ALGORITMOS Y ESTRUCTURAS DE DATOS CURSADA 2025

LICENCIATURA EN SISTEMAS - UNRN

Prof. Sebastián N. Valle

snvalle@unrn.edu.ar

Prof. Guillermo A. Difabio

guillermodifabio@unrn.edu.ar

ARBOLES AVL

ARBOLES AVL

- Definición
- Características
- Operaciones
- Balanceo del Árbol
- Rotación Simple
- Rotación Doble
- Tiempo de Ejecución - Conclusiones

DEFINICIÓN

Un árbol AVL (Adelson–Velskii–Landis) es un **árbol binario de búsqueda** que cumple con la condición de estar **balanceado.**

La propiedad de balanceo que cumple dice:

Para cada nodo del árbol, la diferencia de altura entre el subárbol izquierdo y el subárbol derecho es a lo sumo 1

CARACTERÍSTICAS

- La propiedad de balanceo es fácil de mantener y garantiza que la altura del árbol sea de O(log n).
- En cada nodo del árbol se guarda información de la altura.
- La altura del árbol vacío es -1.

OPERACIONES EN UN AVL

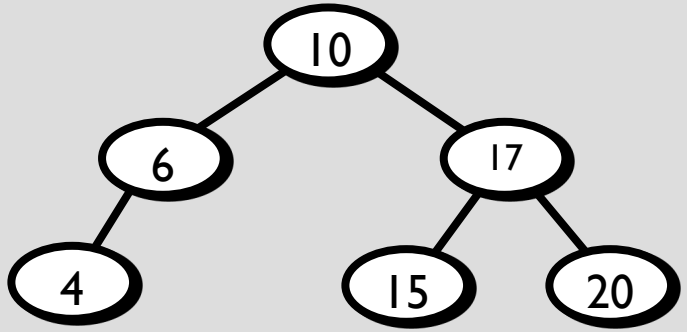
- Búsqueda / Recuperación
- Inserción
- Eliminación
  - Al insertar o eliminar un dato en un AVL se puede perder la propiedad del balanceo.
  - Al insertar o eliminar un dato en un AVL se deben mantener los criterios de ABB.
  - Es fundamental preservar el balanceo y el criterio de orden al realizar estas operaciones sobre el árbol.

INSERCIÓN EN UN AVL

- La inserción se realiza igual que en un árbol binario de búsqueda
- Puede destruirse la propiedad de balanceo
- Entonces, aparece el concepto de Re balancear.
- Al insertar un elemento se actualiza la información de la altura de los nodos que están en el camino desde el nodo insertado a la raíz
- El desbalanceo sólo se produce en ese camino, ya que sólo esos nodos tienen sus subárboles modificados

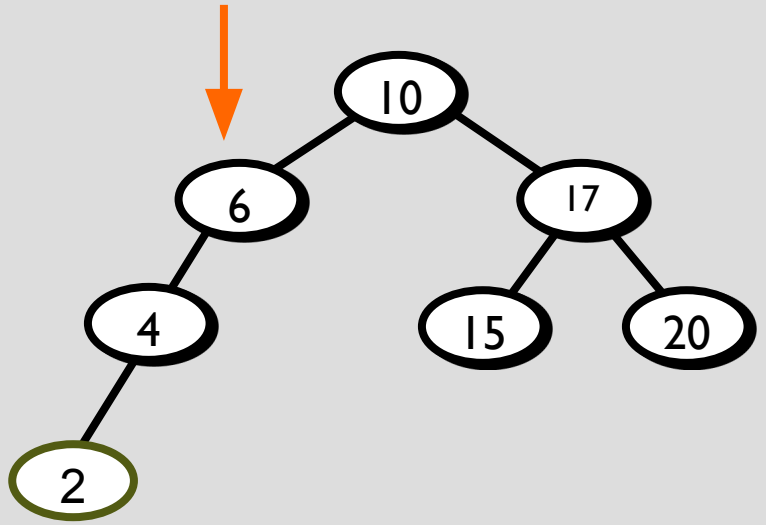
INSERCIÓN EN UN AVL - DESBALANCEO

Al insertar 2



Árbol antes de insertar el 2

Se desbalancea el 6



Árbol después de insertar el 2

BALANCEO DEL ÁRBOL

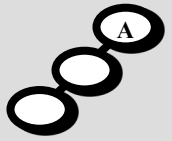
- Para restaurar el balanceo del árbol:

- se recorre el camino de búsqueda en orden inverso
- se controla el equilibrio/balanceo de cada nodo
- si está desbalanceado se realiza una modificación simple: rotación
- después de re balancear el nodo, la inserción termina
- este proceso puede llegar a la raíz

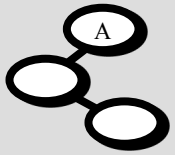
BALANCEO DEL ÁRBOL (2)

- Hay 4 casos posibles de desbalanceo a tener en cuenta, según donde se hizo la Inserción. El nodo A es el nodo desbalanceado.

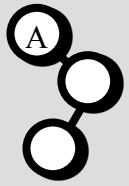
1. Inserción en el Subárbol IZQ del hijo IZQ de A



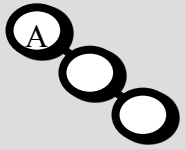
2. Inserción en el Subárbol DER del hijo IZQ de A



3. Inserción en el Subárbol IZQ del hijo DER de A



4. Inserción en el Subárbol DER del hijo DER de A



BALANCEO DEL ÁRBOL (3)

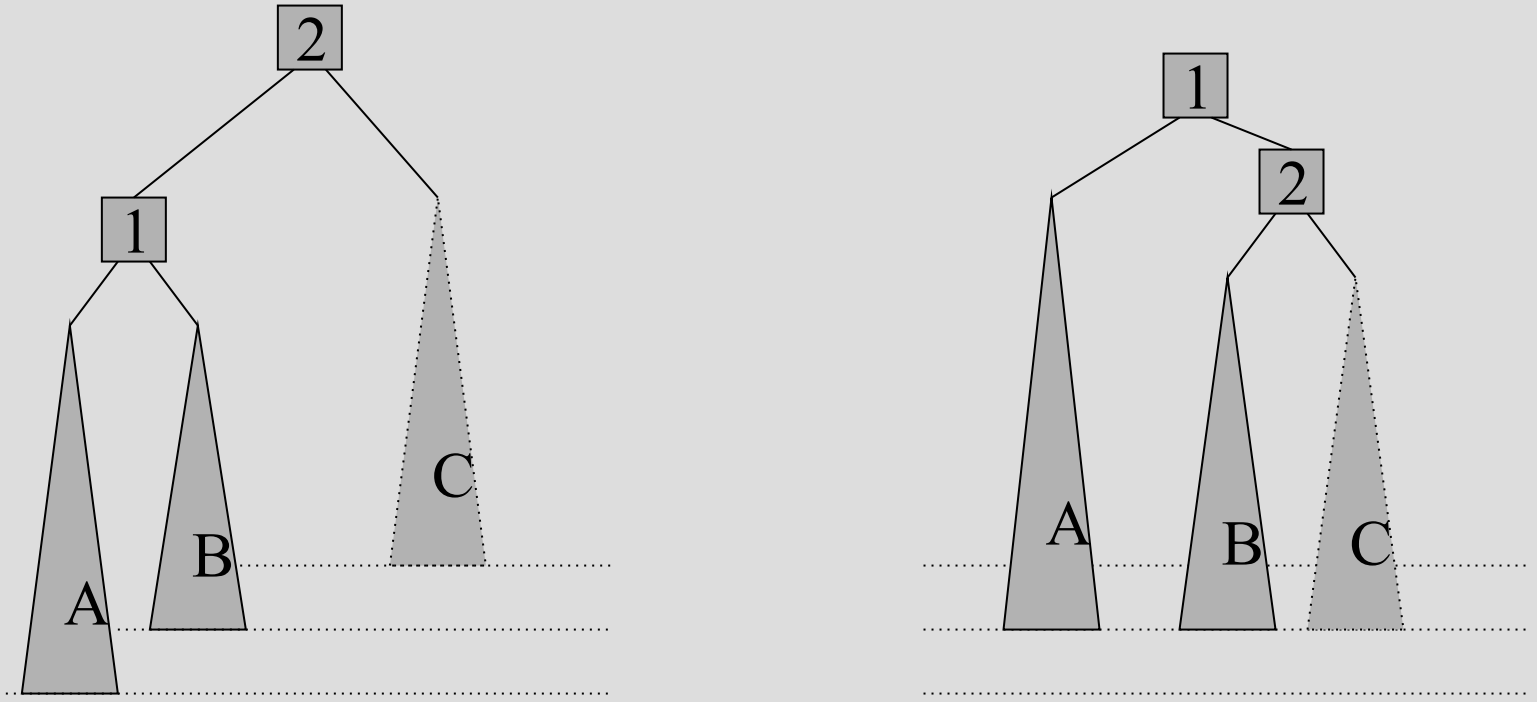
- La solución para restaurar el balanceo es la ROTACIÓN
- La rotación es una modificación simple de la estructura del árbol, que restaura la propiedad de balanceo, preservando el orden de los elementos

- Existen dos clases de rotaciones:
- Rotación Simple: Casos 1 y 4: inserción en el lado externo
- Rotación Doble: Casos 2 y 3: inserción en el lado interno

- Soluciones simétricas: En cada caso, los subárboles están opuestos.

ROTACIÓN SIMPLE

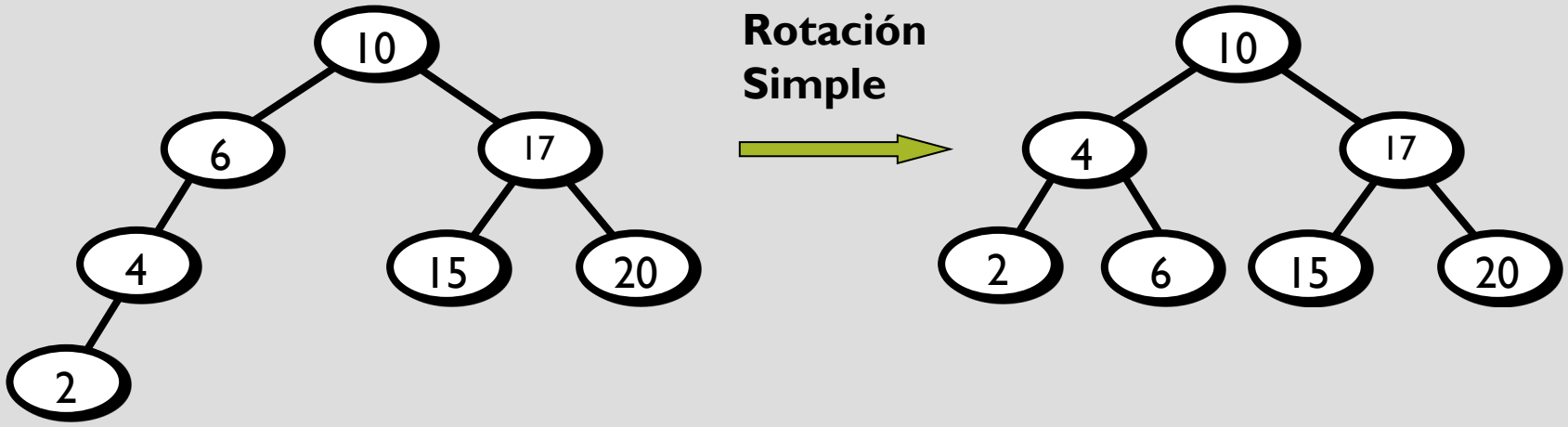
- **Caso 1: Rotación Simple Izquierda**



Se obtuvo nuevamente un árbol balanceado

ROTACIÓN SIMPLE (2)

Siguiendo con el ejemplo



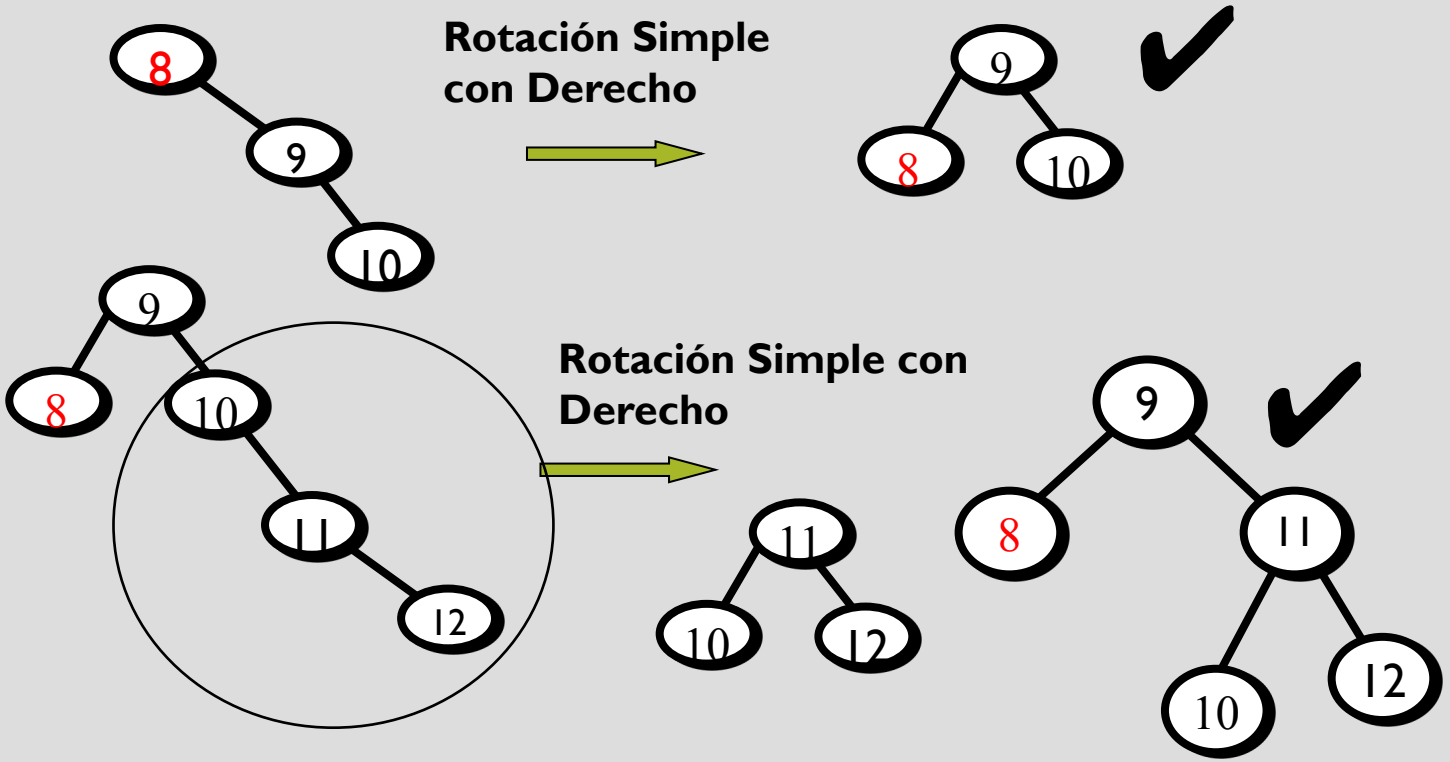
ROTACIÓN SIMPLE (3)

- **Caso 4: Rotación Simple Derecha**

- Es simétrico al caso 1, el desbalanceo se produce hacia el lado derecho.

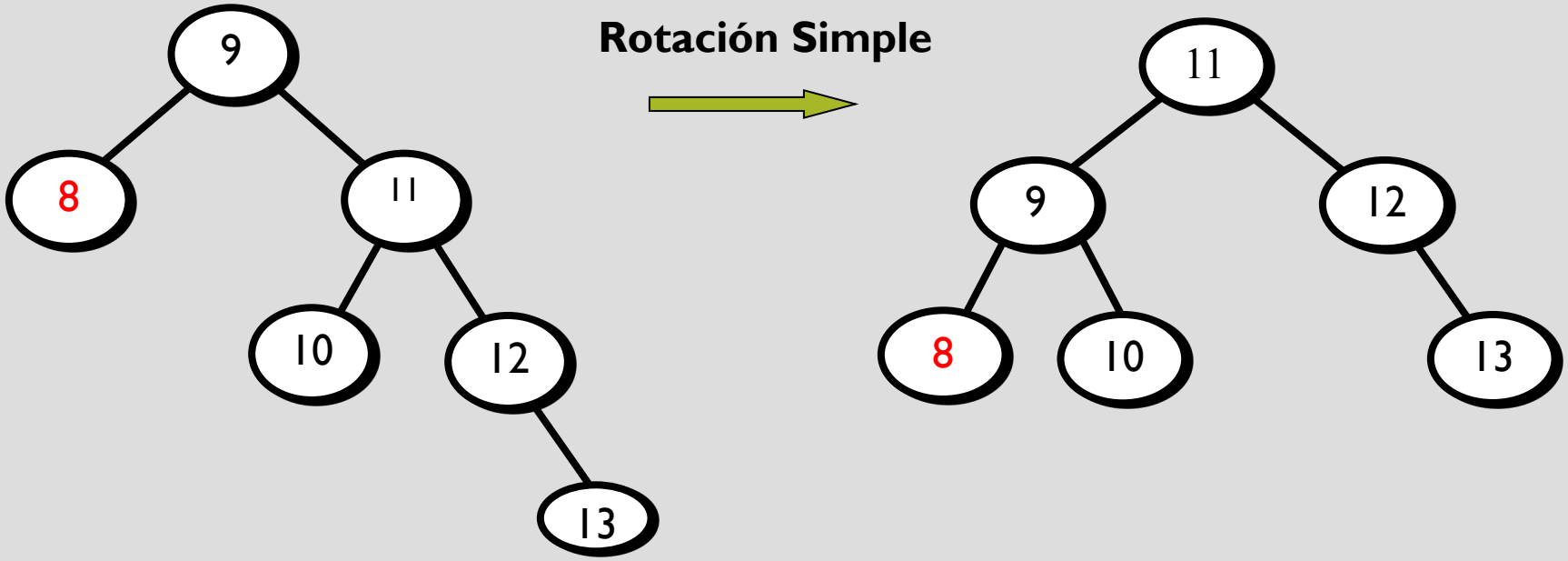
ROTACIÓN SIMPLE (4)

- Ejemplo: insertar las claves del 8 al 14, en ese orden, en un árbol AVL inicialmente vacío. Insertamos 8, 9, 10, 11, 12



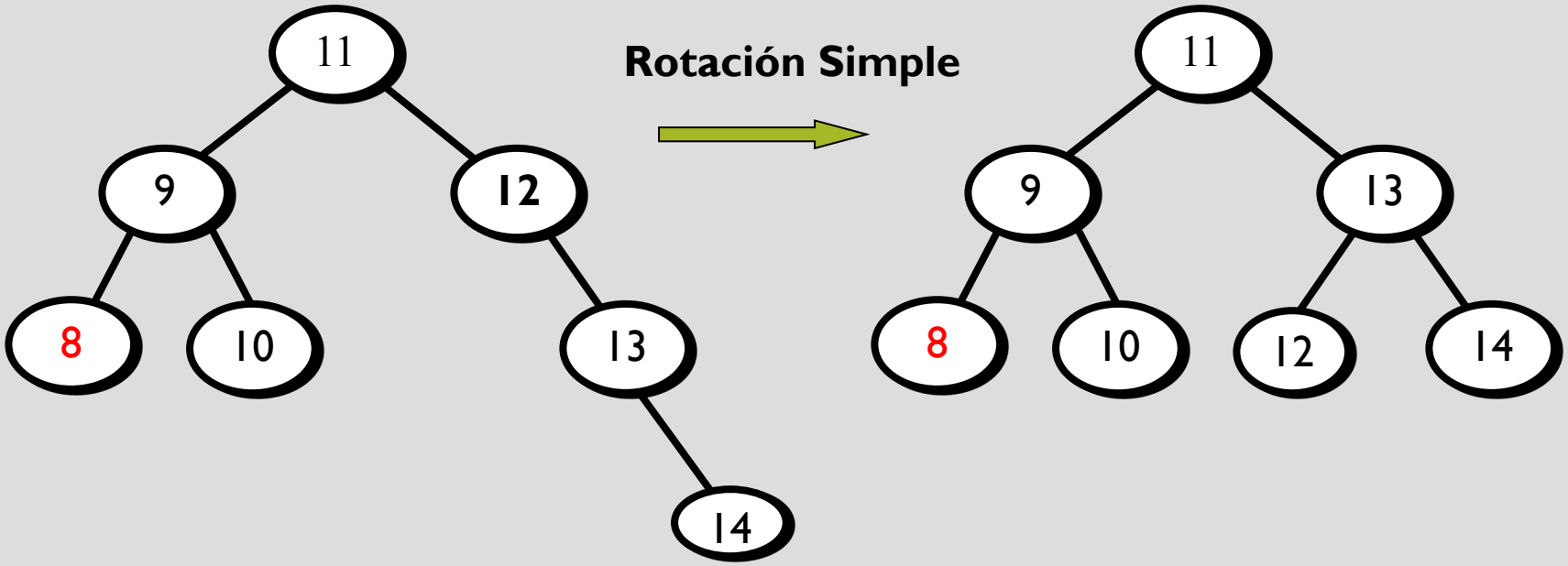
ROTACIÓN SIMPLE (5)

Siguiendo con el ejemplo insertamos 13



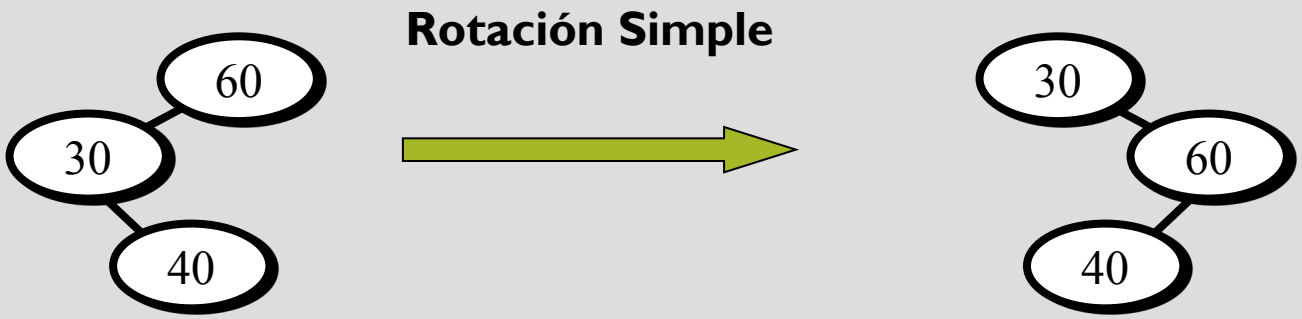
ROTACIÓN SIMPLE (6)

Siguiendo con el ejemplo insertamos 14



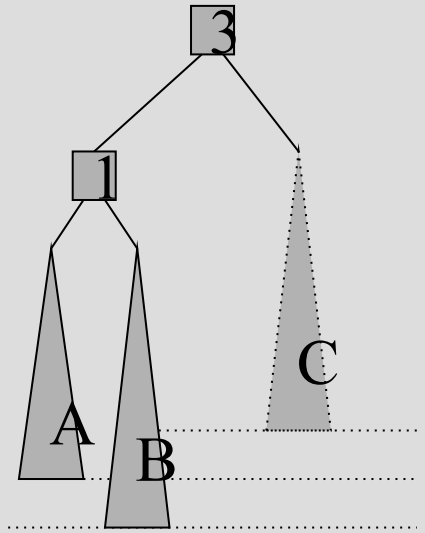
ROTACIÓN DOBLE

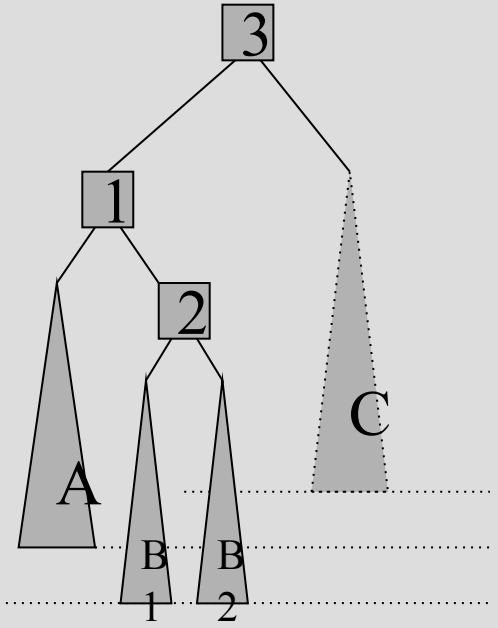
- En algunos casos la rotación simple no resuelve el problema
- **Caso 2: Rotación Doble Izquierda**



ROTACIÓN DOBLE (2)

- Dado que el subárbol B tiene por lo menos un ítem, podemos considerar que está formado por una raíz y dos subárboles

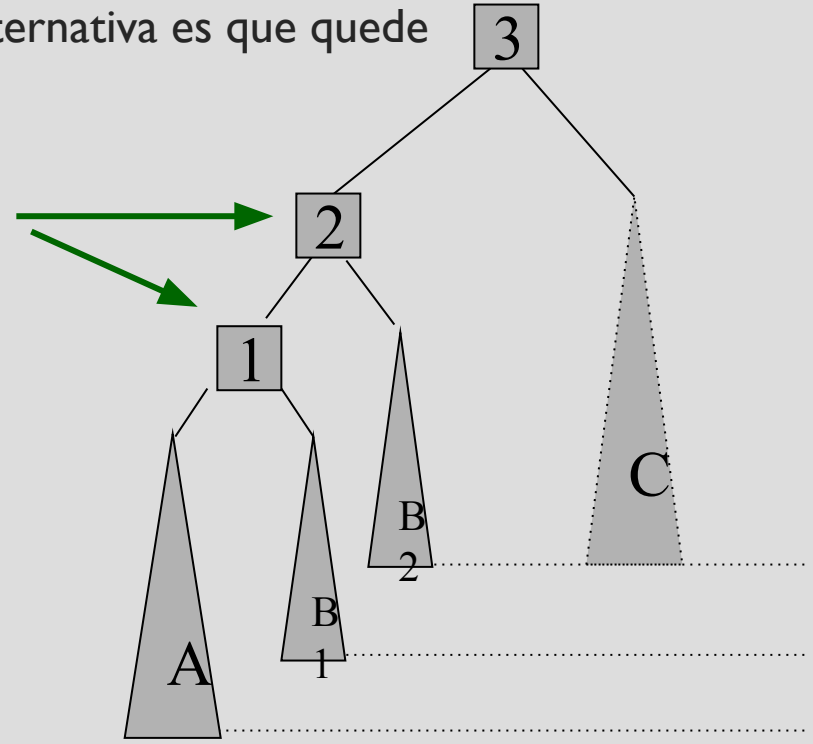




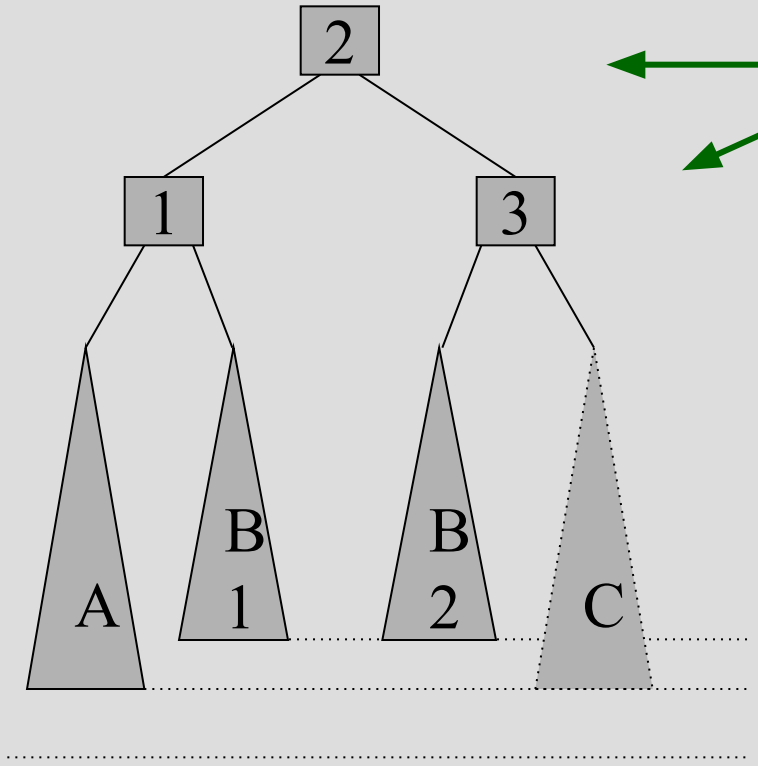
ROTACIÓN DOBLE (3)

- La rotación doble es similar a la simple, sólo que involucra cuatro subárboles en lugar de tres
- Ni los nodos 1 y 3 pueden quedar como raíz, la única alternativa es que quede el nodo 2

Primero se hace una rotación simple entre 1 y 2



ROTACIÓN DOBLE (4)



Luego se hace una rotación simple entre 2 y 3

ROTACIÓN DOBLE (5)

- **Caso 3: Rotación Doble Derecha**

- Es simétrica al caso 2, la inserción se produce en el subárbol izquierdo del hijo derecho.

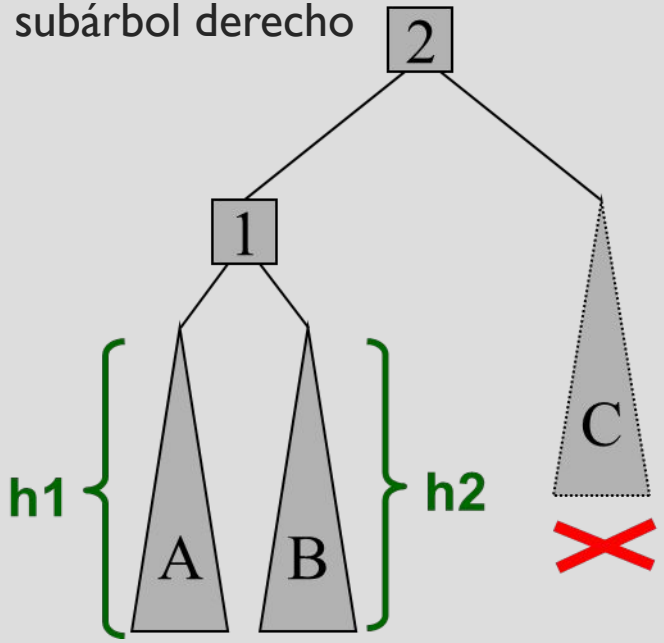
ELIMINACIÓN EN UN AVL

- La eliminación de un nodo es similar al borrado en un árbol binario de búsqueda.
- Luego de realizar el borrado se debe actualizar la altura de todos los nodos, desde el nodo en cuestión hasta la raíz.

- Puede destruirse la propiedad de balanceo por lo que también habrá que **re balancear** el árbol.

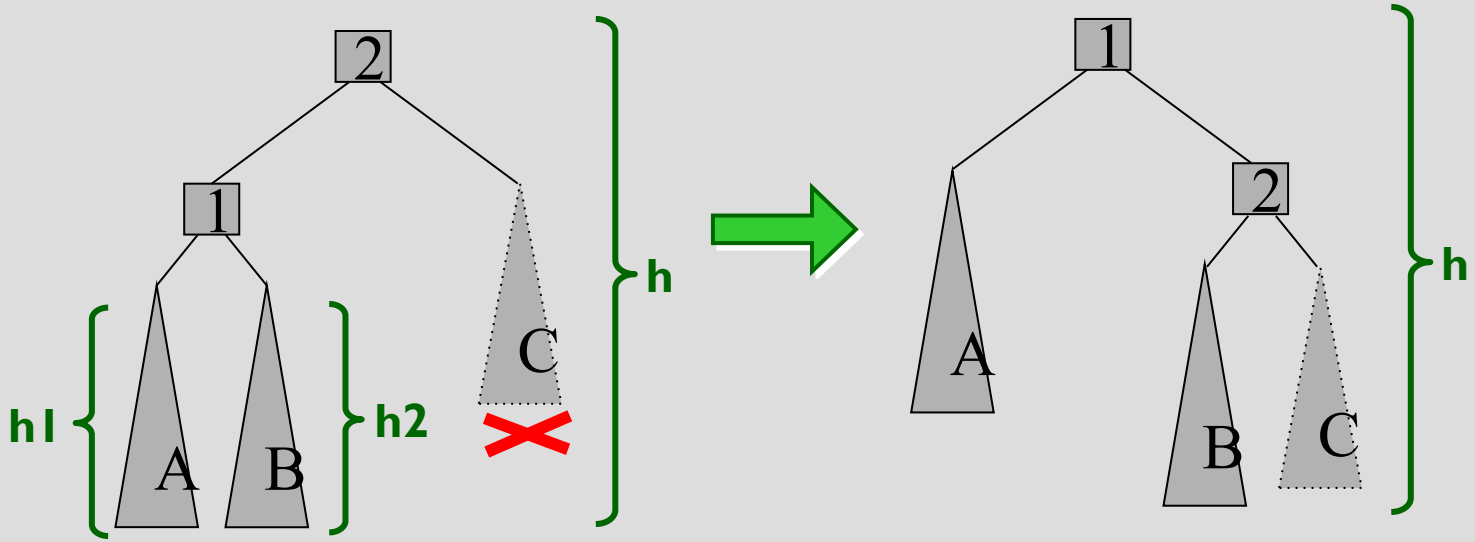
ELIMINACIÓN EN UN AVL (2)

- Se pueden definir 3 casos de desbalanceo a izquierda en un nodo y los simétricos a derecha.
- Los casos de desbalanceo en el subárbol izquierdo del nodo 2 dependen de las alturas h1 y h2 en el subárbol derecho



ELIMINACIÓN EN UN AVL (3)

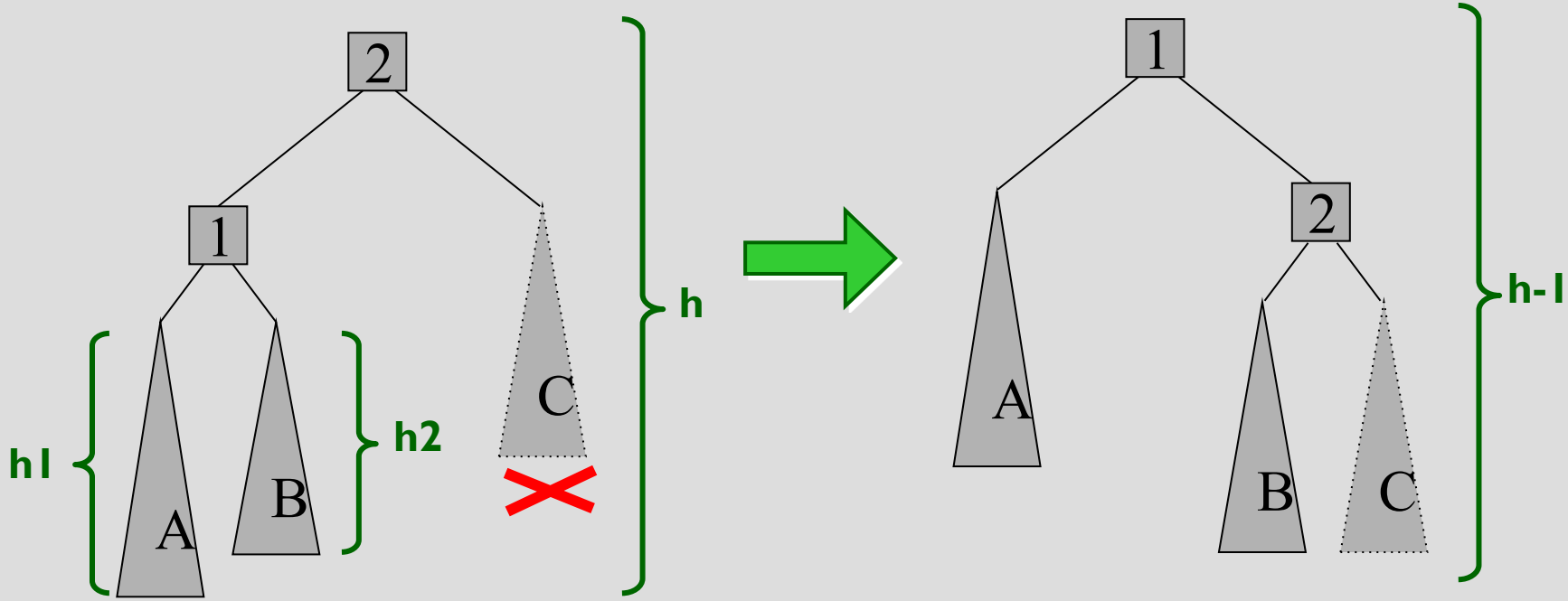
- Caso 1: h1= h2 Solución: RSI en el nodo 2



- El árbol resultante está balanceado
- La altura del árbol no cambia

ELIMINACIÓN EN UN AVL (4)

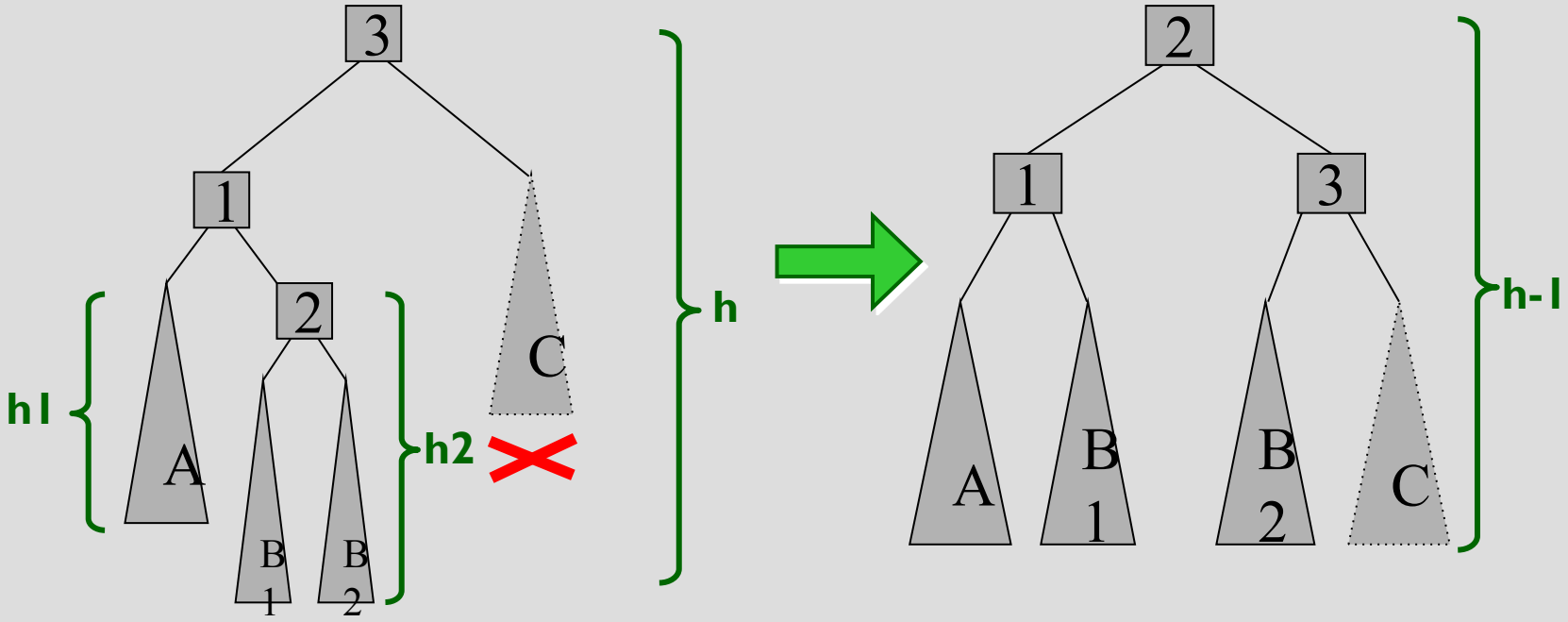
- Caso 2: h1> h2 Solución: RSI en el nodo 2



- En este caso, la altura del árbol disminuye en 1

ELIMINACIÓN EN UN AVL (5)

- Caso 3: h1< h2 Solución: RDI en el nodo 3



- La altura final del árbol disminuye en 1

TIEMPO DE EJECUCIÓN - CONCLUSIONES

- Las operaciones en AVL de
  - Búsqueda
  - Inserción
  - Eliminación
    - Recorren la altura del árbol en el peor de los casos
  - Las operaciones de inserción y eliminación de un nodo son similares a las de un árbol binario de búsqueda.
  - En ambas operaciones se debe actualizar la información de la altura y realizar rotaciones si es necesario. Las rotaciones son simples reasignaciones que tienen O(1).

TIEMPO DE EJECUCIÓN – CONCLUSIONES (2)

- La inserción provoca una única reestructuración.
- La eliminación puede provocar varias reestructuraciones.
- Las operaciones son de O(log n)
- La idea de los árboles binarios de búsqueda es muy útil pero para que funcionen en todos los casos es necesario introducir condiciones de balanceo.

- ABB sin balanceo: mal eficiencia en peor caso.
- En AVL todos los casos están en O(log n) y el balanceo es poco costoso.

TRABAJO PRÁCTICO NRO. 4

- No habrá API propuesta por la cátedra para implementar.
- Será un trabajo practico mas “corto” que los previos.
- La idea es que conozcan el concepto y lo puedan interpretar gráficamente.
- No implementaremos en la práctica ejercicios que usen Arboles AVL pero si aprenderemos sus conceptos y propiedades.
- a trabajar ;-)

DUDAS / CONSULTAS ☺