Temas de Intro II

1.- Archivos

2.- Enumerado, Subrango y Registro

3.- Punteros y Listas

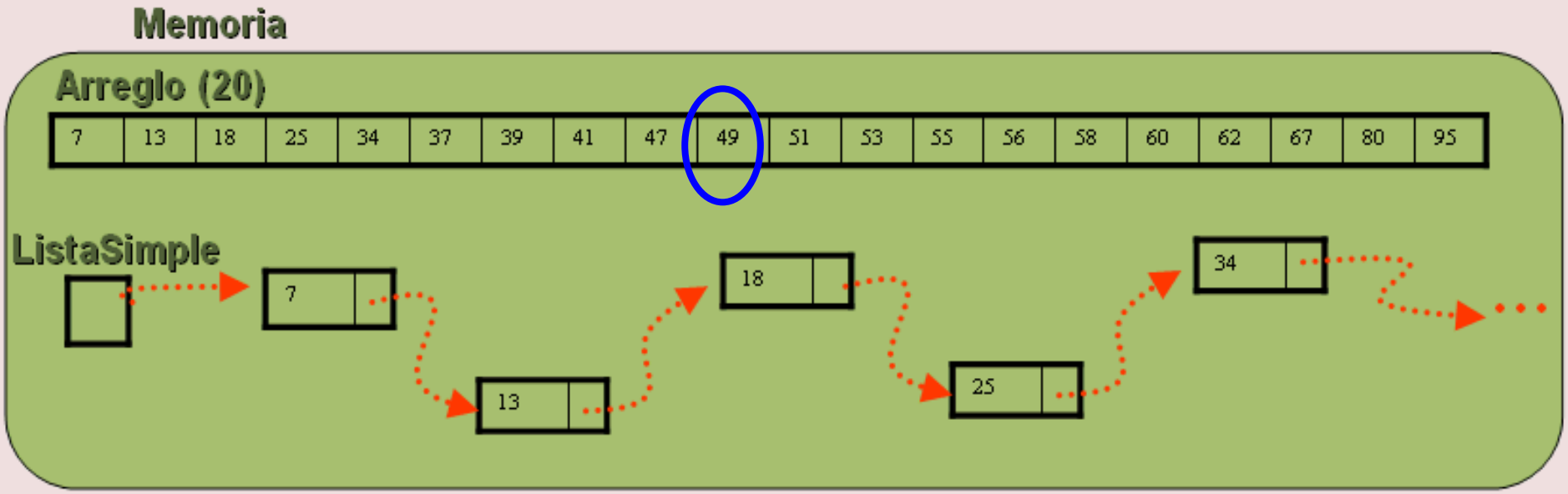
4.- Listas y Listas de Listas

5.- Recursión

6.- Árboles

7.- Trabajo

Costo de Acceso a los Datos



En **arreglos ordenados**: acceso por bisección/binario

→ Costo máximo: $\log_2 N$

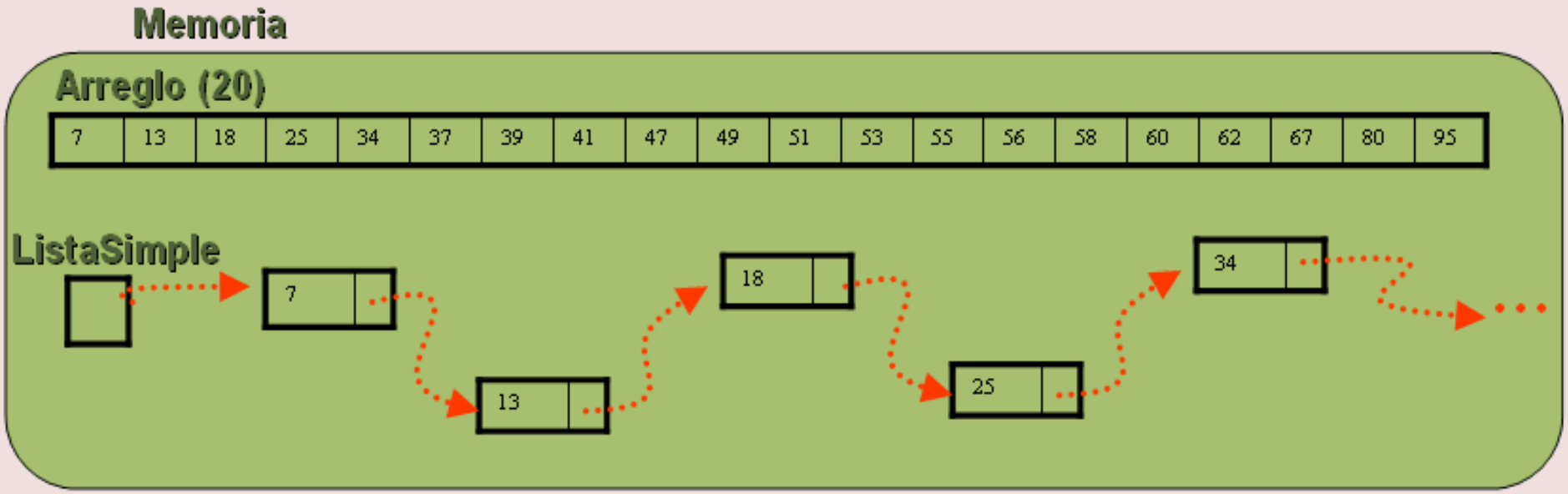
1000 elementos costo=10

En **lista vinculadas ordenadas**: acceso secuencial

→ costo promedio N/2

1000 elementos costo= 500

Costo de Acceso a los Datos

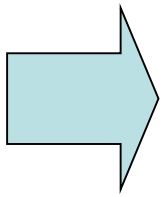


Arreglos: estructura estática

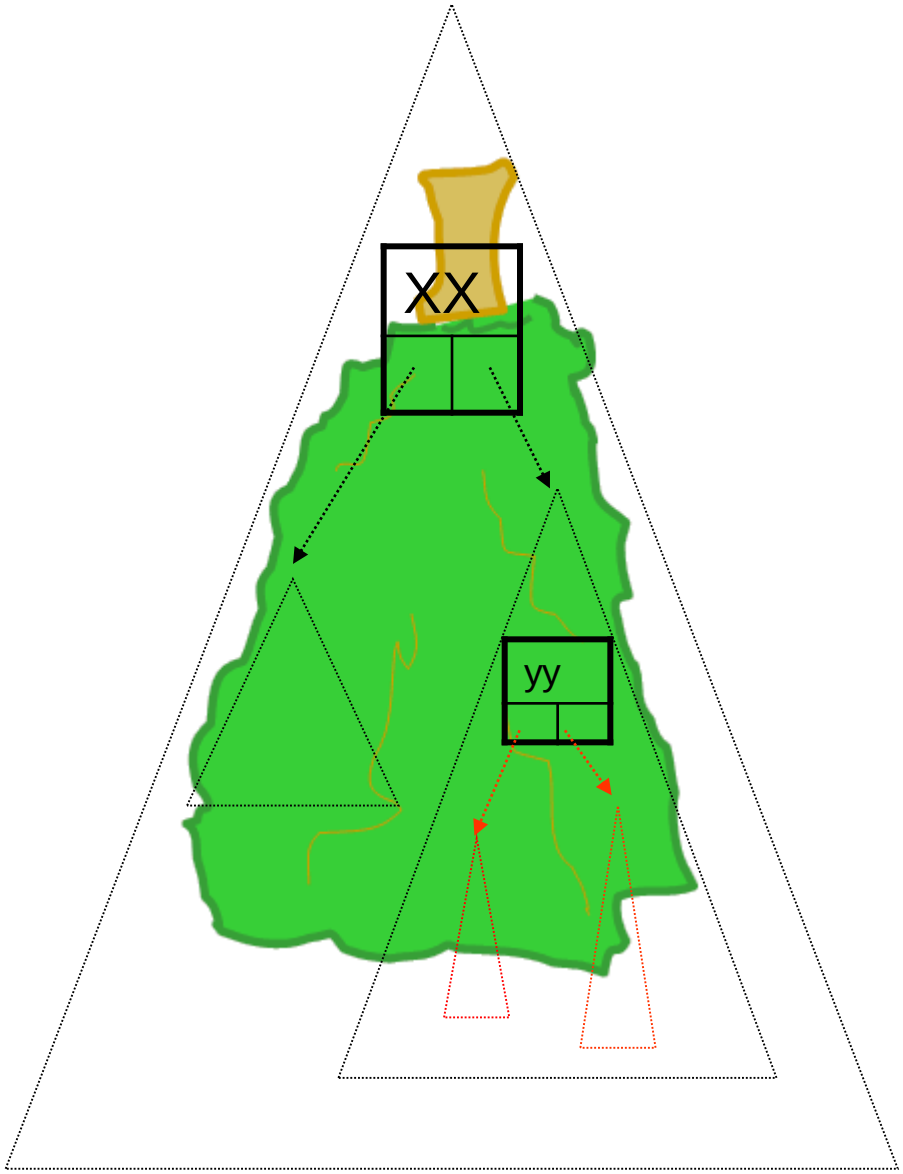
Lista vinculadas: estructura dinámica

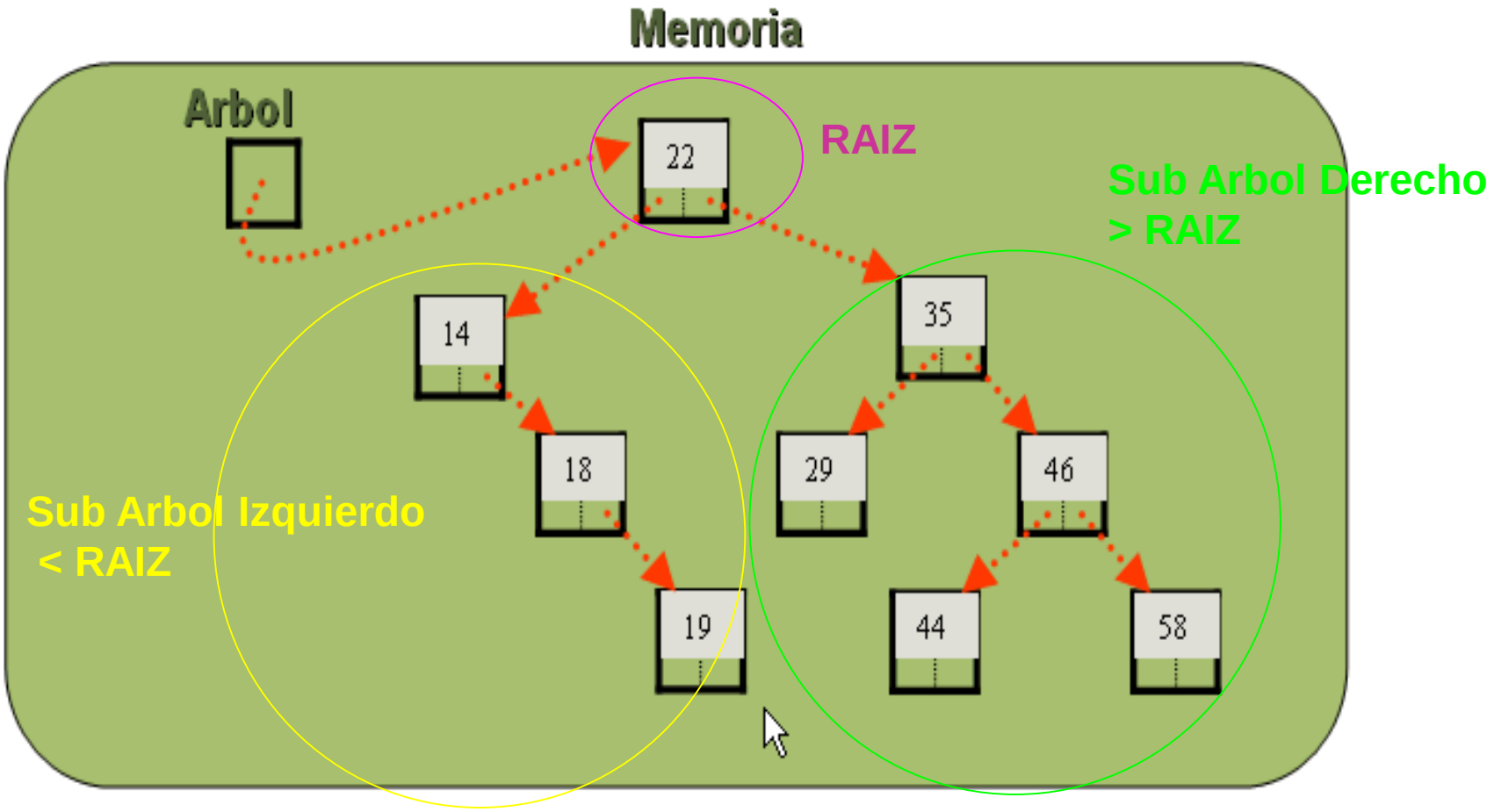
Costo de Acceso a los Datos

	Costo acceso max.	almacenamiento
Arreglos ordenados	✓ Log_2 N	estático
Listas vinculadas ordenados	Costo máximo: N	✓dinámico



Estructura de ARBOL

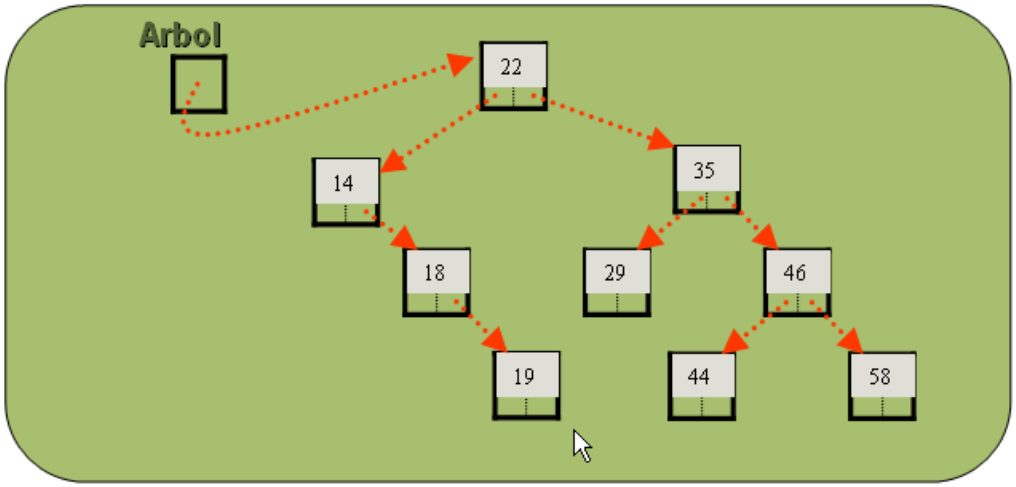

XX
yy



Si hay nodos con claves repetidas, establecer en que subárbol irán y seguir ese criterio

Definición de la estructura de árbol en Pascal

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
          Nro: Integer;
          Menores: PuntArbol;
          Mayores: PuntArbol
End;
```

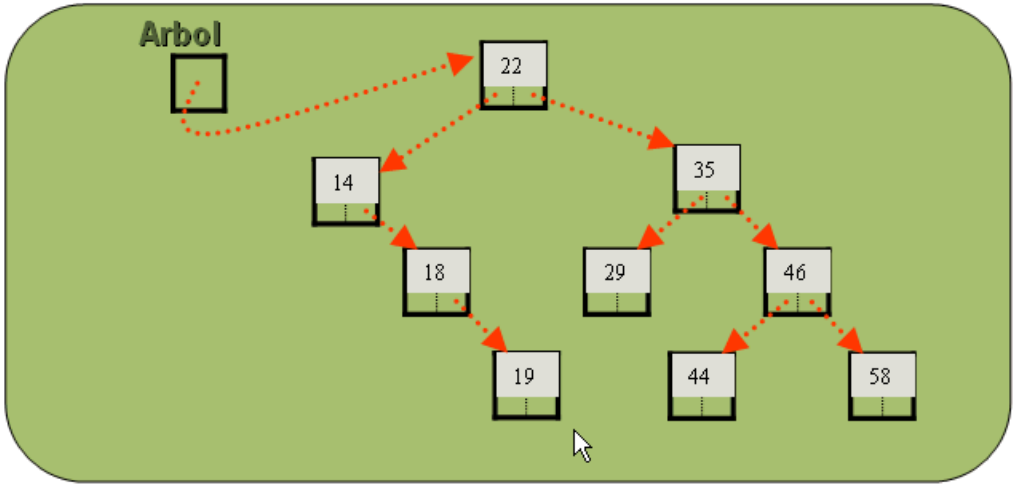



```
Var
   ElArbol: PuntArbol;
```

Comparar estructura con listas doblemente vinculadas

Grado 2 (binario)

Altura 4



Grado: número máximo de hijos que tienen los nodos. (la lista es árbol de grado 1)

Nodo Padre de un nodo X es aquel que apunta al mismo. Cada nodo sólo puede tener un padre

Nodo hijo: de otro nodo Z es cualquier nodo apuntado por el nodo Z

Nodo Raíz: es el único nodo del árbol que no tiene padre.

Hoja: nodo que no tiene hijos

Nodo interior: es aquel que no es ni hoja ni Raíz

Camino: secuencia de nodos en donde dos nodos consecutivos cualesquiera son padre e hijo.

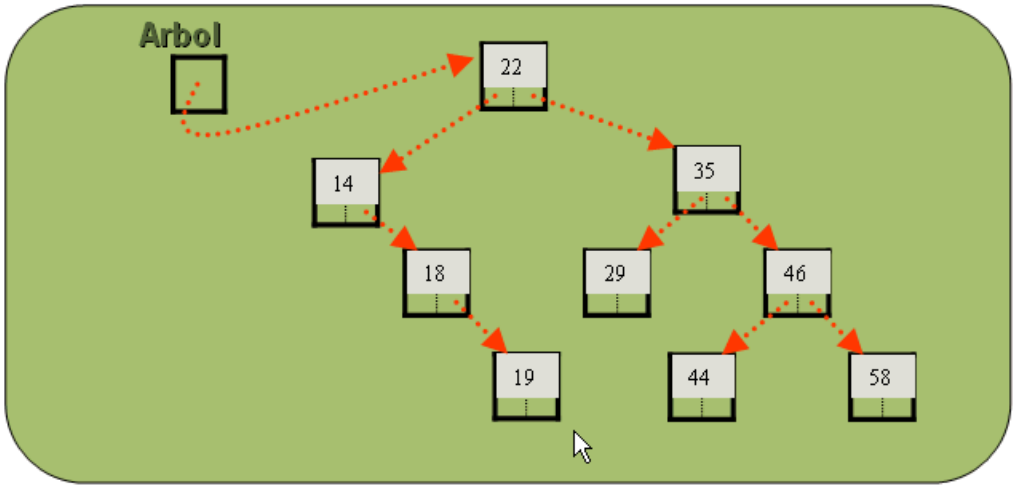
Rama: camino desde el nodo Raíz a una hoja

Nivel de un nodo: número de nodos del camino desde la Raíz hasta dicho nodo. Nivel de Raíz: 1

Altura: la rama más larga

Recorrido de un Árbol en orden Ascendente

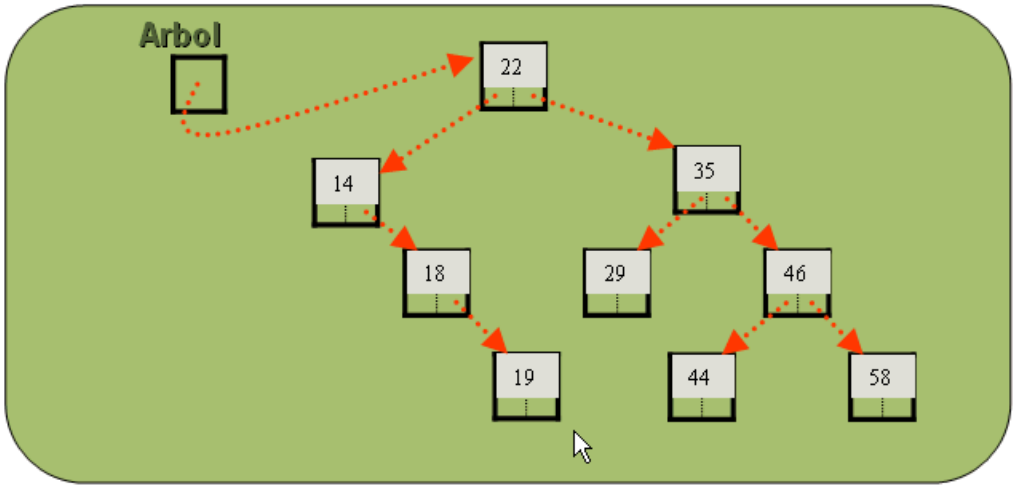
```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
            Nro: Integer;
            Menores: PuntArbol;
            Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```




```
Procedure Mostrardatos (elarbol: Puntarbol);
Begin
   If (elarbol <> nil) then begin
            Mostrardatos (elarbol^.Menores);
            writeln (elarbol^.Nro);
            Mostrardatos (elarbol^.Mayores);
   End;
End;
```

Recorrido de un Árbol en orden Descendente

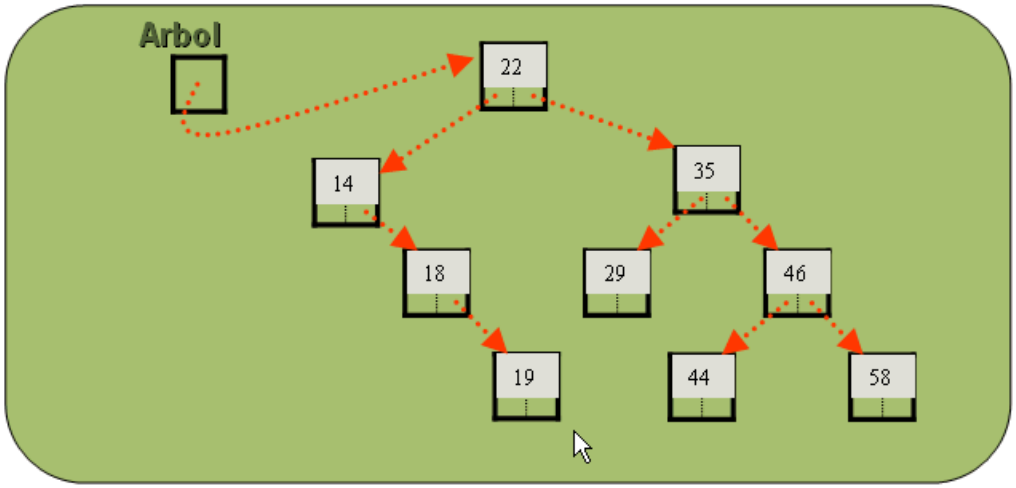
```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
            Nro: Integer;
            Menores: PuntArbol;
            Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```




```
Procedure Mostrardatos (elarbol: Puntarbol);
Begin
   If (elarbol <> nil) then begin
            Mostrardatos (elarbol^.Mayores);
            writeln (elarbol^.Nro);
            Mostrardatos (elarbol^.Menores);
   End;
End;
```

Recorrido de un Árbol: Inorder

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
            Nro: Integer;
            Menores: PuntArbol;
            Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```

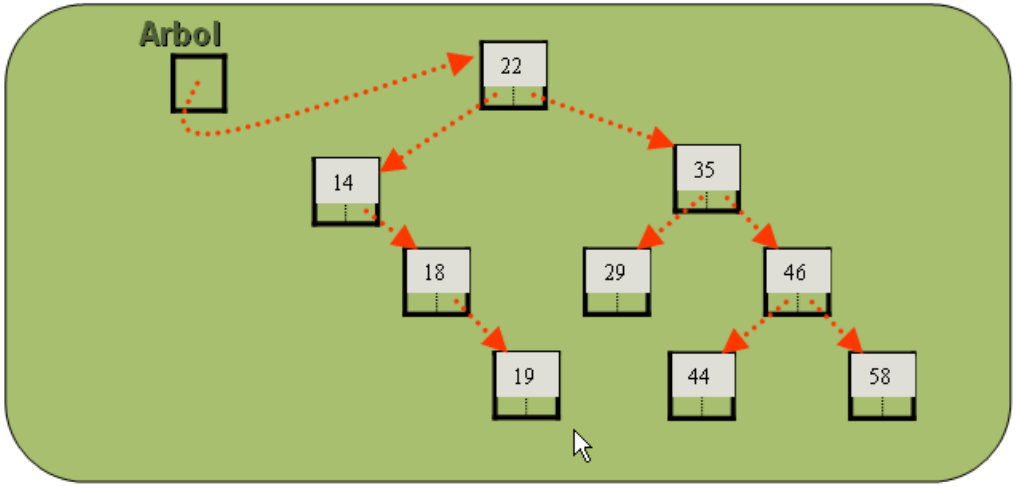


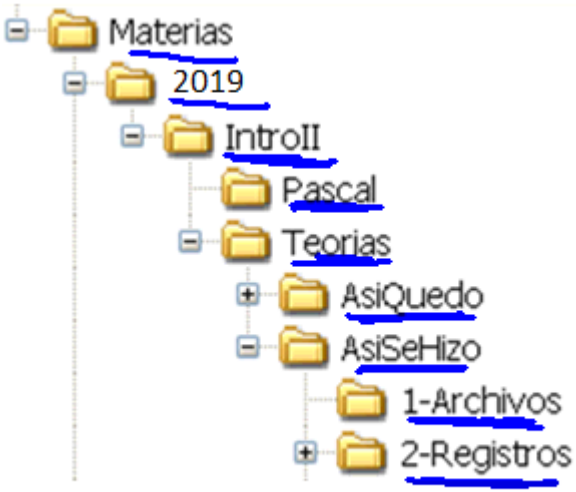

```
Procedure  ProcesardatosInOrder (elarbol:
Puntarbol);
Begin
  If (elarbol <> nil) then begin
            ProcesardatosInOrder (elarbol^.Menores);
            ProcesarNodo (elarbol^.Nro);
            ProcesardatosInOrder (elarbol^.Mayores);
  End;
End;
```

Recorrido INORDER:
Se recorren los nodos del árbol siguiendo el orden ascendente o descendente según el contenido de los nodos

Recorrido de un Árbol: Preorder

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
            Nro: Integer;
            Menores: PuntArbol;
            Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```

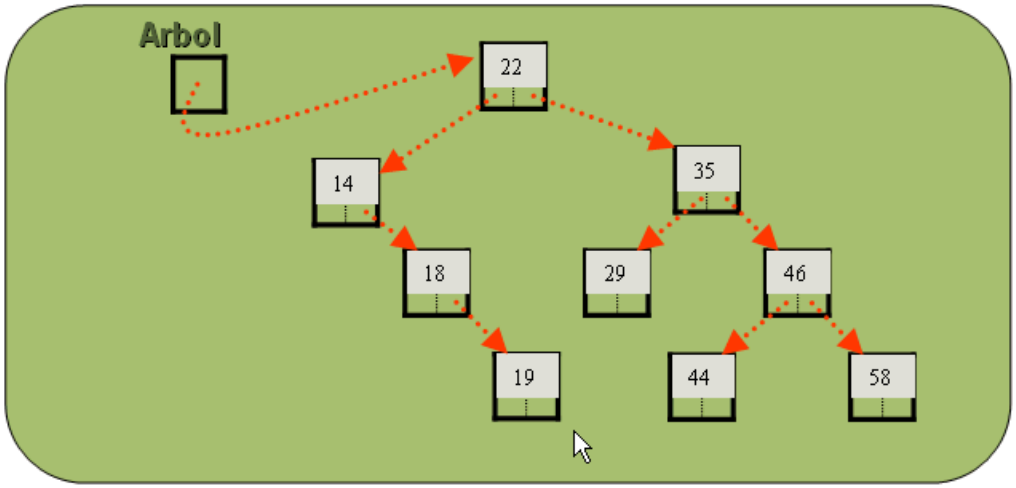





```
Procedure ProcesarDatosPreOrder (elarbol:
Puntarbol);
Begin
  If (elarbol <> nil) then begin
            ProcesarNodo (elarbol^.Nro);
            ProcesarDatosPreOrder (elarbol^.Menores);
            ProcesarDatosPreOrder (elarbol^.Mayores);
  End;
End;
```

Recorrido de un Árbol: PostOrder

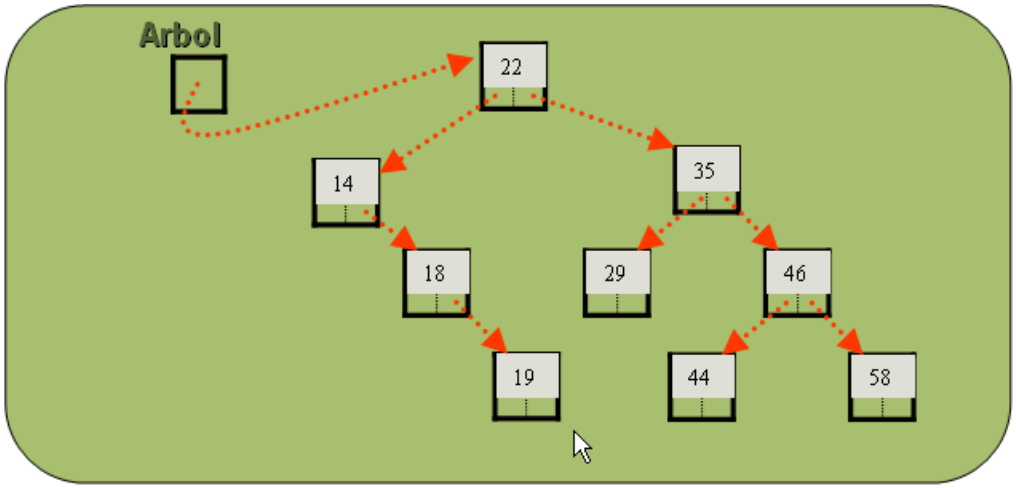
```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
            Nro: Integer;
            Menores: PuntArbol;
            Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```




```
Procedure  ProcesarDatosPostOrder (elarbol: Puntarbol);
Begin
  If (elarbol <> nil) then begin
            ProcesarDatosPostOrder (elarbol^.Menores);
            ProcesarDatosPostOrder (elarbol^.Mayores);
            ProcesarNodo (elarbol^.Nro);
  End;
End;
```

Búsqueda de un dato en un Árbol ordenado

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
                Nro: Integer;
                Menores: PuntArbol;
                Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```



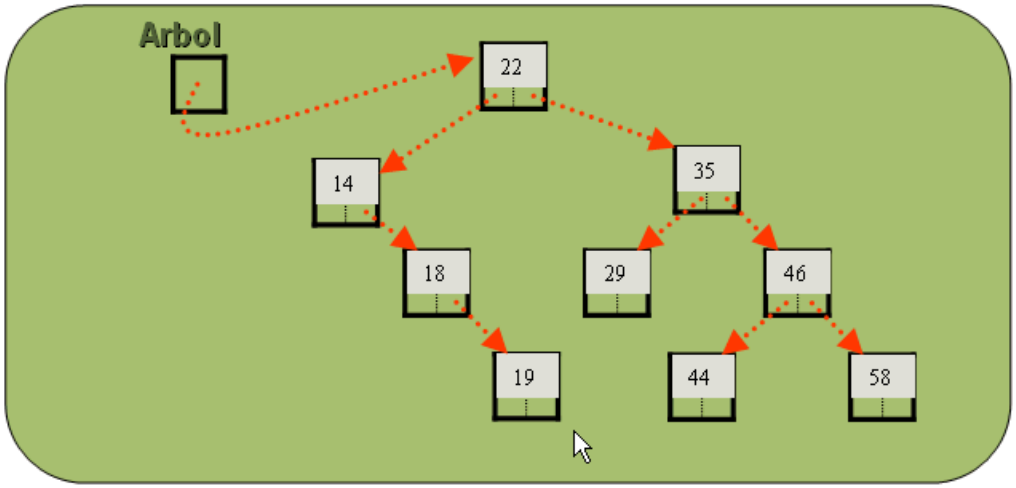
Cómo hago para buscar el elemento 29?

Cómo hago para buscar el elemento 18?

Cómo hago para buscar el elemento 17?

Búsqueda de un dato en un Árbol ordenado

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
                Nro: Integer;
                Menores: PuntArbol;
                Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```



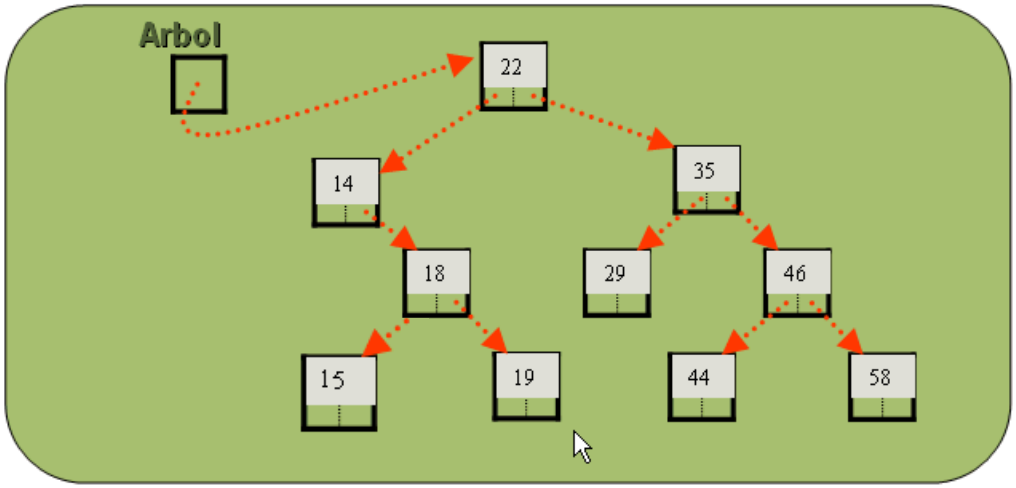

```
Function Punteroalnodo (elarbol: Puntarbol; dato: integer): Puntarbol;
begin
 If elarbol = nil   then
      Punteroalnodo:= nil
 else  if elarbol^.Nro = dato  then
     Punteroalnodo:= elArbol
 else if elarbol^.Nro  < dato  then
       Punteroalnodo := Punteroalnodo (elarbol^.Mayores, dato)
  else
       Punteroalnodo := Punteroalnodo (elarbol^.Menores,dato)
End;
```

```
Function Punteroalnodo(elarbol: Puntarbol; dato: integer):Puntarbol;

begin


If elarbol = nil   then
            Punteroalnodo:= nil
 else  if elarbol^.Nro = dato  then
            Punteroalnodo:= elArbol
 else if elarbol^.Nro  < dato  then
      Punteroalnodo := Punteroalnodo(elarbol ^.Mayores, dato)
  else
      Punteroalnodo := Punteroalnod (elarbol^.Menores,dato)
End;
```

CORTE de la recursión

Alta de un nodo en un Árbol ordenado

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
                Nro: Integer;
                Menores: PuntArbol;
                Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```

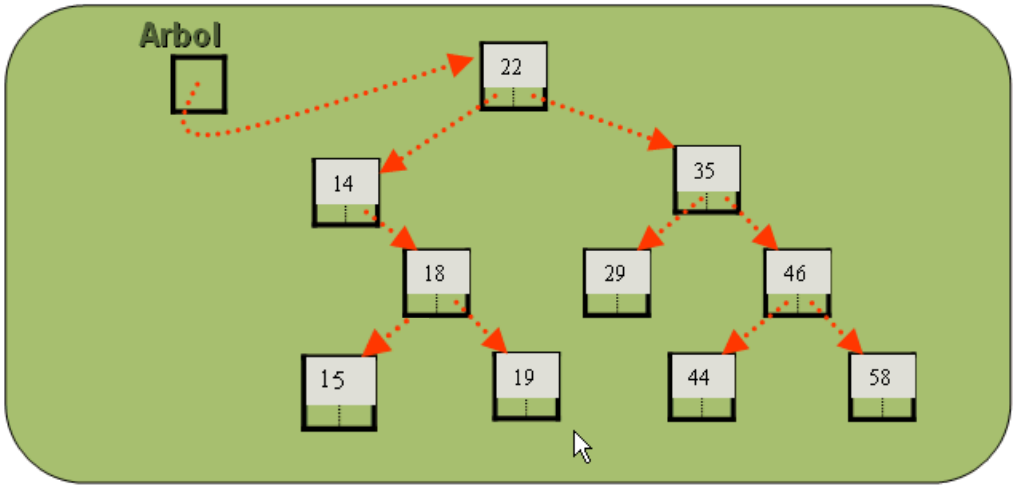


Insertar el 15

El alta se da siempre en las hojas

Alta de un nodo en un Árbol ordenado

```
Type PuntArbol = ^TipoNodoArbol;

TiponodoArbol = record
                Nro: Integer;
                Menores: PuntArbol;
                Mayores: PuntArbol
End;
…
Var
   ElArbol: PuntArbol;
```

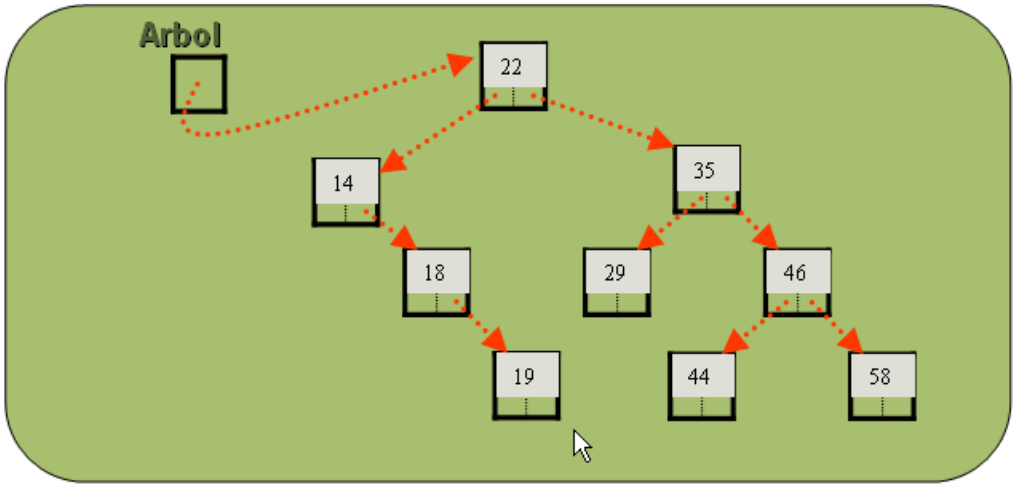



```
Procedure Alta( var elarbol: Puntarbol; NuevoNodo: Puntarbol);
begin
 If elarbol = nil
   then
       elarbol:= NuevoNodo;
    else if (elarbol^.Nro  < NuevoNodo^.dato )
         then  Alta (elarbol^.Mayores, NuevoNodo)
          else  Alta (elarbol^.Menores, NuevoNodo);
 end.
```

```
Procedure Alta(var elarbol: Puntarbol; NuevoNodo: Puntarbol);


begin
  If elarbol = nil
    then
        elarbol:= NuevoNodo;
    else
      if (elarbol^.Nro  < NuevoNodo ^.dato)
            then  Alta (elarbol^.Mayores, NuevoNodo)
            else  Alta (elarbol^.Menores, NuevoNodo);
End;
```

Dar de alta: 22 14 35 18 15 35 46

Baja de un nodo en un Árbol ordenado

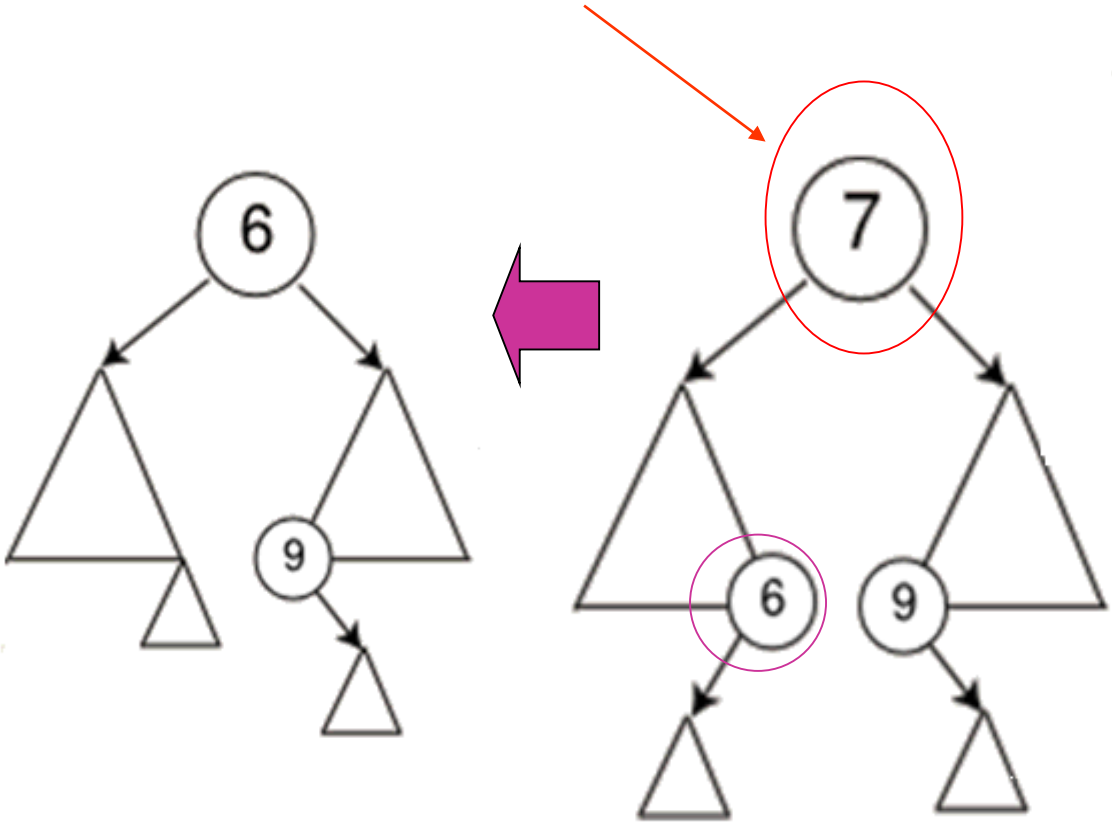


1) Borrar un nodo Hoja (sin hijos): se borrar y se pone en nil el puntero de su nodo padre

2) Borrar un nodo con un solo subárbol: se borra el nodo y se asigna su subárbol hijo como subárbol de su padre

3) Borrar un nodo son dos subárboles

Esquema de Borrado de un nodo son un dos subárboles

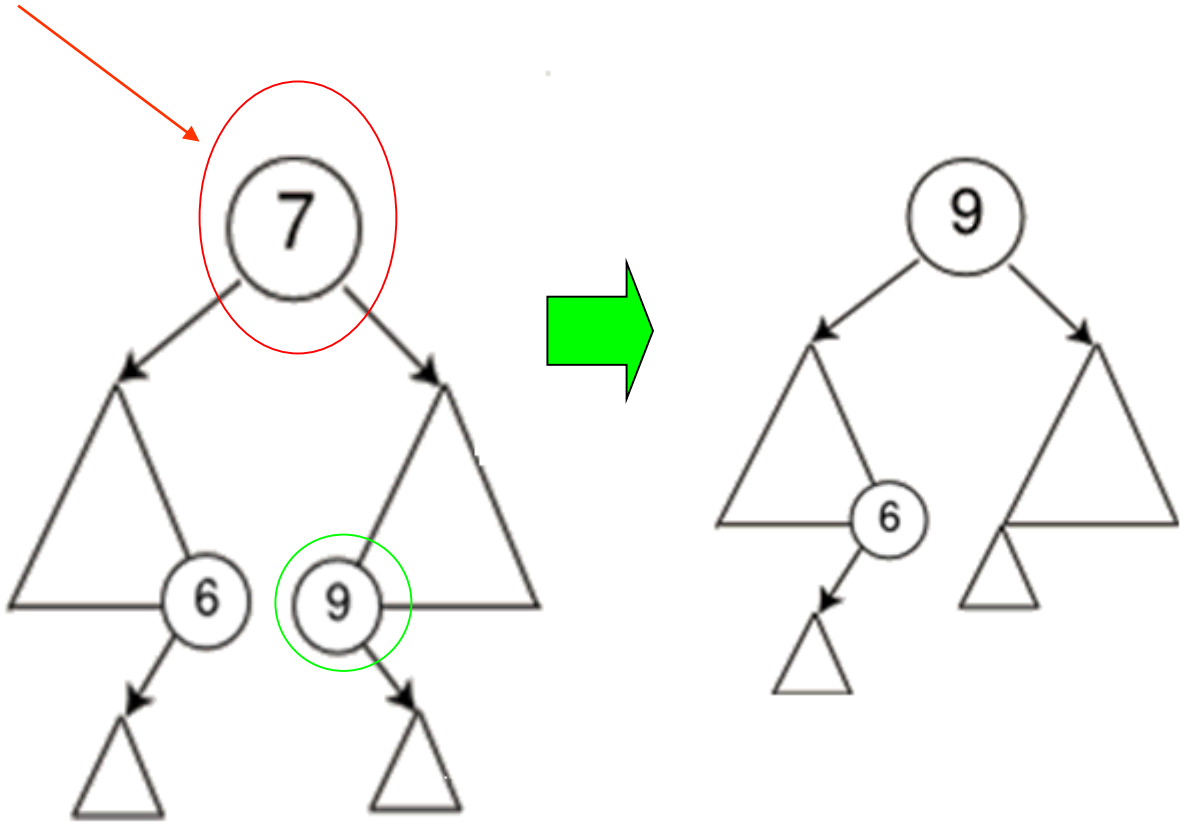


Alternativa 1:

1) Buscar el más grande de los hijos menores (es decir el nodo más grande del subárbol izquierdo)

2) Se “reemplaza” el nodo a borrar por ese nodo encontrado realizando las modificaciones necesarias, considerando que el nodo encontrado puede tener a lo sumo hijos por la rama izquierda

Esquema de Borrado de un nodo son un dos subárboles



Alternativa 2: idem Alt. 1

Busco el más chico de los hijos mayores (es decir el nodo más chico del subárbol derecho)

Modificación de los datos de un árbol

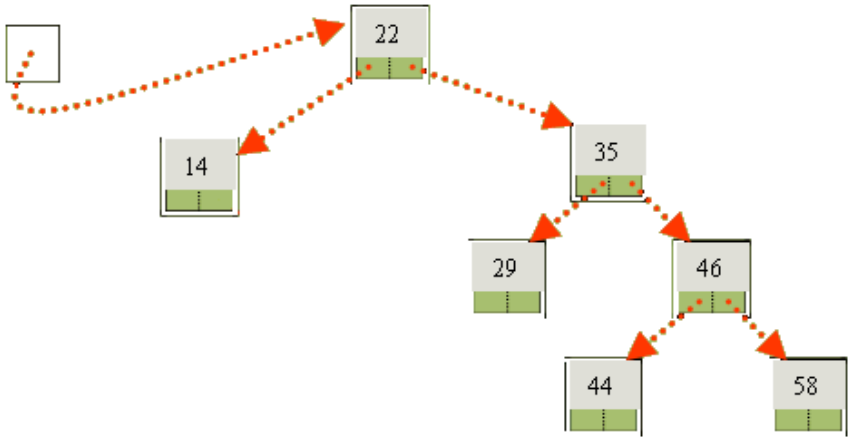
- Si el dato no es clave de ordenamiento: Se modifica el nodo.

Árbol de Alumnos ordenado por DNI donde se modifica la dirección de un alumno.

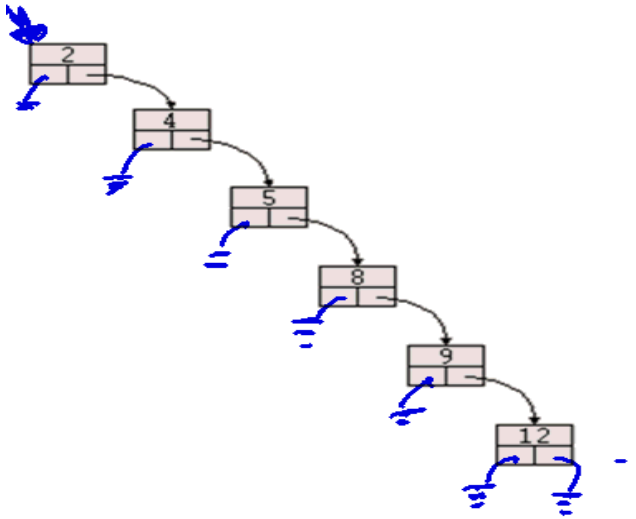
- Si el dato es clave de ordenamiento: Se modifica la posición en el árbol. →
  - Dar de baja y dar de alta

Árbol de Alumnos ordenado por Nro. Alumno.
Se modifica un Nro.Alumno -> puede cambiar el orden.

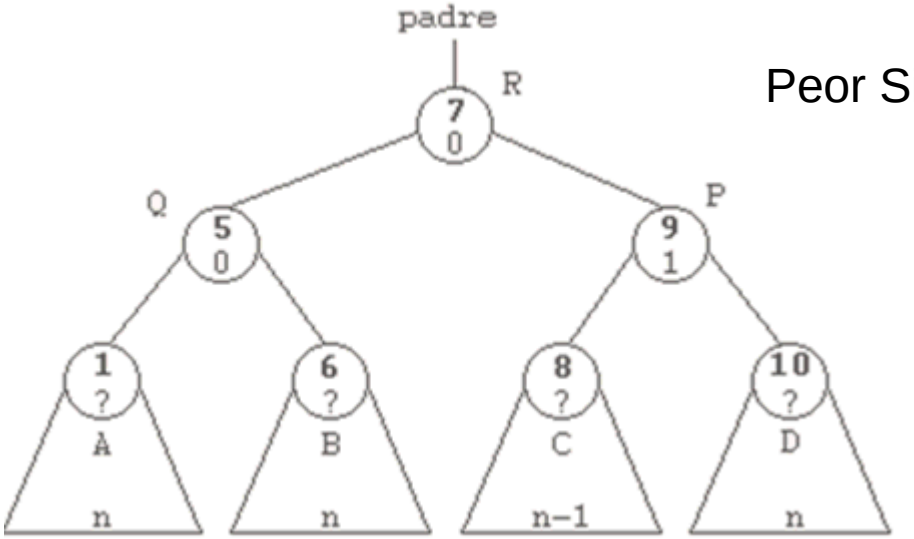
Balanceo de un árbol



Situación intermedia



Peor Situación



Mejor Situación