



Programación II *AVL - Árbol B*

2C 2023 TN – Ing. Elizabeth Barrera



Temas

- 👑 *ABB – costo de búsqueda*
- 👑 *AVL*
- 👑 *Rotaciones*
- 👑 *Ejemplo – Link*
- 👑 *Árbol B*
- 👑 *Inserciones – Eliminaciones*
- 👑 *Ejemplo – Link*

ABB

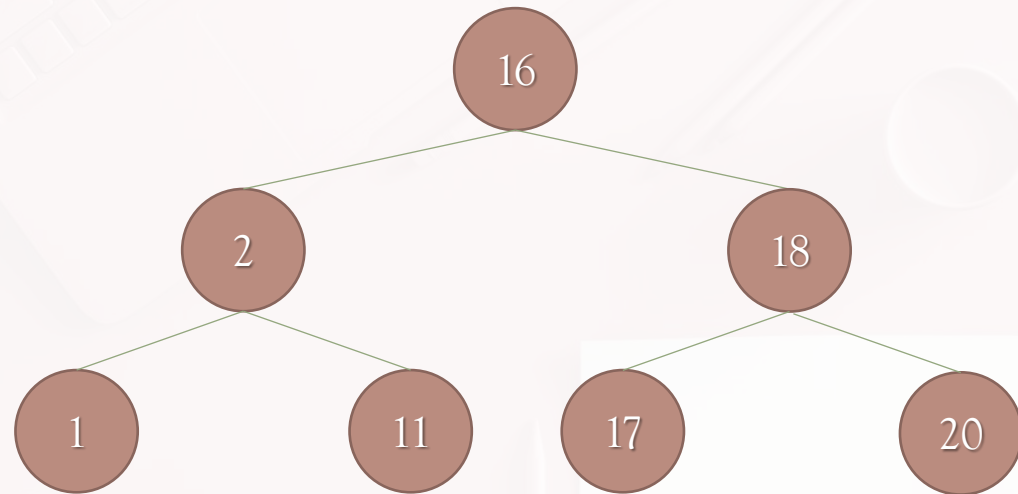
- *Son Árboles Binarios: cada nodo puede tener a lo sumo dos hijos.*
- *Sus elementos están ordenados de izquierda a derecha.*
- *No tiene elementos repetidos.*
- *Árbol balanceado: todas sus hojas están en el mismo nivel o a lo sumo hay un nivel de diferencia entre dos hojas cualesquiera.*

ABB - Costo de búsqueda

- Si se desea calcular el *costo de una búsqueda* en un ABB, debemos estudiar su *profundidad* (no se requiere recorrerlo completamente); buscando la relación entre la profundidad y la cantidad de nodos (tamaño de entrada).
- Al realizar la búsqueda, bajamos de nivel. En el peor caso, se encontrará el valor en el nivel más bajo (o no se encontrará).

ABB - Costo de búsqueda

Caso límite 1: *árbol balanceado*

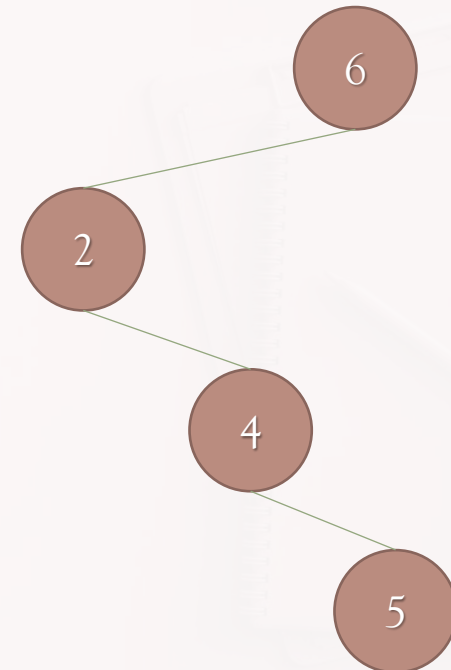


Sea “n” la cantidad de nodos, y “x” la de niveles

$$n = 2^x - 1 \rightarrow n \approx 2^x \rightarrow x \approx \log_2 n$$

La relación entre la profundidad y la cantidad de nodos es **logarítmica**.

Caso límite 2: *árbol degenerado*



La relación entre la profundidad y la cantidad de nodos es **lineal**.

ABB - Costo de búsqueda

- La *profundidad* del ABB nos da un orden de *complejidad de la búsqueda*.
- Se preferirá siempre contar con el *árbol balanceado* al momento de realizar la búsqueda, dado que el costo en este caso es *logarítmico*, en comparación con el costo *lineal* de un árbol degenerado (lista, peor caso).

A top-down view of a desk with various items: a keyboard in the top left, a stack of pens in the top center, a small cup in the center, a mouse on the left, and several notebooks and pens on the right. The entire scene is overlaid with a semi-transparent red filter.

Árboles Binarios de Búsqueda Balanceados - AVL

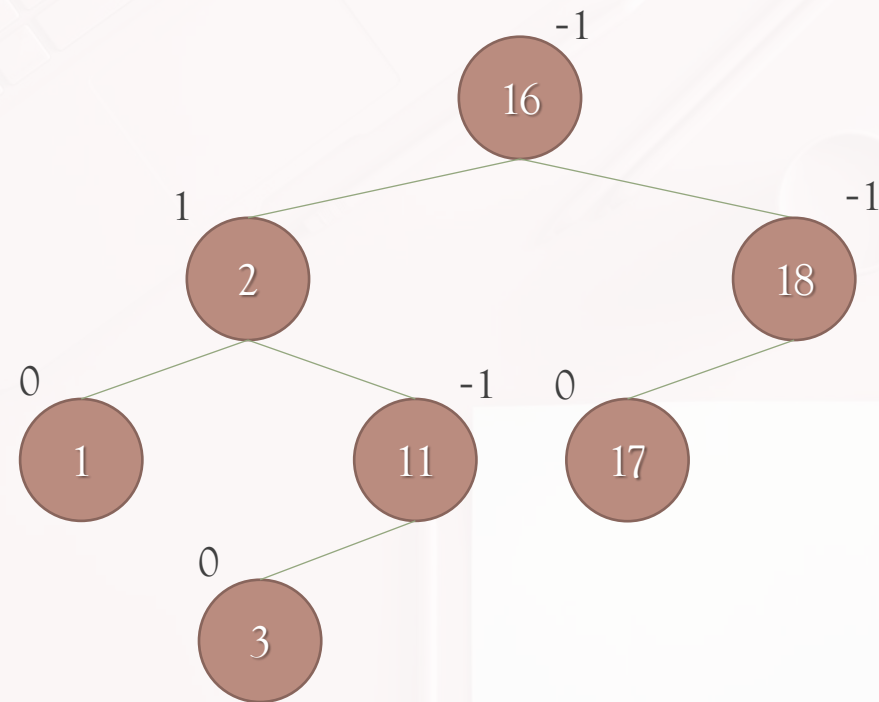
AVL

- El *Árbol Binario de Búsqueda Balanceado*, o AVL, fue ideado por Adelson-Velskii y Landis, siendo el primer ABB auto-balanceable ideado.
- Posee una condición de equilibrio: *“la diferencia entre la cantidad de niveles de los subárboles derecho e izquierdo no debe excederse en más de 1 para cada nodo del árbol”*. Se denomina *Factor de Balanceo* (FB).

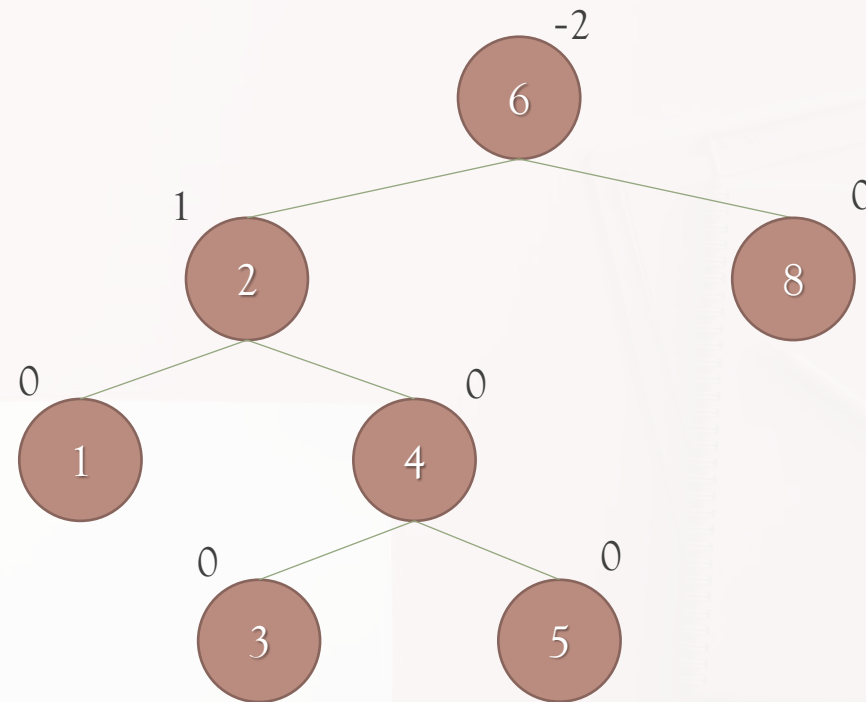
AVL

- La *complejidad* de una búsqueda (así como de agregar y de eliminar) se mantiene siempre *logarítmica*, se evita el árbol degenerado (cuyo orden de complejidad puede ser superior).
- Se utiliza en bases de datos, por la eficiencia de búsqueda (acelerando el acceso a la información en grandes BD).

AVL – Ejemplos



Balanceado



No Balanceado



Rotaciones

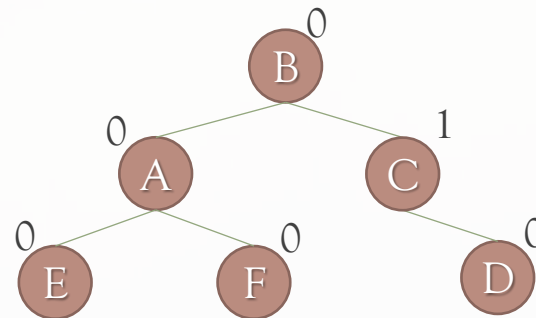
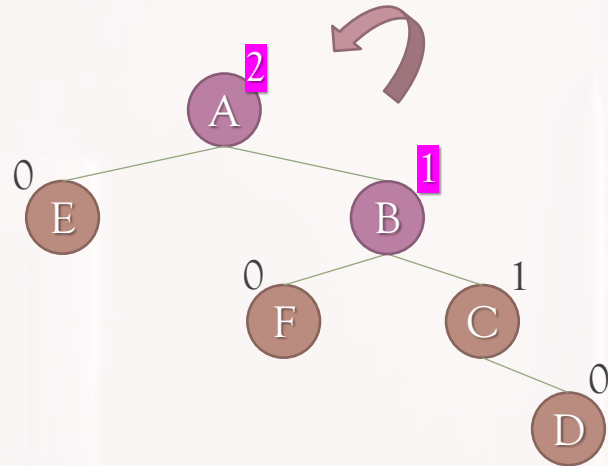
Rotaciones

Para mantener un AVL balanceado se debe, luego de cada inserción o eliminación, *verificar la condición de equilibrio*, calculando los FB.

En caso de no cumplirla, se realiza una corrección. Para restituir la condición de balanceo, se realizan “*Rotaciones*”.

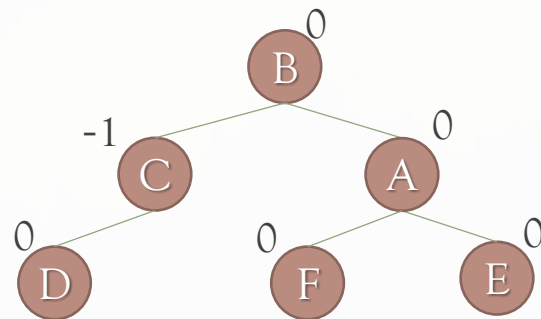
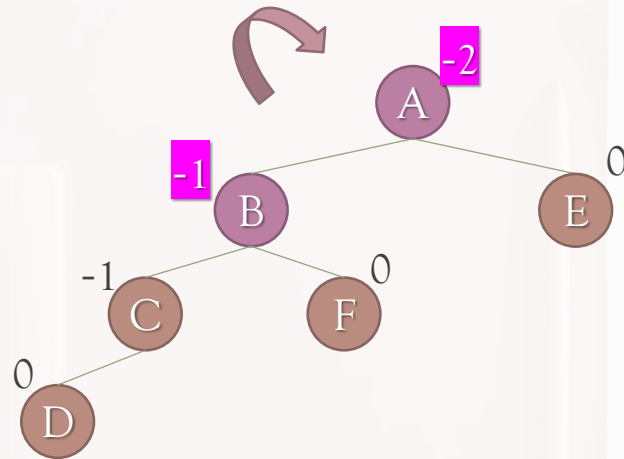
Rotación Simple Izquierda - RSI

- Si está desequilibrado a la derecha, el FB del nodo desbalanceado es > 1 (+) y el FB de su hijo derecho tiene igual signo (+).



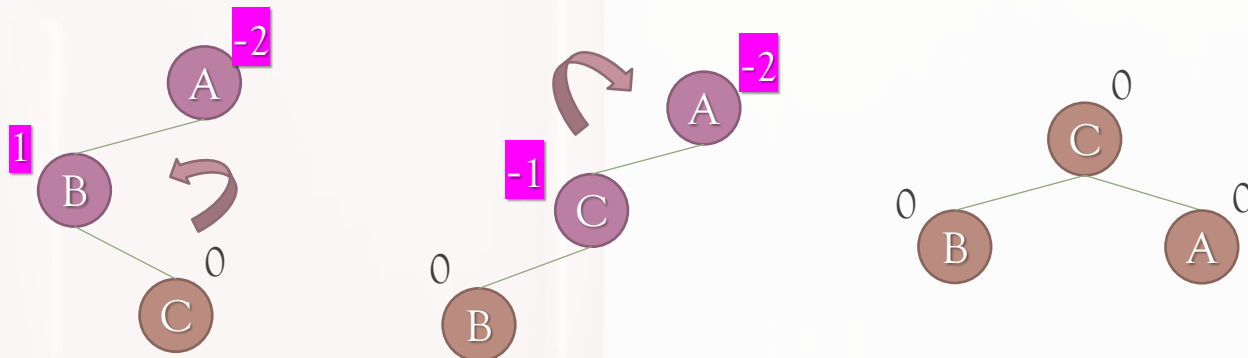
Rotación Simple Derecha - RSD

- Si está desequilibrado a la izquierda, el FB del nodo desbalanceado es < -1 (-) y el FB de su hijo izquierdo tiene igual signo (-).



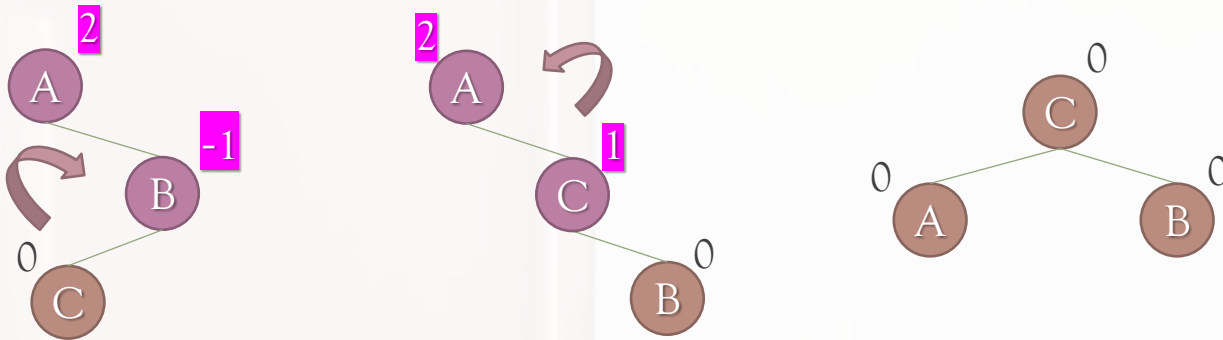
Rotación Doble Izquierda - RDI


- Si está desequilibrado a la izquierda, el FB del nodo desbalanceado es < -1 (-) y el FB de su hijo izquierdo tiene distinto signo (+).
- Luego se realiza una Rotación Simple Derecha.



Rotación Doble Derecha - RDD

- Si está desequilibrado a la derecha, el FB del nodo desbalanceado es $> 1 (+)$ y el FB de su hijo derecho tiene distinto signo $(-)$.
- Luego se realiza una Rotación Simple Izquierda.





Ejemplo

Insertión – Eliminación

◦ *Partimos de un AVL vacío*

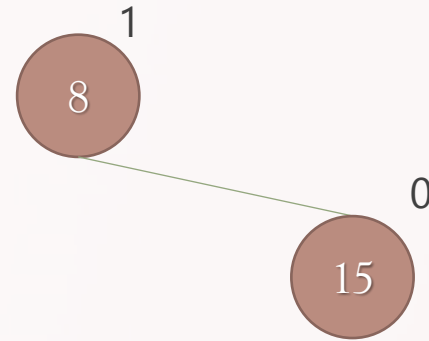
Insertión – Eliminación

- *Partimos de un AVL vacío*
- *Inserto el 8*



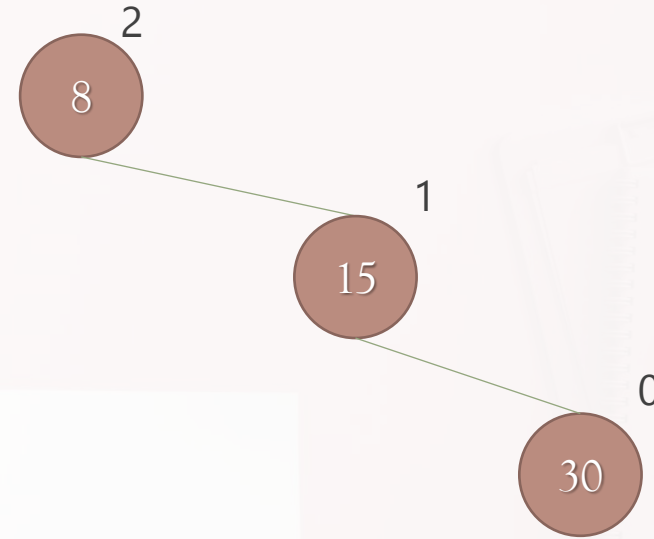
Inserción – Eliminación

- *Partimos de un AVL vacío*
- *Inserto el 8*
- *Luego, el 15*



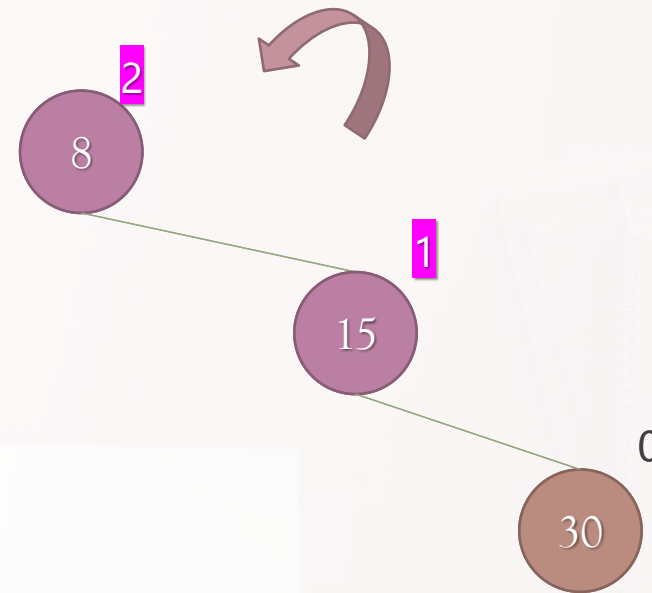
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30



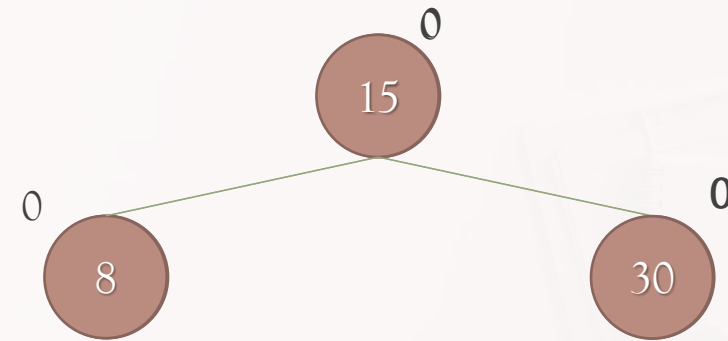
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI



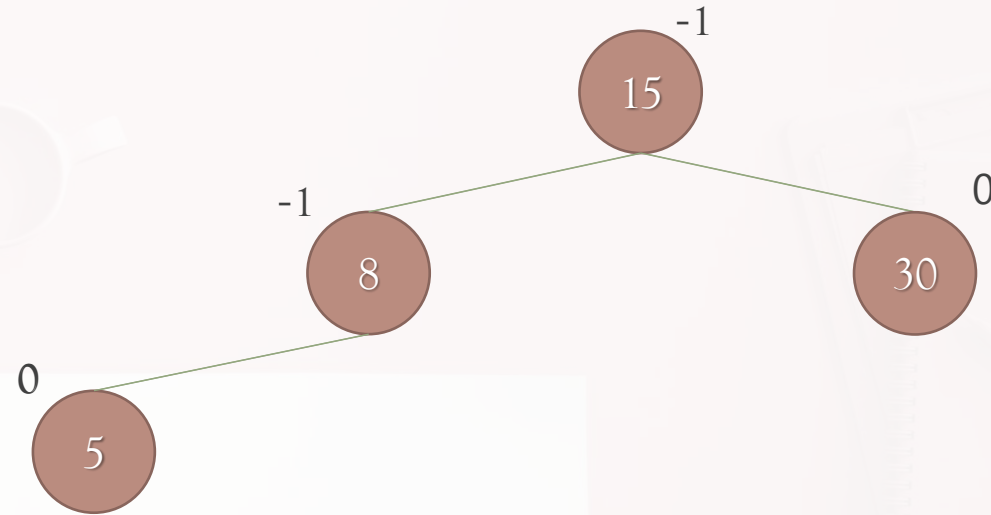
Insertión – Eliminación

- *Partimos de un AVL vacío*
- *Inserto el 8*
- *Luego, el 15*
- *Ahora, el 30*
- *Balanceo – RSI*



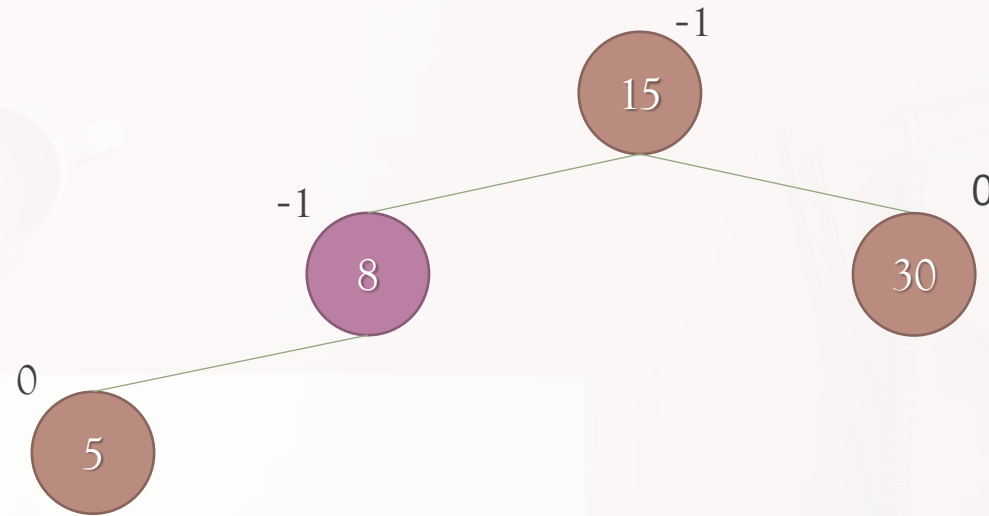
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI
- Inserto el 5



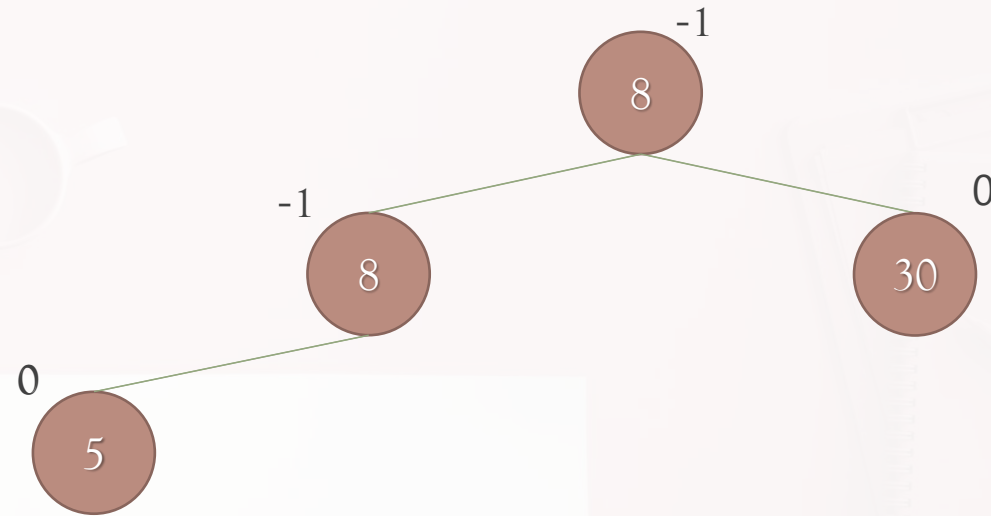
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI
- Inserto el 5
- Ahora, elimino el 15



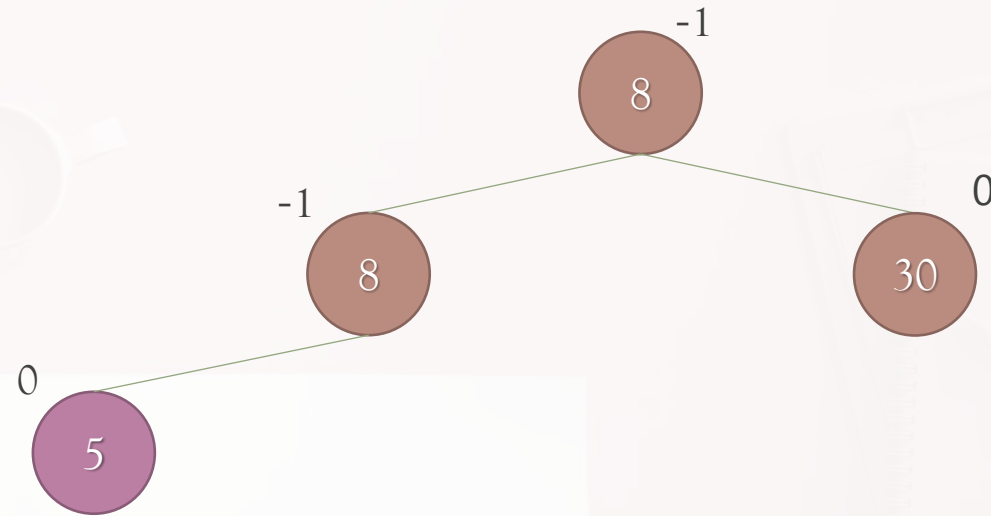
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI
- Inserto el 5
- Ahora, elimino el 15



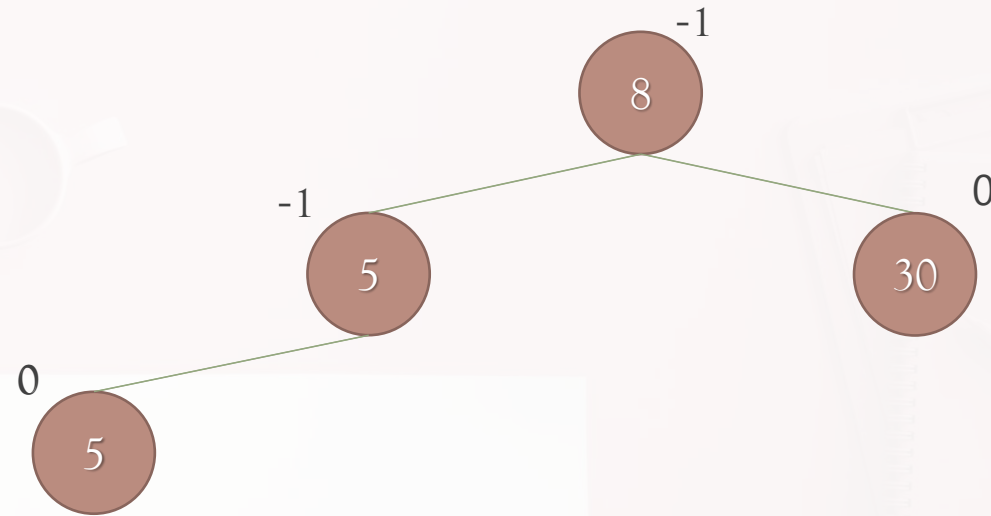
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI
- Inserto el 5
- Ahora, elimino el 15



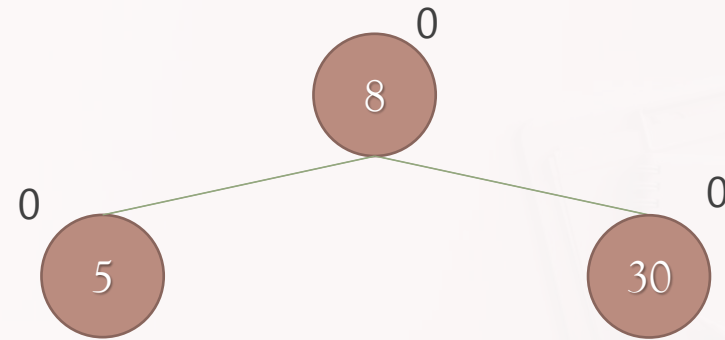
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI
- Inserto el 5
- Ahora, elimino el 15



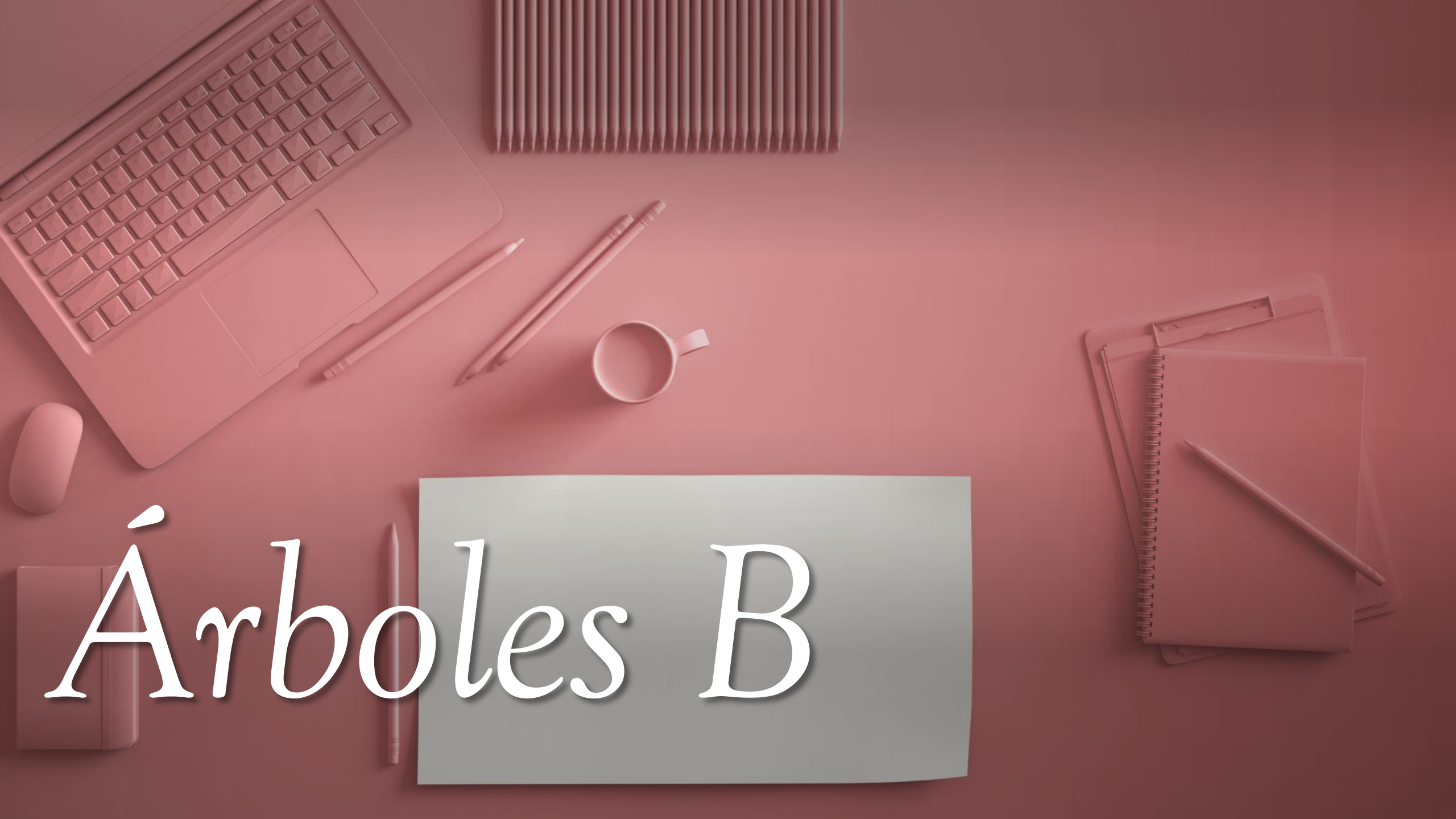
Insertión – Eliminación

- Partimos de un AVL vacío
- Inserto el 8
- Luego, el 15
- Ahora, el 30
- Balanceo – RSI
- Inserto el 5
- Ahora, elimino el 15



Link

<https://yongdanielliang.github.io/animation/web/AVLTree.html>



Árboles B

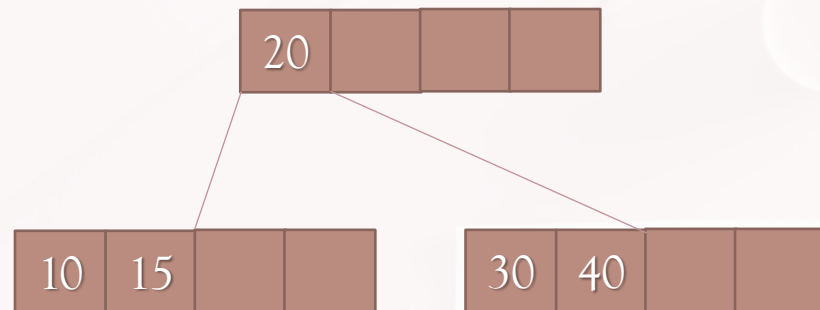
Árbol *B*

- Los *Árboles B* son árboles balanceados de búsqueda, ideados por R. Bayer y E. McCreight.
- Tienen un *orden* q .
- Cada nodo tiene más de una clave (no fijo), con una *cantidad máxima de claves* $(q - 1)$.
- Y una *cantidad mínima de claves* (cantidad máxima $/ 2$), salvo la raíz.

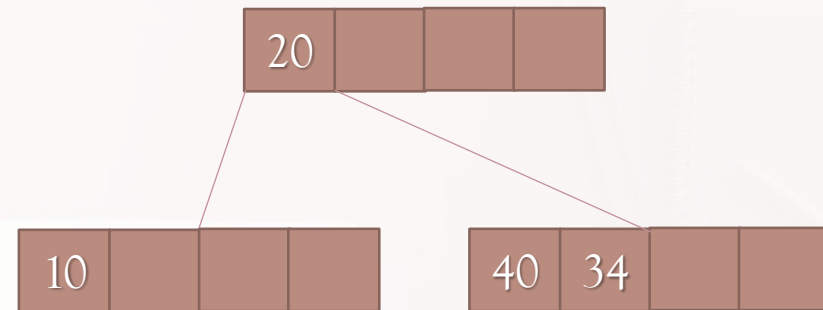
Árbol B

- Cada nodo tiene *múltiples hijos* (cantidad de claves + 1).
- Las *claves* (que *no son repetidas*) se encuentran *ordenadas*.
- Las *hojas* están siempre en el *mismo nivel*.
- Se utiliza en bases de datos y sistemas de archivos, para el mantenimiento de índices, por la eficiencia de búsqueda.

Árbol B – Ejemplos



Árbol B



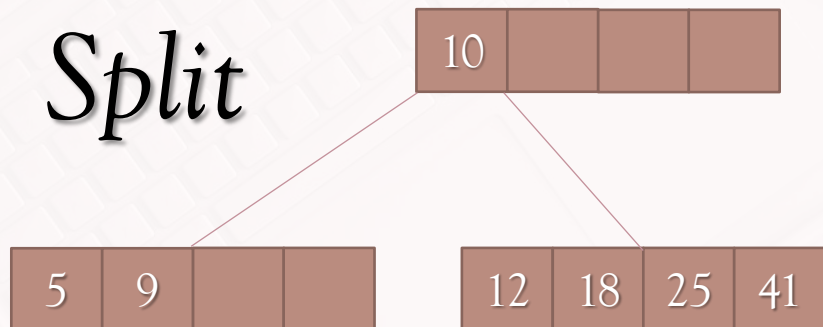
No es Árbol B

Inserciones

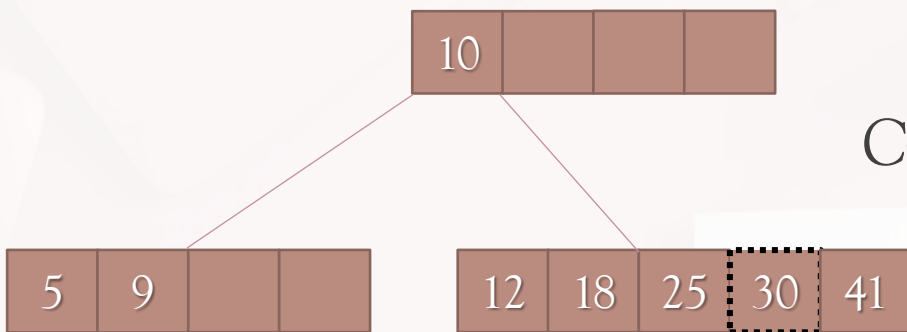
- Se busca el nodo *hoja* correspondiente, se inserta manteniendo el orden.
- Si se supera el valor máximo de claves, se realiza un “*Split*”:
 - se ordena con un elemento “*dummy*”,
 - se *promociona* el elemento central.
- Puede tener que repetirse recursivamente para los nodos ascendentes.

Split

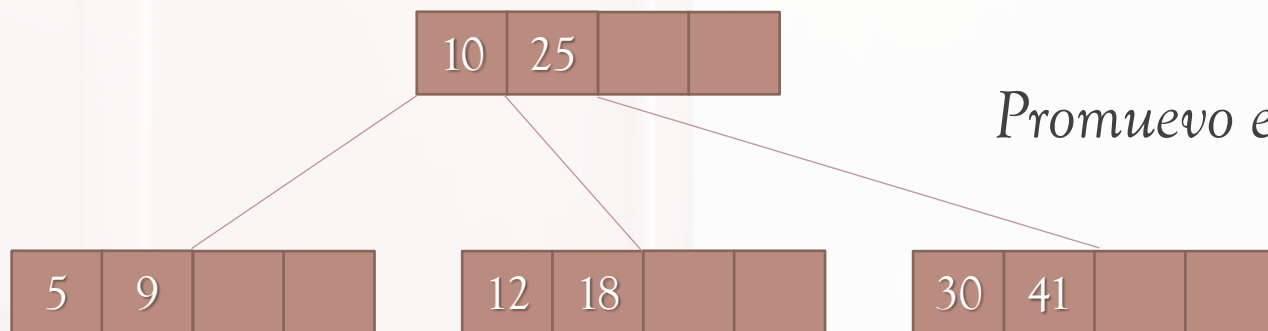
Quiero insertar el 30



Creo el elemento dummy



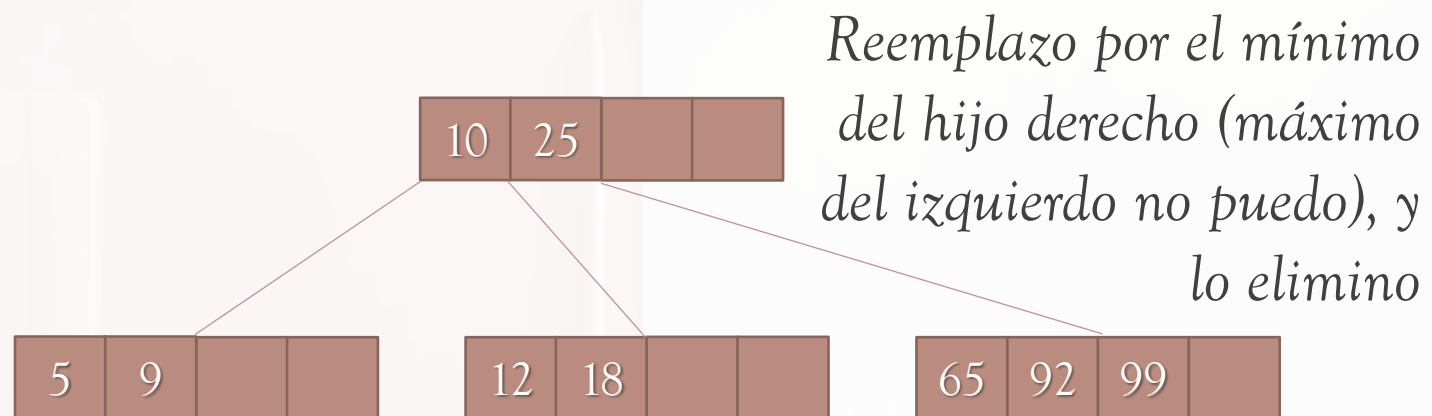
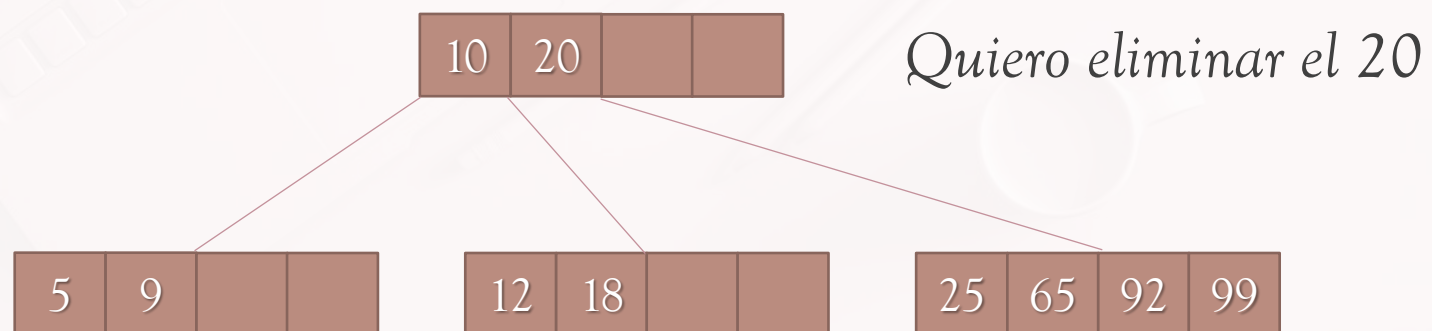
Promuevo el central



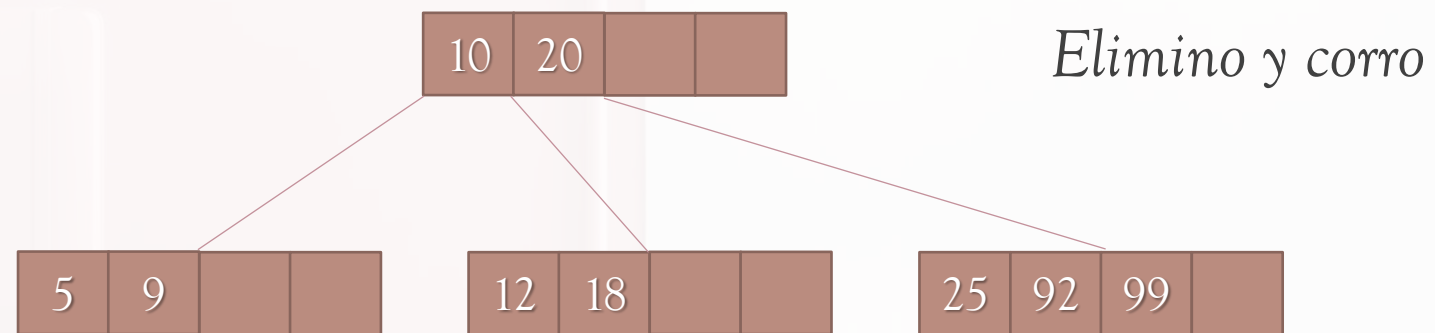
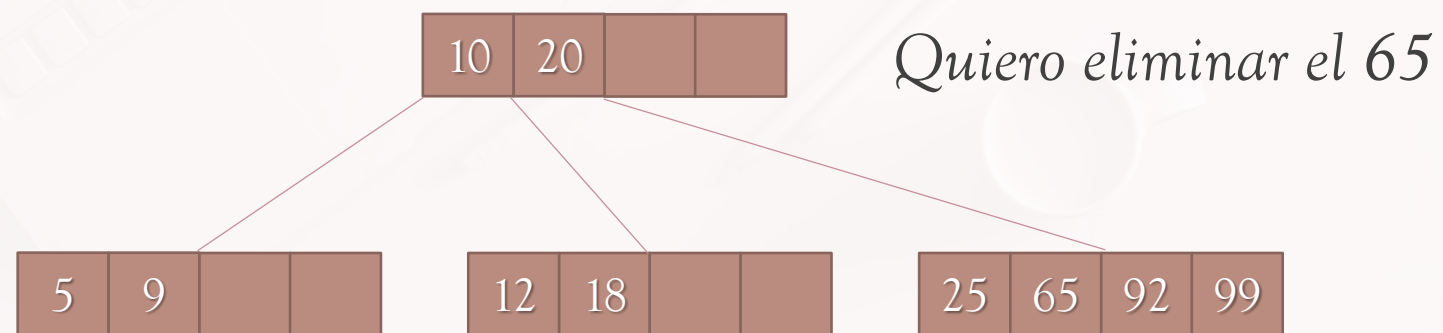
Eliminaciones

- Se busca el nodo correspondiente:
 - si es *nodo interno*, se reemplaza el elemento por el máximo del hijo izquierdo (o, de no poderse, el mínimo del hijo derecho), así hasta la hoja;
 - si es *hoja*, se elimina el elemento (realizando corrimientos en el nodo).

Eliminación simple en nodo interno



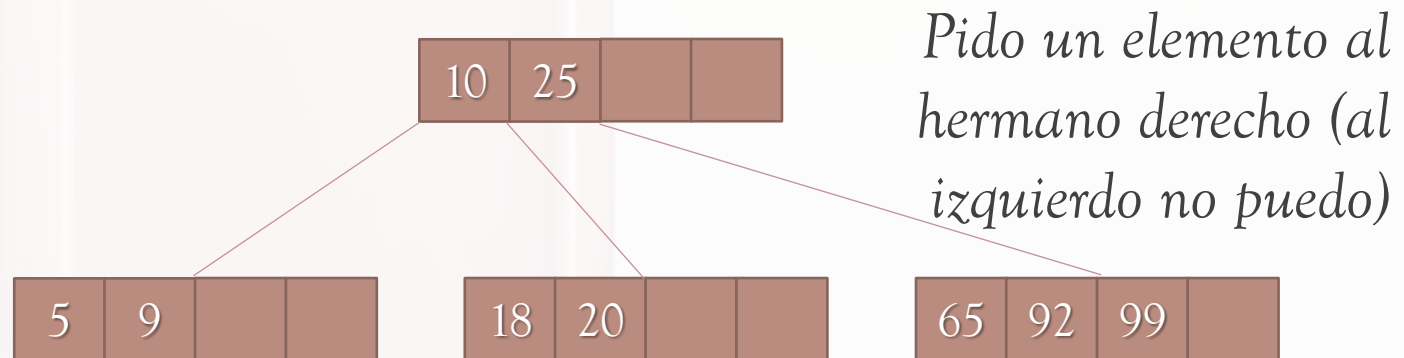
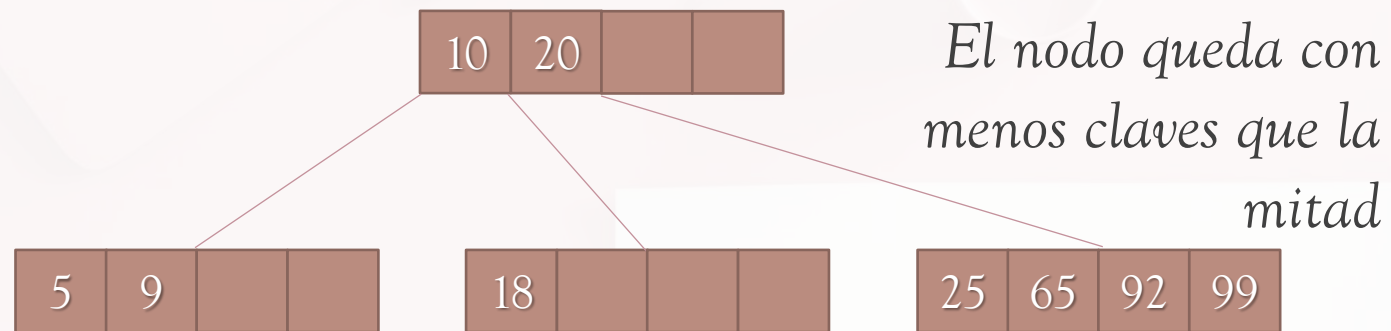
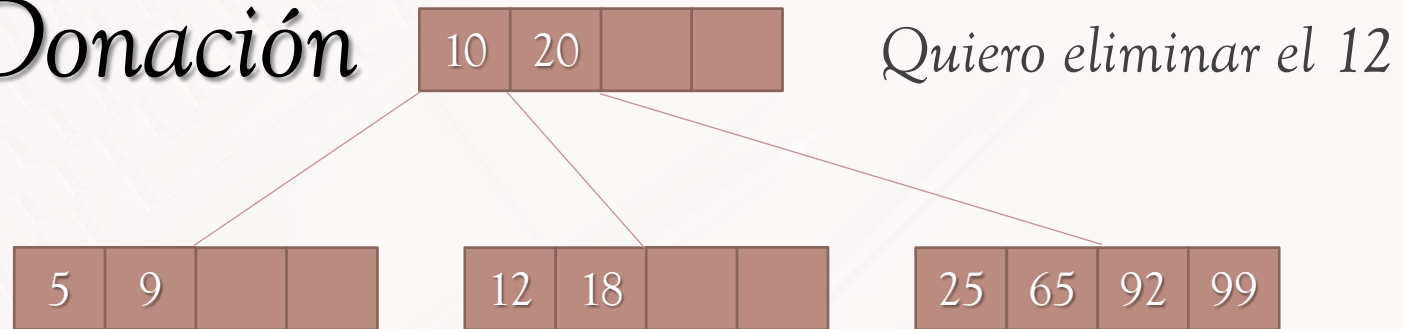
Eliminación simple en hoja



Eliminaciones

- Si al eliminar el elemento, el número de claves es menor al mínimo permitido, se trata de realizar una “*donación*”:
 - *transferencia* de un elemento del hermano izquierdo (o, de no poderse, derecho),
 - se *promociona* el hijo, y degrada el padre al nodo en cuestión.

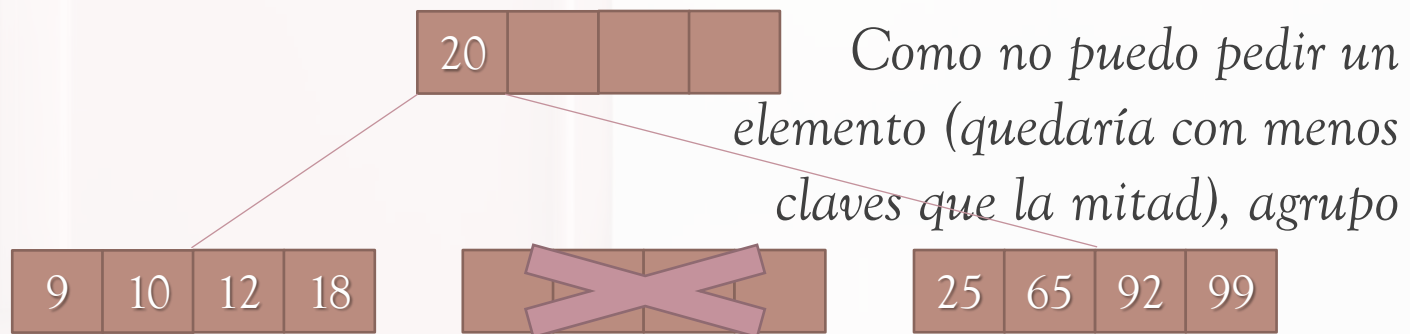
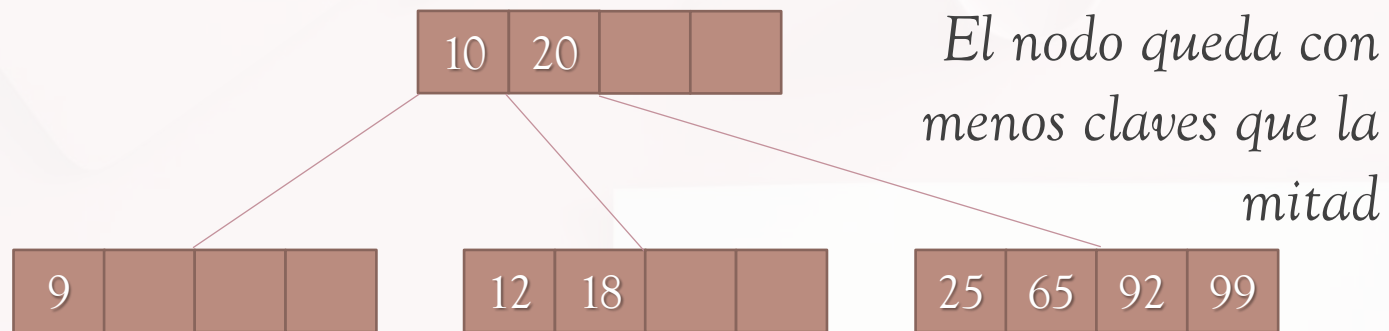
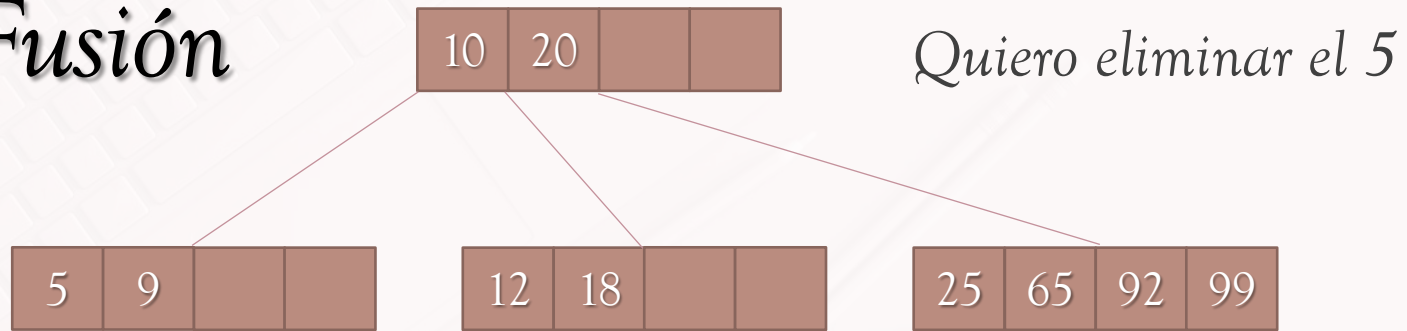
Donación




Eliminaciones

- Si no tienen elementos suficientes, recién en ese caso, se procede a una “*fusión*”:
 - se *agrupa* el padre (degrada) con dos hijos a izquierda (o, de no poderse, a derecha).
- Puede tener que repetirse recursivamente para los nodos ascendentes.

Fusión

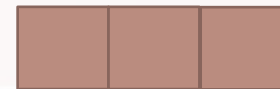




Ejemplo

Insertión – Eliminación

- *Partimos de un Árbol B vacío*



Insertión – Eliminación

- *Partimos de un Árbol B vacío*
- *Agrego el 8*

8		
---	--	--

Insertión – Eliminación

- *Partimos de un Árbol B vacío*
- *Agrego el 8*
- *Agrego el 71*

8	71	
---	----	--

Inserción – Eliminación

- *Partimos de un Árbol B vacío*
- *Agrego el 8*
- *Agrego el 71*
- *Agrego 42*

8	42	71
---	----	----

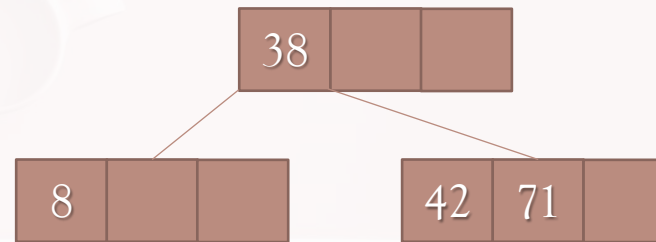
Insertión – Eliminación

- *Partimos de un Árbol B vacío*
- *Agrego el 8*
- *Agrego el 71*
- *Agrego 42*
- *Agrego el 38 – split*

8	38	42	71
---	----	----	----

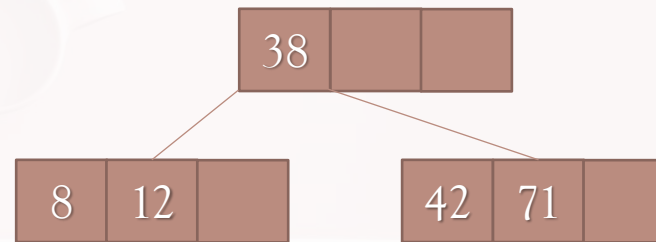
Insertión – Eliminación

- Partimos de un Árbol B vacío
- Agrego el 8
- Agrego el 71
- Agrego 42
- Agrego el 38 – split



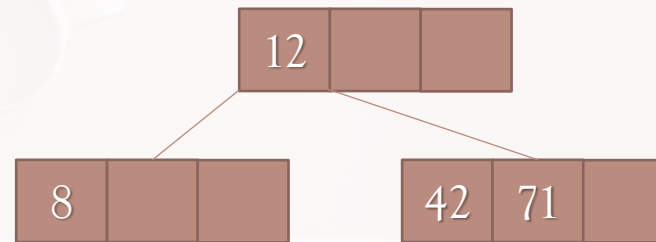
Insertión – Eliminación

- Partimos de un Árbol B vacío
- Agrego el 8
- Agrego el 71
- Agrego 42
- Agrego el 38 – split
- Agrego el 12



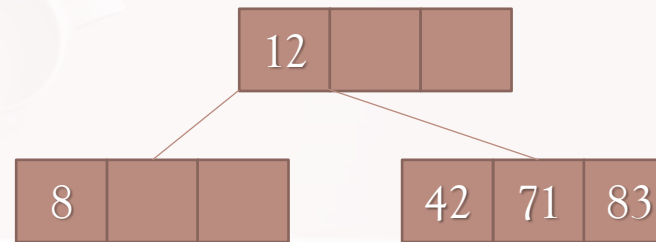
Insertión – Eliminación

- Partimos de un Árbol B vacío
- Agrego el 8
- Agrego el 71
- Agrego 42
- Agrego el 38 – split
- Agrego el 12
- Elimino el 38 – de nodo interno



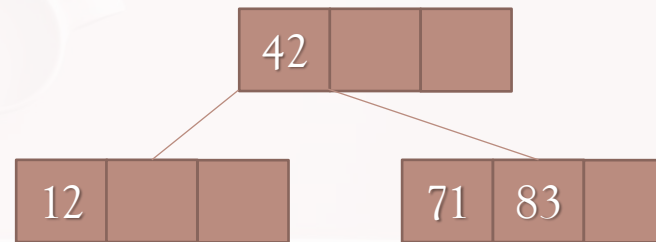
Insertión – Eliminación

- Partimos de un Árbol B vacío
- Agrego el 8
- Agrego el 71
- Agrego 42
- Agrego el 38 – split
- Agrego el 12
- Elimino el 38 – de nodo interno
- Agrego el 83



Insertión – Eliminación

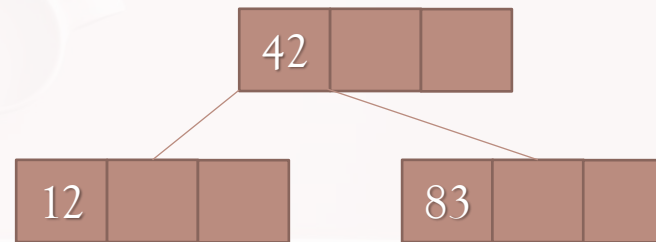
- Partimos de un Árbol B vacío
- Agrego el 8
- Agrego el 71
- Agrego 42
- Agrego el 38 – split
- Agrego el 12
- Elimino el 38 – de nodo interno
- Agrego el 83



- Elimino el 8 – donación

Insertión – Eliminación

- *Partimos de un Árbol B vacío*
- *Agrego el 8*
- *Agrego el 71*
- *Agrego 42*
- *Agrego el 38 – split*
- *Agrego el 12*
- *Elimino el 38 – de nodo interno*
- *Agrego el 83*



- *Elimino el 8 – donación*
- *Elimino el 71*

Insertión – Eliminación

- *Partimos de un Árbol B vacío*

- *Agrego el 8*

- *Agrego el 71*

- *Agrego 42*

12	42	
----	----	--

- *Agrego el 38 – split*

- *Agrego el 12*

- *Elimino el 8 – donación*

- *Elimino el 38 – de nodo interno*

- *Elimino el 71*

- *Agrego el 83*

- *Elimino el 83 – fusión*

Link

<https://www.cs.usfca.edu/~galles/visualization/BTree.html>



¡Muchas Gracias!



Bibliografía

- 👑 *Programación II – Apuntes de
Cátedra – V1.3 – Cuadrado
Trutner – UADE*
- 👑 *Programación II – Apuntes de
Cátedra – Wehbe – UADE*