Árbol binario:

- cada nodo x tiene a lo más dos hijos, uno izquierdo y otro derecho,
- … que, si están, son raíces de los subárboles izquierdo y derecho de x

Árbol binario de búsqueda (ABB):

- la clave almacenada en un nodo x es mayor (o igual) que cualquiera de las claves almacenadas en el subárbol izquierdo de x
- … y menor (o igual) que cualquiera de las claves almacenadas en el subárbol derecho de x

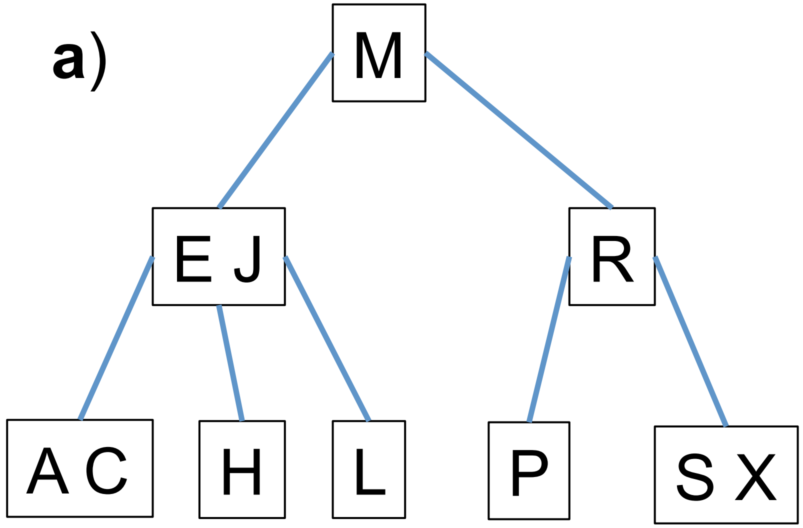
ABB balanceado:

- cumple una propiedad adicional de balance
- p.ej., en un árbol AVL, para cualquier nodo del árbol, las alturas de los subárboles izquierdo y derecho pueden diferir a lo más en 1

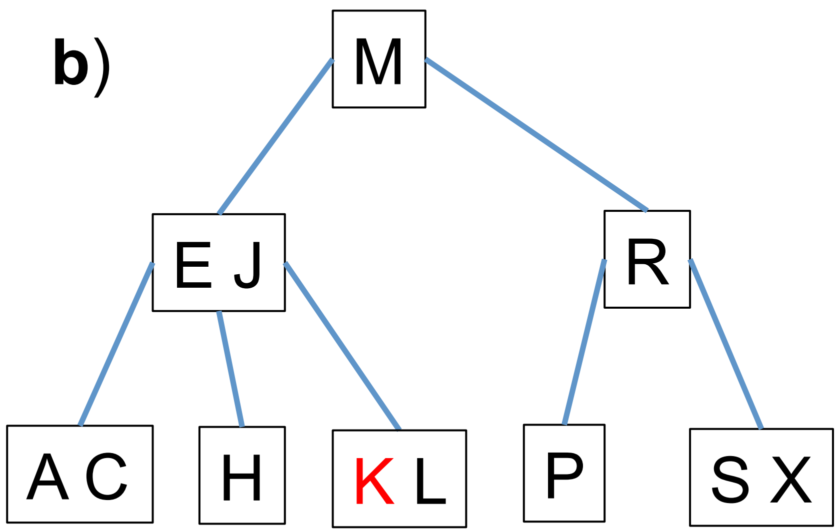
Un **árbol 2-3** es un árbol de búsqueda en que cada nodo puede ser

… un **nodo-2**, con una clave y dos hijos (izquierdo y derecho), tal que, o ambos son nulos (y el nodo es una hoja) o ambos son raíces de (sub)árboles 2-3

… o un **nodo-3**, con dos claves distintas y tres hijos (izquierdo, del medio y derecho), tal que, o los tres son nulos (y el nodo es una hoja) o los tres son raíces de (sub)árboles 2-3 a)



b)



La relación entre la o las claves del nodo y las claves de sus sub-árboles es una generalización de la propiedad de ABB:

en el caso de un nodo-3, con claves k_1 y k_2, $k_1 < k_2$,

… cualquier clave k_{izq} en el subárbol izquierdo cumple $k_{izq} < k_1$

… cualquier clave k_{der} en el subárbol derecho cumple $k_{der} > k_2$

… y cualquier clave k_{med} en el subárbol del medio cumple $k_1 < k_{med} < k_2$

En la diap. #56:

a) árbol 2-3 con 8 nodos y 11 claves

b) el árbol después de que insertamos la clave *K* :

- como *K* va a parar a un nodo-2, simplemente convertimos ese nodo en un nodo-3 y la ponemos ahí en orden

En la próxima diap.:

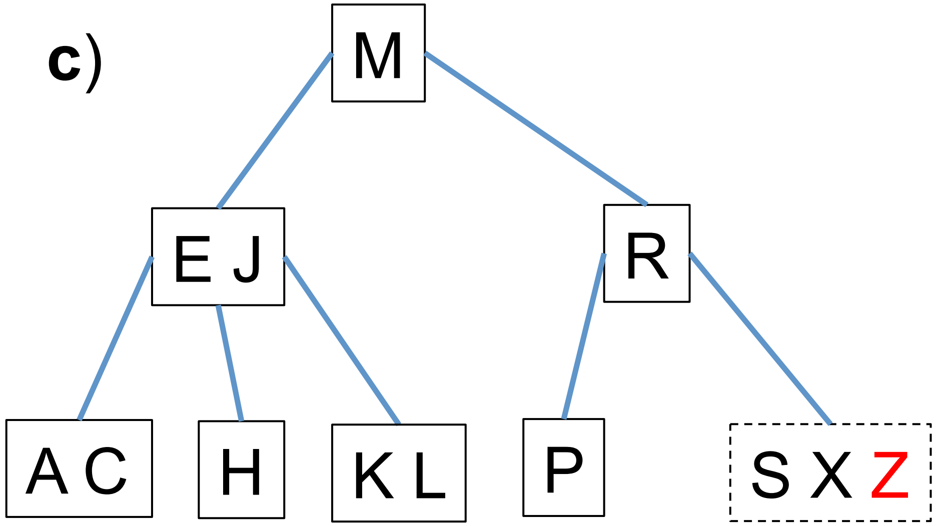
c) situación intermedia al insertar la clave *Z* :

- *Z* va a parar a un nodo-3, que temporalmente convertimos en un nodo-4 para poner *Z* ahí en orden

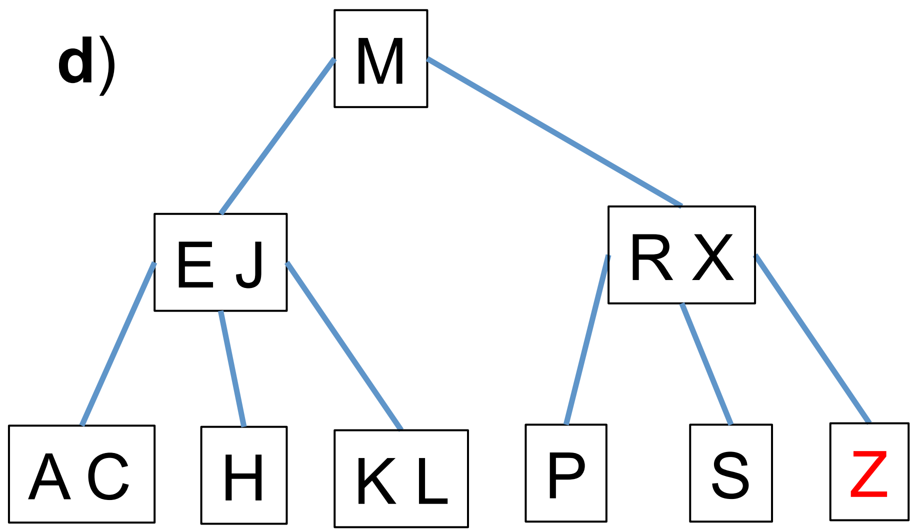
d) el árbol después de que terminamos de insertar *Z* :

- la clave del medio del nodo-4 temporal, *X*, la pasamos al padre, que, al ser un nodo-2, se convierte en un nodo-3, con claves *R*,*X*
- las claves extremas del nodo-4, *S* y *Z*, forman cada una un nodo-2, que junto al nodo con clave *P* son ahora los tres hijos del nuevo nodo-3

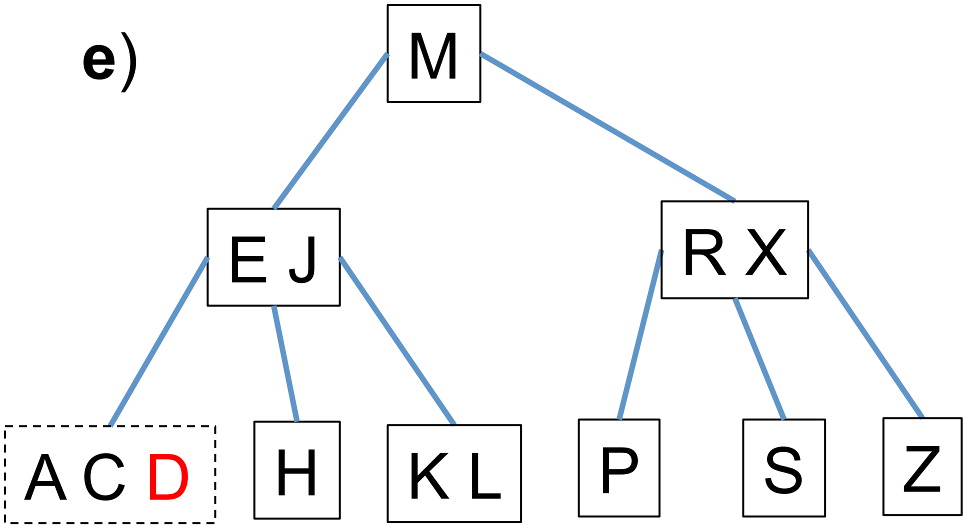
c)



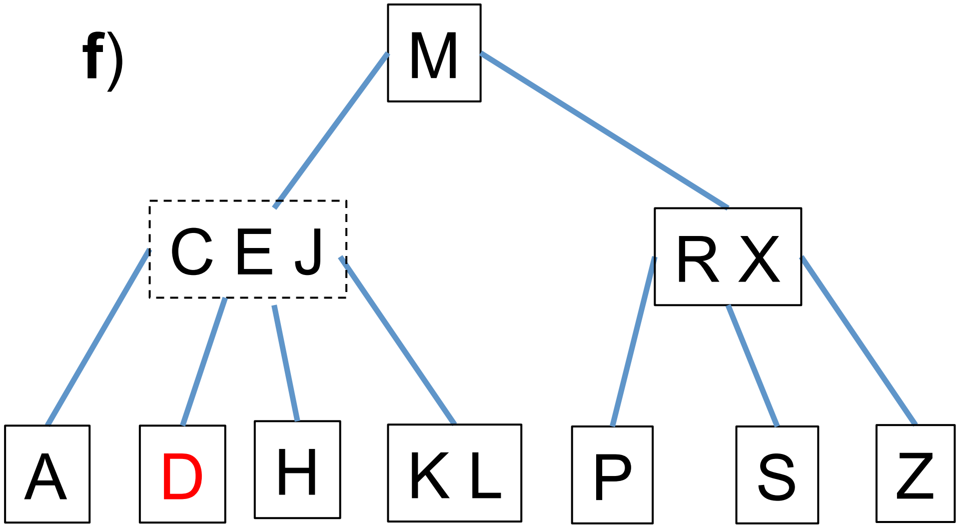
d)



e)



f)



En la diap. anterior:

e) situación intermedia al insertar la clave *D* :

- *D* va a parar a un nodo-3, que temporalmente convertimos en un nodo-4 para poner *D* ahí en orden

f) situación intermedia (y parcial) al insertar la clave *D* :

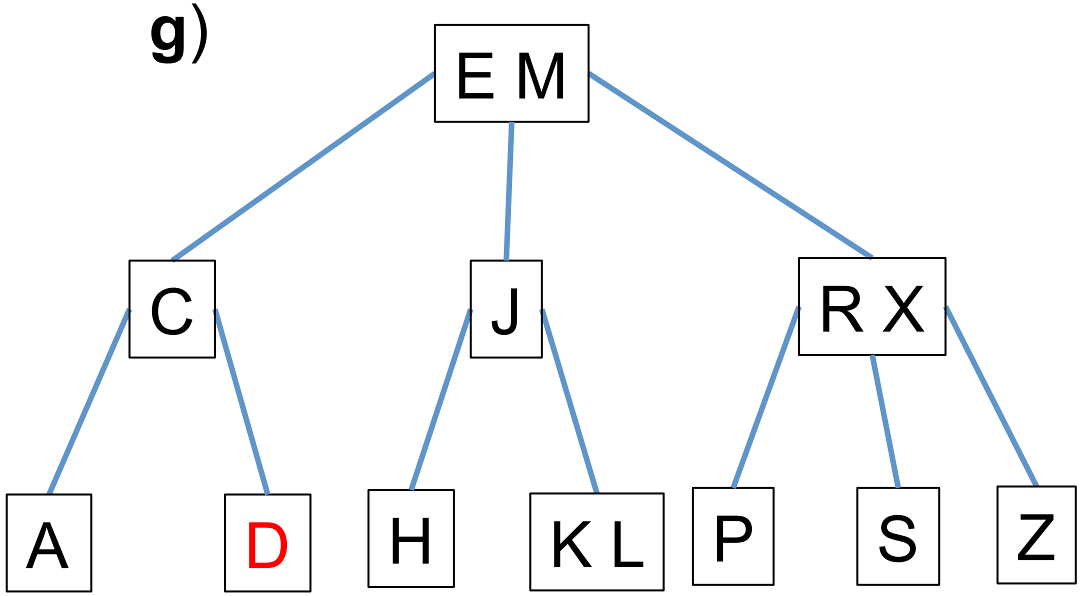
- la clave del medio del nodo-4 temporal, *C*, la pasamos al padre, que, al ser un nodo-3, se convierte en un nodo-4 temporal, con claves *C*,*E*,*J*
- las claves extremas, *A* y *D*, forman cada una un nodo-2, que junto a los nodos con claves *H* y *K*,*L* son ahora los cuatro hijos del nuevo nodo-4

Finalmente, en la próxima diap.:

g) el árbol después de que terminamos de insertar *D* :

- la clave del medio del nodo-4, *E*, pasa al padre (la raíz del árbol), que, al ser un nodo-2, se convierte en un nodo-3, con claves *E* y *M*
- las claves extremas, *C* y *J*, forman cada una un nodo-2, cada uno con dos hijos; ambos nodos son hijos de la nueva raíz

g)



Tomemos un árbol 2-3, tal como **d**) en la diap. #60

… y hagamos lo siguiente con cada nodo-3, p.ej., el nodo con claves *E*,*J*:

i) separémoslo en dos nodos, uno con cada clave

ii) hagamos que el nodo con la clave mayor, *J*, apunte al otro nodo, pero horizontalmente, hacia la izquierda

iii) … y hagamos que este otro nodo —con la clave menor, *E*— sea el padre de los hijos izquierdo y del medio del nodo-3 original

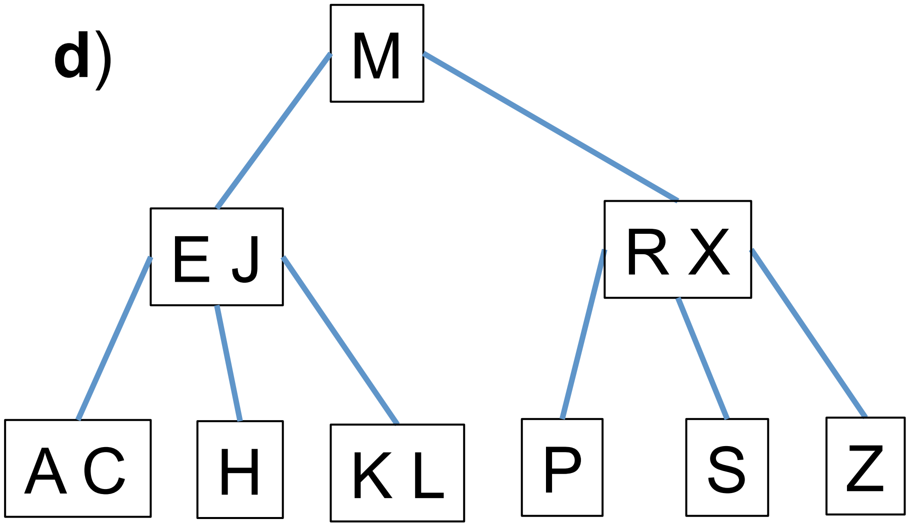
Obtenemos la estructura de la fig. **h**), que básicamente es un árbol binario:

si los nuevos nodos apuntados desde su derecha, como conse-cuencia de las separaciones de los nodos-3, *los pintamos rojos*

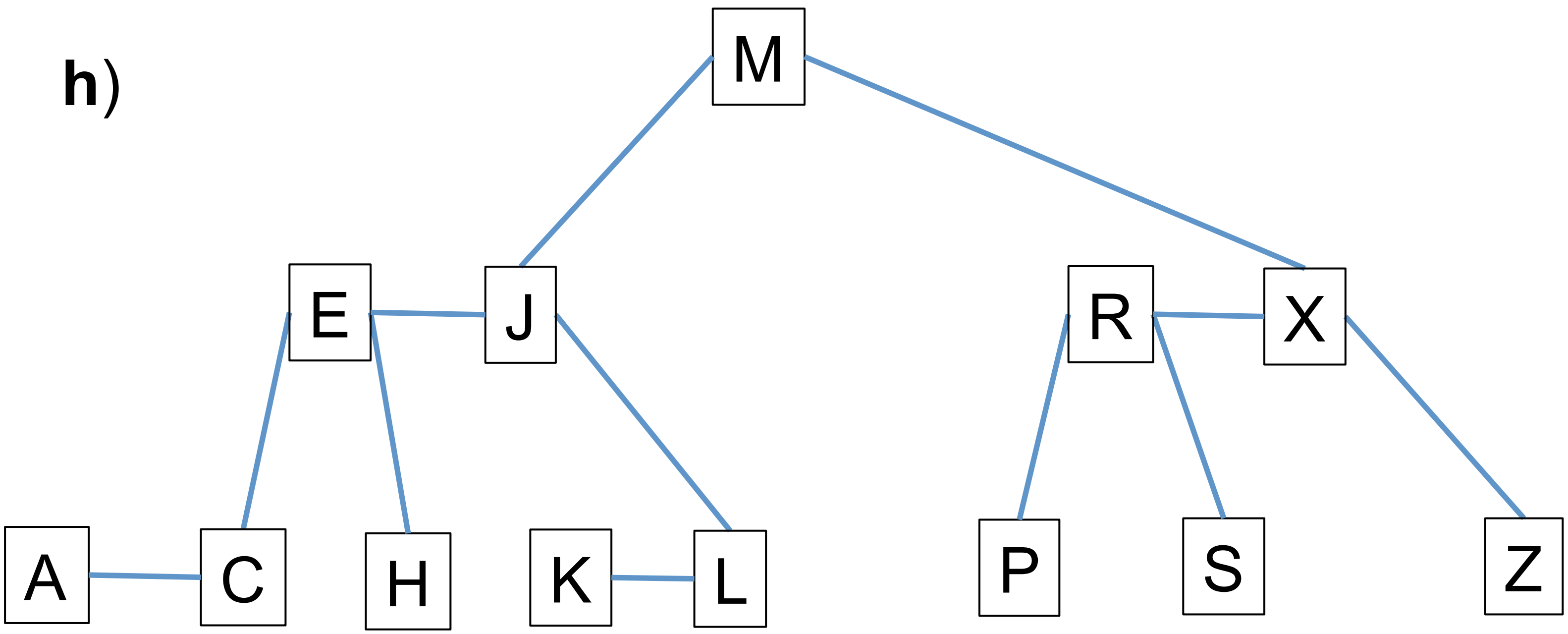
… y todo los otros nodos *los pintamos negros*

… tenemos un **árbol rojo-negro**, como en la fig. **i**)

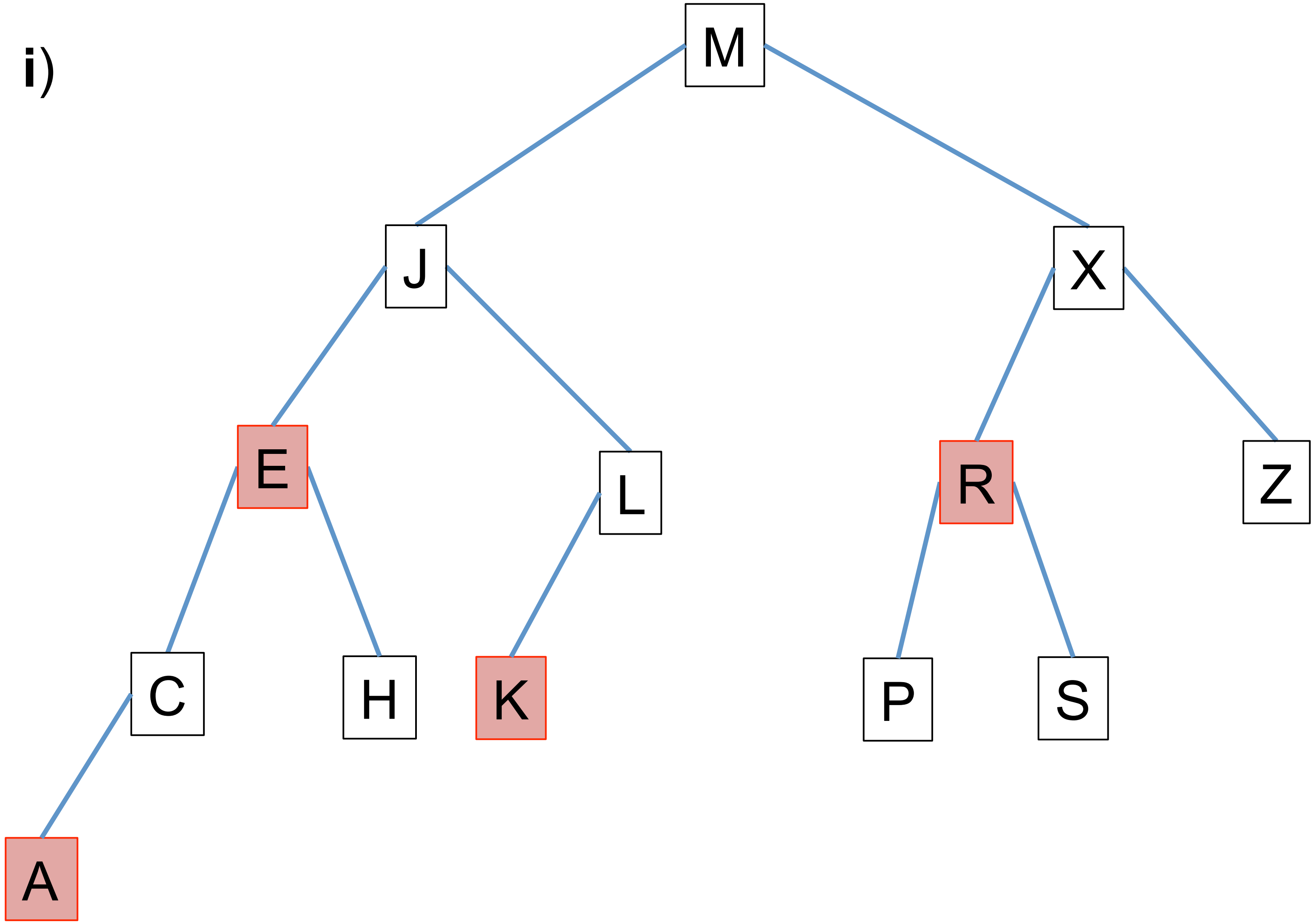
d)



h)



i)



Un **árbol rojo-negro** es un ABB que cumple cuatro *propiedades de árbol rojo-negro* (adicionales)

Propiedades de balance rojo-negro:

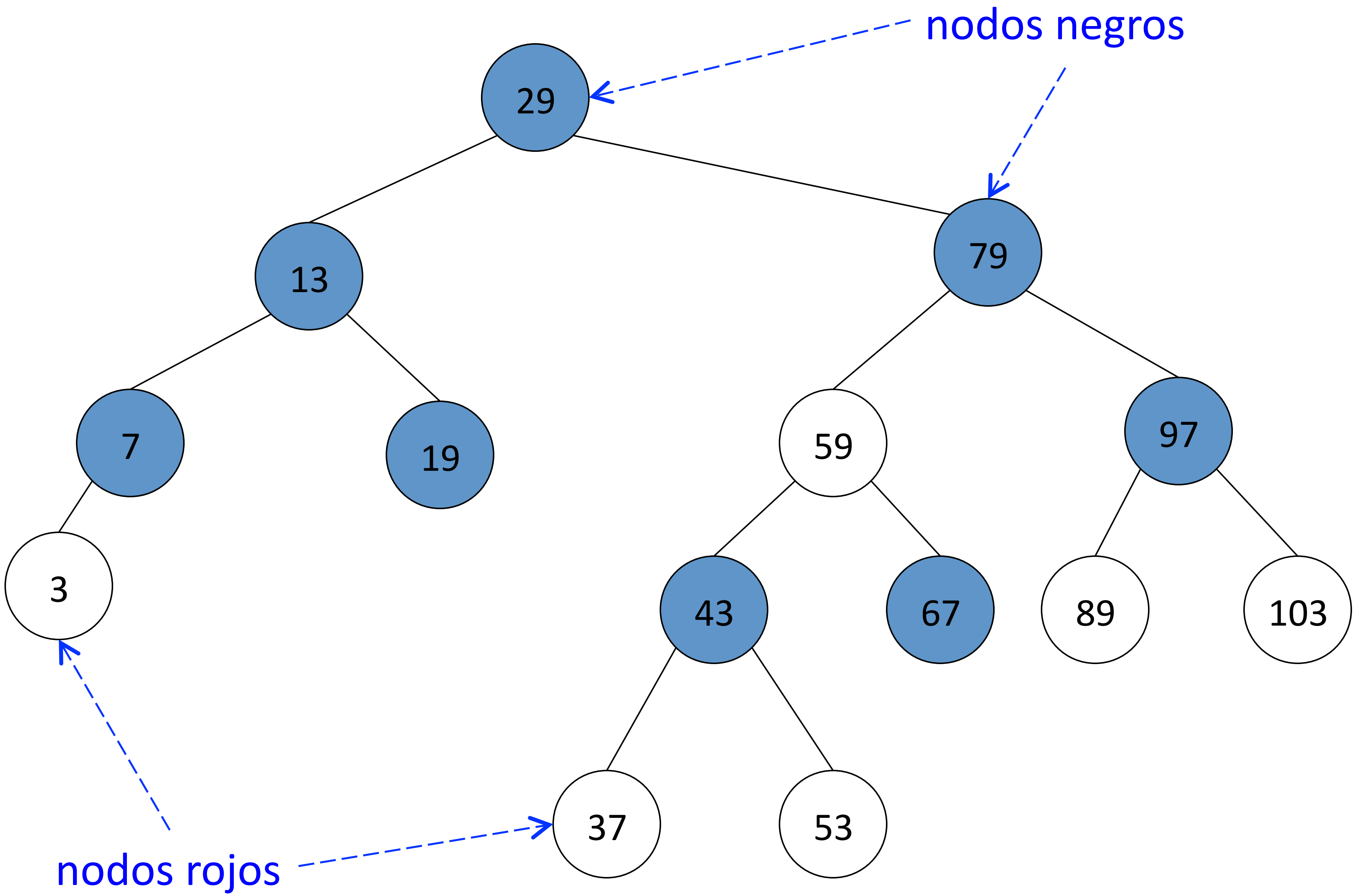
1) *Todo nodo es ya sea rojo o negro*

2) *La raíz es negra*

3) *Si un nodo es rojo, entonces sus hijos son negros*

4) *Para cada nodo, todas las rutas desde el nodo a las hojas descendientes contienen el mismo número de nodos negros*

Las propiedades 3 y 4, que podrían parecer arbitrarias, pueden deducirse de la forma en cómo se obtiene un árbol rojo-negro a partir de un árbol 2-3 (diapositivas anteriores)

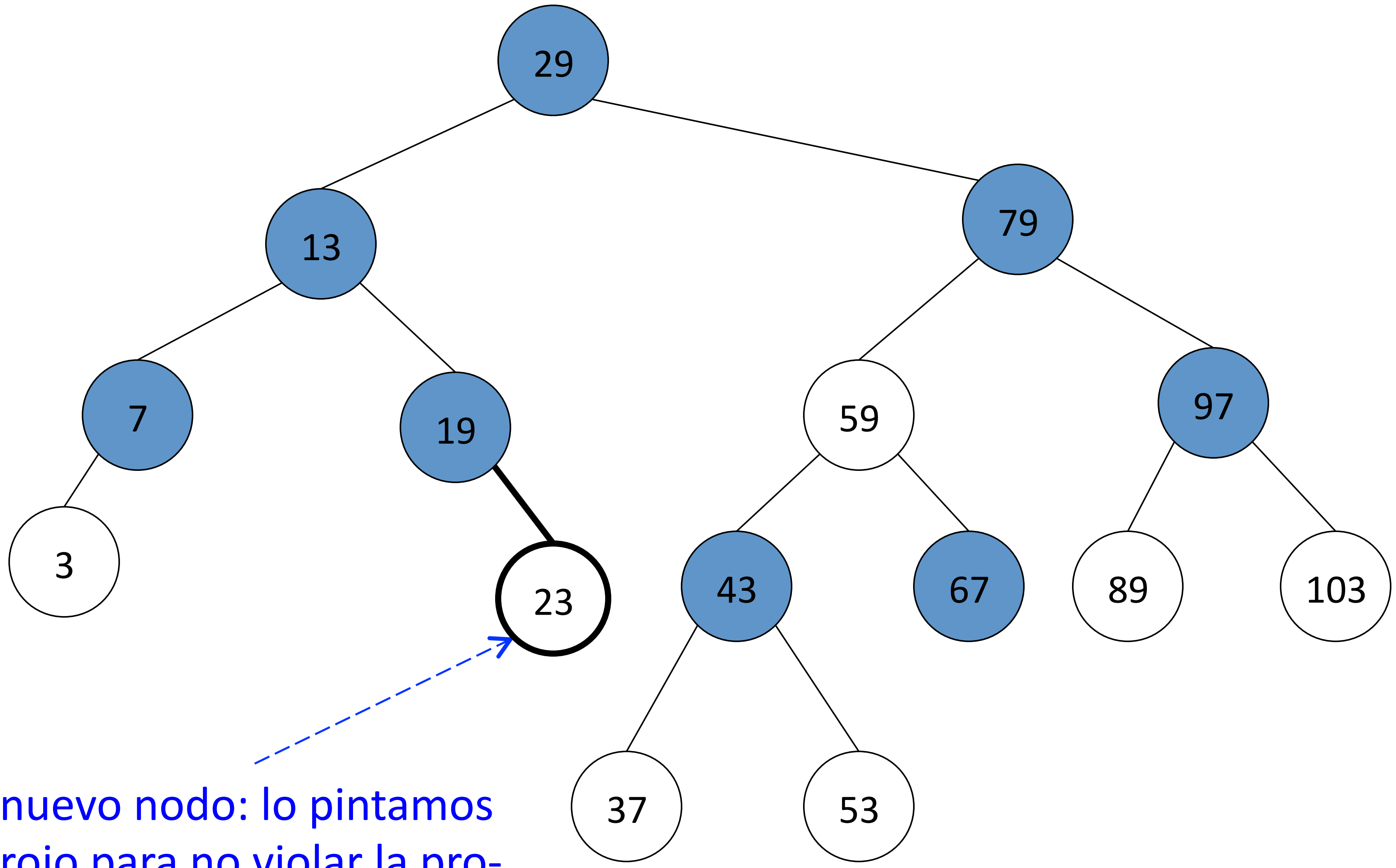

nodos negros
29
13
79
7
19
59
97
3
43
67
89
103
37
53
nodos rojos

Una inserción procede inicialmente igual que en un ABB (no necesariamente balanceado):

- siempre insertamos un nuevo nodo como una hoja
- … y lo pintamos rojo (si lo pintamos negro, violamos la propiedad 4)

Si el padre del nodo recién insertado es negro, terminamos:

p.ej., si en el árbol de la diap. anterior insertamos la clave 23 —ver próxima diap.



nuevo nodo: lo pintamos rojo para no violar la propiedad 4; como su padre es negro, terminamos

En cambio, si el padre es rojo, entonces violamos la propiedad 3:

- tenemos que ajustar el árbol para asegurar la propiedad 3 (los nodos rojos tienen hijos negros)
- … sin violar la propiedad 4 (acerca del número de nodos negros en la rutas hasta las hojas)

Cambiamos colores y hacemos rotaciones

Supongamos que el hermano del padre (del nodo recién ingresado) es negro (si no existe, lo suponemos negro):

p.ej., insertamos las claves 2 o 5 en el árbol de la diap. #70

Sean

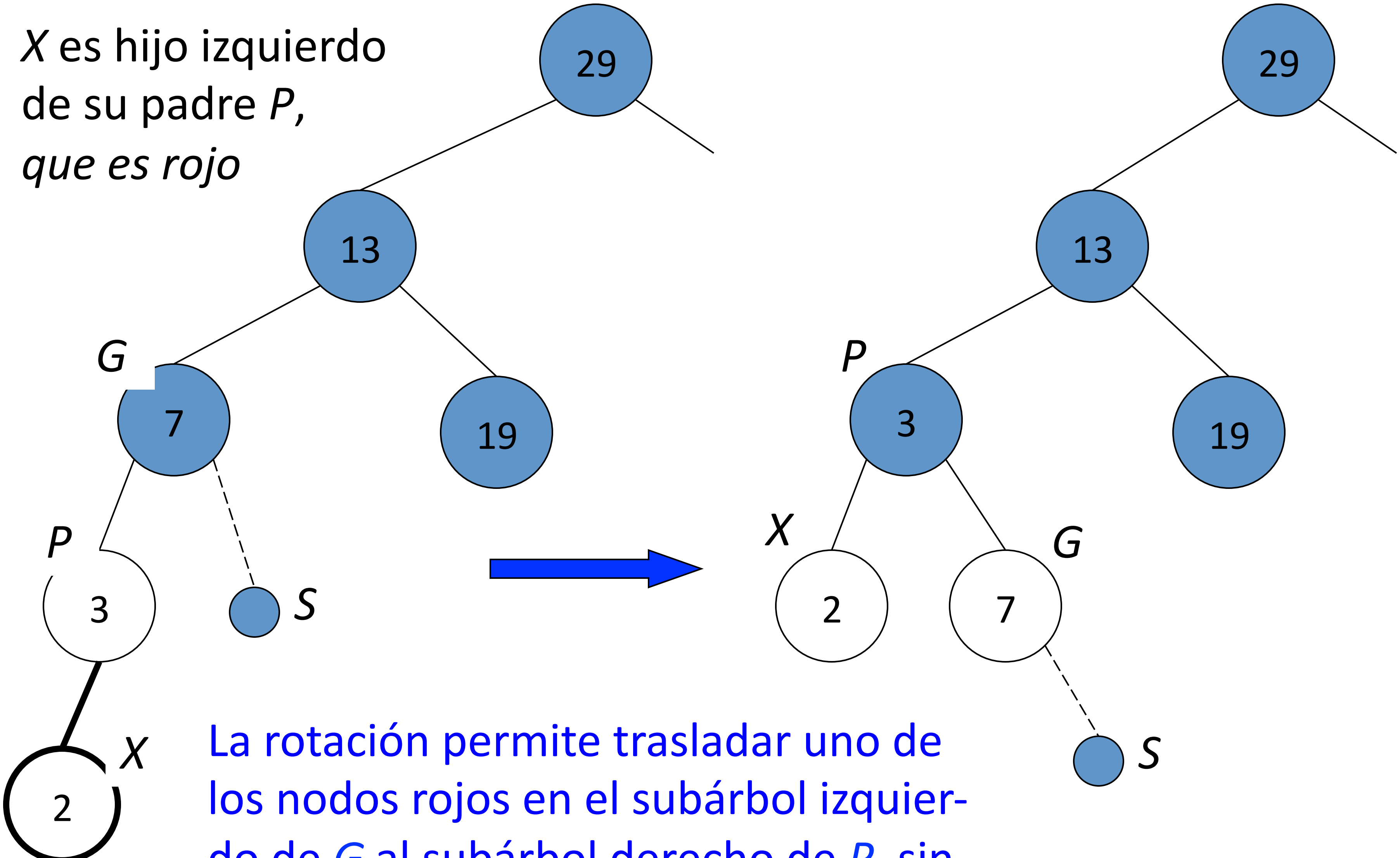
X la nueva hoja, roja

P su padre, rojo

S el hermano del padre, negro (*S* no existe cuando recién se produce la inserción, pero puede existir cuando, resolviendo otros casos, lo convertimos en éste)

G el abuelo, negro (de lo contrario, se violaría la propiedad 3)

X es hijo izquierdo
de su padre *P*,
que es rojo



La rotación permite trasladar uno de los nodos rojos en el subárbol izquierdo de *G* al subárbol derecho de *P*, sin afectar el número de nodos negros en las rutas desde la raíz a las hojas.

Si *X* es hijo izquierdo de *P* (p.ej., insertamos 2, como en la diap. anterior),

… entonces rotamos a la derecha en torno a *G*–*P*

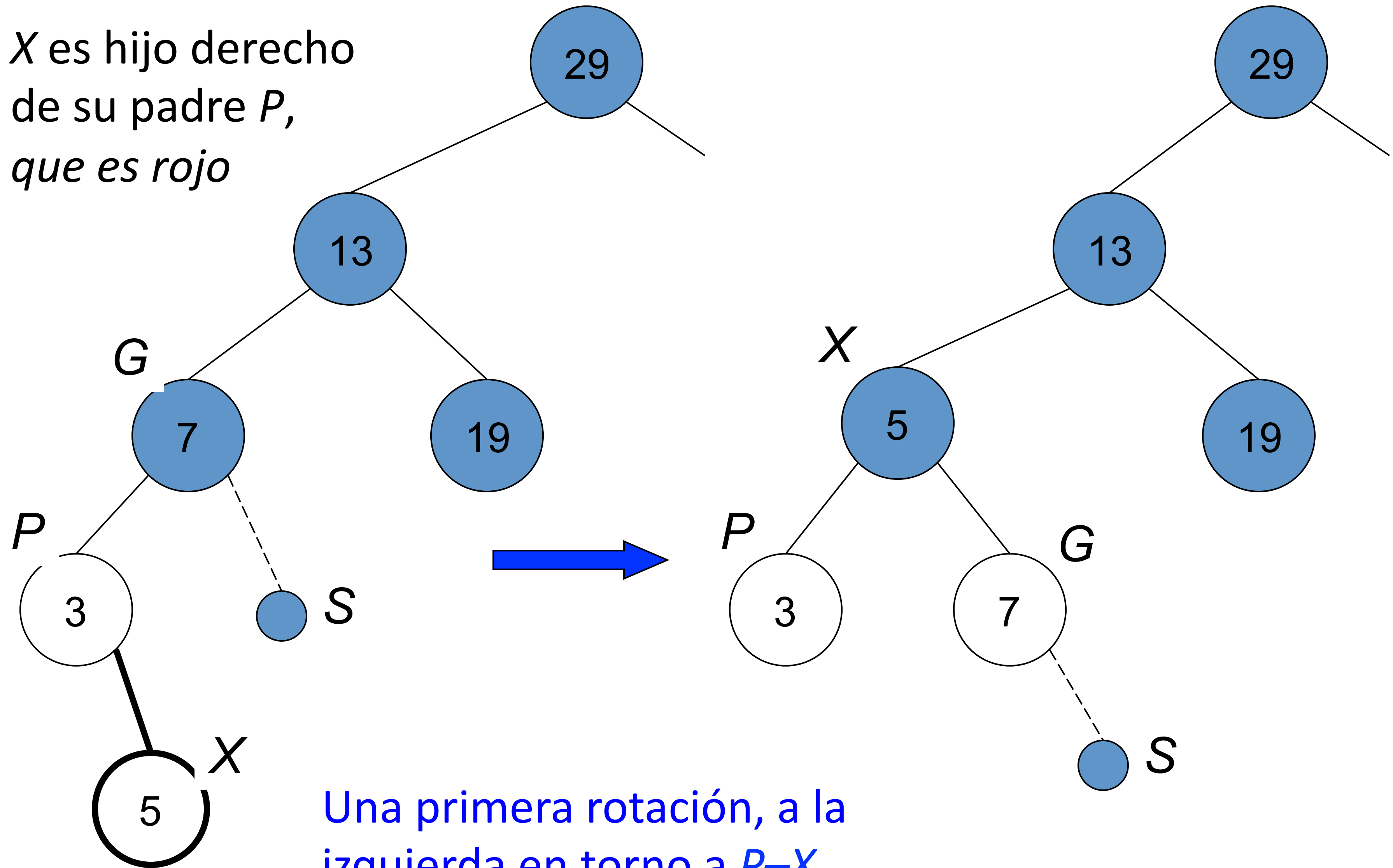
Si *X* es hijo derecho de *P* (p.ej., insertamos 5, como en la próxima diap.),

… entonces hacemos una rotación doble:

- primero, a la izquierda en torno a *P*–*X*, con-virtiendo este caso en el caso en el anterior

- … luego, a la derecha en torno a *G*–*X*, aplicando la misma solución que en el caso anterior

X es hijo derecho
de su padre *P*,
que es rojo



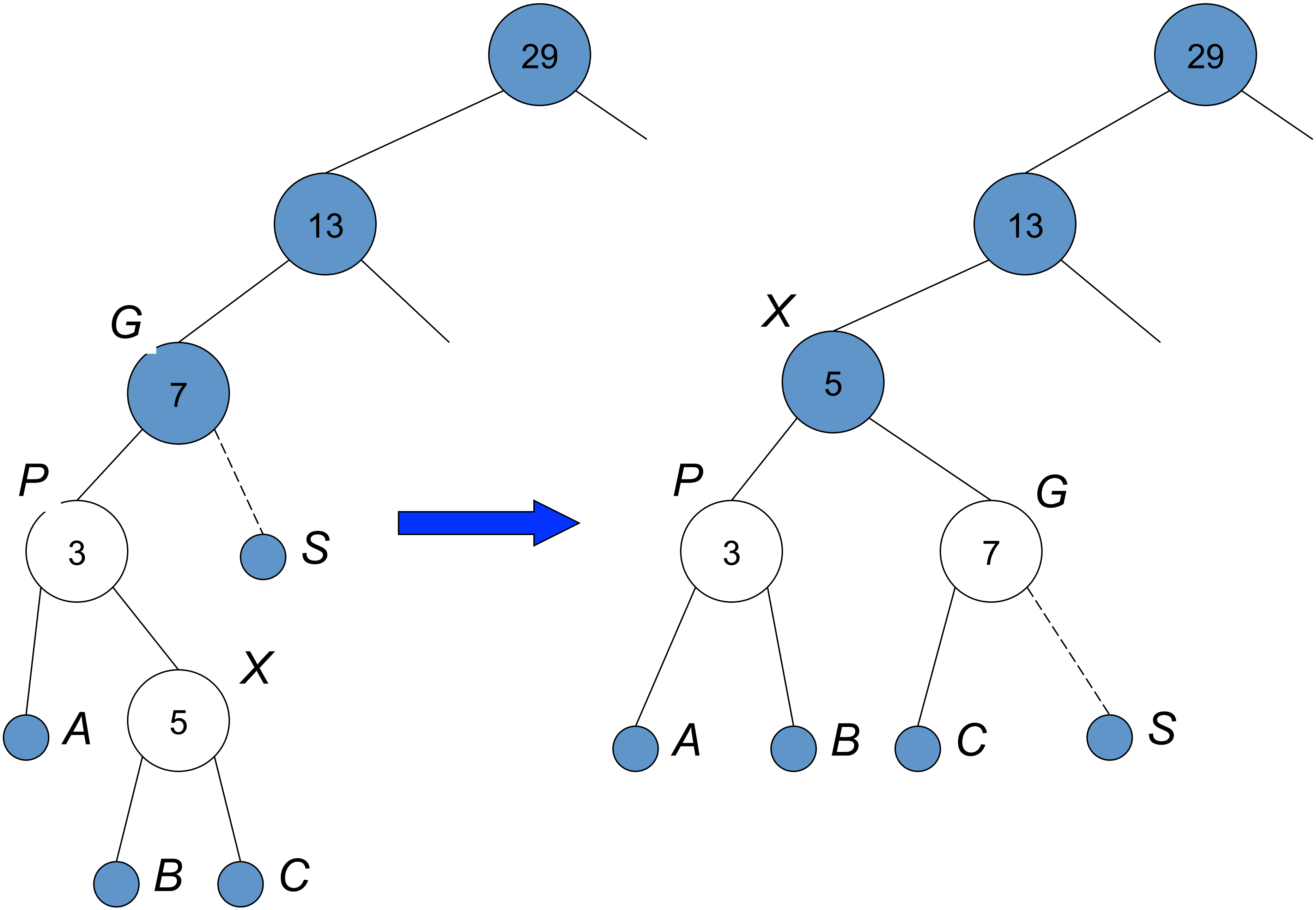
Una primera rotación, a la
izquierda en torno a *P*–*X*,
convierte este caso en el
caso anterior (diap. anterior).

¿Por qué estas rotaciones son correctas? (ver próxima diap.)

Hay que asegurarse que no haya dos nodos rojos consecutivos — las posibilidades son:

- *P* y uno de sus hijos, *A* o *X*
- *G* y su hijo derecho, *S*

Pero *A*, *B* y *C* deben ser negros; de lo contrario, habría habido violaciones adicionales a la propiedad 3 en el árbol original


29
13
G
7
P
3
S
A
5
X
B
C
29
13
X
5
P
3
G
7
A
B
C
S

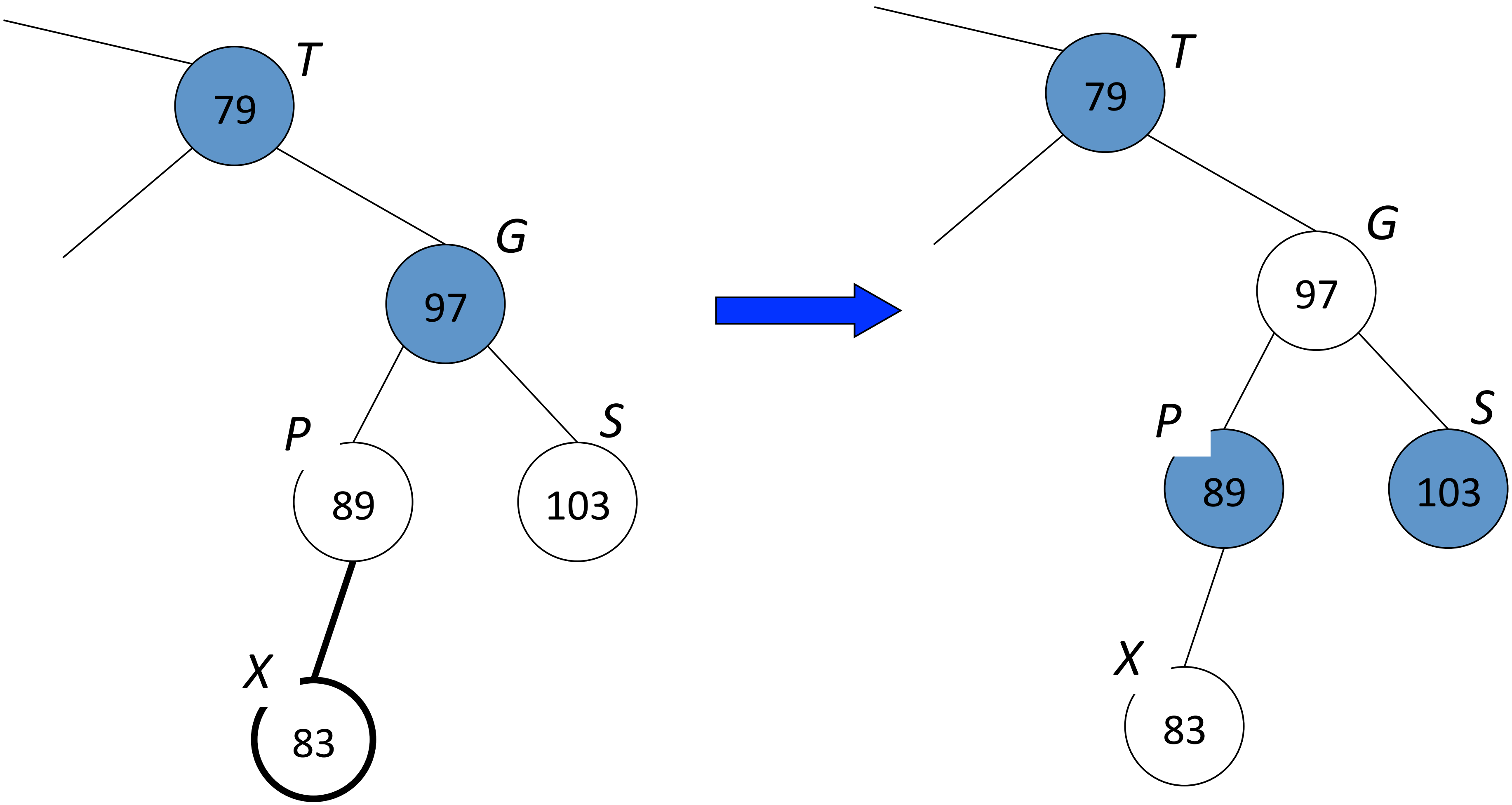
¿Y si el hermano, *S*, del padre, *P*, de *X* es rojo?

- p.ej., insertamos la clave 83 en el árbol de la diap. #70

El abuelo, *G*, de *X* es negro —de lo contrario, no se cumpliría la propiedad 3— por lo tanto:

- pintamos *P* y su hermano *S* de negros —arreglamos el problema de que *X* y su padre sean ambos rojos, pero violamos la propiedad 4
- pintamos el abuelo, *G*, de *X* rojo —restauramos la propiedad 4

Como *P* y *S* son rojos y *G* es negro, pintamos *P* y *S* negros y *G* rojo; así, evitamos que *P* y *X* sean ambos negros, sin cambiar el número de nodos negros en las rutas desde *G* a las hojas. Si *T* fuera rojo, repetimos este procedimiento con *G* en el papel de *X*.



¿Qué pasa si el padre, *T*, de *G* es rojo?

Repetimos los procedimientos anteriores —diaps. #74, 76 y 80— hacia arriba en el árbol

… hasta que no tengamos dos nodos rojos consecutivos

… o lleguemos a la raíz (que pintaremos negra)

La altura de un árbol rojo-negro de n nodos es proporcional a log n, como se demuestra a continuación

Como toda ruta desde la raíz a una hoja contiene el mismo número, B, de nodos negros, el árbol debe contener al menos $2^B - 1$ nodos negros (se puede demostrar por inducción)

Como la raíz es negra

… y no puede haber dos nodos rojos consecutivos en una misma ruta,

… entonces la altura de un árbol rojo-negro de n nodos es a lo más $2\log(n+1)$