

ARBOLES BINARIOS DE BUSQUEDA**INSERCIÓN EN ABB**

```
void Inserta (arbol *A, Tdato X) {
    if (*A == null) {
        *A = (arbol) malloc (sizeof (struct nodo));
        (*A)->Dato = X;
        (*A)->Der = null;
        (*A)->Izq = null;
    }
    else
        if (X > (*A)->Dato)
            Inserta(&(*A)->Der, X);
        else
            Inserta(&(*A)->Izq, X);
}
```

ELIMINACIÓN EN ABB

```
void Borrar (arbol *p, arbol aux) {
    if ((*p)->Der != null)
        Borrar(&(*p)->Der, aux)
    else {
        aux->Dato = (*p)->Dato;
        aux = *p;
        *p = (*p)->Izq;
        free(aux);
    }
};

void Elimina (arbol *A, Tdato X) {
    arbol aux;
    if (*A != null)
        if (X < (*A)->Dato)
            Elimina (&(*A)->Izq, X);
        else
            if (X > (*A)->Dato)
                Elimina (&(*A)->Der, X);
            else {
                aux = *A;
                if (aux->Der == null) { //grado 0 grado 1
                    *A = aux->Izq;
                    free(aux);
                }
                else
                    if (aux->Izq == null) { //grado 1
                        *A = aux->Der;
                        free(aux);
                    }
                    else //grado 2
                        Borrar(&(aux->Izq), aux);
            }
};
```

ÁRBOLES BALANCEADOS POR SU ALTURA (AVL)

La altura H de un árbol binario T se define como:

- ☛ 0 si T contiene solo la raíz
- ☛ $1 + \max(H(T_{\text{izq}}), H(T_{\text{der}}))$ en otro caso

Un árbol AVL (**A**delson-**V**elskii y **L**andis) es un ABB en el que **para todo nodo** del árbol, la diferencia entre la altura de sus subárboles izquierdo y derecho **es a lo sumo 1**.

Factor de Equilibrio

Es la diferencia de altura entre los subárboles izquierdo y derecho:

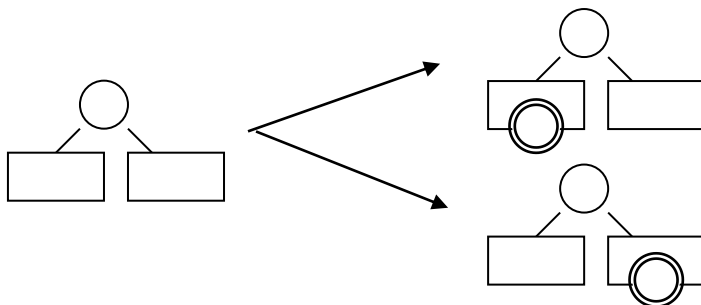
$$FE = \text{altura}(\text{IZQ}) - \text{altura}(\text{DER}) \rightarrow \text{si el árbol está balanceado} \rightarrow -1 \leq FE \leq 1 \quad \text{o} \quad |FE| \leq 1$$

Situaciones

Al insertarse o eliminarse un nodo en un árbol AVL, deben diferenciarse los siguientes casos:

- ✓ Los subárboles izquierdo y derecho tienen la misma altura, por lo que luego de la inserción / eliminación, el árbol se mantiene balanceado:

- Inserción



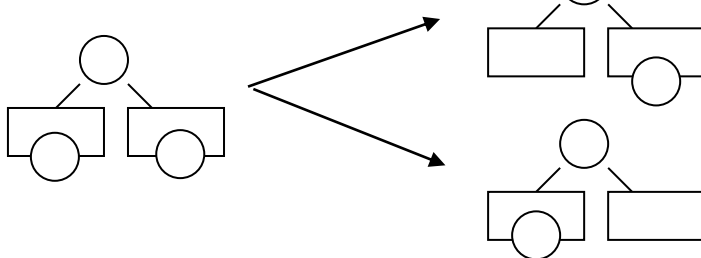
Aumenta la altura del subárbol izquierdo \rightarrow el árbol se mantiene balanceado

$$[\text{altura}(\text{IZQ}) = \text{altura}(\text{DER}) + 1]$$

Aumenta la altura del subárbol derecho \rightarrow el árbol se mantiene balanceado

$$[\text{altura}(\text{IZQ}) + 1 = \text{altura}(\text{DER})]$$

- Eliminación



Disminuye la altura del subárbol izquierdo \rightarrow el árbol se mantiene balanceado

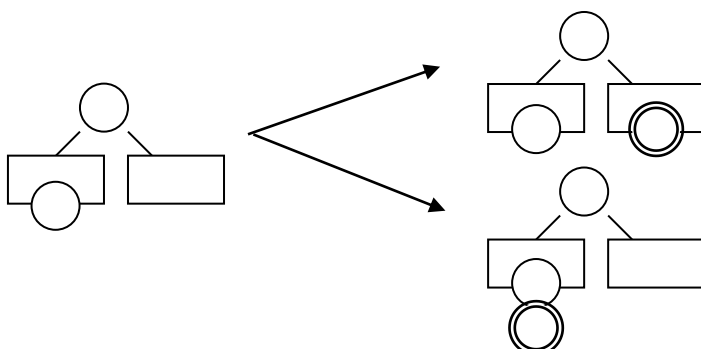
$$[\text{altura}(\text{IZQ}) + 1 = \text{altura}(\text{DER})]$$

Disminuye la altura del subárbol derecho \rightarrow el árbol se mantiene balanceado

$$[\text{altura}(\text{IZQ}) = \text{altura}(\text{DER}) + 1]$$

- ✓ La altura del subárbol izquierdo es mayor a la altura del subárbol derecho:

- Inserción:



Aumenta la altura del subárbol derecho \rightarrow el árbol se mantiene balanceado

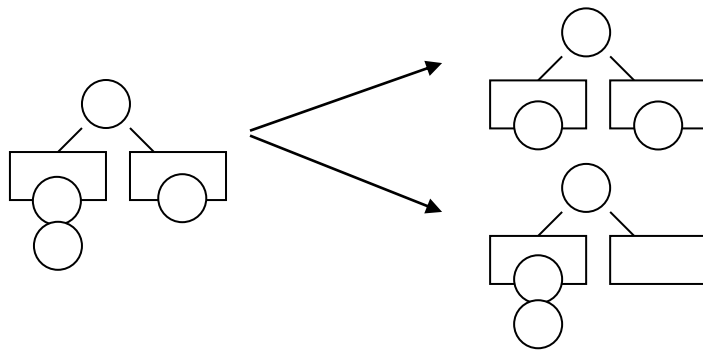
$$[\text{altura}(\text{IZQ}) = \text{altura}(\text{DER})]$$

Aumenta la altura del subárbol izquierdo \rightarrow el árbol se desbalancea

$$[\text{altura}(\text{IZQ}) = \text{altura}(\text{DER}) + 2]$$

Rebalancear el árbol

○ Eliminación:



Disminuye la altura del subárbol izquierdo → el árbol se mantiene balanceado

[altura (IZQ) = altura (DER)]

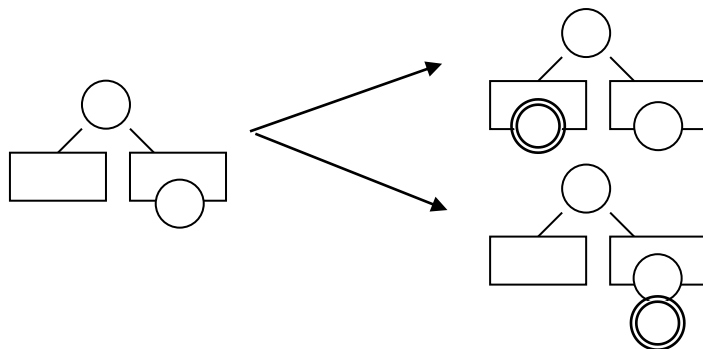
Disminuye la altura del subárbol derecho → el árbol se desbalancea

[altura (IZQ) = altura (DER)+2]

Rebalancear el árbol

✓ La altura del subárbol derecho es mayor a la altura del subárbol izquierdo:

○ Inserción:



Aumenta la altura del subárbol izquierdo → el árbol se mantiene balanceado

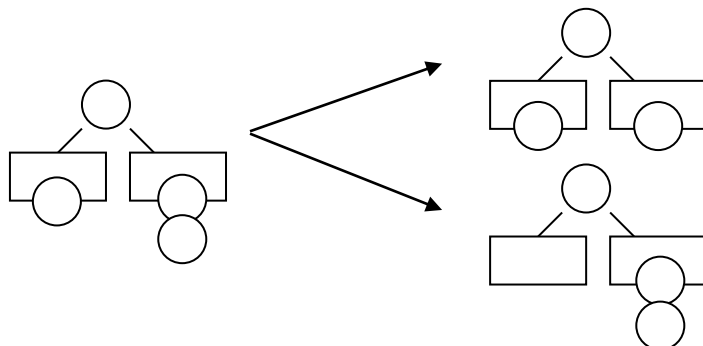
[altura (IZQ) = altura (DER)]

Aumenta la altura del subárbol derecho → el árbol se desbalancea

[altura (IZQ)+2 = altura (DER)]

Rebalancear el árbol

○ Eliminación:



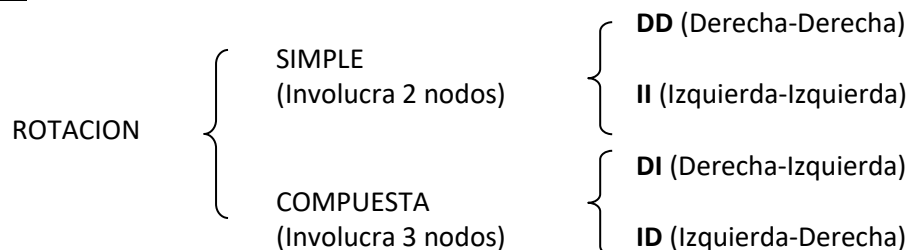
Disminuye la altura del subárbol derecho → el árbol se mantiene balanceado

[altura (IZQ) = altura (DER)]

Disminuye la altura del subárbol izquierdo → el árbol se desbalancea

[altura (IZQ)+2 = altura (DER)]

Rebalancear el árbol

Rotaciones

Una vez que se detecta un nodo para el cual el factor de equilibrio es 2 o -2, se debe avanzar (a partir de éste) dos niveles por la rama más larga. El camino que se sigue determina la rotación.

Algoritmos de Rotación

☛ Rotación DD

```
DERECHA(Nodo) ← IZQUIERDA(Nodo1)
IZQUIERDA(Nodo1) ← Nodo
Nodo ← Nodo1
```

☛ Rotación II

```
IZQUIERDA(Nodo) ← DERECHA(Nodo1)
DERECHA(Nodo1) ← Nodo
Nodo ← Nodo1
```

☛ Rotación DI

```
IZQUIERDA(Nodo1) ← DERECHA(Nodo2)
DERECHA(Nodo2) ← Nodo1
DERECHA(Nodo) ← IZQUIERDA(Nodo2)
IZQUIERDA(Nodo2) ← Nodo
Nodo ← Nodo2
```

☛ Rotación ID

```
DERECHA(Nodo1) ← IZQUIERDA(Nodo2)
IZQUIERDA(Nodo2) ← Nodo1
IZQUIERDA(Nodo) ← DERECHA(Nodo2)
DERECHA(Nodo2) ← Nodo
Nodo ← Nodo2
```

Metodología de Inserción en árboles AVL

- 1- Seguir el camino de búsqueda del árbol hasta localizar la ubicación en la que debe insertarse el nuevo elemento.
- 2- Insertar el nuevo nodo y actualizar su factor de equilibrio (será 0 por ser hoja)
- 3- Regresar por el camino de búsqueda calculando los FE de los distintos nodos que se visiten a lo largo del mismo. Si alguno de los FE es 2 o -2, debe rebalancearse aplicando la rotación que corresponda.
- 4- El proceso termina cuando se vuelve al nodo raíz y todos los FE están entre -1 y 1 y no es necesario realizar ninguna rotación; o cuando se haya efectuado alguna rotación (en este caso, no es necesario efectuar el cálculo de los FE del resto de los nodos)

Metodología de Eliminación en árboles AVL

- 1- Seguir el camino de búsqueda del árbol hasta localizar la posición del nodo a eliminar.
- 2- Eliminar el nodo (según su grado)
- 3- Regresar por el camino de búsqueda calculando los FE de los distintos nodos que se visiten a lo largo del camino. Si alguno de los FE es 2 o -2, debe rebalancearse aplicando la rotación que corresponda.
- 4- El proceso termina únicamente cuando se vuelve al nodo raíz y todos los FE están entre -1 y 1.

A diferencia de la inserción, la eliminación puede provocar más de una rotación

OPERADORES DEL TDA ÁRBOL N – ARI

Sean **A** variable de tipo árbol general y **p** es variable de tipo posición

Vacio(A) Devuelve verdadero si A es árbol Vacío.

Nulo(p) Devuelve verdadero si p es la posición Nula

HijoMasIzq(p,A) Devuelve la posición del hijo más a la izquierda de p, si p es hoja devuelve una posición nula.

HermanoDer(p,A) Devuelve la posición del hermano a la derecha de p (tiene el mismo padre de p), si p es el de la extrema derecha devuelve una posición nula.

Info(p,A) Devuelve el dato del en la posición p en el árbol A.

Raiz(A) Devuelve una posición que es la raíz del árbol A.

Padre(p,A) Devuelve la posición del padre de la posición p en el árbol A, si p es la raíz devuelve una posición nula.