unidad 4

Estructuras no lineales

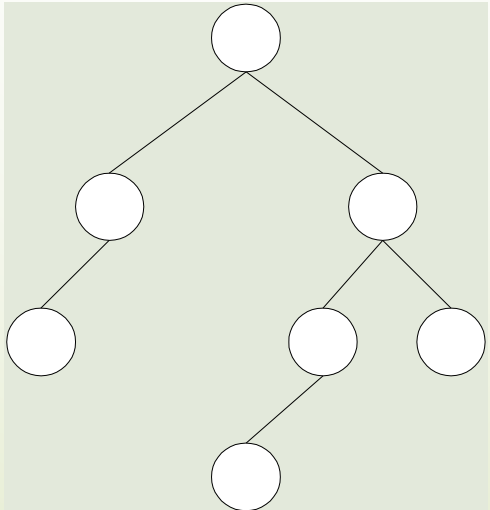
Árbol Binario

Árbol Binario de búsqueda (ABB)

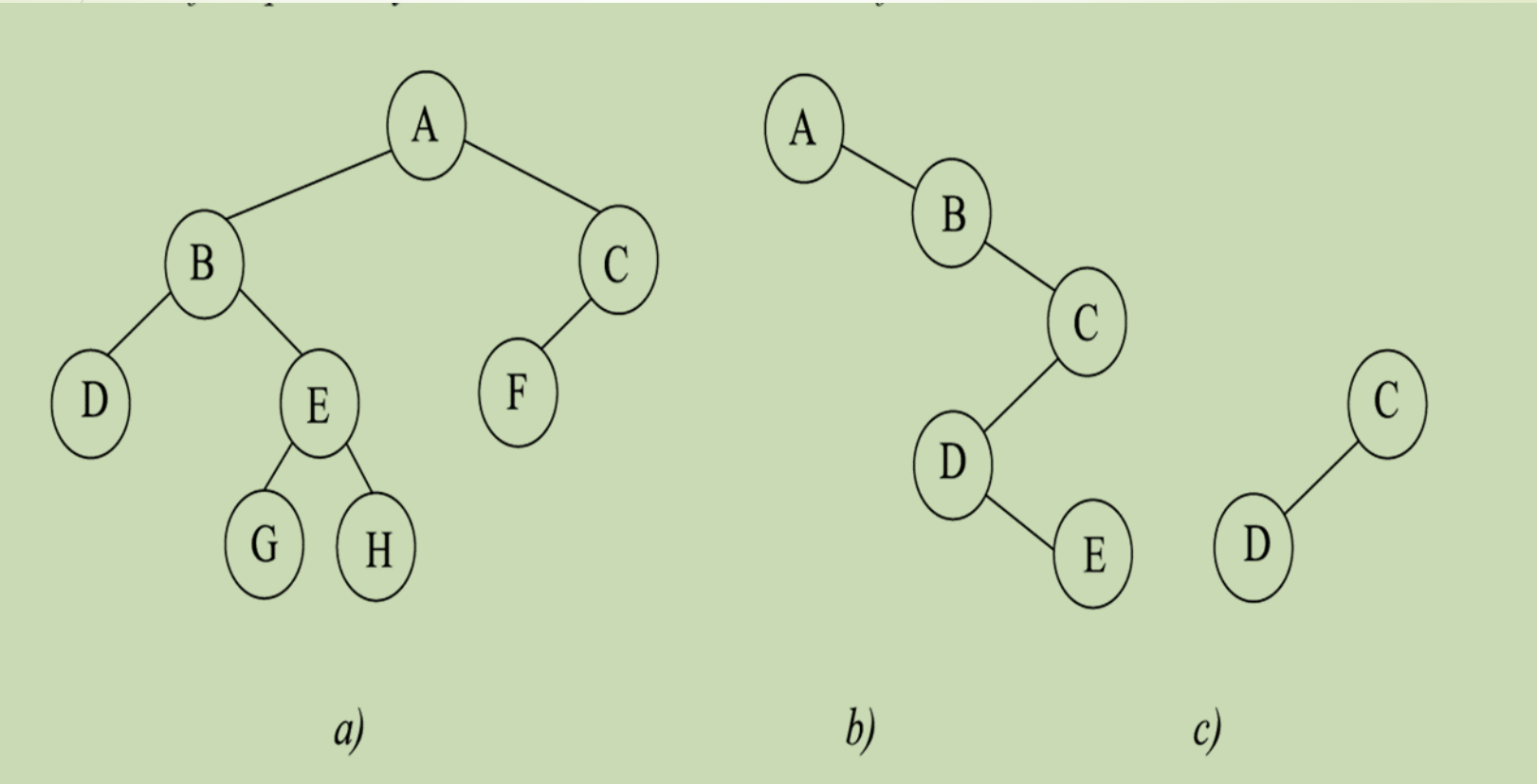
Arbol binario

Arbol binario

- Es un árbol en el que todos sus nodos a lo mas tienen dos hijos (hijo izquierdo o subárbol izquierdo, e hijo derecho o subárbol derecho.
- Los arboles binarios pueden usarse para representar una estructura en la cual es posible tomar decisiones con dos opciones.



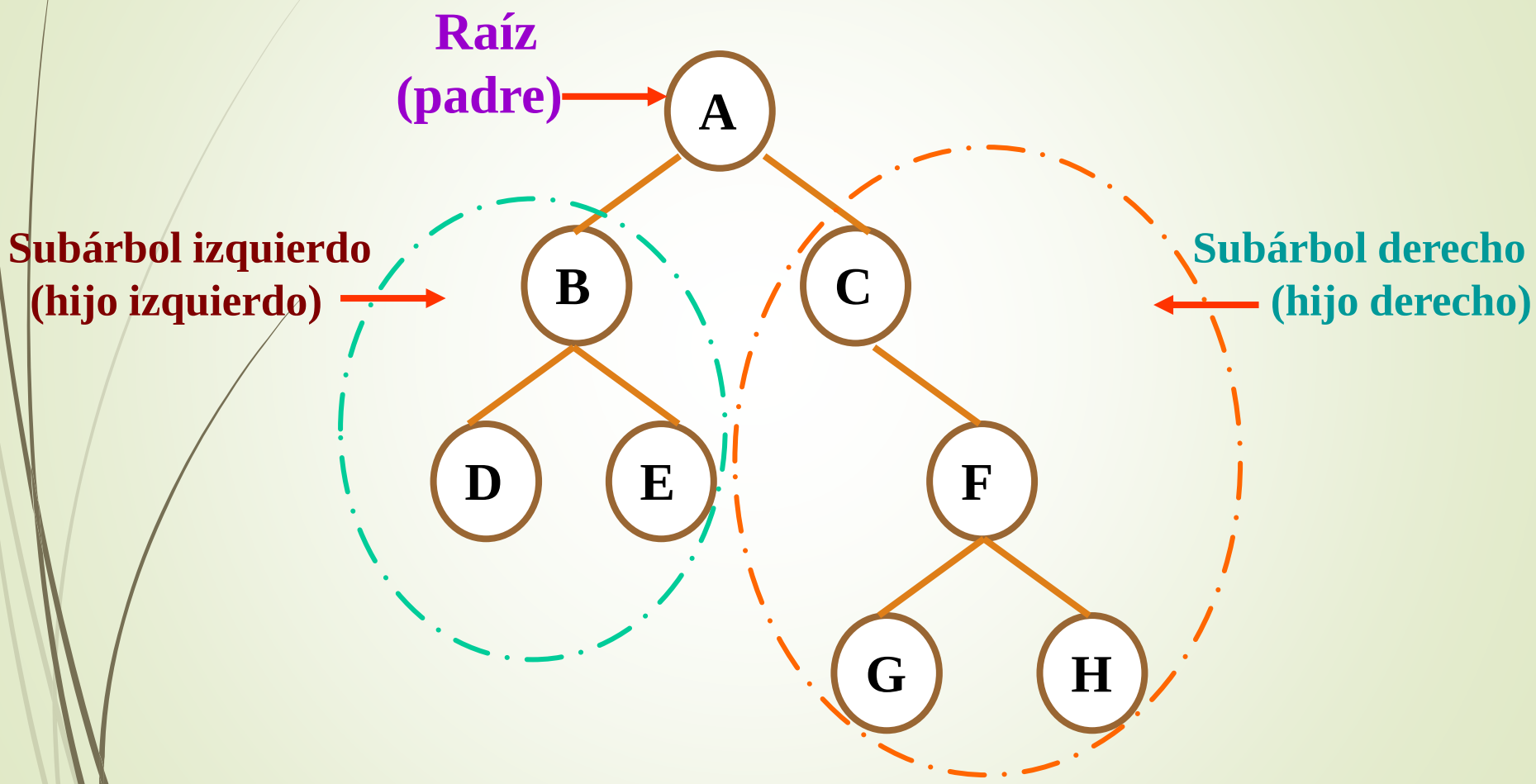
Arboles binarios



Árbol binario – definición recursiva

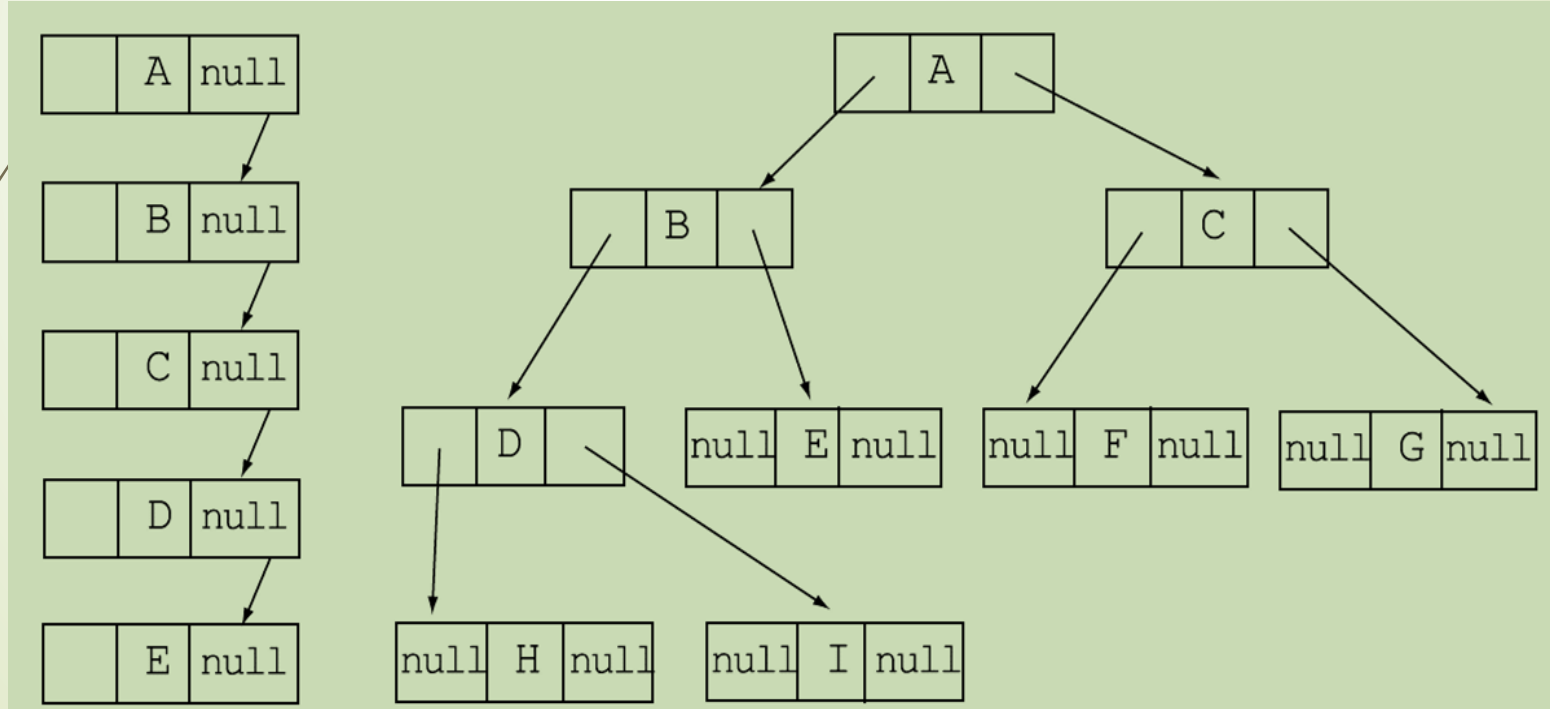
Es un conjunto finito de elementos que puede estar vacío o contener un elemento denominado la raíz o padre del árbol y otros elementos divididos en dos subconjuntos separados, cada uno de los cuales es en sí un árbol binario. Estos dos subconjuntos son denominados subárbol izquierdo o hijo izquierdo y subárbol derecho o hijo derecho.

Representación



ESTRUCTURA DE UN ÁRBOL BINARIO

- Un árbol binario se construye con nodos. Cada nodo debe contener el campo *dato* (datos a almacenar)
- y dos campos de enlace *(apuntador),* uno al subárbol izquierdo **(izquierdo, izdo)** y otro al
- subárbol derecho **(derecho, dcho)**. El valor null indica un árbol o un subárbol vacío



Representación enlazada de dos árboles binarios

operaciones básicas que definen el *TAD árbol binario*

- **Tipo de dato** Dato que se almacena en los nodos del árbol.
- **Operaciones**
- *CrearArbol* Inicia el árbol como vacío.
- *Construir* Crea un árbol con un elemento raíz y dos ramas, izquierda y derecha
- que son a su vez árboles.
- *EsVacio* Comprueba si el árbol no tiene nodos.
- *Raiz* Devuelve el nodo raíz.
- *Izquierdo* Obtiene la rama o subárbol izquierdo de un árbol dado.
- *Derecho* Obtiene la rama o subárbol derecho de un árbol dado.
- *Borrar* Elimina del árbol el nodo con un elemento determinado.
- *Pertenece* Determina si un elemento se encuentra en el árbol.

Operaciones en árboles binarios

- Algunas de las operaciones típicas que se realizan en árboles binarios son las siguientes:
- • Determinar su altura.
- • Determinar su número de elementos.
- • Hacer una copia.
- • Visualizar el árbol binario en pantalla o en impresora.
- • Determinar si dos árboles binarios son idénticos.
- • Borrar (eliminar el árbol).
- • Si es un árbol de expresión, evaluar la expresión.
- Todas estas operaciones se pueden realizar recorriendo el árbol binario de un modo sistemático.
- El recorrido es la operación de visita al árbol o, lo que es lo mismo, la visita a cada nodo del árbol una vez y sólo una.

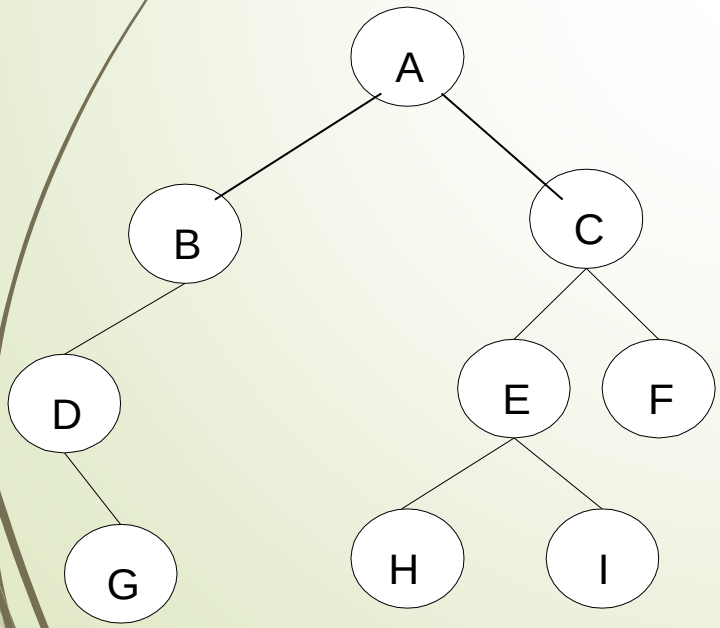
Recorridos de un árbol Binario

- **Recorrido en preorden (prefijo).**
- **Recorrido en inorden (infijo).**
- **Recorrido en postorden (postfijo).**

Recorridos de un árbol Binario

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

RID



Preorden = A B D G C E H I F

Recorridos de un árbol Binario

preorden (NODO)

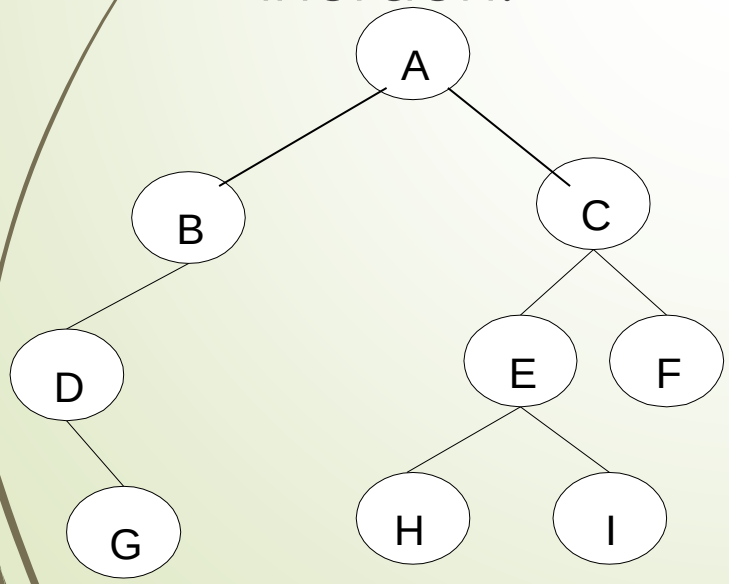
{*NODO es un apuntador a registro*}

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

Recorridos de un árbol Binario

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

IRD



Inorden: D G B A H E I C F

Recorridos de un árbol Binario

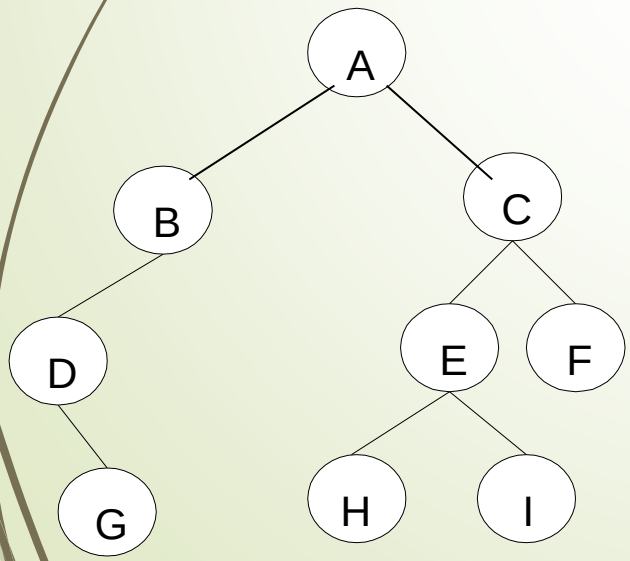
inorden (NODO)

{*NODO es un apuntador a registro*}

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

Recorridos de un árbol Binario

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



Postorden : G D B H I E F C A

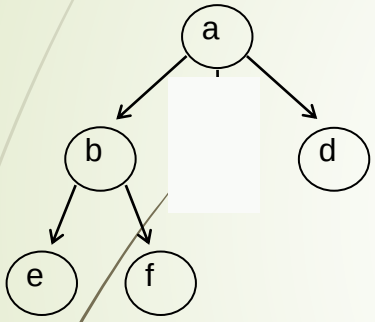
Recorridos de un árbol Binario

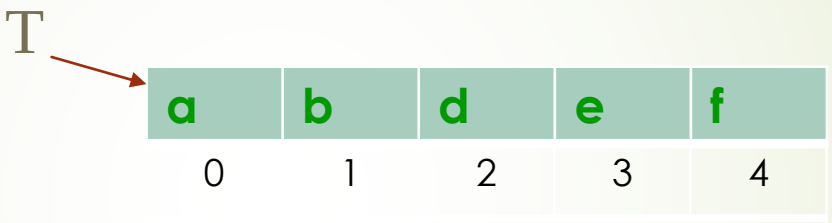
postorden (NODO)

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

Representación en memoria de árboles binarios

- 1. memoria estática: por medio de arreglos



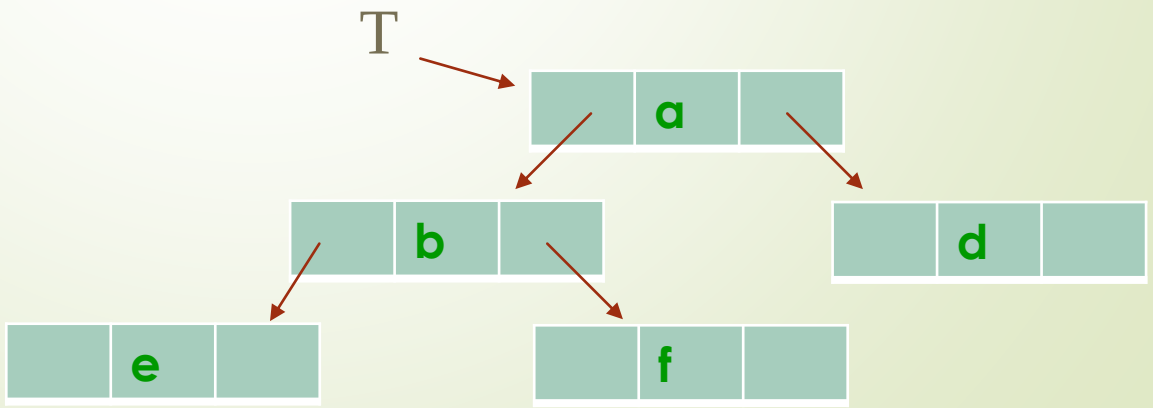


- 2. memoria dinámica: objetos llamados nodos

información del nodo

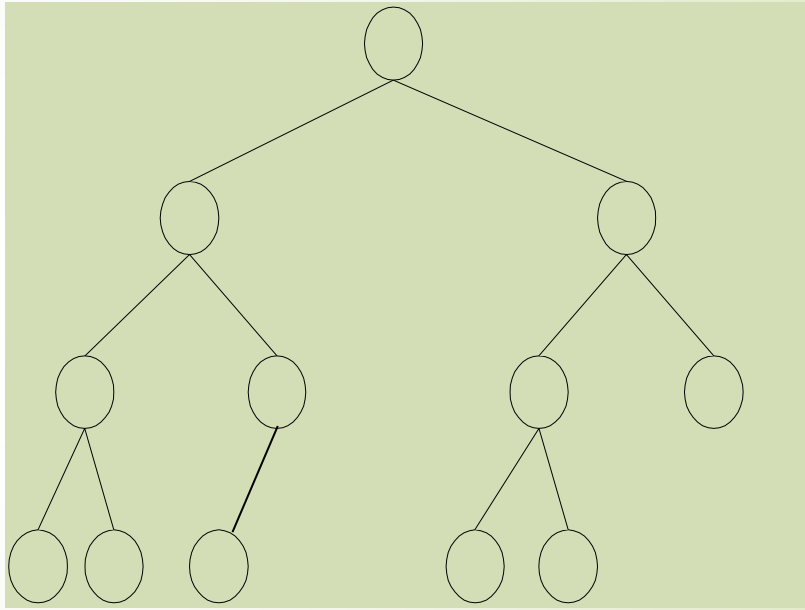
dirección del subárbol derecho

dirección del subárbol izquierdo



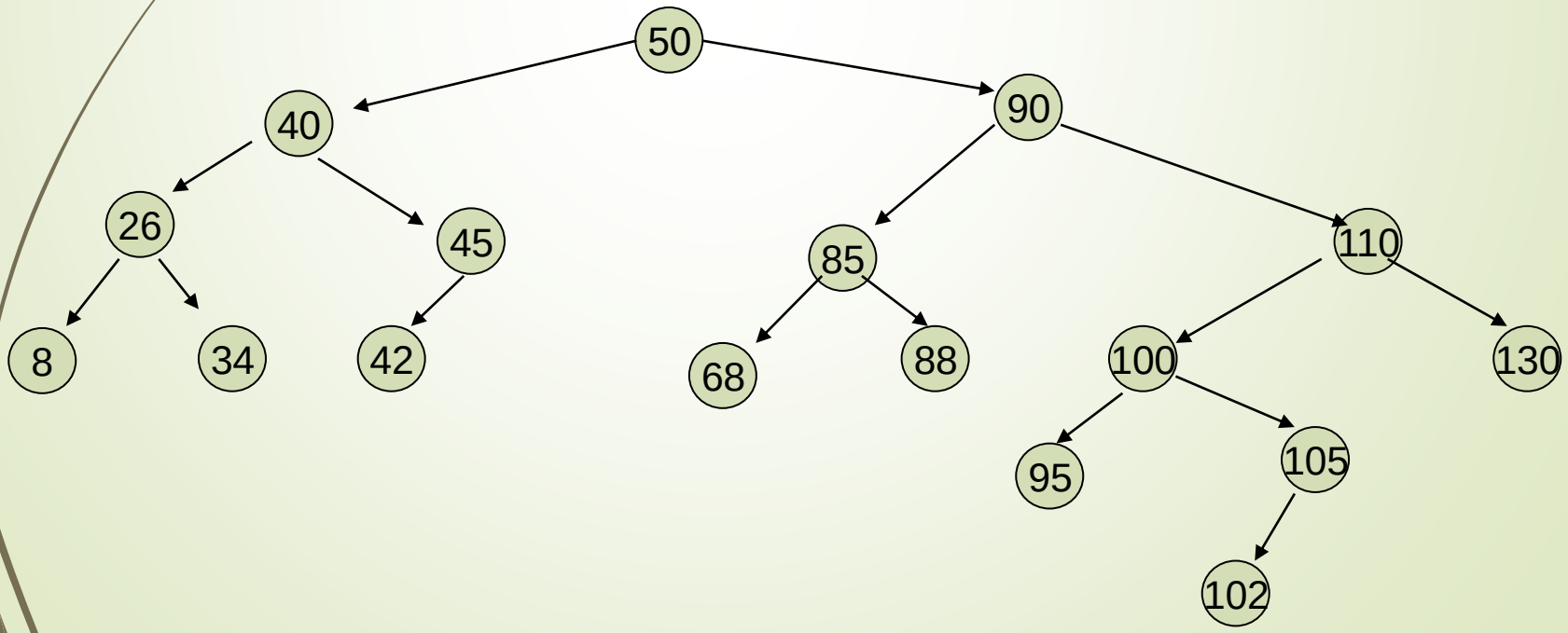
Árboles Binarios de Búsqueda

- 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.



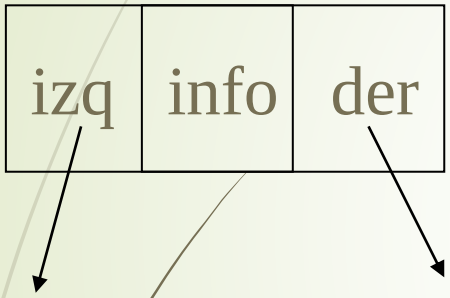
Características de un ABB

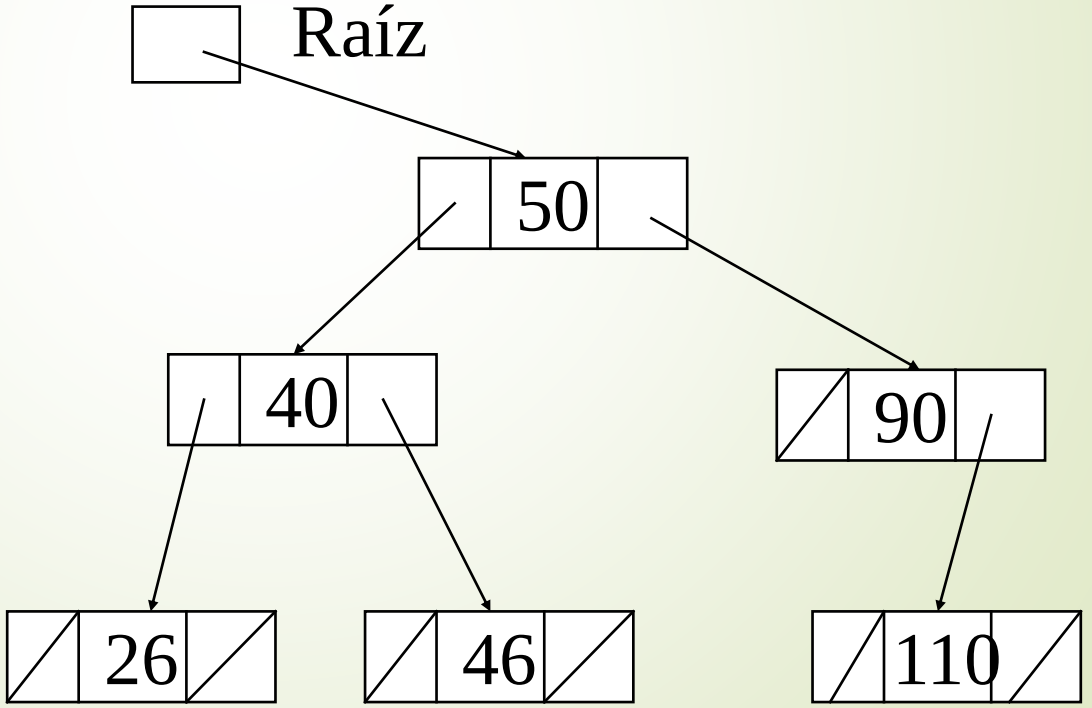
- Todos los nodos a la izquierda son menores al padre.
- Todos los nodos a la derecha son mayores al padre.
- Y solo pueden tener 2 hijos a lo mucho.



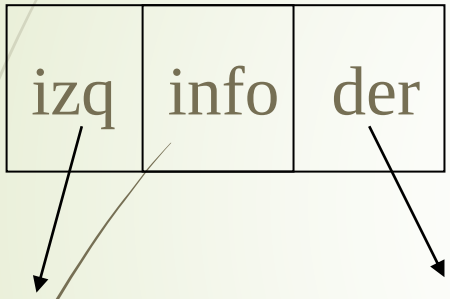
Representación de un árbol binario de búsqueda en la memoria.

- Cada nodo tiene la siguiente forma:





Clase nodo de un ABB




```
class NodoB{
    NodoB izq;
    NodoB der;
    int dato;
}
```

Operaciones sobre un árbol ABB

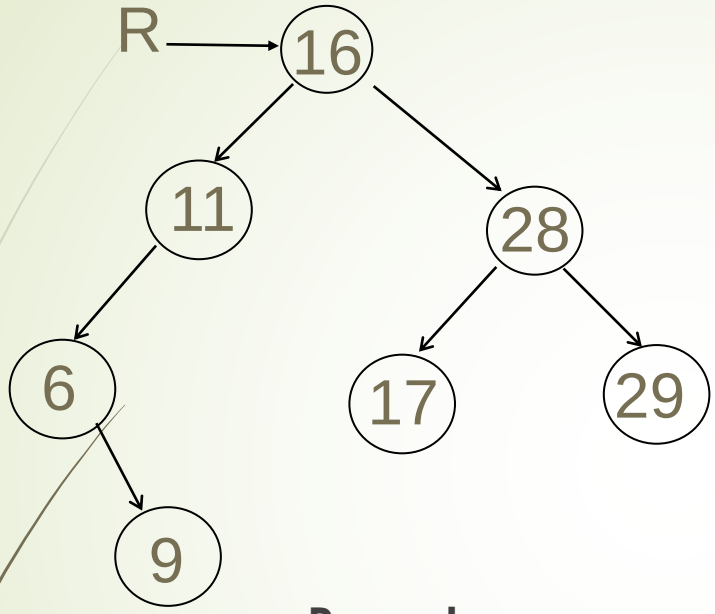
- Recorrer árbol
  - Preorden
  - Inorden
  - Postorden
- Inserción nodo
- Eliminar nodo
- Buscar nodo con información
- Sumar los nodos
- Calcular profundidad del árbol
- Contar nodos
- Contar hojas.

Recorridos de un Árbol de Búsqueda Binario (ABB)

Los recorridos de un ABB son semejantes a los del árbol binario.

- **Recorrido en preorden** — RID
  - Visita la raíz.
  - Recorre el subárbol izquierdo.
  - Recorre el subárbol derecho.
- **Recorrido en inorden (infijo)** — IRD
  - Recorre el subárbol izquierdo.
  - Visita la raíz
  - Recorre el subárbol derecho.
- **Recorrido en postorden (postfijo)** — IDR
  - Recorre el subárbol izquierdo.
  - Recorre el subárbol derecho.
  - Visita la raíz.

Recorridos en un ABB
Ejercicio



- Preorden:

  RID

16	11	6	9	28	17	29

- Inorden:

  IRD

6	9	11	16	17	28	29

- Postorden:

  IDR

9	6	11	17	29	28	16

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.

Inserción en un ABB

1. Comparar 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**.
2. 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.)

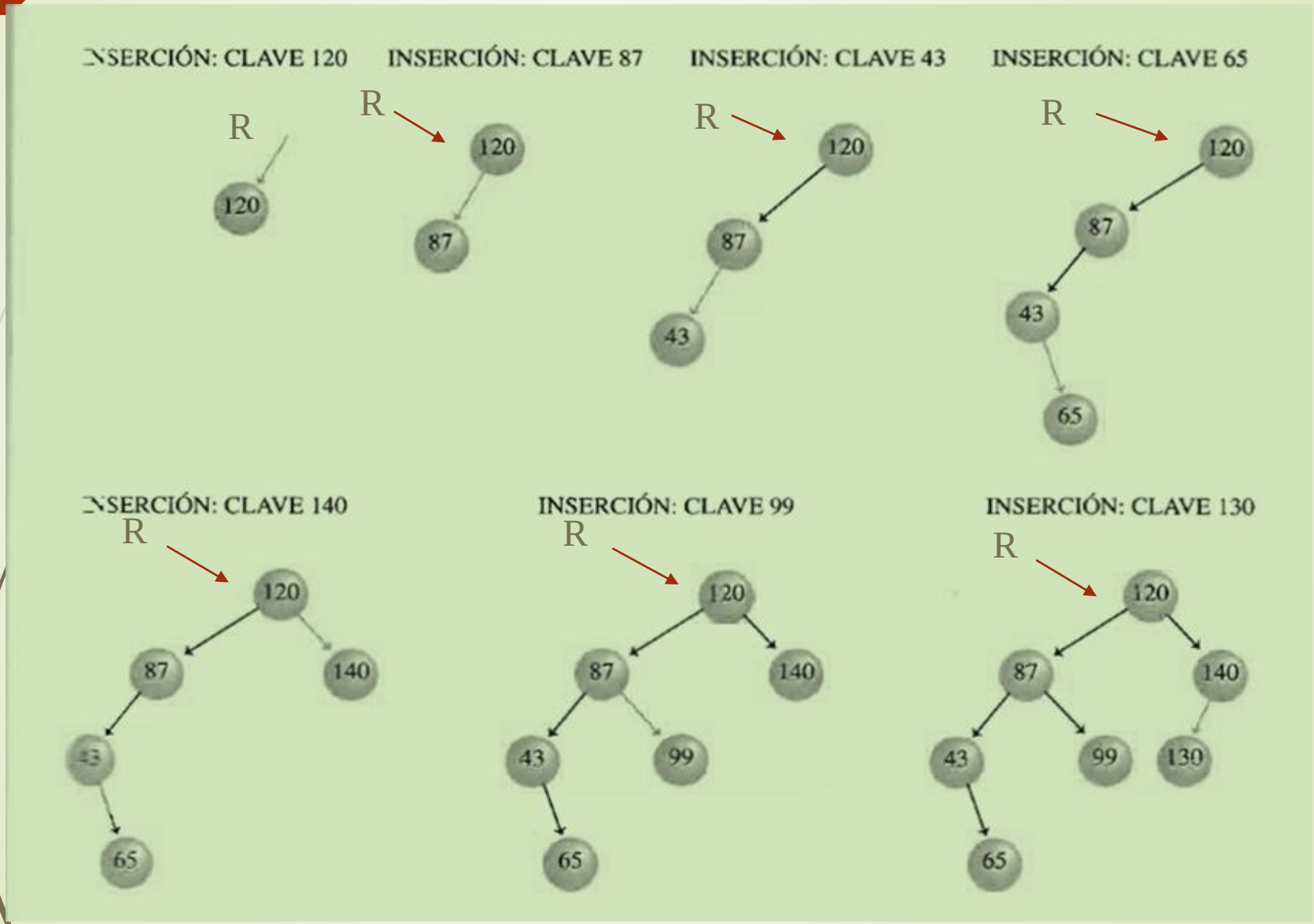

```
Algoritmo INSERCION(NODO, INFOR)
{Si  NODO ≠ Null{
      Si  (INFOR < NODO.INFO)
            INSERCION ( NODO.IZQ , INFOR)
       sino
            si ( INFOR >  NODO.INFO)
                        INSERCION( NODO.DER,  INFOR )
            sino
                Escribir "El nodo ya se encuentra en el árbol"

Else {
      CREA (OTRO)//Crear un nuevo nodo
      Hacer OTRO.IZQ = null,
      OTRO.DER = null,
      OTRO.INFO  = INFOR  y NODO = OTRO }

}
```

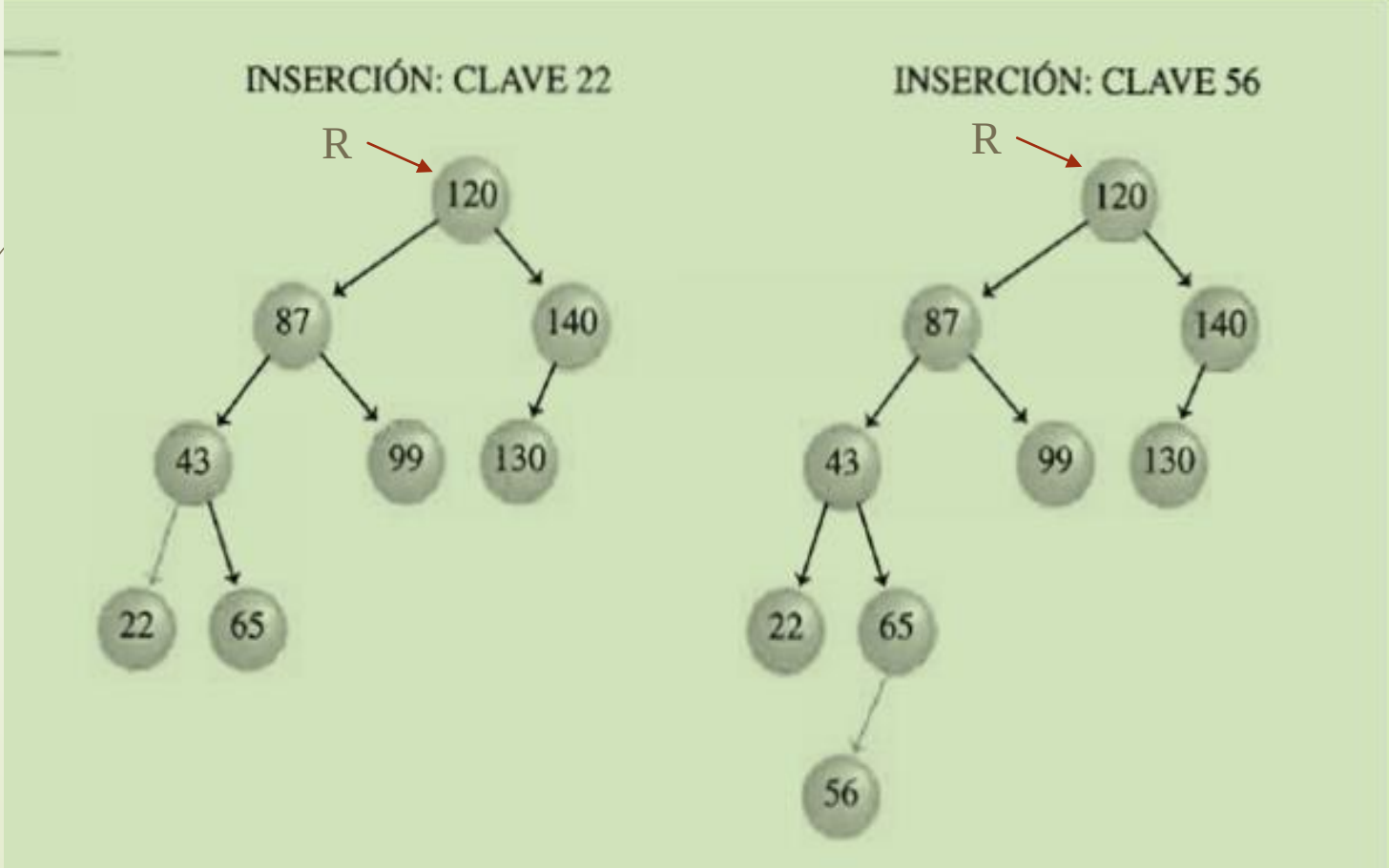
Inserción en un ABB (cont.)

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

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

Solución: se dibuja un árbol por cada elemento insertado.




INSERCIÓN: CLAVE 22
R
120
87
140
43
99
130
22
65
INSERCIÓN: CLAVE 56
R
120
87
140
43
99
130
22
65
56

Referencias

- Estructuras de datos, Osvaldo Cairo, Silvia guardati. Ed Mc Graw-Hill
- Estructura de datos en C++, Dr. Romeo Sánchez Nigenda.
- https://www.centroestudioscervantinos.es/fundamentos-y-aplicaciones-de-la-teoria-de-los-grafos/