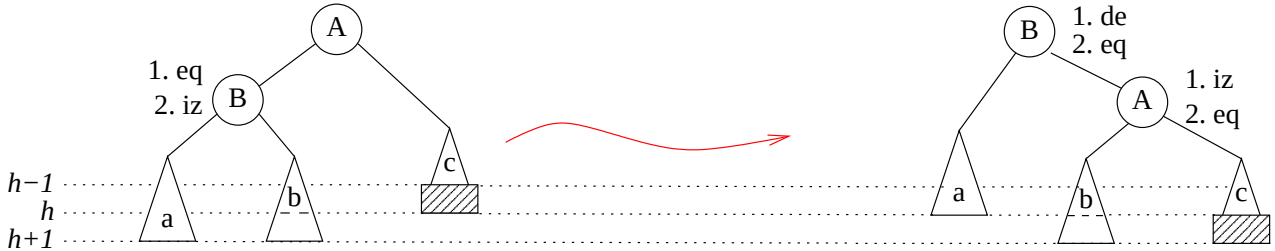
Borrado en el subárbol derecho:

Sin rotación:



1. Disminuye
2. No disminuye

Rotación simple:

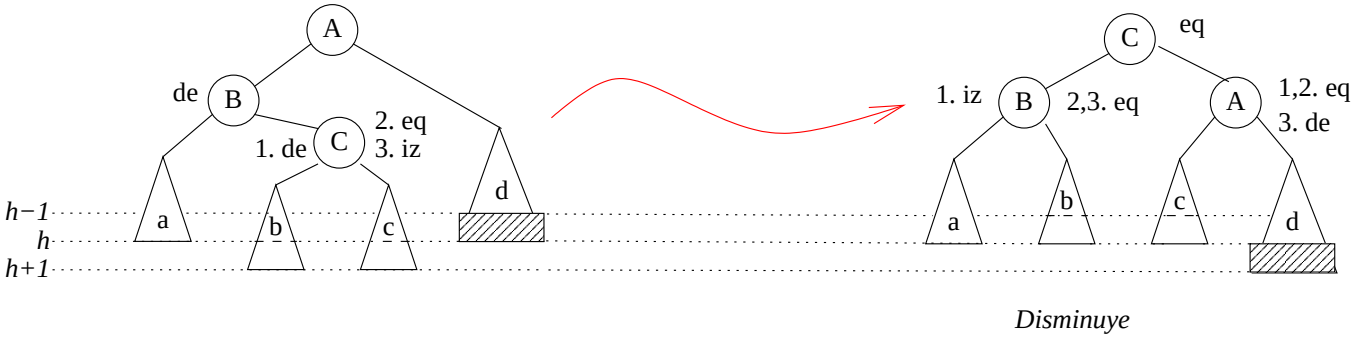


1. No disminuye
2. Disminuye

```
        private static NodoHilvanCambiado reequilibrarBorrado(Hilvan hilvan,
                                                               boolean haCambiado,
                                                               int lado) {
            NodoHilvanCambiado resultado;
            if (!haCambiado){
                resultado=new NodoHilvanCambiado();
                resultado.hilvan=hilvan;
                resultado.haCambiado=false;
            } else {
                NodoArbolHilvanEquilibrado nodo=(NodoArbolHilvanEquilibrado)hilvan.punt
    o;
                switch (lado){
                case ladoDerecho:
                    switch (nodo.indiceEquilibrio){
                    case NodoArbolHilvanEquilibrado.eq:
                        nodo.indiceEquilibrio=NodoArbolHilvanEquilibrado.iz;
                        resultado=new NodoHilvanCambiado();
                        resultado.hilvan=hilvan;
                        resultado.haCambiado=false;
                        break;
                    case NodoArbolHilvanEquilibrado.de:
                        nodo.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                        resultado=new NodoHilvanCambiado();
                        resultado.hilvan=hilvan;
                        resultado.haCambiado=true;
                        break;
                    case NodoArbolHilvanEquilibrado.iz:
                        resultado=rotacionDerechaBorrado(hilvan);
                        break;
                    default:
                        throw new ExcepcionInterna(
                            "Indice de equilibrio incorrecto:"+nodo.indice()+":");
                    }
                    break;
```

```
                case ladoIzquierdo:
                    .... caso simétrico .............
                default:
                    throw new ExcepcionInterna("Lado incorecto:"+lado+":");
310             }
            }
            return resultado;
        }
        private static NodoHilvanCambiado rotacionDerechaBorrado(Hilvan hilvan) {
315         NodoHilvanCambiado resultado=new NodoHilvanCambiado();
            NodoArbolHilvanEquilibrado nodoA=(NodoArbolHilvanEquilibrado)hilvan.puntero
            NodoArbolHilvanEquilibrado nodoB =
                (NodoArbolHilvanEquilibrado)nodoA.izquierdo.puntero;
            int indiceB=nodoB.indiceEquilibrio;
320         Hilvan beta,gamma;
            switch (indiceB){
            case NodoArbolHilvanEquilibrado.eq://rotación simple
            case NodoArbolHilvanEquilibrado.iz://rotación simple
                beta=nodoB.derecho;
325             nodoB.derecho=new Hilvan(nodoA,Hilvan.tipoPuntero);
                if (beta.tipo==Hilvan.tipoHilvan)
                    nodoA.izquierdo=new Hilvan(nodoB,Hilvan.tipoHilvan);
                else nodoA.izquierdo=beta;
                resultado.hilvan=new Hilvan(nodoB,Hilvan.tipoPuntero);
330             if (indiceB==NodoArbolHilvanEquilibrado.eq) {
                    nodoB.indiceEquilibrio=NodoArbolHilvanEquilibrado.de;
                    nodoA.indiceEquilibrio=NodoArbolHilvanEquilibrado.iz;
                    resultado.haCambiado=false;
                } else {//ndiceB==NodoArbolHilvanEquilibrado.iz
335                 nodoB.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                    nodoA.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                    resultado.haCambiado=true;
                }
                break;
```

Rotación doble:



En ocasiones, después de la rotación, la altura total disminuye y en otras no.

Por tanto serán necesarias ulteriores rotaciones.

```
340             case NodoArbolHilvanEquilibrado.de://rotación doble
                    NodoArbolHilvanEquilibrado nodoC =
                        (NodoArbolHilvanEquilibrado)nodoB.derecho.puntero;
                    beta=nodoC.izquierdo; gamma=nodoC.derecho;
                    nodoC.izquierdo=new Hilvan(nodoB,Hilvan.tipoPuntero);
345                 nodoC.derecho=new Hilvan(nodoA,Hilvan.tipoPuntero);
                    resultado.hilvan=new Hilvan(nodoC,Hilvan.tipoPuntero);
                    if (gamma.tipo==Hilvan.tipoHilvan)
                        nodoA.izquierdo=new Hilvan(nodoC,Hilvan.tipoHilvan);
                    else nodoA.izquierdo=gamma;
350                 if (beta.tipo==Hilvan.tipoHilvan)
                        nodoB.derecho=new Hilvan(nodoC,Hilvan.tipoHilvan);
                    else nodoB.derecho=beta;
                    int indiceC=nodoC.indiceEquilibrio;
                    nodoC.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
355                 switch (indiceC){
                    case NodoArbolHilvanEquilibrado.iz:
                        nodoB.indiceEquilibrio=NodoArbolHilvanEquilibrado.eq;
                        nodoA.indiceEquilibrio=NodoArbolHilvanEquilibrado.de;
                        break;
360                 case NodoArbolHilvanEquilibrado.eq:
                        ..... Caso similar ..................
                    case NodoArbolHilvanEquilibrado.de:
                        ..... Caso similar ..................
                    }
365                 resultado.haCambiado=true;
                    break;
                default:
                    throw new ExcepcionInterna("Índice de equilibrio del nodoB"+indiceB);
                }
370             return resultado;
            }
```