DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Análisis de Algoritmos y Estructura de Datos

TDA árbol AVL

Prof. Violeta Chang C

Semestre 2 – 2023

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

TDA árbol AVL

- **Contenidos**:
  - Estructura de datos TDA árbol AVL
  - Operaciones de TDA árbol AVL
- **Objetivos**:
  - Entender impacto del balance en el desempeño de los árboles binarios de búsqueda (ABB)
  - Comprender estructura de datos de TDA árbol AVL
  - Conocer operaciones básicas de TDA árbol AVL y comprender mecanismo de recuperación de balance

Ruta de la sesión

TDA árbol AVL

Operaciones de TDA árbol AVL

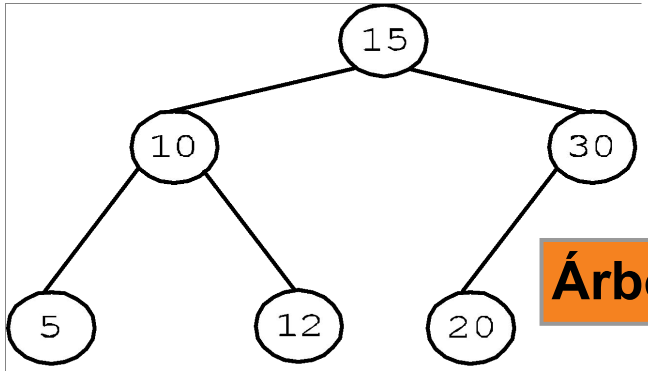
Análisis de TDA árbol AVL

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

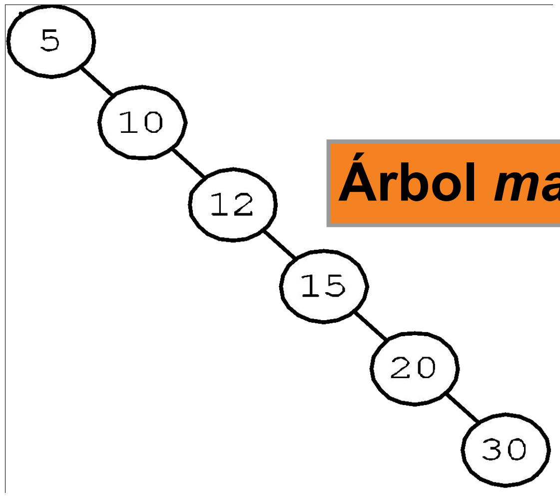
TDA árbol AVL

Árboles auto-balanceados

- Un árbol binario de búsqueda (ABB) puede tener tan mal balance, que el tiempo de búsqueda de $O(\log_2 n)$ en el mejor caso, sea $O(n)$ en el peor caso



Árbol *bueno*!



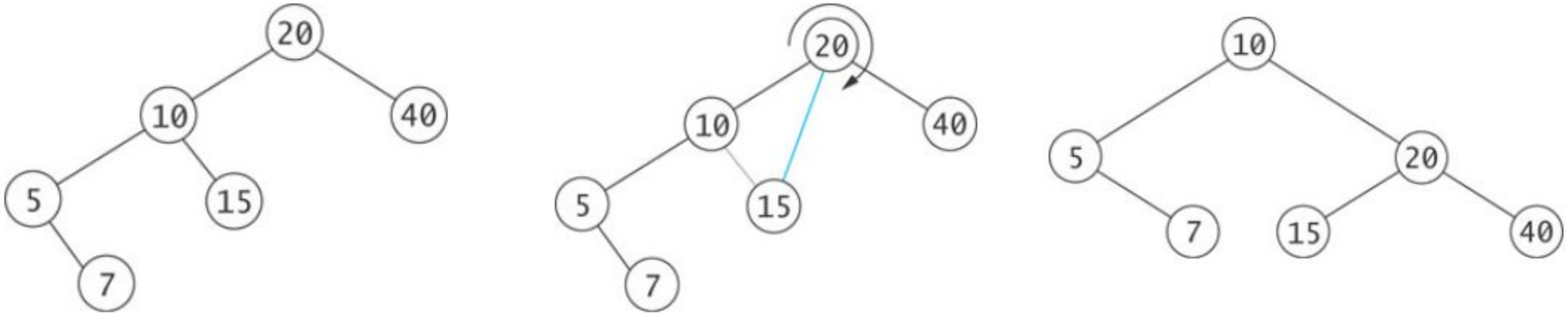
Árbol *malo*!

Mismo número de elementos en cada árbol
pero tiempos de búsqueda muy diferentes (en el peor caso)

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Árboles auto-balanceados

- Un árbol perfectamente balaceado es lo ideal. Un árbol binario completo con *n* nodos tiene una altura de $\lceil \log_2 n \rceil$
- Una operación de *insertar()* o *eliminar()* realizada en un árbol binario completo, puede cambiar la estructura del árbol y se necesitaría mucho trabajo para restaurarla → deberían cambiar las alturas relativas de los subárboles izquierdo y derecho, pero preservando la propiedad básica de un ABB... ***rotación***



TDA árbol AVL

- **Estructura de datos**
  - Un árbol AVL es un ABB con las siguientes restricciones:
    - La **altura de los subárboles izquierdo y derecho de un nodo difieren en a lo más 1**
    - Los **subárboles izquierdo y derecho con un único nodo son árboles AVL**
  - Características:
    - Efectivo en costo
    - Garantía que la altura del árbol es un múltiplo pequeño de $\log_2 n$, por lo que el tiempo de acceso es $O(\log_2 n)$

* Adelson-Velskii y Landis

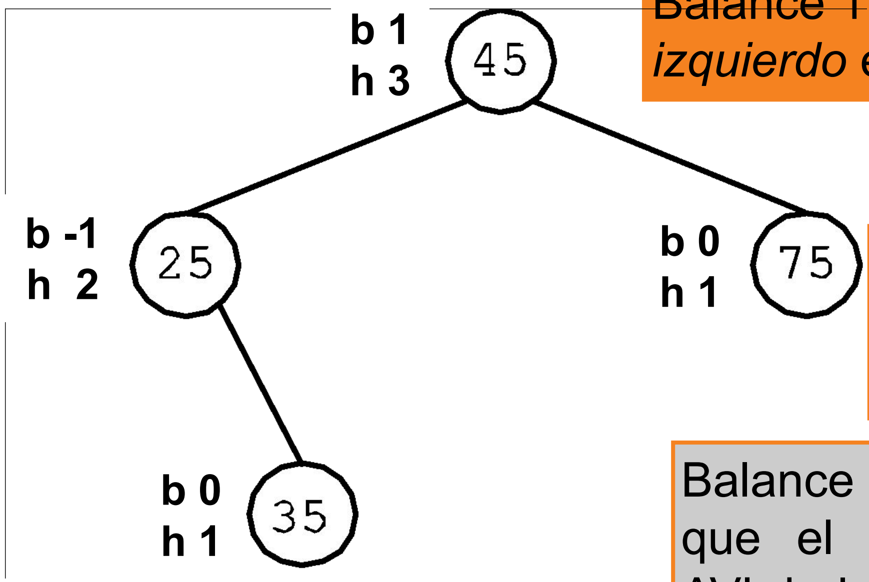
TDA Árbol AVL

- **Operaciones**
  - TDA árbol
  - **insertarNodo(T,x)**: inserta el nodo x en el árbol AVL T
  - **eliminarNodo(T,x)**: elimina el nodo x del árbol AVL T
  - **buscarDato(T,d)**: busca el nodo con dato d en árbol AVL T

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Altura y balance de árbol AVL

- El cálculo de altura y balance es realizado desde las hojas hacia arriba:
  - Un subárbol vacío tiene altura 0
  - Una hoja siempre tiene balance 0 y altura 1
  - altura de un nodo = (altura de su subárbol más grande) + 1
  - balance de un nodo = altura de subárbol izquierdo – altura de subárbol derecho



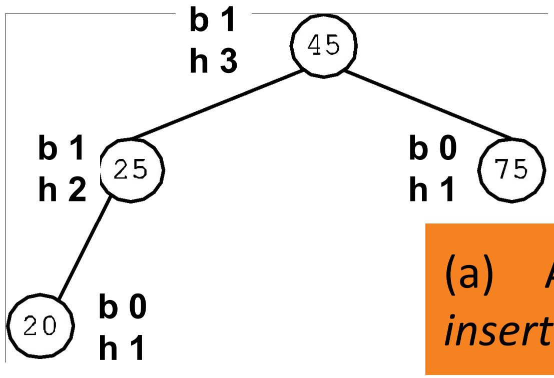
Balance 1 significa que el subárbol *izquierdo* es más profundo

Balance -1 significa que el subárbol *derecho* es más profundo

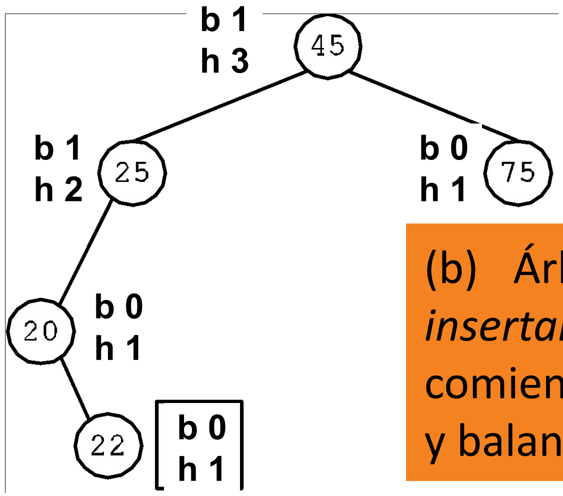
Balance 0 significa que los subárboles tienen la misma altura

Balance 2 o -2 significa que el árbol ya no es AVL-balanceado

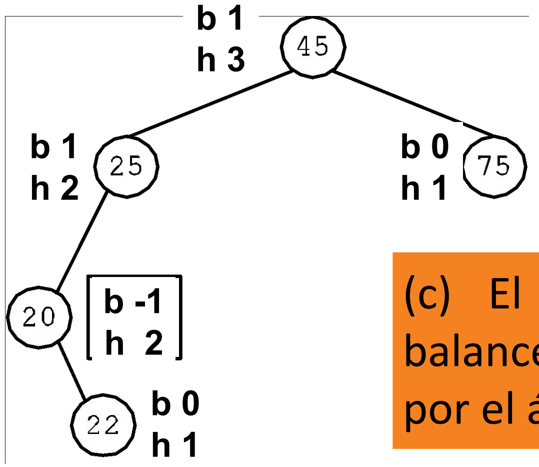
Detección de desbalance



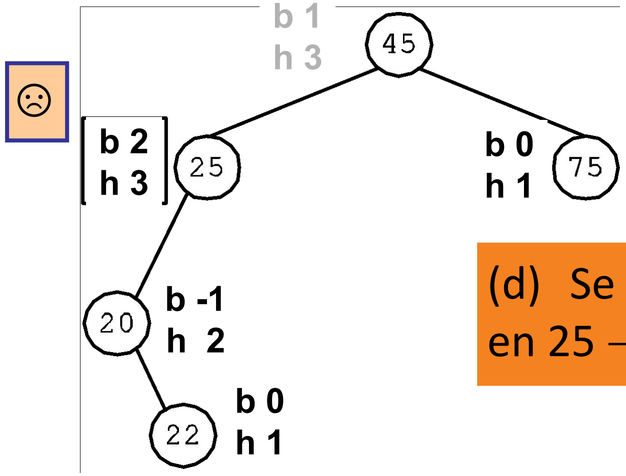
(a) Árbol AVL antes de *insertar*(22)



(b) Árbol AVL después de *insertar*(22) → Ahora comienza recálculo de alturas y balances



(c) El cálculo de alturas y balances se realiza subiendo por el árbol



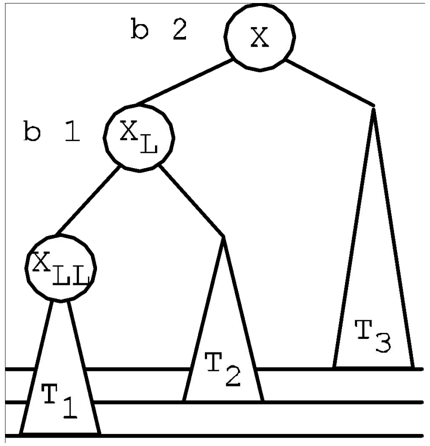
(d) Se encuentra desbalance en 25 → balance es 2

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

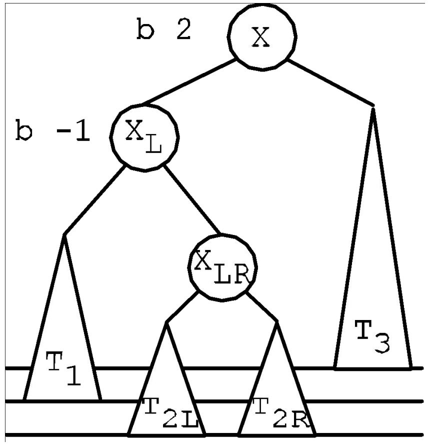
Tipos de desbalance

- Existen 4 casos de desbalance, respecto del nodo X:

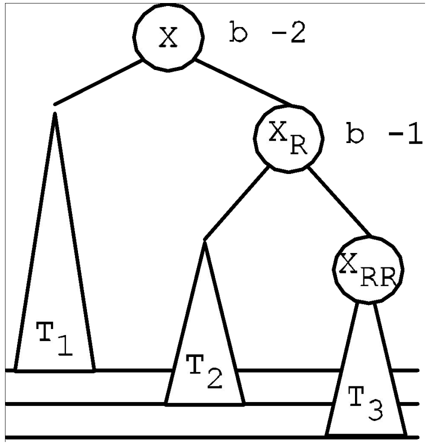
- LL (left-left)



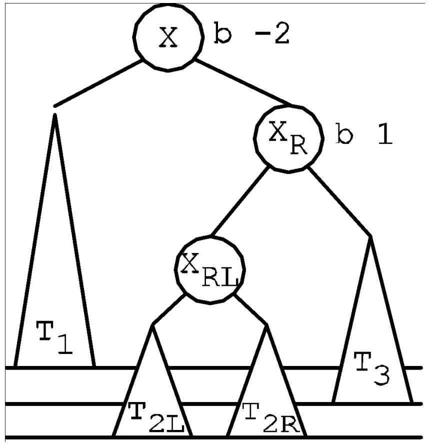
LR (left-right)



- RR (right-right)



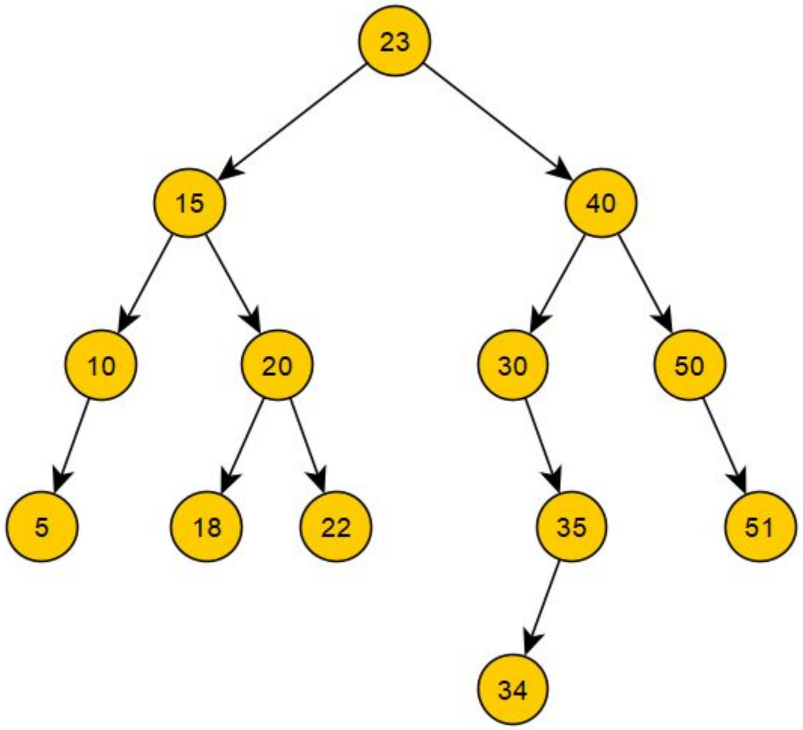
RL (right-left)



DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Recuperación de balance: Rotaciones

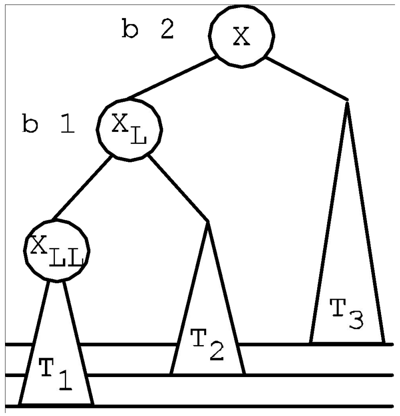
- Después de realizar operaciones de actualización en árbol AVL:
  - Calcular recursivamente balance desde hoja hacia raíz
  - Si se encuentra desbalance → determinar tipo de desbalance
  - Recuperar balance usando **rotaciones de subárboles**



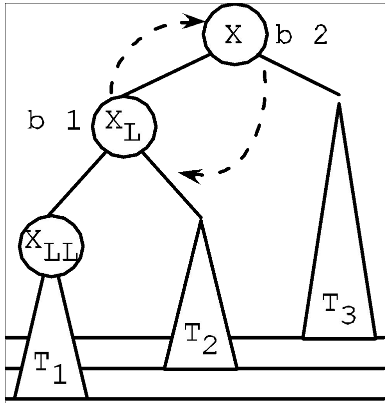
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Rotación a la derecha

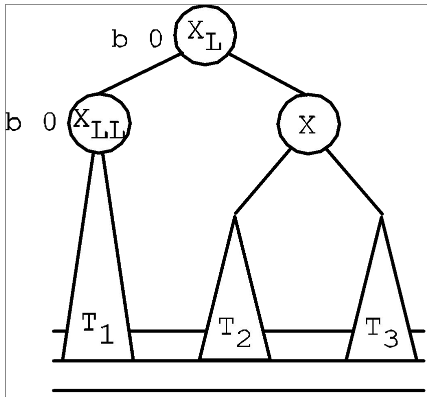
- **Caso de desbalance LL**
- **Solución**:
  - Una rotación simple a la derecha (sentido horario) alrededor de X, moviendo el árbol con raíz X_L un nivel más arriba, tal que:
  - X se convierta en el hijo derecho de X_L
  - El hijo derecho de X_L (T_2) se convierte en el hijo izquierdo de X
  - X_L es la nueva raíz del subárbol



(a) Desbalance LL en X



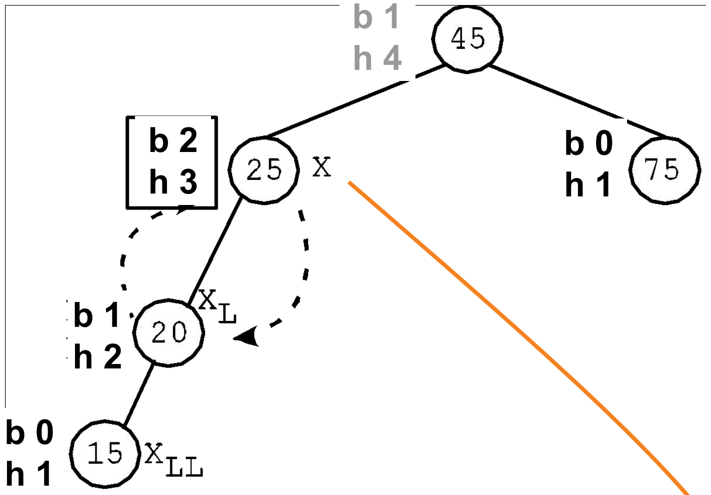
(b) Rotación a la derecha



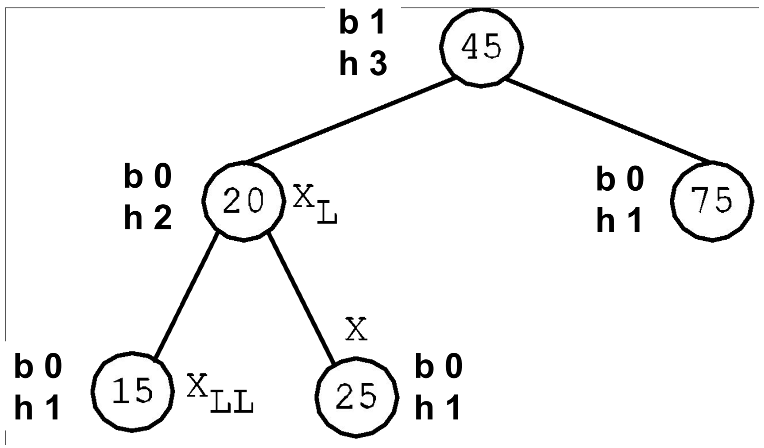
(c) Balance recuperado

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Rotación a la derecha: Ejemplo



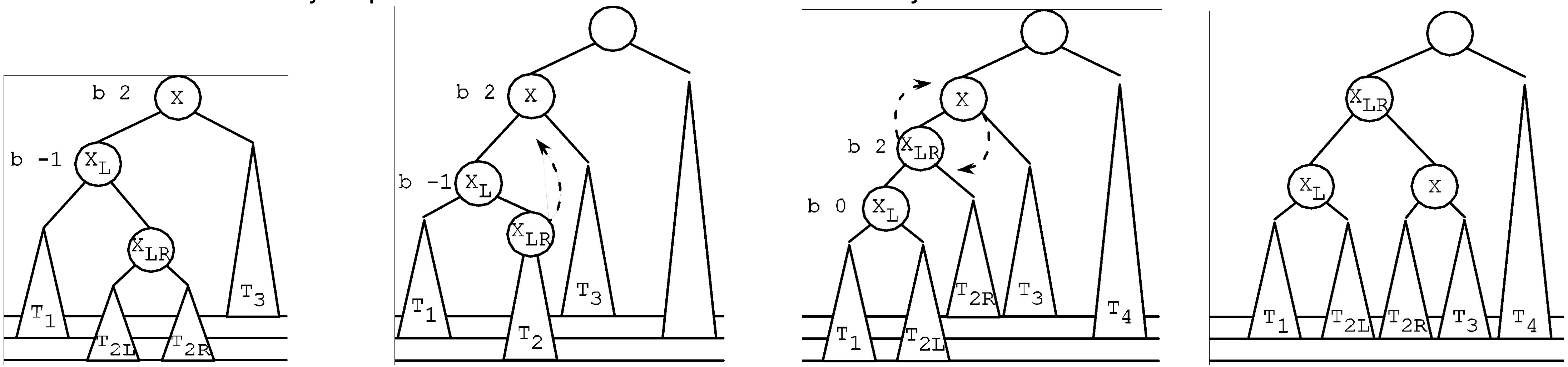
(a) Árbol AVL con desbalance LL en X



(b) Árbol AVL después de rotación a la derecha alrededor de X

Rotación doble izq-der

- **Caso de desbalance LR**
- **Solución**:
  - Una rotación a la izquierda (sentido antihorario) de X_{LR} alrededor de X, moviendo el hijo derecho de X_L (X_{LR}) un nivel más arriba
  - El hijo izquierdo de X_{LR} (T_{2L}) se convierte en el subárbol derecho de X_L
  - El hijo derecho de X_{LR} (T_{2R}) sube un nivel en el árbol con X_{LR}
  - El resultado de esta rotación a la izquierda no recupera el balance de X
  - Una rotación a la derecha (sentido horario) alrededor de X
  - Esto mueve el hijo izquierdo de X un nivel más arriba mientras baja un nivel a X



(a) Desbalance LR en X

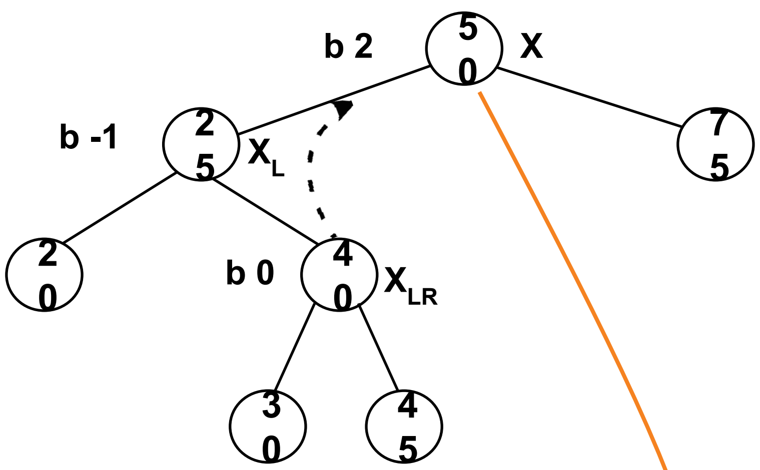
(b) Rotación a la izquierda

(c) Rotación a la derecha

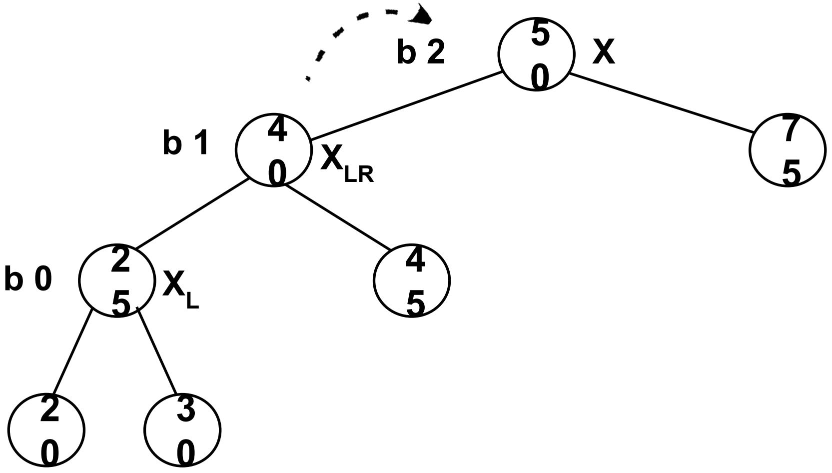
(d) Balance recuperado

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

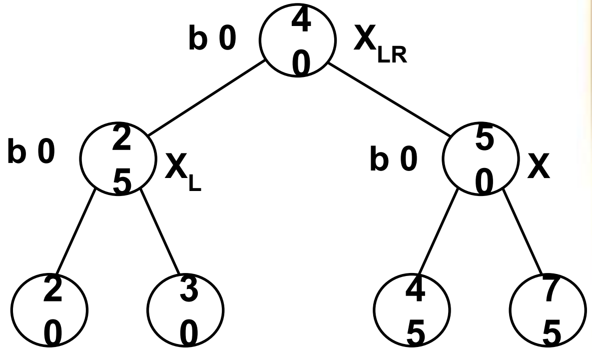
Rotación izq-der: Ejemplo



(a) Árbol AVL con desbalance LR en X



(b) Árbol AVL después de rotación a la izquierda de X_{LR} alrededor de X_L



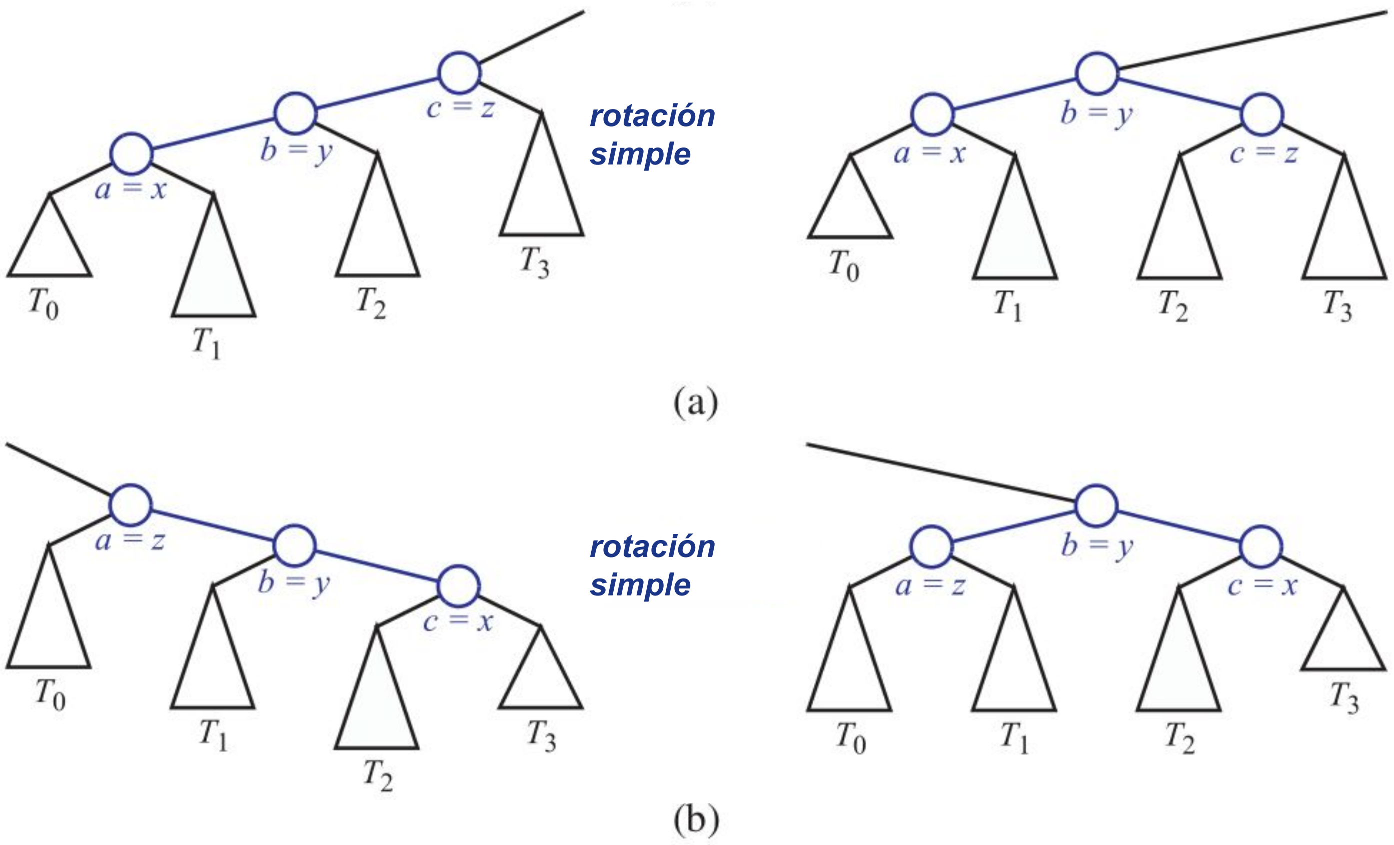
(c) Árbol AVL después de rotación a la derecha alrededor de X

Recuperación de balance: Rotaciones

- Si se encuentra desbalance ⟶ se recupera balance usando **rotaciones de subárboles**
- Existen 4 casos de desbalance:

  a) **Caso LL** (Left-Left): rotación a la derecha de *y* sobre *z*

  b) **Caso RR** (Right-Right): rotación a la izquierda de *y* sobre *z*

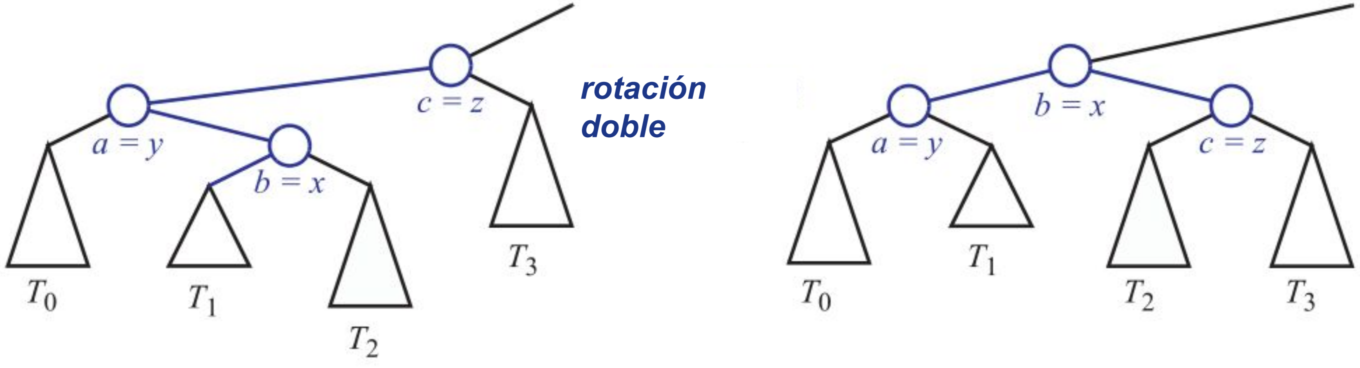


Recuperación de balance: Rotaciones

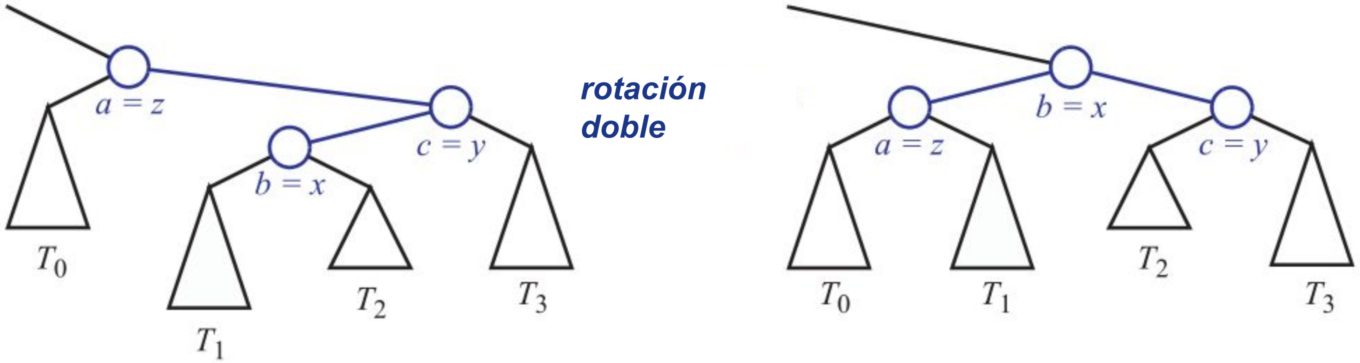
- Si se encuentra desbalance → se recupera balance usando **rotaciones de subárboles**
- Existen 4 casos de desbalance:

  c) **Caso LR** (Left-Right): rotación a la izquierda de *x* sobre *y* y luego rotación a la derecha sobre *z*

  d) **Caso RL** (Right-Left): rotación a la derecha de *x* sobre *y* y luego rotación a la izquierda sobre *z*



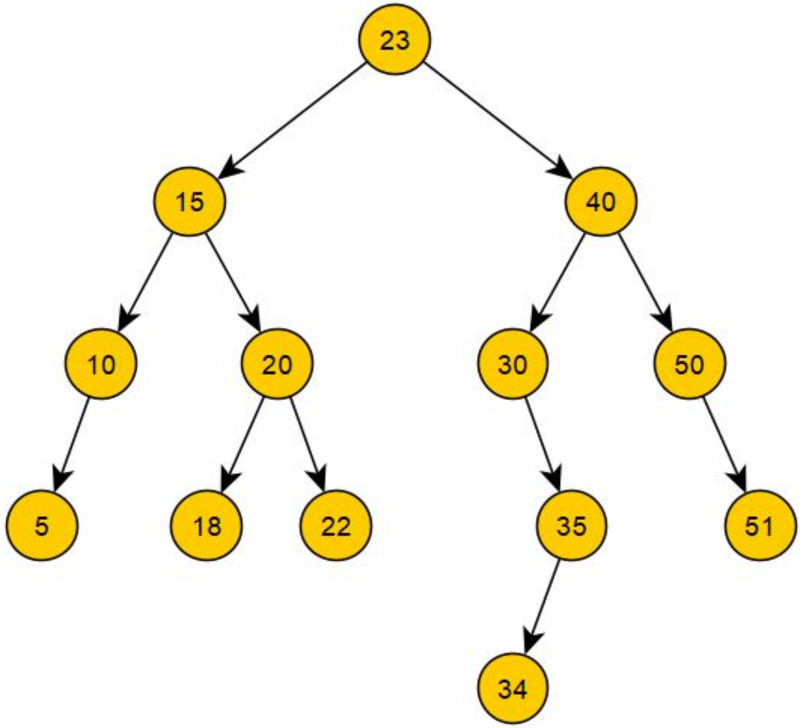
(c)



(d)

Recuperación de balance: Rotaciones

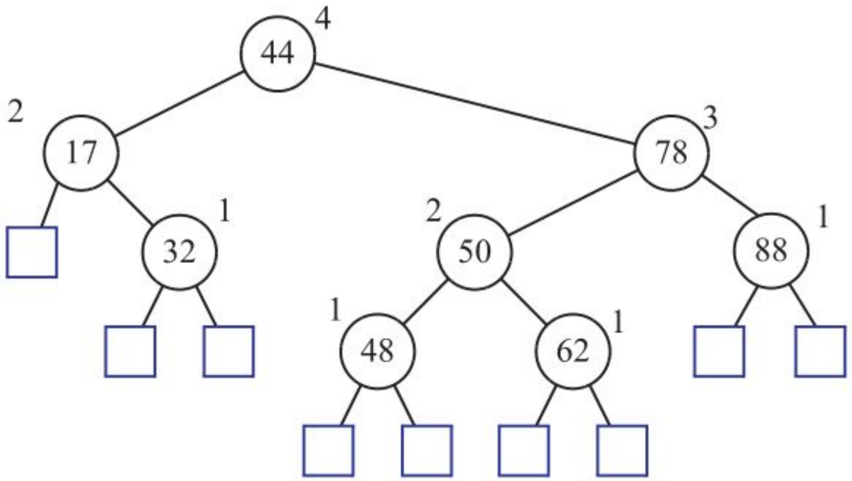
- Después de realizar operaciones de actualización en árbol AVL:
  - Calcular recursivamente balance desde hoja hacia raíz
  - Si se encuentra desbalance → determinar tipo de desbalance
  - Recuperar balance usando rotaciones de subárboles



Inserción en árbol AVL

- Muy parecida a **inserción** en ABB
- Se debe descender por el árbol recursivamente, buscando la posición hoja en la que se debe insertar el nuevo valor
- Diferencia:
  - Conforme avance la recursión, se necesitarán recalcular alturas y balances
  - Si se descubre un desbalance, es necesario identificar su tipo (LL, RR, LR, RL) y restaurar el balance según él

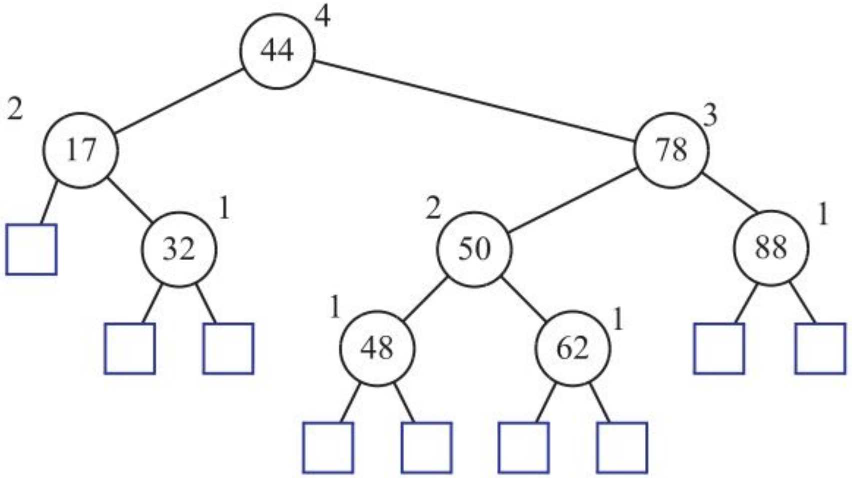
Inserción en árbol AVL - Ejemplo



Insertar 54

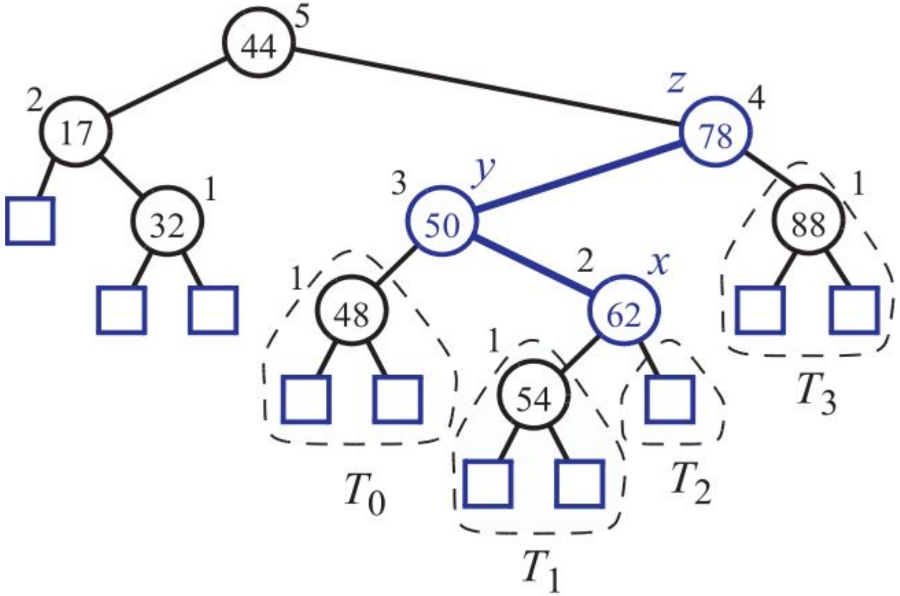
Árbol AVL original

Inserción en árbol AVL - Ejemplo

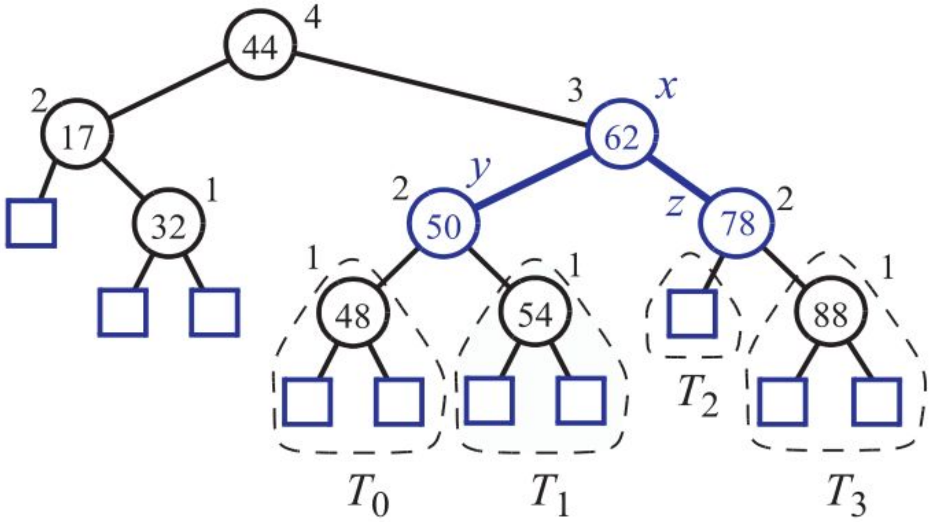


Insertar 54

Árbol AVL original



Inserción de nodo con dato 54. Después de insertar un nuevo nodo para 54, los nodos con datos 78 y 44 se desbalancean

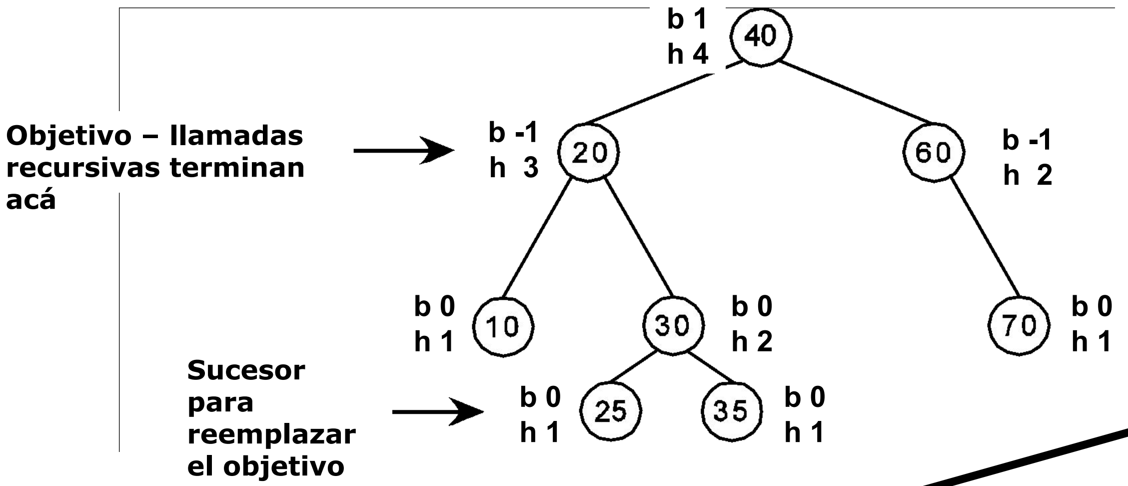


Se recupera balance según caso LR: rotación a la izquierda de *x* sobre *y* y luego rotación a la derecha sobre *z*

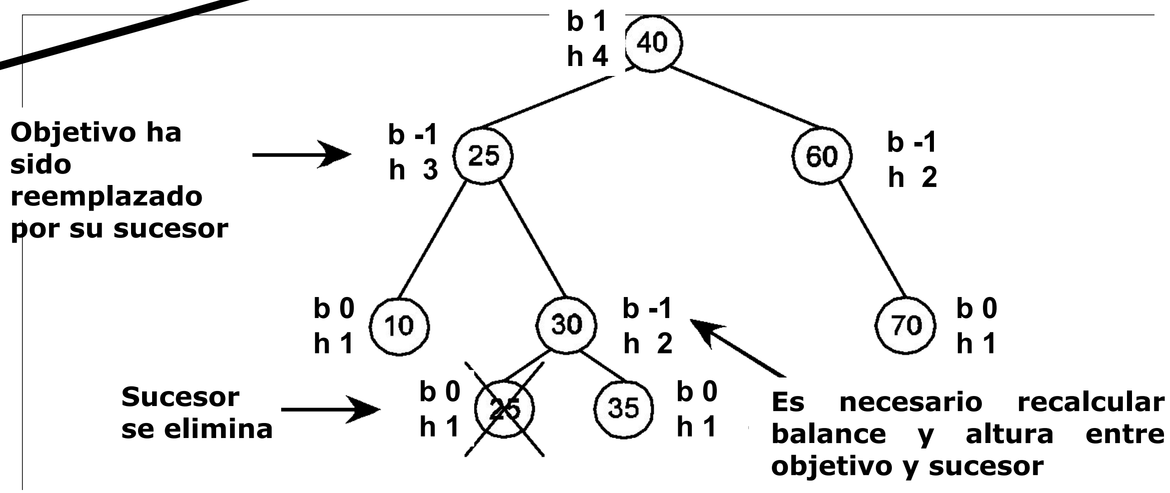
Eliminación en árbol AVL

- Similar a **eliminación** en ABB, un poco más complicada
- Se debe reemplazar el objetivo por su sucesor
- Complicación:
  - Conforme avance la recursión, se necesitarán recalcular alturas y balances
  - PERO las llamadas recursivas no habrán descendido hasta el nodo sucesor
  - Entonces, cómo recalcular las alturas y balances desde el nodo sucesor hacia arriba?

Eliminación en árbol AVL: Ejemplo



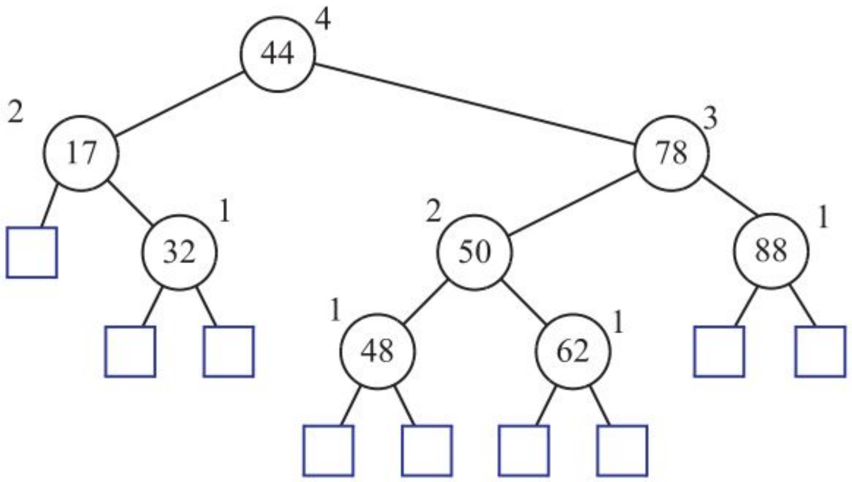
(a) Las llamadas recursivas para remover() llegarán hasta el nodo objetivo (20). El sucesor del objetivo (25) está más abajo en el árbol que el objetivo



(b) Se necesita de un mecanismo adicional para ajustar el árbol entre el sucesor (después de eliminarlo) y el objetivo

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

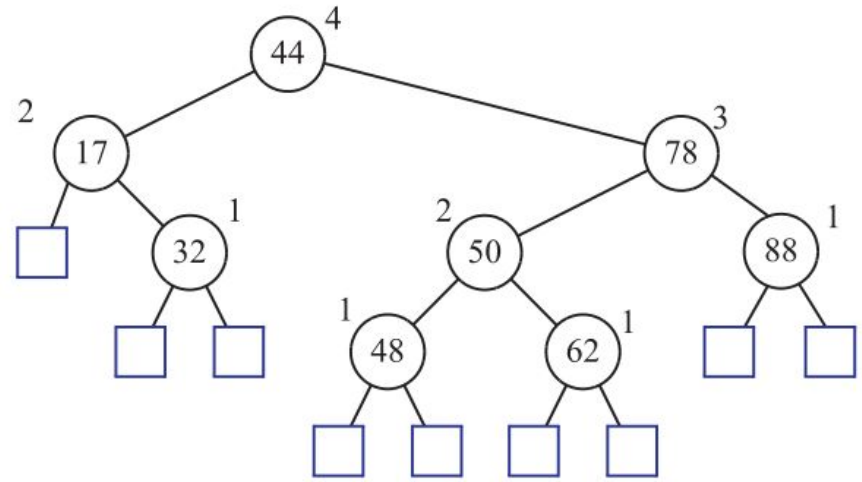
Eliminación en árbol AVL - Ejemplo



Eliminar 32

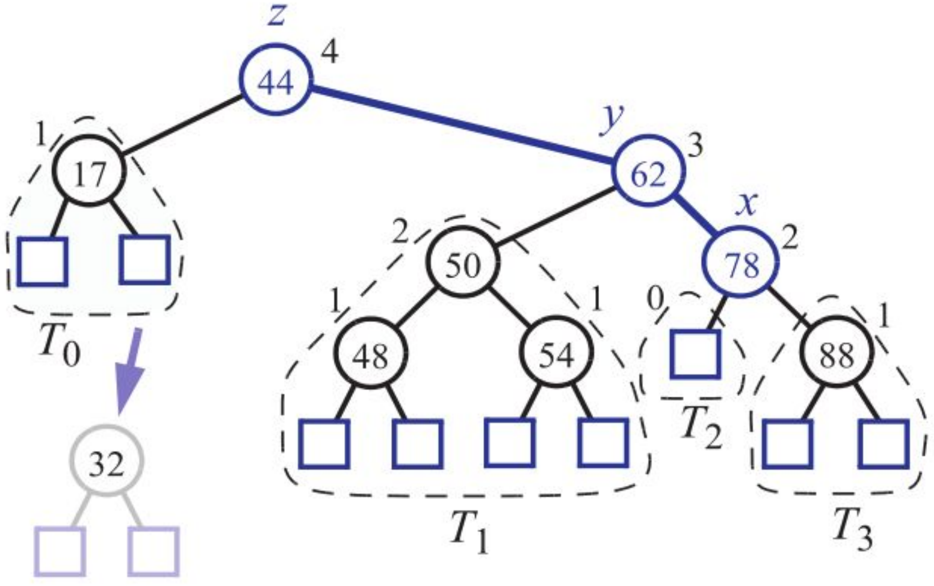
Árbol AVL original

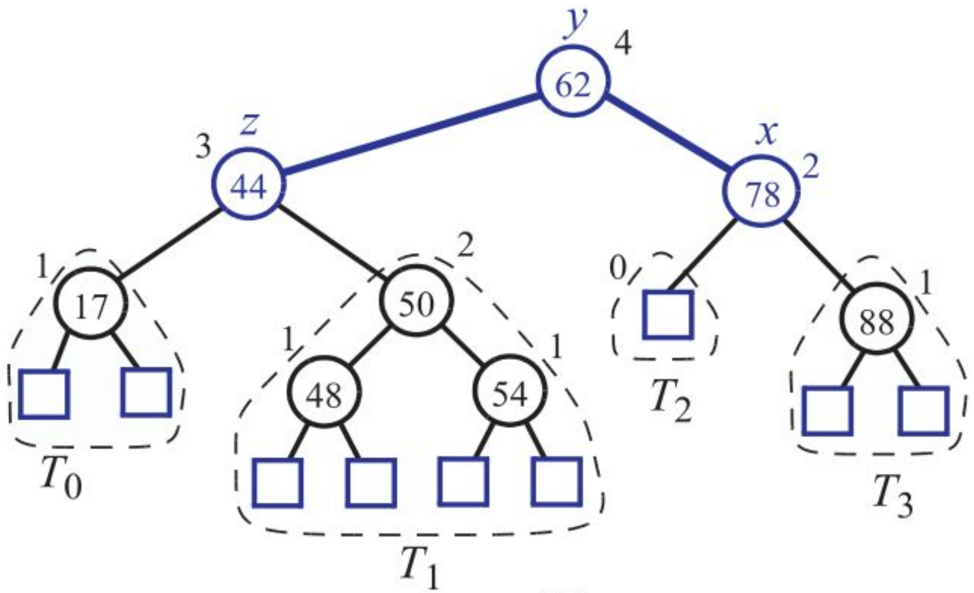
Eliminación en árbol AVL - Ejemplo



Eliminar 32

Árbol AVL original





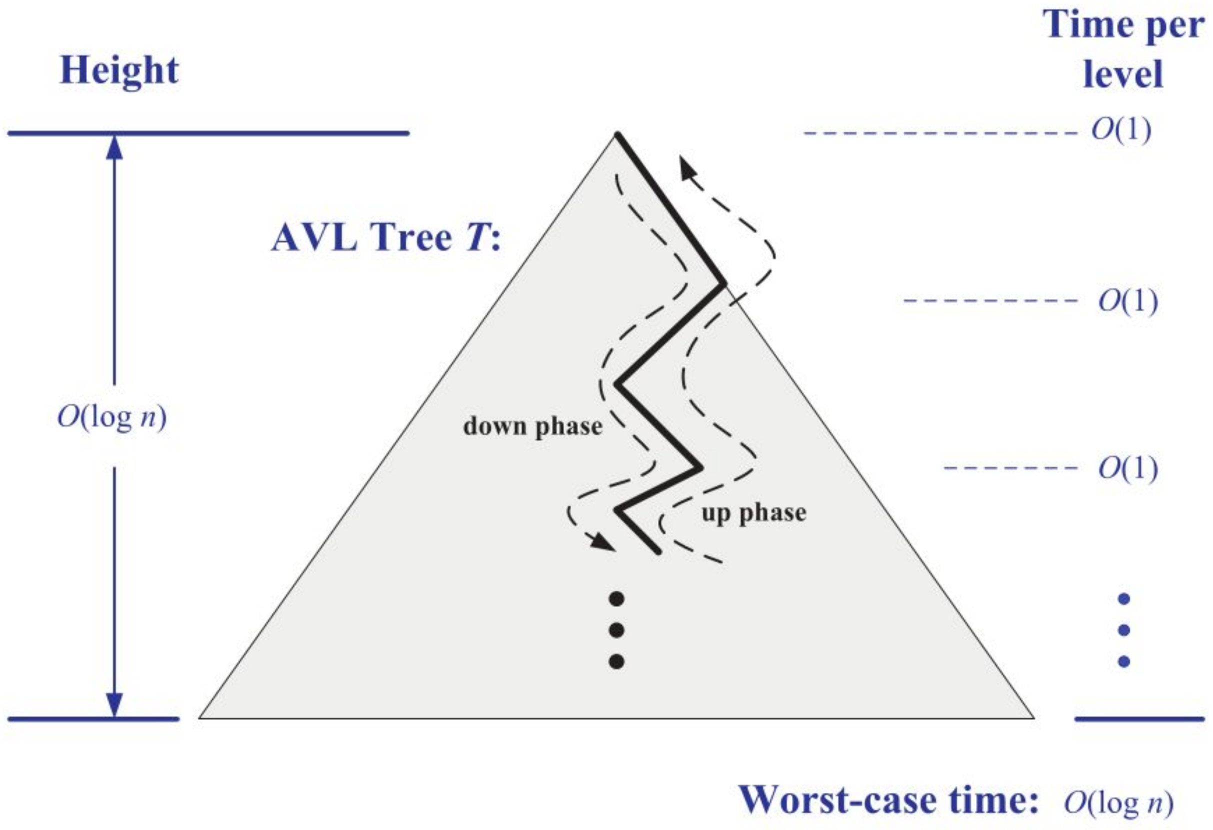
Eliminación de nodo con dato 32. Después de eliminar el nodo con dato 32, la raíz se queda desbalanceada

Se recupera balance según caso RR: rotación a la izquierda de *y* sobre *z*

Análisis de TDA árbol AVL

- Dado que por definición, la altura de los subárboles izquierdo y derecho de un nodo difieren en a lo más 1, ambos subárboles se mantienen lo más balanceados posible

- Garantía que la altura del árbol es un múltiplo pequeño de $\log_2 n$, por lo que el tiempo de acceso es $O(\log_2 n)$, para operaciones
  - **buscar()**
  - **insertar()**
  - **eliminar()**



Actividad de cierre

código: 1809 5000

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Próximas fechas...

- Resumen de la semana:
  - TDA árbol AVL

    ~~cátedra~~ – refuerzo – laboratorio

- Próxima semana:
  - TDA tabla HASH

U4 - S12