Algoritmos y Estructura de Datos

Unidad 2 – Semana 12

Logro de sesión

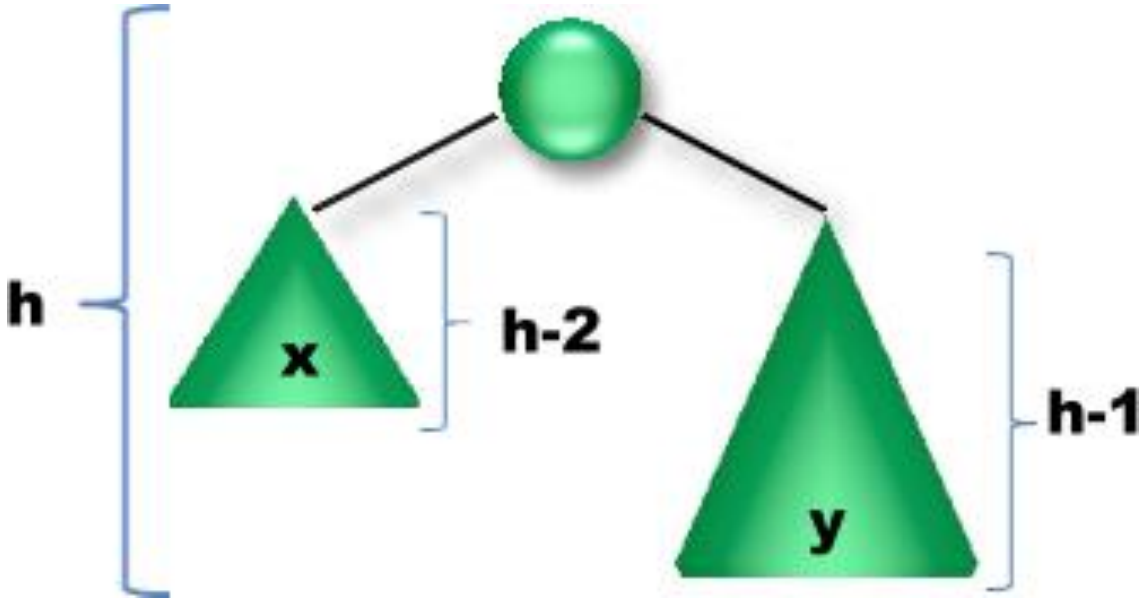
Al finalizar la sesión, el estudiante construye árboles balanceados AVL de manera experta.

Tema : Arboles AVL

Contenido:

- Introducción
- Arboles AVL (Adel’son Vel’skii and Landis

Árbol AVL



Introducción

Consideraciones:

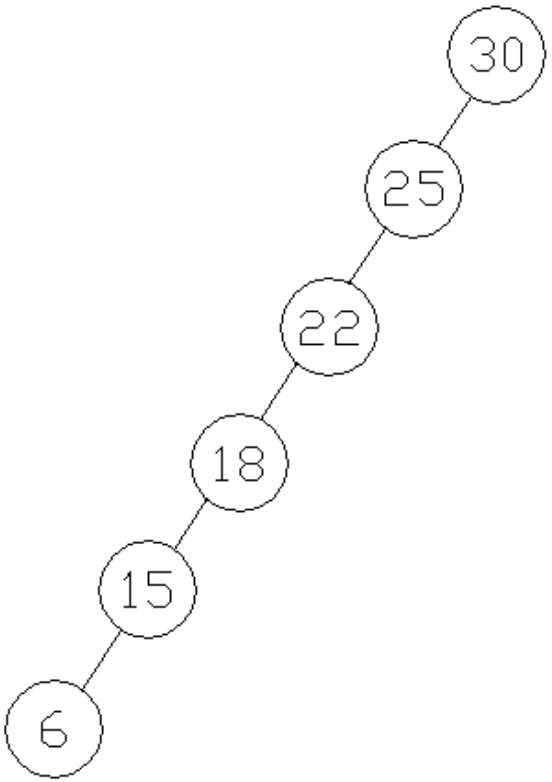
- La búsqueda en un árbol binario de búsqueda funciona eficientemente cuando los valores se han adicionado al árbol de manera aleatoria

- Si los elementos son adicionados, por ejemplo, en orden ascendente (ó descendente) el árbol queda en forma de lista enlazada, con lo cual la búsqueda sería secuencial (como en las listas)

Introducción

Ejemplo:

- Supongamos que creamos un ABB con los siguientes elementos (en ese orden)
- 30, 25, 22, 18, 15, 6



Introducción

Solución:

- **Árboles Balanceados** (árboles binarios)
- Árboles en los que la diferencia de altura por la izquierda y por la derecha es como máximo 1
- Se cumple también para todo los subárboles
- Las operaciones de Adicionar y Eliminar son complejas de implementar ya que deben mantener la propiedad de árbol balanceado

Árbol AVL

- Los árboles AVL fueron introducidos por los matemáticos G. M. Adel´son-Vel´skii and E. M. Landis en 1962 y fueron nombrados así en su honor.

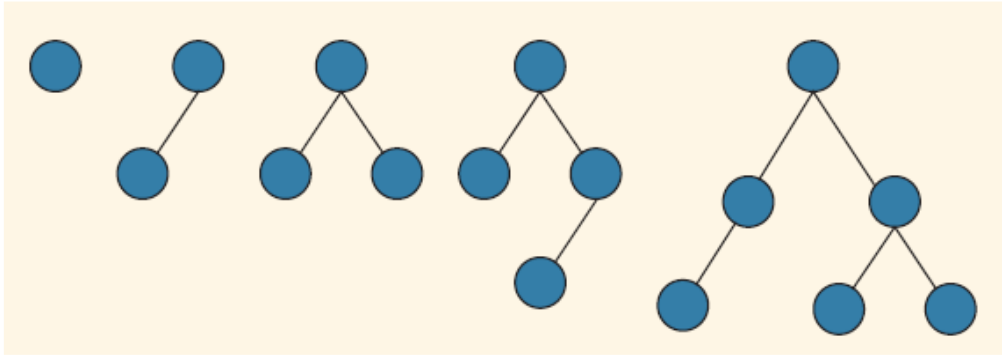
Definiciones:

- Un árbol binario perfectamente balanceado es un árbol binario tal que:
  - Las alturas de los subárboles por izquierda y derecha de la raíz son iguales.
  - Los subárboles por izquierda y derecha de la raíz son árboles perfectamente balanceados

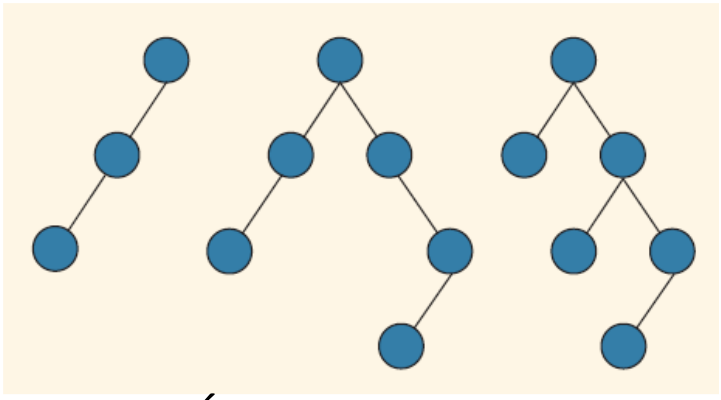
- Un árbol AVL es un árbol binario de búsqueda tal que:
  - La diferencia de la altura de los subárboles por izquierda y derecha de la raíz es máximo 1.
  - Los subárboles por izquierda y derecha de la raíz son arboles AVL.

Árbol AVL

- Todas las operaciones del árbol pueden estar realizados en el tiempo *O*(log*N*) excepto posiblemente la inserción y eliminación.



Árboles AVL



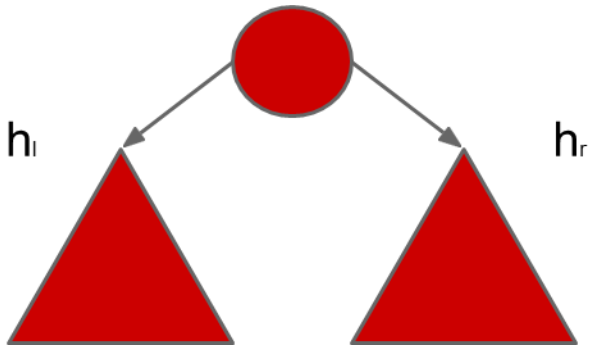
Árboles No-AVL

Árbol AVL

- Sea **x** un nodo en un arbol AVL.
- El **factor de balance** de **x**, escrito como **bf(x)**, es definido por **bf(x) = hr – hl**

  - Si la altura de **x** es superior por la izquierda, entonces **bf(x)= -1**
  - Si la altura de **x** es igual por izquierda y derecha, entonces **bf(x)=0**
  - Si la altura de **x** es superior por la derecha, entonces **bf(x)=1**

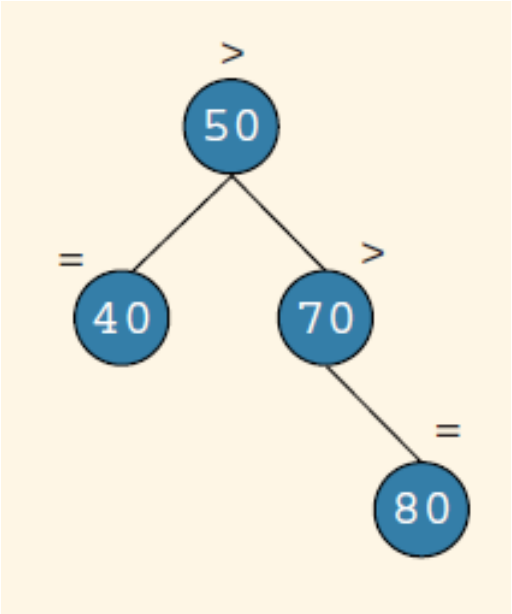
- Decimos que el nodo **x** viola el **criterio de balance** si **bf(x) > 1**, esto es si la diferencia de la altura de los subárboles de la izquierda y de la derecha es mayor a 1.



Árbol AVL

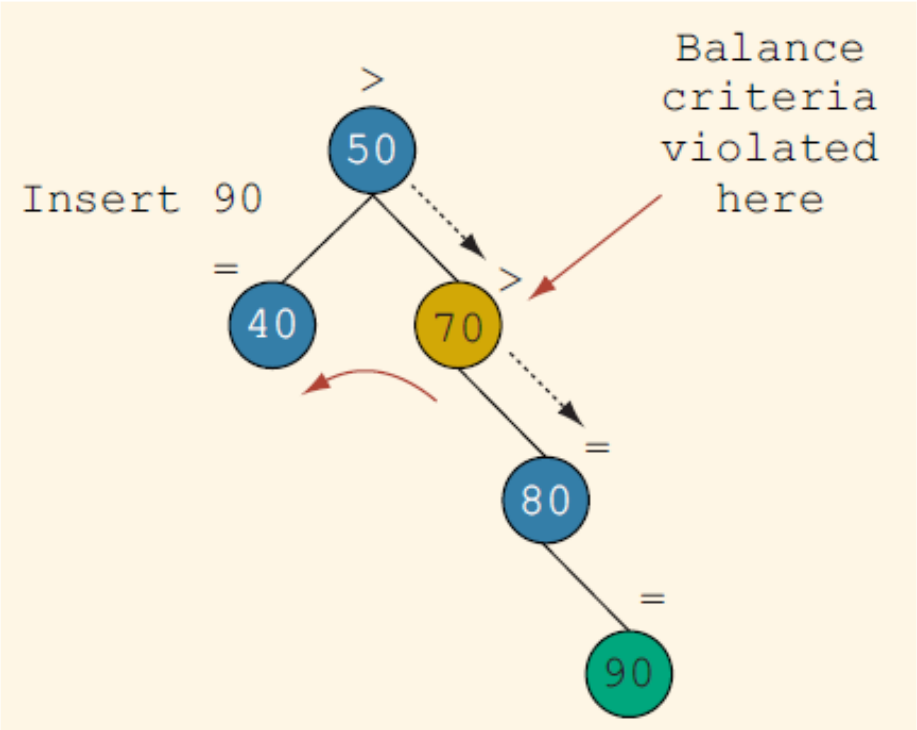
Inserción

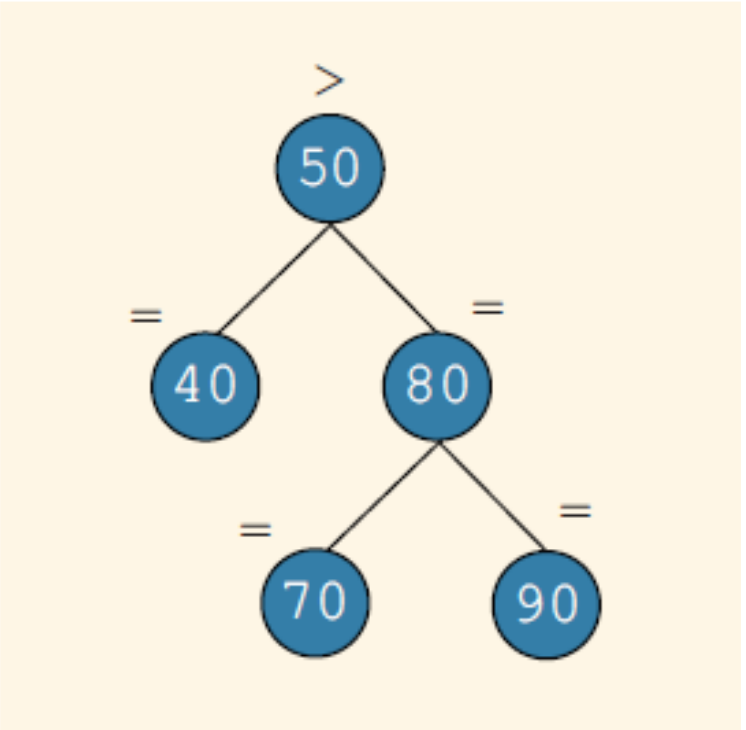
- Sea el árbol AVL de la figura siguiente, por cada nodo se muestra el elemento almacenado en el nodo, además un signo (=) en lo alto del nodo indica que el **factor de balance** de ese nodo es 0. Un signo (<) indica que el factor de balance del nodo es -1 y el signo (>) indica que el factor de balance del nodo es 1.



Árbol AVL

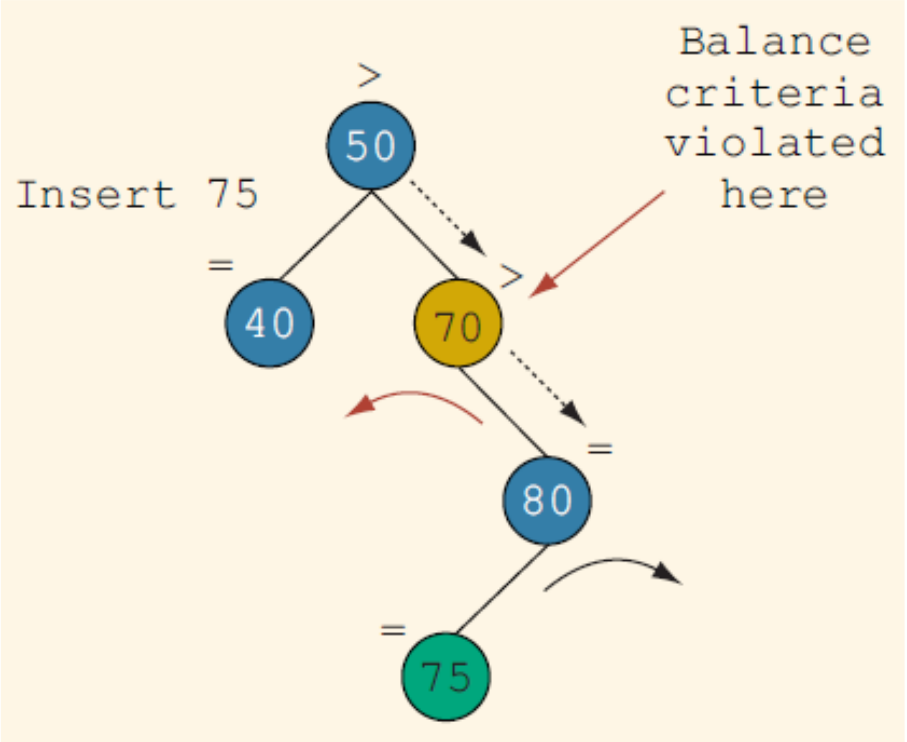
- Vamos a insertar el elemento 90 en el árbol AVL

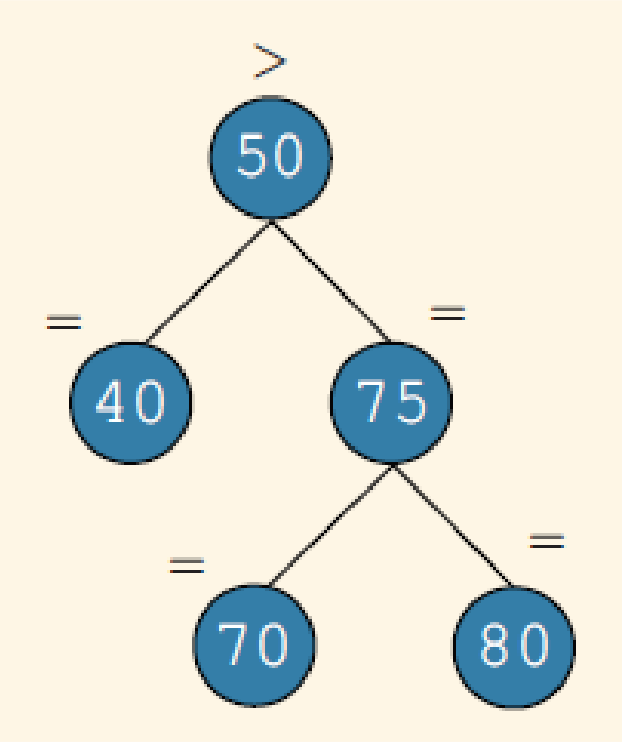




Árbol AVL

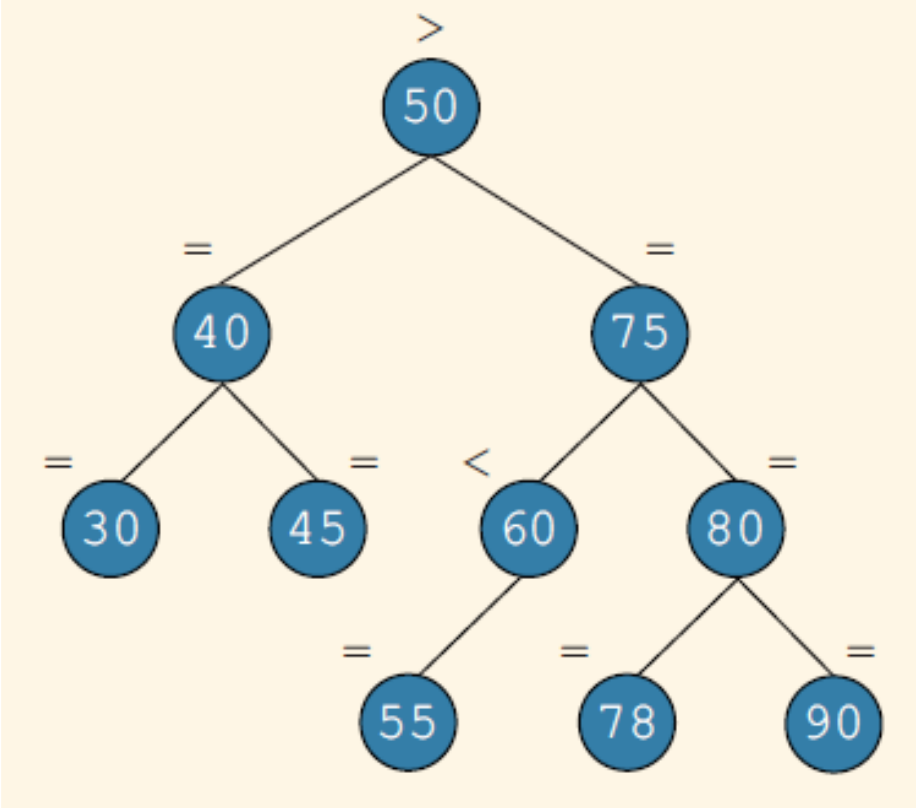
- Si hubiésemos insertado el elemento 75 en el árbol AVL





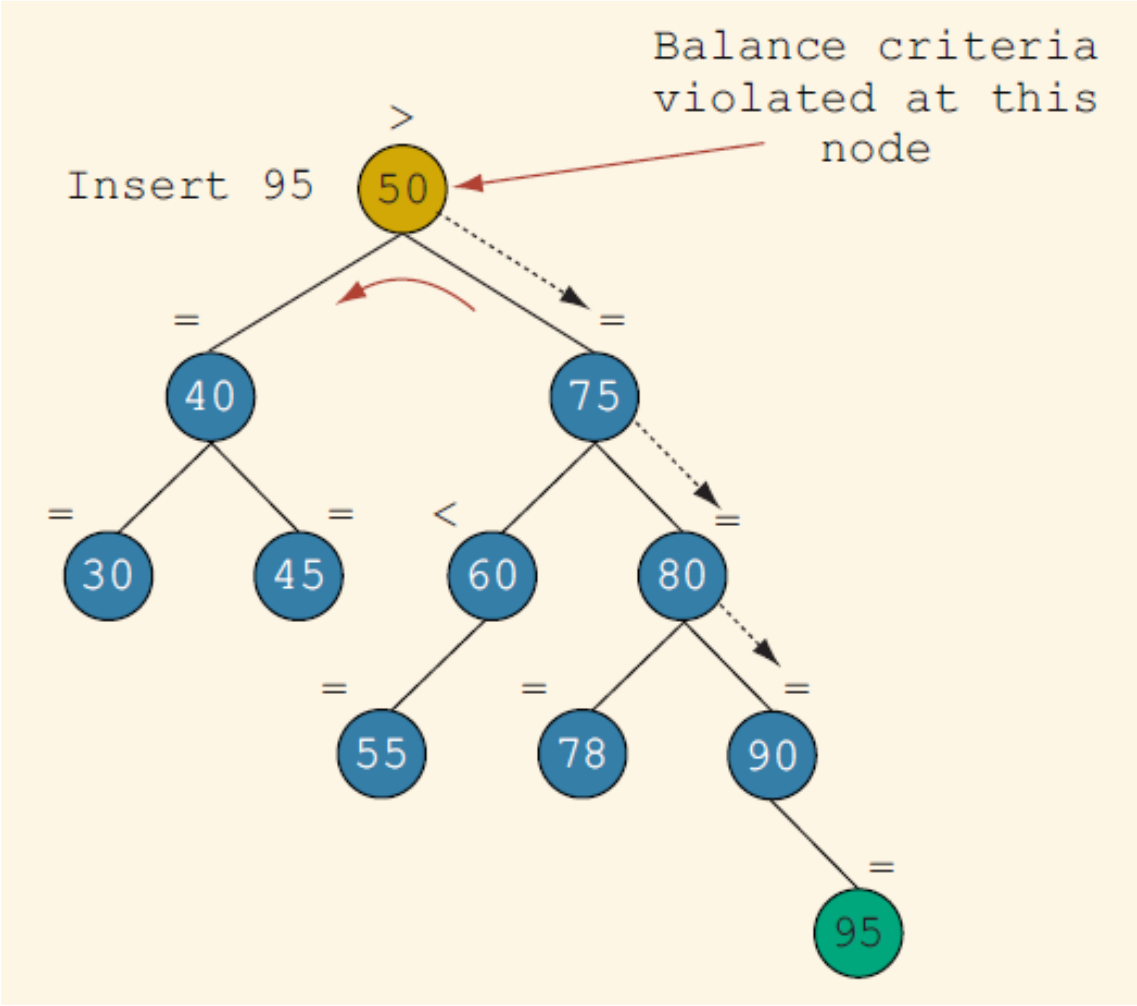
Árbol AVL

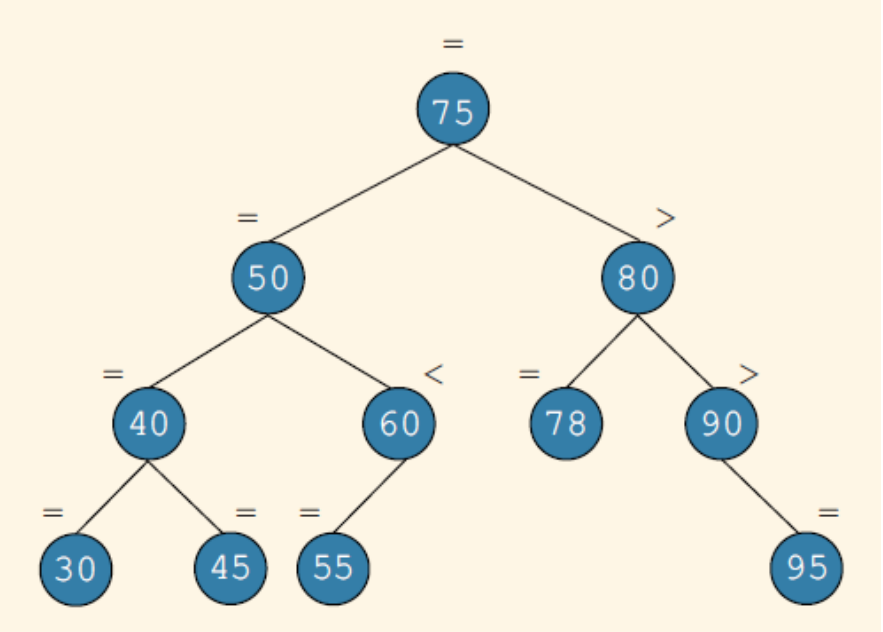
- Sea el árbol AVL siguiente:



Árbol AVL

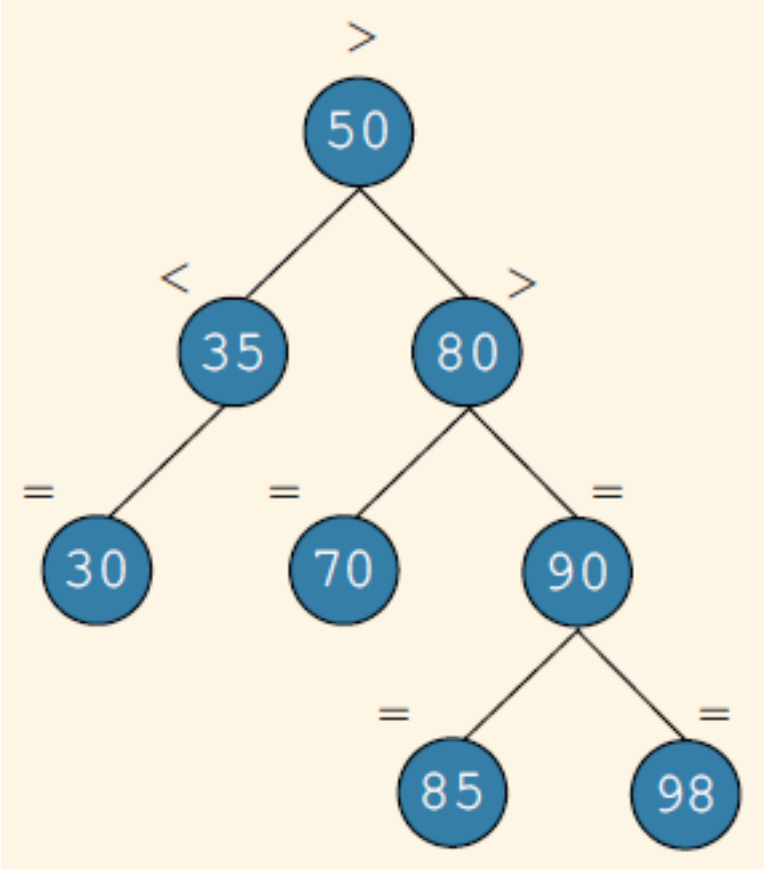
- Insertamos el elemento 95 al árbol AVL:





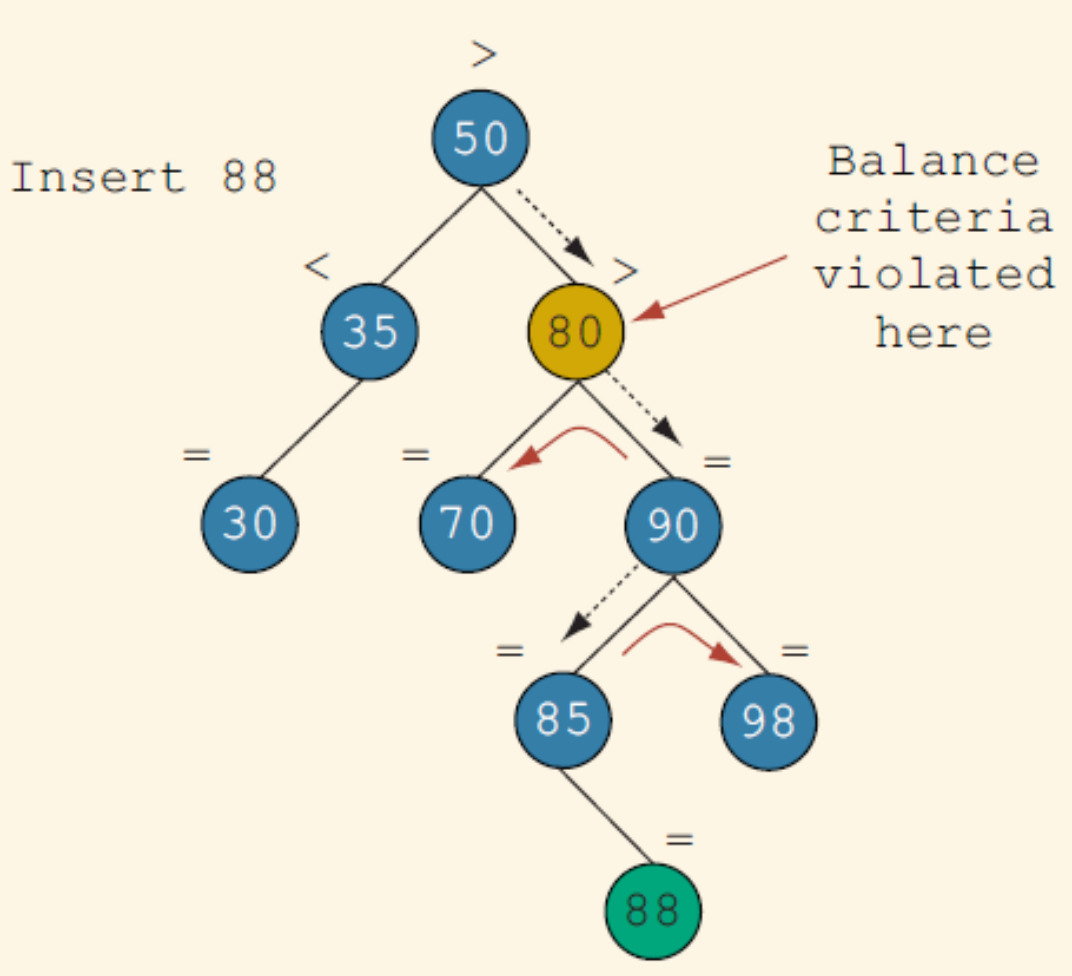
Árbol AVL

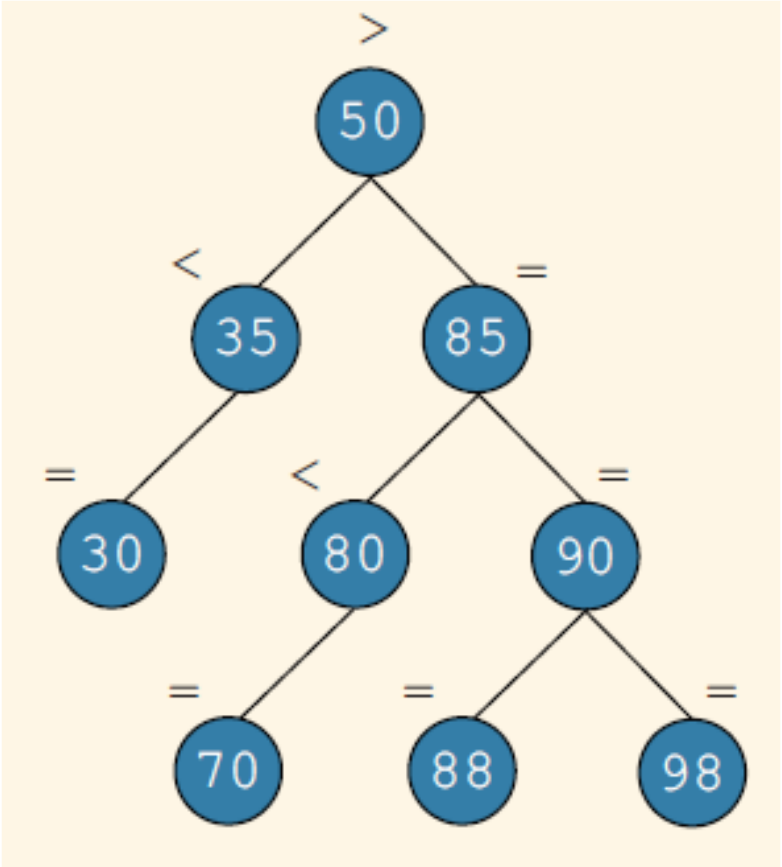
- Sea el siguiente árbol AVL:



Árbol AVL

- Insertamos el elemento 88 al árbol AVL:





Árbol AVL

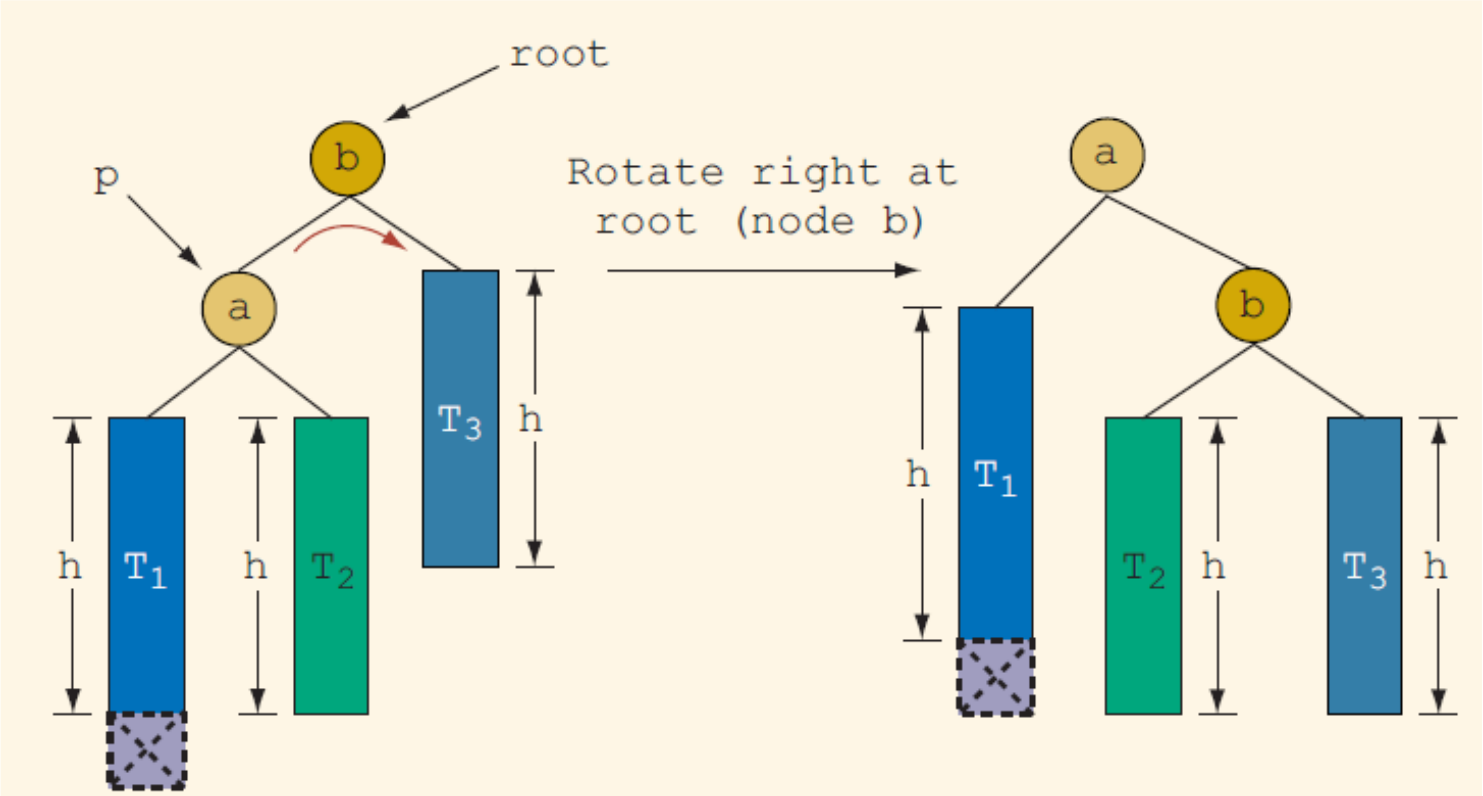
Rotaciones

- Ahora describimos el proceso de reconstrucción, llamado **rotación** del árbol.
- Son dos tipos de rotación, **rotación por izquierda** y **rotación por derecha.**

Árbol AVL

Rotaciones

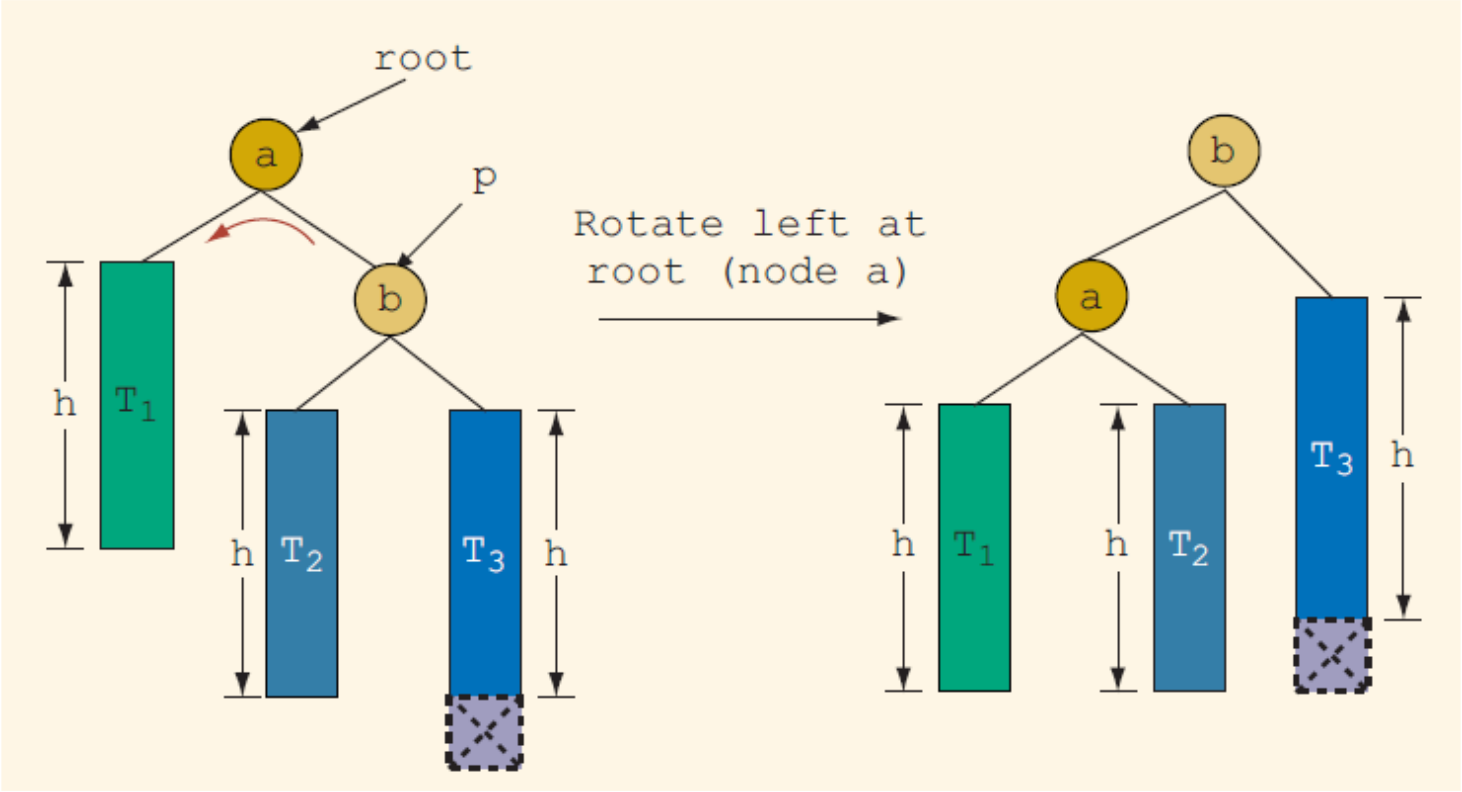
- Caso 1: **Rotación simple**, a la derecha en b



Árbol AVL

Rotaciones

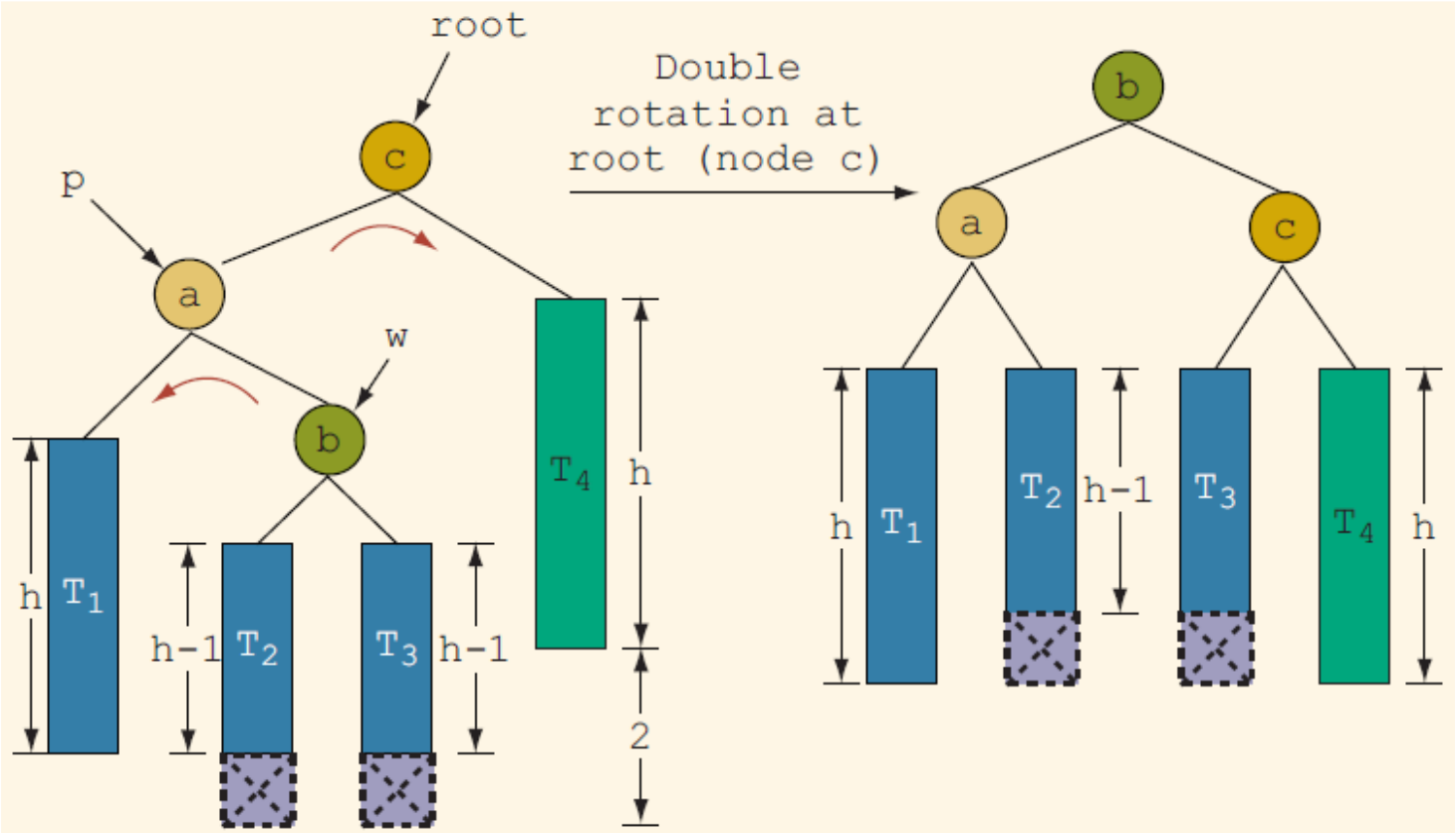
- Caso 2: **Rotación simple**, a la izquierda en a



Árbol AVL

Rotaciones

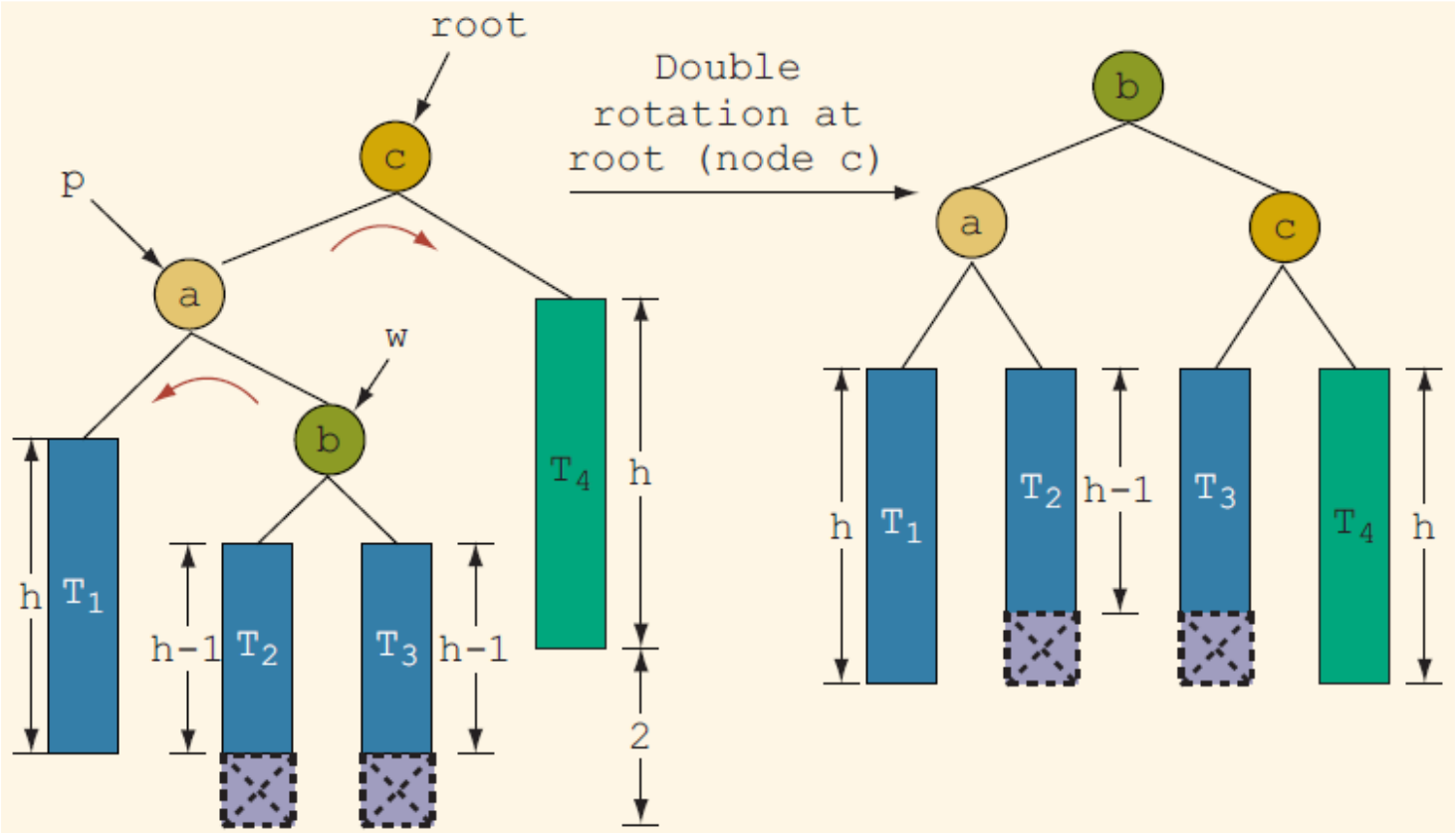
- Caso 3: **Rotación doble**, primero rota a la izquierda en a, luego rota a la derecha en c



Árbol AVL

Rotaciones

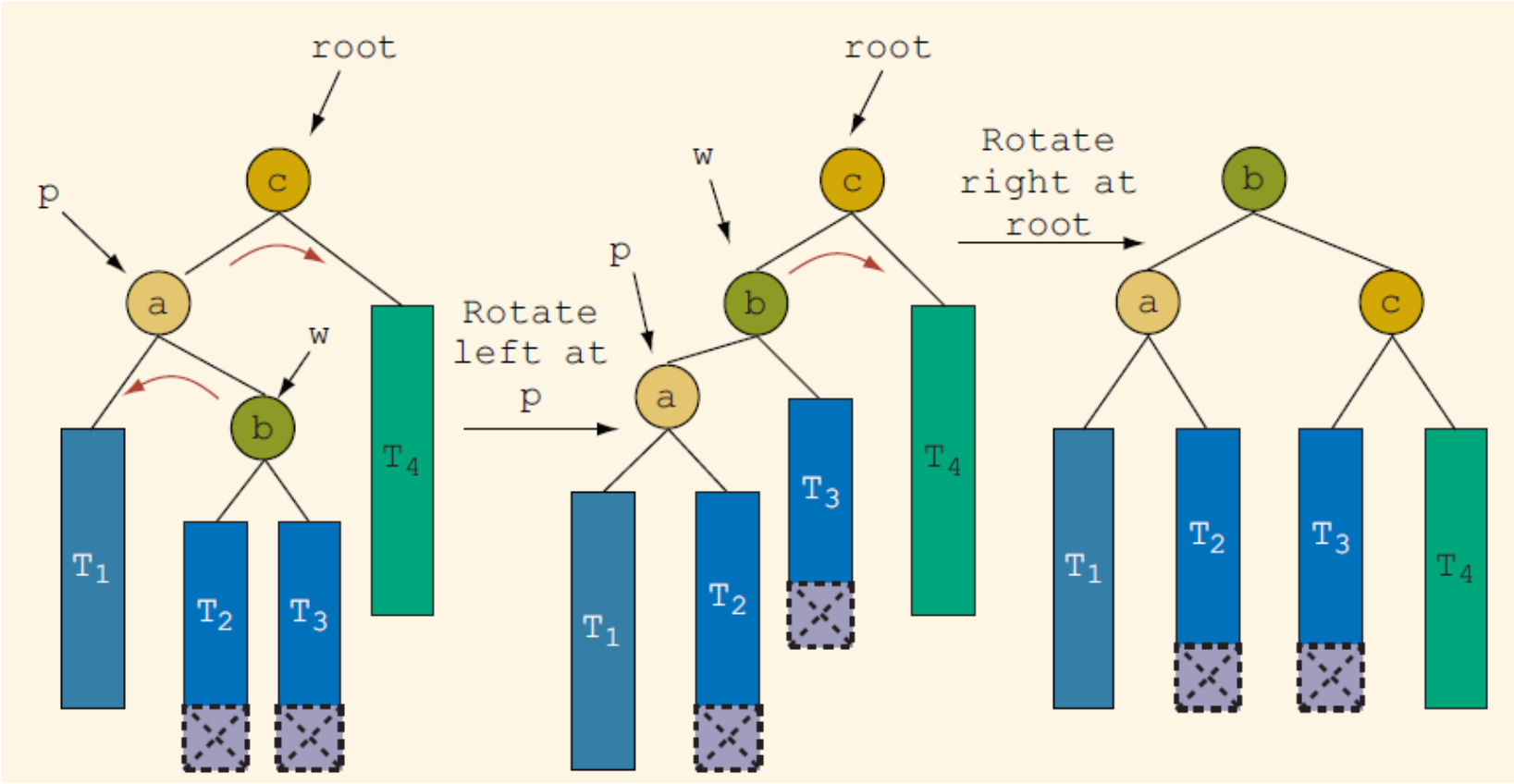
- Caso 3: **Rotación doble**, primero rota a la izquierda en a, luego rota a la derecha en c



Árbol AVL

Rotaciones

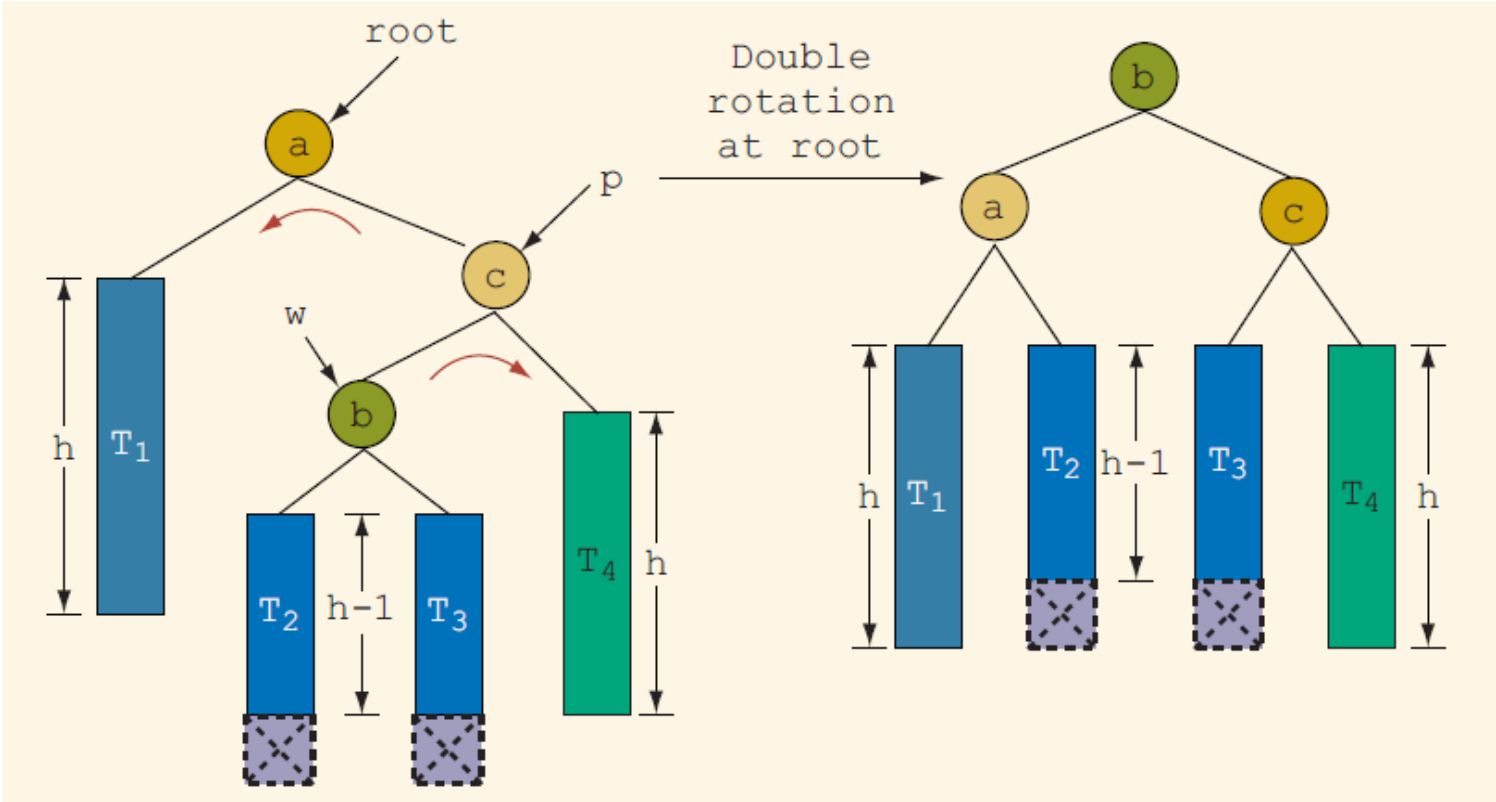
- Se muestra la secuencia entre rotaciones



Árbol AVL

Rotaciones

- Caso 4: **Doble rotación**, primero rota hacia la derecha en c, luego rota hacia la izquierda en a.



Árbol AVL

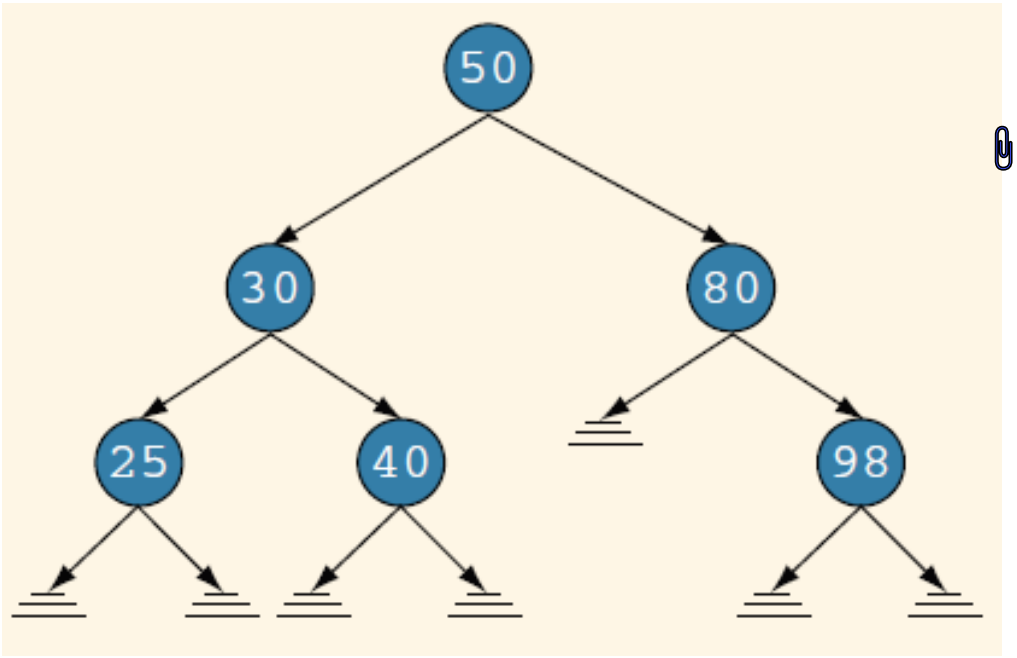
Eliminación

- Para eliminar un ítem de un árbol AVL, primero buscamos el nodo que contiene el ítem a ser eliminado.

- Los siguientes cuatro casos surgen:
  - Caso 1: El nodo a ser eliminado es una hoja.
  - Caso 2: El nodo a ser eliminado no tiene hijos por la derecha que ese, su subárbol derecho es vacío.
  - Caso 3: El nodo a ser eliminado no tiene hijos por la izquierda que ese, su subárbol izquierdo es vacío.
  - Caso 4: El nodo a ser eliminado tiene un hijo por la izquierda y un hijo por la derecha.

Ejercicios

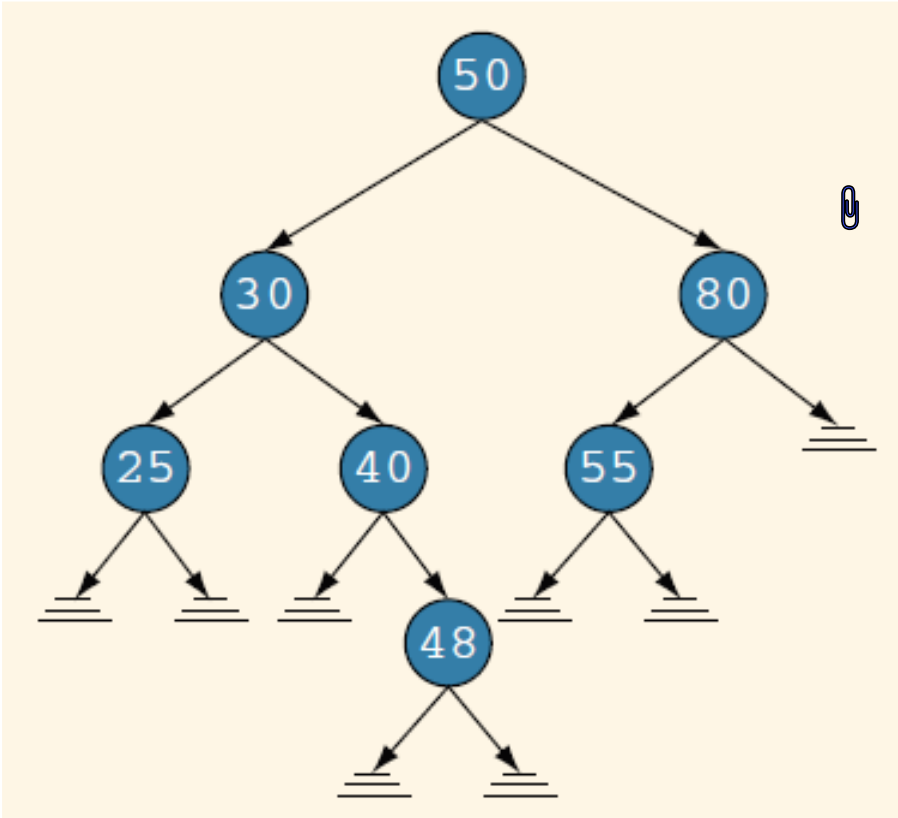
- Insertar 100 en el árbol AVL de la figura siguiente. El árbol resultante debe ser un árbol AVL. Cual es el factor de balance del nodo raíz después de la inserción?



https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Ejercicios

- Insertar 45 en el árbol AVL de la figura siguiente. El resultado del árbol resultante debe ser un árbol AVL. Cual es el factor de balance del nodo raíz después de la inserción?



Ejercicios

- Construir el arbol AVL con los siguientes valores: 15, 20, 24, 10, 13, 7, 30, 36, 25

- Escribir un programa para implementar las clases de un árbol AVL como un ADT. Implementar las operaciones de insertar y eliminar.

- Demostrar que todas las rotaciones anteriores no alteran la propiedad BST (Árbol binario de búsqueda) del árbol. En otras palabras, que al realizar un recorrido transversal **En Orden**, el orden se mantiene.

Referencias

- Sedgewick, R., et. al. (2011) Algorithms, Fourth Edition. Pearson.
- Cormen, H., et. al. (2009) Introduction to Algorithms, MIT Press.
- Allen, Mark (2014) Data Structures and Algorithms Analysis in C++, Fourth Edition. Pearson.

EXIGETE INNOVA