Árboles

Definición

Un árbol dirigido es una estructura:

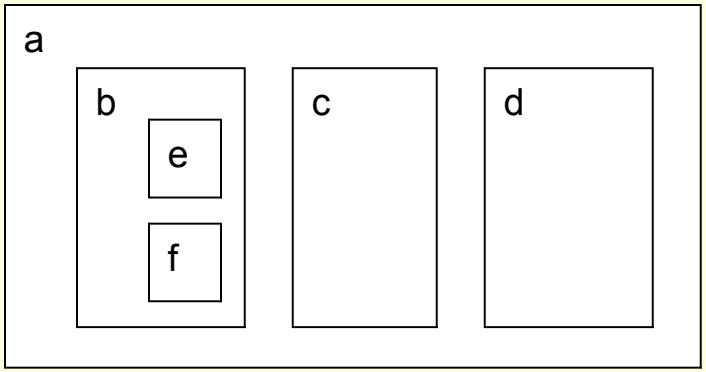
- ***Jerárquica*** porque los componentes están a distinto nivel.
- ***Organizada*** porque importa la forma en que esté dispuesto el contenido.
- ***Dinámica*** porque su forma, tamaño y contenido pueden variar durante la ejecución.

Un árbol puede ser:

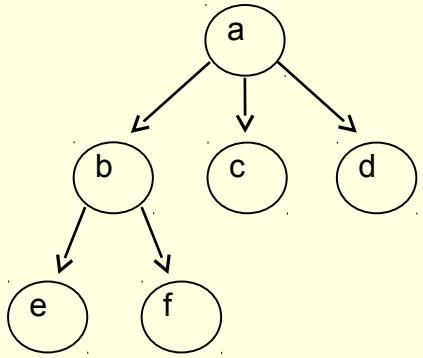
- vacío,
- Una raíz + subárboles.

Representación de un Árbol.

- Mediante diagramas de Venn



- Mediante círculos y flechas



- Mediante paréntesis anidados:

  (a (b (e,f), c, d))

Conceptos Básicos

- ***Si hay un camino de A hasta B***, se dice que A es antecesor de B, y que B es sucesor de A.
- ***Padre*** es el antecesor inmediato de un nodo
- ***Hijo***, cualquiera de sus desc}ientes inmediatos.
- ***Descendiente*** de un nodo, es cualquier sucesor de dicho nodo.
- ***Hermano*** de un nodo, es otro nodo con el mismo padre.
- ***Generación***, es un conjunto de nodos con la misma profundidad.

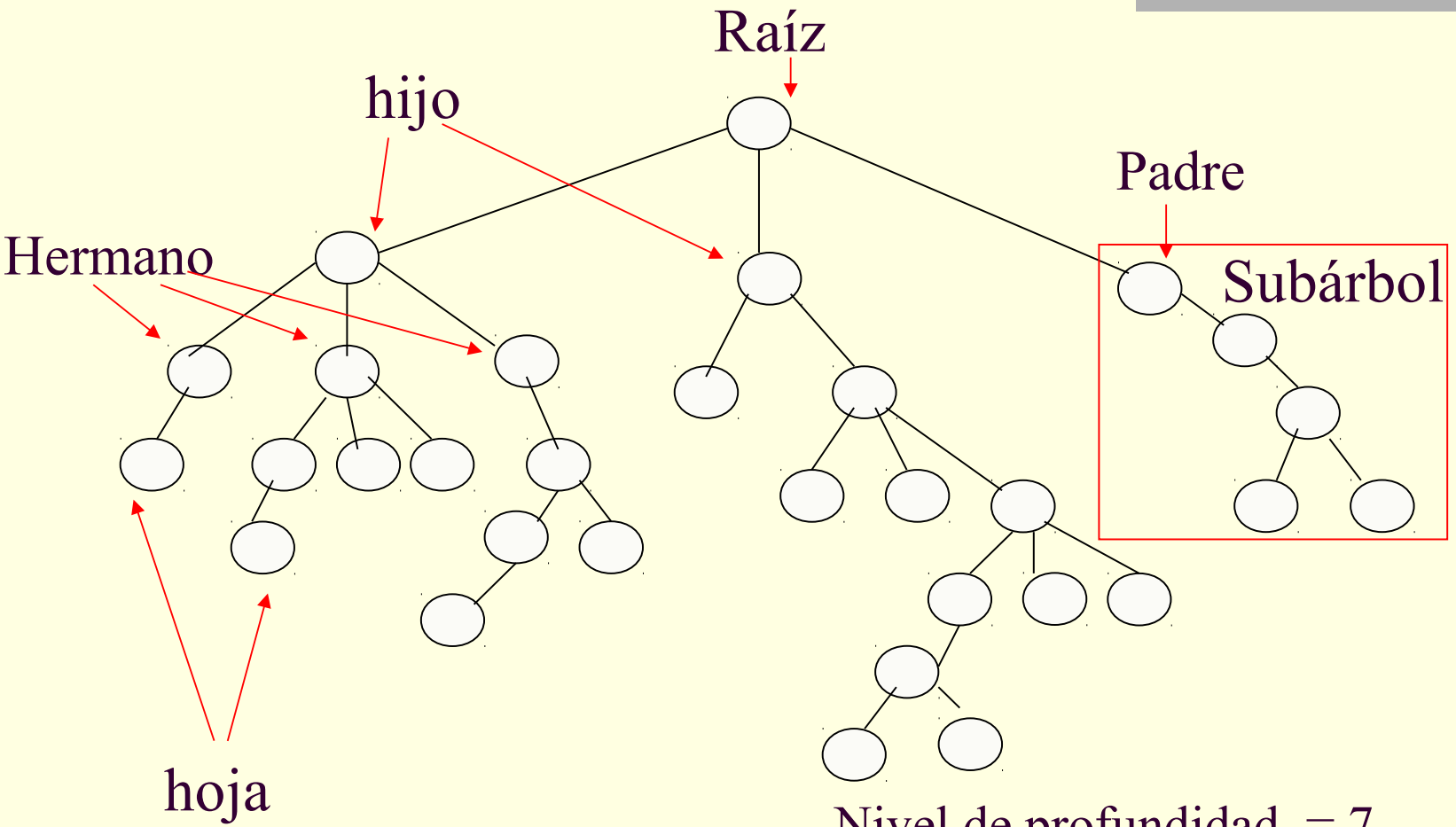
Conceptos Básicos (cont.)

- ***Raíz*** es el nodo que no tiene ningún predecesor (sin padre).
- ***Hoja*** es el nodo que no tiene sucesores (sin hijos) (Terminal). Los que tienen predecesor y sucesor se llaman nodos interiores.
- ***Rama*** es cualquier camino del árbol.
- ***Bosque*** es un conjunto de árboles desconectados.
- ***Nivel*** o ***profundidad de un nodo,*** es la longitud del camino desde la raíz hasta ese nodo.
  El nivel puede de}irse como 0 para la raíz y nivel (predecesor)+1 para los demás nodos.

Conceptos Básicos (cont.)

- Los nodos de la misma generación tienen el mismo nivel.
- ***Grado de un nodo***, es el número de flechas que salen de ese nodo (hijos). El número de las que entran siempre es uno.
- ***Grado de un árbol***, es el mayor grado que puede hallarse en sus nodos.
- ***Longitud del camino entre 2 nodos***: es el número de arcos que hay entre ellos.

Conceptos Básicos (cont.)



Nivel de profundidad = 7

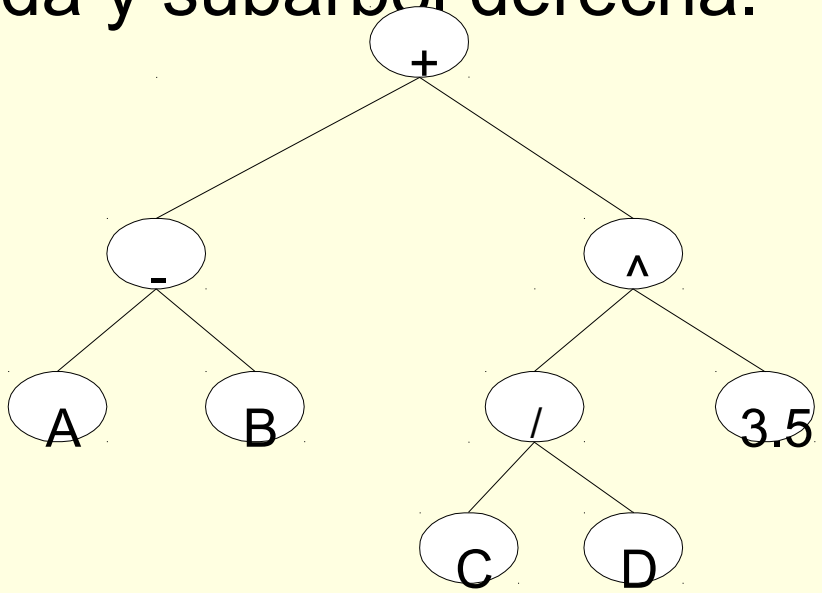
Grado de un nodo = 3

Grado del árbol = 3

Tipos de árboles

Un árbol ordenado: Es aquel en el que las ramas de los nodos están ordenadas.

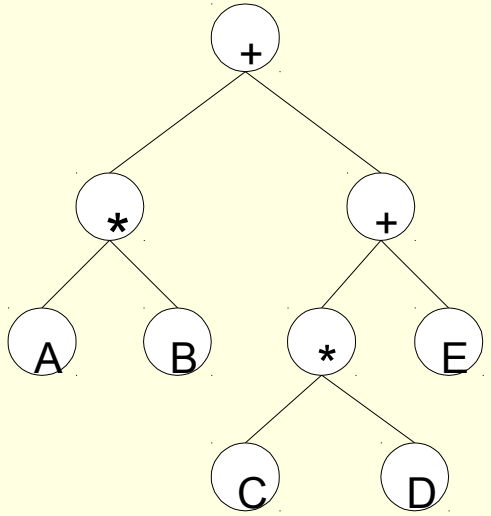
- Los de grado 2 se llaman árboles binarios.
- Cada árbol binario tiene un subárbol izquierda y subárbol derecha.



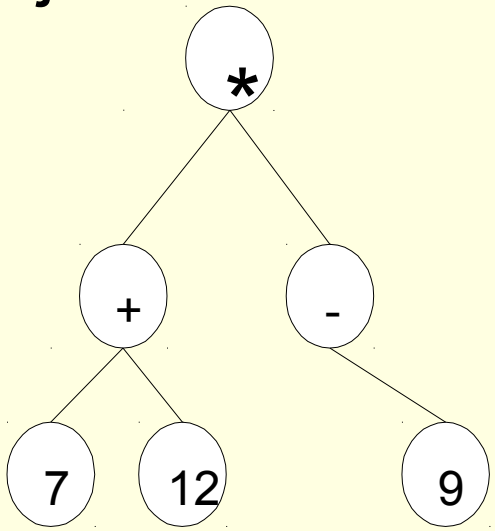
Tipos de árboles (cont.)

Árboles de expresión

- Representan un orden de ejecución



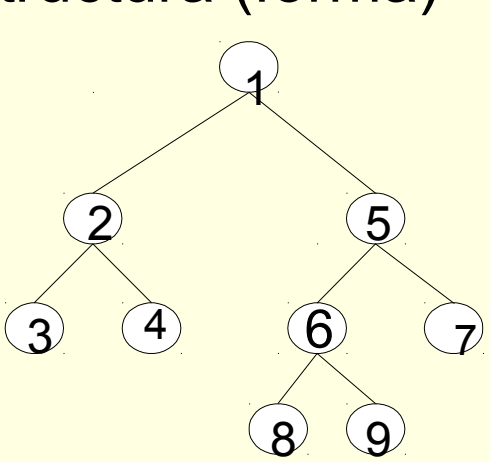
(A* B) + C * D + E

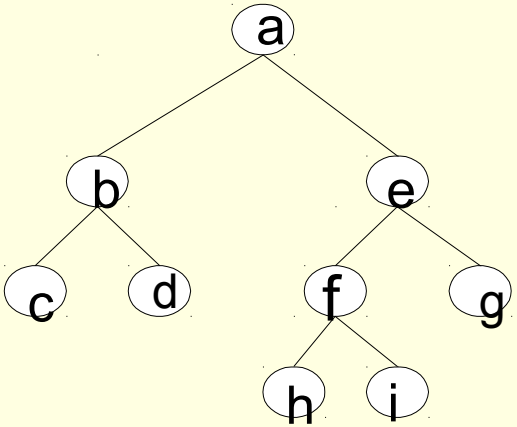


(7 + 12) * (-9) → -171

Tipos de árboles (cont.)

- ***Árboles similares***: Los que tienen la misma estructura (forma)



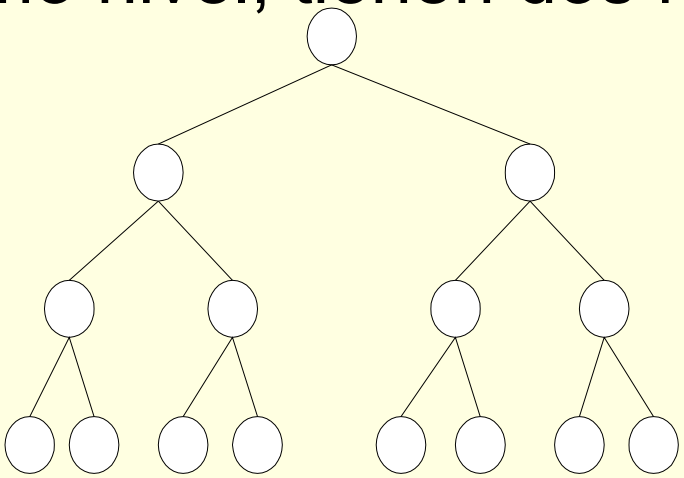


- ***Árboles Equivalentes:*** Son los árboles similares y sus nodos contienen la misma información.
- ***Árboles n-ario:*** Es un árbol ordenado cuyos nodos tiene N subárboles, y donde cualquier número de subárboles puede ser árboles vacíos

Tipos de árboles (cont.)

Árbol binario completo:

- Es un árbol en el que todos sus nodos, excepto los del ultimo nivel, tienen dos hijos.



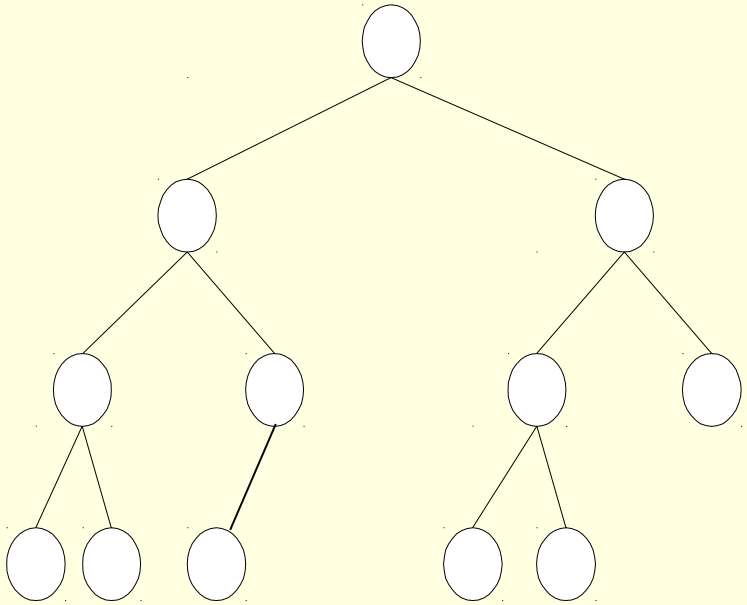
- Número de nodos en un árbol binario completo = $2^h - 1$ (en el ejemplo h = 4, → 15) esto nos ayuda a calcular el nivel de árbol necesario para almacenar los datos de una aplicación.

Árboles Binarios de Búsqueda (ABB)

Udem OT 2005

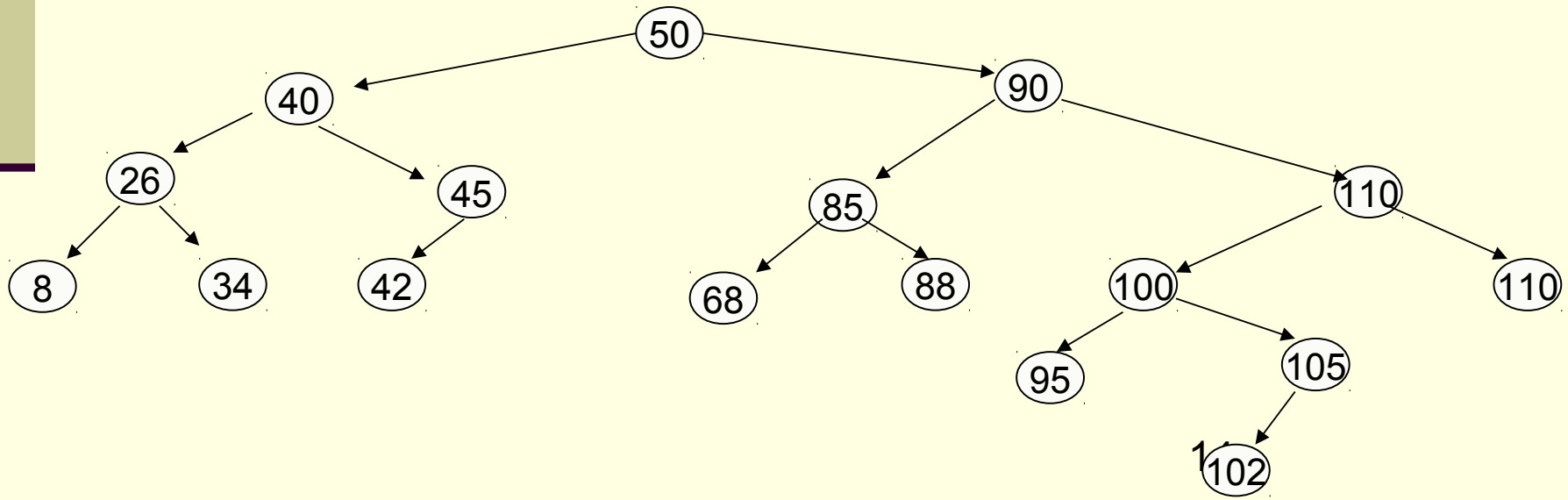
Árboles Binarios de Búsqueda

- Un árbol es un *ABB* si éste es binario y sus nodos son subárboles de búsqueda binarios y contienen información ordenada de tal que todos los elementos a la izquierda de la raíz son menores a la raíz y todos lo elementos a la derecha de la raíz son mayores a la raíz.



Características de un ABB

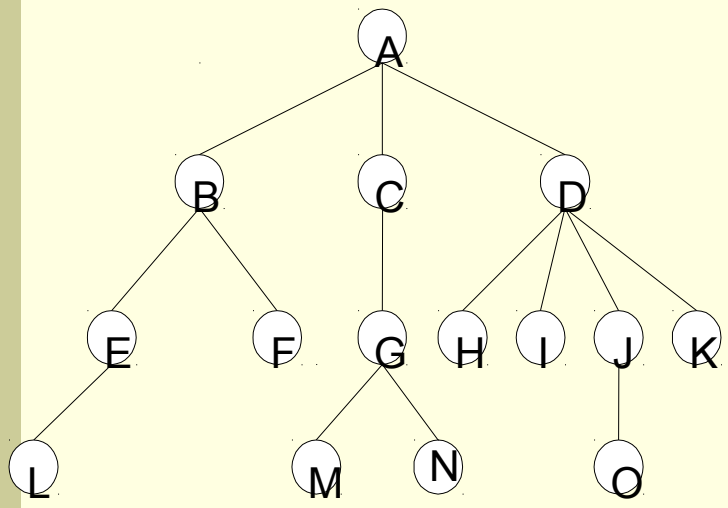
- Todos los nodos a la izquierda son menores al padre.
- Todos los nodos a la derecha son mayores al padre.
- Y solo pueden tener 2 hijos a lo mucho.



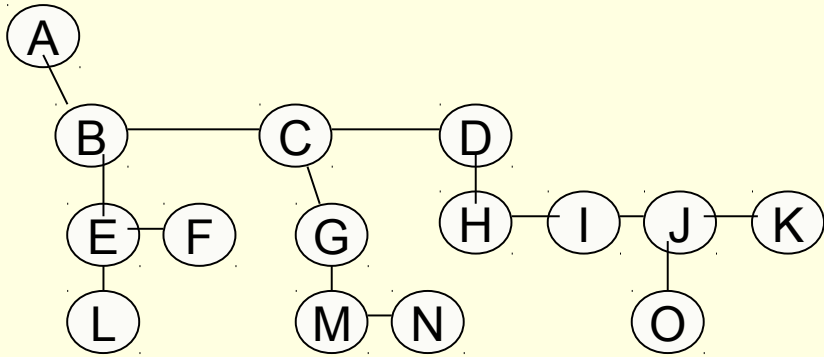
Conversión de un árbol general en un árbol binario

- Los hermanos se enlazan en forma horizontal (lineal)
- Se enlaza en forma vertical el padre con el hijo que se encuentra mas a la izquierda y se elimina el enlace de este padre con los demás hijos.
- Se rota el diagrama resultante 45 grados hacia la izquierda.

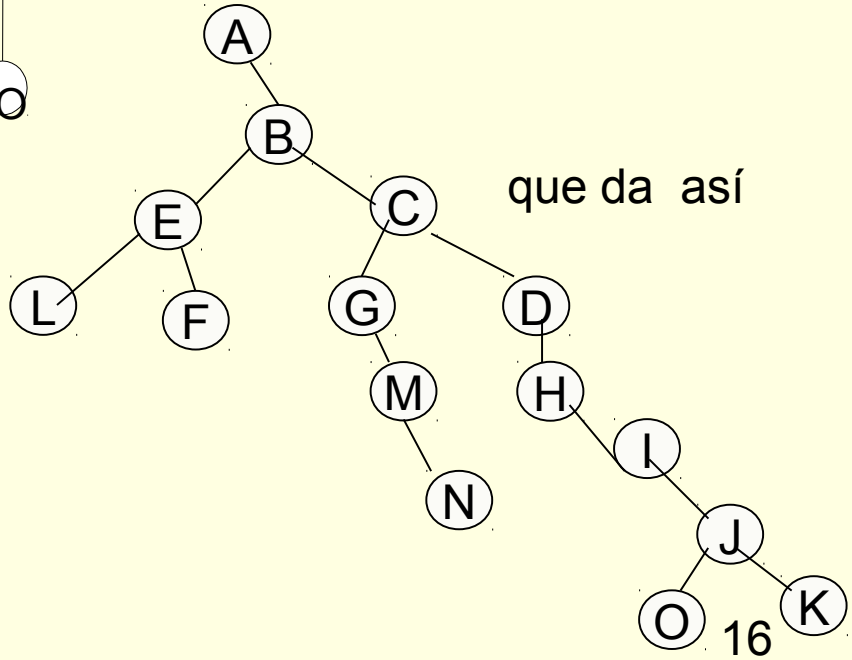
Conversión de un árbol general en un árbol binario (cont.)



=

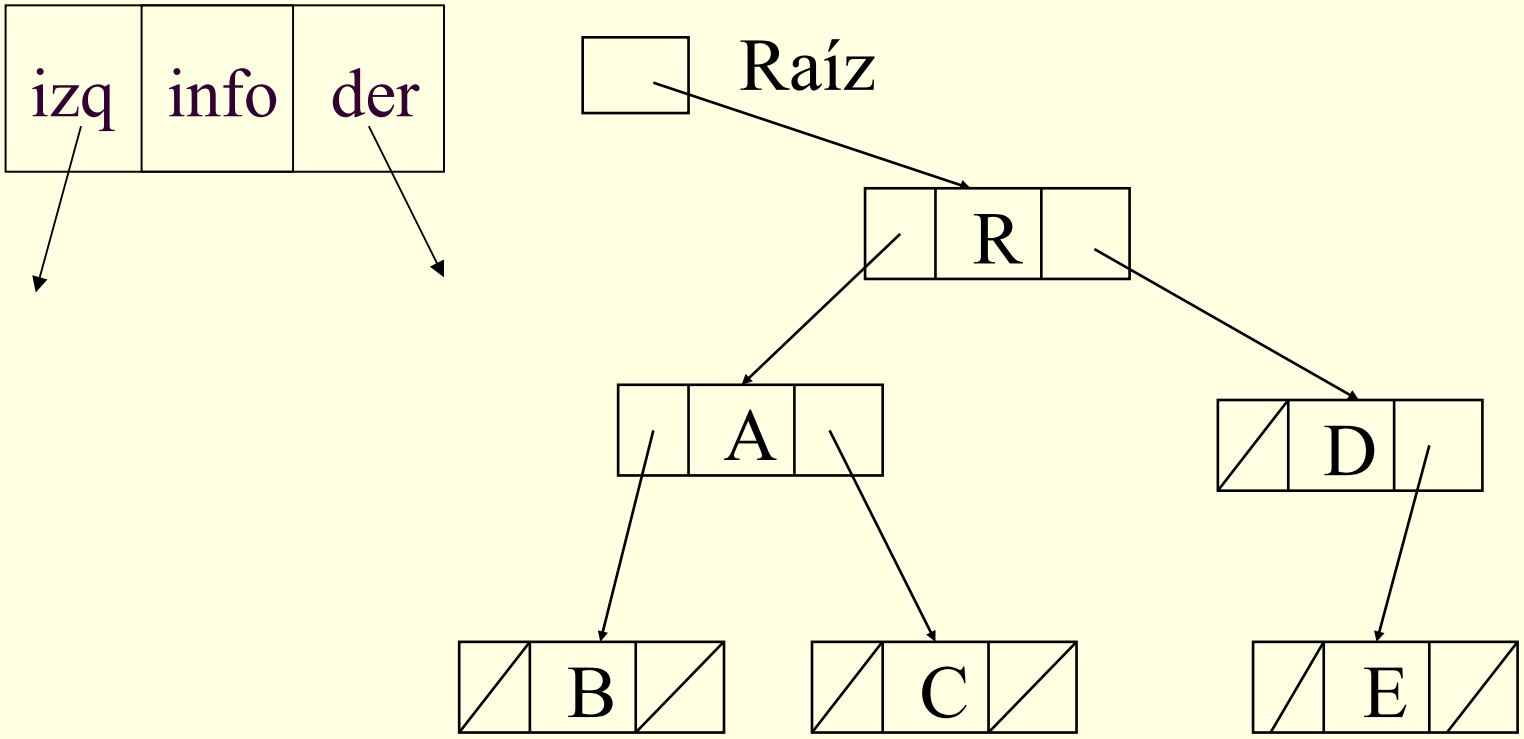


que da así



Representación de un árbol binario en la memoria.

- Cada noto tiene la siguiente forma:



Clase nodo de un ABB

```
Class Nodo{
   nodo izq;
   nodo der;
   int dato;
}
```

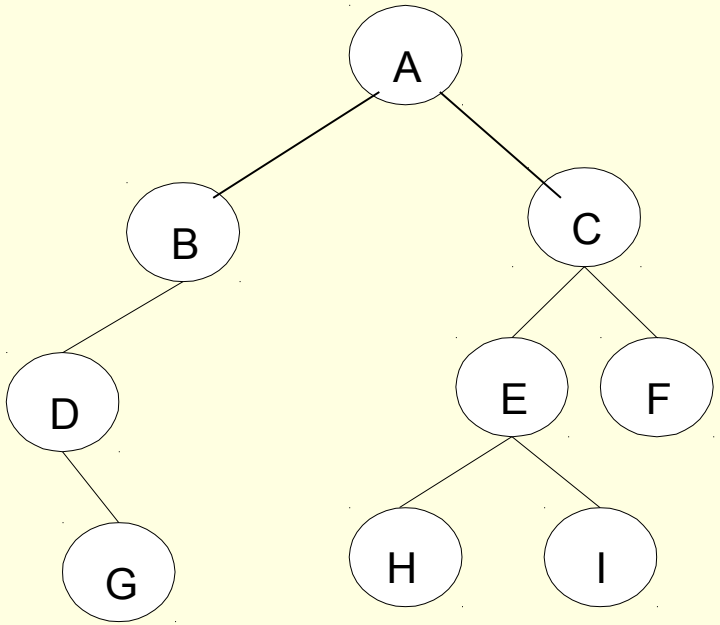
Operaciones sobre un árbol

- Recorrer árbol
  - Preorden
  - Inorden
  - Postorden
- Inserción nodo
- Eliminar nodo
- Buscar nodo con información
- Sumar los nodos
- Calcular profundidad del árbol
- Contar nodos
- Contar hojas.

Recorridos de un árbol de Búsqueda Binaria (ABB)

- Recorrido en preorden (prefijo)
  - Visita la raíz.
  - Recorre el subárbol izquierdo.
  - Recorre el subárbol derecho.

RID



Preorden = A B D G C E H I F

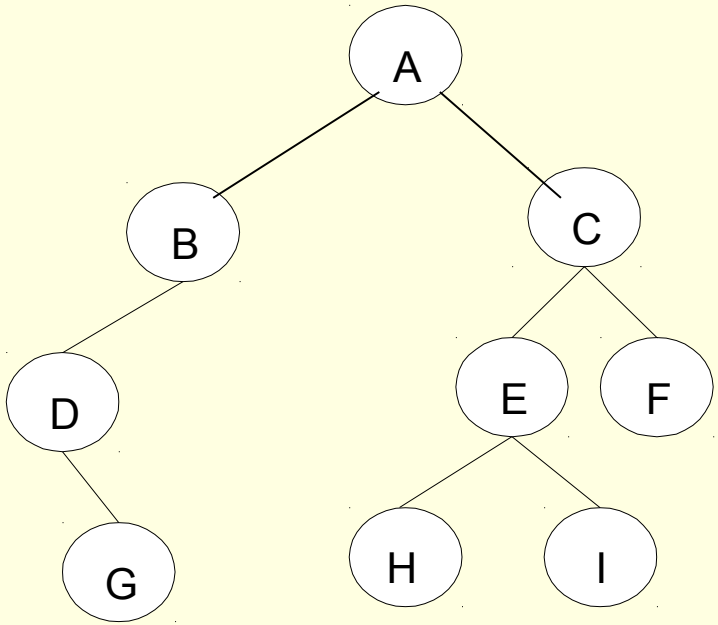
Recorridos de un árbol de Búsqueda Binaria (ABB) (cont.)

PREORDEN (NODO)
{NODO es un dato de tipo apuntador}
{INFO, IZQ, DER son campos del registro}
Si nodo != null entonces
{
Visitar NODO {escribir NODO.INFO}
llamar a preorden con NODO.IZQ
{llamada recursiva a preorden con la rama izquierda del nodo en cuestión}
llamar a preorden con NODO.DER
{llamada recursiva a preorden}
}

Recorridos de un árbol de Búsqueda Binaria (ABB) (cont.)

- Recorrido en inorden (infijo)
  - Recorre el subárbol izquierdo.
  - Visita la raíz
  - Recorre el subárbol derecho.

IRD



Inorden: D G B A H E I C F

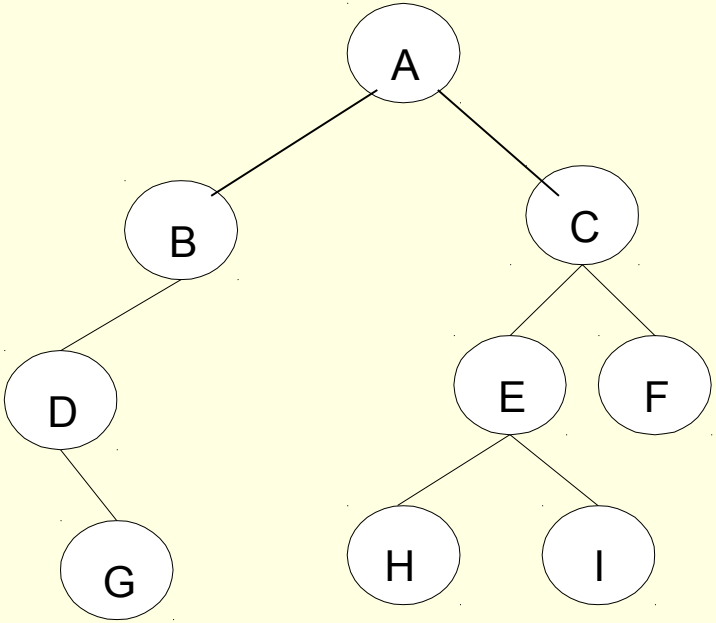
Recorridos de un árbol de Búsqueda Binaria (ABB) (cont.)

```
INORDEN (NODO)
   {NODO es un apuntador a registro}
Si NODO != null entonces
        {
                INORDEN (NODO.IZQ);
                Escribir NODO.INFO;
                INORDEN (NODO.DER);
        }
```

Recorridos de un árbol de Búsqueda Binaria (ABB) (cont.)

- Recorrido en postorden (postfijo)
  - Recorre el subárbol izquierdo.
  - Recorre el subárbol derecho.
  - Visita la raíz.

IDR



Postorden : G D B H I E F C A

Recorridos de un árbol de Búsqueda Binaria (ABB) (cont.)

POSTORDEN (NODO)

```
Si NODO != null entonces
        {
                POSTORDEN (NODO.IZQ);
                POSTORDEN (NODO.DER);
                Escribir NODO.INFO;
        }
```

Inserción en un ABB

- La *inserción* es una operación que se puede realizar eficientemente en un árbol binario de búsqueda. La estructura crece conforme se inserten elementos al árbol.
- Los pasos que deben realizarse para insertar un elemento a un ABB son los siguientes:
  - Debe compararse el valor o dato a insertar con la raíz del árbol. **Si es mayor**, debe avanzarse hacia el **subárbol derecho**. **Si es menor**, debe avanzarse hacia el **subárbol izquierdo**.

Inserción en un ABB (cont.)

- Repetir sucesivamente el paso 1 hasta que se cumpla alguna de las siguientes condiciones
  - El subárbol derecho es igual a vació, o el subárbol izquierdo es igual a vació; en cuyo caso se procederá a insertar el elemento en el lugar que le corresponde.
  - El valor o dato que quiere insertarse es igual a la raíz del árbol; en cuyo caso no se realiza la inserción.

Inserción en un ABB (cont.)

Algoritmo

```
Si  NODO ≠ Null{
      Si  (INFOR < NODO.INFO)
       Regresar a INSERCION1 con NODO.IZQ e INFOR
       sino
            si ( INFOR >  NODO.INFO)
            Regresar a INSERCION1 con NODO.DER e INFOR
            sino
                Escribir “El nodo ya se encuentra en el árbol”
      } // } si
else
       CREA (OTRO) {Crear un nuevo nodo}
       Hacer OTRO.IZQ = null,
       OTRO.DER = null,
       OTRO.INFO  = INFOR  y NODO = OTRO
}
```

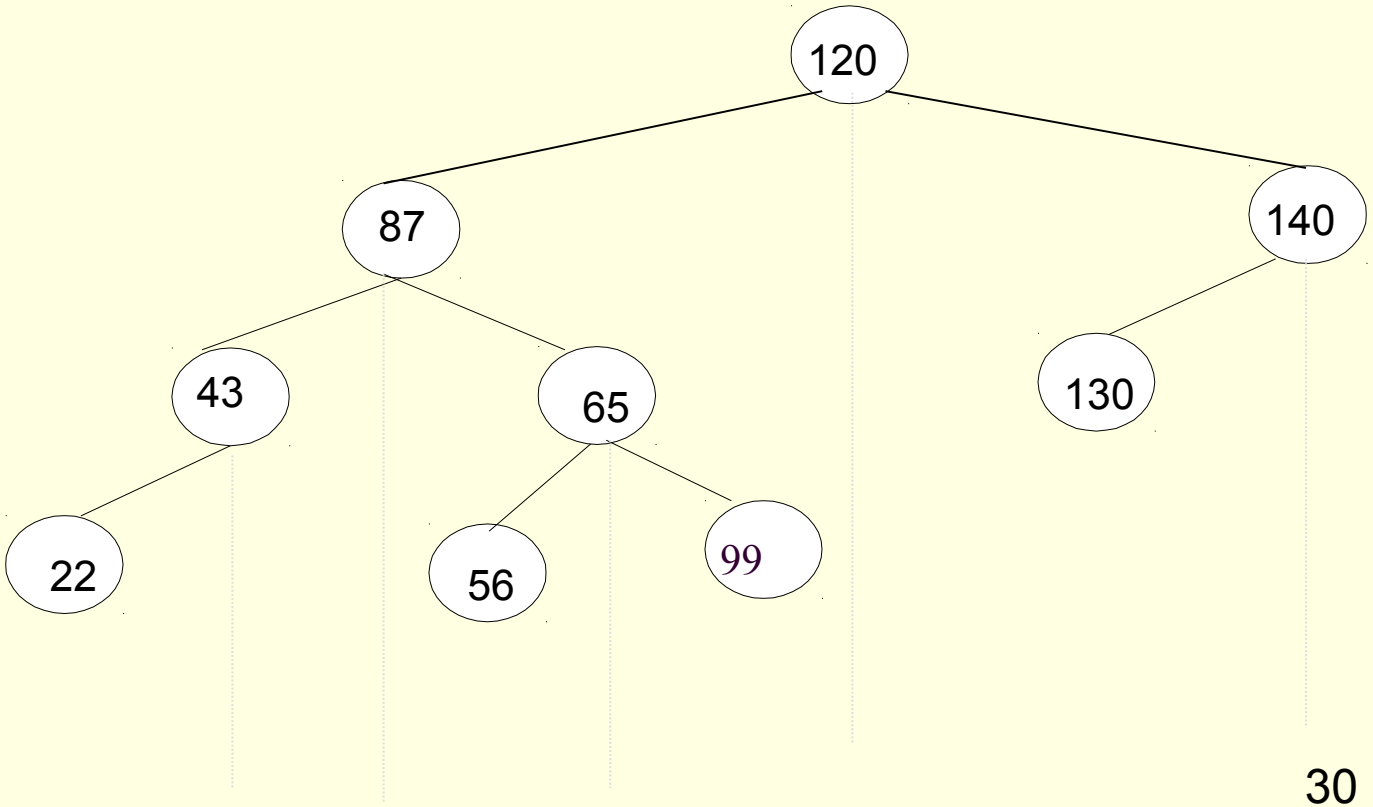
Inserción en un ABB (cont.)

- Supóngase que quieren insertarse las siguientes los siguientes datos en un árbol binario de búsqueda que se encuentra vació.

120 –87 – 43 – 65 – 140 – 99 – 130 – 22 – 56

Inserción en un ABB (cont.) Solución

120 –87 – 43 – 65 – 140 – 99 – 130 – 22 – 56



Eliminar un nodo

Para eliminar un nodo existen los siguientes casos:

1. Si el elemento a borrar es Terminal (hoja),
2. Si el elemento a borrar tiene un solo hijo,
3. Si el elemento a borrar tiene los dos hijo,

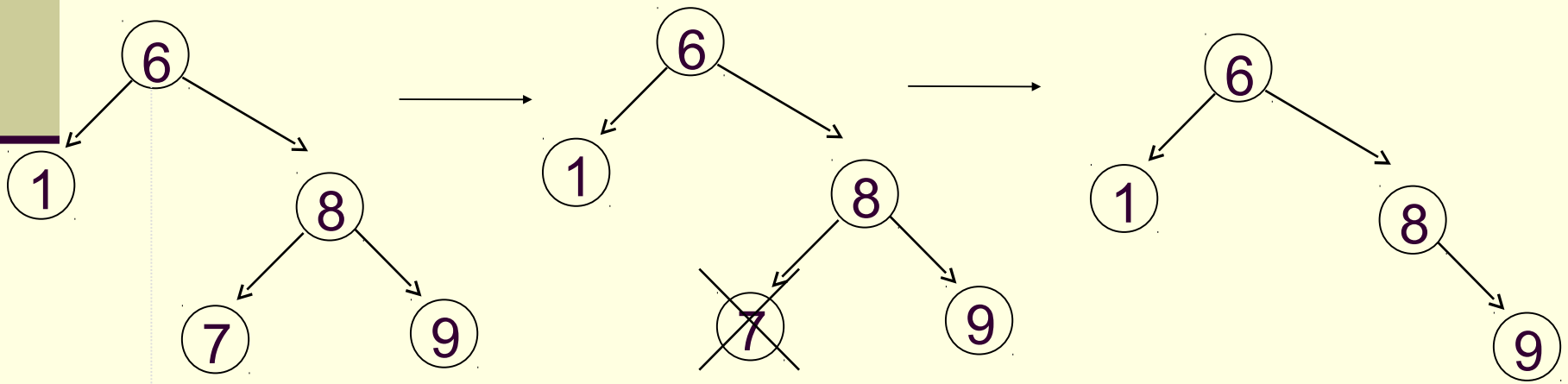
Eliminar un nodo (cont.)

- Caso 1

Si el elemento a borrar es terminal (hoja), simplemente se elimina.

aux = aux.izq = null

Ejemplo eliminar nodo 7

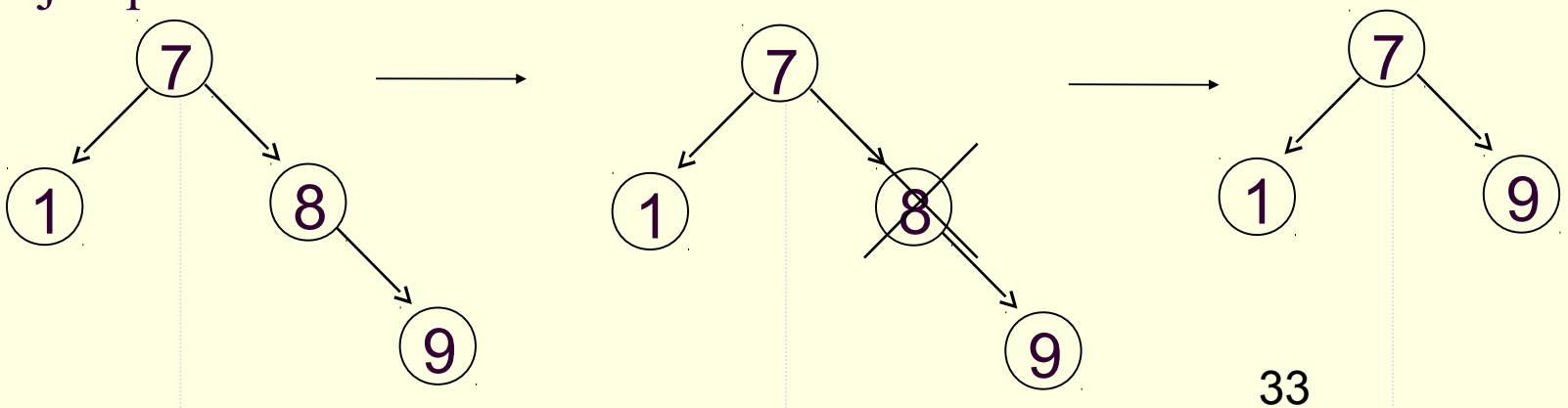


Eliminar un nodo (cont.)

- Caso 2

Si el elemento a borrar tiene un solo hijo, entonces tiene que sustituirlo por el hijo

Ejemplo: eliminar nodo 8

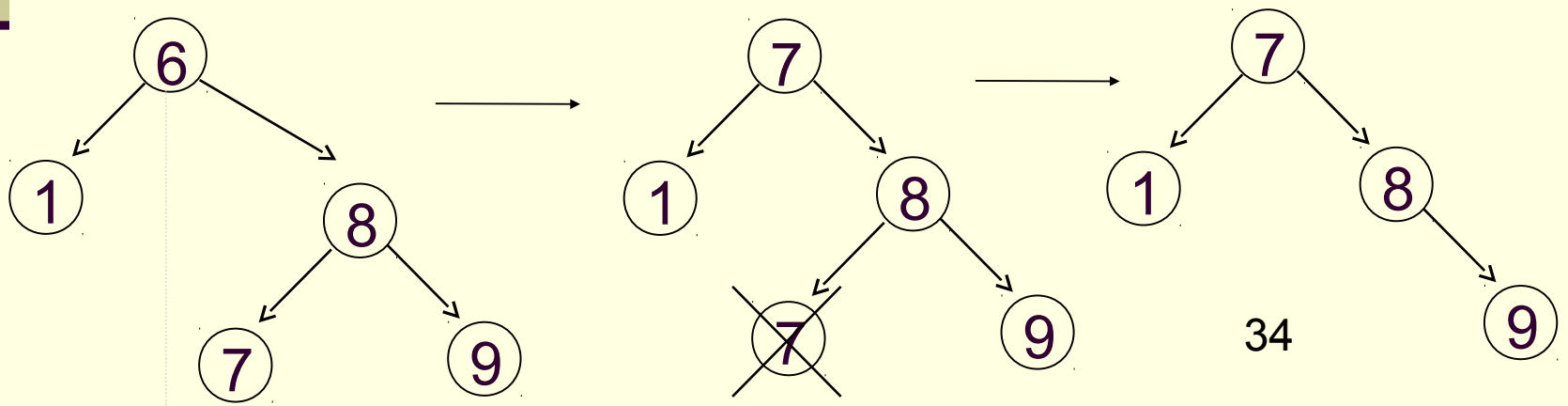


Eliminar un nodo (cont.)

- Caso 3

Si el elemento a borrar tiene los dos hijos, entonces se tienen que sustituir por el nodo que se encuentra mas a la izquierda en el subárbol derecho, o por el nodo que se encuentra mas a la derecha en el subárbol izquierdo.

Ejemplo: eliminar el 6



Eliminar un nodo (cont.)

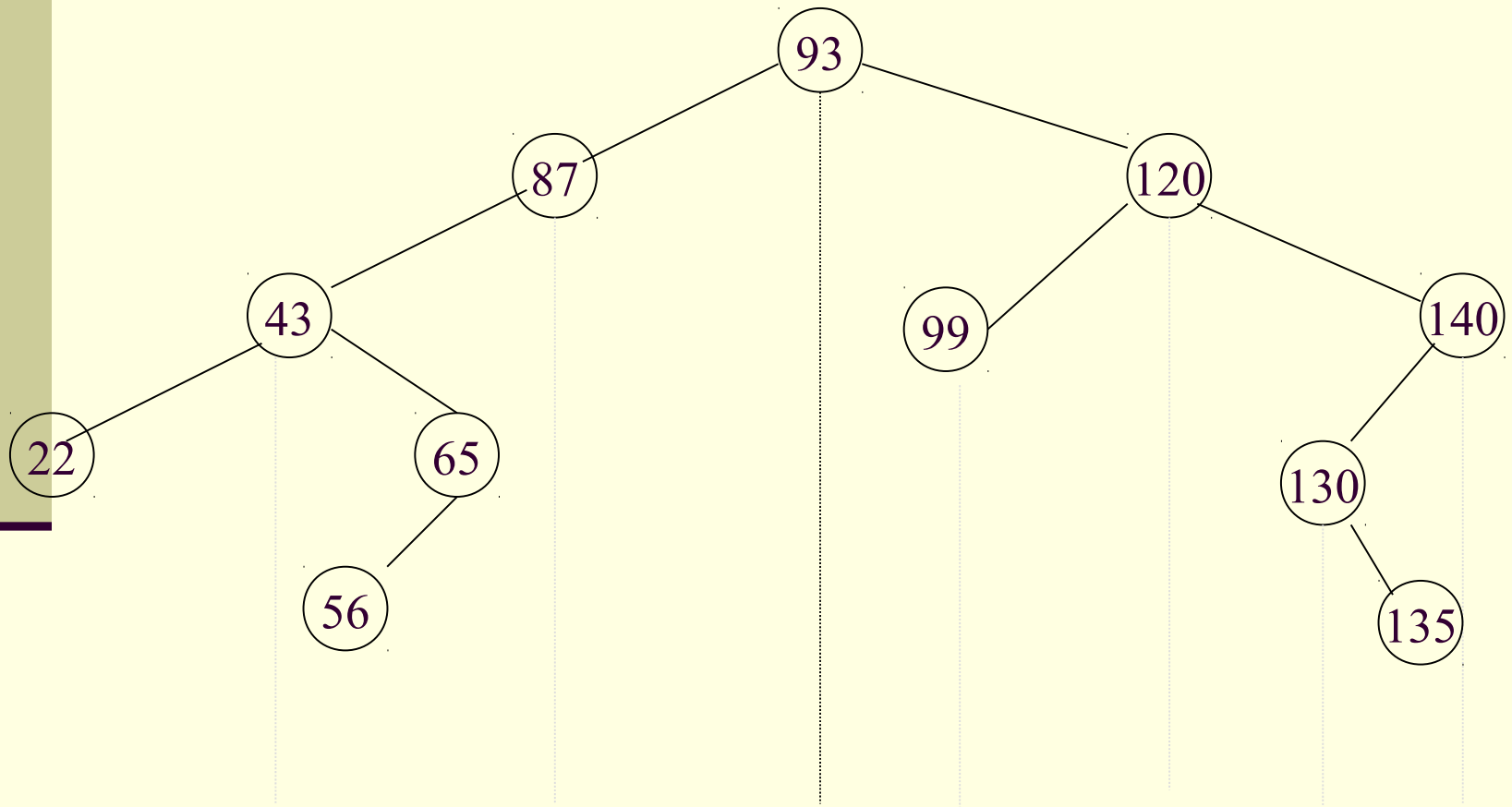
```
si NODO !=null entonces
si Dato < NODO.info
     Eliminación  (NODO.izq, Dato)
          si no
           si dato > NODO.INFO
            entonces Eliminación (NODO.der, Dato
                        si no
                        otro = NODO
                              si otro.der == null
                             entonces   NODO = otro.izq
                        si no
                            si otro.izq == null
                        entonces NODO = otro.der
```

Eliminar un nodo (cont.)

```
Si no
 {aux = otro.izq
   aux1 = Aux
while (aux.der != null )
aux1 = aux
aux   = aux.der
 }
      otro.info = aux.info
      otro = aux
     aux1.der = aux.izq
    quita (otro) (null)
si no Escribir ('el nodo no se encuentra en el árbol')
}
```

Eliminar un nodo (cont.)

- Elimina el 22,. 99, 87, 120, 140, 135, 56



Buscar nodo con información

```
Si dato < NODO.info
    Si NODO.IZQ == null
            Escribir “El nodo no se encuentra en el árbol”

    Si no
            Búsqueda (NODO.izq,dato)
Si no
          Si Dato > NODO.INFO entonces  Si NODO.DER = null
           Escribir “No se encuentra”
           Si no Búsqueda (NODO.der,dato)
    Si no Escribir “El NODO se encuentra en el árbol”
           si nodo != null
                    si Dato < NODO .INFO
                                Búsqueda (Nodo.izq,Dato)
           si no
                    si Dato > No dato.der
                                Búsqueda (No dato.der,dato)
                    si no
                                escribir “El Dato se encuentra
Si no Escribir “El dato no se encuentra en el árbol”
```

Contar nodos

```
//cuenta los nodos que hay en el árbol


public static int nodo (nodo raiz){
        if (raiz == null)   return 0;
        else
return (1+ Nodo( raiz.der) + Nodo( raiz.izq))
   }
```

Sumar los nodos

```
//suma  los nodos que hay en el árbol
public static int sumaNodo( nodo raiz){
        if(raiz == null)   return 0;
        else
Return (sumaNodo (raiz.der) + sumaNodo
   (raiz.izq) )
   }
```

Calcular profundidad del árbol

```
// calcula la profundidad de un árbol
public int profundidad (Nodo raiz) {
   if (raiz == null)  return 0;
lf (profundidad (raiz.der) > profundidad
   (raiz.izq))
   return profundidad (raiz.der) + 1;
else
   return profundidad( raiz.izq) + 1
}
```

Contar hojas.

```
// Cuenta  hojas de un árbol
public int contarHojas (Nodo raiz) {
    if (raiz = = null)  return 0;
If ((raiz.der == null) && (raiz.izq == null))
    Return  1;
else
    return contarHojas (raiz.izq) + contarHojas
    (raiz.der)
}
```

Ligas

- Inserta elimina y busca en un árbol binario
  - http://www.cs.jhu.edu/~goodrich/dsa/trees/btree.html
  - http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm
- Genera apartir de expresiones aritmeticas los árboles binarios
  - http://www.cs.jhu.edu/~goodrich/dsa/05trees/Demo1/