Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

TEMA 4. ÁRBOLES

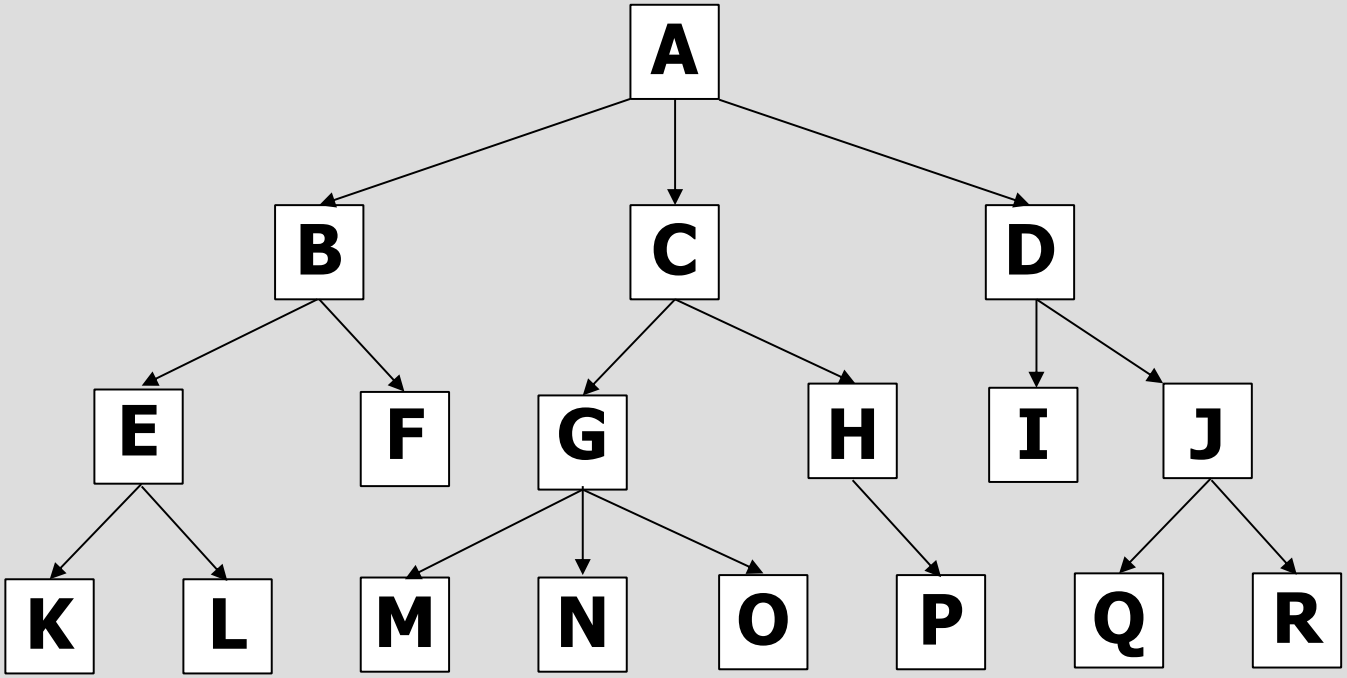
Departamento de Ingeniería
Informática y de Sistemas
Universidad de La Laguna

1. **Introducción.**
2. **Árboles binarios.**
3. **Árboles de búsqueda.**
4. **Árboles AVL**

   Árboles binarios de búsqueda balanceados

Concepto de árbol basado nodos

- Un **árbol** es un TDA formado por nodos cada uno de los cuales es apuntado por un sólo nodo de su mismo tipo excepto el nodo ***raíz***, que no es apuntado por ninguno y desde el que se accede a todos los demás por una serie de enlaces.
- Un árbol de ***grado*** *n* está formado por nodos con *n* apuntadores.
- Las listas simples se pueden interpretar como árboles de grado 1.
- Los ***árboles binarios*** son árboles de grado 2.



Definición recursiva de árbol

El concepto abstracto de árbol se define *recursivamente*.

Un ***árbol*** *T* es una estructura compuesta por:

- una raíz y
- una serie finita de estructuras disjuntas de su mismo tipo.

Formalmente $T = T\,(r\,;\,T_1, T_2, ..., T_k)$, con $T_1, T_2, \ldots T_k$ ***disjuntos***

Las estructurase T_1, T_2, ... y T_k son los ***subárboles*** de *T*.

Si se asume que los subárboles de un árbol están dados en un orden concreto se habla de árbol *ordenado*.

Por lo general los árboles son ordenados aunque no se diga.

Definición recursiva de nodo

Los nodos de un árbol también se definen **recursivamente:**

*"**los nodos de un árbol son:***

***su raíz y los nodos de sus subárboles**"*

Relaciones entre nodos

El nodo raíz del árbol T se dice que:

- es **padre** de los nodos raíz de los árboles T_1, T_2, ...y T_k.

Si el nodo i es padre del nodo j se dice que:

- el nodo j es **hijo** del nodo i.

Se pueden considerar definidas otras **relaciones** entre nodos:

***hermano*, *abuelo, nieto, ascendiente, descendiente*, ...**

Enlaces

- El número de hijos de un nodo se llama ***grado*** del nodo;
- El *grado* de un árbol es el máximo grado de sus nodos.
- Entre los nodos de un árbol se puede distinguir:
  - la ***raíz***,
  - las ***hojas*** (o nodos terminales) que no tienen hijos, y
  - los ***nodos interiores*** que sí los tienen.
- La relación entre padres e hijos se representa por ***enlaces*** que van de cada nodo **hacia** cada uno de sus hijos.
- Un ***camino*** es una secuencia de enlaces de forma que cada enlace parte del nodo al que llega el enlace anterior.
- La ***longitud*** de un camino es el número de enlaces que lo componen.
- Una ***rama*** de un árbol es un árbol que tiene uno de sus nodos como raíz; y está compuesto por todos los descendientes de esta raíz y todos los enlaces de los caminos que parten de ella.

Niveles

Los nodos de un árbol están organizados por ***niveles***.
Las raíces de los subárboles de un árbol cualquiera se dice que tienen un **nivel** *más que la raíz de dicho árbol.*

El **nivel** de un nodo es:

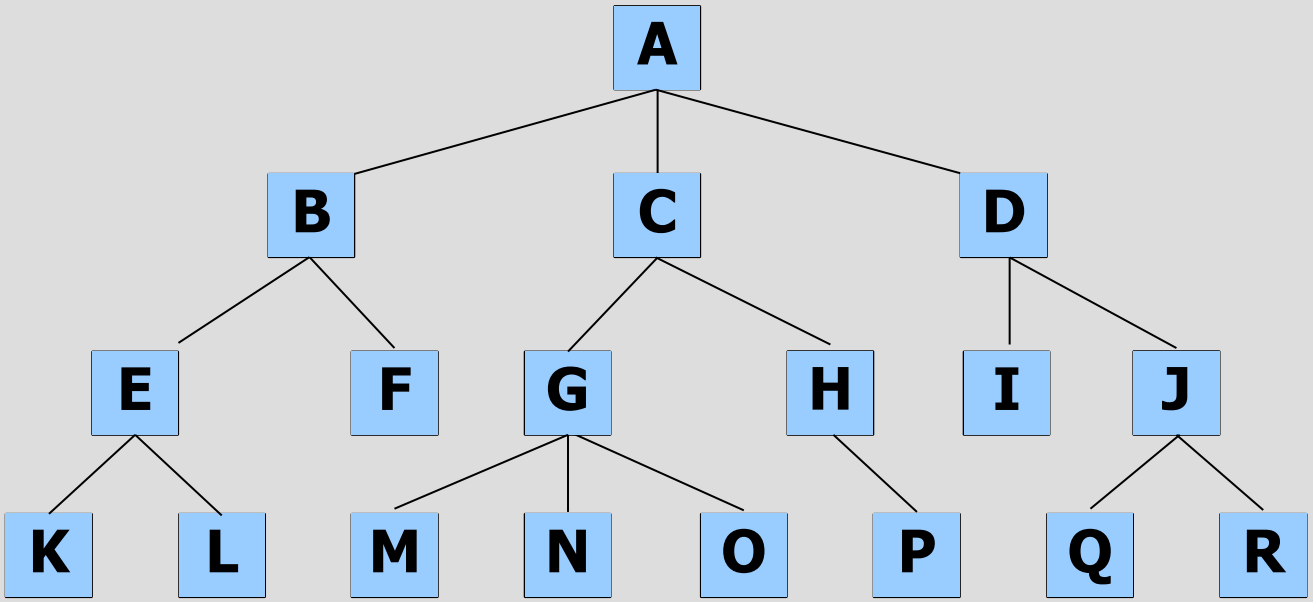
- uno más que el nivel de su padre y
- uno menos que el nivel de cualquiera de sus hijos.

Los niveles se cuentan a partir de la **raíz** empezando por **0** (aunque a veces se empieza por **1)**.

La ***altura*** de un árbol es el nivel máximo de sus nodos.
(***profundidad*** es un sinónimo de nivel o altura)

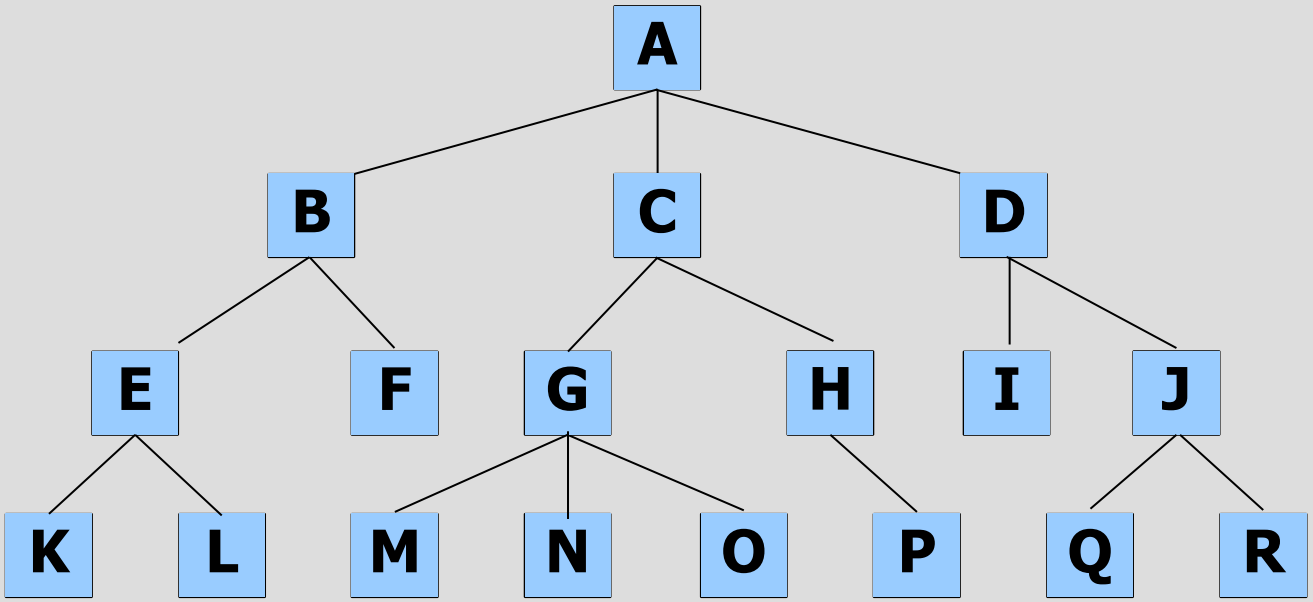
El **tamaño** de un árbol es el número de nodos que tiene.

Ejemplo



- La raíz del árbol T es A.
- El nodo A es padre de B, C y D.
- Los hijos de E son K y L.
- E, F y H son nietos de A.
- El nodo C es abuelo de P.
- Los nodos I y J son hermanos.
- El árbol T tiene tres subárboles de raíces B, C y D (en este orden).
- El nodo B tiene grado 2, H tiene grado 1 y G tiene grado 3.
- Los nodos K, L, F, P e I son nodos hoja.
- Los nodos B, D, G y J son interiores.

Ejemplo



- Existe un arco de C a G y una arista entre J y R.
- Los nodos A,C,G y N, con los arcos que los unen, constituyen un camino.
- La longitud del camino anterior es 3 (tiene tres arcos y 4 nodos).
- El árbol con nodos G, M, N y O es una rama de T pero no un subárbol.
- Si el nodo A es de nivel 0, los nodos de nivel 1 son B, C y D.
- Hay 6 nodos de nivel 2 y 8 de nivel 3.
- El grado del árbol es al menos 3.
- La profundidad del árbol es 3.
- El tamaño de árbol es 18.

TEMA 4. ÁRBOLES

1. **Introducción.**
2. **Árboles binarios.**
3. **Árboles de búsqueda.**
4. **Árboles AVL**

   Árboles binarios de búsqueda balanceados

2. Árboles Binarios.

Los árboles de grado 2 se llaman árboles ***binarios***.

Son los árboles en los que *ningún nodo tiene más de dos hijos*.

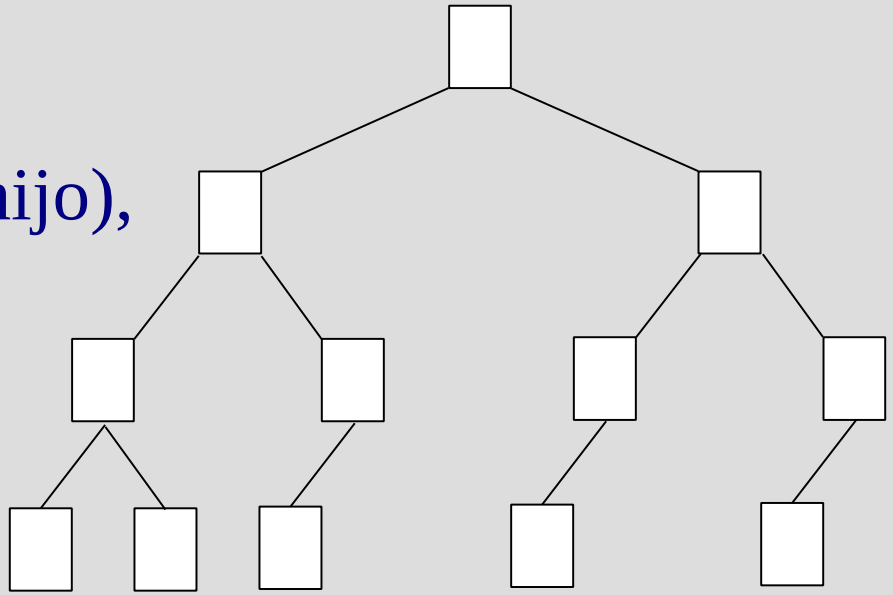
En un árbol binario todos los nodos tienen **grado** 0, 1 ó 2.

Un árbol es binario si ningún nodo tiene ***grado superior*** a 2.

Los árboles:

- con sólo dos nodos (padre e hijo),
- con un solo nodo (la raíz)

son árboles también ***binarios***.



Además, un árbol binario puede ser ***vacío***.

Esto motiva una definición recursiva *distinta* a la de árbol.

Subárboles izquierdo y derecho

Los dos *subárboles* de un árbol binario se denominan: subárbol ***izquierdo*** y subárbol ***derecho***.

Los dos *hijos* de cada nodo de un árbol binario se denominan hijo ***izquierdo*** e hijo ***derecho***.

Puesto que estos dos árboles se consideran ordenados, si se intercambian, el árbol **no** es el mismo árbol.

Además, si un nodo tiene un sólo hijo ***no*** es indiferente que sea el hijo derecho a que sea el izquierdo.

En general, el *orden* entre los hijos es de izquierda a derecha.

Se puede definir un árbol binario como un árbol donde: ***todas sus ramas tienen exactamente dos subárboles*** (que pueden ser vacíos).

Resultados formales

Definición. Un **árbol binario** B se define recursivamente como un árbol vacío o un árbol constituido por:

un nodo raíz y un par de subárboles binarios B_I y B_D

Formalmente: $B = \varnothing$ o $B = B\,(\,r\,;\,B_I\,,\,B_D\,)$.

Teorema. Sea 0 el nivel de la raíz de un árbol binario. Entonces:

- El número máximo de nodos de nivel i es 2^i.
- El número máximo de nodos de un árbol profundidad p es $2^{p+1}-1$
- El número de nodos terminales de un árbol es 1 más el número de nodos de grado 2.
- La profundidad mínima de un árbol de n nodos es $p = \lfloor \log_2 n \rfloor$
- La profundidad máxima de un árbol de n nodos es $p = n$

Demostración.

Todos los apartados se demuestran fácilmente

por inducción en el número n de nodos.

Demostraciones

- El número máximo de nodos de nivel i es 2^i.

Cierto para $i = 0,1,2, \ldots$, Si a cada nodo de nivel i le ponemos dos hijos se obtienen $2 \cdot 2^i = 2^{i+1}$ nodos de nivel $i+1$

- El número máximo de nodos de un árbol profundidad p es $2^{p+1}-1$

Cierto para $p = 0,1,2, \ldots$ Si al árbol de profundidad p añadimos los 2^{p+1} de nivel $p+1$ obtenemos uno de profundidad $p+1$ con $2^{p+1} + 2^{p+1}-1 = 2^{p+2} -1 = 2^{(p+1)+1} -1$ nodos

- El nº de nodos terminales es 1 más el nº de nodos de grado 2.

Si t es el nº de nodos terminales y d el de nodos de grado 2. La fórmula $t = 1+d$ es cierta para un árbol con $n = 1,2,3, \ldots$, nodos. Si a este árbol le añadimos un nodo puede ser como hijo de un nodo de grado 1 o de grado 0 (terminal). En el primer caso aumenta el nº de nodos terminales y de grado 2; $t+1 = 1 +(d+1)$. En el segundo caso no cambian dichos números y sigue siendo cierta $t = 1 + d$

- La profundidad mínima de un árbol de n nodos es $p = \lfloor \log_2 n \rfloor$

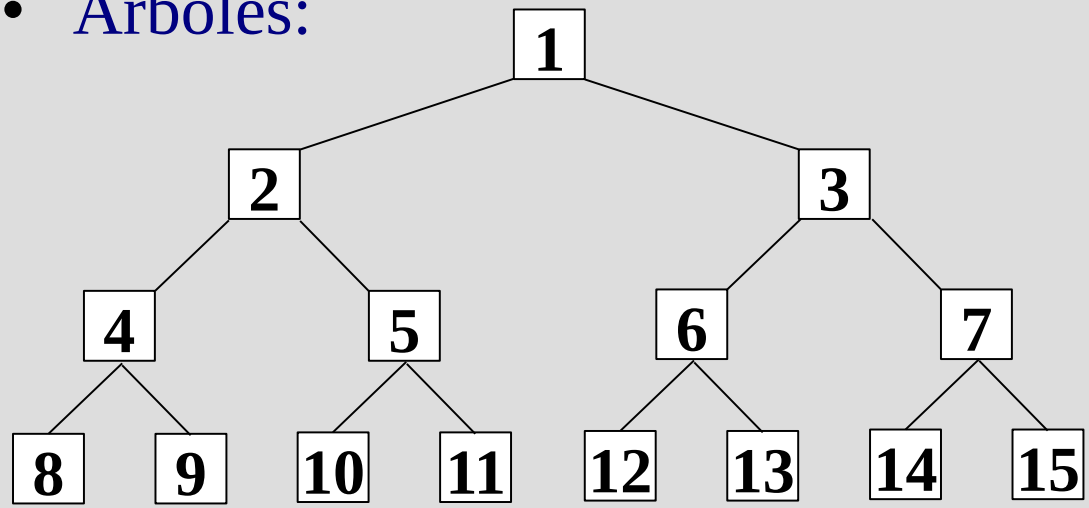
Los árboles de menor profundidad son árboles completos por niveles. El nº n de nodos de un árbol completo por niveles de profundidad 2 está entre 8 y 15. En general, para un árbol completo por niveles de profundidad p, el nº n de nodos está entre 2^p y $2^{p+1}-1$; es decir $2^p \le n < 2^{p+1}$. Por tanto $p = \lfloor \log_2 n \rfloor$ es la profundidad mínima del árbol de n nodos

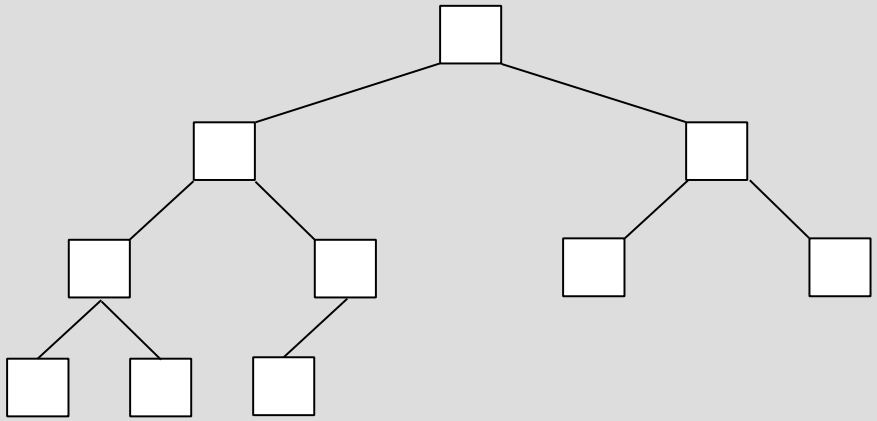
- La profundidad máxima de un árbol de n nodos es $p = n$

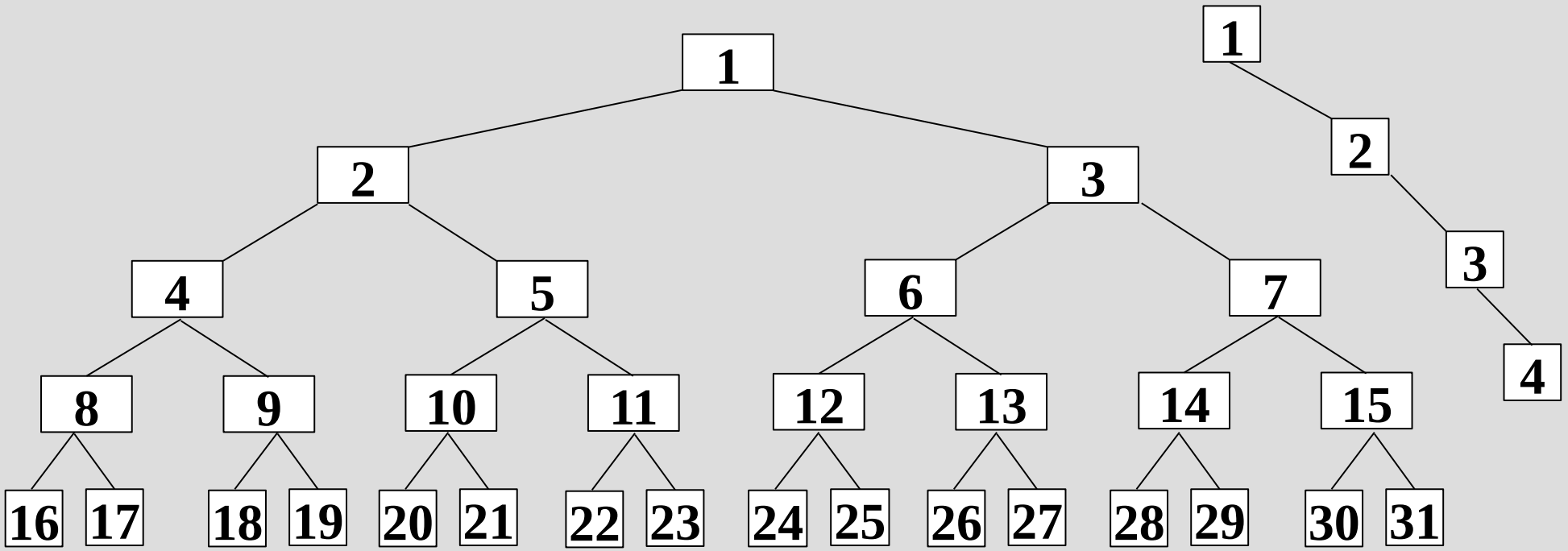
El árbol de máxima profundidad es una cadena de $p = n$ nodos

Ejemplos de árboles

- Árboles:







Árboles completos y equilibrados

- Un árbol ***lleno*** es el que tiene todos sus nodos con grado 2 (se dice que los nodos están ***saturados***) excepto los del último nivel que no tienen hijos y son nodos terminales.
- Un árbol ***completo por niveles*** de tamaño *n* tiene ocupadas todas las posiciones posibles en el orden por niveles hasta su tamaño.
- Un árbol binario está ***equilibrado*** si la diferencia de los tamaños de sus dos subárboles es uno a lo sumo.
- Un árbol está ***perfectamente equilibrado*** o ***totalmente equilibrado*** si están equilibradas todas sus ramas.

  (generalmente, cuando se habla de árboles ***equilibrados*** se refiere a árboles perfectamente o totalmente equilibrados)
- Un árbol (totalmente) ***equilibrado*** con *n* nodos *se obtiene* a partir de dos subárboles hijos equilibrados con:

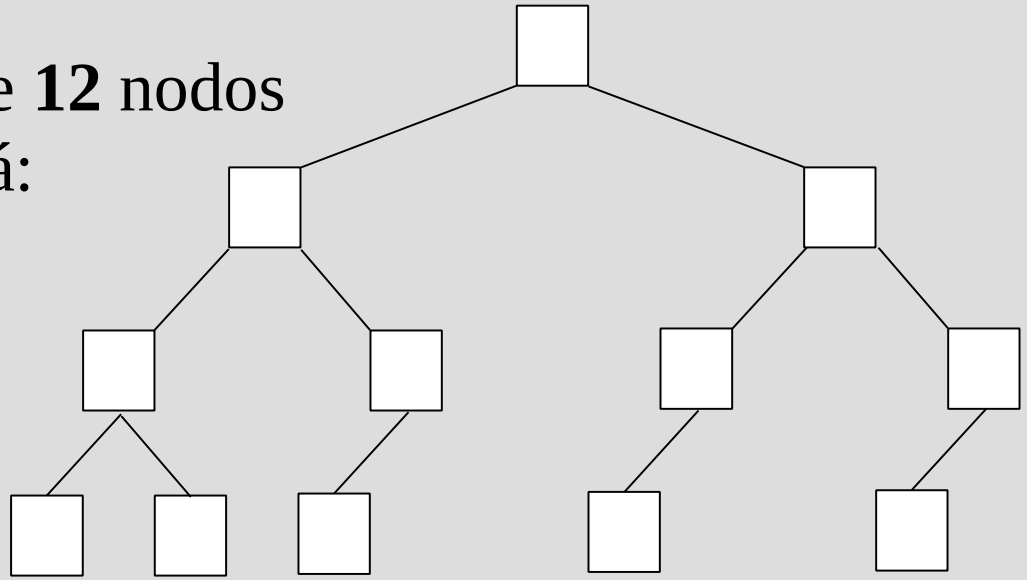
  $n_I = \lfloor n/2 \rfloor$ y $n_D = n - \lfloor n/2 \rfloor - 1$ nodos, respectivamente.

Ejemplo de árbol equilibrado

Si los árboles equilibrados son construidos de esta forma y
$\lfloor n/2 \rfloor$ es el número de nodos del subárbol izquierdo:
Entonces los árboles resultantes tienen una forma **fijada a priori**.

Ejemplo:

Un árbol binario equilibrado de **12** nodos
construido de esta forma tendrá:
dos subárboles de 6 y 5 nodos;
el primero de éstos tendrá:
un subárbol de 3 nodos y
otro de 2 nodos,
y el otro tendrá
dos subárboles de 2 nodos.



Los árboles ***equilibrados de 2 nodos*** tendrán el único hijo de la raíz en el lado izquierdo; están formados por la raíz y su hijo izquierdo.

Las tres ordenaciones básicas

Existen diversas formas usuales de establecer una ordenación para ***recorrer*** todos los nodos de un árbol binario.

- En el recorrido en ***preorden***: se recorre la raíz del árbol antes que los dos subárboles; por tanto el recorrido consiste en recorrer siempre primero la raíz, luego el subárbol izquierdo y finalmente el subárbol derecho; *recorriendo estos subárboles por el mismo procedimiento*.
- En el recorrido en ***postorden***: se recorre la raíz después de sus hijos; el orden es: primero el hijo izquierdo, luego el hijo derecho y por último la raíz, *recorriendo los subárboles por el mismo procedimiento para cada raíz de sus ramas*.
- En el recorrido en ***inorden***; el nodo raíz se recorre entre sus dos hijos; el orden es siempre el hijo izquierdo, la raíz y el hijo derecho.
  (este orden es también llamado *orden simétrico*)

(se recorre siempre el subárbol izquierdo antes que el derecho)

Otras ordenaciones

Si el orden entre los dos subárboles se establece *en sentido contrario* se obtienen otros tres recorridos inversos.

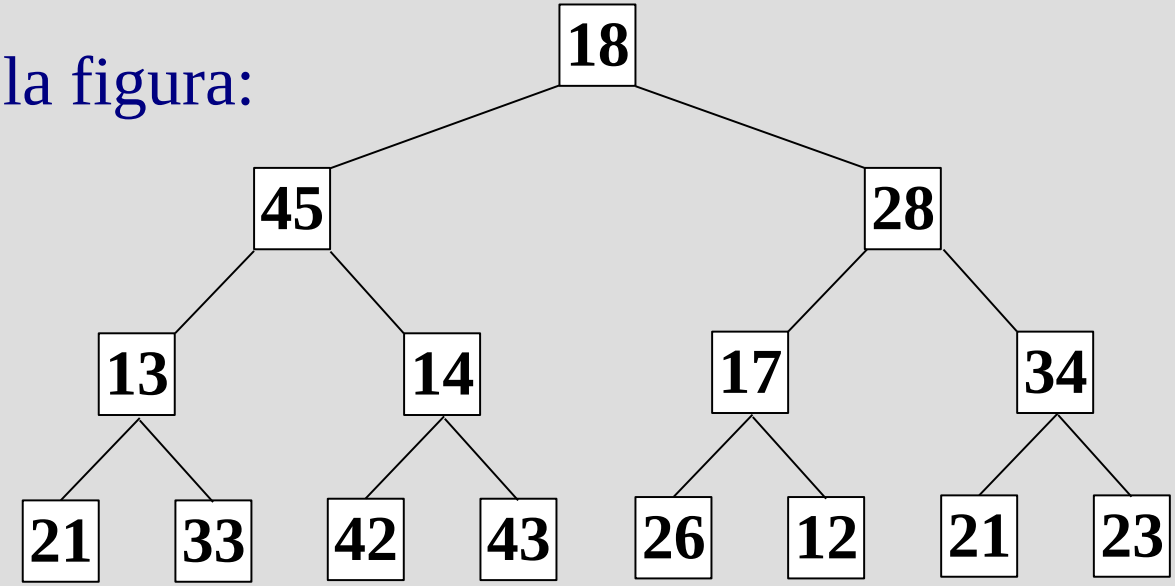
Además de estos tres recorridos está el **recorrido por niveles**.
En la ordenación o recorrido ***por niveles*** se recorren los nodos de los diferentes niveles del árbol en orden creciente desde la raíz y dentro de cada nivel de izquierda a derecha;
En otras palabras:

Se recorre el padre antes que sus hijos y, dentro de cada nivel, los descendientes del hijo izquierdo se recorren antes que los del hijo derecho.

Un árbol binario con *n* nodos está ***completado por niveles*** si tiene todos sus nodos en las *n* primeras posiciones posibles en el orden por niveles.

Ejemplo de recorridos

El árbol binario de la figura:



es un árbol binario completo de altura 3 que tiene $15 = 2^4 - 1$ nodos.
Tiene $8 = 2^3$ nodos de nivel 3, que son sus hojas;
El número de nodos de grado 2 es 7 (= 8 – 1).

Los recorridos de sus nodos encuentran los valores siguientes:

Preorden:	18	45	13	21	33	14	42	43	28	17	26	12	34	21	23
Postorden:	21	33	13	42	43	14	45	26	12	17	21	23	34	28	18
Inorden:	21	13	33	45	42	14	43	18	26	17	12	28	21	34	23
Por niveles:	18	45	28	13	14	17	34	21	33	42	43	26	12	21	23

Implementación de árboles binarios

El TDA árbol binario se suele implementar usando ***estructuras dinámicas*** de datos, aunque es posible usar estructuras estáticas.

El **TDA** árbol binario se basa en el tipo de datos ***nodo binario***.

Los ***nodos binarios*** son tipos de datos cuyos elementos son registros con campos de datos y dos enlaces o apuntadores a otros nodos del mismo tipo o a la dirección nula; los nodos apuntados por esos enlaces son sus hijos ***izquierdo*** y ***derecho***.

Un **árbol binario** es un TDA formado por nodos binarios cada uno de los cuales es apuntado por un sólo nodo binario excepto uno, denominado ***raíz***, que no es apuntado por ninguno y desde el que se accede a todos los demás nodos a través de enlaces.

A los nodos del árbol binario se accede sólo a través de la raíz.

Operaciones básicas

Iniciales:

- **Crear_Arbol**. Crear un árbol vacío o crear un árbol equilibrado.
- **Es_Vacío.** Decir si el árbol es vacío.
- **Imprimir_Arbol**. Imprimir los nodos del árbol según su estructura.

Recorridos:

- **Pre_orden**. Recorrer: **R**aíz, Subárbol **I**zdo, Subárbol ***D***cho.
- **Post_orden**. Recorrer: Subárbol **I**zdo, Subárbol **D**cho, **R**aíz.
- **In_orden**. Recorrer: Suba. **I**zdo, **R**aíz, Suba. **D**cho. (O***rd simétrico***)

Inserciones:

- **Insertar_hoja**.
- **Insertar_raíz.**
- **Insertar_equilibrado**.
- **Insertar_ordenado.**

Eliminaciones:

- **Buscar_datos**.
- **Eliminar** hoja o nodo interior.
- **Eliminar** el primer nodo o último en orden.

Inserciones

- La operación de inserción de un *nuevo* nodo se realiza de forma muy sencilla enlazándolo como hijo de un nodo ***hoja***.
- Igual de sencilla es la inserción como ***raíz***; la raíz anterior pasa a ser hijo de la nueva raíz, en cualquiera de sus lados.
- También es simple la inserción como hijo de un nodo interior de ***grado uno***; enlazándolo como hijo del lado vacante.
- En la mayoría de las aplicaciones, el lugar de inserción viene determinado por la ***información asociada*** a los nodos.
- Si el nuevo nodo se inserta como un nodo interior se modifica la relación padre/hijo entre algunos nodos existentes.
- Se debe mantener las propiedades deseables de la estructura de datos (***ordenación de los nodos*** y ***equilibrio del árbol***).

Eliminaciones

- Para eliminar un nodo primero habrá que **localizarlo**.
- A falta de información adicional sobre la disposición de los nodos, cualquiera de los ***recorridos*** puede ser usado.
- Si resulta ser un **nodo hoja**, la eliminación es muy sencilla.
- Sin embargo, si se elimina un **nodo interior** es necesario alterar los enlaces padre/hijo de otros nodos.
- Para mantener el ***equilibrio*** en la eliminación, se reemplaza la información del nodo a eliminar por la de un nodo hoja cuya eliminación no destruya el equilibrio.
- Partiendo de la raíz y pasando al subárbol de mayor tamaño (o a cualquiera de ellos si son de igual tamaño) se alcanza un nodo hoja cuya eliminación ***mantiene*** el equilibrio.

TEMA 4. ÁRBOLES

1. **Introducción.**

2. **Árboles binarios.**

3. **Árboles de búsqueda.**

4. **Árboles AVL**

   Árboles binarios de búsqueda balanceados

3. Árboles de Búsqueda.

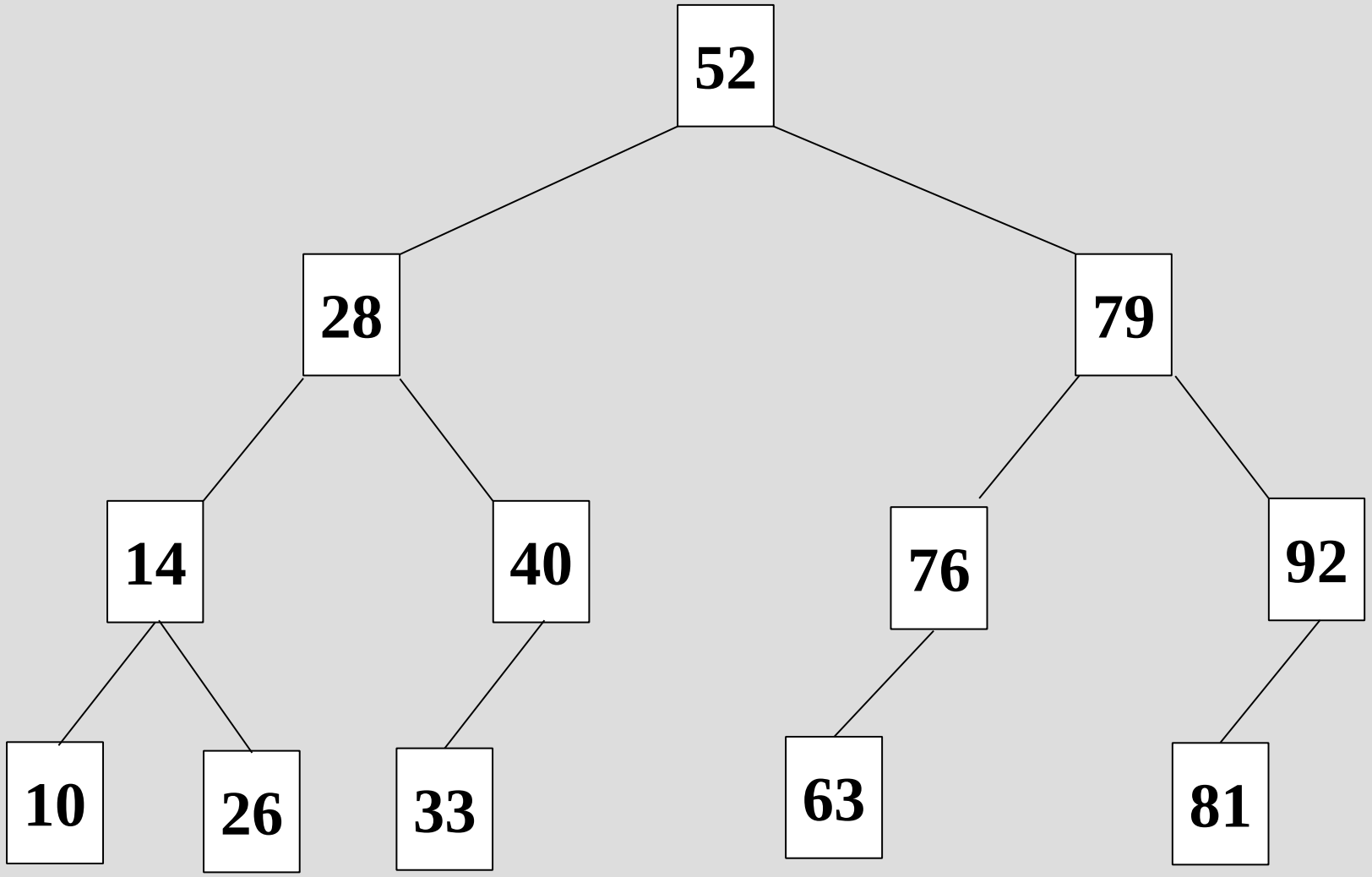
- Los árboles binarios constituyen una herramienta muy útil para almacenar y gestionar un conjunto de datos cuyos elementos se organizan de acuerdo a una determinada **clave**.
- Los valores de la clave deben admitir **comparaciones** de acuerdo a una relación de orden (*menor*, *anterior*, …).
- La estructura de árbol se relaciona con dicha clave, presente en los nodos, de forma que la búsqueda de los elementos a través del árbol aprovechando la clave es muy **eficiente**.
- Un **árbol de búsqueda** es aquel en el que:

  *la clave de la **raíz** de cada **rama** es:*

  ***mayor** que las de los nodos de su subárbol **izquierdo** y*
  ***menor** que las de los nodos de su subárbol **derecho**;*

  es decir, están en ***orden simétrico.***

Ejemplo de árbol de búsqueda.



La búsqueda

La búsqueda del nodo asociado a una clave se realiza:

accediendo por la ***raíz*** y,

- si la clave de la raíz ***coincide*** con la clave buscada
  la búsqueda finaliza con **éxito**.
- en otro caso, *recursivamente*,
  - si la clave buscada es ***menor*** que la de la raíz:
    se pasa al árbol ***izquierdo,***
  - si la clave buscada es ***mayor*** que la de la raíz
    se pasa al árbol ***derecho.***

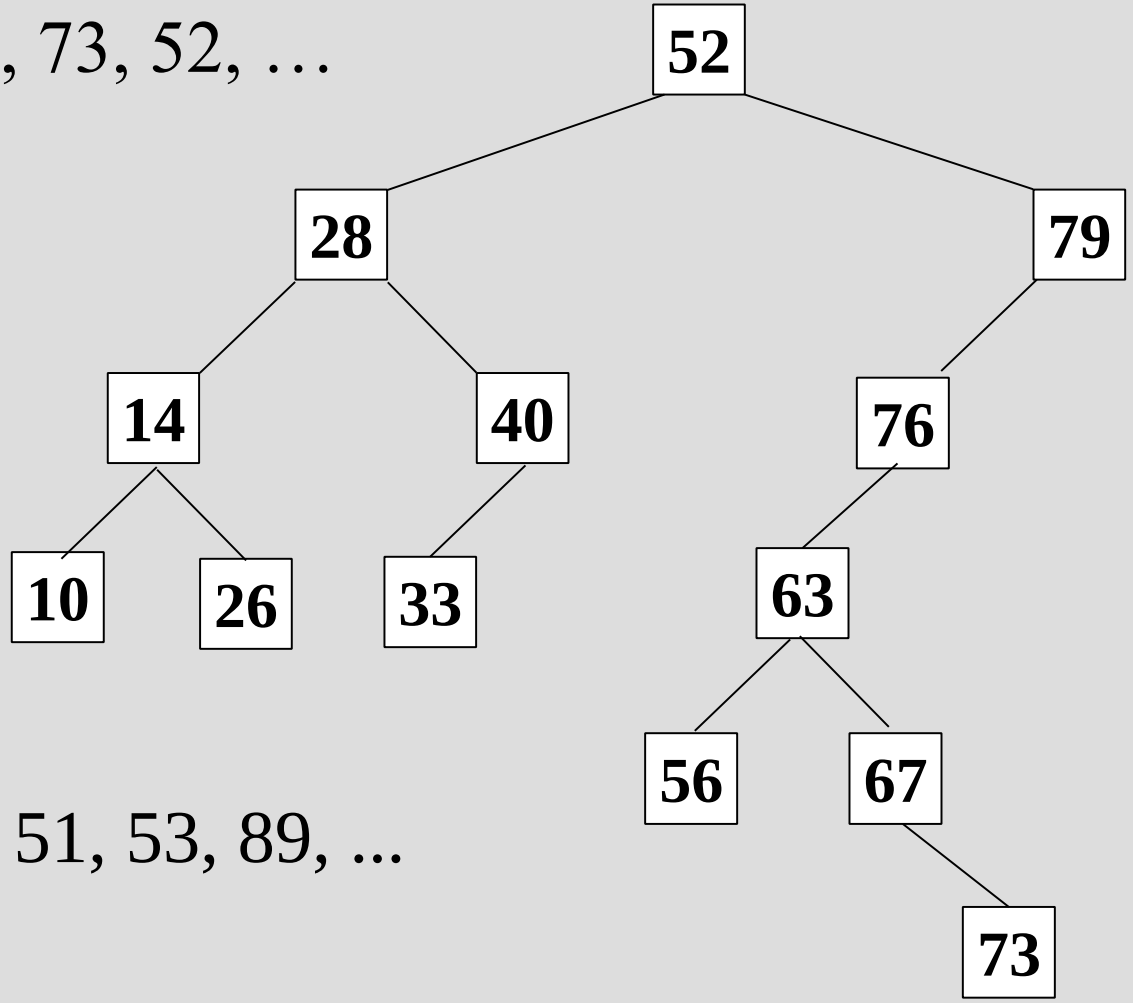
la búsqueda finaliza en **fracaso** si llega a un árbol vacío.

La inserción

- Para ***insertar*** un nuevo nodo en un árbol de búsqueda de forma que se mantenga el orden simétrico en el árbol de búsqueda se aplica el ***procedimiento de búsqueda*** de la clave del nuevo nodo como si estuviera ya insertado.
- Si el nodo correspondiente *no se encuentra* en el árbol la búsqueda **acaba** en una posición en la que se puede insertar el nuevo nodo ***manteniendo*** el orden simétrico.
- Si se admite que la misma clave se *repita* en varios nodos una posición de inserción posible se encuentra desde la raíz ***prolongando***, hasta que el árbol correspondiente esté vacío, siguiendo el mismo procedimiento de la búsqueda y tomando, en el caso de que las claves coincidan, uno *cualquiera* de los dos subárboles.

Ejemplos de Inserción y Búsqueda

Buscar 26, 76, 40, 67, 73, 52, …



Insertar 27, 51, 53, 89, ...

Eliminación

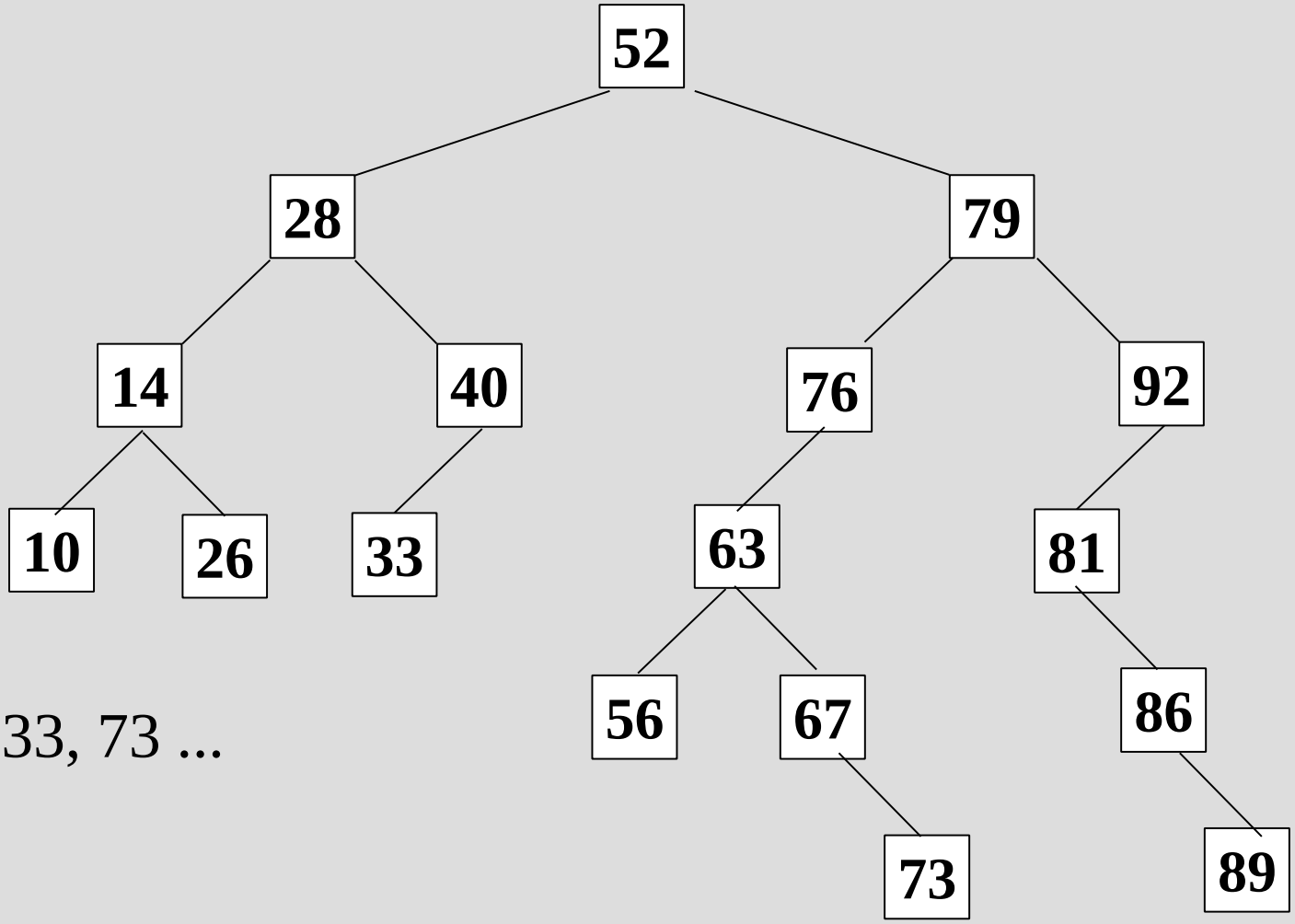
Para eliminar o borrar un elemento, una vez **localizado**,

- Si está en nodo **hoja** se puede eliminar sin más.
- En otro caso
  - Si tiene **un** solo **hijo**, se sustituye por éste, que trae consigo todo el subárbol correspondiente (*ver ejemplo*)
  - Si el nodo a eliminar tiene **dos hijos** se puede sustituir
    - por el de nodo *inmediatamente* **anterior** en el orden simétrico, manteniendo así el orden, o
    - por el de nodo *inmediatamente* **posterior** en el orden simétrico, manteniendo también el orden.

    Este nodo ***sustituto*** está en las **condiciones anteriores**, (es decir, no tiene hijos o tiene sólo uno) y se elimina.

La información del sustituto sustituye la del nodo a eliminar.

Eliminaciones sencillas.



Eliminar 10, 33, 73 ...

Eliminar 86

Eliminar 40 (33 cambia de lado),

Eliminar 76 (63 sube con su rama), ...

Buscando sustituto

Si el nodo a eliminar tiene dos hijos,

- El nodo con clave **inmediatamente anterior** en el orden simétrico es el nodo de ***mayor*** clave de su subárbol ***izquierdo***
- El nodo con clave **inmediatamente posterior** en el orden simétrico es el nodo de ***menor*** clave de su subárbol ***derecho***

- El elemento con **menor** clave de un árbol se determina a partir de la raíz pasando al hijo izquierdo hasta que no exista.
- El elemento de **mayor** clave se obtiene desde la raíz pasando al hijo derecho mientras exista.

En cualquiera de los casos, el nodo resultante **no tiene** *grado 2.*

Para ***concretar*** el procedimiento de eliminación hay que optar por una de estas dos posibilidades (a *derecha* o a *izquierda*).

Concretando el procedimiento

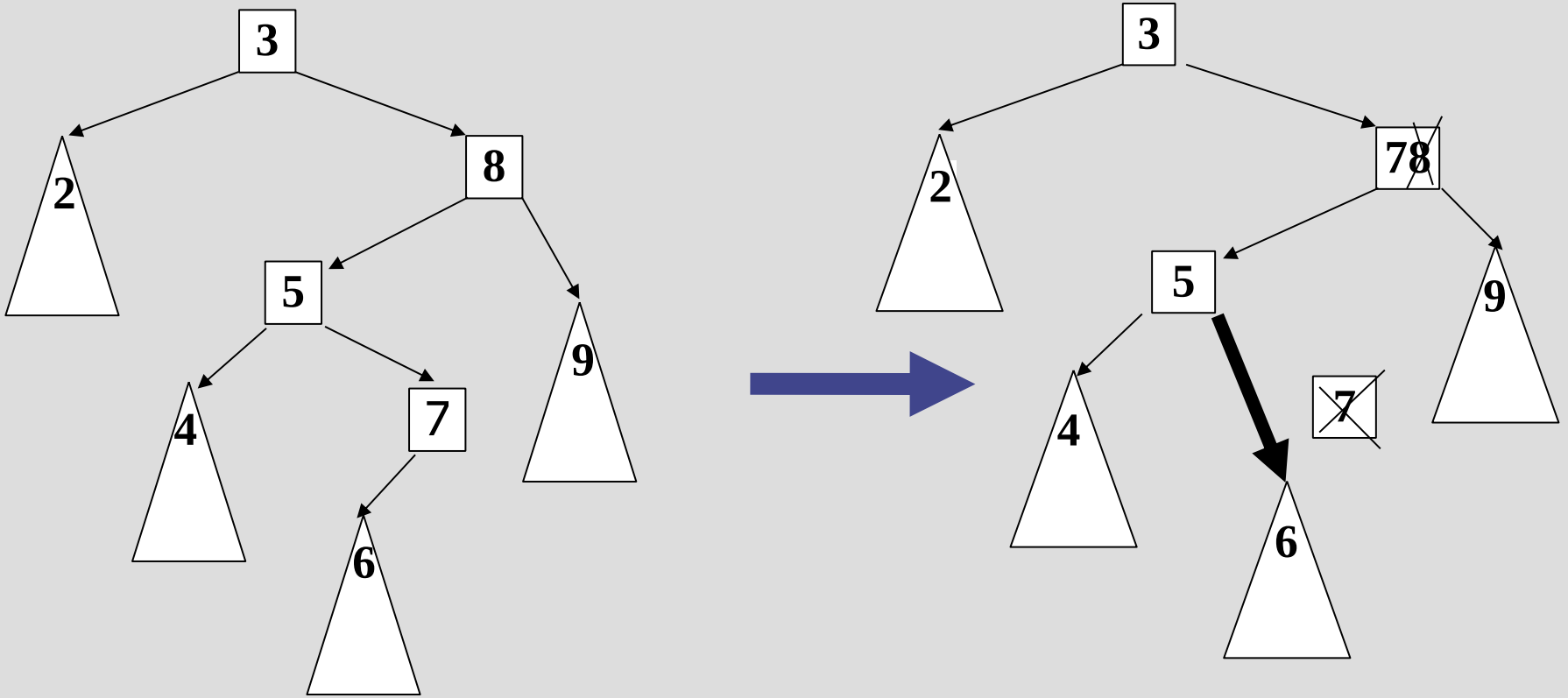
- Si el nodo a eliminar tiene dos hijos optamos por sustituirlo por el de ***mayor clave de su subárbol izquierdo***.
- Por tanto, tras localizar el nodo a eliminar, si tiene dos hijos, accedemos a su hijo ***izquierdo*** y desde este nodo pasamos *recursivamente* al hijo ***derecho*** hasta que no exista, en cuyo caso hemos determinado el sustituto.
- ***Reemplazamos*** los valores de la clave y datos del nodo a eliminar por los correspondientes del sustituto y éste nodo es el nodo que eliminamos *físicamente* del árbol.
- El nodo sustituto es una ***hoja*** o tiene **sólo** un hijo izquierdo.
  - Si es una ***hoja*** no se necesita ninguna otra modificación.
  - En otro caso (*el sustituto tiene sólo un hijo izquierdo*), enlazamos el hijo izquierdo del nodo sustituto con el padre en el ***mismo*** lado, izquierdo o derecho, que está el sustituto.

Ejemplo de eliminación

- Si el nodo a eliminar tiene dos hijos lo sustituimos por el nodo de mayor clave de su subárbol izquierdo;

desde su hijo izquierdo, pasamos recursivamente al hijo derecho hasta que no exista, en cuyo caso hemos llegado al sustituto.

Eliminación del nodo 8



TEMA 4. ÁRBOLES.

1. Introducción.
2. Árboles binarios.
3. Árboles de búsqueda.
4. **Árboles AVL**

   ~~*Árboles binarios de búsqueda equilibrados*~~

   Árboles binarios de búsqueda balanceados

Ventajas del equilibrio

- La principal ventaja del uso de árboles binarios de búsqueda es que el **número de operaciones** en las búsquedas está acotado por la **profundidad** del árbol.

- Si un árbol de *n* nodos está perfectamente equilibrado (*la diferencia entre el número de nodos de los dos subárboles de cada rama no supera la unidad*) la profundidad del árbol está acotada por $\log_2 n$.
- Lo importante es que la profundidad máxima del árbol crezca de forma ***logarítmica*** con el tamaño del árbol; que la profundidad sea **O(log *n*)** si tiene *n* nodos

Árbol equilibrado y ordenado

- Si se conocen todas las claves que se van a utilizar en un árbol de búsqueda se pueden ordenar previamente.
- Una vez ***creado el árbol equilibrado*** del tamaño requerido según vimos se pueden introducir los datos ***ordenadamente*** mediante un recorrido simétrico.
- Pero al **modificar** las datos insertando o eliminando elementos sería necesario **reequilibrar** el árbol.
- **Mantener** el equilibrio perfecto de un árbol tras las operaciones de inserción y borrado de nodos es una operación ***muy costosa***, lo que hace que no sea conveniente en muchas aplicaciones.

Alternativas al equilibrio

- ***No se ha encontrado*** una forma eficiente de combinar el concepto de equilibrio (perfecto) con la ordenación que requiere un árbol de búsqueda
- Por ello se han buscado unas condiciones, ***no tan estrictas,*** pero que también permitan acotar *logarítmicamente* la profundidad del árbol en relación a su tamaño y que realmente sean más ***eficientes*** de mantener.
- Una de las propuestas para mantener la profundidad o altura en **O(log *n*)** es la de los **árboles AVL** (***Adelson-Velskii*** y ***Landis***).

4. Árboles AVL

Un árbol binario está ***balanceado*** si la diferencia de las alturas de los dos subárboles de ***cada rama*** no supera a la unidad.

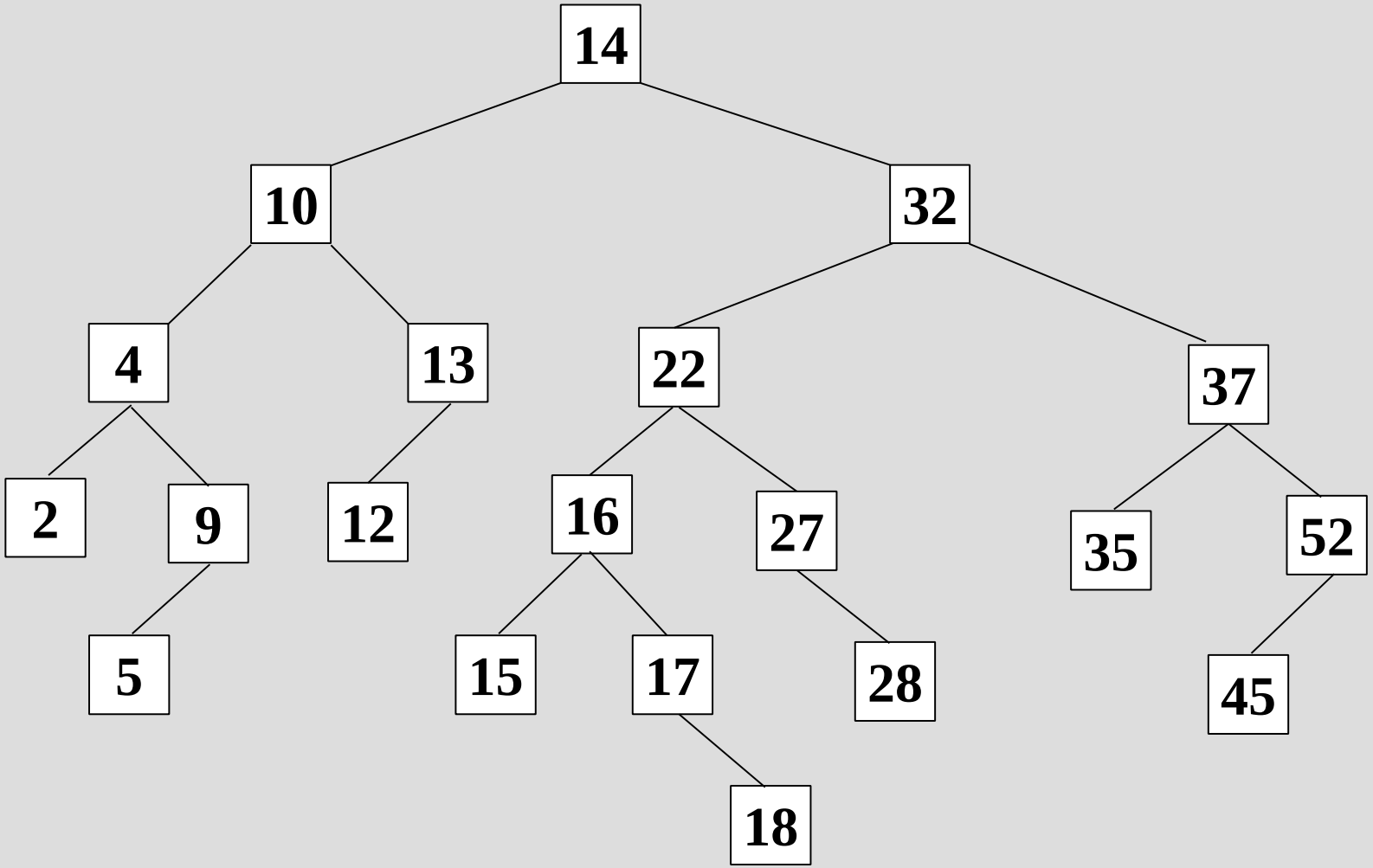
- ***Se sustituye tamaño por altura de las ramas en el equilibrio***

El ***rebalanceo*** tras una inserción o eliminación es más sencillo.

La altura de un árbol binario se obtiene ***recursivamente*** como uno más el máximo de las alturas de sus dos subárboles (*tomando la profundidad de un árbol vacío como* 0).

Un árbol AVL es un árbol binario de búsqueda balanceado

Ejemplo de AVL



El desbalanceo

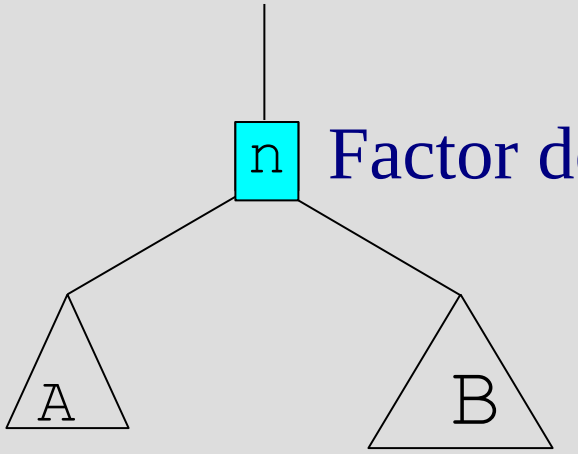
- Cuando se inserta un nuevo nodo en un árbol de búsqueda balanceado (un árbol AVL) en el orden que le corresponde según el orden simétrico, se ***inserta*** como una nueva hoja, por lo que puede producir un *desbalanceo* pero sólo en los nodos del *recorrido* de la búsqueda desde la raíz para encontrar la posición de inserción.
- Análogamente cuando se **elimina** un nodo de un árbol de búsqueda balanceado (un árbol AVL) siguiendo el procedimiento anteriormente descrito para mantener el orden simétrico, sólo puede producir un *desbalanceo* en los nodos del *recorrido* de la búsqueda del nodo correspondiente.
- Además, en ambos casos, el desbalanceo sólo puede llegar a producirse porque la ***diferencia*** de las alturas de los dos subárboles de una rama llega a ser **dos (2)**.

Detectando desbalanceos

- Los posibles ***desbalanceos*** tras la inserción o eliminación del árbol AVL se van detectando al recorrer los nodos seguidos por la búsqueda correspondiente en orden *inverso* al proceso de búsqueda; desde la hoja afectada hasta la raíz.
- Las operaciones de **rebalanceo** en cada nodo podrán producir o no una modificación de la altura de su rama lo que eventualmente permitiría detectar *recursivamente* si se produce un desbalanceo en el padre.
- Esta comprobación se realiza de forma más eficiente si se añade a cada nodo del árbol un campo que refleje la diferencia entre las alturas de los dos subárboles de la rama que lo tiene por raíz, denominado **factor de balanceo.**

El factor de balanceo

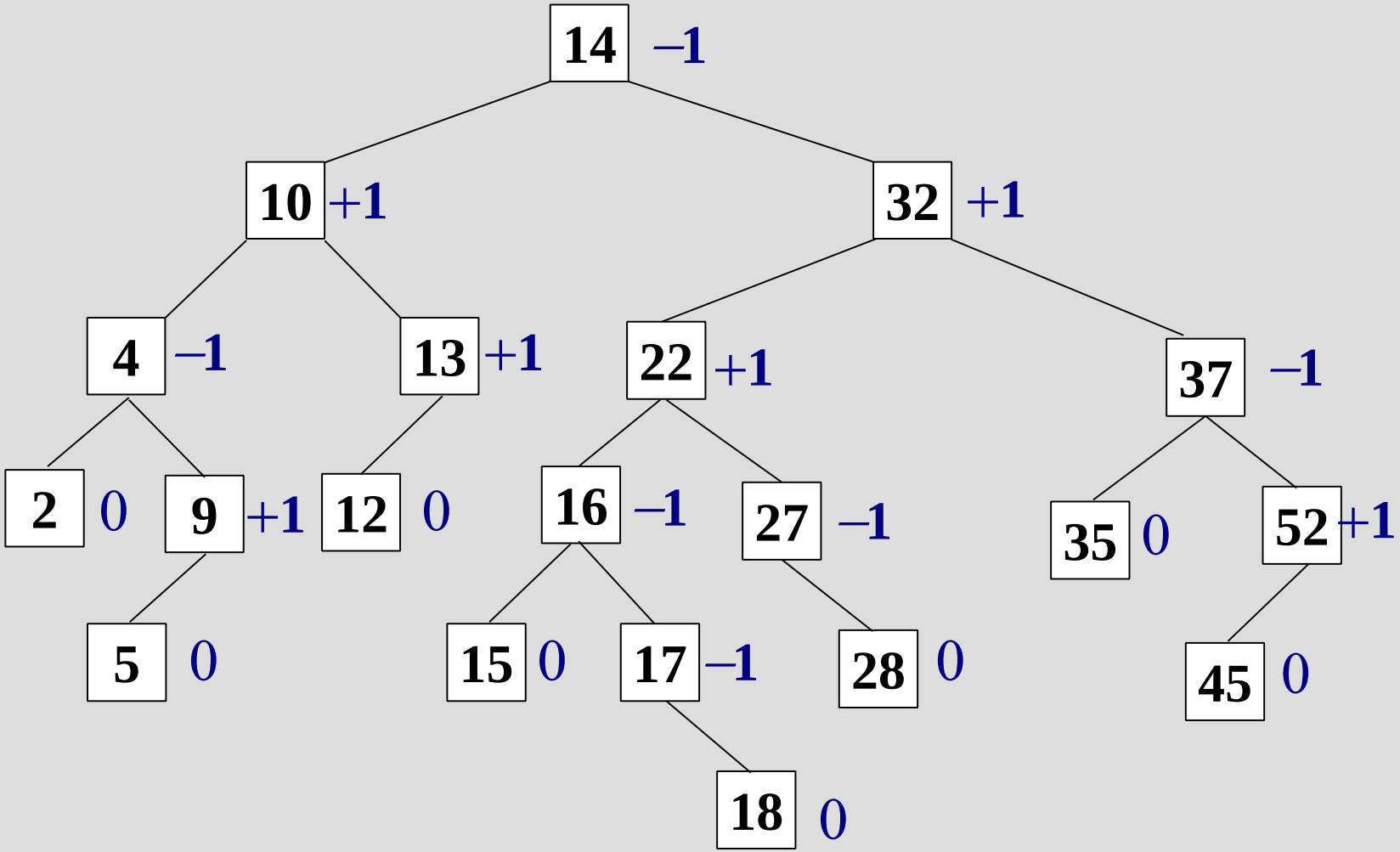
- El **factor de balanceo** de un nodo en un árbol es:
  la altura del subárbol izquierdo de la rama que lo tiene por raíz menos la altura del subárbol derecho de dicha rama.



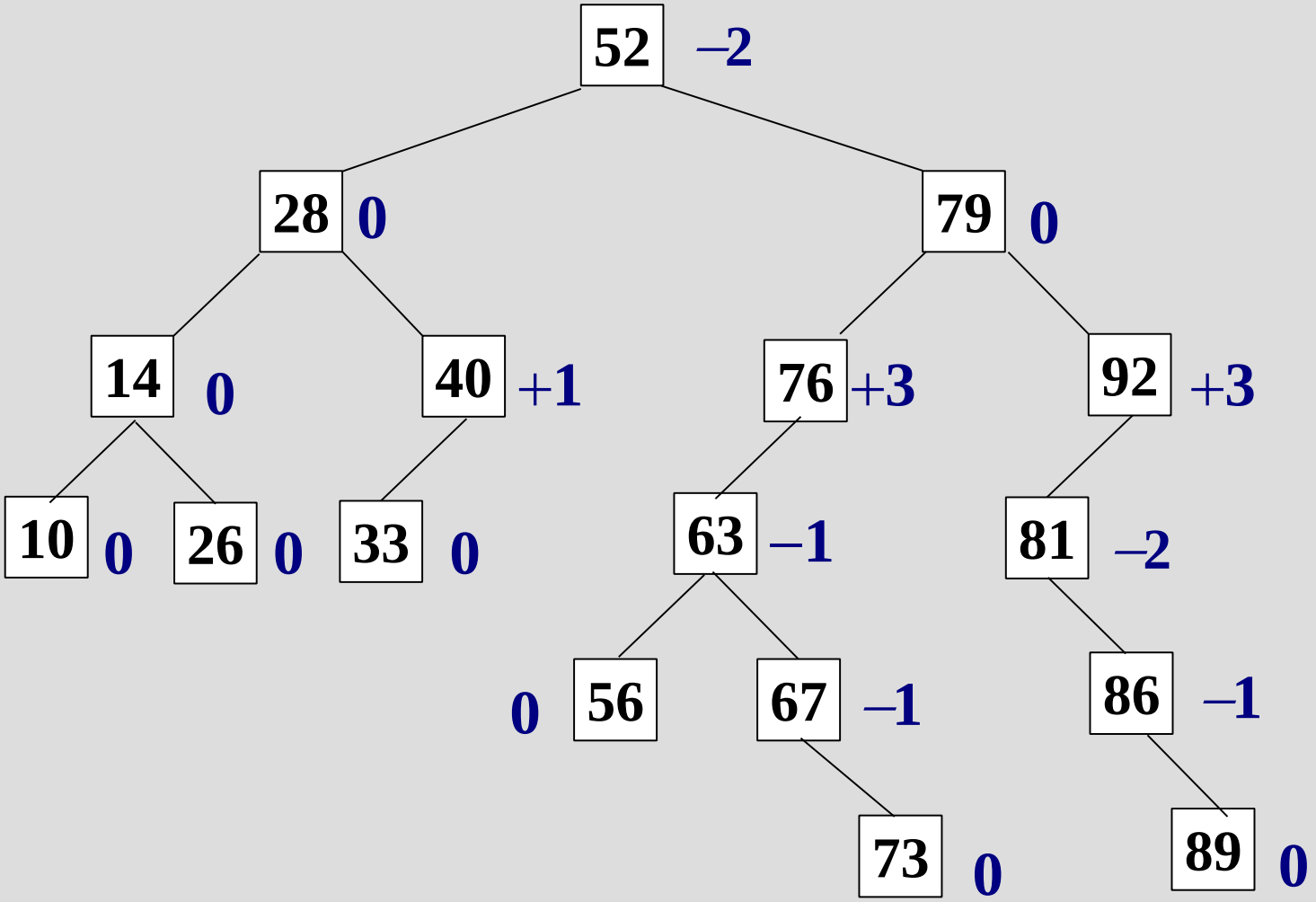
Factor de balanceo de n = altura de A – altura de B

- Si el árbol está balanceado el **factor de balanceo** sólo podrá **tener** los valores **–1**, **0** y **+1**
- El desbalanceo tras una inserción o eliminación sólo se produce porque este factor llega a **tomar** el valor **–2** o **+2**

Factores de balanceo de AVL

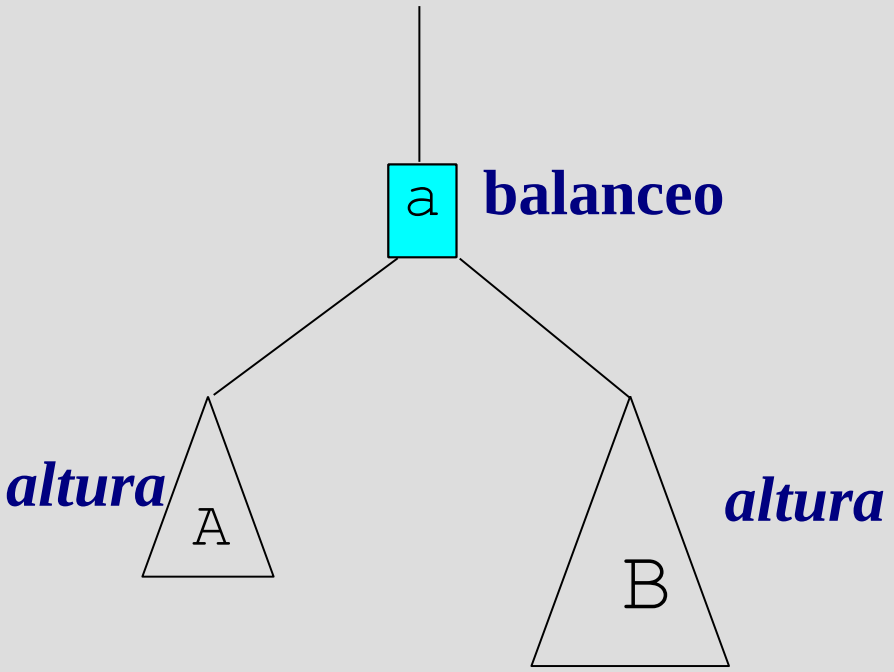


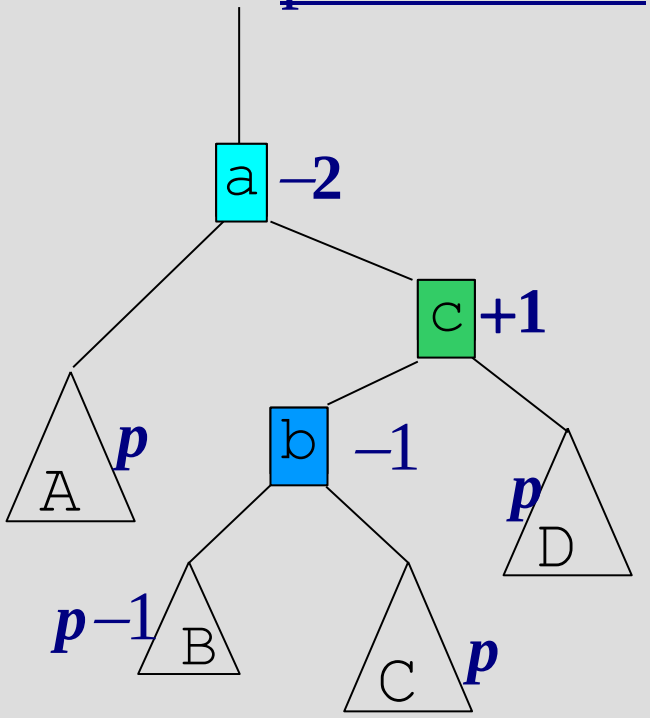
Factores de balanceo no AVL



Representando el balanceo

- Para describir las distintas situaciones y la forma de actuar, ***representamos*** los árboles con ***letras mayúsculas*** *consecutivas* según el **orden simétrico** y los nodos con ***letras minúsculas*** también *correlativas*.
- ***En las representaciones***: los **nodos** van acompañados de su factor de balanceo y los **árboles** de su altura o profundidad

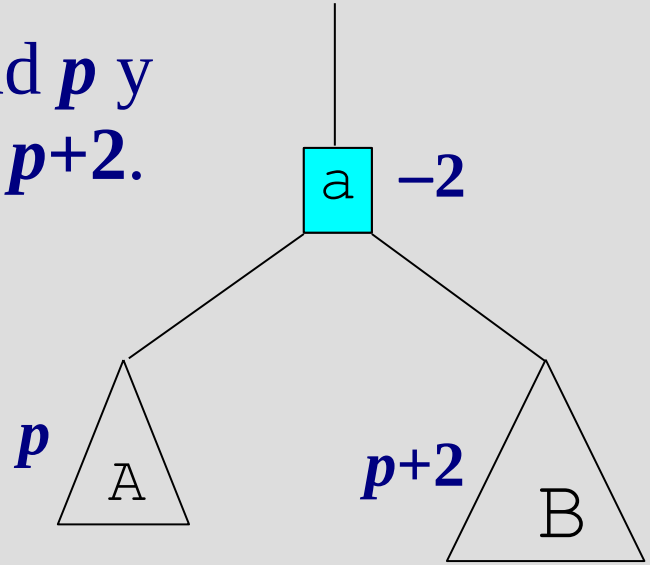




El desbalanceo

- Sea **a** el nodo donde se produce el desbalanceo con ***bal* = –2**:

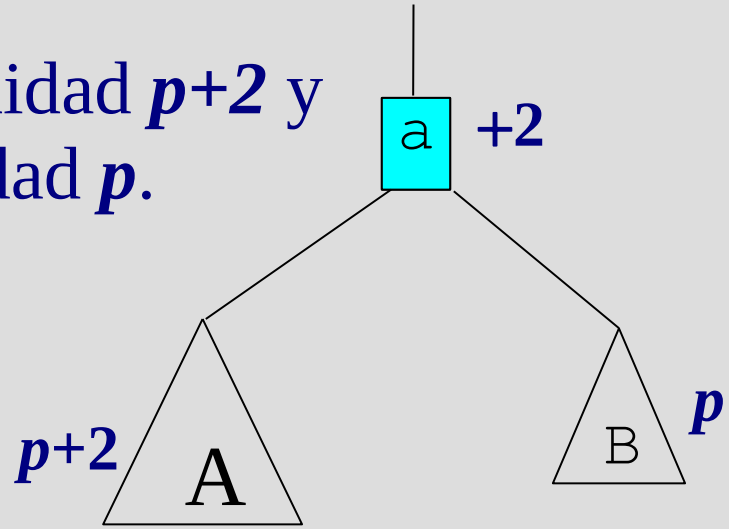
- El subárbol izquierdo **A** tiene profundidad ***p*** y
- El subárbol derecho **B** tiene profundidad ***p*+2**.



***La situación simétrica es análoga*:**

- El nodo **a** donde se produce el desbalanceo tiene ***bal* = +2**:

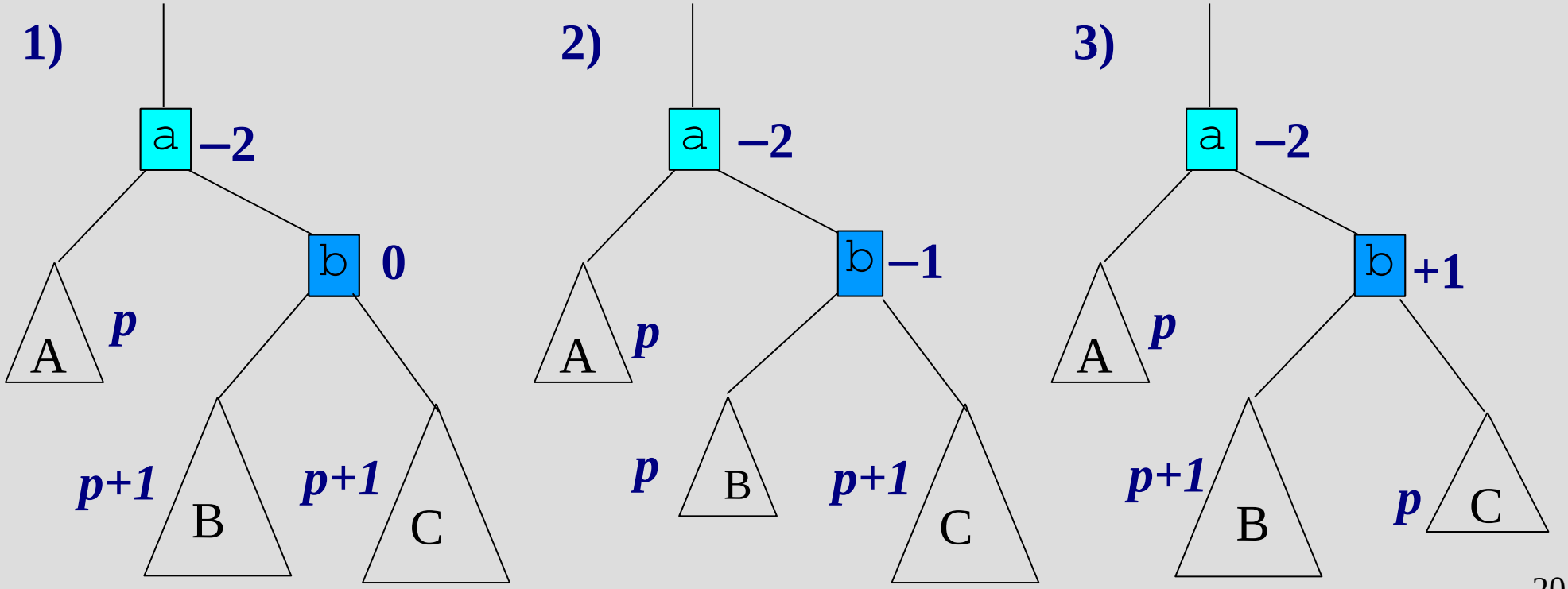
- El subárbol izquierdo **A** tiene profundidad ***p+2*** y
- El subárbol derecho **B** tiene profundidad ***p***.



Casos de desbalanceo

Si el desbalanceo de **a** es **–2**. Sea **b** su hijo derecho.
Entonces **b** tiene al menos un hijo de profundidad ***p*+1**.

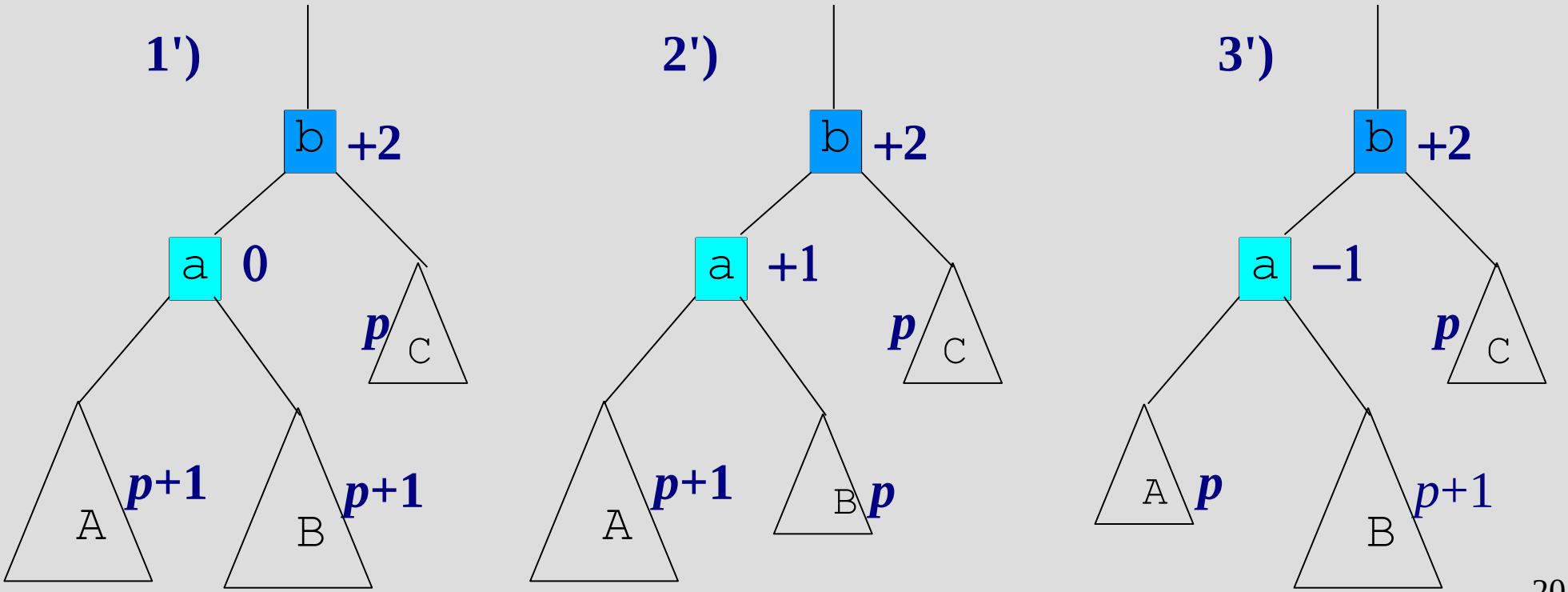
- Caso 1. Los dos hijos del nodo **b** son de profundidad ***p*+1**
- Caso 2. El hijo izquierdo tiene profundidad ***p*** y el hijo derecho ***p*+1**
- Caso 3. El hijo izquierdo tiene profundidad ***p*+1** y el hijo derecho ***p***.



Casos simétricos

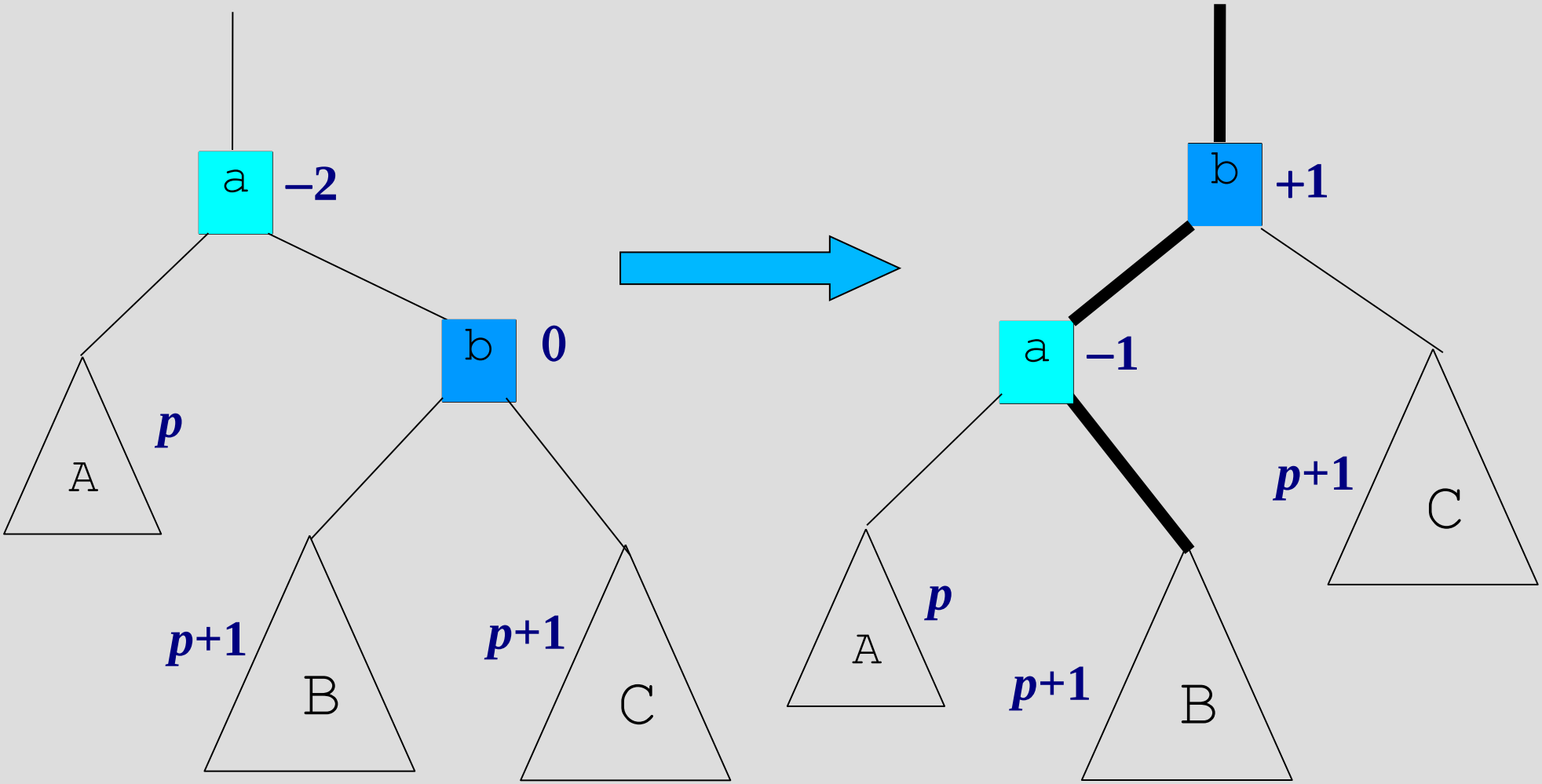
Si el desbalanceo es **+2**. Sea **b** este nodo y **a** su hijo izquierdo. Entonces **a** tiene al menos un hijo de profundidad ***p*+1**.

- Caso 1': Los dos hijos de **a** son de profundidad ***p*+1**
- Caso 2': El hijo izquierdo tiene profundidad ***p*+1** y el hijo derecho ***p***
- Caso 3': El hijo izquierdo tiene profundidad ***p*** y el hijo derecho ***p*+1**



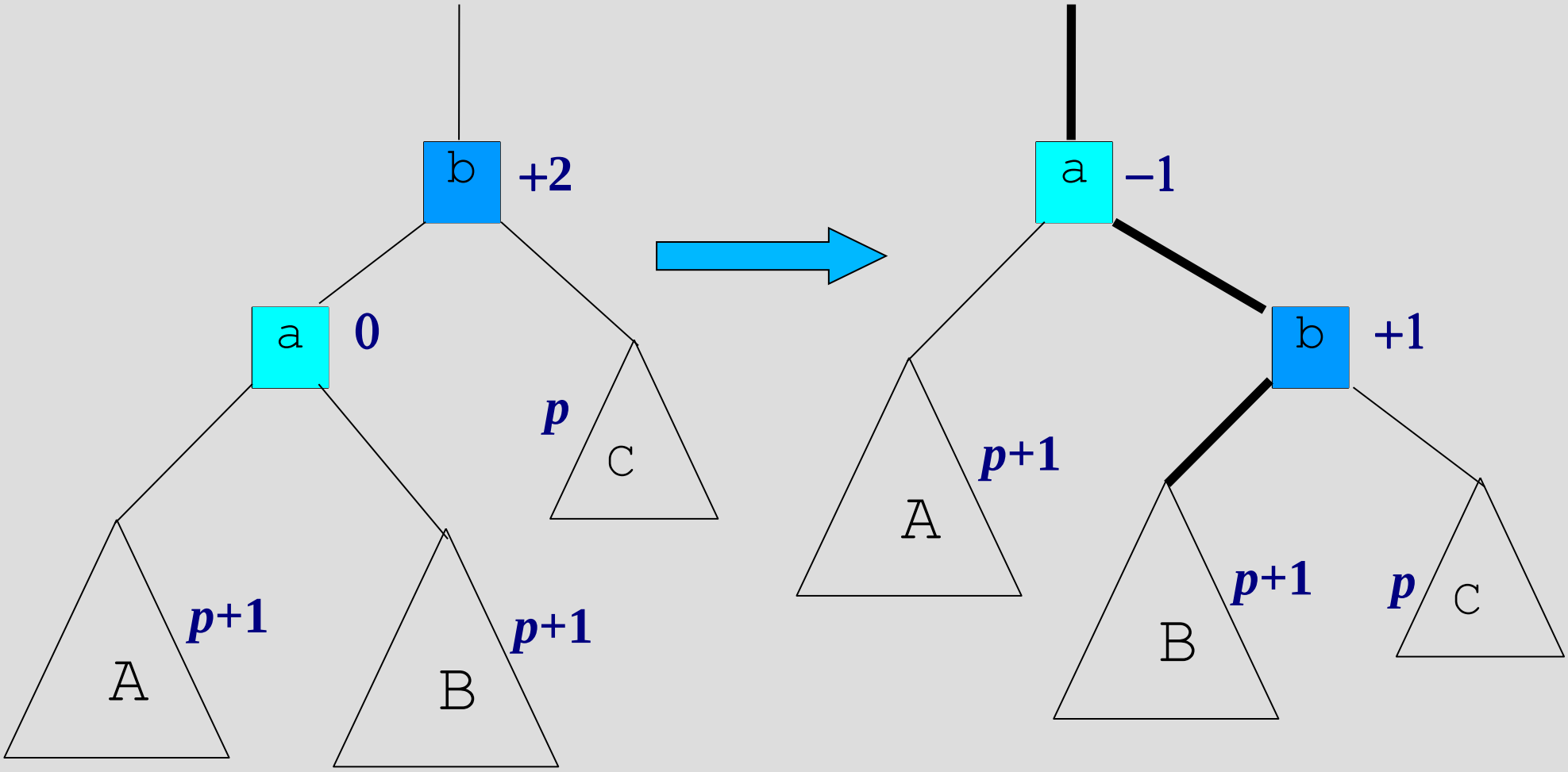
Rebalanceo del Caso 1)

Desbalanceo con ***bal* = –2**; [hijo derecho con profundidad ***p*+2**]
Hijo derecho con ***bal* = 0** [los dos hijos son de profundidad ***p*+1**]



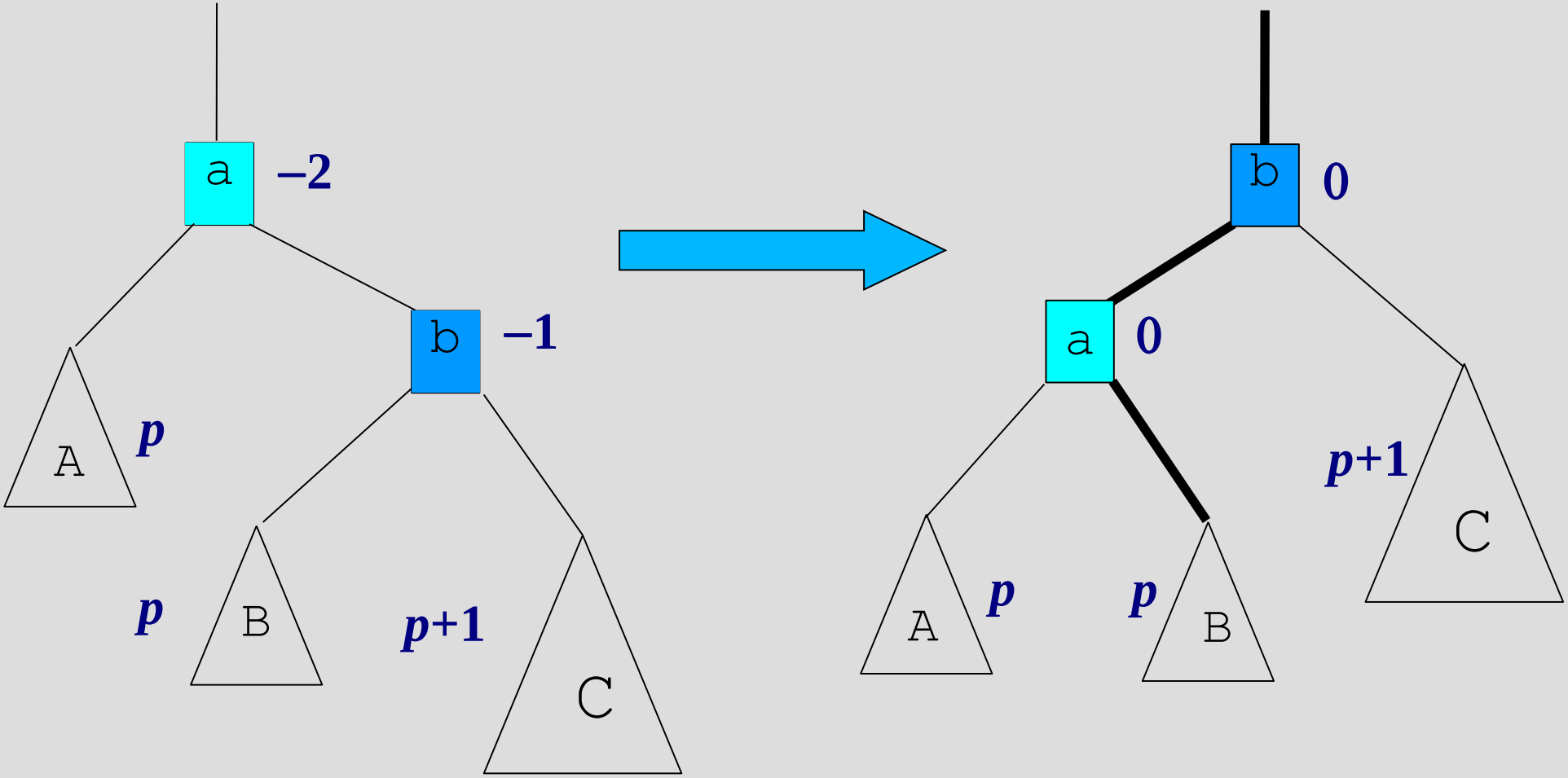
Rebalanceo del Caso 1')

Desbalanceo con ***bal* = +2**; [hijo izquierdo con profundidad ***p*+2**]
Hijo izquierdo con ***bal* = 0** [los dos hijos son de profundidad ***p*+1**]



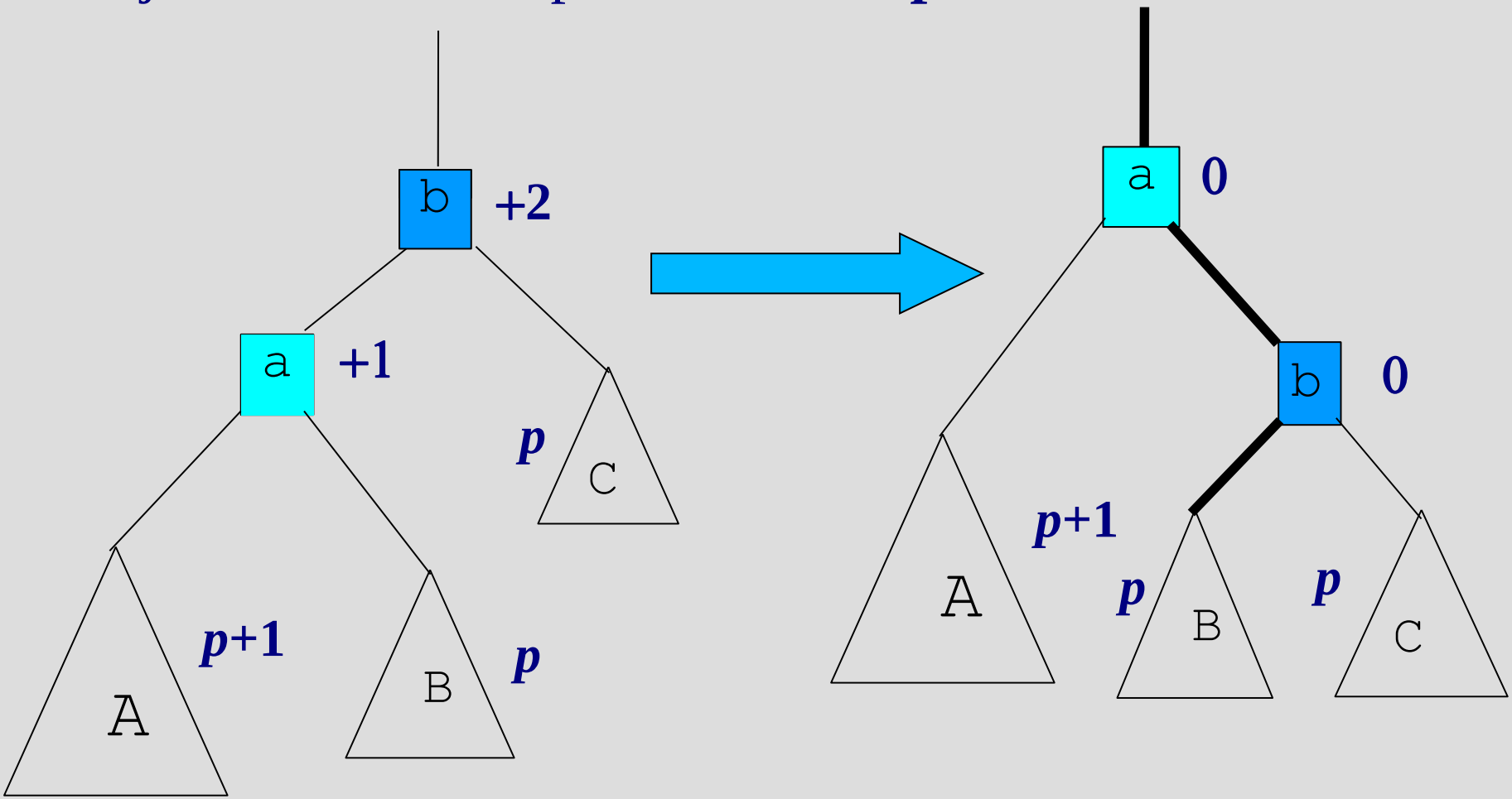
Rebalanceo del Caso 2)

Desbalanceo con ***bal* = –2**; hijo derecho con ***bal* = –1**.
su hijo izquierdo tiene profundidad ***p*** y
su hijo derecho tiene profundidad ***p*+1**



Rebalanceo del Caso 2')

Desbalanceo con ***bal* = +2**; hijo izquierdo con ***bal* = +1**.
su hijo izquierdo tiene profundidad ***p*+1** y
su hijo derecho tiene profundidad ***p***



Rebalanceo del Caso 3)

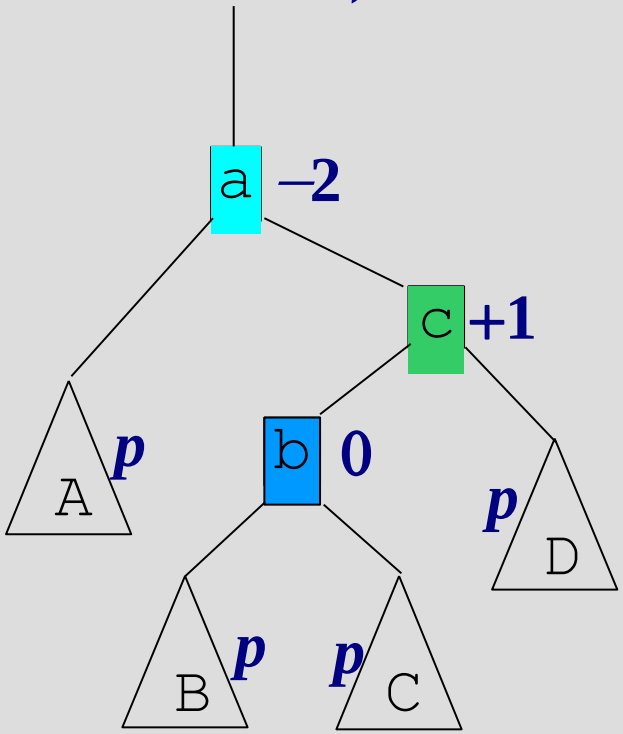
Desbalanceo con ***bal* = –2**; hijo derecho con ***bal* = +1**.

[hijo derecho de profundidad ***p*** e hijo izquierdo de ***p*+1**]

Este hijo izquierdo tiene ***bal* = 0 , –1, o +1**:

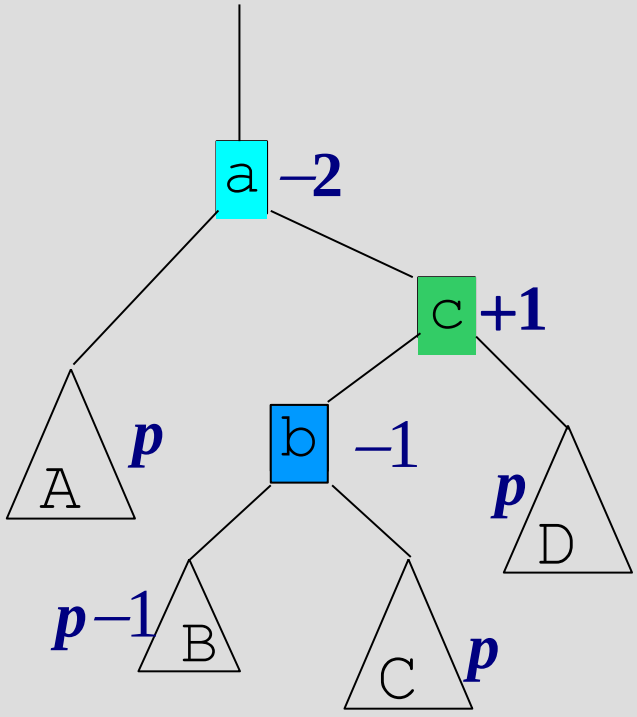
***bal* = 0**

Subcaso 3a)



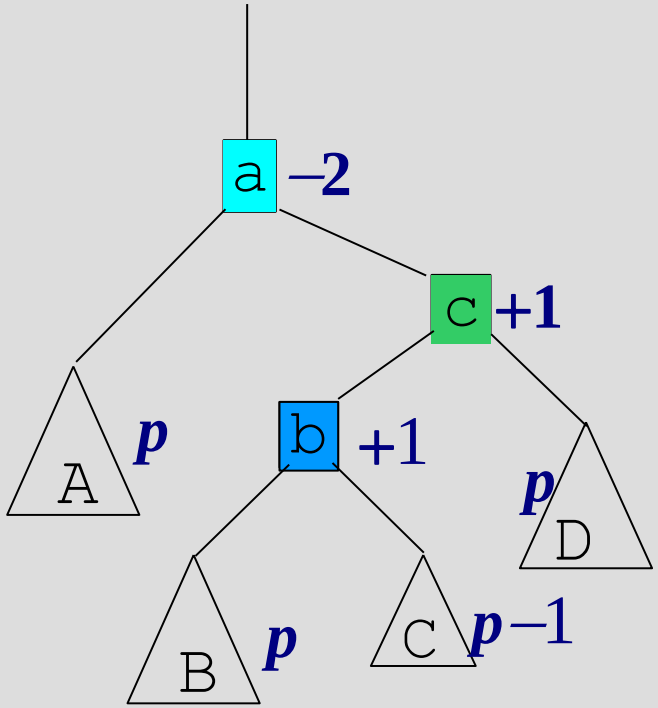
***bal* = –1**

Subcaso 3b)



***bal* = +1**

Subcaso 3c)



Rebalanceo del Subcaso 3a)

Desbalanceo en **a** con ***bal* = –2**,

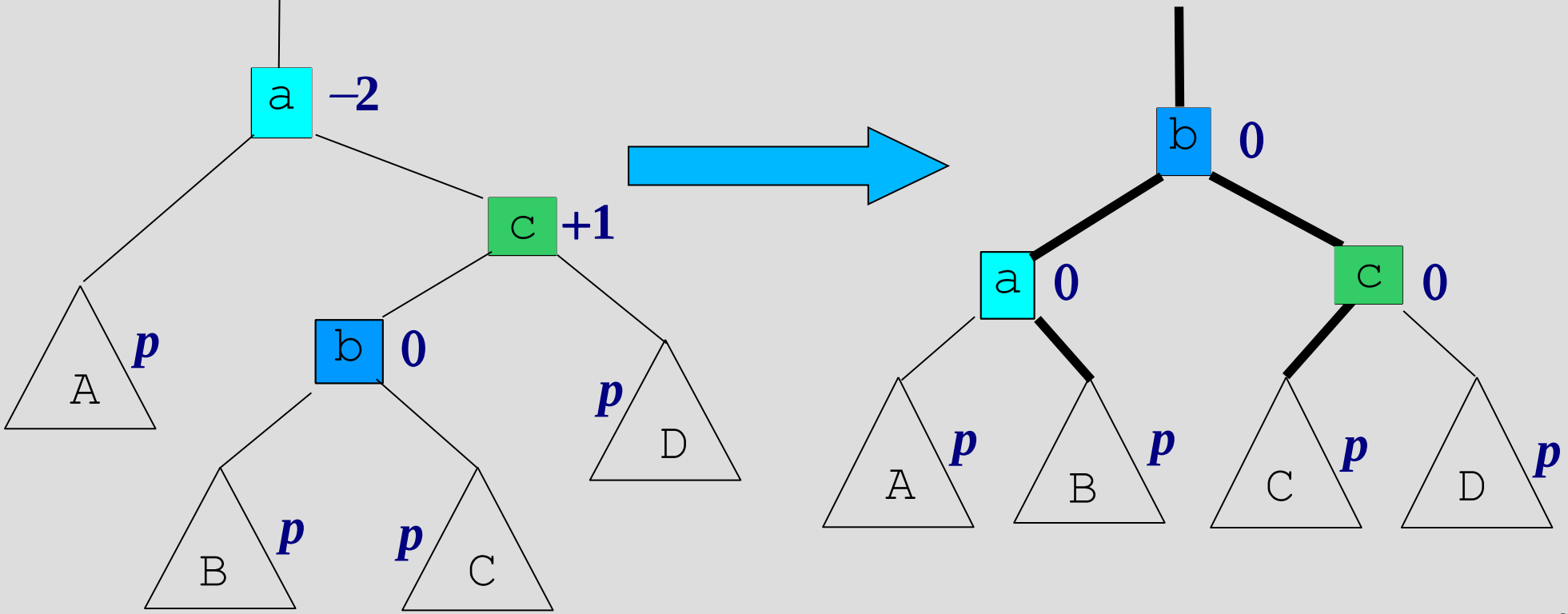
[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p*+2**]

Su hijo derecho **c** tiene ***bal* = +1**,

[hijo izquierdo con profundidad ***p*+1** e hijo derecho con ***p***]

Su hijo izquierdo **b** tiene ***bal* = 0**.

[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p***]



Rebalanceo del Subcaso 3b)

Desbalanceo en **a** con ***bal* = –2**,

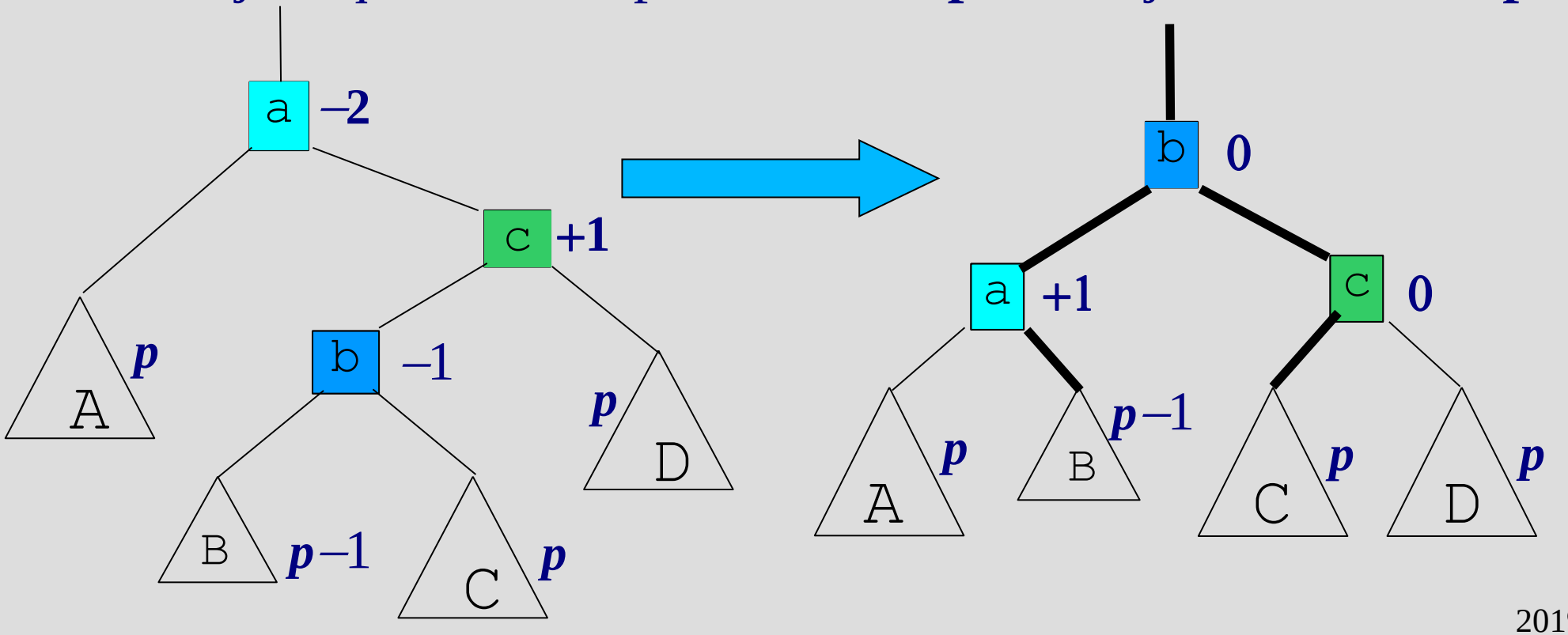
[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p*+2**]

Su hijo derecho **c** tiene ***bal* = +1**,

[hijo izquierdo con profundidad ***p*+1** e hijo derecho con ***p***]

Su hijo izquierdo **b** tiene ***bal* = –1.**

[hijo izquierdo con profundidad ***p*–1** e hijo derecho con ***p***]



Rebalanceo del Subcaso 3c)

Desbalanceo en **a** con ***bal* = –2**,

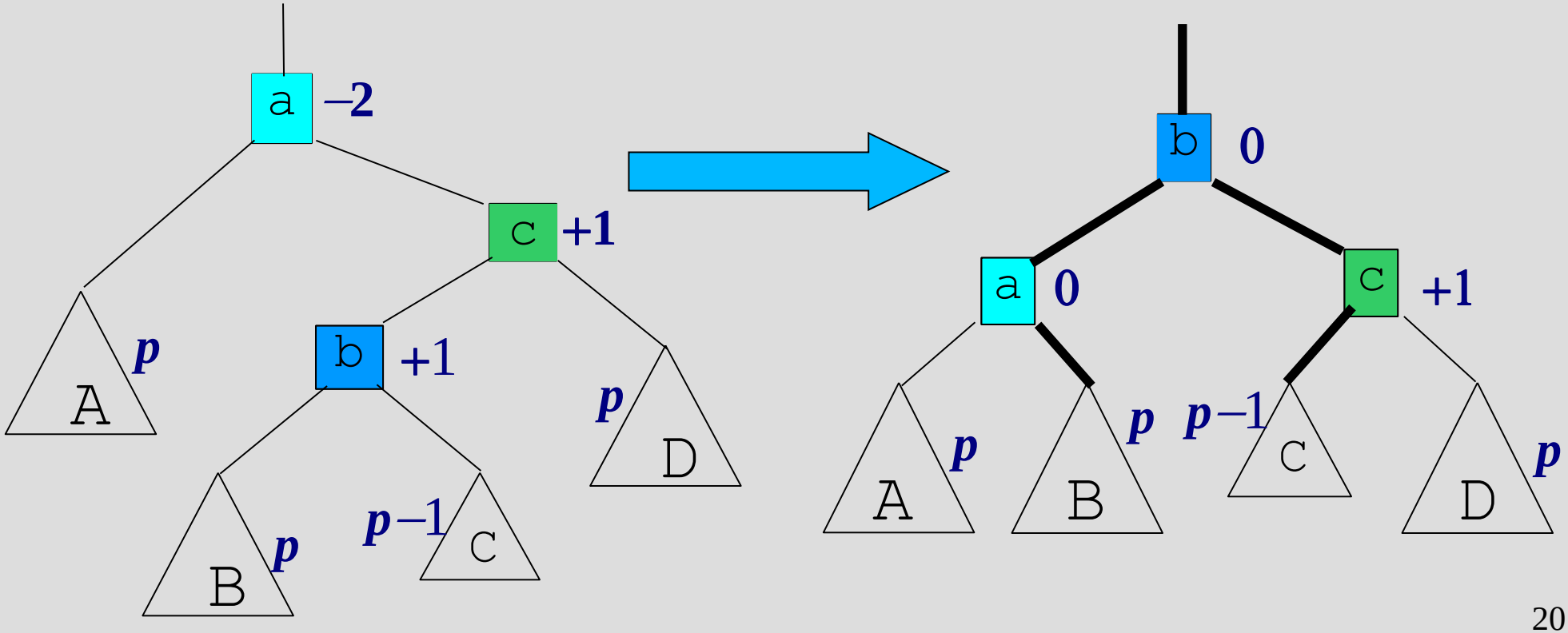
[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p*+2**]

Su hijo derecho **c** tiene ***bal* = +1**,

[hijo izquierdo con profundidad ***p*+1** e hijo derecho con ***p***]

Su hijo izquierdo **b** tiene ***bal* = +1.**

[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p*–1**]



Rebalanceo del Subcaso 3a')

Desbalanceo en **c** con ***bal* = +2**,

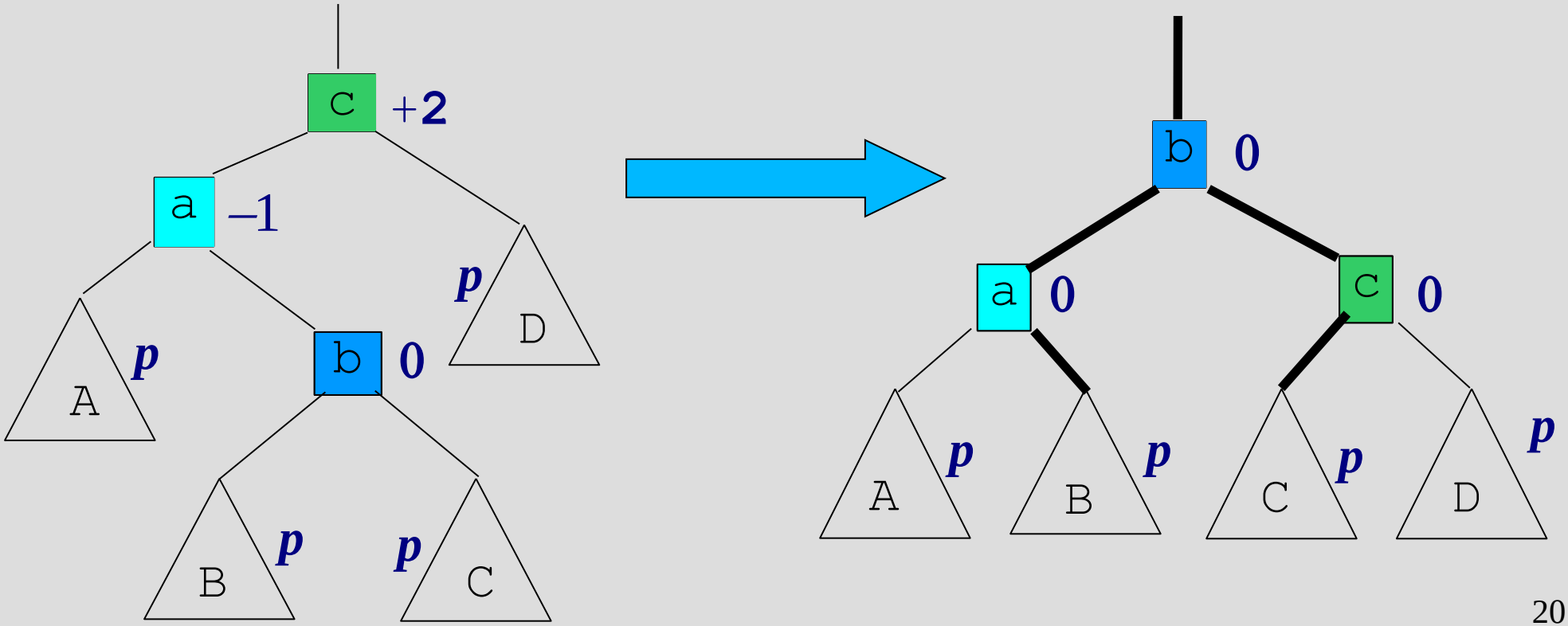
[hijo izquierdo con profundidad ***p*+2** e hijo derecho con ***p***]

Su hijo izquierdo **a** tiene ***bal* = –1**,

[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p*+1**]

Su hijo derecho **b** tiene ***bal* = 0**.

[hijo izquierdo con profundidad ***p*** e hijo derecho con ***p***]



Operaciones de rebalanceo

Las operaciones de rebalanceo se expresan de manera algebraica:
representando los árboles con su profundidad como su subíndice y
expresando por ***T*(*I*, *r*, *D*)** el árbol
con raíz ***r***, subárboles izquierdo ***I*** y derecho ***D***.

1) $T(A_p, a, T(B_{p+1}, b, C_{p+1})) \Rightarrow T(T(A_p, a, B_{p+1}), b, C_{p+1})$

1') $T(T(A_{p+1}, a, B_{p+1}), b, C_p) \Rightarrow T(A_{p+1}, a, T(B_{p+1}, b, C_p))$

2) $T(A_p, a, T(B_p, b, C_{p+1})) \Rightarrow T(T(A_p, a, B_p), b, C_{p+1})$

2') $T(T(A_{p+1}, a, B_p), b, C_p) \Rightarrow T(A_{p+1}, a, T(B_p, b, C_p))$

3) $T(A_p, a, T(T(B_{p/p-1}, b, C_{p/p-1}), c, D_p)) \Rightarrow$
$T(T(A_p, a, B_{p/p-1}), b, T(C_{p/p-1}, c, D_p))$

3') $T(T(A_p, a, T(B_{p/p-1}, b, C_{p/p-1})), c, D_p) \Rightarrow$
$T(T(A_p, a, B_{p/p-1}), b, T(C_{p/p-1}, c, D_p))$

Operaciones de rebalanceo (cont.)

El detalle, excluyendo el símbolo T, de los tres subcasos del caso 3) y de su simétrico 3')

3) $(A_p, a, ((B_{p/p-1}, b, C_{p/p-1}), c, D_p)) \Rightarrow ((A_p, a, B_{p/p-1}), b, (C_{p/p-1}, c, D_p))$

a) $(A_p, a, ((B_p, b, C_p), c, D_p)) \Rightarrow ((A_p, a, B_p), b, (C_p, c, D_p))$

b) $(A_p, a, ((B_{p-1}, b, C_p), c, D_p)) \Rightarrow ((A_p, a, B_{p-1}), b, (C_p, c, D_p))$

c) $(A_p, a, ((B_p, b, C_{p-1}), c, D_p)) \Rightarrow ((A_p, a, B_p), b, (C_{p-1}, c, D_p))$

3') $((A_p, a, (B_{p/p-1}, b, C_{p/p-1})), c, D_p) \Rightarrow ((A_p, a, B_{p/p-1}), b, (C_{p/p-1}, c, D_p))$

a) $((A_p, a, (B_p, b, C_p)), c, D_p) \Rightarrow ((A_p, a, B_p), b, (C_p, c, D_p))$

b) $((A_p, a, (B_{p-1}, b, C_p)), c, D_p) \Rightarrow ((A_p, a, B_{p-1}), b, (C_p, c, D_p))$

c) $((A_p, a, (B_p, b, C_{p-1})), c, D_p) \Rightarrow ((A_p, a, B_p), b, (C_{p-1}, c, D_p))$

Rebalanceo tras inserción

- Tras la inserción sólo puede producirse desbalanceo en los nodos recorridos por el proceso de búsqueda de la posición de inserción.
- El desbalanceo se presenta si aumenta la altura del subárbol por el que continua la búsqueda y ya era más alto que el otro subárbol.
- El balanceo se restablece en esos nodos en sentido inverso al recorrido de la búsqueda mediante la recursividad de la operación.
- Sólo si crece la altura de la rama tras la inserción y rebalanceo será necesario plantear el rebalanceo en los nodos antecesores.
- Para identificar las distintas operaciones de rebalanceo tras una inserción, en la que aumenta de profundidad una de las ramas,
  - llamamos ***n*** al nodo donde se produce el desbalanceo y
  - ***n'*** y ***n''*** a los siguientes nodos en el recorrido de búsqueda.
- Se pueden presentar 4 situaciones según de qué lado sean ***n'*** hijo de ***n*** y ***n''*** de ***n'***.

Casos de rebalanceo tras inserción

Sea ***n*** donde se produce el desbalanceo y ***n'*** y ***n''*** los nodos siguientes en el recorrido de búsqueda. Se pueden presentar 4 situaciones:

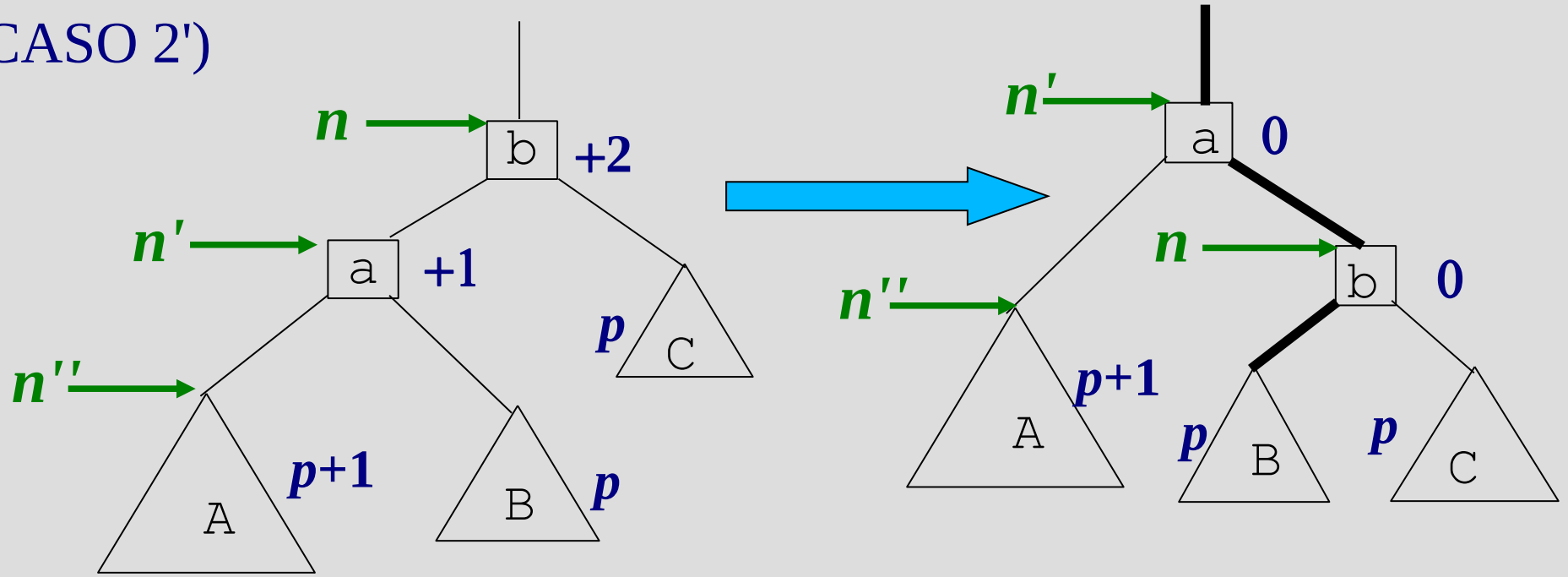
1. **Izquierda-Izquierda**.
   n' es hijo izquierdo de ***n*** y ***n''*** es hijo izquierdo de ***n'***.
   Si se presenta desbalanceo en ***n*** es ***bal*(*n*) = +2** y ***bal*(*n'*) = +1**
   La situación es la descrita en el caso 2' de desbalanceo.
2. **Derecha-Derecha**.
   n' hijo derecho de ***n*** y ***n''*** es hijo derecho de ***n'***.
   Si se presenta desbalanceo en ***n*** es ***bal*(*n*) = –2** y ***bal*(*n'*) = –1**
   La situación es la descrita en el caso 2 de desbalanceo.
3. **Izquierda-Derecha**.
   n' es hijo izquierdo de ***n*** y ***n''*** es hijo derecho de ***n'***.
   Si se presenta desbalanceo en ***n*** es ***bal*(*n*) = +2** y ***bal*(*n'*) = –1**
   La situación es la descrita en el caso 3' de desbalanceo.
4. **Derecha-Izquierda**.
   n' es hijo derecho de ***n*** y ***n''*** es hijo izquierdo de ***n'***.
   Si se presenta desbalanceo en ***n*** es ***bal*(*n*) = –2** y ***bal*(*n'*) = +1**
   La situación es la descrita en el caso 3 de desbalanceo.

1. Rotación Izquierda-Izquierda

El nodo ***n'*** es hijo izquierdo de ***n*** y el nodo ***n''*** es hijo izquierdo de ***n'***.

Si se presenta un desbalanceo en ***n*** es ***bal*(*n*) = +2** y ***bal*(*n'*) = +1**.

CASO 2')



- ***i*(*n*) ← *d*(*n'*), *d*(*n'*) ← *n*, *n* ← *n'*.**
- ***bal*(*n*) ← 0** y **bal(*n'*) ← 0.**

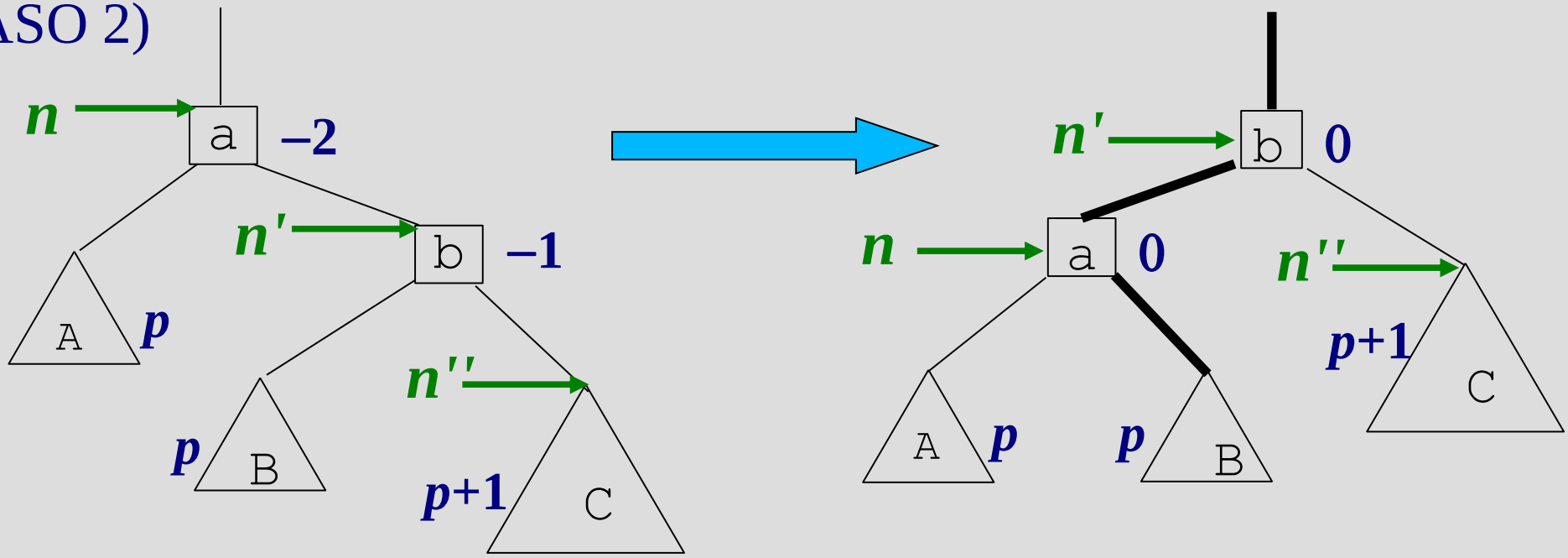
La rama queda con profundidad *p*+2; NO crece

2. Rotación Derecha-Derecha

El nodo ***n'*** es hijo derecho de ***n*** y el nodo ***n''*** es hijo derecho de ***n'***.

Si se presenta un desbalanceo en ***n*** es ***bal*(*n*) = –2** y ***bal*(*n'*) = –1**.

CASO 2)



- ***d*(*n*) ← *i*(*n'*), *i*(*n'*) ← *n*, *n* ← *n'*.**
- ***bal*(*n*) ← 0 y *bal*(*n'*) ← 0.**

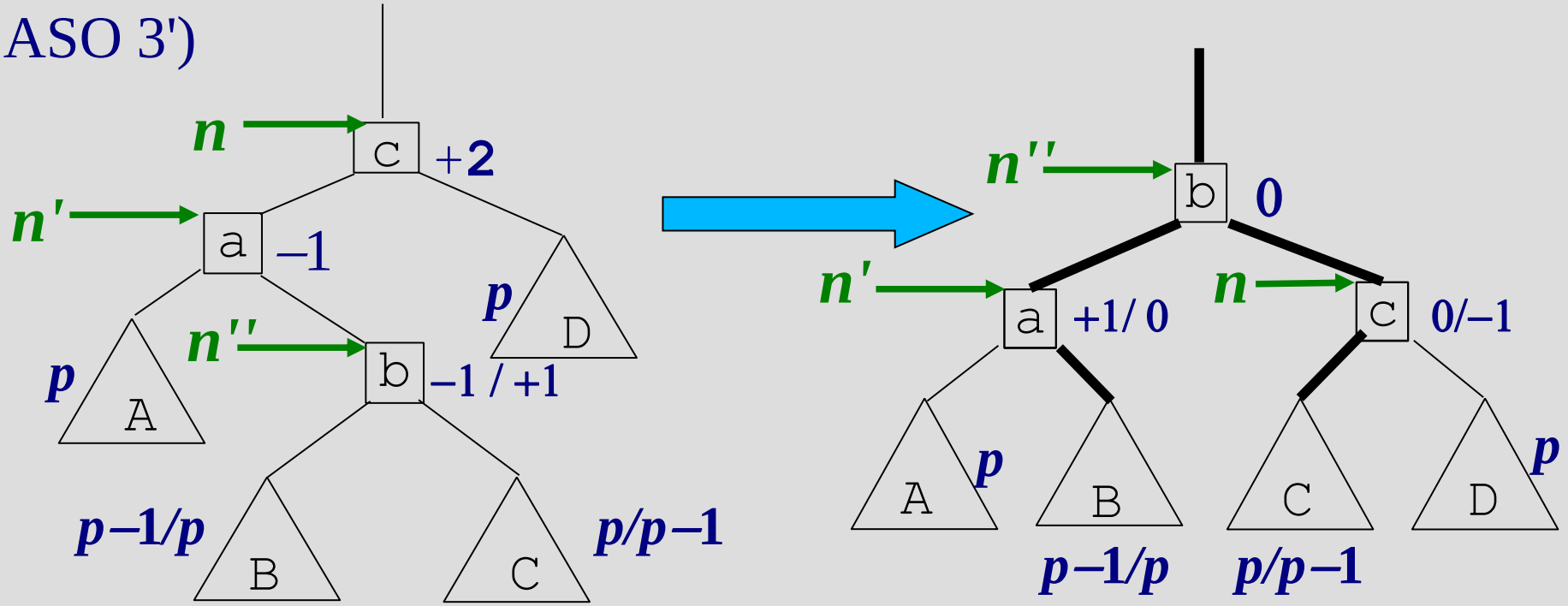
La rama queda con profundidad *p*+2; NO crece

3. Rotación Izquierda-Derecha

El nodo ***n'*** es hijo izquierdo de ***n*** y el nodo ***n''*** es hijo derecho de ***n'***.

Si se presenta un desbalanceo en ***n*** es ***bal*(*n*) = +2** y ***bal*(*n'*) = –1**.

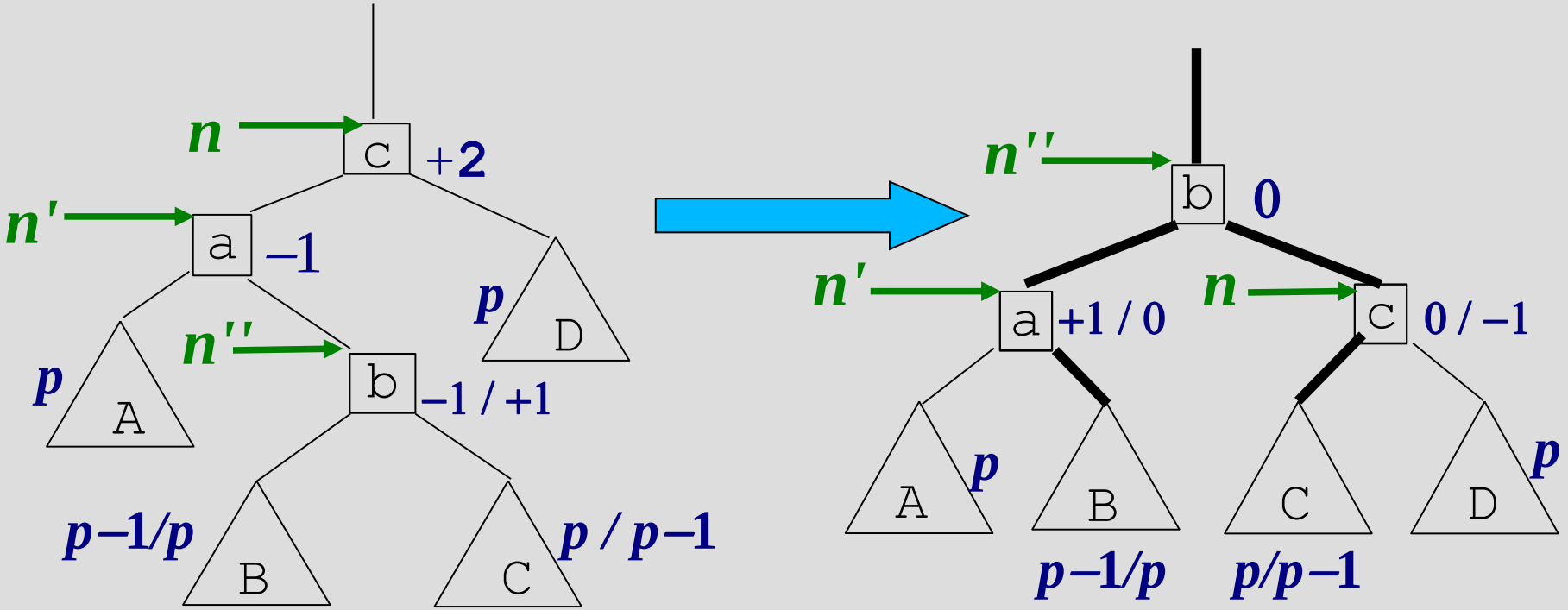
CASO 3')



- ***d*(*n'*) ← *i*(*n''*), *i*(*n*) ← *d*(*n''*), *i*(*n''*) ← *n'*, *d*(*n''*) ← *n*, *n* ← *n''*.**
- ***bal*(*n''*) ← 0 , *bal*(*n'*) ← +1 / 0 y *bal*(*n'*) ← 0 / –1.**

La rama queda con profundidad *p*+2; NO crece

3. Rotación Izquierda-Derecha (II)



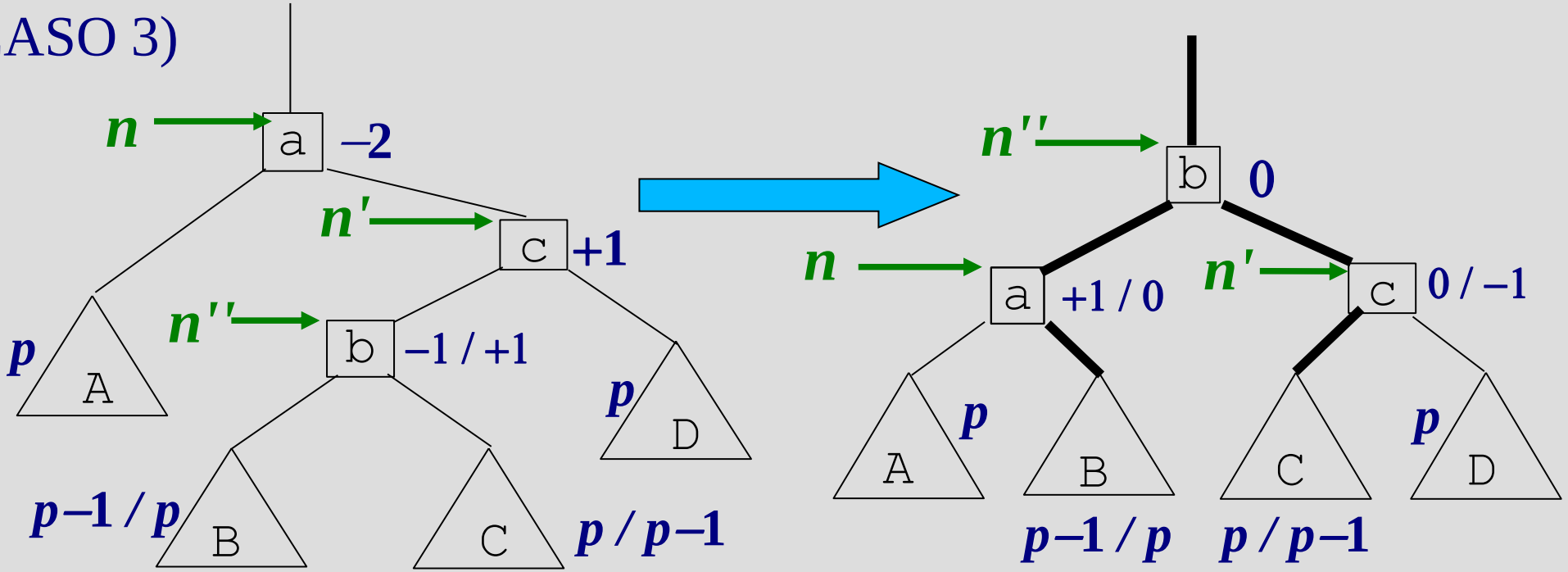
- $d(n') \leftarrow i(n'')$, $i(n) \leftarrow d(n'')$, $i(n'') \leftarrow n'$, $d(n'') \leftarrow n$, $n \leftarrow n''$.
  - Si $bal(n'') = 0$ no pudo haber desbalanceo.
  - Si $bal(n'') = -1$: $bal(n') \leftarrow +1$ y $bal(n) \leftarrow 0$
  - Si $bal(n'') = +1$: $bal(n') \leftarrow 0$ y $bal(n) \leftarrow -1$
  - $bal(n'') \leftarrow 0$

La rama queda con profundidad p+2; NO crece

4. Rotación Derecha-Izquierda

El nodo ***n'*** es hijo derecho de ***n*** y el nodo ***n''*** es hijo izquierdo de ***n'***.
Si se presenta un desbalanceo en ***n*** es ***bal*(*n*) = –2** y ***bal*(*n'*) = +1**.

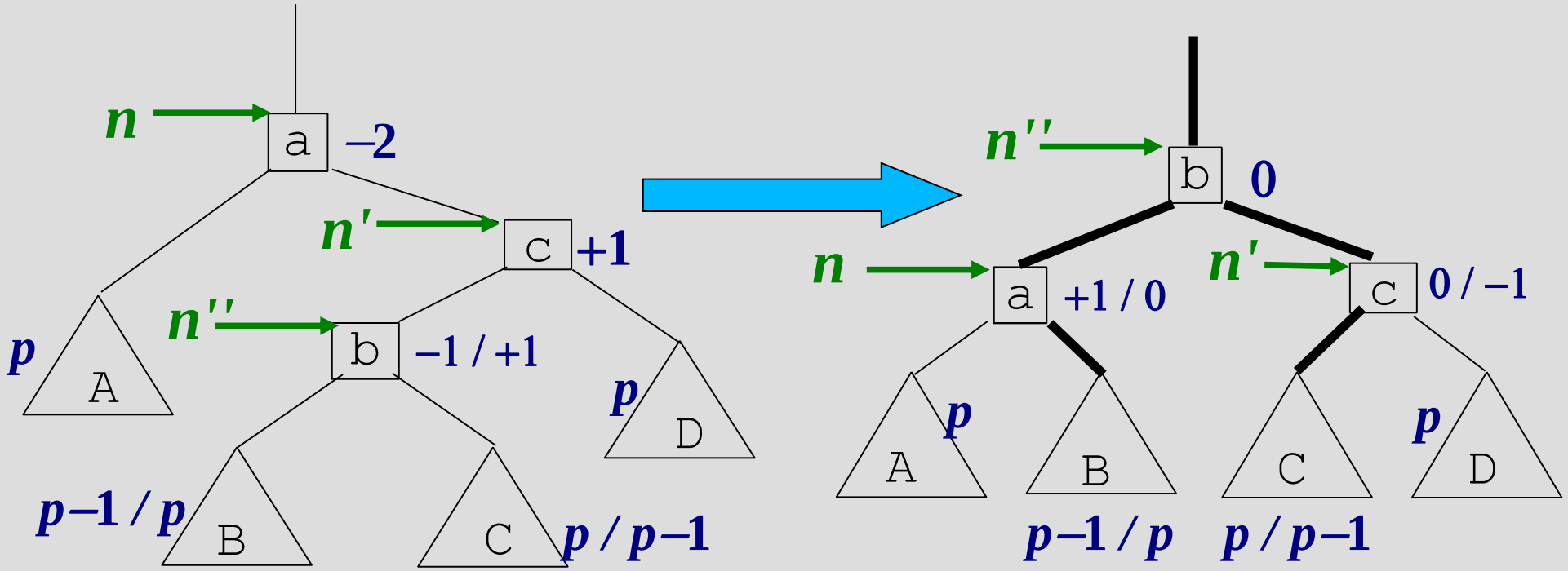
CASO 3)



- ***d*(*n*) ← *i*(*n''*), *i*(*n'*) ← *d*(*n''*), *i*(*n''*) ← *n*, *d*(*n''*) ← *n'*, *n* ← *n''*.**
- ***bal*(*n''*) ← 0 , *bal*(*n'*) ← +1 / 0 y *bal*(*n'*) ← 0 / –1.**

La rama queda con profundidad *p*+2; NO crece

4. Rotación Derecha-Izquierda (II)



- $d(n) \leftarrow i(n'')$, $i(n') \leftarrow d(n'')$, $i(n'') \leftarrow n$, $d(n'') \leftarrow n'$, $n \leftarrow n''$.
  - Si $bal(n'') = 0$ no pudo haber desbalanceo.
  - Si $bal(n'') = -1$: $bal(n') \leftarrow 0$ y $bal(n) \leftarrow +1$
  - Si $bal(n'') = +1$: $bal(n') \leftarrow -1$ y $bal(n) \leftarrow 0$
  - $bal(n'') \leftarrow 0$

La rama queda con profundidad p+2; NO crece

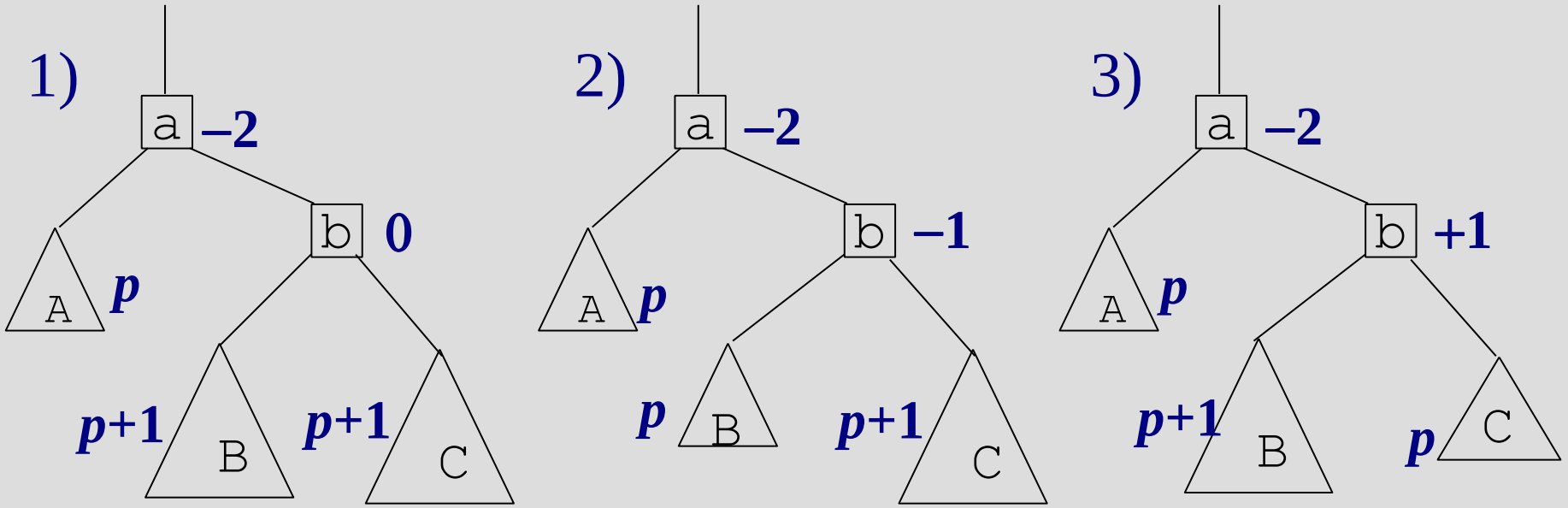
Rebalanceo tras eliminación

- La eliminación de un nodo en un árbol de búsqueda supone la desaparición de un nodo hoja o de uno con un sólo hijo modificando la altura de la rama correspondiente.
- Por tanto, se puede producir un desbalanceo en el padre del nodo que desparece si la disminución de la profundidad de una de sus ramas hace el factor de balanceo alcance los valores +2 o –2.
- En ese caso, hay que aplicar una de las operaciones de rebalanceo ya consideradas y, si la altura de la rama resultante disminuye, aplicar las operaciones de balanceo recursivamente regresando por el camino de la búsqueda.
- Por tanto, hay que considerar los desbalanceos que se pueden producir al disminuir la altura de un subárbol de alguna rama.

Eliminación a la izquierda

Si decrece la altura del subárbol izquierdo de una rama:

- Si el balanceo de la raíz es +1 pasa a ser 0 y la altura decrece
- Si el balanceo de la raíz es 0 pasa a ser –1 y la altura no decrece
- Si el balanceo de la raíz es –1, pasaría a ser –2 y se produce uno de los desbalanceos descritos como casos 1), 2) o 3) según el factor de balanceo del hijo derecho de la raíz de la rama.



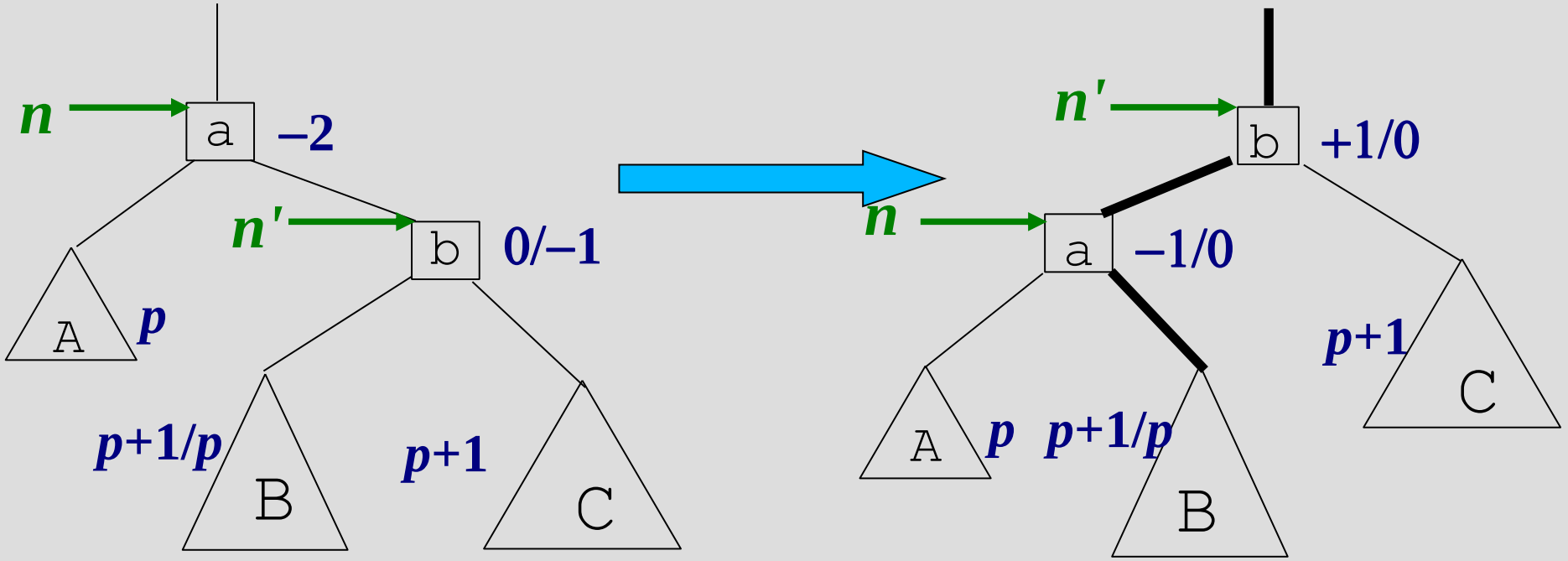
Rebalanceo por la izquierda

Si decrece la altura del subárbol izquierdo de una rama, cuya raíz tiene factor de balanceo es –1 se presenta una de las siguientes situaciones según el balanceo de su hijo derecho:

- Si el factor de balanceo de su hijo derecho es 0 (caso 1):
  - Se rebalancea con la rotación DD y la altura se mantiene.
- Si el factor de balanceo de su hijo derecho es –1 (caso 2):
  - Se rebalancea con la rotación DD y la altura decrece.
- Si el factor de balanceo de su hijo derecho es +1 (caso 3):
  - Se rebalancea con la rotación DI y la altura decrece.

Rotación Derecha-Derecha

El hijo derecho n' de n tiene $bal(n') = 0$ o $bal(n') = -1$ (Casos 1,2)

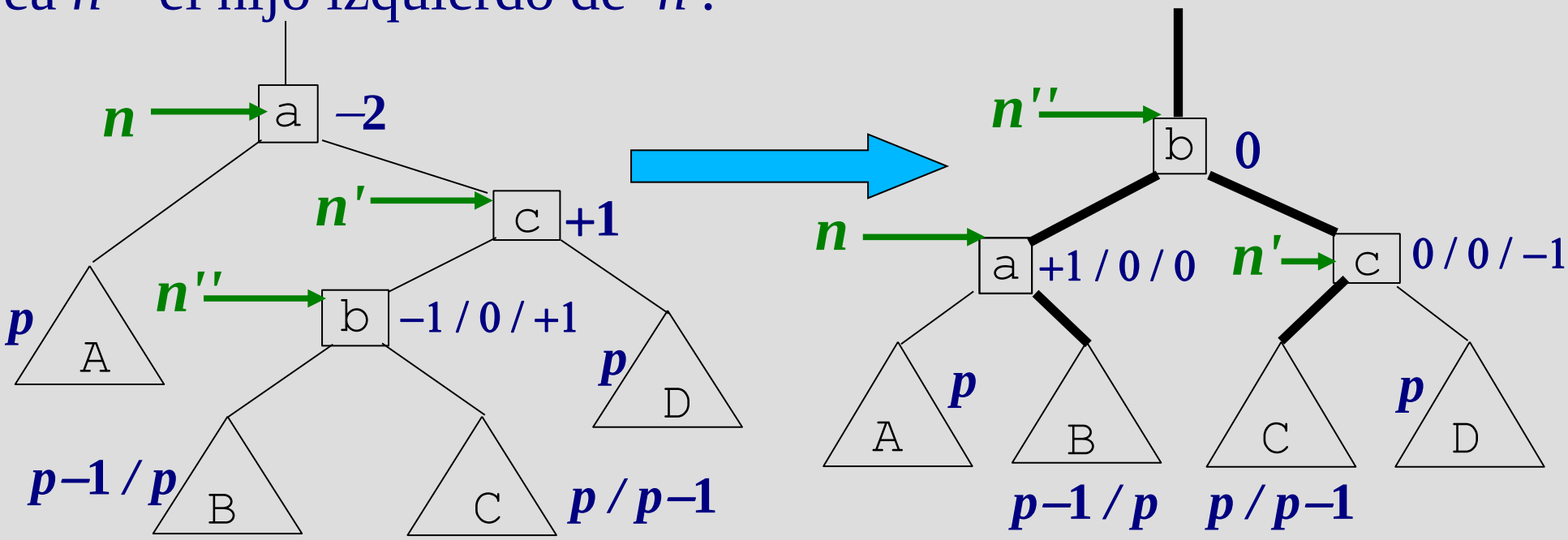


- $d(n) \leftarrow i(n')$, $i(n') \leftarrow n$, $n \leftarrow n'$.
- **Si $bal(n') = 0$ entonces $bal(n) \leftarrow -1$ y $bal(n') \leftarrow +1$.**
  La rama pasa de profundidad p+3 a p+3. **NO DECRECE**
- **Si $bal(n') = -1$ entonces $bal(n) \leftarrow 0$ y $bal(n') \leftarrow 0$.**
  La rama pasa de profundidad p+3 a p+2; **SI DECRECE**

Rotación Derecha-Izquierda

El hijo derecho n' de n tiene $bal(n') = +1$ (Caso 3)

Sea n'' el hijo izquierdo de n'.



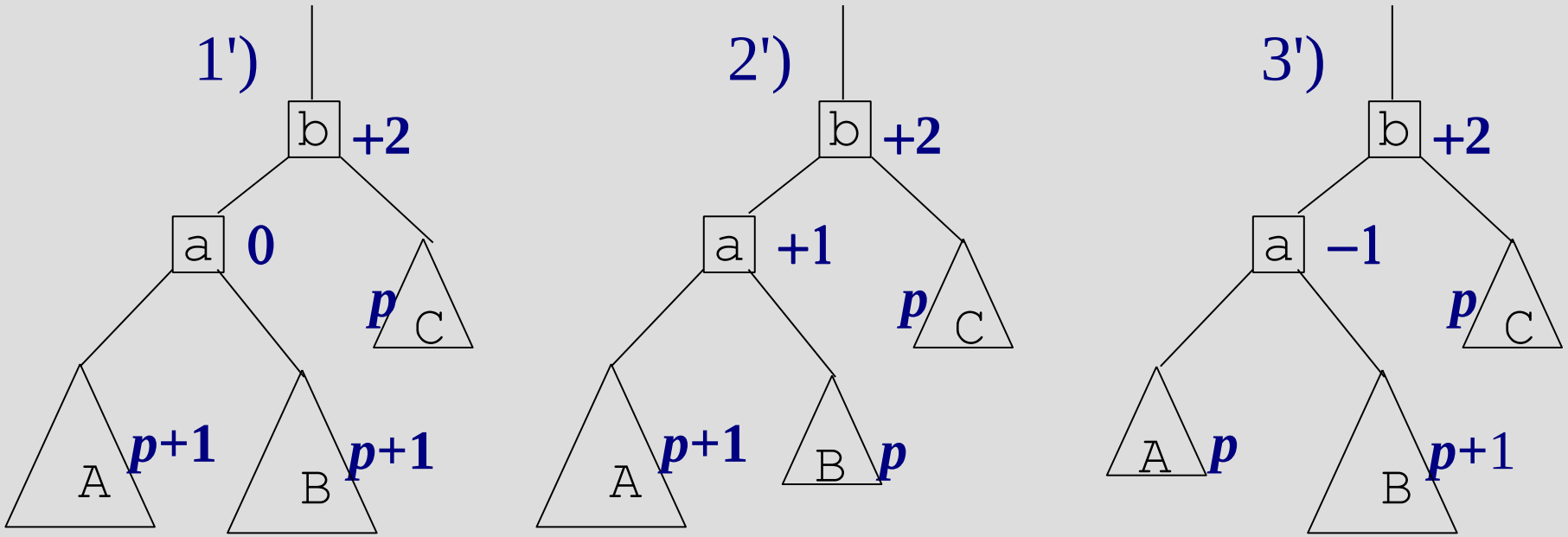
- $d(n) \leftarrow i(n'')$, $i(n') \leftarrow d(n'')$, $i(n'') \leftarrow n$, $d(n'') \leftarrow n'$, $n \leftarrow n''$.
  - Si $bal(n'') = -1$: $bal(n'') \leftarrow 0$, $bal(n') \leftarrow 0$ y $bal(n) \leftarrow +1$.
  - Si $bal(n'') = 0$: $bal(n'') \leftarrow 0$, $bal(n') \leftarrow 0$ y $bal(n) \leftarrow 0$.
  - Si $bal(n'') = +1$: $bal(n'') \leftarrow 0$, $bal(n') \leftarrow -1$ y $bal(n) \leftarrow 0$.

La rama pasa de profundidad p+3 a p+2; SI DECRECE

Eliminación a la derecha

Si decrece la altura del subárbol derecho de una rama:

- Si el balanceo de la raíz es −1 pasa a ser 0 y la altura decrece
- Si el balanceo de la raíz es 0 pasa a ser +1 y la altura no decrece
- Si el balanceo de la raíz es +1, pasaría a ser +2 y se produce uno de los desbalanceos descritos como casos 1'), 2') o 3') según el balanceo del hijo izquierdo de la raíz de la rama



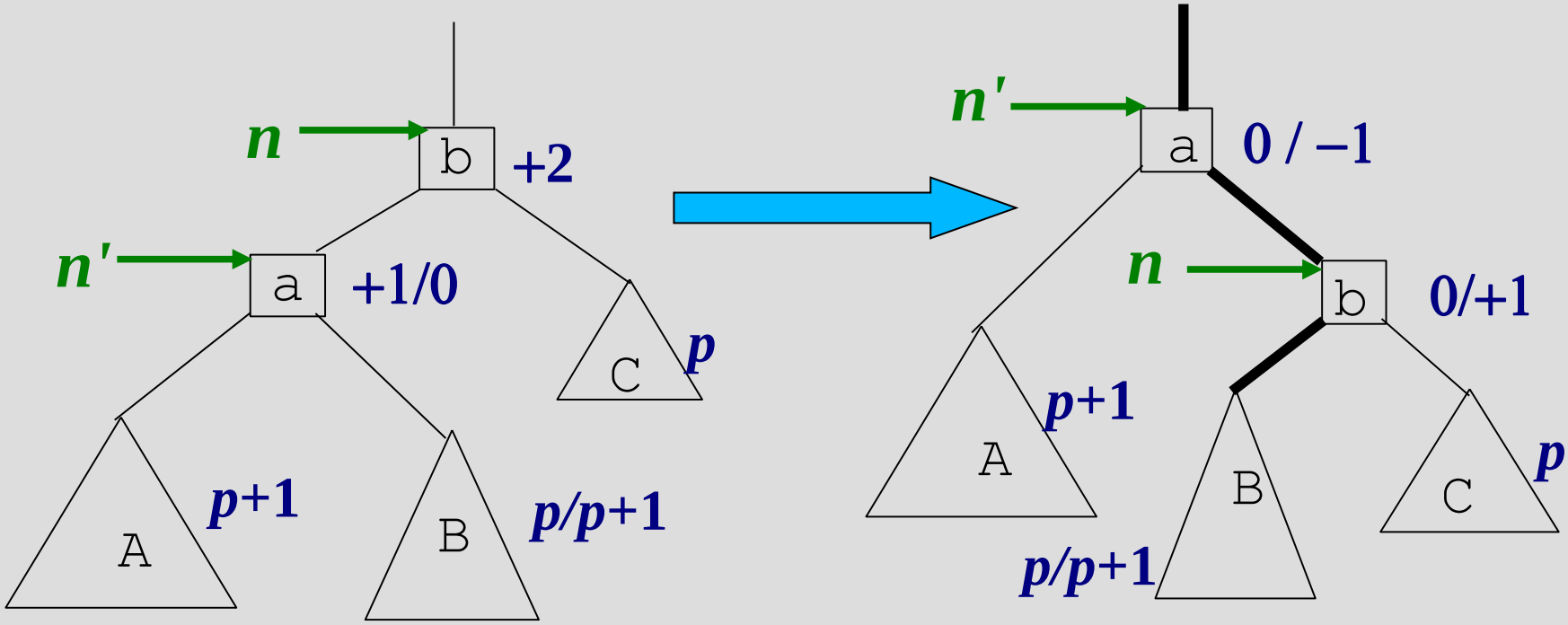
Rebalanceo por la derecha

Si decrece la altura del subárbol derecho de un nodo,
Con factor de balanceo +1

- Si el factor de balanceo de su hijo izquierdo es +1 (caso 1'):
  - Se rebalancea con la rotación II y la altura decrece
- Si el factor de balanceo de su hijo izquierdo es 0 (caso 2'):
  - Se rebalancea con la rotación II y la altura se mantiene
- Si el factor de balanceo de su hijo izquierdo es –1 (caso 3'):
  - Se rebalancea con la rotación ID y la altura decrece

Rotación Izquierda-Izquierda

El hijo izquierdo n' de n tiene $bal(n') = +1$ o $bal(n') = 0$. Casos 2') o 1')

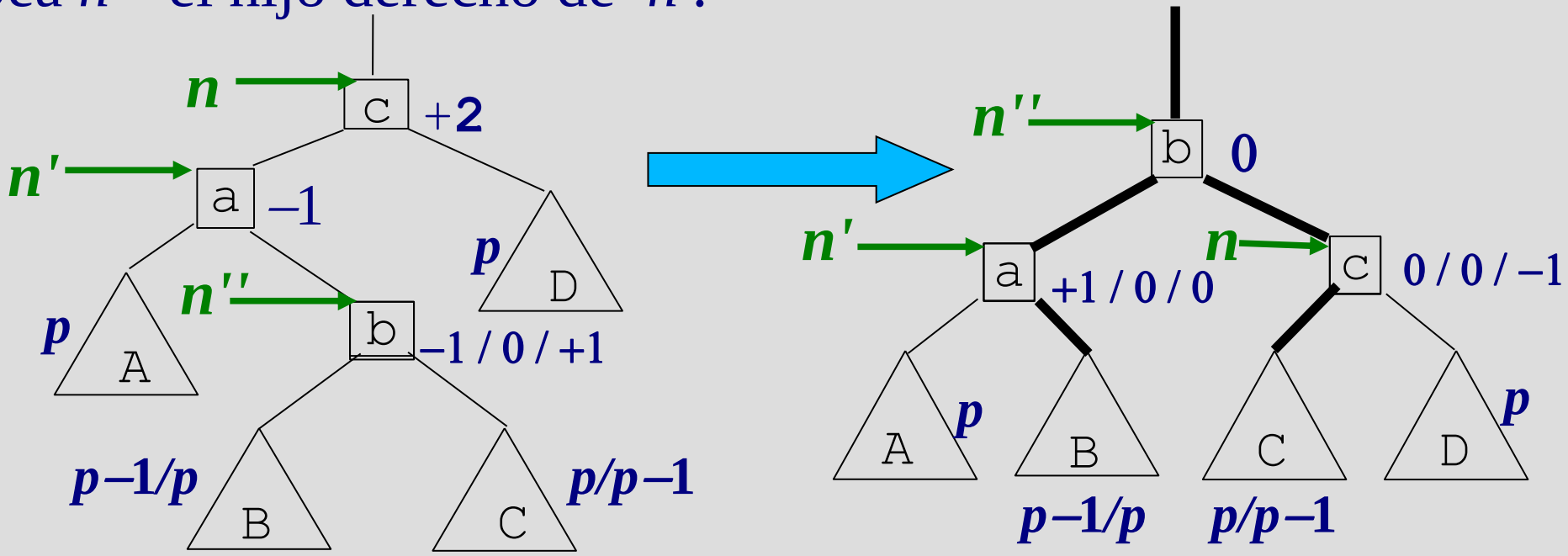


- $i(n) \leftarrow d(n')$, $d(n') \leftarrow n$, $n \leftarrow n'$.
- **Si $bal(n') = +1$ entonces $bal(n) \leftarrow 0$** y **$bal(n') \leftarrow 0$.**
  La rama pasa de profundidad p+3 a p+2; **SI decrece**
- **Si $bal(n') = 0$ entonces $bal(n) \leftarrow +1$** y **$bal(n') \leftarrow -1$.**
  La rama pasa de profundidad p+3 a p+3; **NO decrece**

Rotación Izquierda-Derecha

El hijo izquierdo n' de n tiene $bal(n') = -1$. (Caso 3')

Sea n'' el hijo derecho de n'.



- $d(n') \leftarrow i(n'')$, $i(n) \leftarrow d(n'')$, $i(n'') \leftarrow n'$, $d(n'') \leftarrow n$, $n \leftarrow n''$.
  - Si $bal(n'') = -1$: $bal(n'') \leftarrow 0$, $bal(n') \leftarrow +1$ y $bal(n) \leftarrow 0$.
  - Si $bal(n'') = 0$: $bal(n'') \leftarrow 0$, $bal(n') \leftarrow 0$ y $bal(n) \leftarrow 0$.
  - Si $bal(n'') = +1$: $bal(n'') \leftarrow 0$, $bal(n') \leftarrow 0$ y $bal(n) \leftarrow -1$.

La rama pasa de profundidad p+3 a p+2; **SI decrece**

Ejemplo de AVL