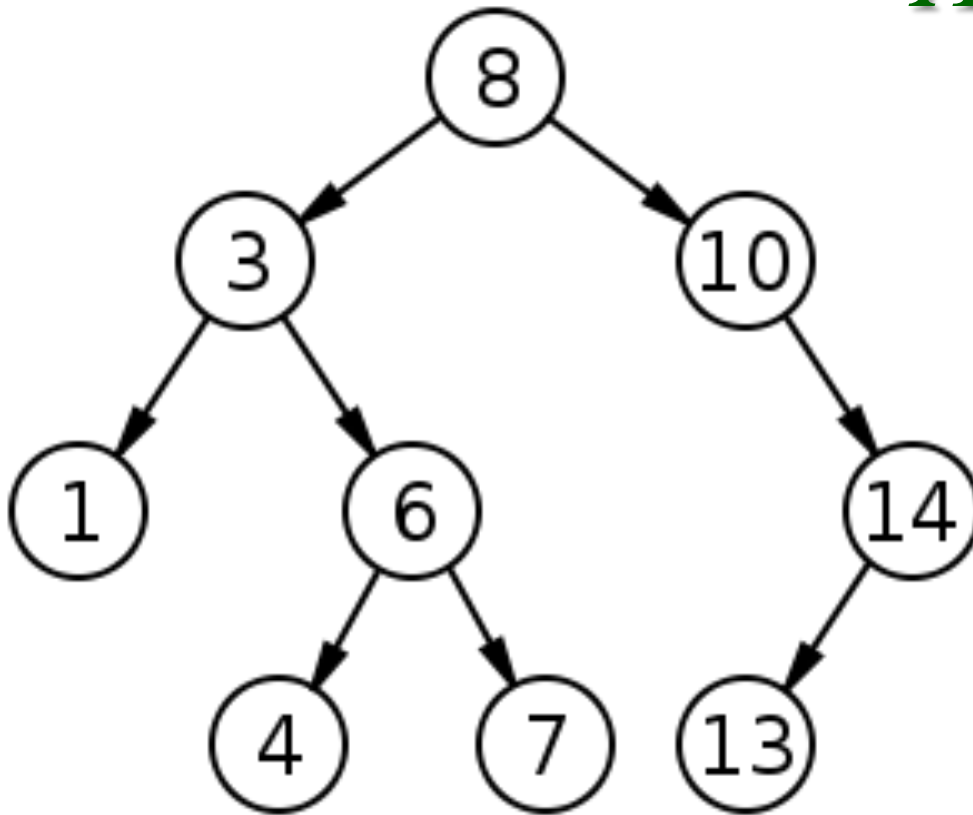




```
int main()  
{  
    return 0;  
}
```


# Árboles Binarios de búsqueda (AVL)



*Programación III.*



# Objetivos

- 
- A photograph of a large white ship with a dark hull, sailing on a blue sea. In the foreground, a vertical ruler is held up, partially obscuring the ship. The ruler has markings in centimeters and millimeters. The ship is centered in the frame, and the background is a hazy horizon.
- Repasar los conceptos de árboles.
  - Revisar aplicaciones con árboles equilibrados (AVL)

# Introducción

Como ya se revisó, el comportamiento de los árboles ABB, no es tan bueno como podría desearse. Para minimizar el problema de los ABB, sea cual sea el grado de desequilibrio que tengan, se puede recurrir a algoritmos de equilibrado de árboles globales. En cuanto a éstos algoritmos, existen varios, por ejemplo. Crear una lista mediante la lectura inorden del árbol, y volver a reconstruirlo equilibrado. Conociendo el número de elementos no es demasiado complicado.

# Definición

Un árbol AVL (Llamado así por las iniciales de sus inventores Adelson-Velskii y Landis) es un árbol binario de búsqueda en el que para cada nodo, las alturas de sus subárboles izquierdo y derecho no difieren en más de 1.

---

No se trata de árboles perfectamente equilibrados, pero si son lo suficientemente equilibrados como para que su comportamiento sea lo bastante bueno como para usarlos donde los ABB no garantizan tiempos de búsqueda óptimos. El algoritmo se basa en el reequilibrio local, de modo que no es necesario recorrer todo el árbol después de cada inserción o borrado

# Factor de equilibrio

Cada nodo, además de la información que se pretende almacenar, debe tener los dos punteros a los árboles derecho e izquierdo, igual que los ABB, y además un miembro nuevo: el factor de equilibrio.

El factor de equilibrio es la diferencia entre las alturas del árbol derecho y el izquierdo:

$FE = \text{altura subárbol derecho} - \text{altura subárbol izquierdo}$

Por definición, para un árbol AVL, este valor debe ser -1, 0 ó 1

# Rotación de nodos

---

Los reequilibrados se realizan mediante rotaciones, a continuación se muestran cuatro casos:

Rotación simple a la derecha (SD)

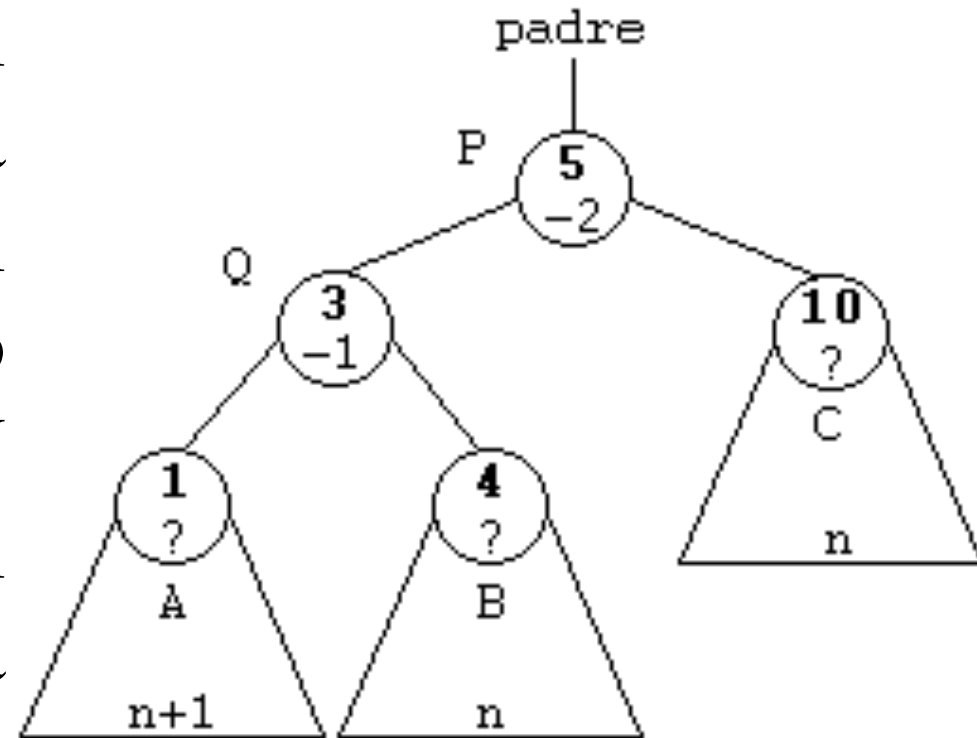
Rotación simple a la izquierda (SI)

Rotación doble a la derecha (DD)

Rotación doble a la izquierda (DI)

# Rotación simple a la derecha (SD)

Esta 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 ó 0, es decir, que esté cargado a la izquierda o equilibrado.





# Rotación simple a la derecha (SD)

Se procede del siguiente modo:

Llamaremos P al nodo que muestra el desequilibrio, el que tiene una FE de -2. Y llamaremos Q al nodo raíz del subárbol izquierdo de P. Además, llamaremos A al subárbol izquierdo de Q, B al subárbol derecho de Q y C al subárbol derecho de P.

En el gráfico que puede observar que tanto B como C tienen la misma altura ( $n$ ), y A es una unidad mayor ( $n+1$ ). Esto hace que el FE de Q sea -1, la altura del subárbol que tiene Q como raíz es ( $n+2$ ) y por lo tanto el FE de P es -2.

# Rotación simple a la derecha (SD)

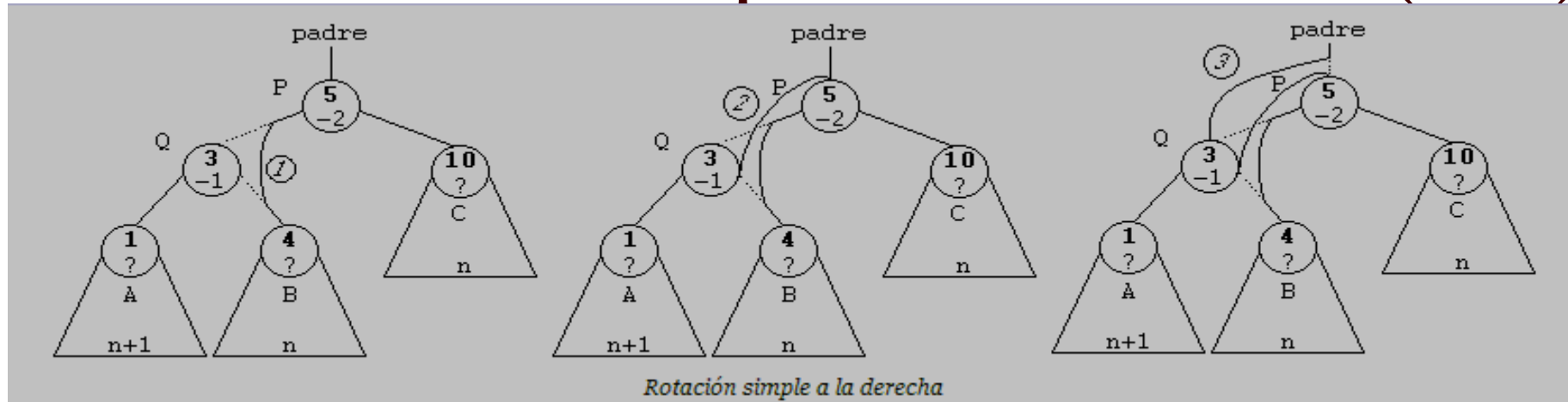
---

Pasamos el subárbol derecho del nodo Q como subárbol izquierdo de P. Esto mantiene el árbol como ABB, ya que todos los valores a la derecha de Q siguen estando a la izquierda de P.

El árbol P pasa a ser el subárbol derecho del nodo Q.

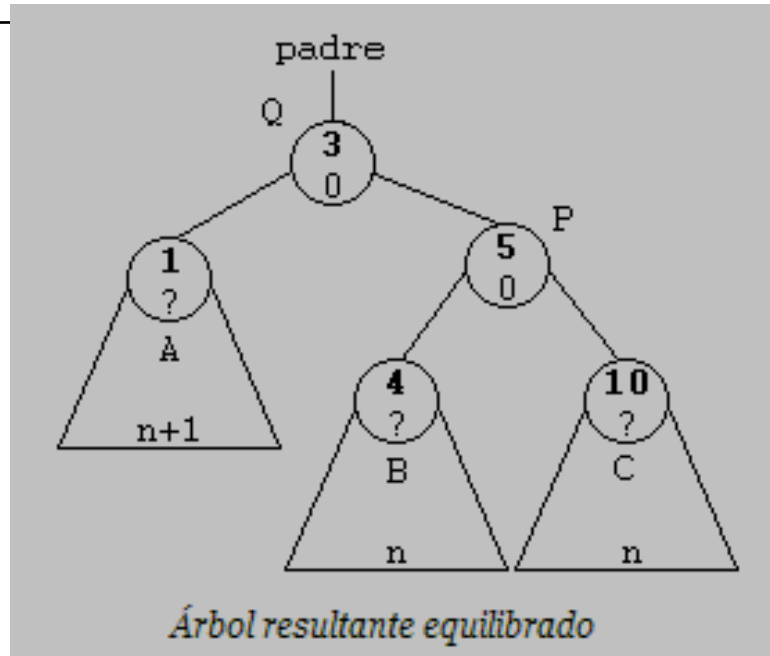
Ahora, el nodo Q pasa a tomar la posición del nodo P, es decir, hacemos que la entrada al árbol sea el nodo Q, en lugar del nodo P. Previamente, P puede que fuese un árbol completo o un subárbol de otro nodo de menor altura.

# Rotación simple a la derecha (SD)



En el árbol resultante se puede ver que tanto P como Q quedan equilibrados en cuanto altura. En el caso de P porque sus dos subárboles tienen la misma altura ( $n$ ), en el caso de Q, porque su subárbol izquierdo A tiene una altura ( $n+1$ ) y su subárbol derecho también, ya que a P se añade la altura de cualquiera de sus subárboles.

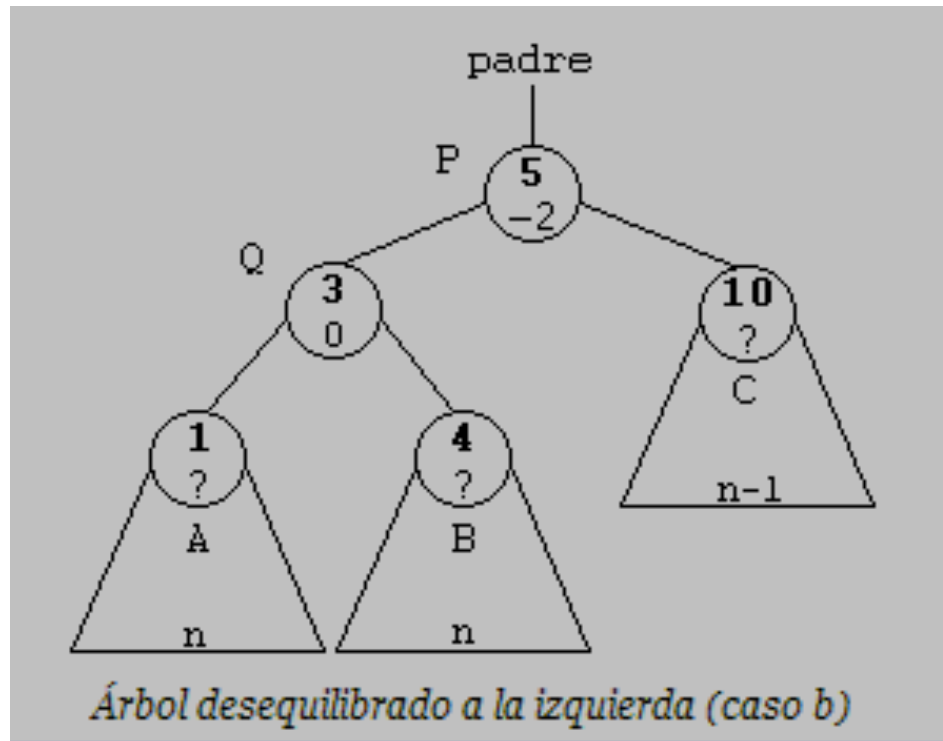
# Rotación simple a la derecha (SD)



En el caso de que el subárbol izquierdo esté equilibrado, el procedimiento es similar, pero los FE de los nodos P y Q en el árbol resultante son diferentes.

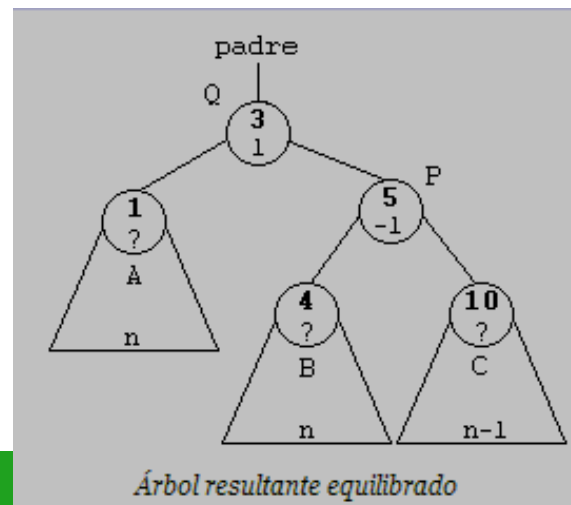
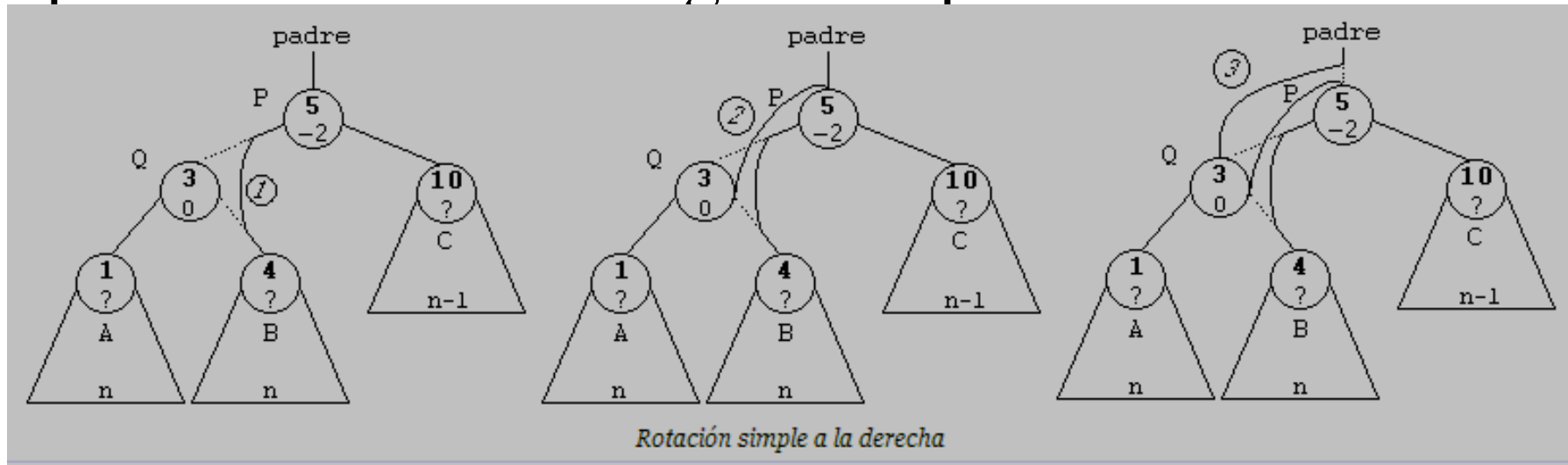
# Rotación simple a la derecha (SD)

En principio, parece poco probable que nos encontremos un árbol con esta estructura, pero es posible encontrarlos cuando se borran nodos.



# Rotación simple a la derecha (SD)

Aplicamos el mismo algoritmo para la rotación:



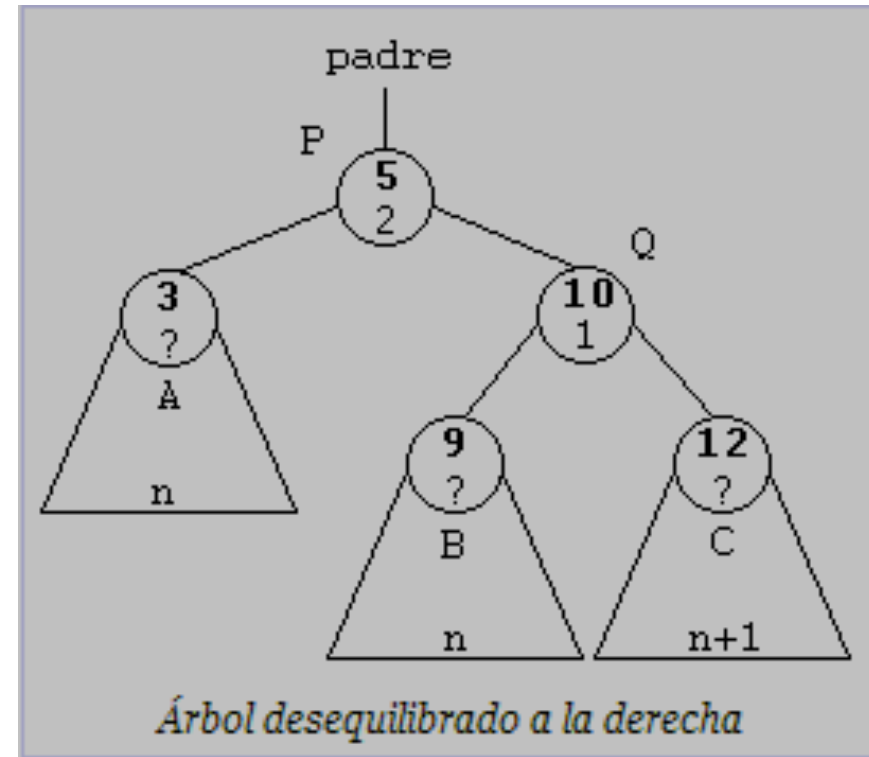
# Rotación simple a la derecha (SD)

En el árbol resultante se puede ver que tanto P como Q quedan equilibrados en cuanto altura. En el caso de P porque su subárbol izquierdo es una unidad más alto que el derecho, quedando su FE en -1. En el caso de Q, porque su subárbol derecho una altura  $(n+1)$  y su subárbol izquierdo, una altura de  $n$ .

De modo que, aunque aplicamos el mismo algoritmo, ya que en ambos casos se trata de una rotación simple, deberemos tener en cuenta estos detalles a la hora de ajustar los nuevos valores de FE en nuestro programa

# Rotación simple a la izquierda (SI)

Se trata del caso simétrico del anterior. Esta 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 una FE de 1 ó 0, es decir, que esté cargado a la derecha o esté equilibrado





# Rotación simple a la izquierda (SI)

Se procede del siguiente modo:

Llamaremos P al nodo que muestra el desequilibrio, el que tiene una FE de 2. Y llamaremos Q al nodo raíz del subárbol derecho de P. Además, llamaremos A al subárbol izquierdo de P, B al subárbol izquierdo de Q y C al subárbol derecho de Q.

En el gráfico que puede observar que tanto A como B tienen la misma altura ( $n$ ), y C es una unidad mayor ( $n+1$ ). Esto hace que el FE de Q sea 1, la altura del subárbol que tiene Q como raíz es ( $n+2$ ) y por lo tanto el FE de P es 2.

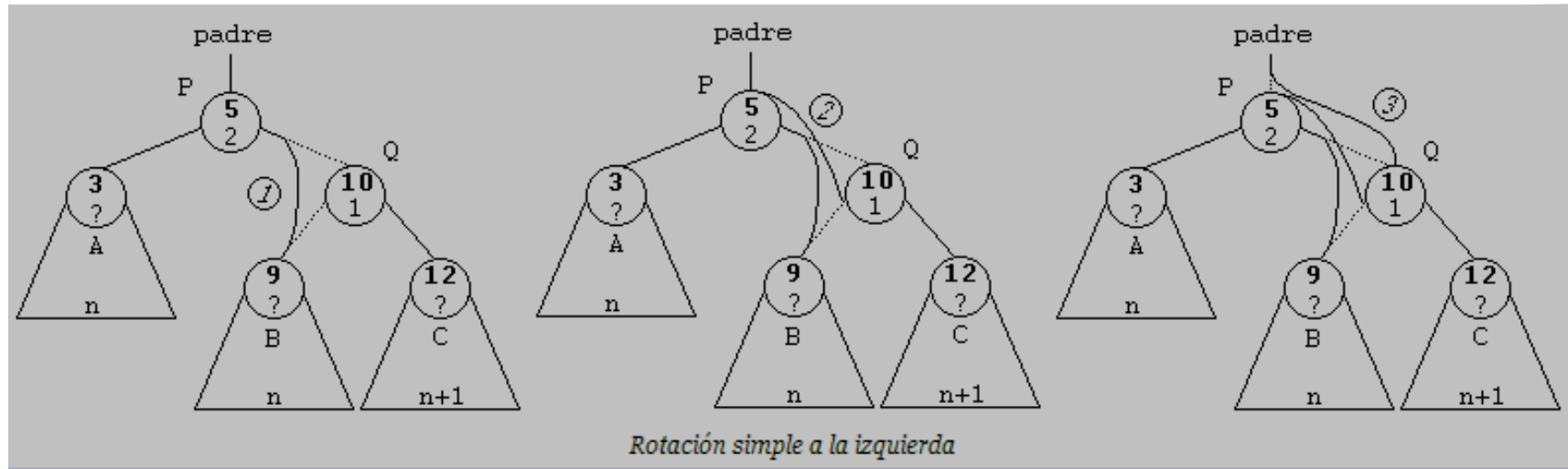
# Rotación simple a la izquierda (SI)

Pasamos el subárbol izquierdo del nodo Q como subárbol derecho de P. Esto mantiene el árbol como ABB, ya que todos los valores a la izquierda de Q siguen estando a la derecha de P.

El árbol P pasa a ser el subárbol izquierdo del nodo Q.

Ahora, el nodo Q pasa a tomar la posición del nodo P, es decir, hacemos que la entrada al árbol sea el nodo Q, en lugar del nodo P. Previamente, P puede que fuese un árbol completo o un subárbol de otro nodo de menor altura

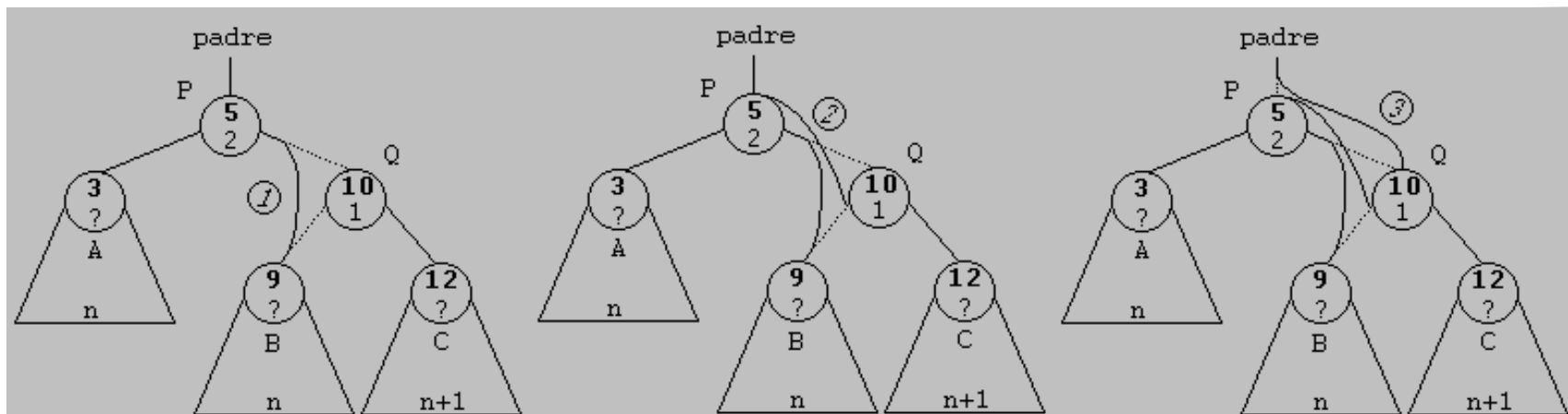
# Rotación simple a la izquierda (SI)



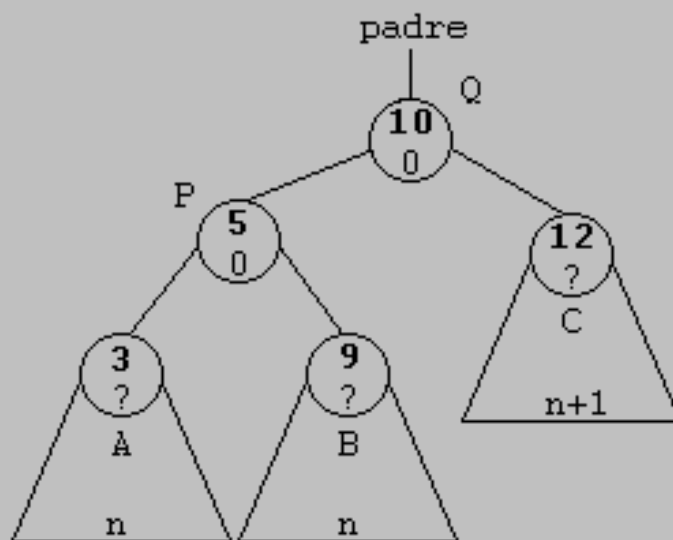
En el árbol resultante se puede ver que tanto P como Q quedan equilibrados en cuanto altura. En el caso de P porque sus dos subárboles tienen la misma altura ( $n$ ), en el caso de Q, porque su subárbol izquierdo A tiene una altura ( $n+1$ ) y su subárbol derecho también, ya que a P se añade la altura de cualquiera de sus subárboles

```
int main()
{
    return 0;
}
```

# Rotación simple a la izquierda (SI)



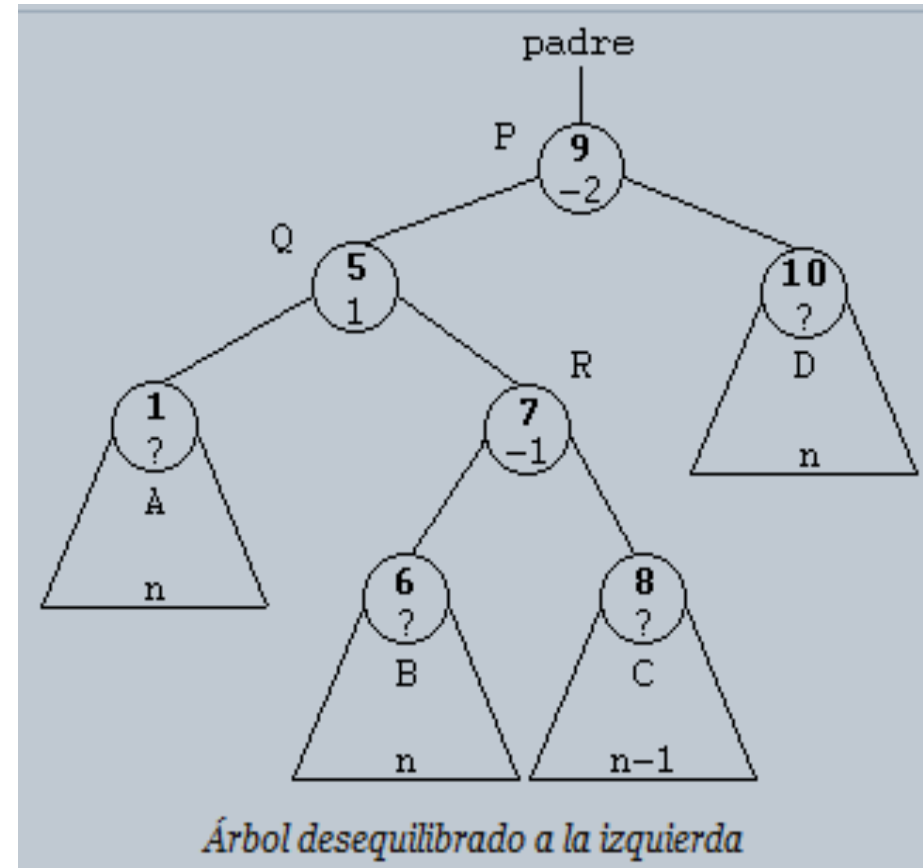
*Rotación simple a la izquierda*



*Árbol resultante equilibrado*

# Rotación doble la derecha (DD)

Esta 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 derecha



# Rotación doble la derecha (DD)

Este es uno de los posibles árboles que pueden presentar esta estructura, pero hay otras dos posibilidades. El nodo R puede tener una FE de -1, 0 ó 1. En cada uno de esos casos los árboles izquierdo y derecho de R (B y C) pueden tener alturas de  $n$  y  $n-1$ ,  $n$  y  $n$ , o  $n-1$  y  $n$ , respectivamente.

El modo de realizar la rotación es independiente de la estructura del árbol R, cualquiera de las tres produce resultados equivalentes. Haremos el análisis para el caso en que FE sea -1

# Rotación doble la derecha (DD)

En este caso tendremos que realizar dos rotaciones.

- ❑ Llamaremos P al nodo que muestra el desequilibrio, el que tiene una FE de -2. Llamaremos Q al nodo raíz del subárbol izquierdo de P, y R al nodo raíz del subárbol derecho de Q.
- ❑ Haremos una rotación simple de Q a la izquierda.
- ❑ Después, haremos una rotación simple de P a la derecha

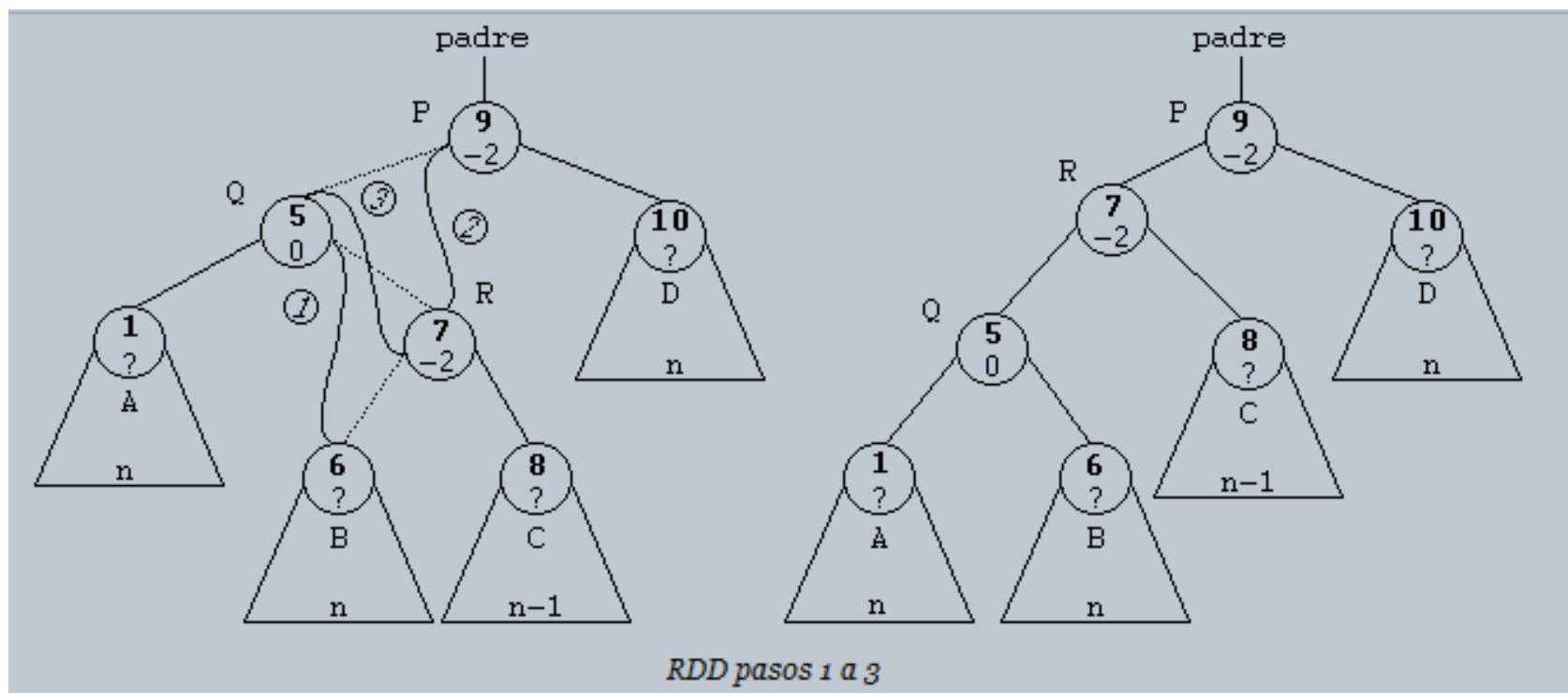
# Rotación doble la derecha (DD)

Con más detalle, procederemos del siguiente modo:

1. Pasamos el subárbol izquierdo del nodo R como subárbol derecho de Q. Esto mantiene el árbol como ABB, ya que todos los valores a la izquierda de R siguen estando a la derecha de Q.
2. Ahora, el nodo R pasa a tomar la posición del nodo Q, es decir, hacemos que la raíz del subárbol izquierdo de P sea el nodo R en lugar de Q.
3. El árbol Q pasa a ser el subárbol izquierdo del nodo R



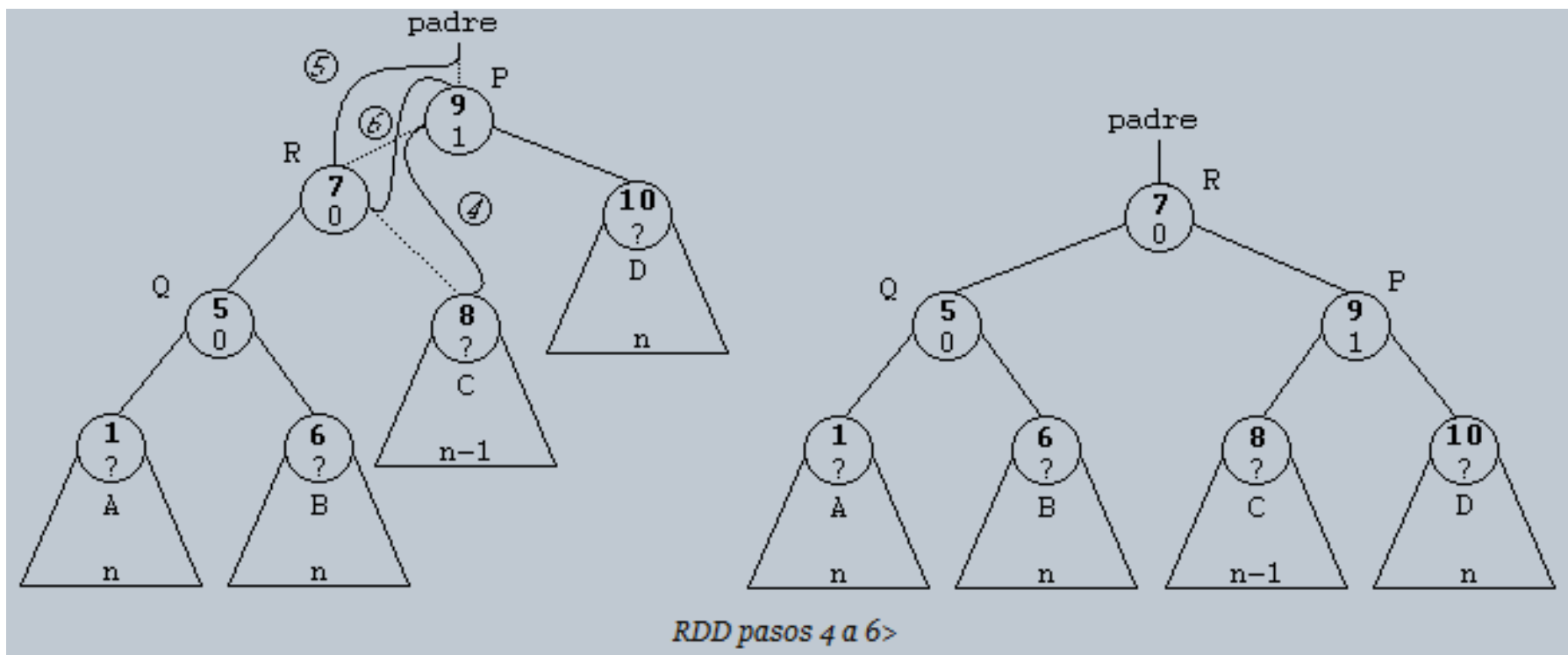
# Rotación doble la derecha (DD)



# Rotación doble la derecha (DD)

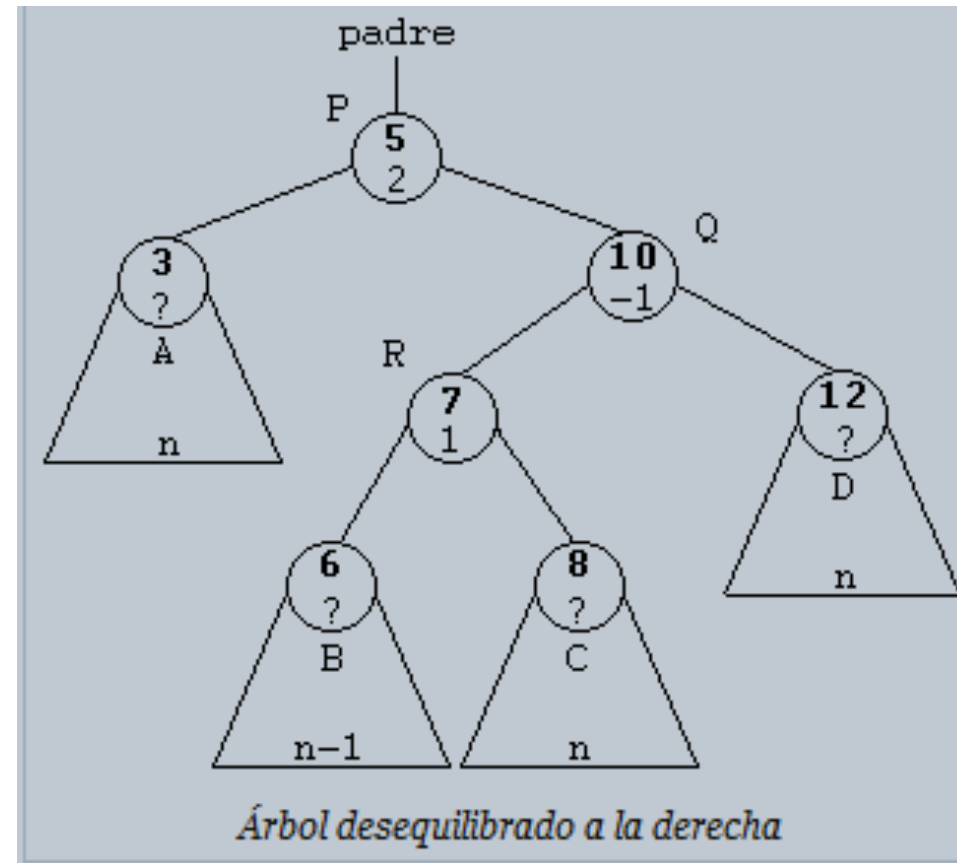
4. Pasamos el subárbol derecho del nodo R como subárbol izquierdo de P. Esto mantiene el árbol como ABB, ya que todos los valores a la derecha de R siguen estando a la izquierda de P.
5. Ahora, el nodo R pasa a tomar la posición del nodo P, es decir, hacemos que la entrada al árbol sea el nodo R, en lugar del nodo P. Como en los casos anteriores, previamente, P puede que fuese un árbol completo o un subárbol de otro nodo de menor altura.
6. El árbol P pasa a ser el subárbol derecho del nodo R.

# Rotación doble la derecha (DD)



# Rotación doble la izquierda (DI)

Esta 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 una FE de -1, es decir, que esté cargado a la izquierda. Se trata del caso simétrico del anterior



# Rotación doble la izquierda (DI)

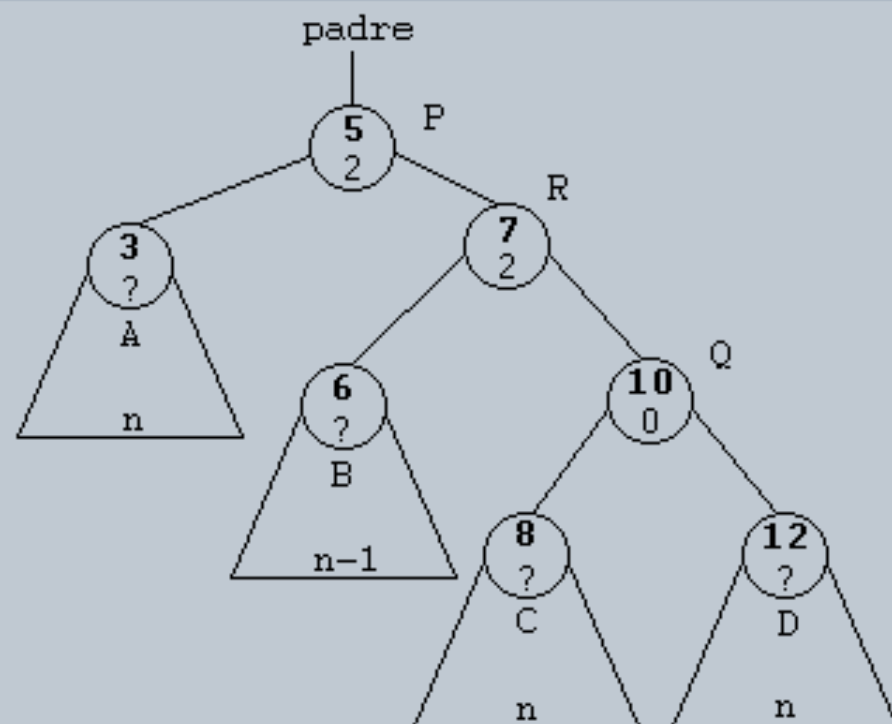
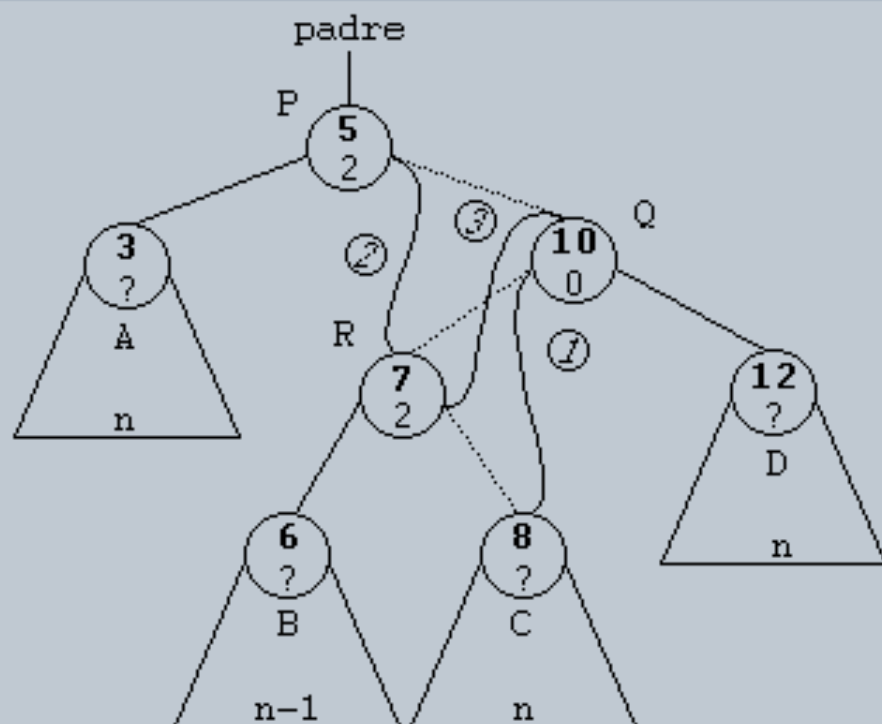
- ❑ Llamaremos P al nodo que muestra el desequilibrio, el que tiene una FE de 2. Llamaremos Q al nodo raíz del subárbol derecho de P, y R al nodo raíz del subárbol izquierdo de Q.
- ❑ Haremos una rotación simple de Q a la derecha.
- ❑ Después, haremos una rotación simple de P a la izquierda

# Rotación doble la izquierda (DI)

Con más detalle, procederemos del siguiente modo:

1. Pasamos el subárbol derecho del nodo R como subárbol izquierdo de Q. Esto mantiene el árbol como ABB, ya que todos los valores a la derecha de R siguen estando a la izquierda de Q.
2. Ahora, el nodo R pasa a tomar la posición del nodo Q, es decir, hacemos que la raíz del subárbol derecho de P sea el nodo R en lugar de Q.
3. El árbol Q pasa a ser el subárbol derecho del nodo R

# Rotación doble la izquierda (DI)



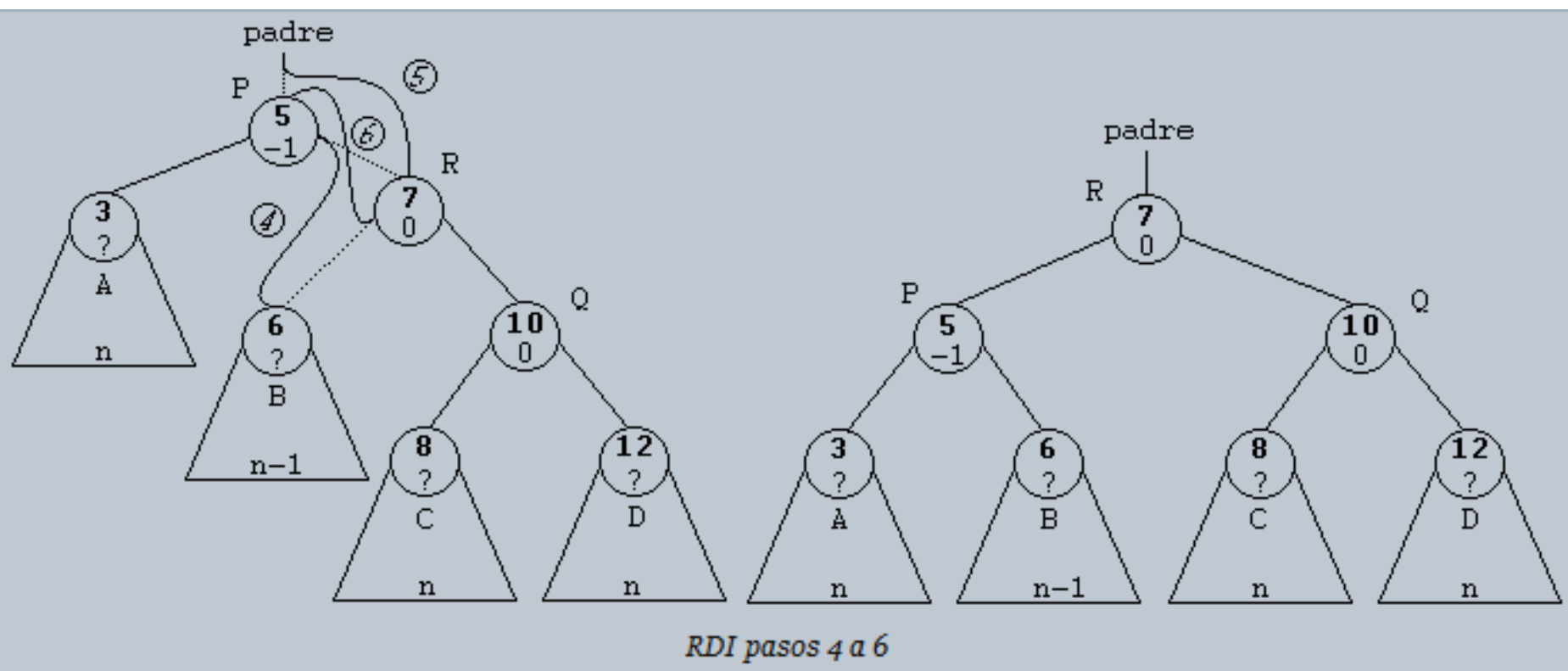
*RDI pasos 1 a 3*

# Rotación doble la izquierda (DI)

4. Pasamos el subárbol izquierdo del nodo R como subárbol derecho de P. Esto mantiene el árbol como ABB, ya que todos los valores a la izquierda de R siguen estando a la derecha de P.
5. Ahora, el nodo R pasa a tomar la posición del nodo P, es decir, hacemos que la entrada al árbol sea el nodo R, en lugar del nodo P. Como en los casos anteriores, previamente, P puede que fuese un árbol completo o un subárbol de otro nodo de menor altura.
6. El árbol P pasa a ser el subárbol izquierdo del nodo R.



# Rotación doble la izquierda (DI)



# Reequilibrados en árboles AVL

- Cada vez que insertemos o eliminemos un nodo en un árbol AVL pueden suceder dos cosas: que el árbol se mantenga como AVL o que pierda esta propiedad. En el segundo caso siempre estaremos en uno de los explicados anteriormente, y recuperaremos el estado AVL aplicando la rotación adecuada.

# Reequilibrados en árboles AVL

- Ya comentamos que necesitamos añadir un nuevo miembro a cada nodo del árbol para averiguar si el árbol sigue siendo AVL, el Factor de Equilibrio. Cada vez que insertemos o eliminemos un nodo deberemos recorrer el camino desde ese nodo hacia el nodo raíz actualizando los valores de FE de cada nodo. Cuando uno de esos valores sea 2 ó -2 aplicaremos la rotación correspondiente.

# Reequilibrados en árboles AVL

- ❑ Debido a que debemos ser capaces de recorrer el árbol en dirección a la raíz, añadiremos un nuevo puntero a cada nodo que apunte al nodo padre. Esto complicará algo las operaciones de inserción, borrado y rotación, pero facilita y agiliza mucho el cálculo del FE, y veremos que las complicaciones se compensan en gran parte por las facilidades obtenidas al disponer de este puntero.

# Reequilibrados en árboles AVL

- Cuando estemos actualizando los valores de FE no necesitamos calcular las alturas de las dos ramas de cada nodo, sabiendo el valor anterior de FE, y sabiendo en qué rama hemos añadido o eliminado el nodo, es fácil calcular el nuevo valor de FE. Si el nodo ha sido añadido en la rama derecha o eliminado en la izquierda, y ha habido un cambio de altura en la rama, se incrementa el valor de FE; si el nodo ha sido añadido en la rama izquierda o eliminado en la derecha, y ha habido un cambio de altura en la rama, se decrementa el valor de FE.

# Reequilibrados en árboles AVL

Los cambios de altura en una rama se producen sólo cuando el FE del nodo raíz de esa rama ha cambiado de 0 a 1 ó de 0 a -1. En caso contrario, cuando el FE cambia de 1 a 0 ó de -1 a 0, no se produce cambio de altura.

Si no hay cambio de altura, los valores de FE del resto de los nodos hasta el raíz no pueden cambiar, recordemos que el factor de equilibrio se define como la diferencia de altura entre las ramas derecha e izquierda de un nodo, la altura de la rama que no pertenece al camino no puede cambiar, puesto que sigue teniendo los mismos nodos que antes, de modo que si la altura de la rama que pertenece al camino no cambia, tampoco puede cambiar el valor de FE.

# Algoritmos

## De inserción de un nodo

- En general, la inserción de nodos en un árbol AVL es igual que en un árbol ABB, la diferencia es que en un árbol AVL, después de insertar el nodo debemos recorrer el árbol en sentido hacia la raíz, recalculando los valores de FE, hasta que se cumpla una de estas condiciones: que lleguemos a la raíz, que se encuentre un nodo con valor de FE de 2, ó -2, o que se llegue a un nodo cuyo FE no cambie o decrezca en valor absoluto, es decir, que cambie de 1 a 0 ó de -1 a 0.
- Podemos considerar que el algoritmo de inserción de nodos en árboles AVL es una ampliación del que vimos para árboles ABB

# Algoritmos

## De borrado de un nodo

- ❑ Lo mismo pasa cuando se eliminan nodos, el algoritmo es el mismo que en árboles ABB, pero después de eliminar el nodo debemos recorrer el camino hacia la raíz recalculando los valores de FE, y equilibrando el árbol si es necesario.



# Algoritmos

## De recalcular FE

Ya comentamos más atrás que para seguir el camino desde el nodo insertado o borrado hasta el nodo raíz tenemos dos alternativas:

- ❑ Guardar en una pila los punteros a los nodos por los que hemos pasado para llegar al nodo insertado o borrado, es decir, almacenar el camino.
- ❑ Añadir un nuevo puntero a cada nodo que apunte al padre del nodo actual. Esto nos permite recorrer el árbol en el sentido contrario al normal, es decir, en dirección a la raíz.

# Algoritmos

**Para calcular los nuevos valores de FE de los nodos del camino hay que tener en cuenta los siguientes hechos:**

- ❑ El valor de FE de un nodo insertado es cero, ya que siempre insertaremos nodos hoja.
- ❑ Si el nuevo valor de FE para cualquiera de los siguientes nodos del camino es cero, habremos terminado de actualizar los valores de FE, ya que la rama mantiene su altura, la inserción o borrado del nodo no puede influir en los valores de FE de los siguientes nodos del camino.
- ❑ Cuando se elimine un nodo pueden pasar dos cosas. Siempre eliminamos un nodo hoja, ya que cuando no lo es, lo intercambiamos con un nodo hoja antes de eliminarlo. Pero algunas veces, el nodo padre del nodo eliminado se convertirá a su vez en nodo hoja, y en ese caso no siempre hay que dar por terminada la actualización del FE del camino. Por lo tanto, cuando eliminemos un nodo, actualizaremos el valor de FE del nodo padre y continuaremos el camino, independientemente del valor de FE calculado.

❑ A la hora de actualizar el valor de FE de un nodo

# Algoritmos

## Para calcular los nuevos valores de FE de los nodos del camino hay que tener en cuenta los siguientes hechos:

- ❑ A la hora de actualizar el valor de FE de un nodo, tenemos que distinguir cuando el equilibrado sea consecuencia de una inserción o lo sea de una eliminación. Incrementaremos el valor de FE del nodo si la inserción fue en la rama derecha o si la eliminación fue en la rama izquierda, decrementaremos si la inserción fue en la izquierda o la eliminación en la derecha.
- ❑ Si el valor de FE es -2, haremos una rotación doble a la derecha si el valor de FE del nodo izquierdo es 1, y simple si es 1 ó 0.
- ❑ Si el valor de FE es 2, haremos una rotación doble a la izquierda si el valor de FE del nodo izquierdo es -1, y simple si es -1 ó 0.
- ❑ En cualquiera de los dos casos, podremos dar por terminado el recorrido del camino, ya que la altura del árbol cuya raíz es un nodo rotado no cambia.
- ❑ En cualquier otro caso, seguiremos actualizando hasta llegar al nodo raíz.

# Algoritmos

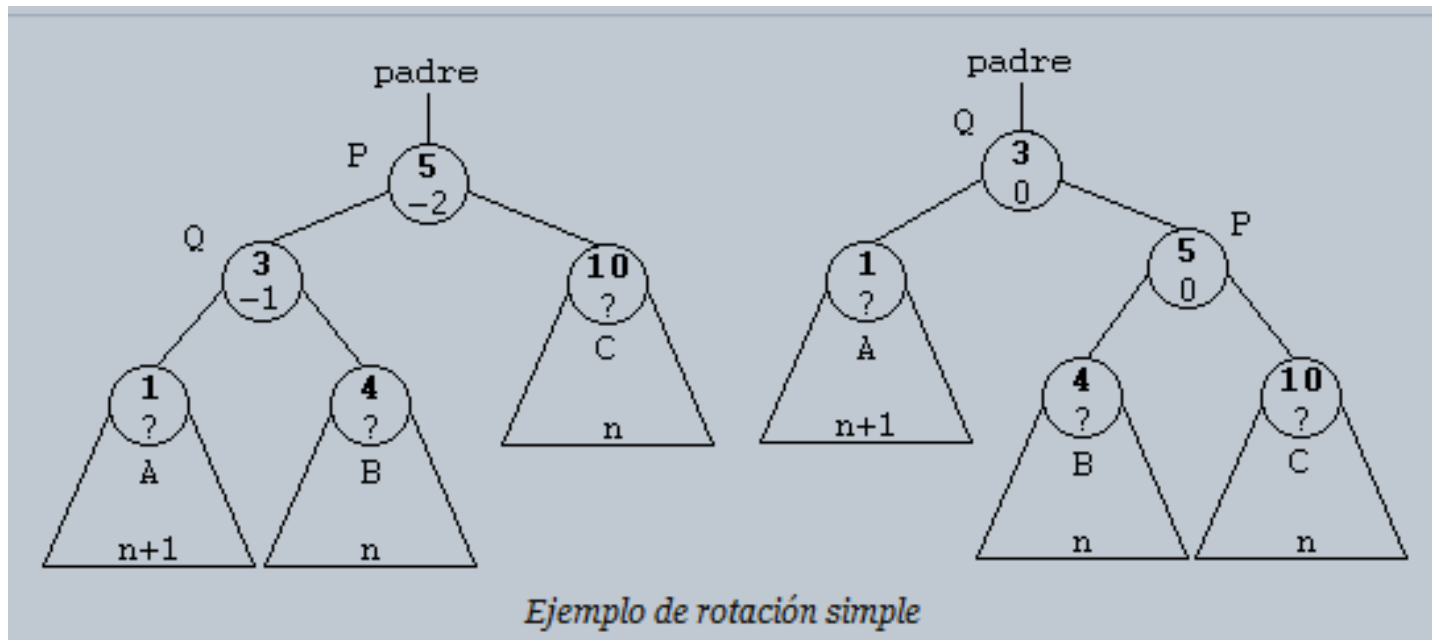
## Rotación simple

- A la hora de implementar los algoritmos que hemos visto para rotaciones simples tenemos dos opciones: seguir literalmente los pasos de los gráficos, o tomar un atajo, y hacerlo mediante asignaciones. Nosotros lo haremos del segundo modo, ya que resulta mucho más rápido y sencillo.
- Primero haremos las reasignaciones de punteros, de modo que el árbol resultante responda a la estructura después de la rotación. Después actualizaremos los punteros al nodo padre para los nodos que han cambiado de posición. Por último actualizaremos los valores de FE de esos mismos nodos.
- Para la primera fase usaremos punteros auxiliares a nodo, que en el caso de rotación a la derecha necesitamos un puntero P al nodo con FE igual a -2. Ese será el parámetro de entrada, otro puntero al nodo izquierdo de P: Q. Y tres punteros más a los árboles A, B y C.

# Algoritmos

## Rotación simple

- En realidad, si nos fijamos en los gráficos, los punteros a A y C no son necesarios, ya que ambos conservan sus posiciones, A sigue siendo el subárbol izquierdo de Q y C el subárbol derecho de P.



# Algoritmos

## Rotación simple

Usaremos otro puntero más: Padre, que apunte al padre de P. Disponiendo de los punteros Padre, P, Q y B, realizar la rotación es muy sencillo

```
if(Padre)
    if(Padre->derecho == P) Padre->derecho = Q;
    else Padre->izquierdo = Q;
else raíz = Q;

// Reconstruir árbol:
P->izquierdo = B;
Q->derecho = P;
```

Hay que tener en cuenta que P puede ser la raíz de un subárbol derecho o izquierdo de otro nodo, o incluso la raíz del árbol completo. Por eso comprobamos si P tiene padre, y si lo tiene, cual de sus ramas apunta a P, cuando lo sabemos, hacemos que esa rama apunte a Q. Si Padre es NULL, entonces P era la raíz del árbol, así que hacemos que la nueva raíz sea Q.

Sólo nos queda trasladar el subárbol B a la rama izquierda de P, y Q a la rama derecha de P

# Algoritmos

## Rotación simple

La segunda fase consiste en actualizar los punteros padre de los nodos que hemos cambiado de posición: P, B y Q

```
P->padre = Q;  
if(B) B->padre = P;  
Q->padre = Padre;
```

El padre de P es ahora Q, el de Q es Padre, y el de B, si existe es P.

La tercera fase consiste en ajustar los valores de FE de los nodos para los que puede haber cambiado.

Esto es muy sencillo, después de una rotación simple, los únicos valores de FE que cambian son los de P y Q, y ambos valen 0.

# Algoritmos

## Rotación doble

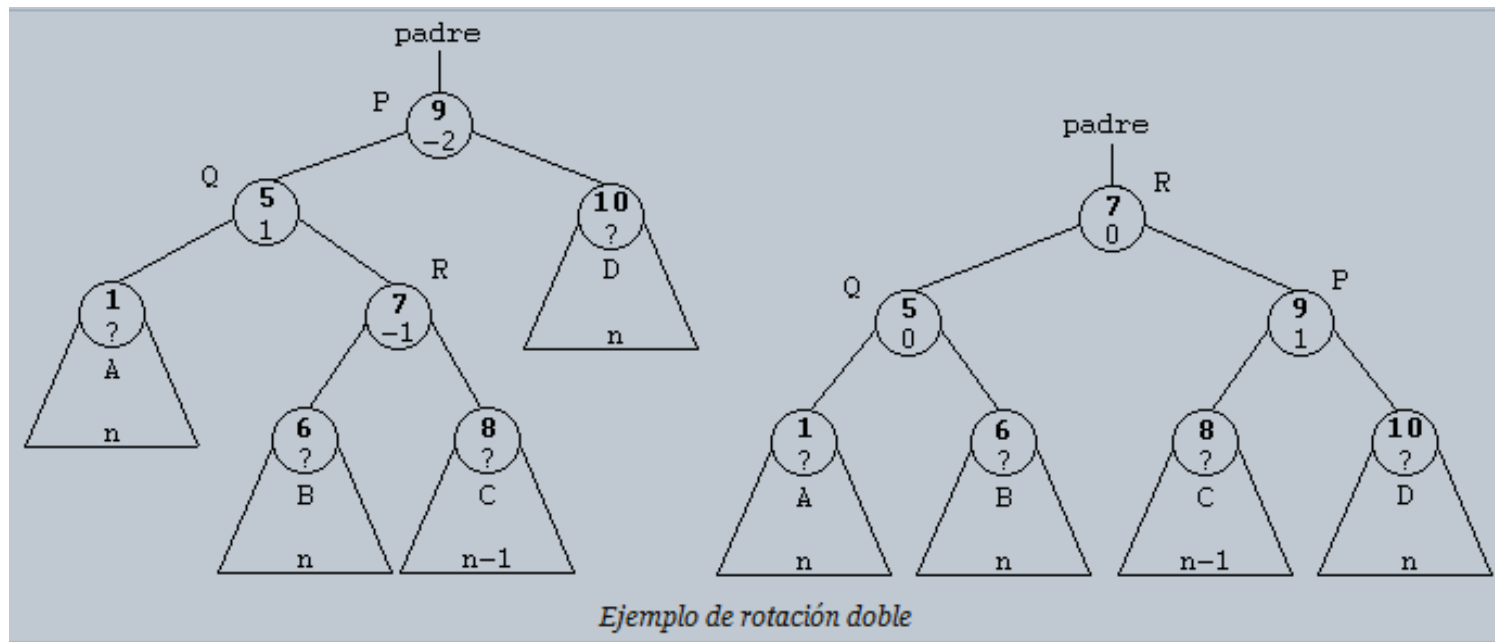
- ❑ Para implementar las rotaciones dobles trabajaremos de forma análoga.
- ❑ Primero haremos las reasignaciones de punteros, de modo que el árbol resultante responda a la estructura después de la rotación. Después actualizaremos los punteros al nodo padre para los nodos que han cambiado de posición. Por último actualizaremos los valores de FE de esos mismos nodos.
- ❑ Para la primera fase usaremos punteros auxiliares a nodo, que en el caso de rotación a la derecha necesitamos un puntero P al nodo con FE igual a -2. Ese será el parámetro de entrada, otro puntero al nodo izquierdo de P: Q. Un tercero al nodo derecho de Q: R. Y cuatro punteros más a los árboles A, B, C y D.



# Algoritmos

## Rotación doble

- En realidad, si nos fijamos en los gráficos, los punteros a A y D no son necesarios, ya que ambos conservan sus posiciones, A sigue siendo el subárbol izquierdo de Q y D el subárbol derecho de P.
- También en este caso usaremos otro puntero más: Padre, que apunte al padre de P. Disponiendo de los punteros Padre, P, Q, R, B y C



# Algoritmos

## Rotación doble

- Ahora también hay que tener en cuenta que P puede ser la raíz de un subárbol derecho o izquierdo de otro nodo, o incluso la raíz del árbol completo. Por eso comprobamos si P tiene padre, y si lo tiene, cual de sus ramas apunta a P, cuando lo sabemos, hacemos que esa rama apunte a R. Si Padre es NULL, entonces P era la raíz del árbol, así que hacemos que la nueva raíz sea R.
- Sólo nos queda trasladar el subárbol B a la rama derecha de Q, C a la rama izquierda de P, Q a la rama izquierda de R y P a la rama derecha de R.
- La segunda fase consiste en actualizar los punteros padre de los nodos que hemos cambiado de posición: P, Q, R, B y C

# Algoritmos

## Rotación doble

```
R->padre = Padre;
P->padre = Q->padre = R;
if(B) B->padre = Q;
if(C) C->padre = P;
```

- ❑ El padre de R es ahora Padre, el de P y Q es R, y el de B, si existe es Q, y el de C, si existe, es P.
- ❑ La tercera fase consiste en ajustar los valores de FE de los nodos para los que puede haber cambiado.
- ❑ En las rotaciones dobles esto se complica un poco ya que puede suceder que el valor de FE de R antes de la rotación sea -1, 0 o 1. En cada caso, los valores de FE de P y Q después de la rotación serán diferentes

# Algoritmos

## Rotación doble

```
// Ajustar valores de FE:
switch(R->FE) {
    case -1: Q->FE = 0; P->FE = 1; break;
    case 0:  Q->FE = 0; P->FE = 0; break;
    case 1:  Q->FE = -1; P->FE = 0; break;
}
R->FE = 0;
```

- Si la altura de B es  $n-1$  y la de C es  $n$ , el valor de FE de R es 1. Después de la rotación, la rama B pasa a ser el subárbol derecho de Q, por lo tanto, la FE de Q, dado que la altura de su rama izquierda es  $n$ , será 0. La rama C pasa a ser el subárbol izquierdo de P, y dado que la altura de la rama derecha es  $n$ , la FE de P será -1.

# Algoritmos

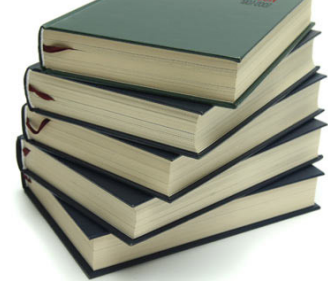
## Rotación doble

- Si la altura de B es  $n$  y la de C es  $n-1$ , el valor de FE de R es  $-1$ . Después de la rotación, la rama B pasa a ser el subárbol derecho de Q, por lo tanto, la FE de Q, dado que la altura de su rama izquierda es  $n$ , será  $0$ . La rama C pasa a ser el subárbol izquierdo de P, y dado que la altura de la rama derecha es  $n$ , la FE de P será  $0$ .
- Por último, si la altura de B y C es  $n$ , el valor de FE de R es  $0$ . Después de la rotación, la rama B pasa a ser el subárbol derecho de Q, por lo tanto, la FE de Q, dado que la altura de su rama izquierda es  $n$ , será  $0$ . La rama C pasa a ser el subárbol izquierdo de P, y dado que la altura de la rama derecha es  $n$ , la FE de P será  $0$ .



```
int main()  
{  
    return 0;  
}
```





# Bibliografía

---

<http://c.conclase.net/edd/index.php?cap=007#inicio>

<http://es.wikipedia.org/wiki/>

[%C3%81rbol binario de b%C3%BAsqueda](#)

Programación en C++, 1ª Edición, Luis Joyanes  
Aguilar, capítulo 21, Pág. 431-434