Arboles

(Definición)

Un árbol «A» es un conjunto finito de uno o más nodos, tales que:

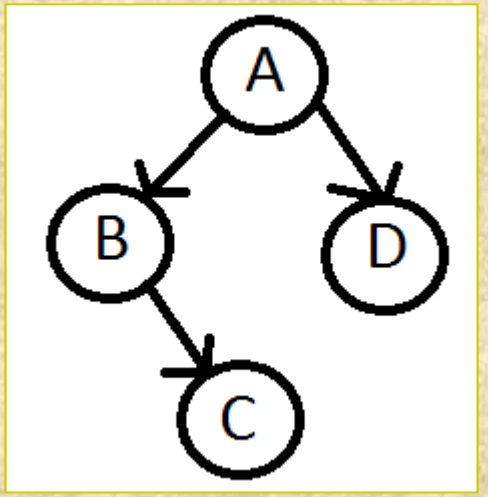
1. Existe un nodo especial denominado RAIZ(v1) del árbol.
2. Los nodos restantes (v2, v3, ..., vn) se dividen en *m* >= 0 conjuntos disjuntos denominado A1, A2,…Am, cada uno de los cuales es, a su vez, un árbol. Estos árboles se llaman *subárboles* del RAIZ. (Aguilar-Joyanes)

La definición de árbol implica una estructura recursiva. Esto es, la definición del árbol se refiere a otros árboles. Un árbol con ningún nodo es un «*árbol nulo*»; no tiene raíz.

Arboles

(Ejemplos-Conceptos)

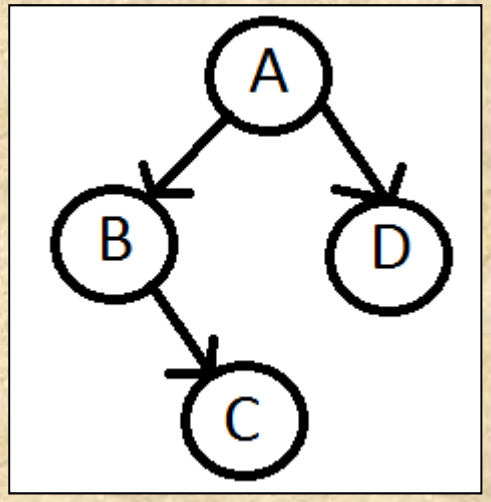
- En la siguiente figura tenemos 2 conceptos:
  - Nodo Raíz (Nodo A)
  - Nodos Hojas (Nodos C y D)
  - Nodos Internos o Interiores (Nodo B)
  - Sub-Arboles (B, C, D)

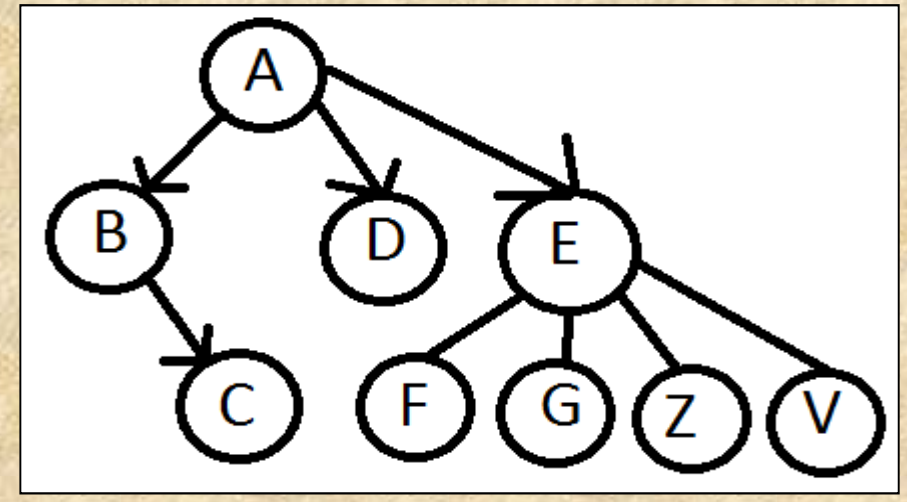


Arboles

(Tipos)

- Hay 2 tipos de arboles bien identificados:
  - Arboles Binarios
    - Tienen como máximo 2 hijos (Hijo Izquierdo e Hijo Derecho)
  - Arboles n-arios
    - Tienen «N» hijos donde N >= 0

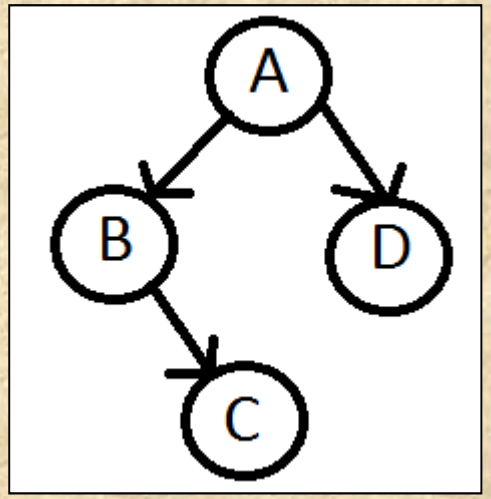


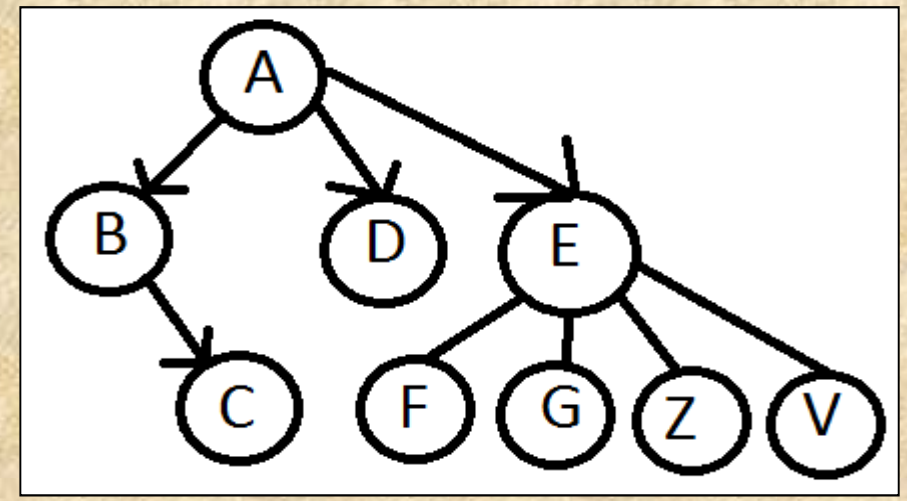


Arboles

(Tipos-Conceptos)

- Grado del Arbol
  - El grado del árbol se define en función a la cantidad de hijos de los nodos.
- Arboles n-arios
  - EL grado lo define el nodo con mayor cantidad de hijos. En este caso el «nodo E».

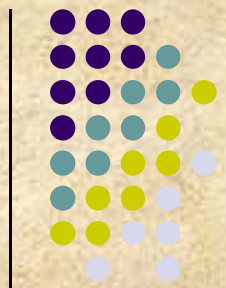




Arboles
(Otros Conceptos)

- Hermanos
  - Son los nodos que dependen del mismo padre. Para el caso de «B» es «D» en el árbol binario y «D,E» en el árbol n-ario.
- Padre
  - Todos los nodos tienen un padre a excepción de la raíz.
- Camino
  - Enlace entre dos nodos consecutivos.
- Rama
  - Es un camino que termina en una hoja.
- Nivel
  - Se determina por la longitud de la rama desde la Raíz a un nodo especifico.
- Altura
  - Se determina por el numero máximo de nodos de una rama o «rama mas larga desde la raíz a las hojas».
- Peso
  - Se determina por la cantidad de nodos terminales u hojas.

Arboles Binarios (definición)



- Arbol Binario

  Un *árbol binario* es un conjunto finito de cero o más nodos, tales que:
  • Existe un nodo denominado raíz del árbol.
  • Cada nodo puede tener 0, 1 o 2 subárboles, conocidos como *subárbol izquierdo* y *subárbol derecho*.

- Arbol Similar

  Se dice que 2 arboles binarios son similares si tienen la misma estructura pero no necesariamente su contenido.

- Arbol Equivalente

  Se dice que 2 arboles binarios son equivalentes si tienen la misma estructura y el mismo contenido.

- Arbol Equilibrado

  Se dice que un árbol esta equilibrado si de cada nodo del árbol la diferencia de las alturas del subarbol izquierdo versus el derecho a lo sumo difiere en una unidad.

  $$AE = |H(SI) - H(SD)| <= 1$$

Arboles Binarios (definición)

- Arbol Binario Completo

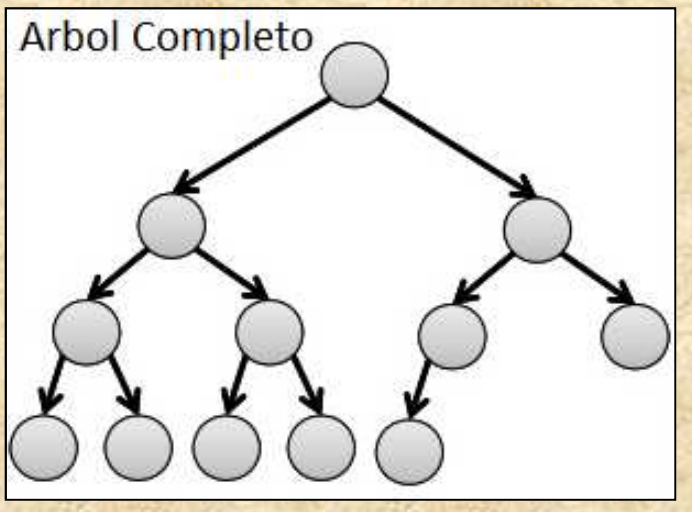
  Un árbol binario se llama *completo* si todos sus nodos tienen exactamente dos subárboles, excepto los nodos de los niveles más bajos que tienen cero.

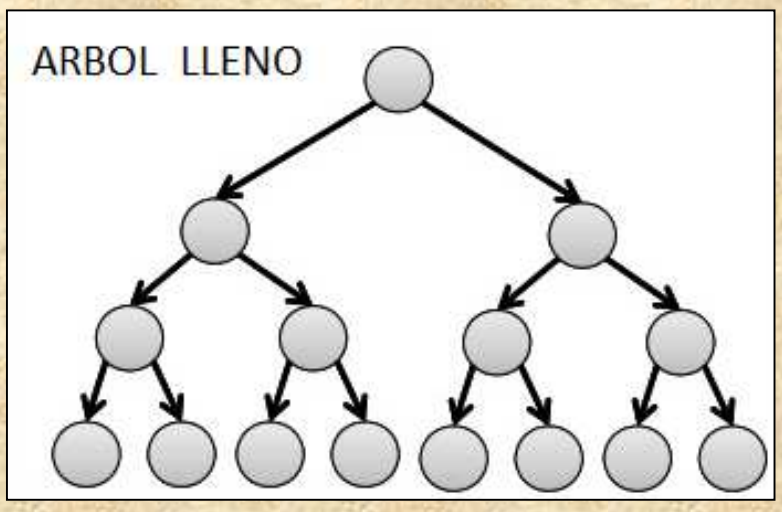
- Arbol Binario Lleno

  Un árbol binario completo, tal que todos los niveles están llenos, se llama *árbol binario lleno*.

- Arbol Binario Degenerado

  Todos sus nodos tienen solamente un subárbol, excepto el último.





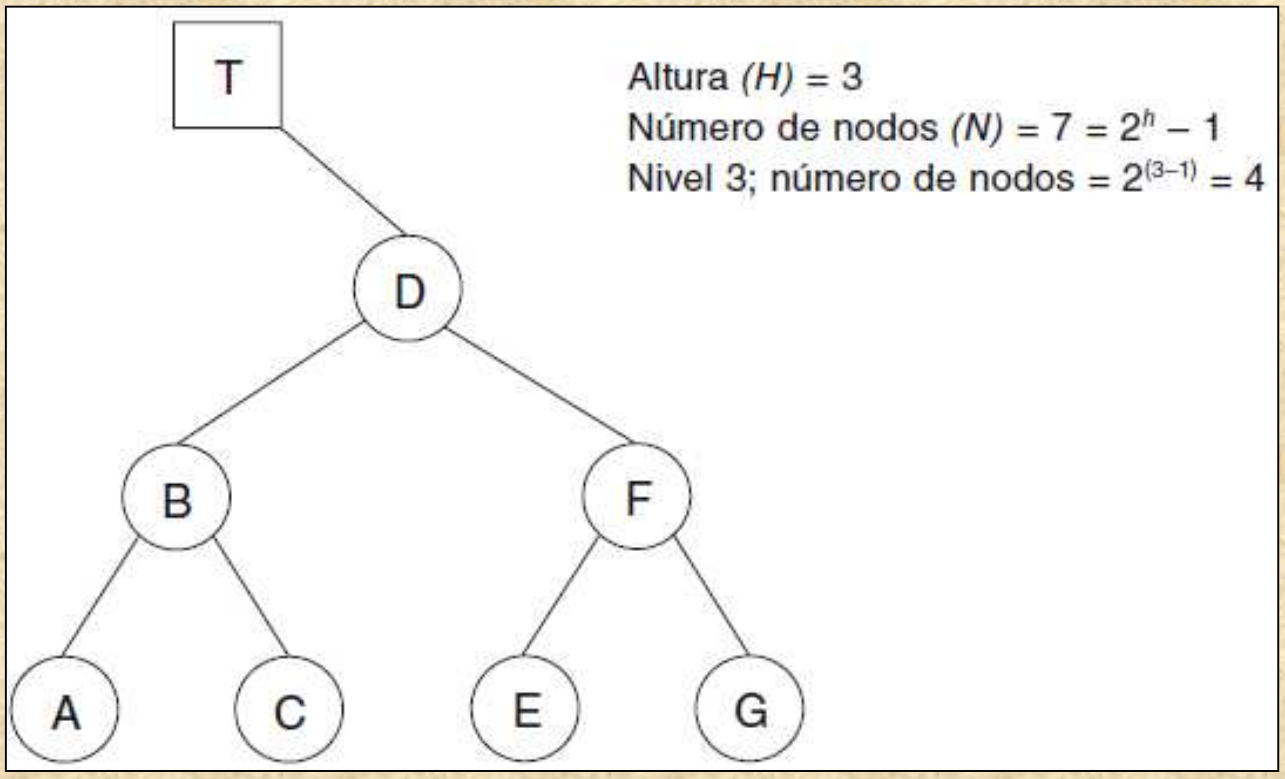
Arboles Binarios (def. matemáticas)

- Cantidad Máxima de Nodos

  Un árbol binario T de nivel «*H*» puede tener como máximo $2^h - 1$.

- Altura de Arbol Binario Lleno

  La altura de un árbol binario lleno de «N» nodos es $\log2(n + 1)$.



Arboles N-Arios (Conversión En Binario)

- Método de conversión (Transformada de Knuth)

  Supongamos que se tiene el árbol A y se quiere convertir en un árbol binario B. El algoritmo de conversión tiene tres pasos fáciles:

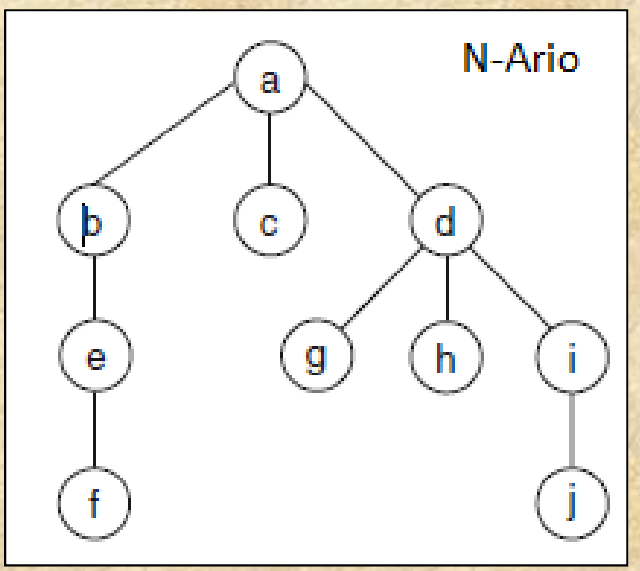
  1. La raíz de B es la raíz de A.
  2. *a*) Enlazar al nodo raíz con el camino que conecta el nodo más a la izquierda (su hijo).
     b) Enlazar este nodo con los restantes descendientes del nodo raíz en un camino, con lo que se forma el nivel 1.
     c) A continuación, repetir los pasos *a*) y *b*) con los nodos del nivel 2, enlazando siempre en un mismo camino todos los hermanos —descendientes del mismo nodo—. Repetir estos pasos hasta llegar al nivel más alto.
  3. Girar el diagrama resultante 45° para diferenciar e ntre los subárboles izquierdo y derecho.

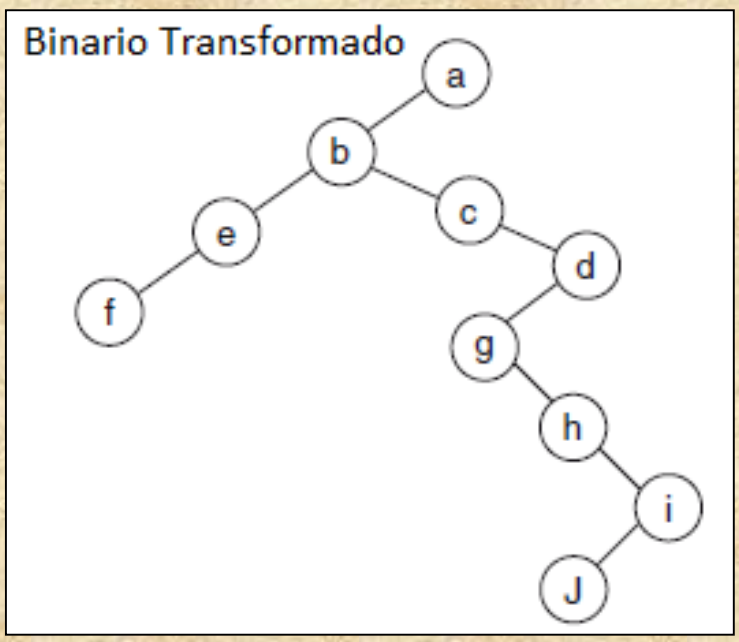
Arboles N-Arios (Ejemplos de Conversión)

- Método de conversión (Transformada de Knuth)

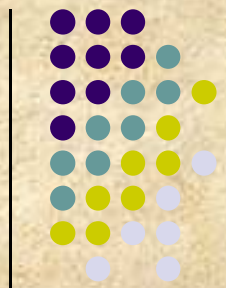
  Al hijo que desciende de nivel se le llama «Hijo Extremo Izquierdo»

  Los hijos derechos del árbol binario transformado se le llaman «Hermanos Derechos».





Arboles Binarios (Modos de Implementarlos)



- Existen 2 modos diferentes o alternativas:
  - Implementar el árbol mediante apuntadores
  - Implementar el árbol mediante cursores

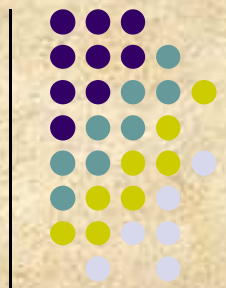
- Definición de la estructura

  Se define el nodo «NodoArbol» que debe contener como mínimo:

  1. Datos (puede ser un dato simple o un objeto de datos)
  2. Puntero al Hijo Izquierdo
  3. Puntero al Hijo Derecho
  4. Factor de Equilibrio (opcional usado para el balanceo)

  Dependiente del tipo de implementación el árbol puede ser simplemente el punto a raíz o bien la definición del cursor.

Arboles Binarios (Recorridos)

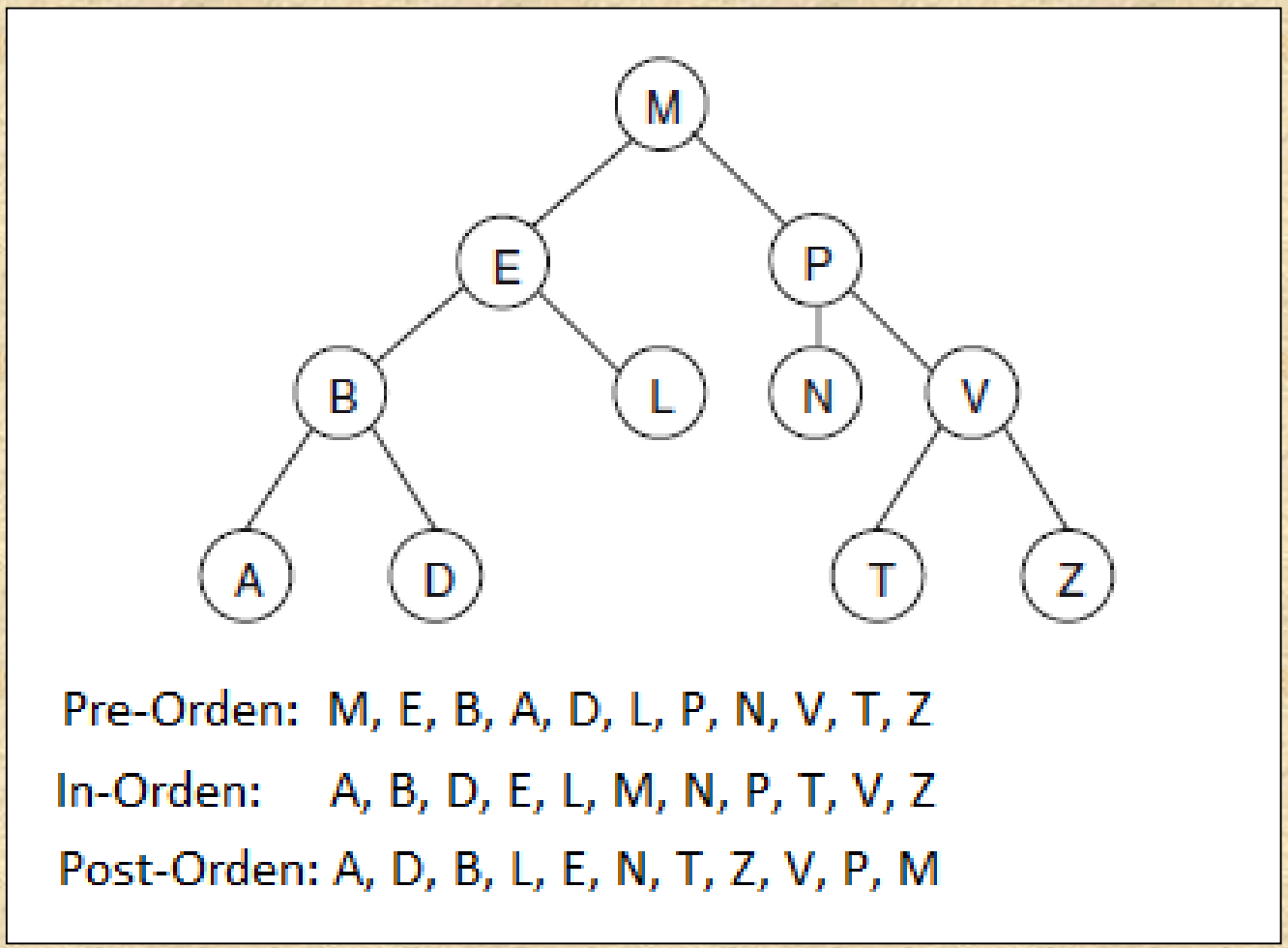


- Existen 2 modos de recorrer el árbol:
  - Recorrido en Profundidad
  - Recorrido en Anchura

- Recorrido en profundidad
  - Existen 3 formas de recorrer un árbol en profundidad:
    1. Pre-Orden (Raíz, SA Izquierdo, SA Derecho)
    2. In-Orden (SA Izquierdo, Raíz, SA Derecho)
    3. Post-Orden (SA Izquierdo, SA Derecho, Raíz)

- Recorrido en Anchura
  - También llamado recorrido por niveles:
    1. Anchura Izquierda-Derecha
    2. Anchura Derecha-Izquierda

Arboles Binarios
(Ej. Recorridos Profundidad)

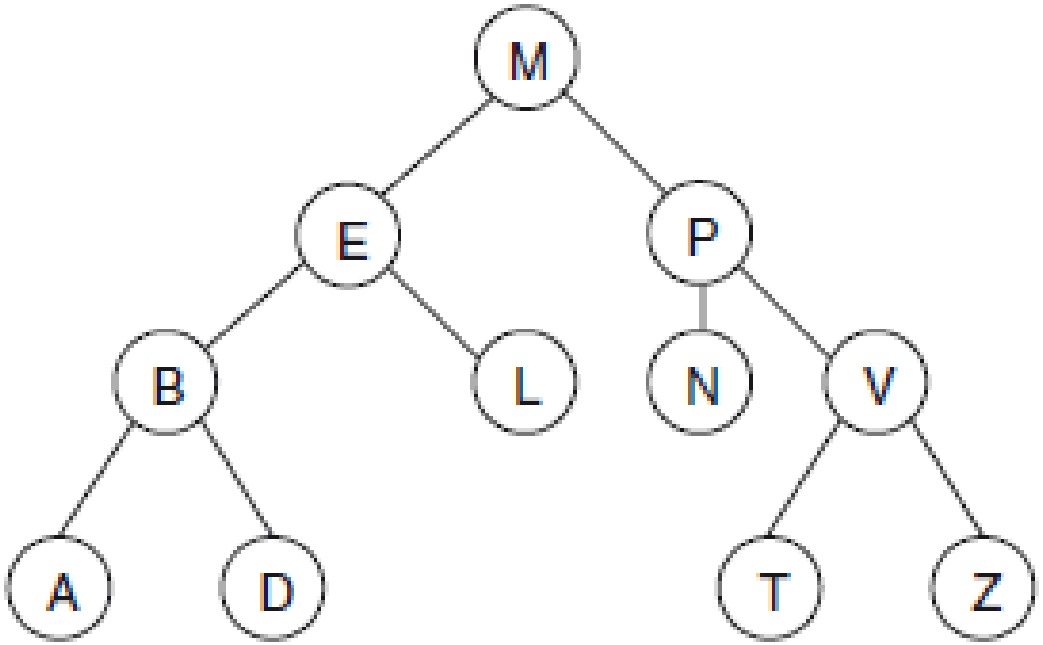


Pre-Orden: M, E, B, A, D, L, P, N, V, T, Z

In-Orden: A, B, D, E, L, M, N, P, T, V, Z

Post-Orden: A, D, B, L, E, N, T, Z, V, P, M

Arboles Binarios
(Ej. Recorridos Anchura)



Anchura I-D: M, E P, B L N V, A D T Z

Anchura D-I: M, P E, V N L B, Z T D A

Arboles Binarios de Búsqueda

- Características

1. El primer elemento se utiliza para crear el nodo raíz.

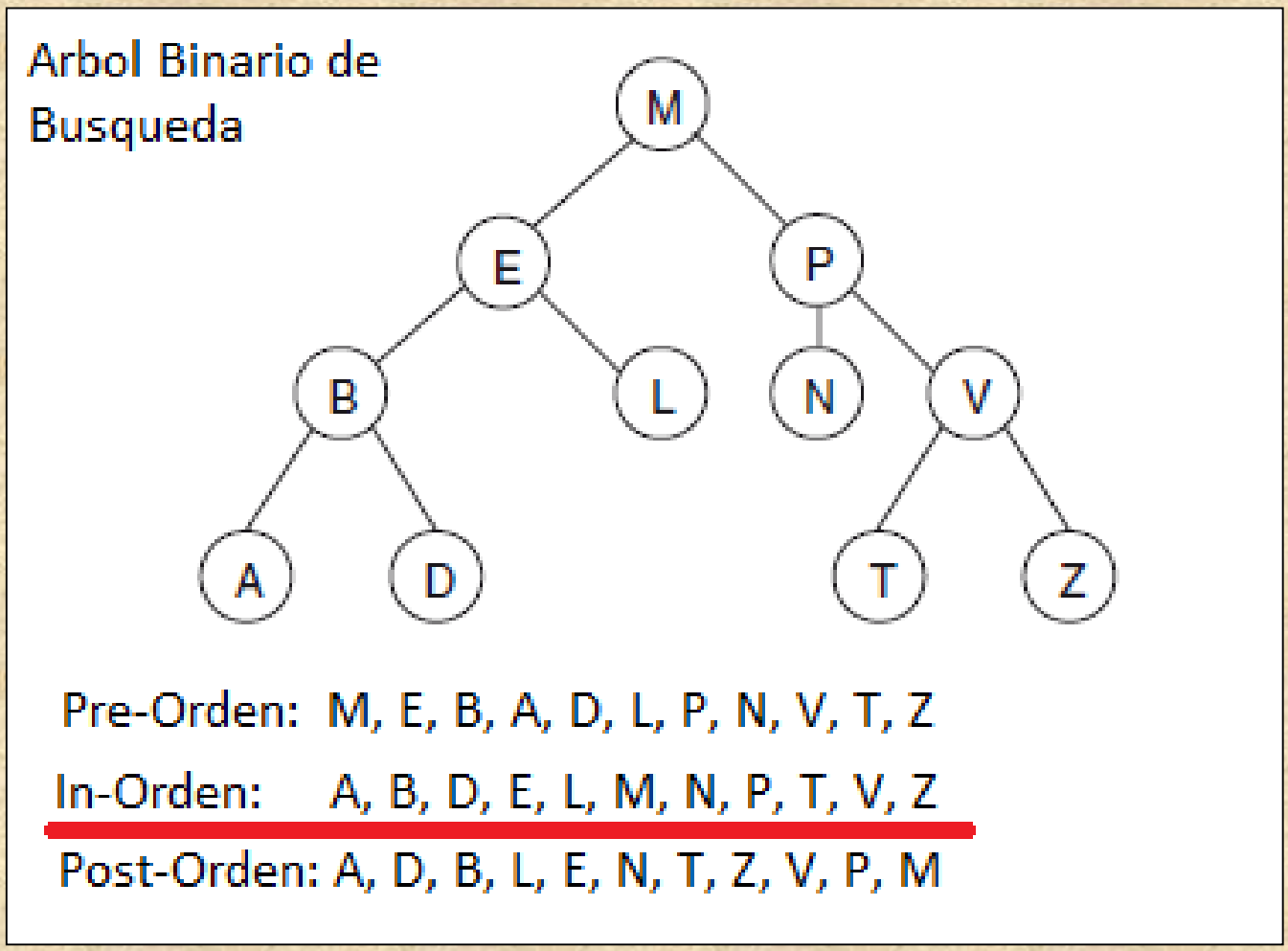
2. Los valores del árbol deben ser tales que pueda existir un orden (entero, real, lógico o carácter e incluso definido por el usuario si implica un orden).

3. En cualquier nodo todos los valores del subárbol izquierdo del nodo son menores al valor del nodo padres.

4. Todos los valores del subárbol derecho deben ser mayores o iguales que el valor del nodo padre.

SI < R < SD

Arboles Binarios Búsqueda (Ejemplo)



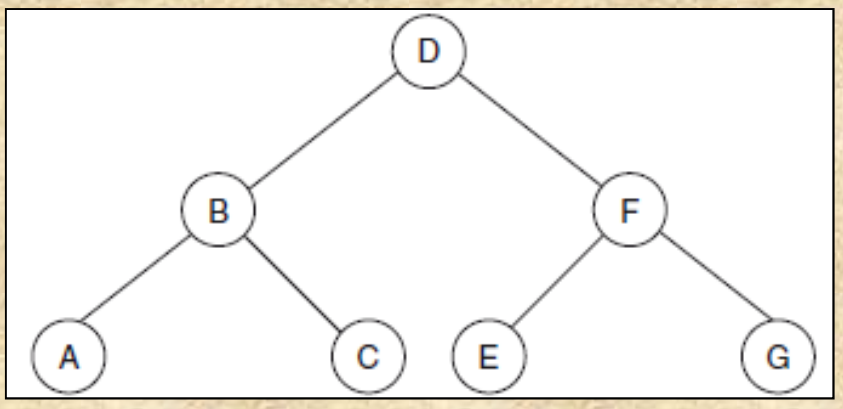
Arboles Binarios de Búsqueda

- Pasos para construir el árbol

Se tenemos los siguientes caracteres: D F E B A C G

Los pasos para la construcción del algoritmo son:

1. Nodo raíz del árbol: «D».
2. El siguiente elemento se convierte en el descendente derecho, dado que «F» alfabéticamente es mayor que «D».
3. A continuación, se compara «E» con el raíz. Dado que «E» es mayor que «D», pasará a ser un hijo de «F» y como «E» < «F» será el hijo izquierdo.
4. El siguiente elemento «B» se compara con el raíz «D» y como «B < D» y es el primer elemento que cumple esta condición, «B» será el hijo izquierdo de «D».
5. Se repiten los pasos hasta el último elemento.



Arboles Binarios de Búsqueda (Operación de Inserción)

- Insertar_Clave (Clave)

  1. Las claves nuevas se insertan siempre en la hojas.

  2. Si es un árbol vacío se crea la raíz.

  3. Caso contrario se compara la clave con la de la raíz y se decide si se va por la rama izquierda o derecha dependiente si es menor o mayor (igual en caso que soporte repetidas).

  4. El proceso 3 se repite por cada nodo del árbol hasta llegar a un nodo hoja.

  5. Se crea el nuevo nodo, se asigna la clave (dato) y se ve por cual de los 2 hijos se lo debe de conectar. Si es menor se lo conecta al hijo izquierdo, caso contrario al hijo derecho.

Arboles Binarios de Búsqueda (Operación de Eliminación)

- Eliminar_Clave (Clave)

1. Se debe ubicar la clave (Búsqueda Binaria)
2. Una vez ubicada puede suceder lo siguiente:
   a). Es un nodo hoja (Se elimina directamente)
   b). Es un nodo que posee un solo hijo
      Según si tiene izquierdo o derecho se hace el puente con el nodo padre (similar a la eliminación de un nodo en una lista enlazada)
   c). Es un nodo que posee los 2 hijos
      No se puede eliminar físicamente el nodo.
      Se busca la una clave alternativa para reemplazar en el nodo.
      Se hace una eliminación lógica de la clave.
      Luego se elimina el nodo de la clave reemplazante.

Arboles Binarios de Búsqueda (Ejemplos 1 hijo)

En la Figura 13.20 se muestra el caso de eliminación de un nodo con un subárbol en un gráfico comparativo antes y después de la eliminación.

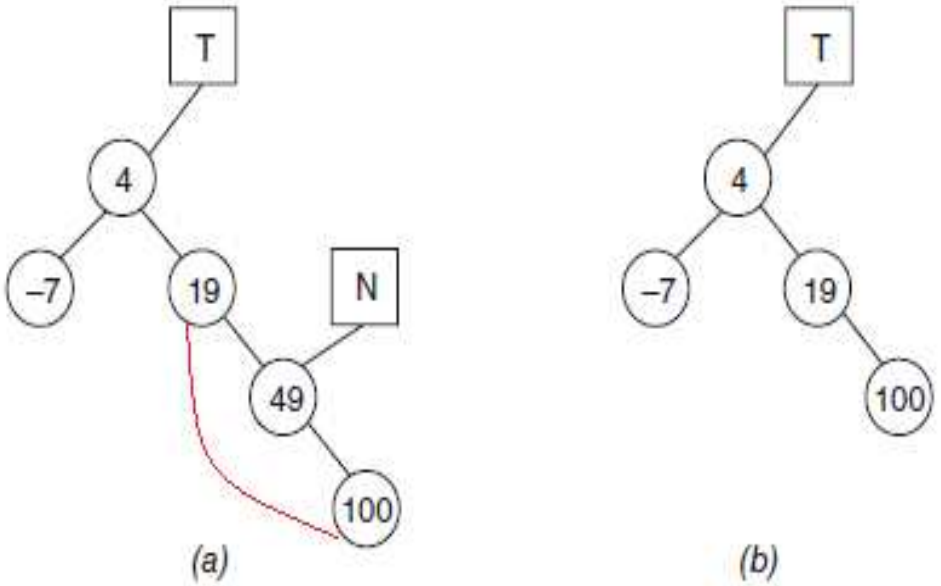


Figura 13.20. Eliminación de un nodo con un subárbol.

Arboles Binarios de Búsqueda (Ejemplos 2 hijos)

En la Figura 13.21 se muestra el caso de la eliminación de un nodo (27) que tiene dos subárboles no nulos. En este caso se busca el nodo sucesor cuyo campo de información le siga en orden ascendente, es decir, 42, se intercambia entonces con el elemento que se desea borrar, 27.

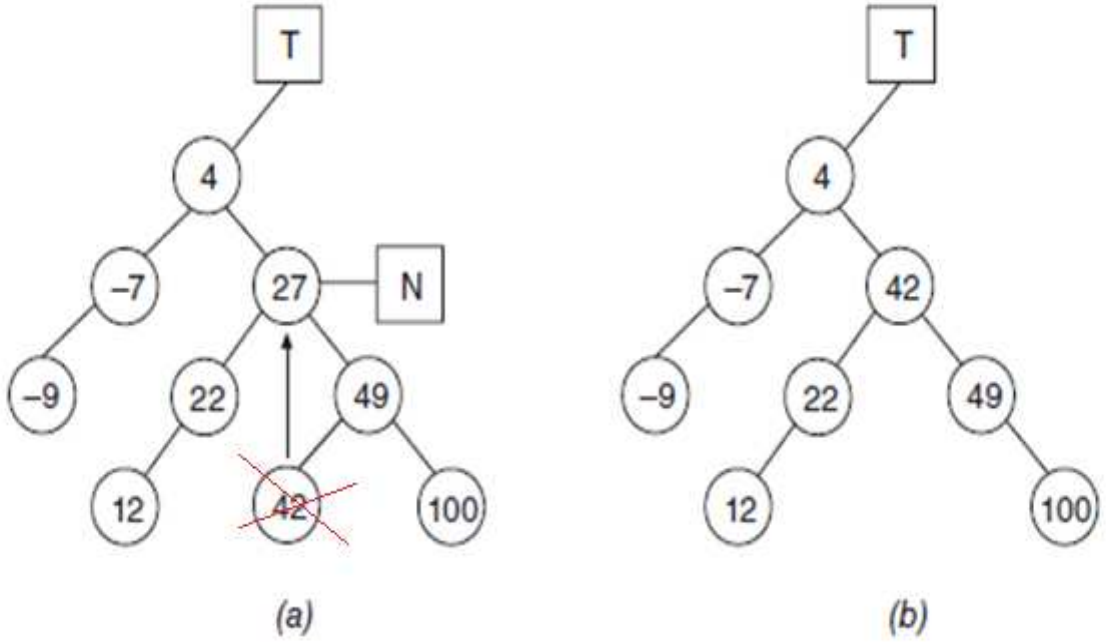


Figura 13.21. Eliminación de un nodo con dos subárboles no nulos.

Arboles Binarios de Búsqueda Balanceados (AVL)

- Un **árbol AVL** es un tipo especial de árbol binario ideado por los matemáticos rusos Adelson, Velskii y Landis. Fue el primer árbol de búsqueda binario auto-balanceable que se ideó.
- Problema del Árbol Binario sin Balanceo

  1. Depende de la generación de las claves.

  2. Si la clave inicial es muy pequeña se tendrá un árbol con un subárbol izquierdo muy pequeño y un subárbol derecho muy extenso.

  3. No garantiza que la búsqueda dentro del árbol tenga una complejidad del O(n) = log(n).

  4. Los tiempos de búsqueda pueden degenerar en tiempos de búsqueda lineales O(n) = n.

  La solución al problema es «Balancear el Árbol»

Arboles Binarios de Búsqueda Balanceados (AVL)

- Características del Balanceo

1. Para poder balancear el árbol se debe utilizar en cada nodo un dato adicional que se llama «factor de equilibrio» (FE) que se define como la diferencia de alturas entre los sub-arboles izquierdo y derecho.

FE = -Altura(SI) + Altura(SD)

2. Los valores posibles para el «FE» son: [-1, 0, 1]

3. Fuera de los valores posibles se debe reestructurar el árbol desde esa nodo (clave) hacia abajo.

4. Se debe conectar al resto del árbol la nueva raíz del subárbol resultado de la transformación (balanceo).

Arboles Binarios de Búsqueda Balanceados (AVL)

- Casos de Des-balanceo

1. Existen 4 casos de des-balanceo que debe ser resuelto y que son detectados a través del factor de equilibrio.

2. Para detectar estos se usan el «FE» cuando se va fuera del rango [-1..1]. Es decir que podría tomar los valores «-2» y «2».

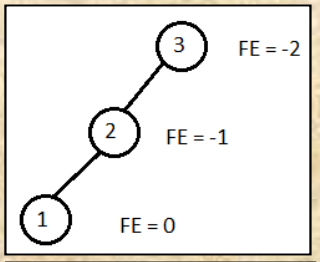
3. Cuando el valor es -2 esto indica que el des-balanceo se produce por el subárbol izquierdo, caso contrario por el derecho.

4. A su vez se analizan los «FE» de los hijos en problema, esto derivada en diferentes tipos de rotaciones que se deben producir para volver a tener el árbol balanceado.

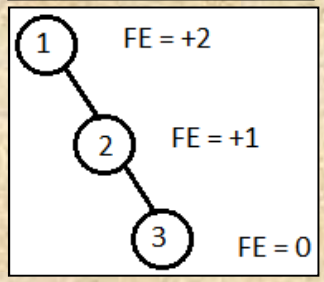
Arboles Binarios de Búsqueda Balanceados (AVL)

- Claves que producen los 4 tipos de Des-balanceo

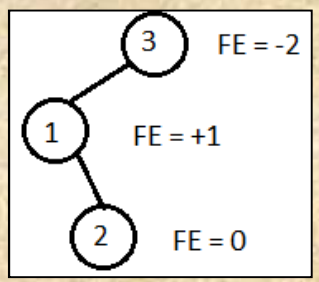
1. Insertar las claves (3,2,1)



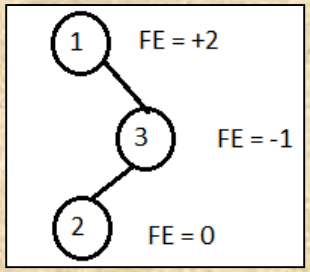
2. Insertar las claves (1, 2, 3)



3. Insertar las claves (3, 1, 2)



4. Insertar las claves (1, 3, 2)



Arboles Binarios de Búsqueda Balanceados (AVL)

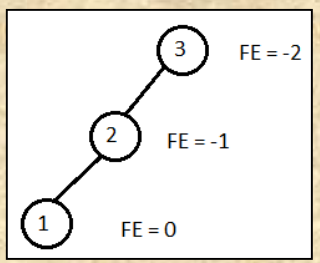
- Tabla de Desbalanceo / Rotaciones

Para recordar que rotación se debe aplicar usamos la siguiente tabla:

FE (Raíz)	Hijo con Desequilibrio	FE (Hijo)	Tipo Rotación	Sentido Rotación
-2	Izquierdo	-1	Simple	Derecha
-2	Izquierdo	0	Simple	Derecha
-2	Izquierdo	1	Doble	Derecha-Izquierda
2	Derecho	1	Simple	Izquierda
2	Derecho	0	Simple	Izquierda
2	Derecho	-1	Doble	Izquierda-Derecha

Arboles Binarios de Búsqueda Balanceados (AVL)

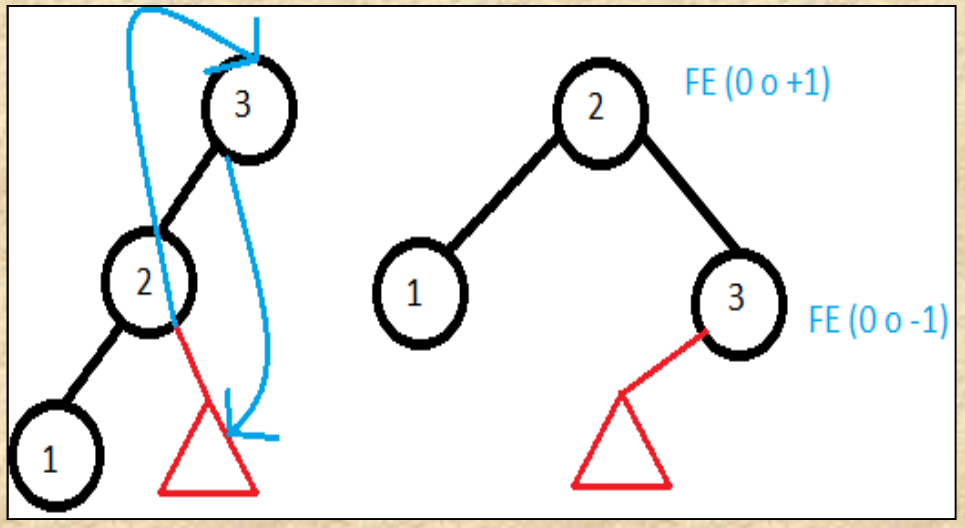
- Rotación Simple Derecha (Des-Izq)



1. Insertar las claves (3,2,1)

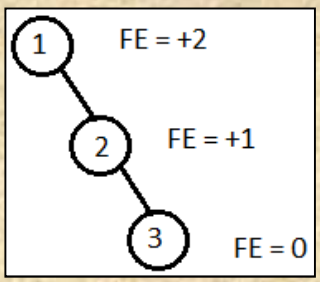
Rotación:
a. HI(3) = HD(2)
b. HD(2)= Raíz(3)

Nuevos FE:
Si FE(2) = -1 Entonces
FE(3) = 0
FE(2) = 0
Sino
FE(3) = -1
FE(2) = +1



Arboles Binarios de Búsqueda Balanceados (AVL)

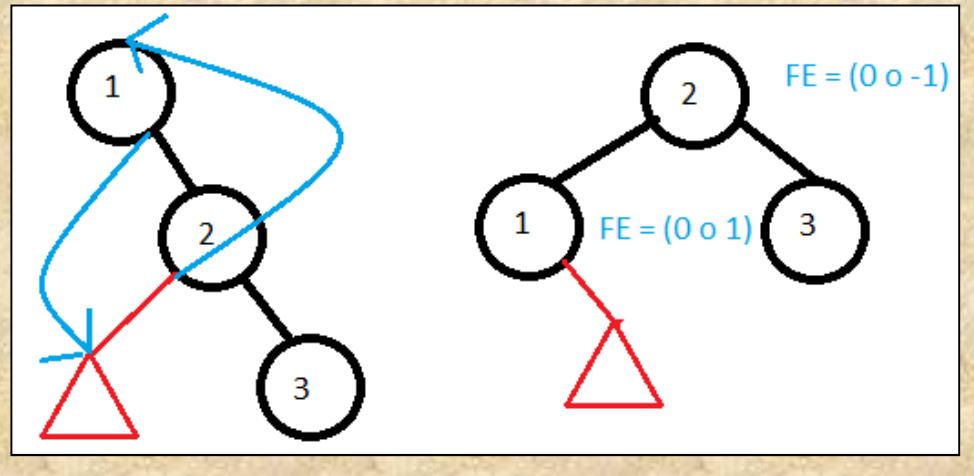
- Rotación Simple Izquierda (Des-Der)



1. Insertar las claves (1,2,3)

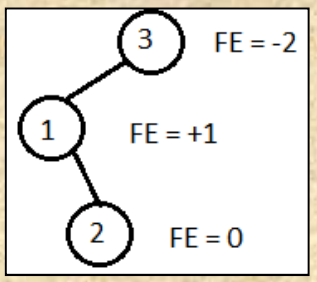
Rotación:
a. HD(1) = HI(2)
b. HI(2)= Raíz(1)

Nuevos FE:
Si FE(2) = 1 Entonces
FE(1) = 0
FE(2) = 0
Sino
FE(1) = +1
FE(2) = -1



Arboles Binarios de Búsqueda Balanceados (AVL)

- Rotación Doble Derecha-Izquierda (Des-ID)



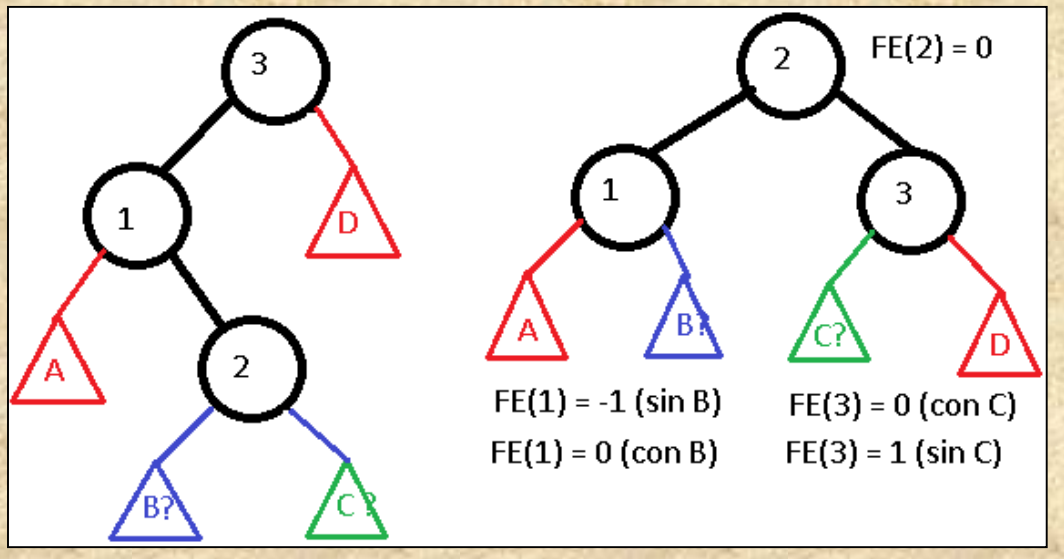
1. Insertar las claves (3,1,2)

Rotación:

a. HI(3) = HD(2)
b. HD(2)= Raíz(3)
c. HD(1)= HI(2)
d. HI(2) = N(1)

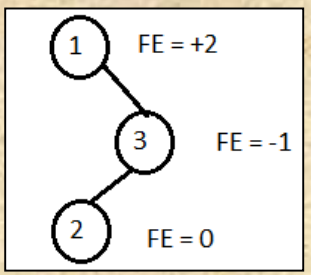
Nuevos FE:

Si FE(2) = -1 entonces
FE(3) = 1
Sino FE(3) = 0
Si FE(2) = 1 entonces
FE(1) = -1
Sino FE(1) = 0



Arboles Binarios de Búsqueda Balanceados (AVL)

- Rotación Doble Izquierda-Derecha (DesDI)



1. Insertar las claves (1,3,2)

Rotación:

a. HD(1) = HI(2)
b. HI(2)= Raíz(1)
c. HI(3)= HD(2)
d. HD(2) = N(3)

Nuevos FE:

Si FE(2) = -1 entonces
FE(3) = 1
Sino FE(3) = 0
Si FE(2) = 1 entonces
FE(1) = -1
Sino FE(1) = 0