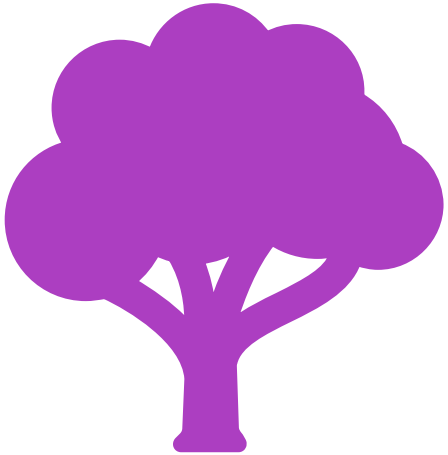
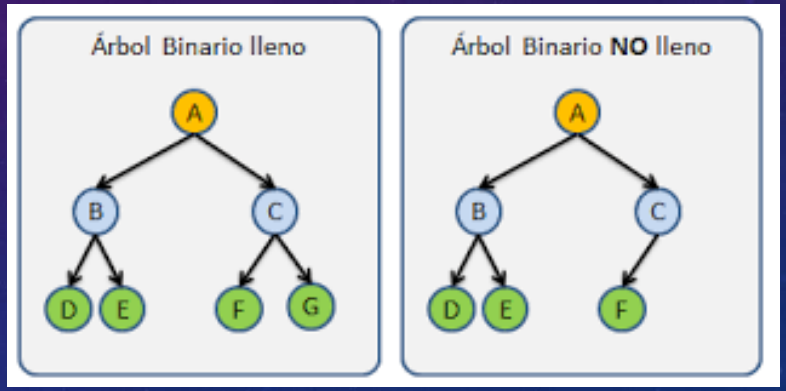
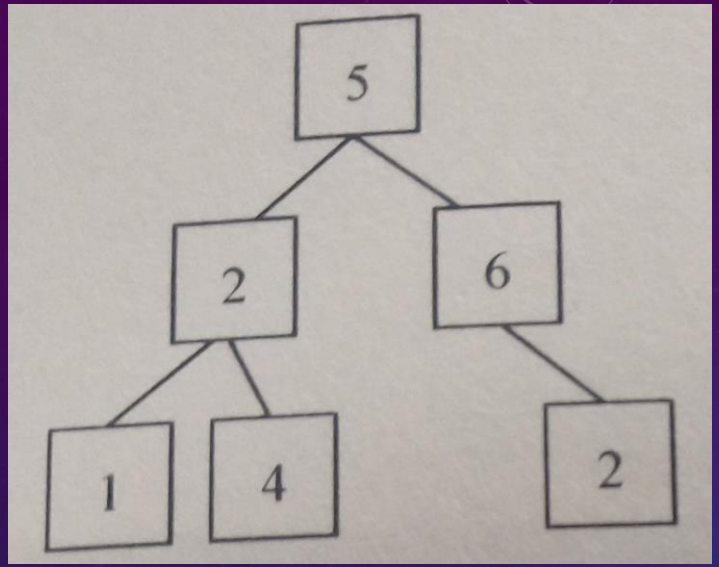
ÁRBOLES BINARIOS

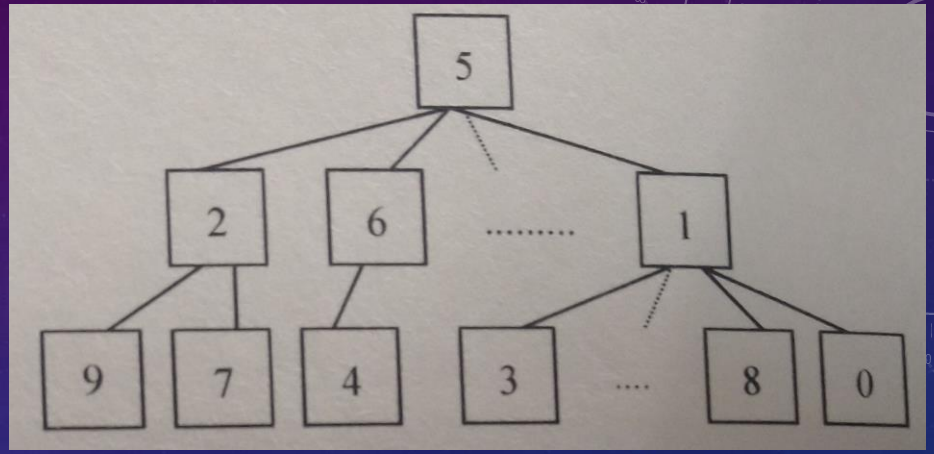


ÁRBOL

- Llamamos árbol a una estructura de datos no lineal, en la que se observa que desde cada nodo descienden uno o varios nodos de su mismo tipo salvo los últimos, que no tienen descendencia.







Árbol binario

Árbol n-ario

Cada nodo puede tener solo hasta 2 hijos.

Cada nodo puede tener una cantidad **n** de hijos, siendo **n** un valor que puede variar entre un nodo padre y otro.

- **Árbol binario:** de cada nodo (padre) descienden hasta 2 nodos (hijos).

- **Árbol n-ario:** la cantidad de nodos (hijos) que descienden de cada nodo (padre) es variable y está representada por los puntos suspensivos.

- De cualquier manera, siempre tendremos un nodo inicial al que llamaremos “raíz” desde el cual descenderán una cantidad finita, fija o variable, de nodos de su mismo tipo, a los que llamaremos “hijos”.

- A su vez, cada nodo hijo puede ser raíz o “padre” de otros hijos y así sucesivamente hasta llegar a las “hojas” del árbol, que no son otra cosa que nodos sin descendencia.

Los árboles pueden clasificarse en función de la cantidad de hijos que desciendan de cada padre.

Según la cantidad de hijos que cada nodo padre pueda llegar a tener diremos que el árbol es binario (2 hijos), ternario (3 hijos), cuaternario (4 hijos) o n-ario, siendo **n** un valor mayor o igual a 1 y, tal vez, diferente para cada nodo padre.

IMPLEMENTACIÓN DE LA ESTRUCTURA DE DATOS

```
typedef  struct Nodo
{
      int v;
      struct Nodo * izq;
      struct Nodo * der;
}Nodo;
```

Nodo de un árbol binario

```
typedef  struct NodoN
{
      int v;
      struct NodoN * hijos[];
}NodoN;
```

Nodo de un árbol n-ario

- El nodo del árbol binario tiene dos punteros (izq y der) para referenciar a sus, a lo sumo, dos hijos.
- El nodo del árbol n-ario tiene un array de punteros que nos permitirá hacer referencia a una cantidad de hijos diferente para cada padre o raíz.

ÁRBOLES BINARIOS

- Son árboles cuyos nodos contienen dos ligas (ninguna, una, o ambas de las cuales pueden ser NULL).
- El nodo raíz es el primer nodo del árbol. Cada liga del nodo raíz hace referencia a un hijo.
- El hijo izquierdo es el primer nodo del subárbol izquierdo, y el hijo derecho es el primer nodo del subárbol derecho.

- A los hijos de un nodo se les conoce como hermanos.
- A un nodo sin hijos se le conoce como nodo hoja.
- Los científicos en computación generalmente dibujan árboles del nodo raíz hacia abajo; exactamente de manera contraria a los árboles naturales.
- Dado un árbol binario, si consideramos que la raíz constituye el primer nivel del árbol (nivel 0) entonces sus nodos hijos constituirán el segundo nivel (nivel 1), sus “nietos” el tercero (nivel 2) y así sucesivamente.

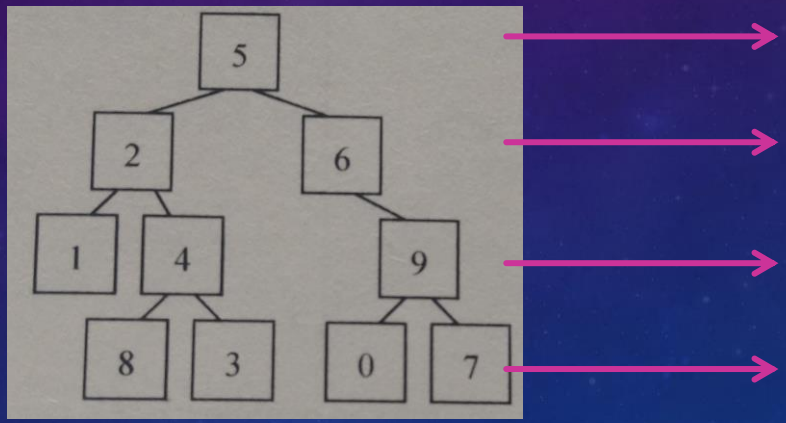
RECORRIDOS EN UN ÁRBOL BINARIO

- Recorrer un árbol implica desplazarnos a través de todos sus nodos respetando un determinado orden o criterio. Según cuál sea el orden o criterio que utilicemos para emprender la recorrida, tendremos que hablar de:

  - Recorridos en amplitud (también conocido como recorrido por niveles).
  - Recorridos en profundidad.

RECORRIDO EN AMPLITUD

- El recorrido “por niveles” consiste en listar los nodos del árbol desde arriba hacia abajo y de izquierda a derecha.



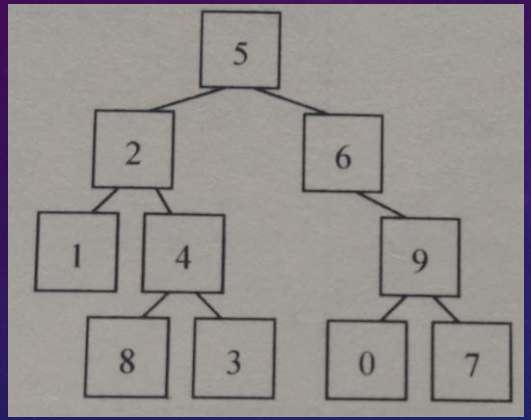
- El recorrido “por niveles” sobre este árbol binario es el siguiente:

5, 2, 6, 1, 4, 9, 8, 3, 0, 7

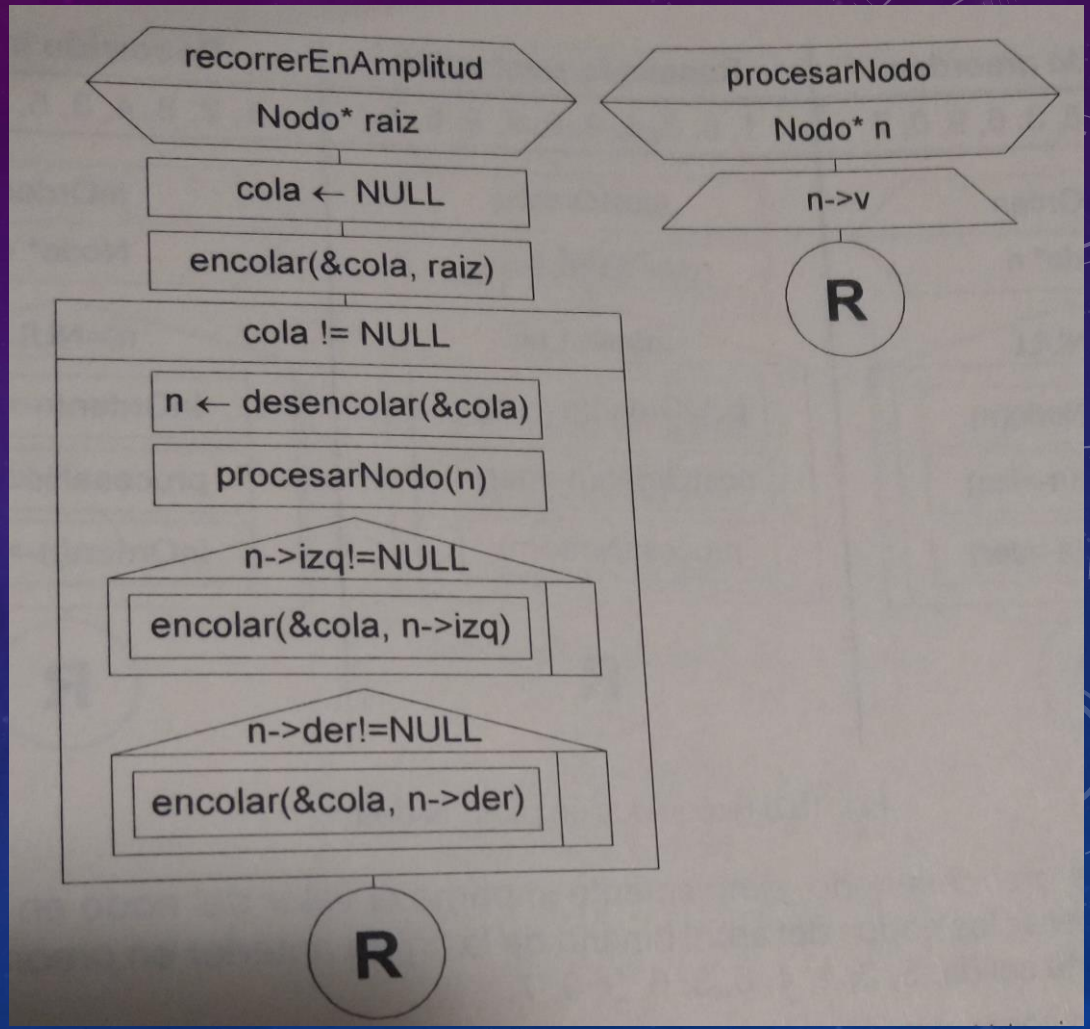
- Para implementar un algoritmo que nos permita emprender este recorrido tenemos que utilizar una cola:

- Encolamos la raíz del árbol.
- Luego, mientras la cola tenga elementos.
  - Tomamos un elemento (nodo de la cola).
  - Procesamos el elemento (por ejemplo, mostramos su valor en pantalla).
  - Encolamos a sus hijos, primero el hijo izquierdo y después el hijo derecho.

Recorrido en amplitud
o “por niveles”



5, 2, 6, 1, 4, 9, 8, 3, 0, 7



RECORRIDOS EN PROFUNDIDAD

PREORDEN, POSTORDEN, INORDEN

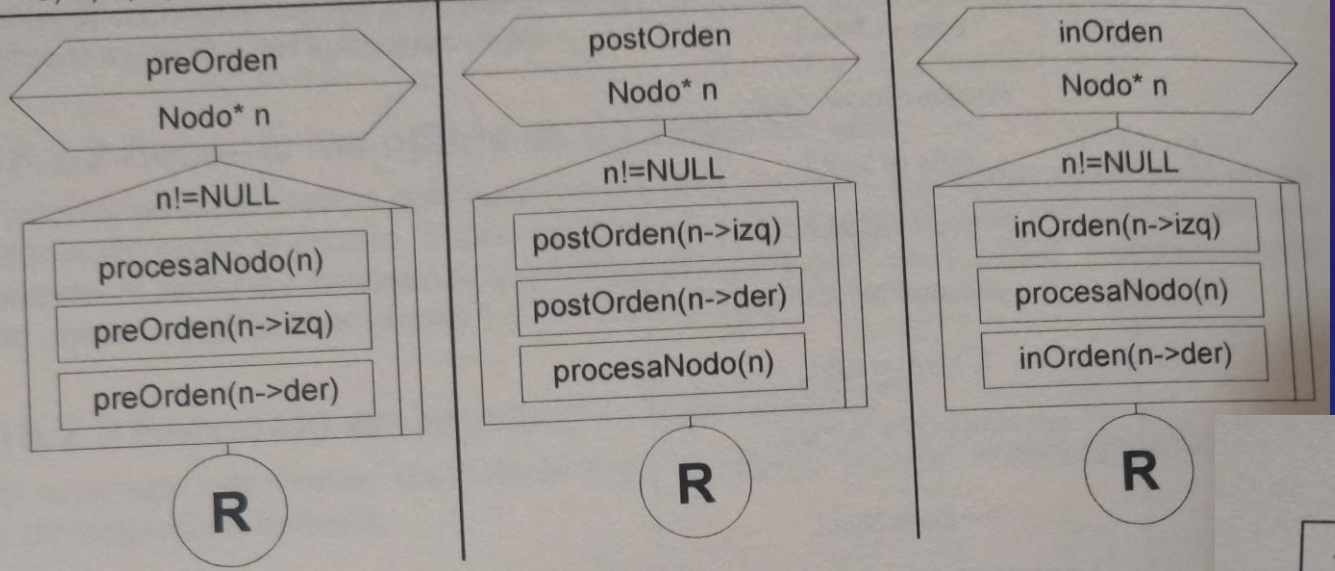
- Cada nodo de un árbol binario es en sí mismo la raíz, de un subárbol binario.

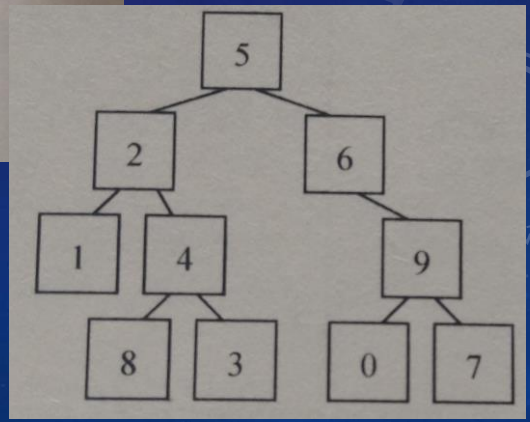
- Luego, dependiendo del orden en que se procese el nodo raíz y cada uno de sus hijos se definen los recorridos pre, post e inorden.

- Preorden consiste en procesar primero la raíz y luego su hijo izquierdo y su hijo derecho.
- Postorden se refiere a que el proceso de la raíz será posterior al proceso de sus hijos.
- Inorden u orden simétrico significa que primero se procesará al hijo izquierdo, luego la raíz y finalmente al hijo derecho.
- Como cada hijo es en sí mismo la raíz de un subárbol estos recorridos, generalmente, se implementan como funciones recursivas.

Recorridos en profundidad preorden, postorden, inorden

Recorrido preorden	Recorrido postorden	Recorrido inorden
5, 2, 1, 4, 8, 3, 6, 9, 0, 7	1, 8, 3, 4, 2, 0, 7, 9, 6, 5	1, 2, 8, 4, 3, 5, 6, 0, 9, 7





Supongamos que la función **procesaNodo** simplemente imprime el valor del nodo en la consola.

ÁRBOL BINARIO DE BÚSQUEDA

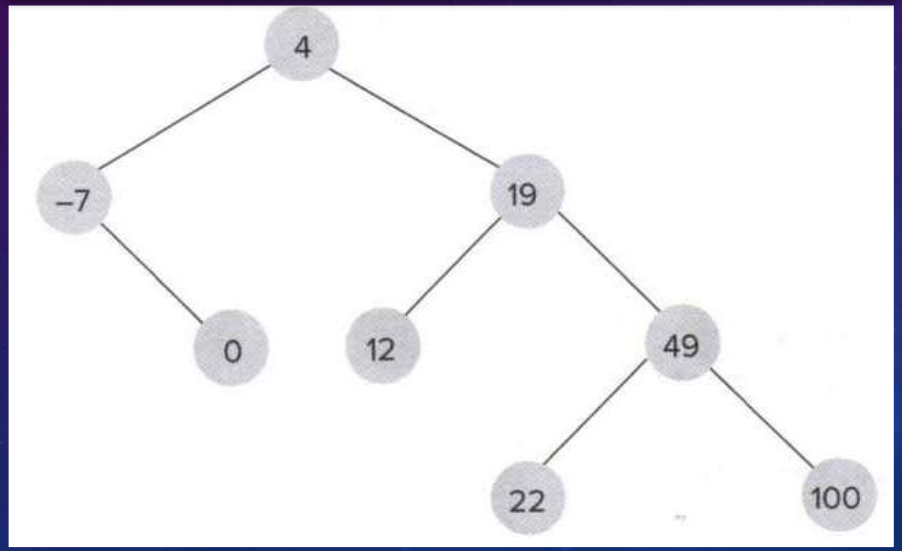
ÁRBOL BINARIO DE BÚSQUEDA

- Un árbol binario de búsqueda (sin valores duplicados de nodos) tiene la característica de que los valores de cualquier subárbol izquierdo son menores que el valor de su nodo padre, y que los valores de cualquier subárbol derecho son mayores que el valor de su *nodo padre.*

ÁRBOL BINARIO DE BÚSQUEDA

- Construir el árbol binario de búsqueda correspondiente a la lista de números.

4	19	-7	49	100	0	22	12

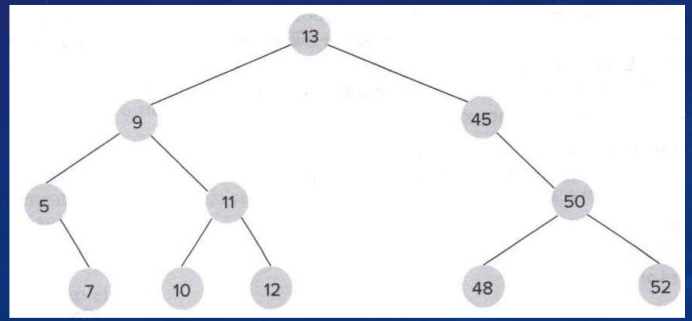


- El primer valor, es la raíz del árbol: 4.
- El siguiente valor, 19, se compara con 4; como es más grande se lleva al subárbol derecho de 4.
- El siguiente valor: -7 se compara con la raíz y es menor que su valor 4, por tanto, se mueve al subárbol izquierdo.

En la figura de la izquierda se muestran los pasos sucesivos para crear el árbol.

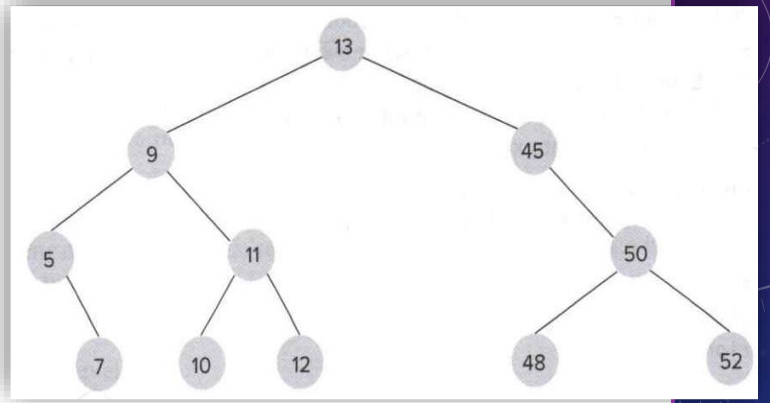
BÚSQUEDA DE UN ELEMENTO EN UN ÁRBOL BINARIO

- La búsqueda en un árbol binario es dicotómica, ya que a cada visita de un nodo se elimina aquel de los subárboles que no contienen el valor buscado (valores todos inferiores o todos superiores).
- El algoritmo de búsqueda del elemento –clave x- se realiza con la clave del raíz del árbol.
- Si no es el mismo, se pasa al subárbol izquierdo o derecho, según el resultado de la comparación, y se repite la búsqueda en ese subárbol.
- La terminación del procedimiento se producirá cuando:
  - Se encuentra la clave.
  - No se encuentra la clave, se continúa hasta encontrar un subárbol vacío.



Búsqueda de un elemento en un árbol binario

```
procedimiento buscar (E punt:  RAIZ; E <tipo_elemento>: elemento; S punt:  actual, anterior)
     var
          lógico: encontrado
     inicio
      encontrado ← falso
      anterior ← nulo
      actual ← raiz
      mientras no encontrado Y (actual <> nulo) hacer
          si actual → .elemento = elemento entonces
                encontrado ← verdad
                si_no
                anterior ← actual
                si actual → .elemento > elemento entonces
                      actual  ← actual → .izq
                si_no
                      actual  ← actual → .der
                fin_si
          fin_si
      fin_mientras
      si no encontrado entonces
          escribir ('no existe', elemento)
      si_no
          escribir (elemento, 'existe')
      fin_si
fin_procedimiento
```

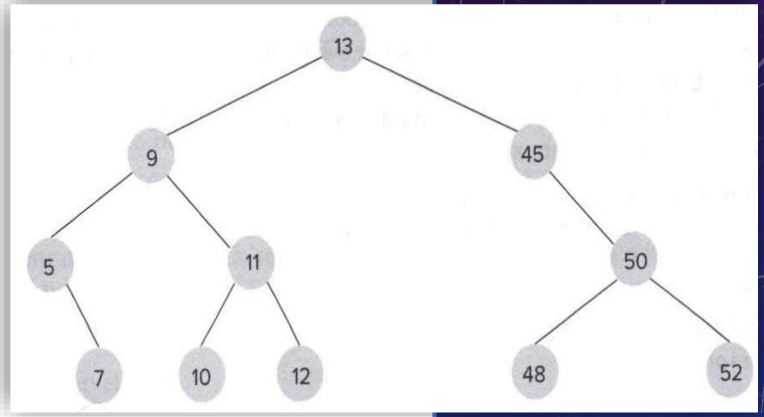


INSERTAR UN ELEMENTO EN UN ÁRBOL BINARIO

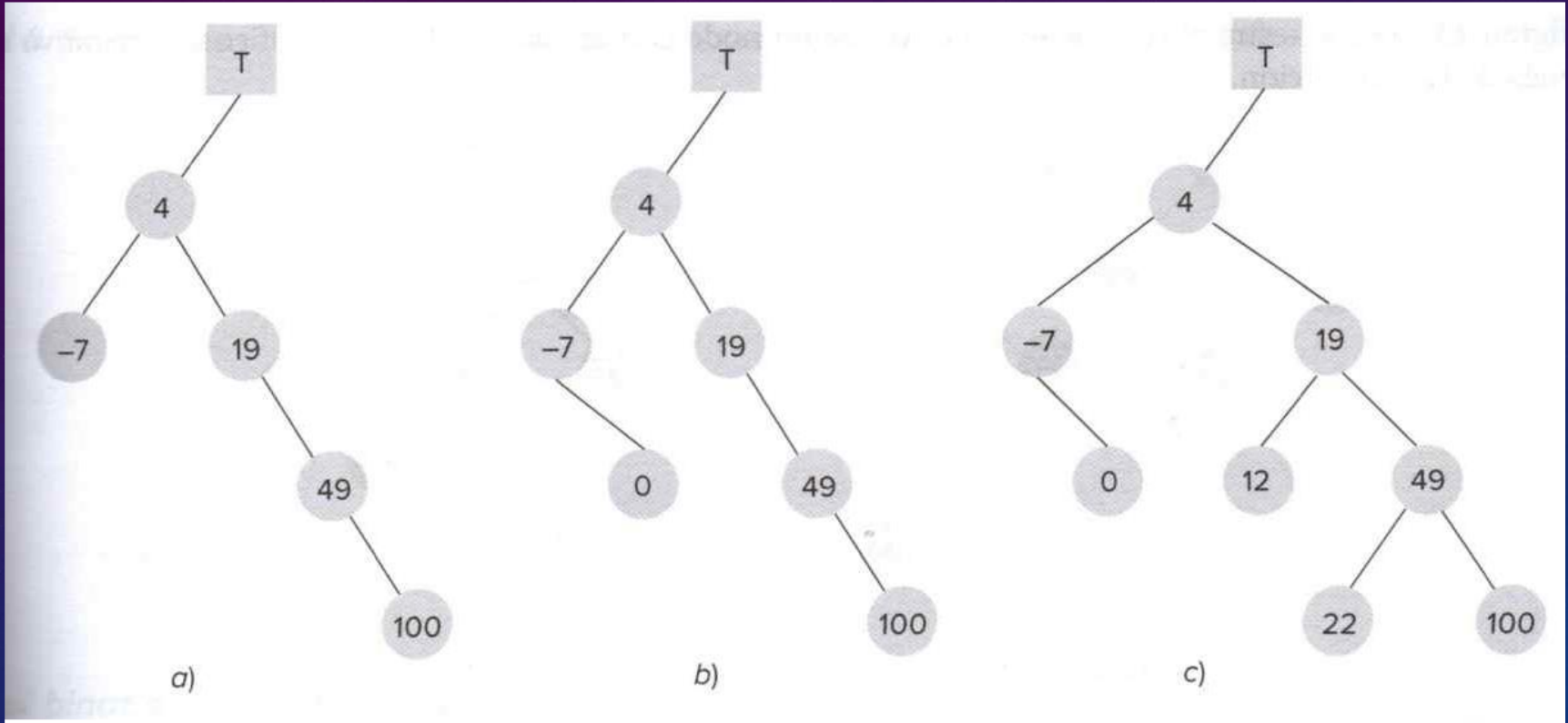
- Para insertar un elemento en el árbol A se ha de comprobar, en primer lugar, que el elemento no se encuentra en el árbol, ya que en su caso no precisa ser insertado.
- Si el elemento no existe, la inserción se realiza en un nodo en el que al menos uno de los dos punteros ***izq*** o ***der*** tenga valor ***nulo***.
- Para realizar la condición anterior se desciende en el árbol a partir del nodo raíz, dirigiéndose de izquierda a derecha de un nodo, según que el valor a insertar sea inferior o superior al valor del campo clave INFO de este nodo.
- Cuando se alcanza un nodo del árbol en que no se puede continuar, el nuevo elemento se engancha a la izquierda o derecha de este nodo en función de que su valor sea inferior o superior al del nodo alcanzado.

Insertar un elemento en un arbol binario

```
procedimiento insertar (E/S punt: raiz; E <tipo_elemento>: elemento)
     var
     punt: nuevo,
          actual,
          anterior
     inicio
     buscar (raiz, elemento, actual, anterior)
     si actual <> NULO entonces
          escribir ('elemento duplicado')
     si_no
          reservar (nuevo)
          nuevo → .elemento ← elemento
          nuevo → .izq ← nulo
          nuevo → .der ← nulo
          si anterior = nulo entonces
               raiz ← nuevo
          si_no
               si anterior → .elemento > elemento entonces
                    anterior → .izq ← nuevo
               si_no
                    anterior → .der ← nuevo
               fin_si
          fin_si
     fin_si
fin_procedimiento   // <tipo_elemento> es un tipo simple
```



INSERTAR UN ELEMENTO EN UN ÁRBOL BINARIO



Inserciones en un árbol de búsqueda binaria: *a*) insertar 100, *b*) insertar (0), *c*) insertar 22 y 12.

ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

- La eliminación de un elemento debe conservar el orden de los elementos del árbol.

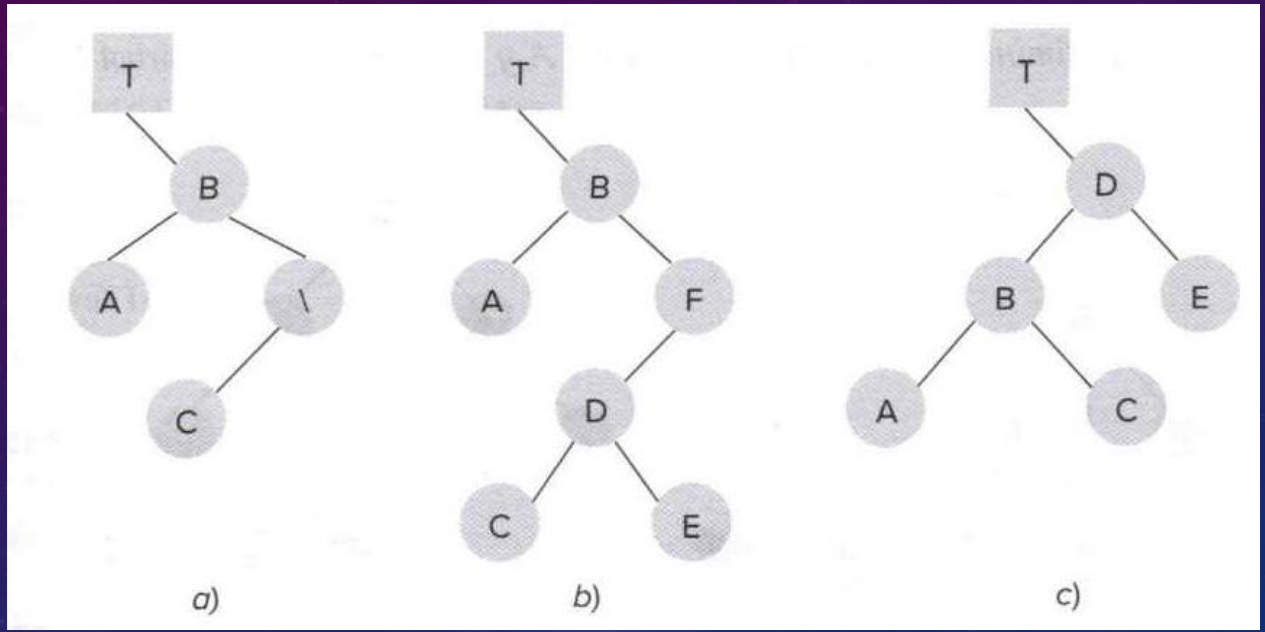
- Se consideran diferentes casos, según la posición del elemento o nodo en el árbol.
  - Si el elemento es una hoja, simplemente se suprime.
  - Si el elemento no tiene más que un descendiente, se sustituye entonces por ese descendiente.
  - Si el elemento tiene dos descendientes, se sustituye por el elemento inmediato inferior situado lo más a la derecha posible de su subárbol izquierdo.

ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

- Para poder realizar estas acciones será preciso conocer la siguiente información del nodo a eliminar.

  - Conocer su posición en el árbol.
  - Conocer la dirección de su padre.
  - Conocer si el nodo a eliminar tiene hijos, si son 1 o 2 hijos, y en el caso de que solo sea uno, si es hijo derecho o izquierdo.

ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

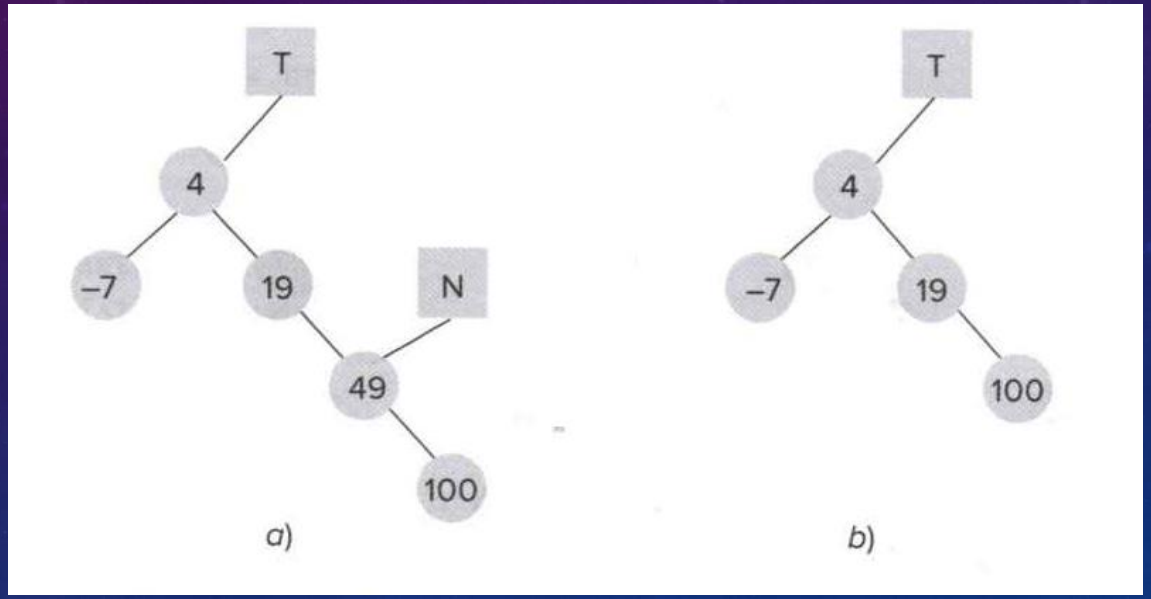


A continuación se muestran los tres posibles casos de eliminación de un nodo.

a) eliminar C
b) eliminar F
c) eliminar B

ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

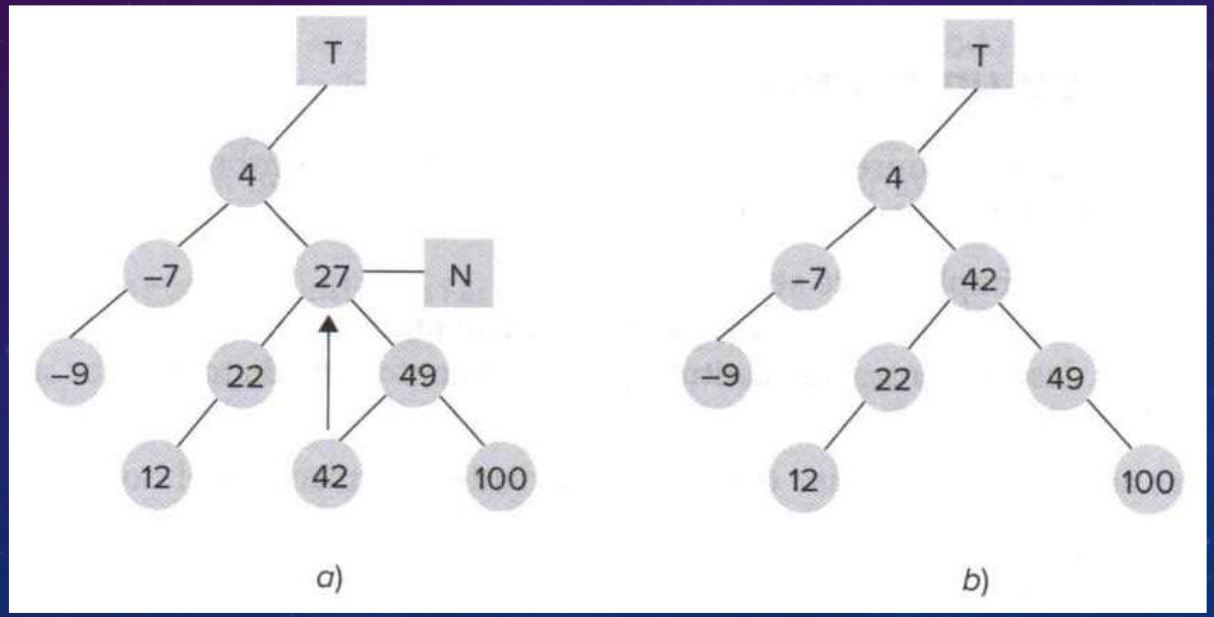
- Caso de eliminación de un nodo con un subárbol en un gráfico comparativo antes y después de la eliminación.



Eliminación de un nodo con un subárbol

ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

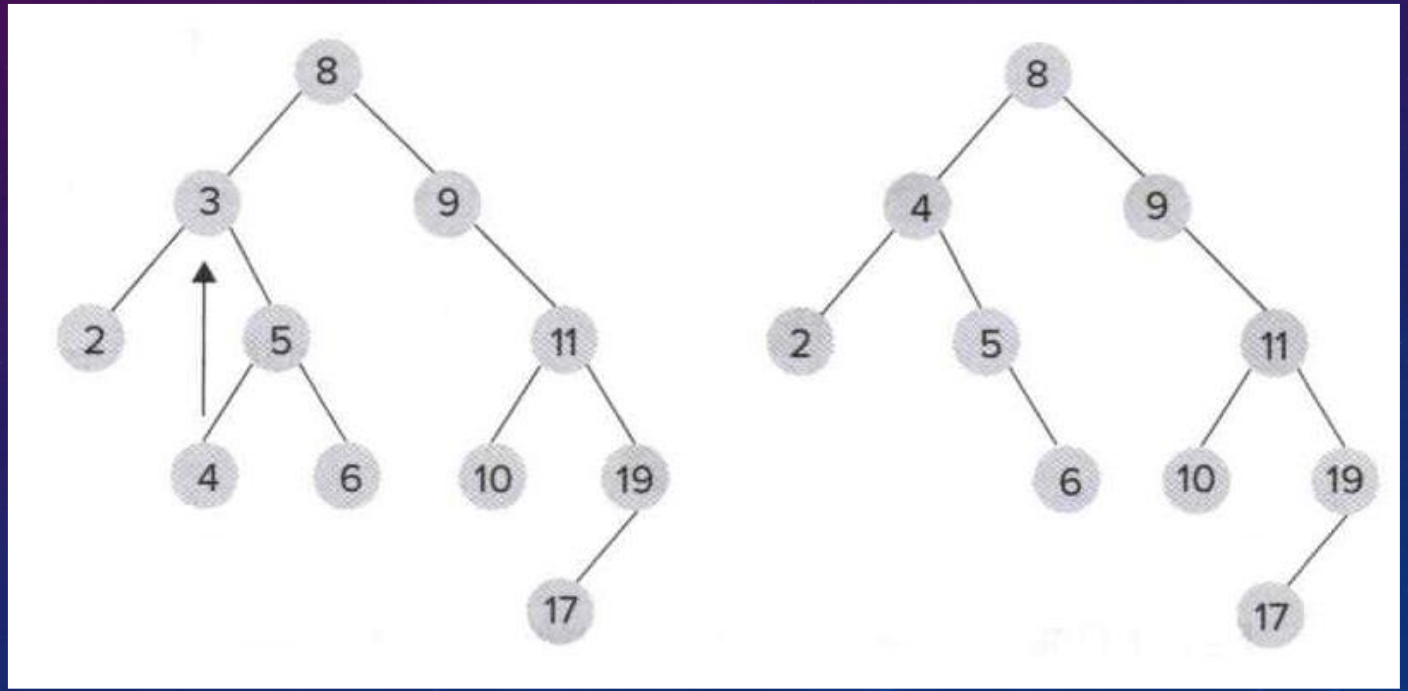
- Caso de eliminación de un nodo (27) que tiene dos subárboles no nulos. En este caso se busca el nodo sucesor cuyo campo de información le siga en orden ascendente, es decir 42, se intercambia entonces con el elemento que se desea borrar, 27.



Eliminación de un nodo con dos subárboles no nulos.

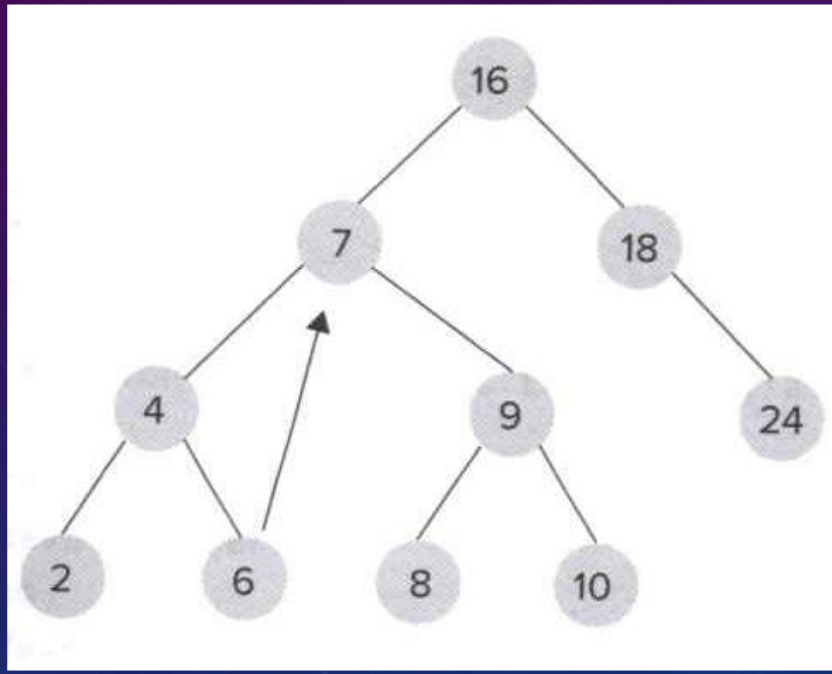
ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

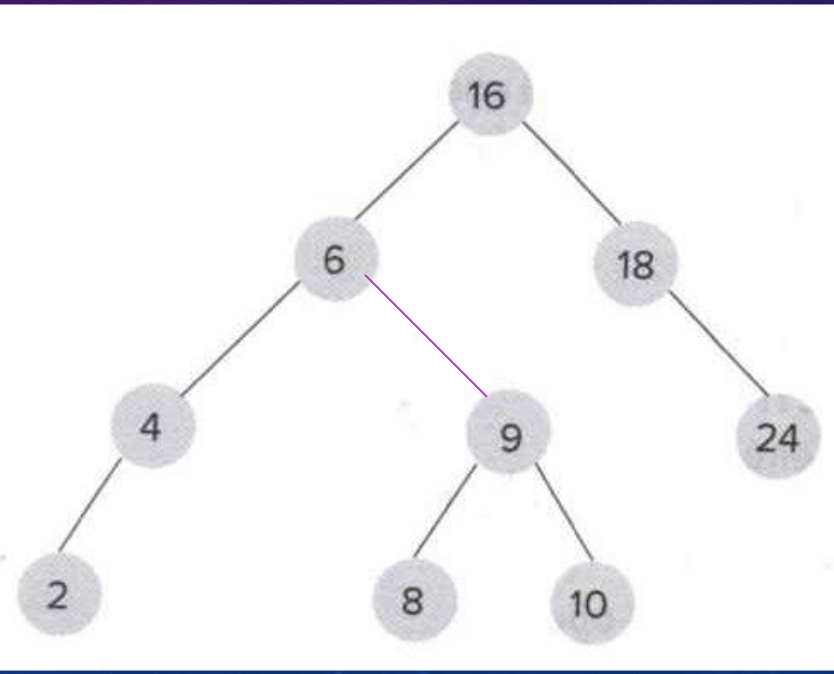
- Eliminar el elemento 3 en el **árbol A**.



ELIMINACIÓN DE UN ELEMENTO EN UN ÁRBOL BINARIO

- Eliminar el elemento 7 en el **árbol B**.





PROGRAMA DE RECORRIDOS EN PROFUNDIDAD EN UN ÁRBOL BINARIO

EJEMPLO 1

```
Los numeros colocados en el arbol son:
 29
 18
Izquierdo - (valor,papa): ( 18 , 29 )
  7
Izquierdo - (valor,papa): ( 7 , 29 )
Izquierdo - (valor,papa): ( 7 , 18 )
  6
Izquierdo - (valor,papa): ( 6 , 29 )
Izquierdo - (valor,papa): ( 6 , 18 )
Izquierdo - (valor,papa): ( 6 , 7 )
 36
Derecho - (valor,papa): ( 36 , 29 )
 42
Derecho - (valor,papa): ( 42 , 29 )
Derecho - (valor,papa): ( 42 , 36 )
  2
Izquierdo - (valor,papa): ( 2 , 29 )
Izquierdo - (valor,papa): ( 2 , 18 )
Izquierdo - (valor,papa): ( 2 , 7 )
Izquierdo - (valor,papa): ( 2 , 6 )
 40
Derecho - (valor,papa): ( 40 , 29 )
Derecho - (valor,papa): ( 40 , 36 )
Izquierdo - (valor,papa): ( 40 , 42 )
 12
Izquierdo - (valor,papa): ( 12 , 29 )
Izquierdo - (valor,papa): ( 12 , 18 )
Derecho - (valor,papa): ( 12 , 7 )
 35
Derecho - (valor,papa): ( 35 , 29 )
Izquierdo - (valor,papa): ( 35 , 36 )


El recorrido en preorden es:
 29 18  7  6  2 12 36 35 42 40

El recorrido en inorden es:
  2  6  7 12 18 29 35 36 40 42

El recorrido en postorden es:
  2  6 12  7 18 35 40 42 36 29
--------------------------------
```

Preorden: (raíz) (hijo izquierdo) (hijo derecho)

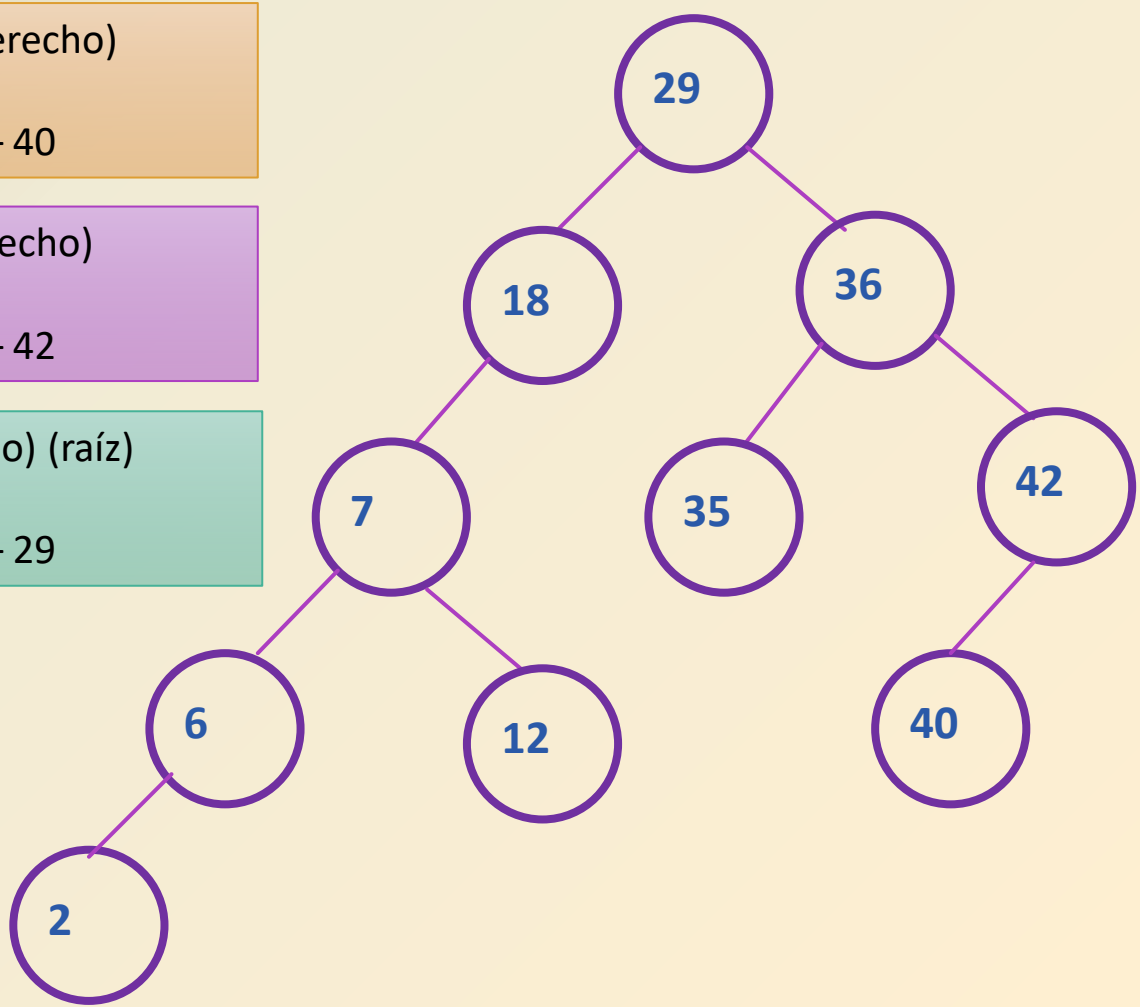
29 – 18 – 7 – 6 – 2 – 12 – 36 – 35 – 42 – 40

Inorden: (hijo izquierdo) (raíz) (hijo derecho)

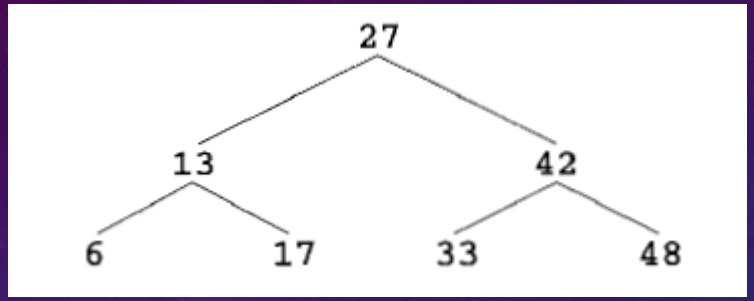
2 – 6 – 7 – 12 – 18 – 29 – 35 – 36 – 40 – 42

Postorden: (hijo izquierdo) (hijo derecho) (raíz)

2 – 6 – 12 – 7 – 18 – 35 – 40 – 42 – 36 – 29



EJEMPLO 2:



PreOrden	27 13 6 17 42 33 48
PostOrden	6 17 13 33 48 42 27
InOrden	6 13 17 27 33 42 48