Instituto Politécnico Nacional

Escuela Superior de Cómputo

Estructuras de Datos

Tema 11: Árbol balanceado AVL

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom *edgardoadrianfrancom*

Contenido

- Problema de los árboles binarios de búsqueda
- Variantes de los árboles binarios de búsqueda
- Árbol balanceado AVL
  - Definición
  - Condición de equilibrio
  - Características
  - Operaciones sobre un AVL
    - Insertar nodos
    - Balancear
    - Eliminar Nodos
    - Calcular altura
- Complejidad de la búsqueda
- Ordenes de complejidad
- Búsquedas y recorridos
- Ejercicios de árboles AVL
  - *Inserción de claves*
  - *Eliminación de claves*
- Liga Web

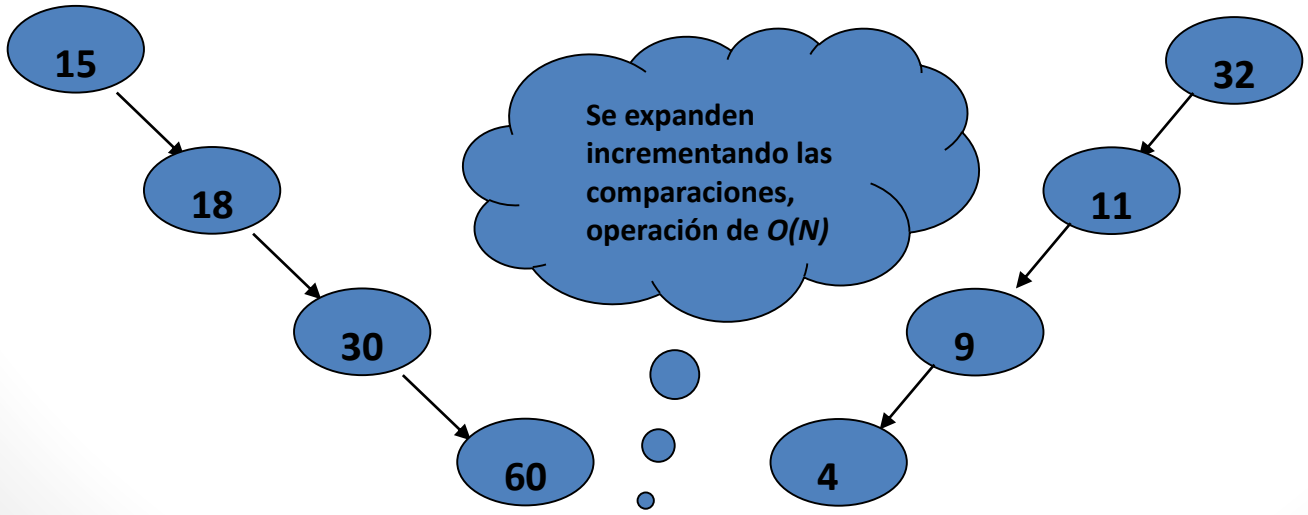
Problema de los árboles binarios de búsqueda

- Los árboles binarios, son eficientes en las operaciones de búsqueda, inserción y eliminación cuando el árbol crece o decrece descontroladamente, la eficiencia de la estructura de datos decae.

- Una situación critica es cuando se insertan elementos ordenados.

Insertar los datos 15,18,30,60

Insertar los datos 32,11,9,4,



Variantes de los árboles binarios de búsqueda

- Con el objetivo de mantener la eficiencia en la operación de búsqueda surgen modificaciones a las reglas de operación del árbol binario de búsqueda.
  - Árbol rojo-negro
  - **Árbol balanceado (AVL)**
  - Árbol biselado
- Estas variantes presentan ventajas en cuanto al rendimiento que ofrecen a la hora de realizar búsquedas principalmente.

Árbol balanceado AVL

- La principal característica de estos es la de realizar reacomodos o **balanceos**, después de inserciones o eliminaciones de elementos.

- Estos árboles también reciben el nombre de AVL *(autores: 2 matemáticos rusos G.M. **A**delson-**V**elskii y E.M **L**andis en 1962).*

- *Formalmente **se define un árbol balanceado como un árbol de búsqueda**, en el cual se debe cumplir la siguiente condición: **"Para todo nodo T del árbol la altura de los subárboles izquierdo y derecho no deben diferir en a lo sumo una unidad".***

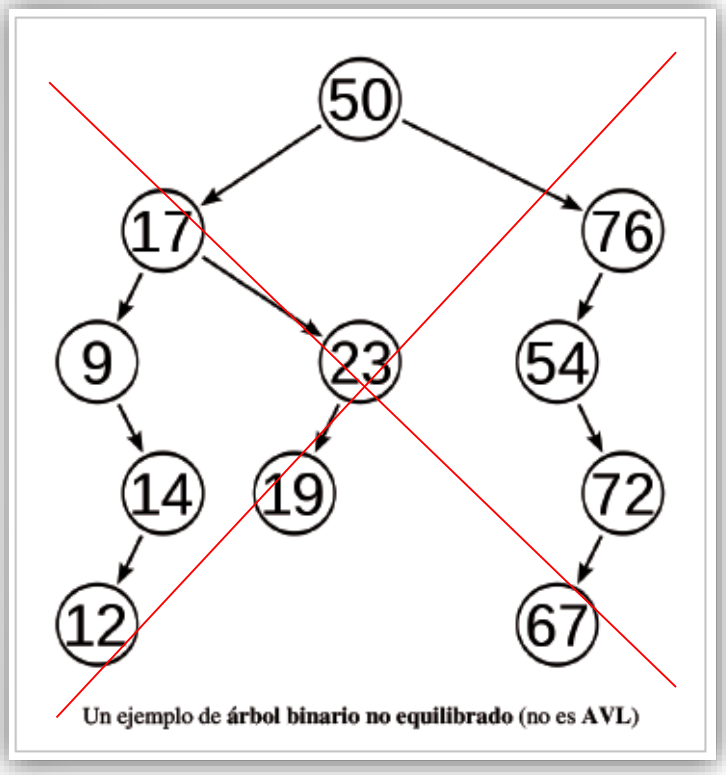
Definición

- Básicamente un árbol AVL es un ***Árbol Binario de Búsqueda*** al que se le añade una ***condición de equilibrio.***

“Para todo nodo la altura de sus subárboles izquierdo y derecho pueden diferir a lo sumo en 1”.

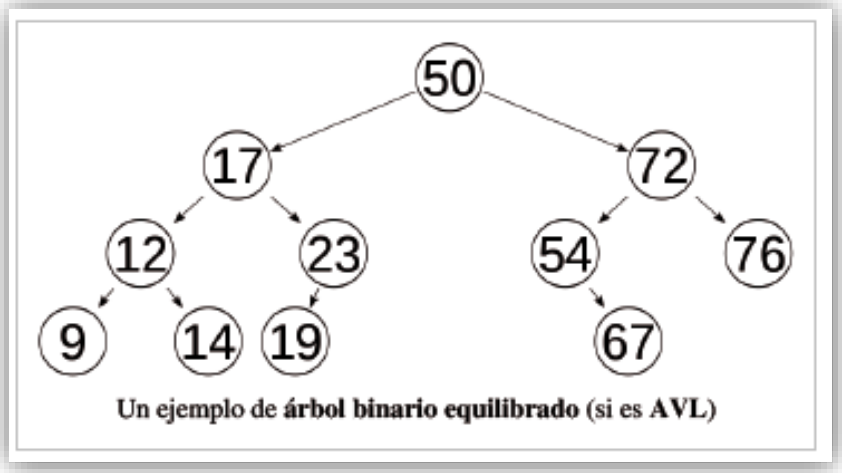
- Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad **O(log n).**

Condición de equilibrio



Un ejemplo de **árbol binario no equilibrado** (no es AVL)

"Para todos los nodos, la altura de la rama izquierda no difiere en mas de una unidad de la altura de la rama derecha"

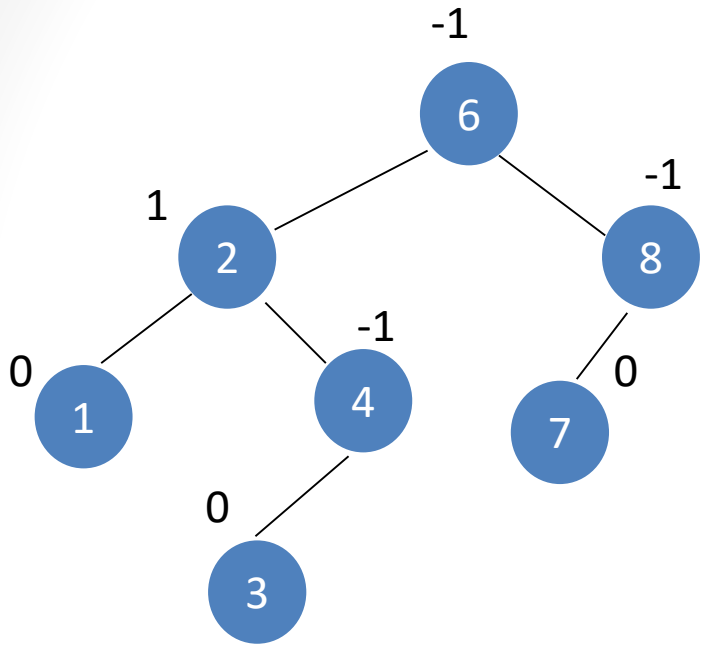


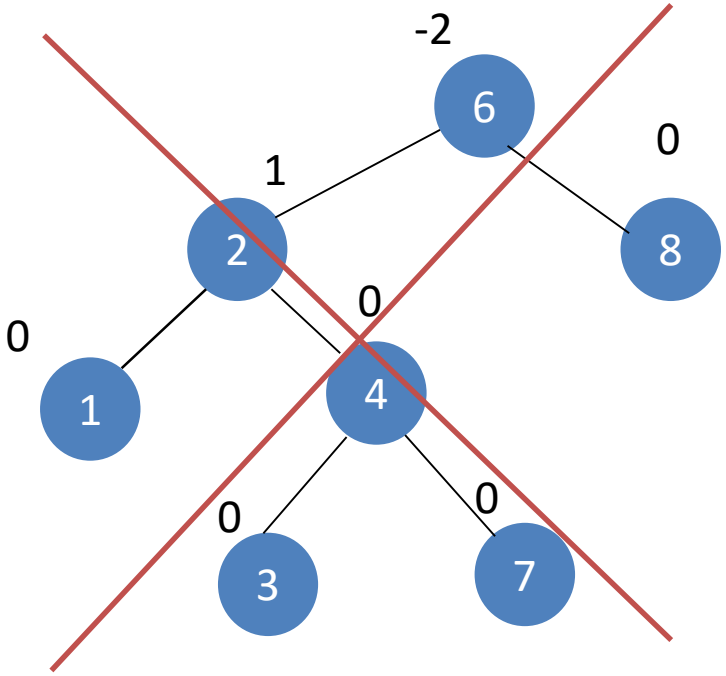
Un ejemplo de **árbol binario equilibrado** (si es AVL)

Características

- Un **AVL** es un ***ABB.***
- La ***diferencia entre las alturas*** de los subárboles. derecho e izquierdo ***no*** debe excederse en ***más de 1***.
- ***Cada nodo tiene asignado un peso*** de acuerdo a las alturas de sus subárboles.
- Un nodo tiene un peso de **1** si su ***subárbol derecho*** es más alto, **-1** si su ***subárbol izquierdo*** es más alto y **0** si las ***alturas son las mismas***.
- La inserción y eliminación en un árbol AVL es la misma que en un ABB.

Ejemplo de AVL





Sólo el árbol de la ***izquierda es AVL***. El de la derecha ***viola la condición de equilibrio*** en el ***nodo 6,*** ya que su subárbol izquierdo tiene altura 3 y su subárbol derecho tiene altura 1.

Equilibrio

- ***Equilibrio = (altura derecha)– (altura izquierda)***

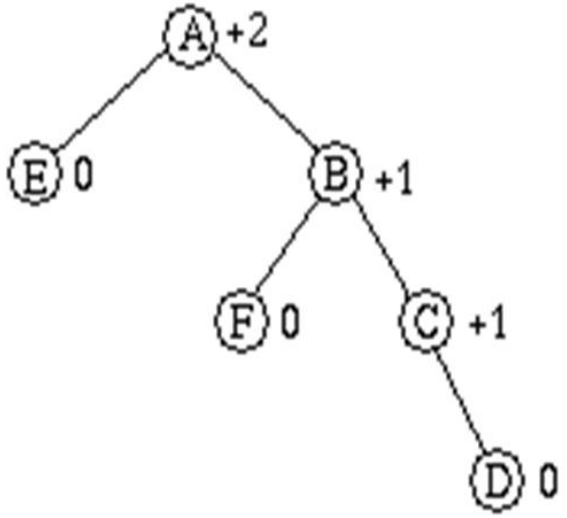
- Describe relatividad entre subárbol *derecho* y subárbol *izquierdo*.
  - **+ (positivo)** → derecha mas alto (profundo)
  - **- (negativo)** → izquierda mas alto (profundo)

"Un árbol binario es un AVL si y sólo si cada uno de sus nodos tiene un equilibrio de –1, 0, + 1"

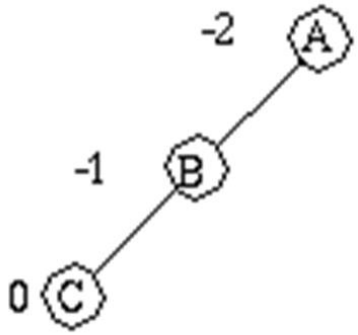
- Si alguno de los pesos de los nodos se modifica en un valor no válido **(2 ó -2) debe seguirse un esquema de rotación**.

Desequilibrios

- Desequilibrio hacia la izquierda ***(Equilibrio > +1)***



- Desequilibrio hacia la derecha ***(Equilibrio < -1)***



Operaciones sobre un AVL

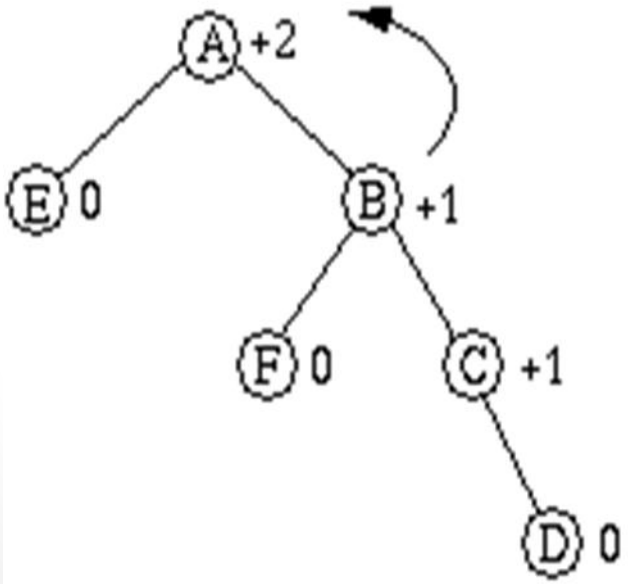
1. **Insertar nodo**
2. **Balancear**
   - **Caso 1** Rotación simple izquierda **RSI**
   - **Caso 2** Rotación simple derecha **RSD**
   - **Caso 3** Rotación doble izquierda **RDI**
   - **Caso 4** Rotación doble derecha **RDD**
3. **Eliminar nodo**
4. **Calcular altura**

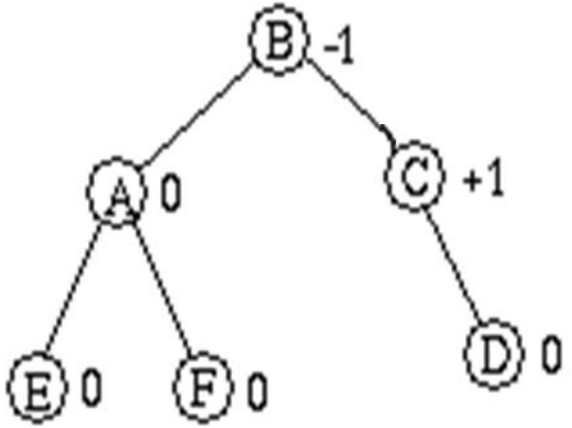
Insertar un nodo

1. Se usa la misma técnica que para insertar un nodo en un ABB ordenado
2. Trazamos una ruta desde el nodo raíz hasta un nodo hoja (donde hacemos la inserción).
3. Insertamos el nodo nuevo.
4. Volvemos a trazar la ruta de regreso al nodo raíz, ajustando el equilibrio a lo largo de ella.
5. Si el equilibrio de un nodo llega a ser + - 2, volvemos a ajustar los subárboles de los nodos para que su equilibrio se mantenga acorde con los lineamientos AVL (que son +- 1)

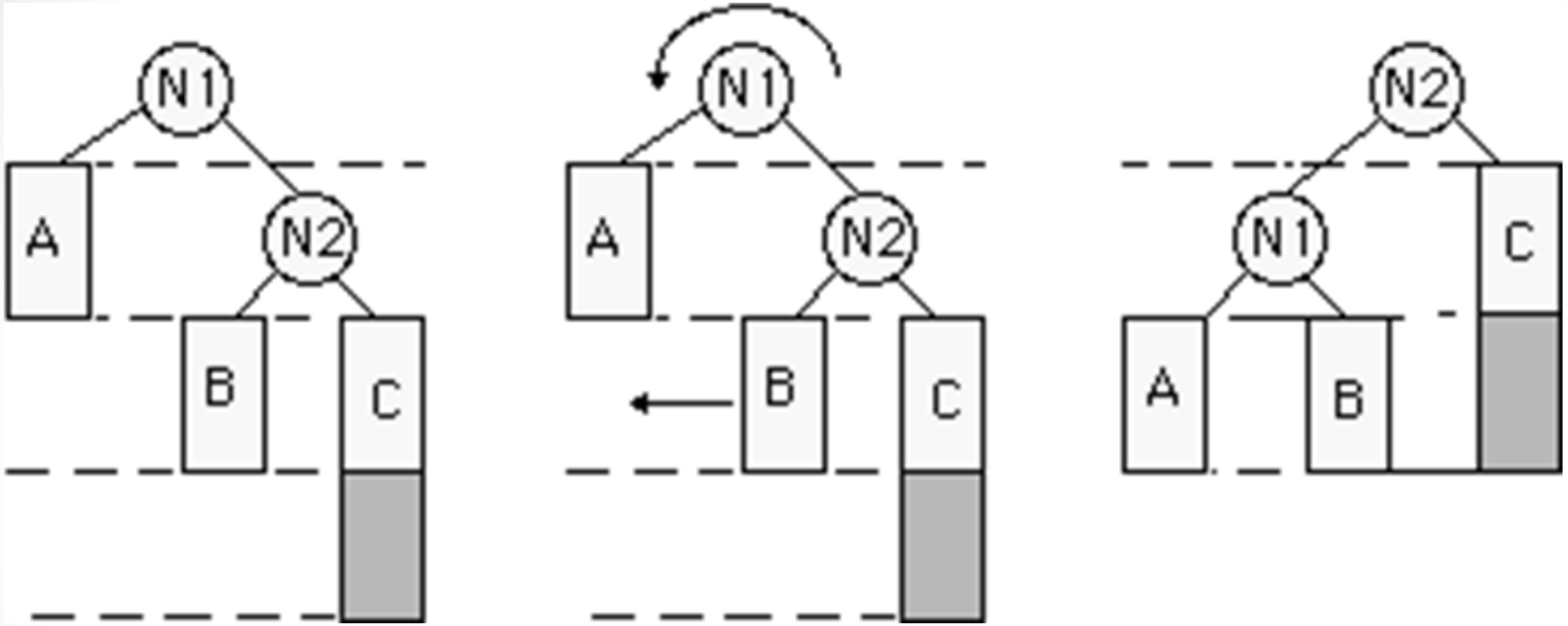
Balancear

- ***Caso 1: Rotación simple izquierda RSI***
  - Si esta desequilibrado a la izquierda (E>+1) y su hijo derecho tiene el mismo signo (+) hacemos rotación sencilla izquierda.

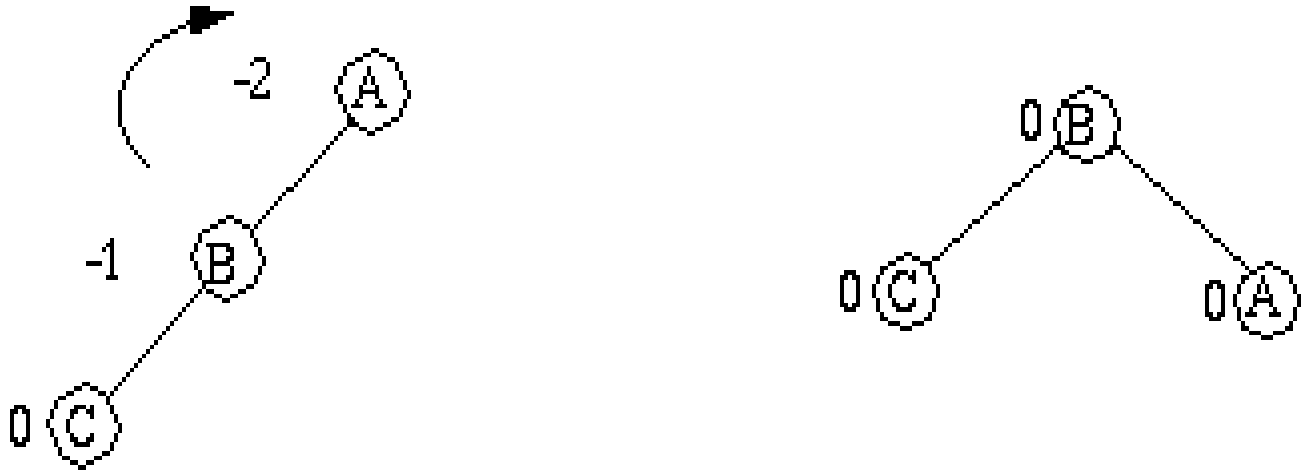




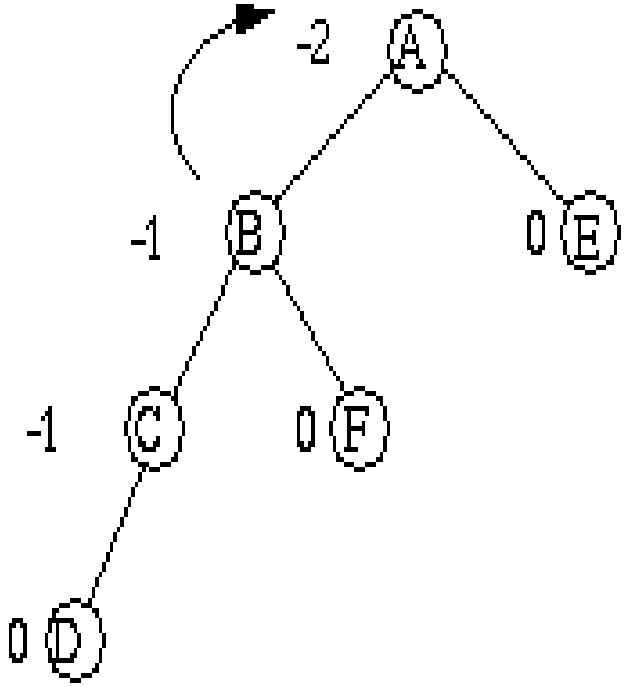
- ***Caso 1: Rotación simple izquierda RSI***

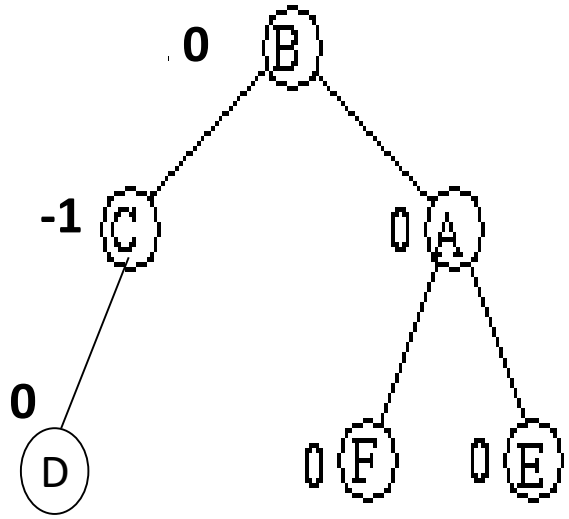


- ***Caso 2: Rotación simple derecha RSD***
  - Si esta desequilibrado a la derecha (E<-1) y su hijo izquierdo tiene el mismo signo (-) hacemos rotación sencilla derecha.

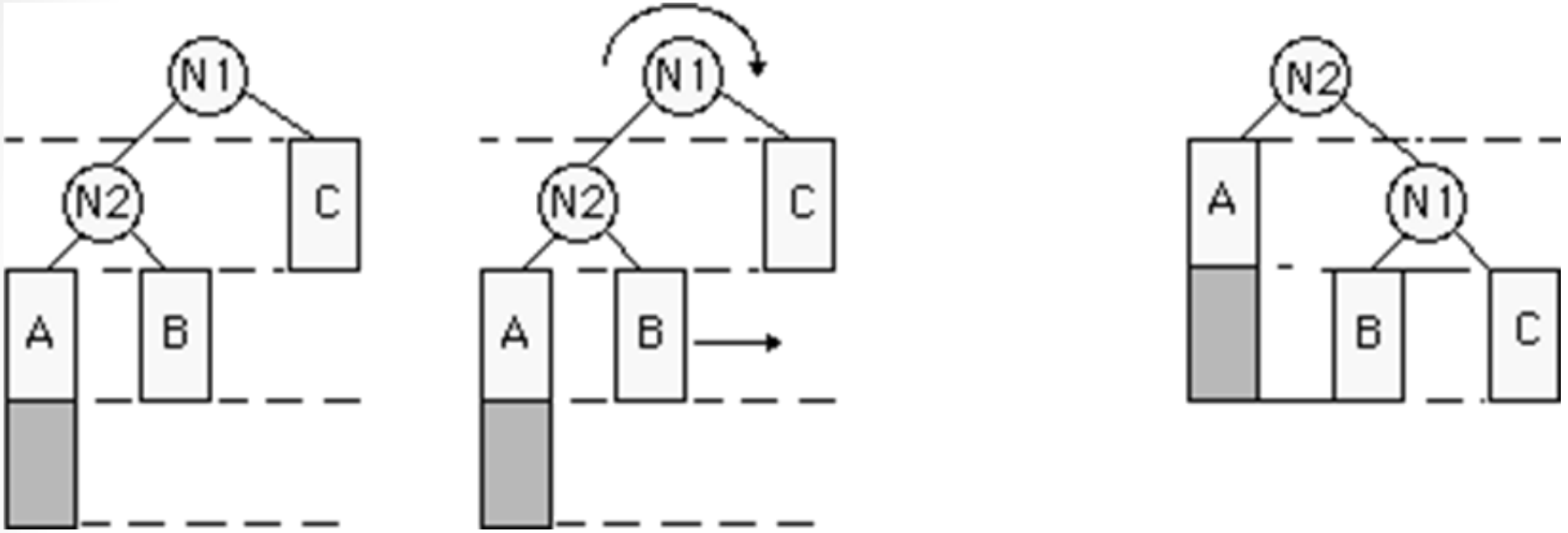


Caso 2: Rotación simple derecha RSD





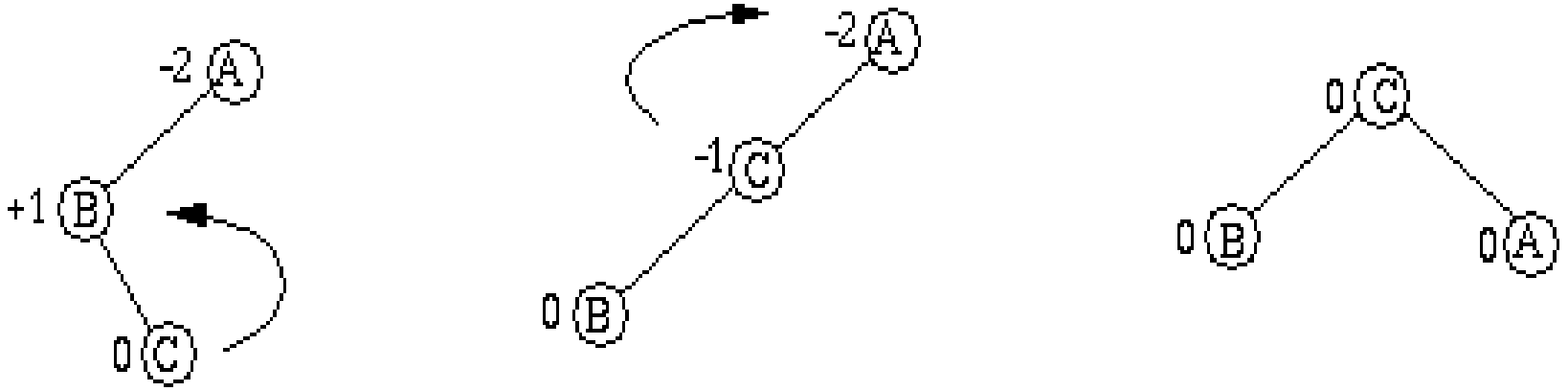
Caso 2: Rotación simple derecha RSD



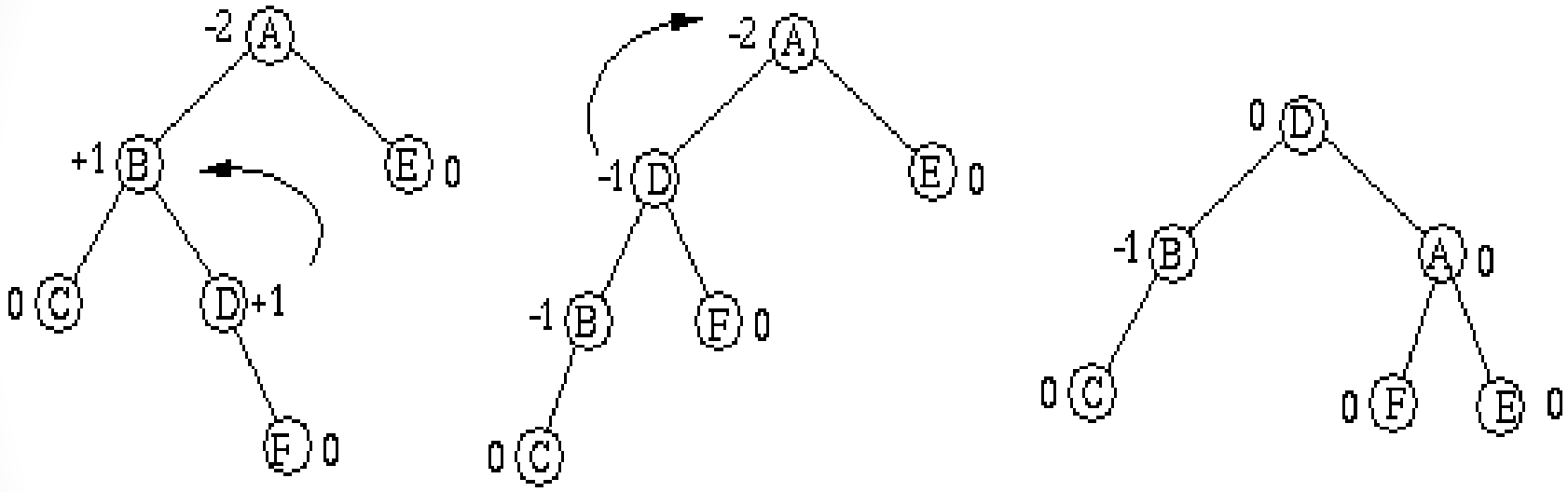
Rotación simple izquierda o derecha.

- **Observaciones:**
  - Se conserva el ***orden apropiado*** del árbol.
  - Restablece todos los nodo a equilibrios apropiados AVL
  - ***Conserva el recorrido en orden*** que el árbol anterior.
  - Sólo se necesita a lo más ***modificar 3 apuntadores*** para lograr el nuevo equilibrio (con la de la raíz)

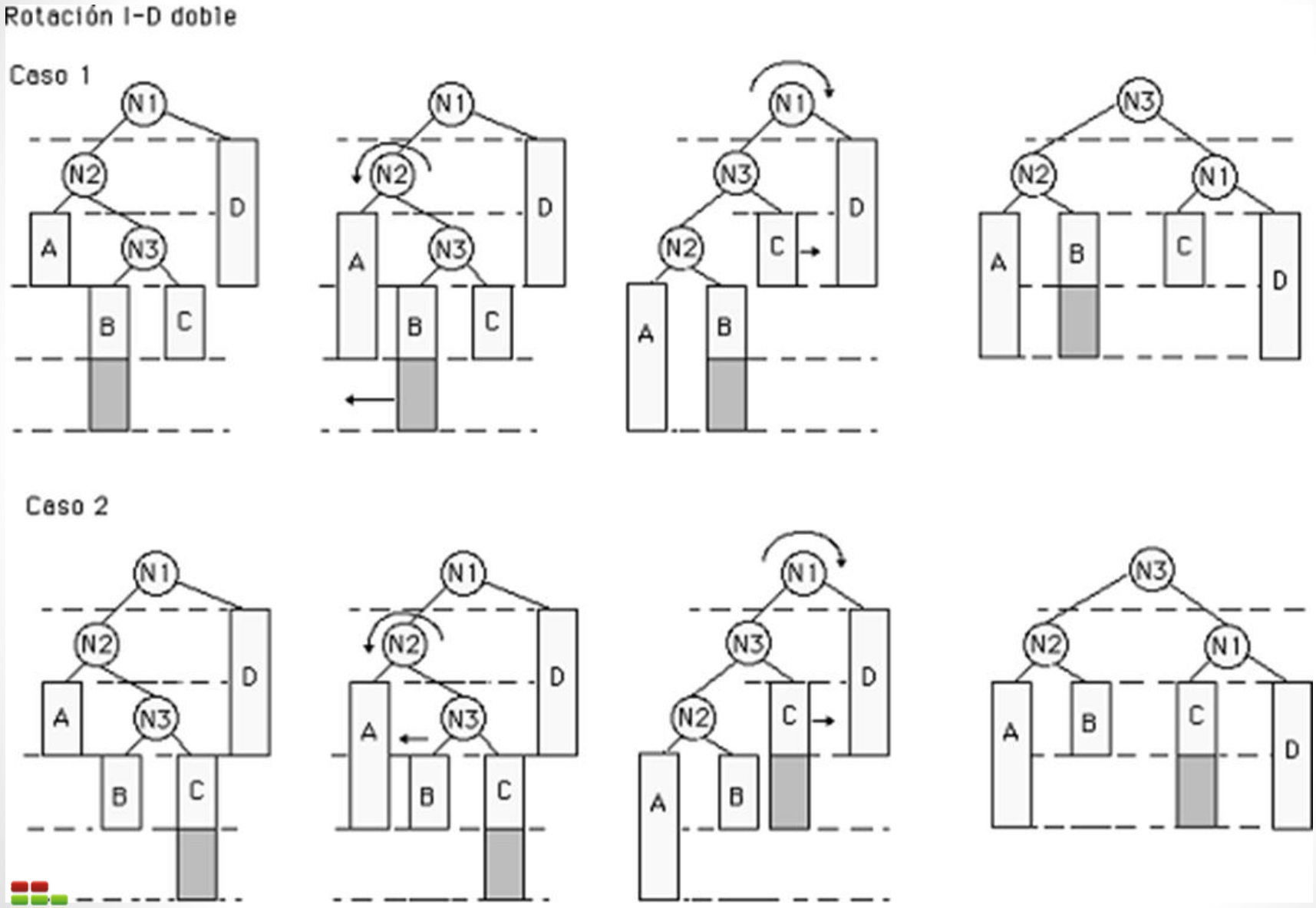
- ***Caso 3: Rotación doble izquierda RDI***
  - Si está desequilibrado a la derecha (E< –1), y su hijo izquierdo tiene distinto signo (+) hacemos rotación doble **izquierda-derecha**.



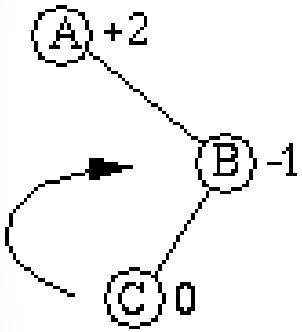
- ***Caso 3: Rotación doble izquierda RDI***

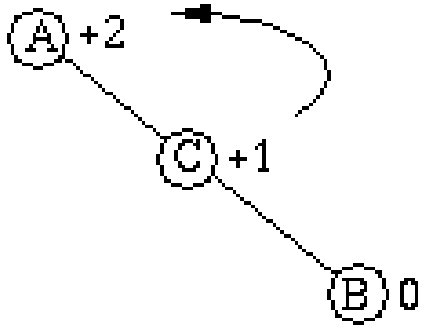


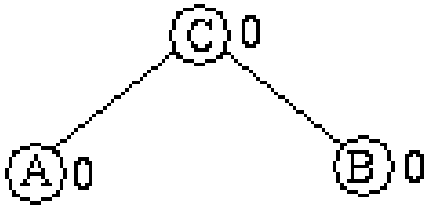
• *Caso 3: Rotación doble izquierda RDI*



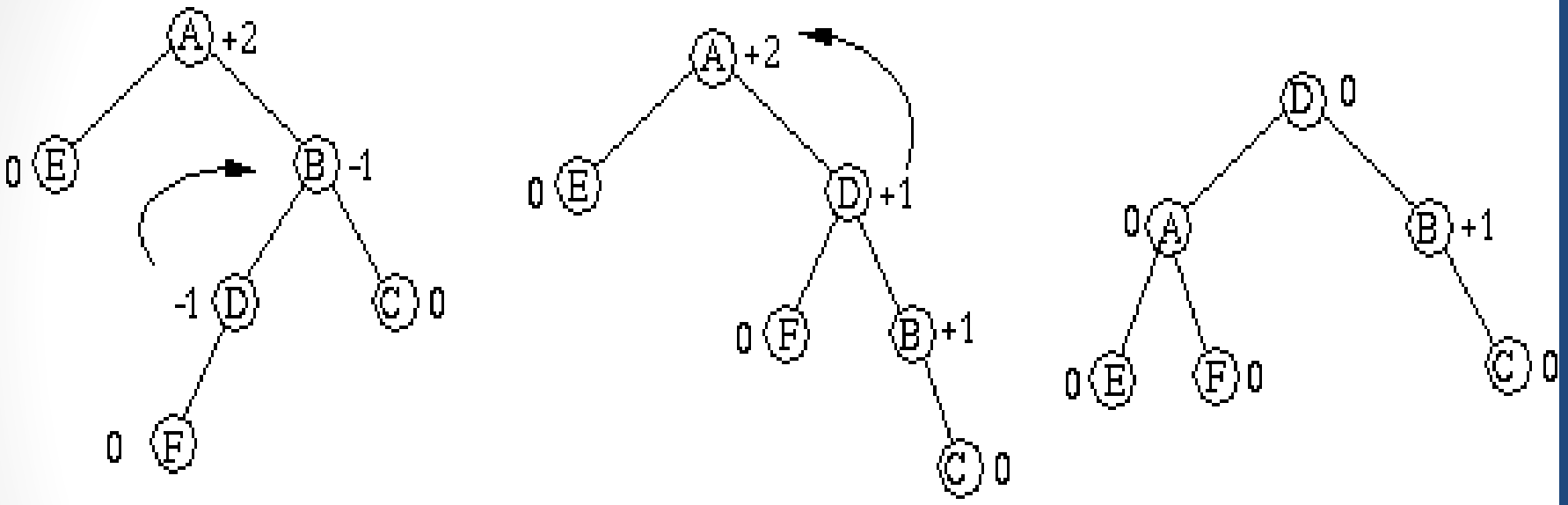
- ***Caso 4: Rotación doble derecha RDD***
  - Si esta desequilibrado a la izquierda (E>+1), y su hijo derecho tiene distinto signo (–) hacemos rotación doble **derecha-izquierda**.



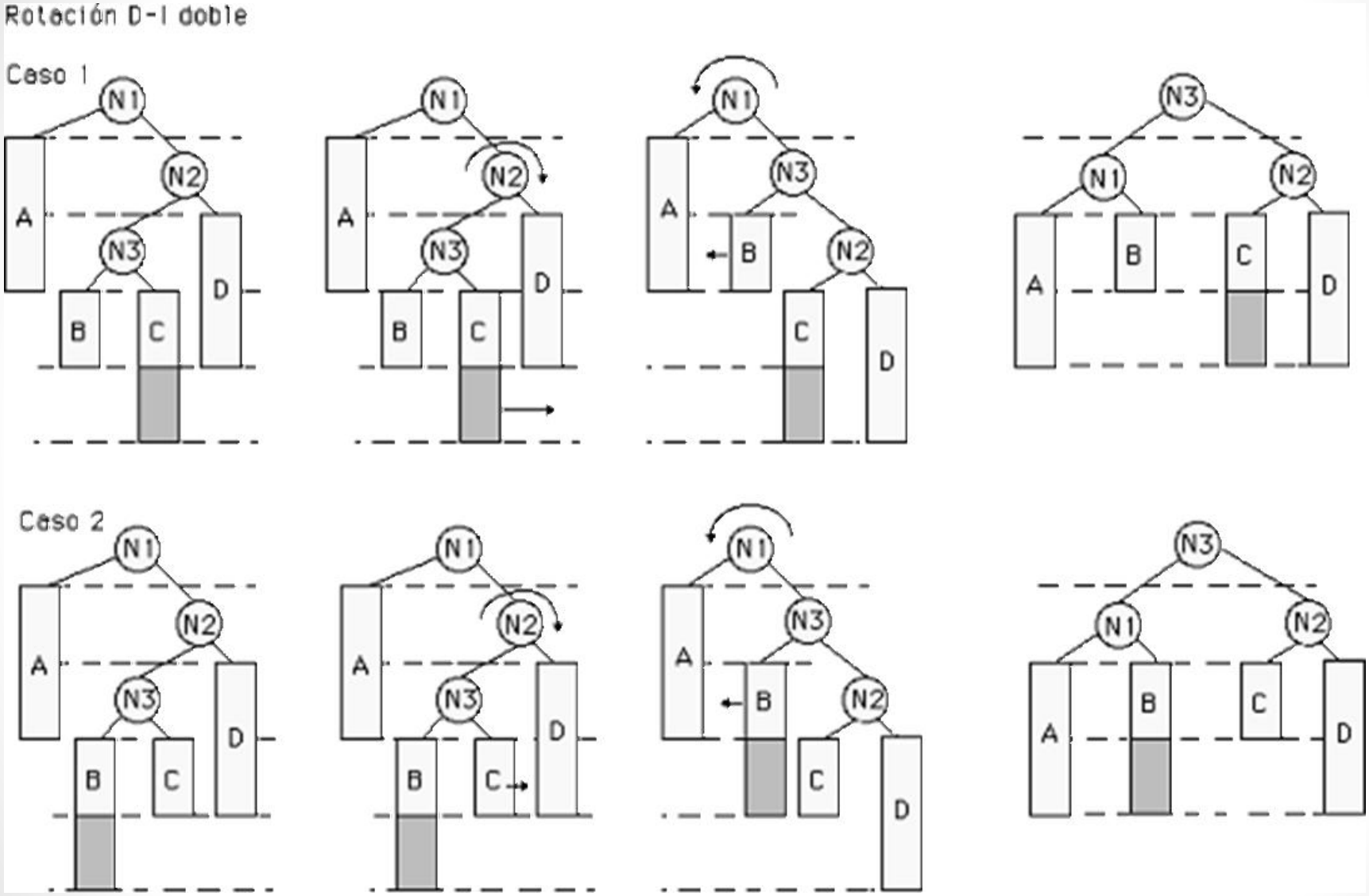




- ***Caso 4: Rotación doble derecha RDD***



- ***Caso 4: Rotación doble derecha RDD***



Eliminar

- Al eliminar un nodo en un árbol AVL puede afectar el equilibrio de sus nodos. Entonces hay que hacer rotaciones simples o dobles.

- Eliminar un nodo se realiza de la misma manera que en un árbol binario ordenado. Al localizar el nodo que se desea eliminar se realiza el siguiente procedimiento:
  - Si el nodo es un nodo hoja, simplemente lo eliminamos.
  - Si el nodo solo tiene un hijo, lo sustituimos con su hijo.
  - Si el nodo eliminado tiene dos hijos, lo sustituimos por el nodo que se encuentra mas a la derecha en el subárbol izquierdo o más a la izquierda en el subárbol derecho.

- Una vez que se ha eliminado el nodo, se tiene que equilibrar el árbol:
  - Si el equilibrio del padre del nodo eliminado cambia de *0* a *+-1* el algoritmo concluye.
  - Si el padre del nodo eliminado cambio de *+-1* a *0*, la altura del árbol ha cambiado y se afecta el equilibrio de su abuelo.
  - Si el equilibrio del padre del nodo eliminado cambia de *+-1* a *+- 2* hay que hacer una rotación. Después de concluirla, el equilibrio del padre podría cambiar, lo que, a su vez, podría forzarnos a hacer otros cambios (y probables rotaciones) en toda la ruta hacia arriba a medida que ascendemos hacia la raíz. Si encontramos en la ruta un nodo que cambie de *0* a *+- 1* entonces se concluye.

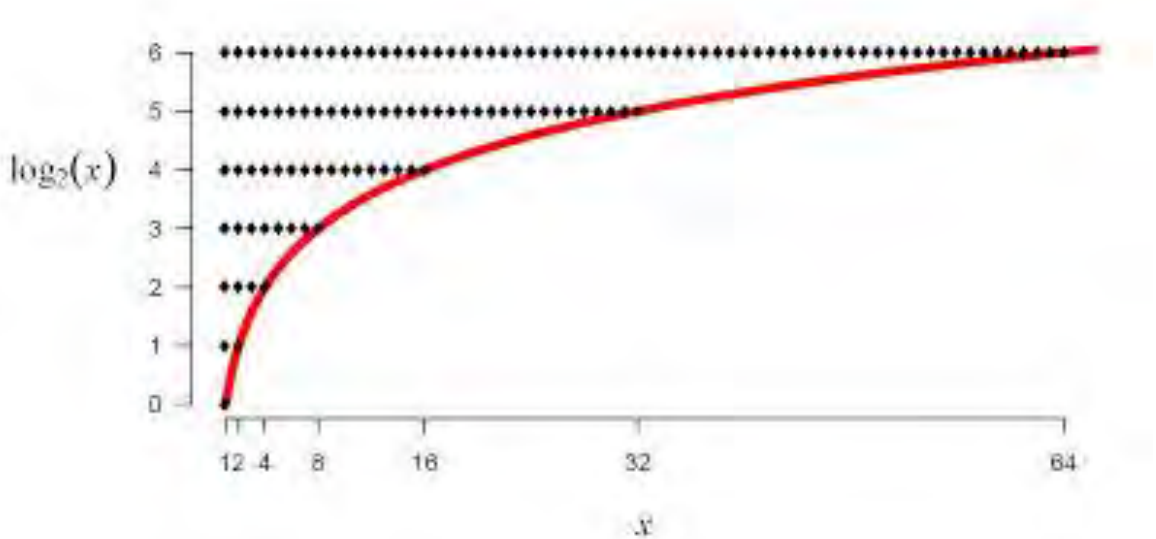
Calcular altura

- Una función para calcular la altura a partir de una posición puede escribirse recursivamente como:

```
//Calcula la altura de un árbol a partir de una posición p
int Altura(arbol *a, posicion p)
{
   if(NullNode(a,p))
      return 0;
   //Retornar la mayor altura (Izquierda o Derecha)
   else
      return 1 + Max(Altura(a,LeftSon(a,p)),Altura(a,RightSon(a,p)));
}
```

Complejidad de búsqueda

- Los ***árboles AVL están siempre equilibrados de tal*** modo que para todos los nodos, la altura de la rama izquierda no difiere en mas de una unidad de la altura de la rama derecha. Gracias a esta forma de equilibrio (o balanceo), la ***complejidad de una búsqueda*** en uno de estos arboles se mantiene siempre en ***orden de complejidad*** $O(\log_2 n)$.



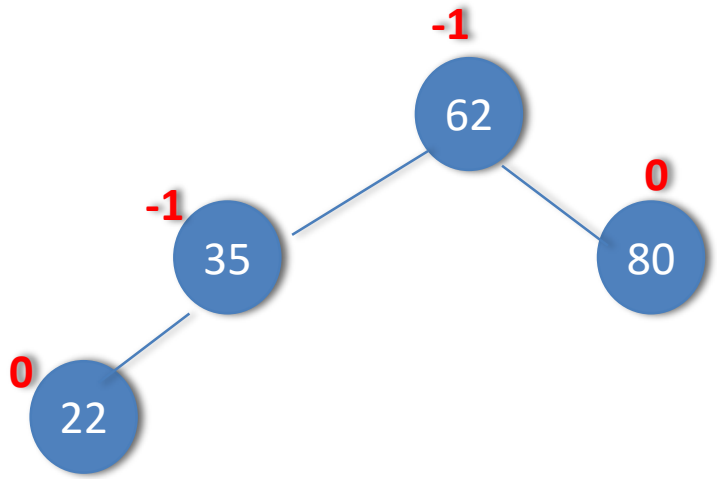
Ordenes de complejidad

- ***Ejemplo:*** 1 h de cómputo para un problema de tamaño N=100.

O(f(n))	*N=100*	*N=200*	*En un t=2h (N=?)*
log n	**1 h**	**1.15 h**	**10000**
n	1 h	2 h	200
n log n	1 h	2.30 h	199
n^2	1 h	4 h	141
n^3	1 h	8 h	126
2^n	1 h	10^{30} h	101

Ejercicios de inserción

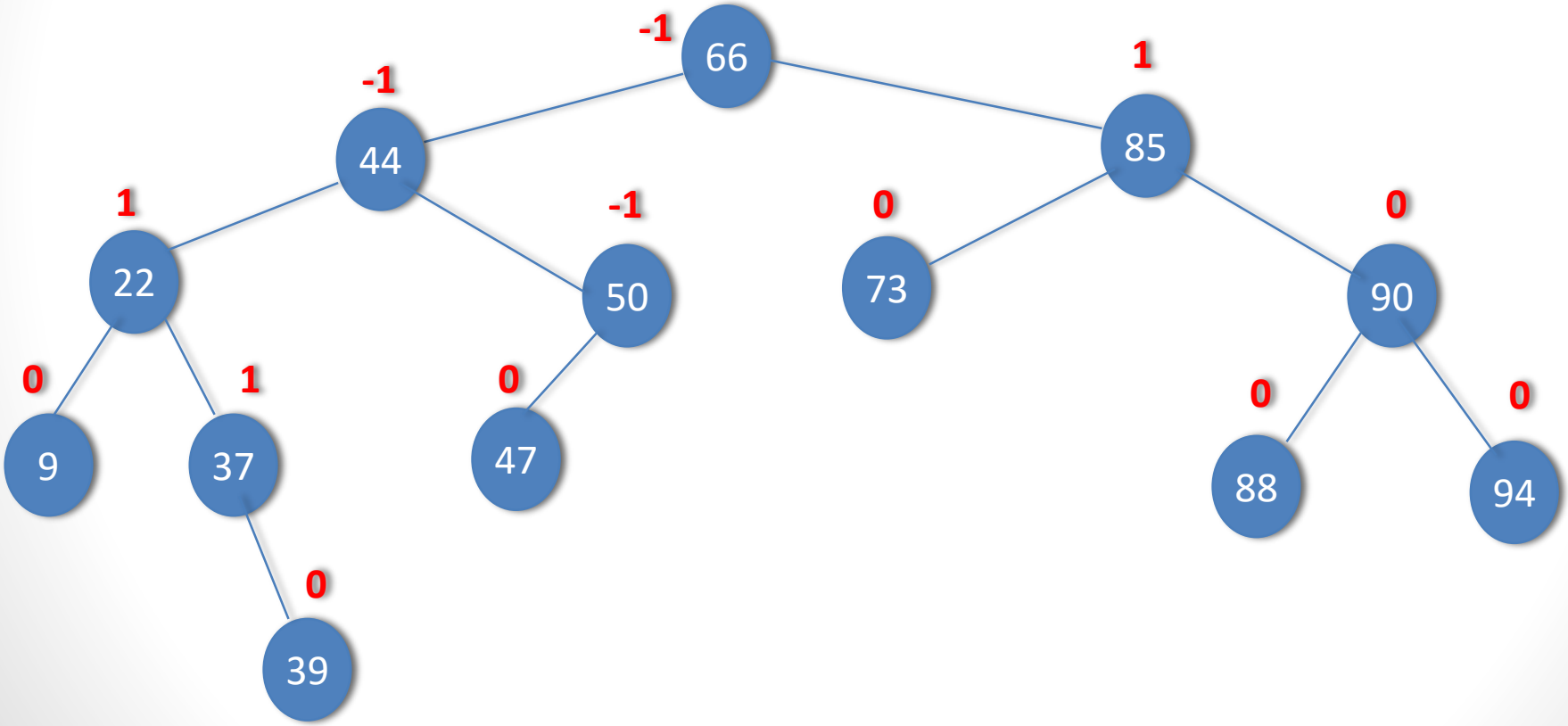
1. Inserte las claves 10 – 47 – 38 – 06 – 55 – 90 – 49 – 50 – 51 – 58 en el árbol balanceado que se da a continuación.



Ejercicios de eliminación

2. Elimine las siguientes claves del árbol balanceado siguiente:

73 – 66 – 50 – 47 – 39 – 94



Liga Web

- *Simulación del funcionamiento de un Árbol AVL*

http://www.cs.jhu.edu/~goodrich/dsa/trees/avltree.html