Árboles Binarios

Estructuras de Datos no Lineales

Unidad 4

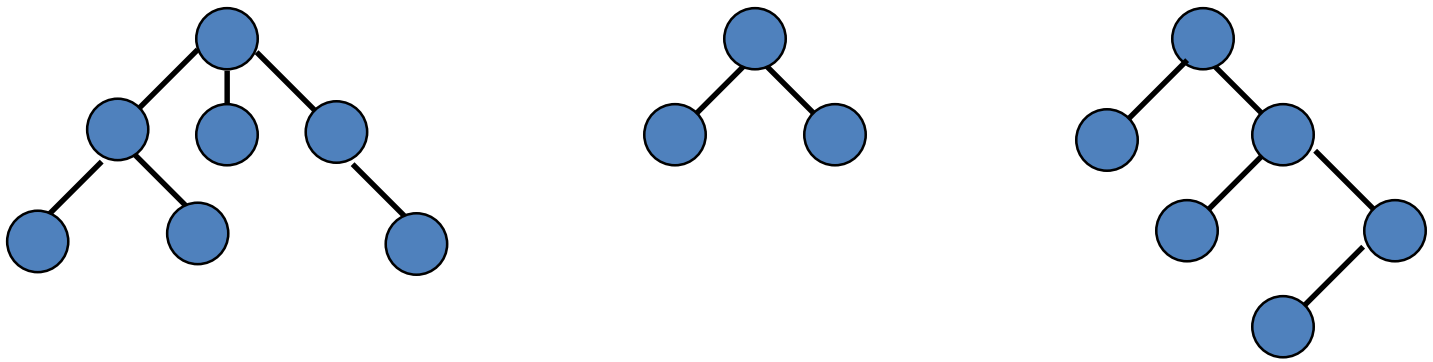
Las estructuras dinámicas son las en la ejecución varia el número de elementos y uso de memoria a lo largo del programa)

Entre estas tenemos:

Lineales (listas enlazadas, pilas y colas)
No lineales (arboles binarios y grafos o redes)

¿Qué es un Árbol?

- Es una estructura de datos jerárquica.
- La relación entre los elementos es de uno a muchos.



Arboles

• un árbol representa una jeraquía
ejemplos:

estructura organizativa de una empresa

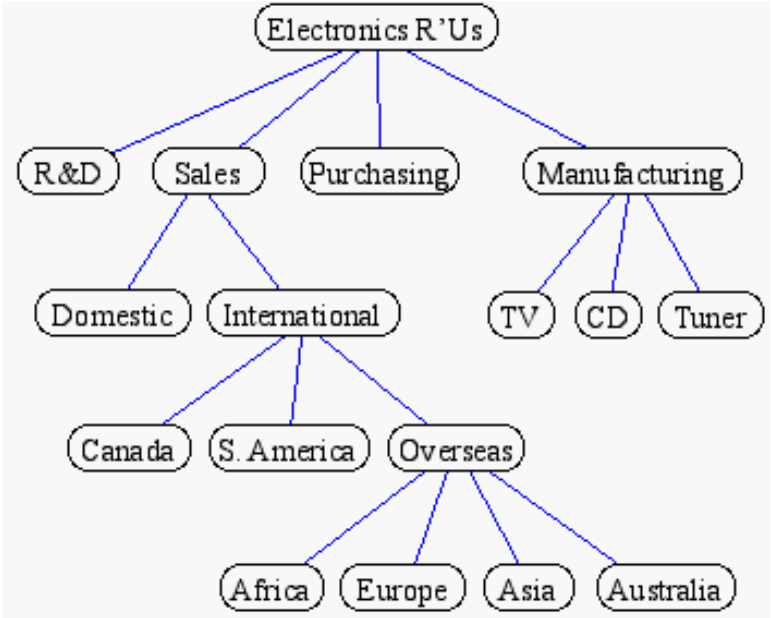
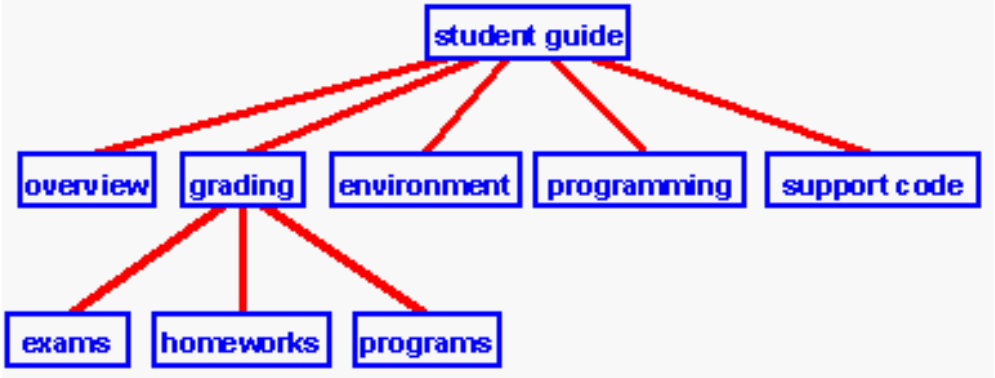
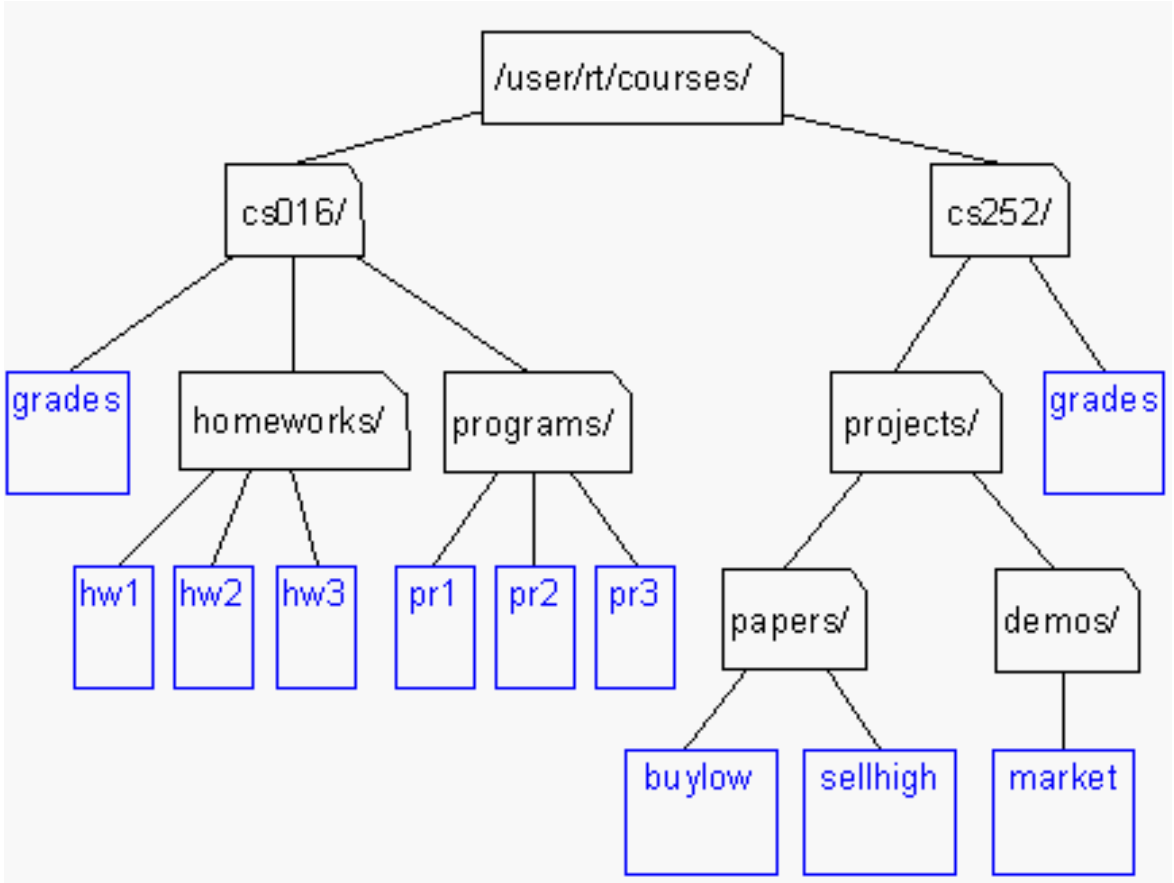


tabla de contenido de un lib



Arboles (1)

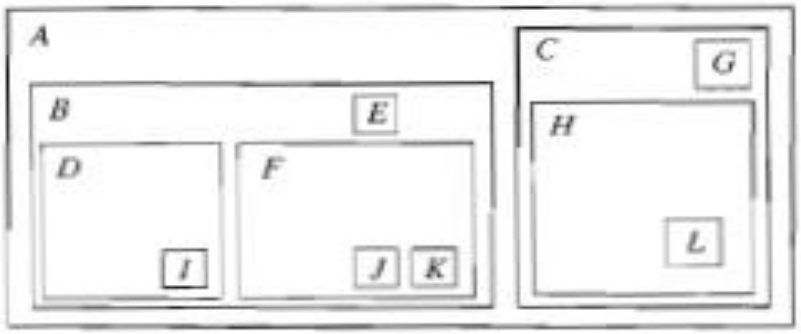
- ` Sistema de ficheros de Unix o DOS/Windows



Representación de Arboles

a) Diagramas de Venn
b) Anidación de paréntesis
c) Notación decimal de Dewey
d) Notación indentada
e) Grafo

Representación de Arboles



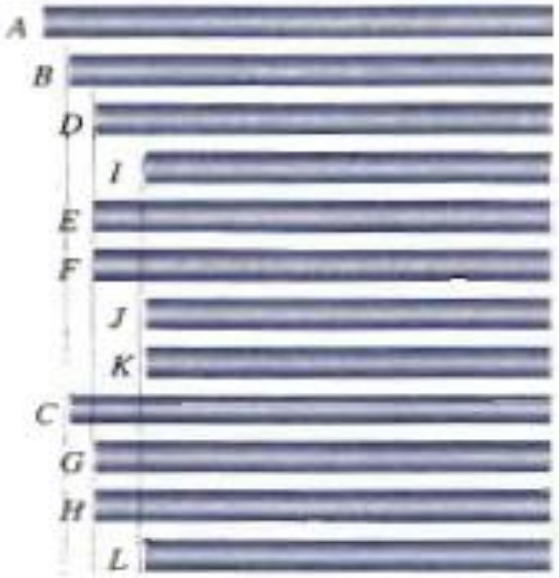
a)

(A (B (D (I), E, F, (J, K)), C (G, H (L))))

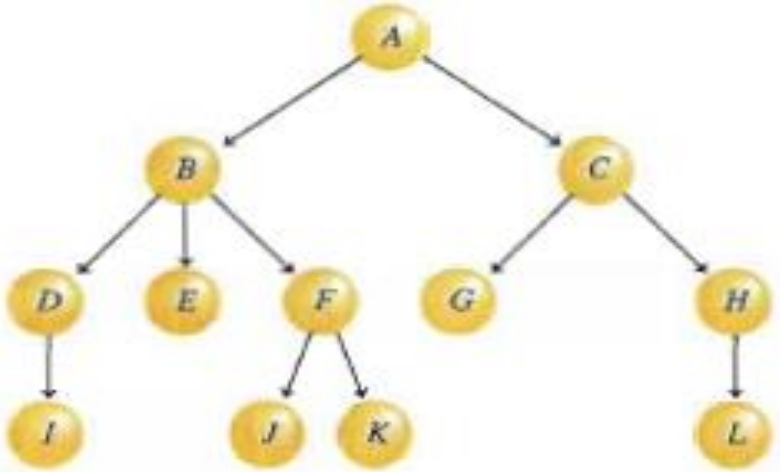
b)

1.A, 1.1.B, 1.1.1.D, 1.1.1.1.I, 1.1.2.E, 1.1.3.F, 1.1.3.1.J, 1.1.3.2.K, 1.2.C, 1.2.1.G, 1.2.2.H, 1.2.2.1.L

c)



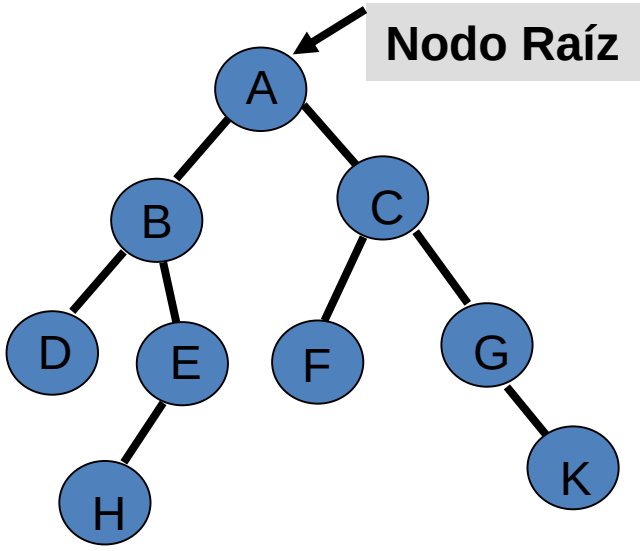
d)



e)

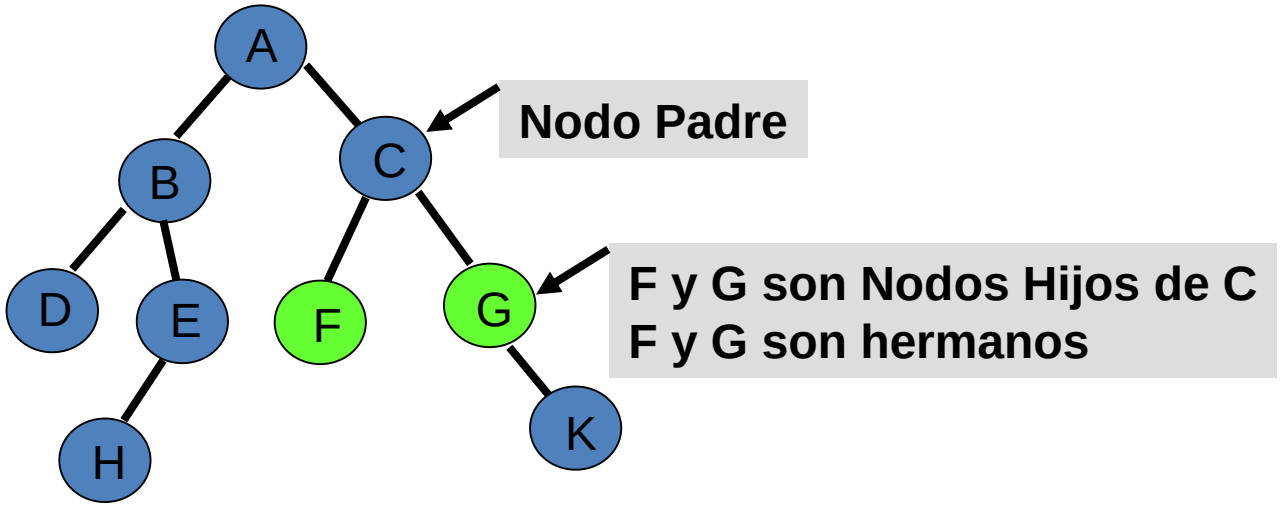
Terminología

- Nodo: Cada elemento en un árbol.
- Nodo Raíz: Primer elemento agregado al árbol.



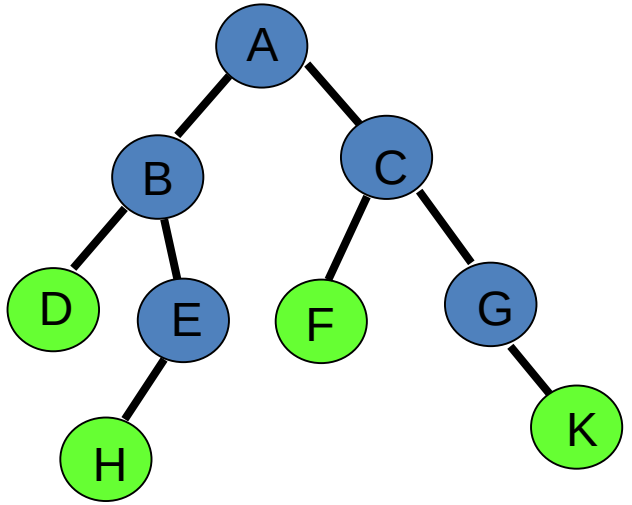
Más terminología

- Nodo Padre: Se le llama así al nodo predecesor de un elemento.
- Nodo Hijo: Es el nodo sucesor de un elemento.
- Hermanos: Nodos que tienen el mismo nodo padre.



Más terminología

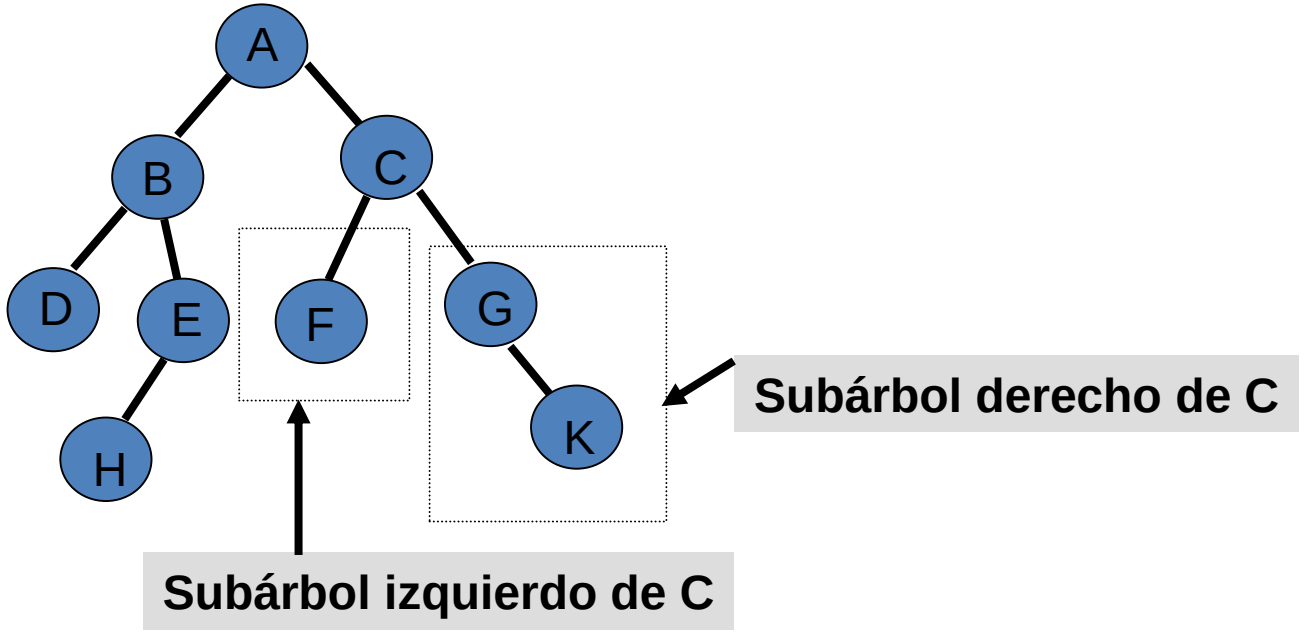
- Nodo Hoja: Aquel nodo que no tiene hijos.



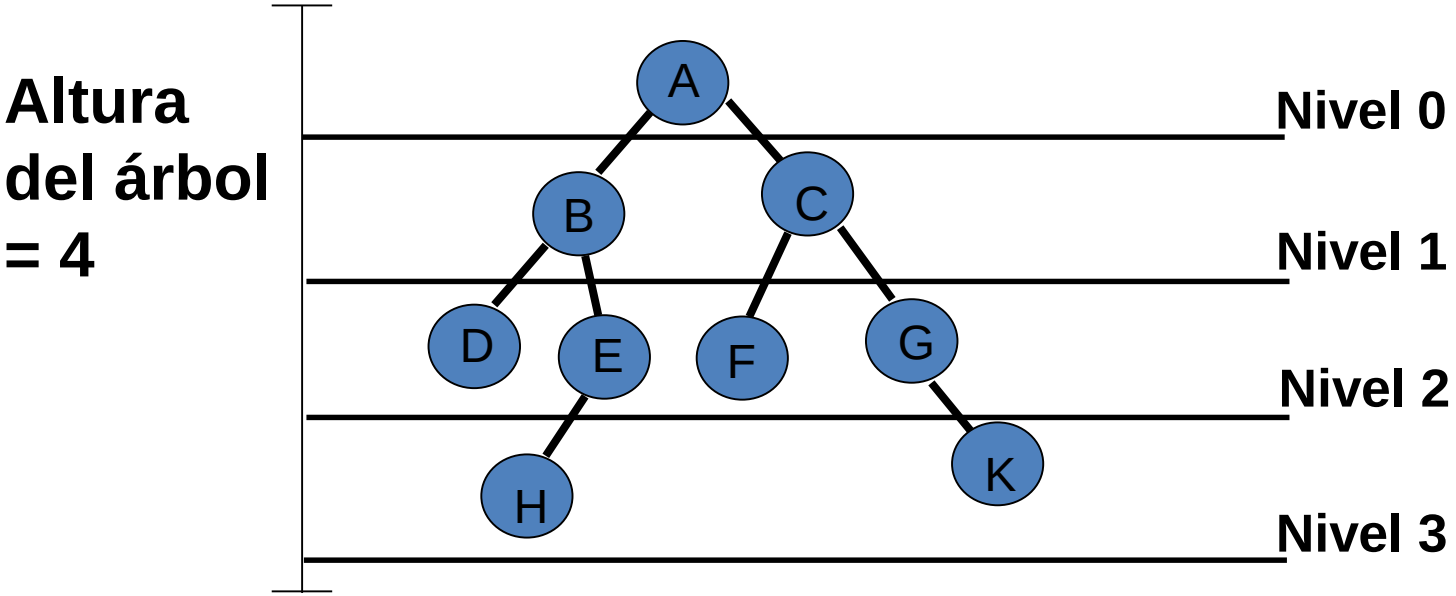
D, H, F y K son Nodos Hojas

Más terminología

- Subárbol: Todos los nodos descendientes por la izquierda o derecha de un nodo.



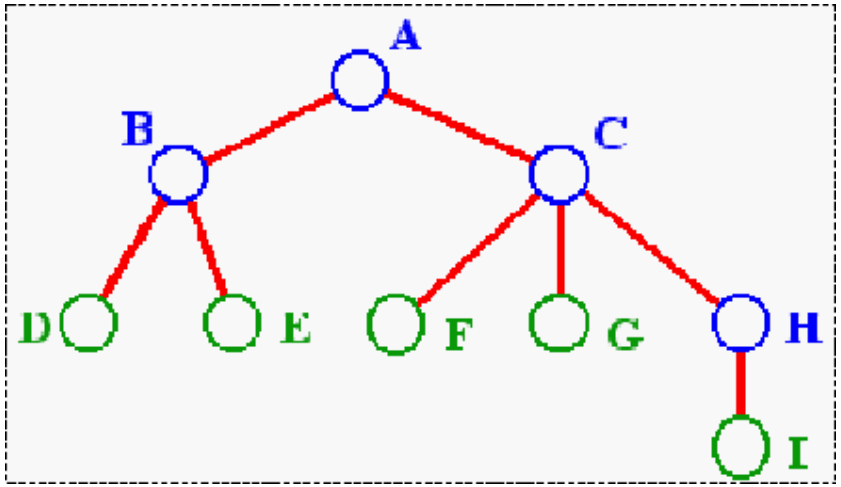
Altura y Niveles



La Altura es la cantidad de niveles.

Terminología de Arboles

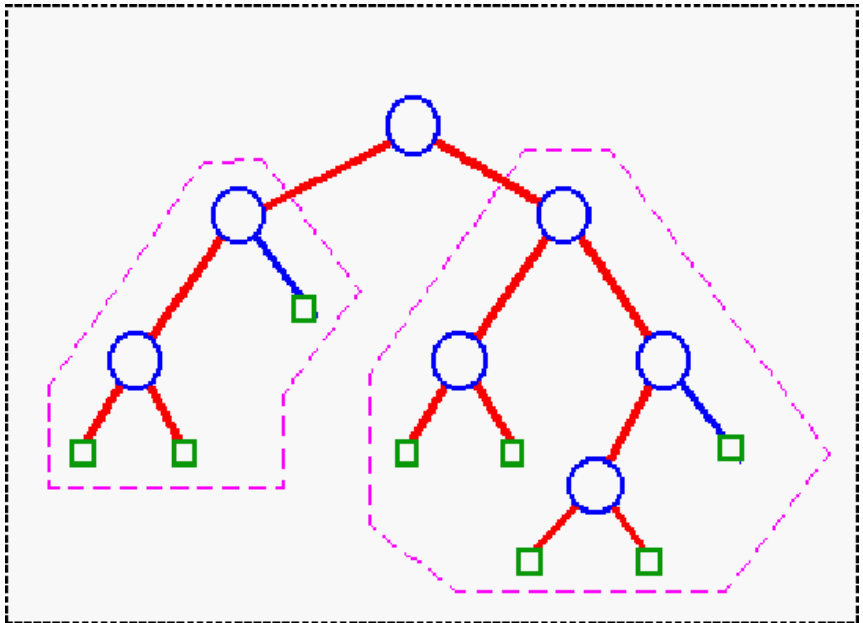
- A es el nodo raíz
- B es el padre de D y E
- C es el hermano de B
- D y E son los hijos de B
- D, E, F, G, I son nodos externos o hojas
- B, C, H son nodos internos
- La profundidad (nivel) de E es 2
- La altura del árbol es 4
- El grado del nodo B es 2



- Propiedad: (#aristas) = (#nodos) - 1

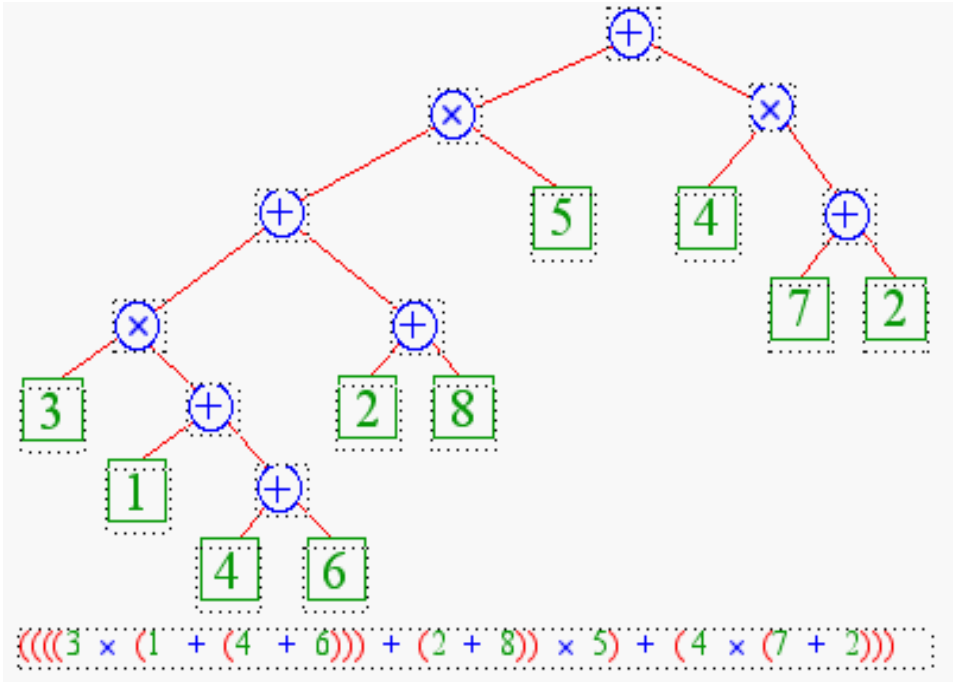
Arboles binarios

- Arbol ordenado: el hijo de cada nodo está ordenado
- Arbol binario: árbol ordenado con todos los nodos internos de grado 2
- Definición recursiva de árbol binario:
- Un árbol binario es:
  - un nodo externo (hoja) o
  - un nodo interno (la raíz) y dos árboles binarios (subárbol izquierdo y subárbol derecho)

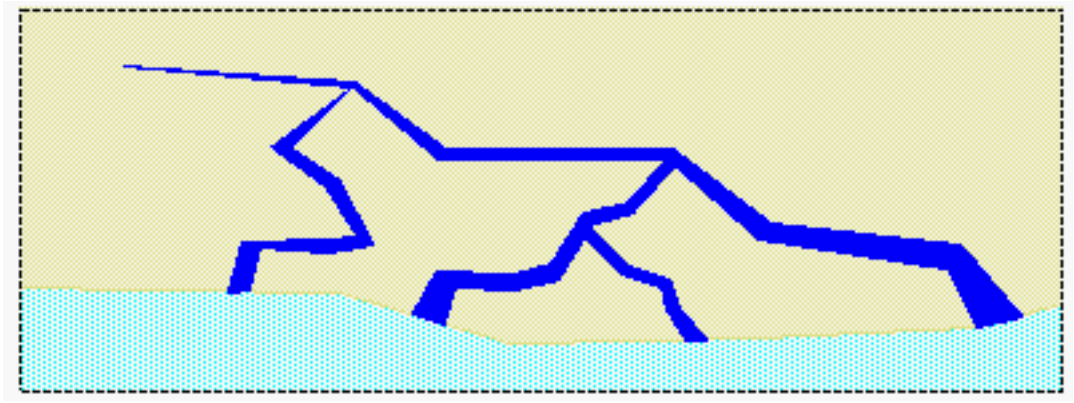


Ejemplos de Arboles Binarios

- expresión aritmética

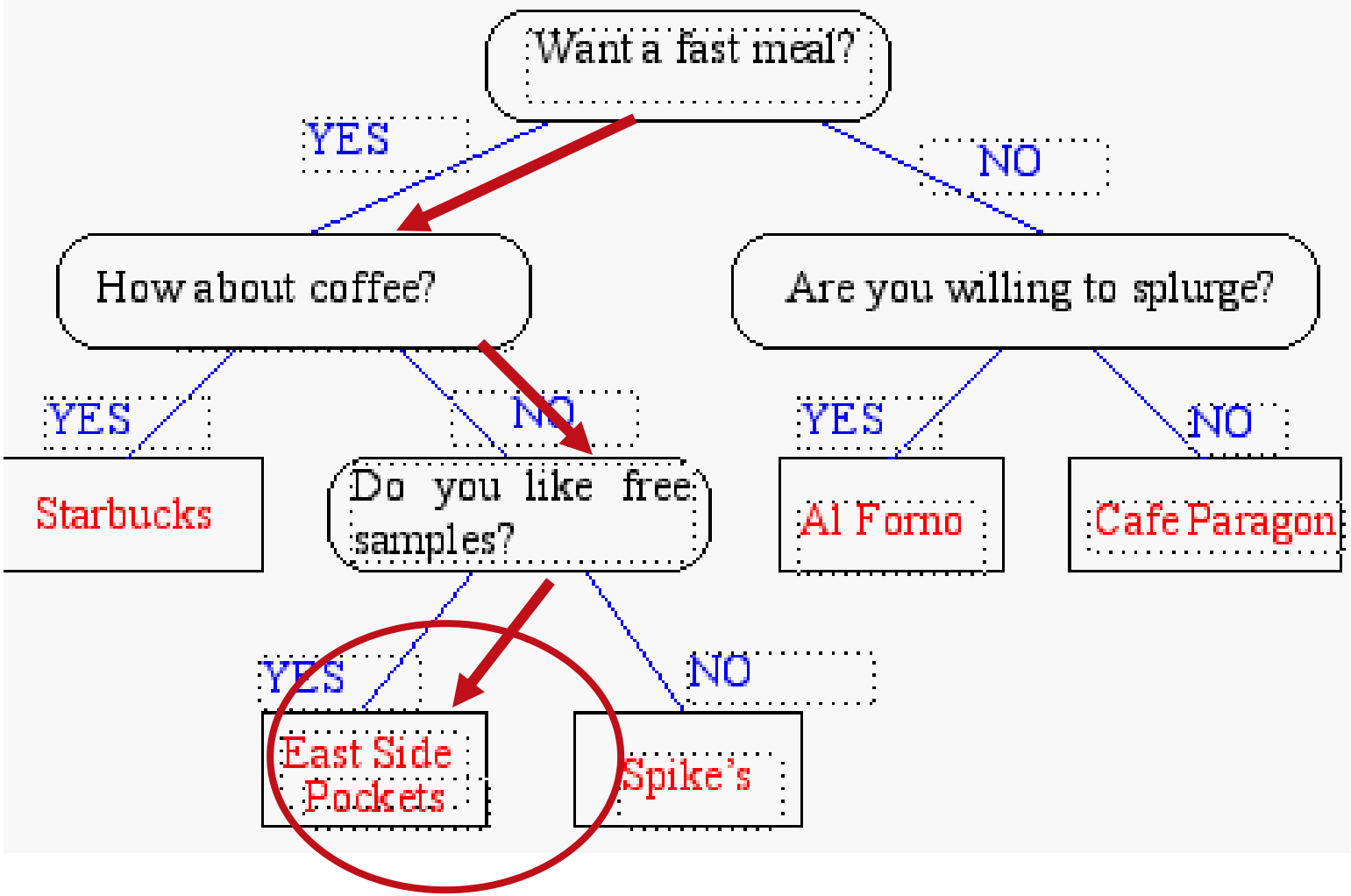


- río especial



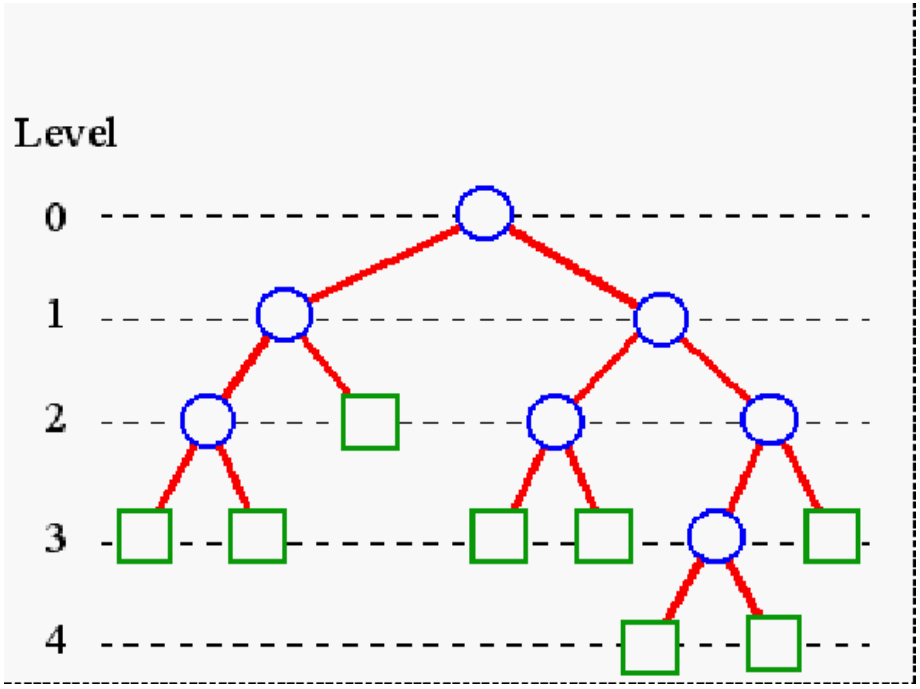
Ejemplos de Arboles Binarios

- Árboles de decisión

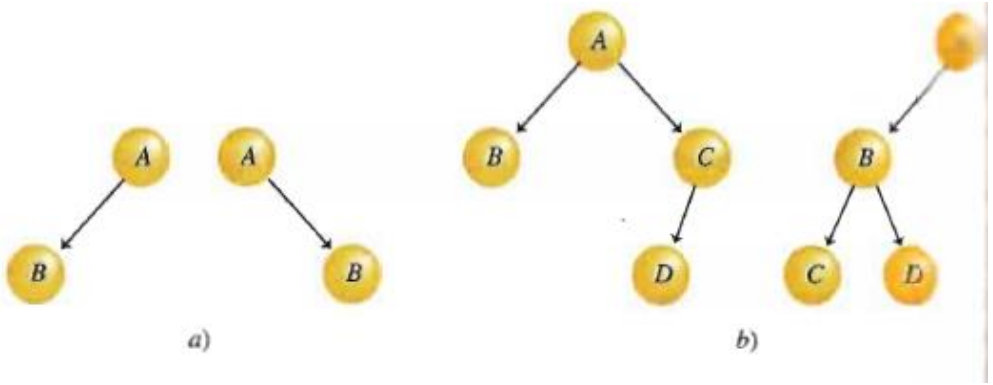


Propiedades de Arboles Binarios

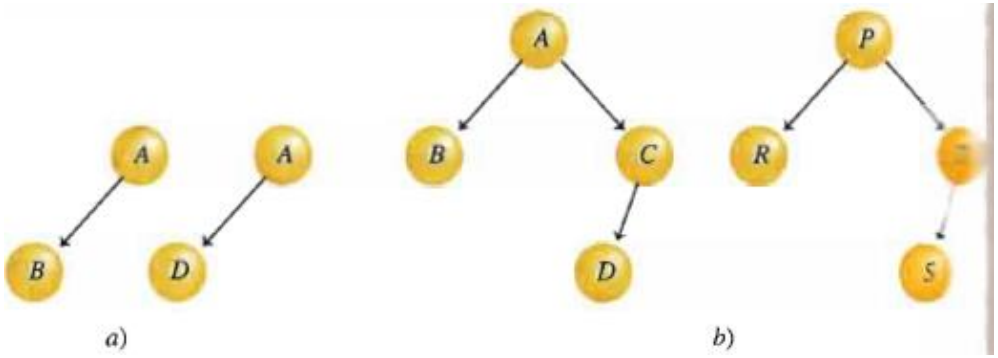
- (# nodos externos) = (# nodos internos) + 1
- (# nodos nivel i) $\leq 2^{i}$
- (# nodos externos) $\leq 2^{\text{altura}}$
- (altura) $\geq \log_2$ (# nodos externos)
- (altura) $\geq \log_2$ (# nodos) - 1
- (altura) $\leq$ (# nodos internos) - ((# nodos) - 1)/2



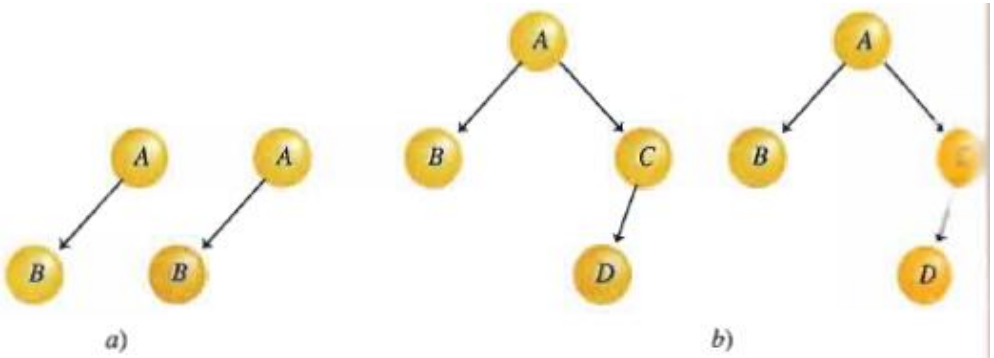
- Dos árboles binarios son distintos cuando sus estructuras son diferentes.



- Dos árboles binarios son similares cuando sus estructuras son idénticas.

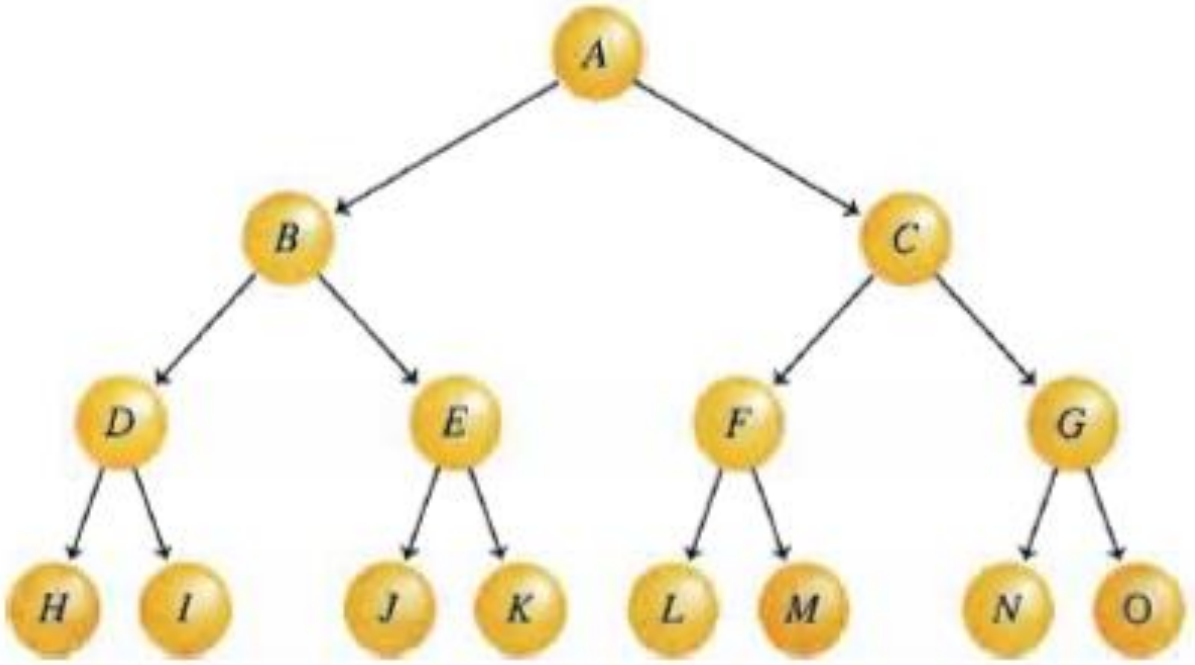


- Dos árboles binarios son equivalentes cuando son similares y contienen la misma información.



Árboles Binarios Completos

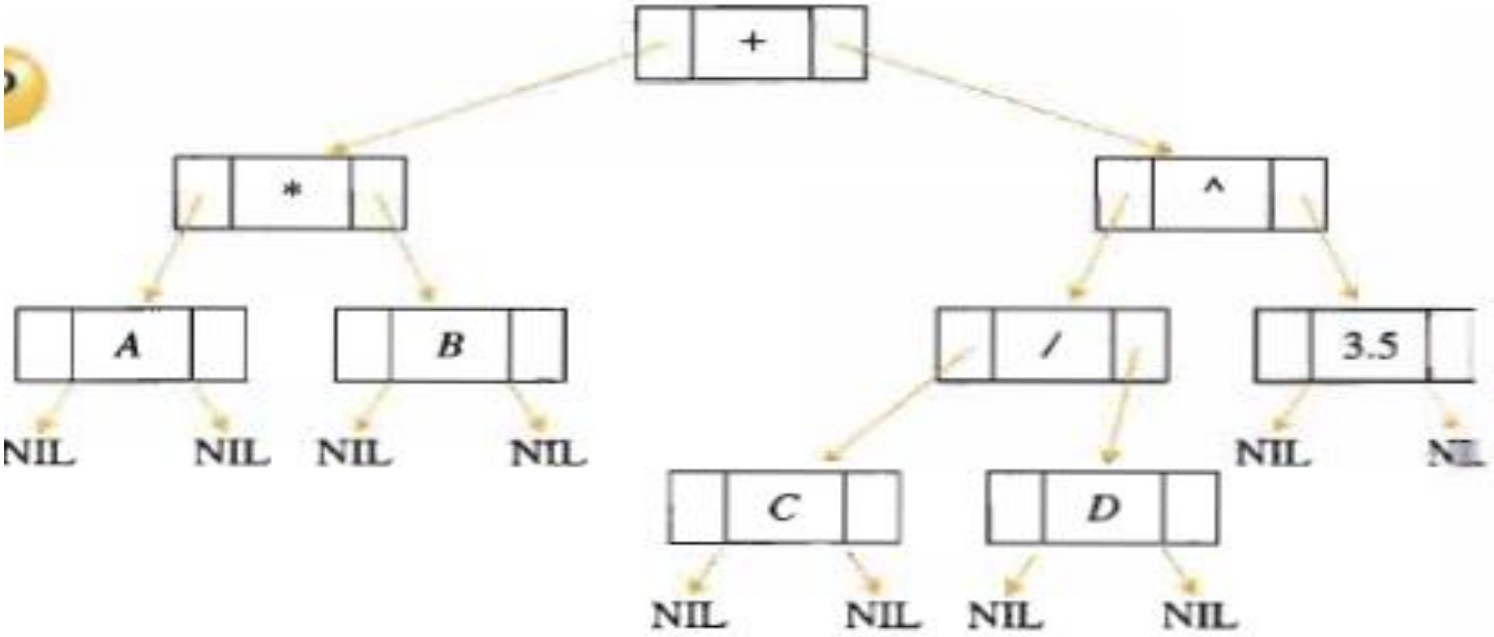
- Se define un Árbol Binario Completo(ABC) como un árbol en el que todos sus nodos, excepto los del último nivel, tienen 2 hijos: el subárbol izquierdo y el subárbol derecho.
- **Número de Nodos(ABC)=2^h-1**



Representación de árboles binarios en memoria.

En él se distinguen tres campos:

- **IZQ:** es el campo donde se almacena la dirección del subárbol izquierdo del nodo *T*.
- **INFO:** representa el campo donde se almacena la información del nodo. Normalmente en este campo y en el transcurso de este libro se almacenará un valor simple: número o carácter. Sin embargo, en la práctica es común almacenar en este campo cualquier tipo de dato.
- **DER:** es el campo donde se almacena la dirección del subárbol derecho del nodo *T*.

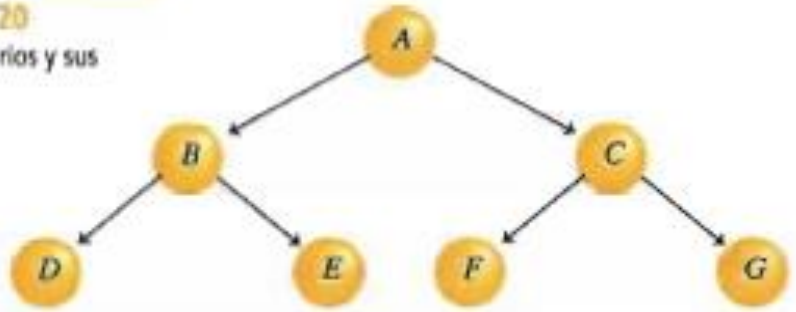


RECORRIDOS EN ARBOLES BINARIOS

- Una de las operaciones más importantes a realizar en un árbol binario es el recorrido de los mismos. Recorrer significa visitar los nodos del árbol en forma sistemática; de tal manera que todos los nodos del mismo sean visitados una sola vez. Existen tres formas diferentes de efectuar el recorrido y todas ellas de naturaleza recursiva, éstas son:
- **Recorrido en preorden**
  - Visitar la raíz
  - Recorrer el subárbol izquierdo
  - Recorrer el subárbol derecho
- **Recorrido en inorden**
  - Recorrer el subárbol izquierdo
  - Visitar la raíz
  - Recorrer el subárbol derecho
- **Recorrido en postorden**
  - Recorrer el subárbol izquierdo
  - Recorrer el subárbol derecho
  - Visitar la raíz
- El termino visitar puede ser reemplazado por escribir la información el nodo.

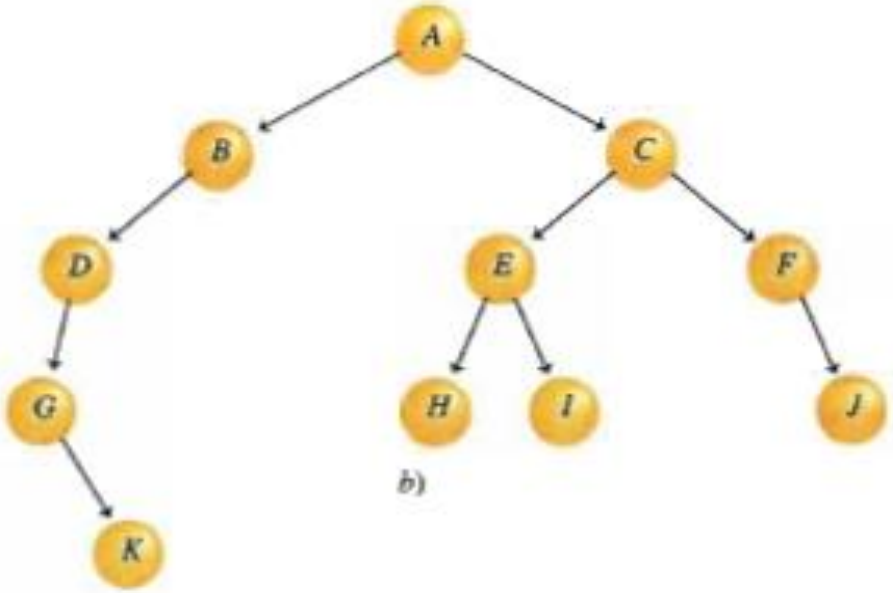
RECORRIDOS EN ARBOLES BINARIOS

FIGURA 6.20
Árboles binarios y sus recorridos.



a)

PREORDEN: A B D E C F G
INORDEN: D B E A F C G
POSORDEN: D E B F G C A



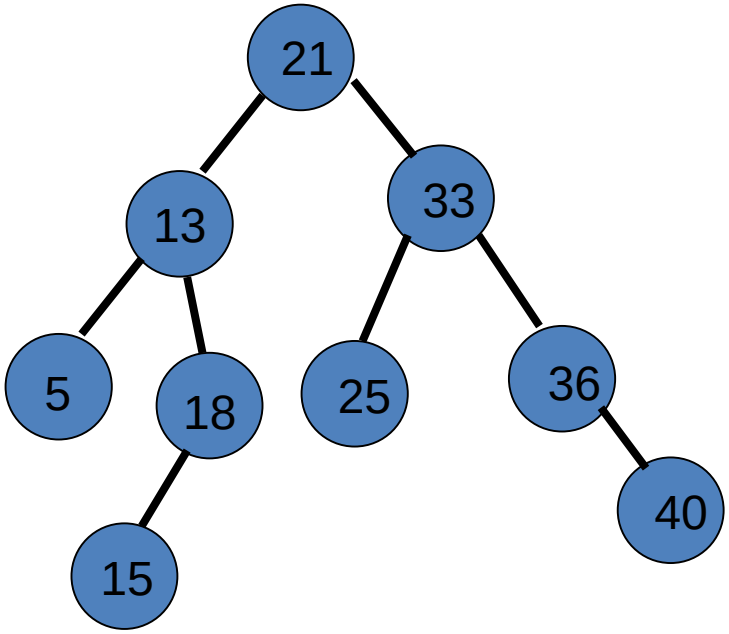
b)

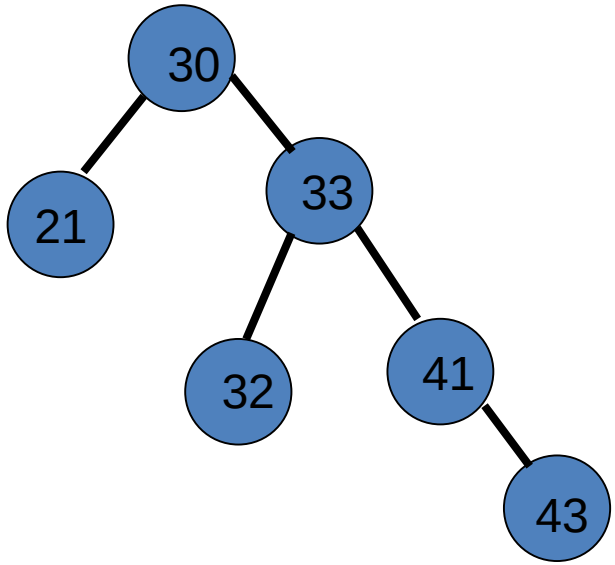
PREORDEN: A B D G K C E H I F
INORDEN: G K D B A H E I C F J
POSORDEN: K G D B H I E J F C A

Árbol Binario de Búsqueda (ABB)

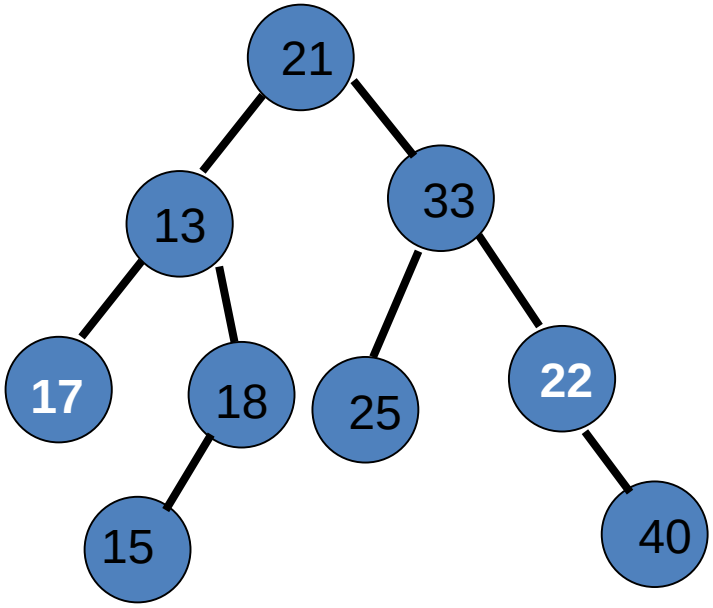
- Este tipo de árbol permite almacenar información ordenada.
- Reglas a cumplir:
  - Cada nodo del árbol puede tener 0, 1 ó 2 hijos.
  - Los descendientes izquierdos deben tener un valor menor al padre.
  - Los descendientes derechos deben tener un valor mayor al padre.

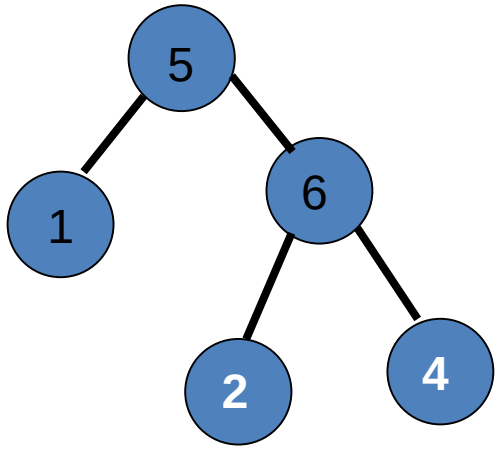
Ejemplos de ABB…





¿Por qué no son ABB?





Implementación de un ABB…

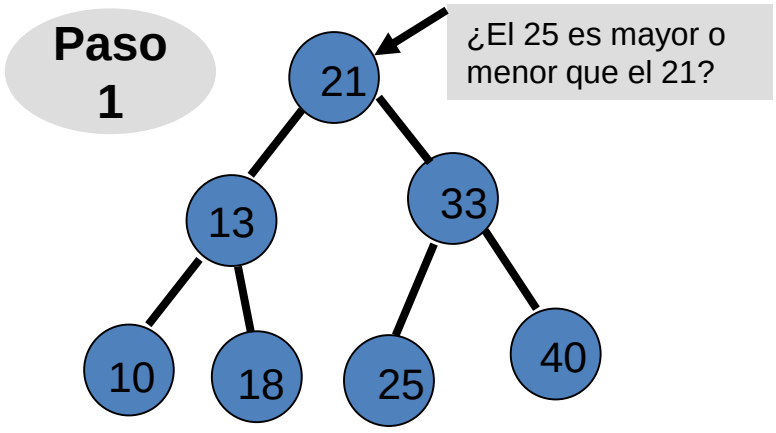
```
class NodoArbol
{
    public:
        int info;
        NodoArbol *izq, *der;
        NodoArbol( );
        NodoArbol(int dato);
};
NodoArbol(void) { izq = der = NULL; }
NodoArbol(int dato) { info = dato; izq = der = NULL; }
```

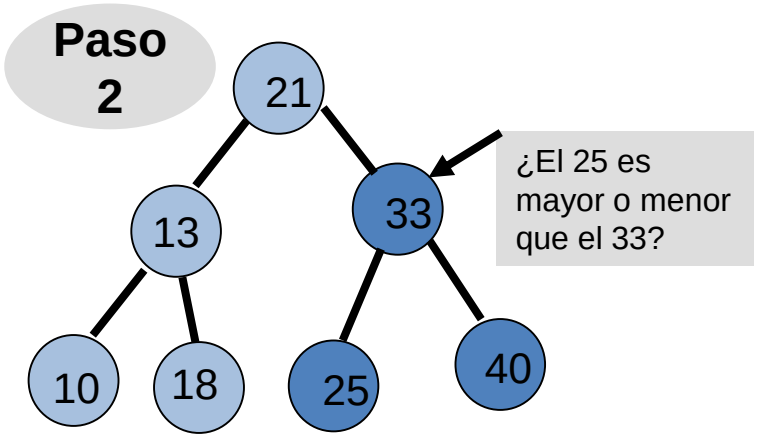
Continuación…

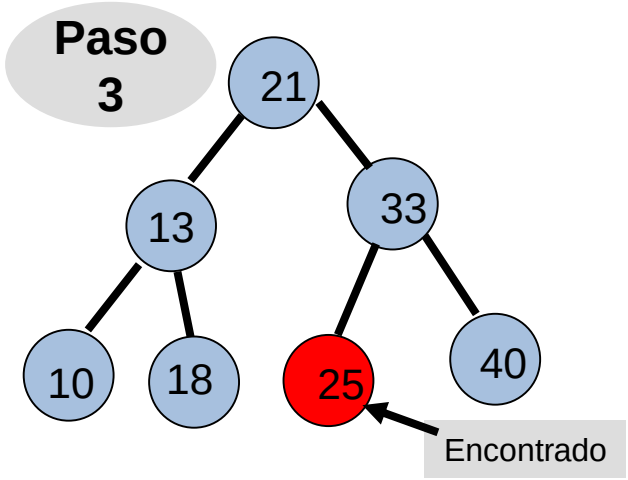
```
class ABB
{
    private:
        NodoArbol *raiz;
    public:
        ABB( );      // constructor
        ~ABB( );     // destructor
        //otros métodos
};
```

Proceso para buscar un nodo...

Buscar el 25







Implementación de la búsqueda

```
...
p=raiz;
while (p != NULL)
{ if (p->info == valor)
        return  p;
   else
      p=(p->info > valor? p->izq: p->der);
}
return NULL;
...
```

P contiene la dirección del nodo que tiene el valor buscado

No se encontró el valor por lo que se regresa un NULL

Equivalente a:

```
if ( p -> info > valor )
     p = p -> izq;
else p = p-> der;
```

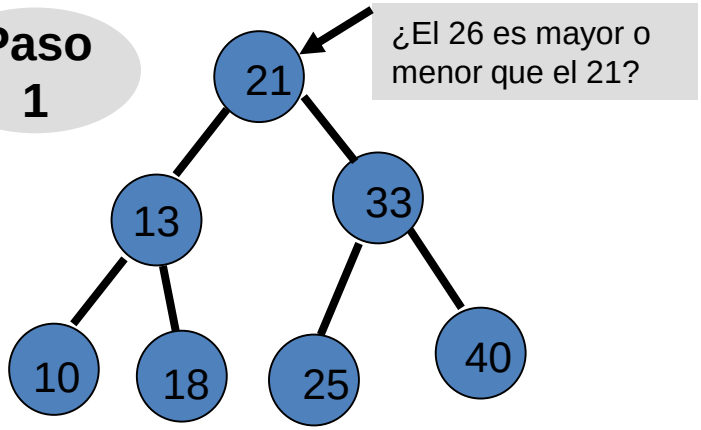
Proceso para agregar nodos...

- Reglas:
  - El valor a insertar no existe en el árbol.
  - El nuevo nodo será un Nodo Hoja del árbol.
- Procedimiento
  1. Buscar el Nodo Padre del nodo a agregar.
  2. Agregar el nodo hoja.

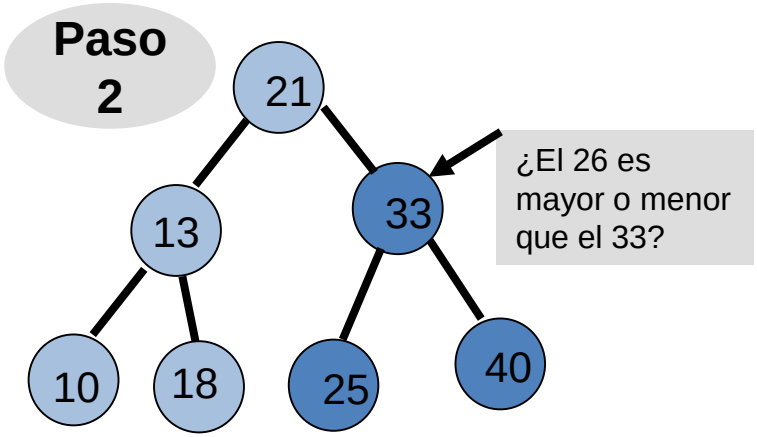
Ejemplo

Agregar el valor 26

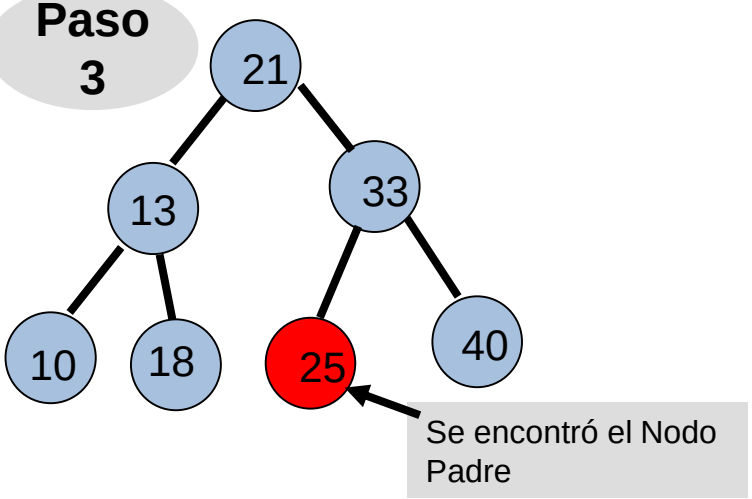
Paso 1



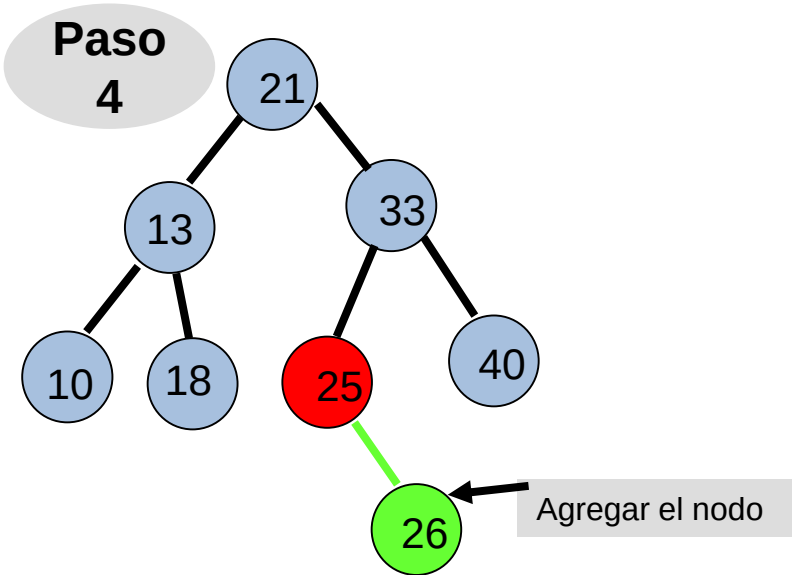
Paso 2



Paso 3

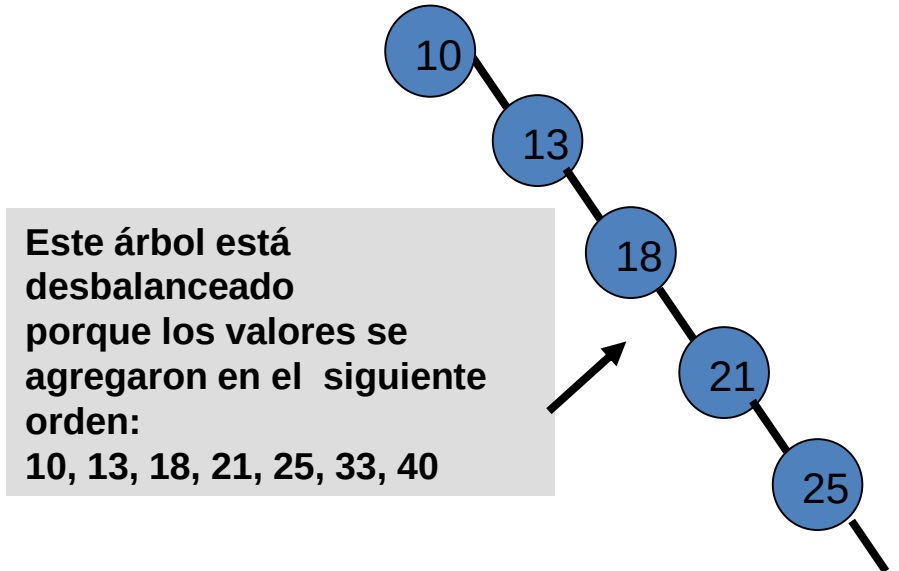


Paso 4



Comentarios importantes....

- El orden de inserción de los datos, determina la forma del ABB.
- ¿Qué pasará si se insertan los datos en forma ordenada?
- La forma del ABB determina la eficiencia del proceso de búsqueda.
- Entre menos altura tenga el ABB, más balanceado estará, y más eficiente será.



Implementación....

```
bool ABB::Insertar(int valor)
{
    NodoArbol *nuevo, *actual, *anterior;
    nuevo = new NodoArbol(valor);
    actual = raiz;
    anterior = NULL;
    while ( actual != NULL )
    {
        if ( valor == actual -> info ) return 0;
        anterior = actual;
        actual = (actual->info > valor ? actual->izq : actual->der);
    }
    if(anterior==NULL)
        raiz=nuevo;
    else {
             if ( anterior -> info > valor )
                 anterior -> izq = nuevo;
             else
                 anterior -> der = nuevo;
    }
    return 1;
}
```

Busca el Nodo Padre.
Al final, Anterior será el padre del nuevo nodo.

Agrega el nodo como nodo hoja.
Si Anterior es igual a NULL quiere decir que el árbol está vacío por lo que el nodo a agregar será la raíz.

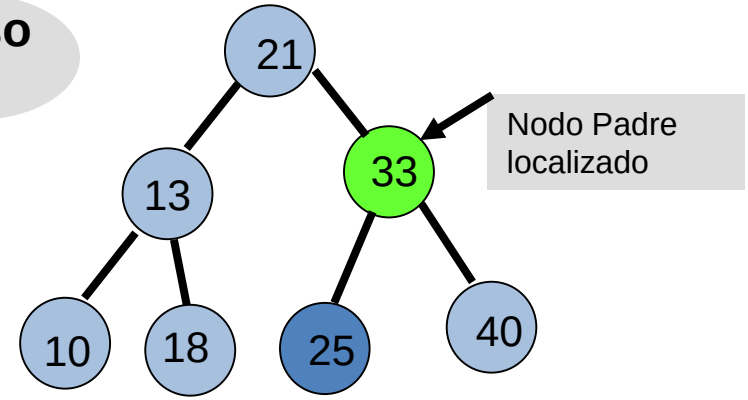
Proceso para eliminar un nodo

- Si el nodo a eliminar es un:
  - Nodo hoja
    - Buscar el Nodo Padre del nodo a borrar.
    - Desconectarlo.
    - Liberar el nodo.
  - Nodo con un hijo
    - Buscar el Nodo Padre del nodo a borrar.
    - Conectar el hijo con el padre del nodo a borrar.
    - Liberar el nodo.
  - Nodo con dos hijos
    - Localizar el nodo predecesor o sucesor del nodo a borrar.
    - Copiar la información.
    - Eliminar el predecesor o sucesor según sea el caso.

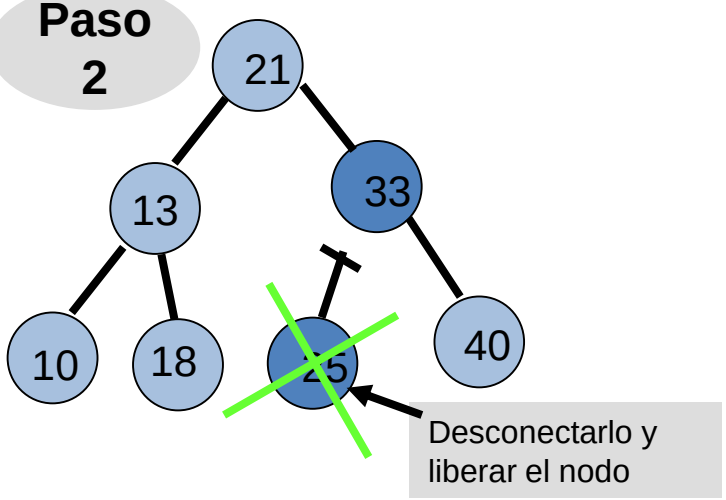
Caso: Eliminar Nodo hoja

Eliminar el valor 25

Paso 1



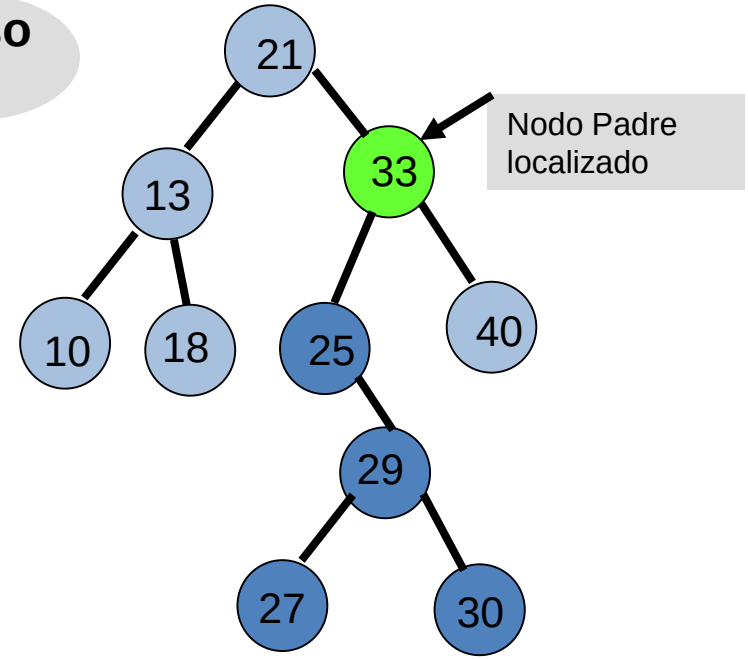
Paso 2



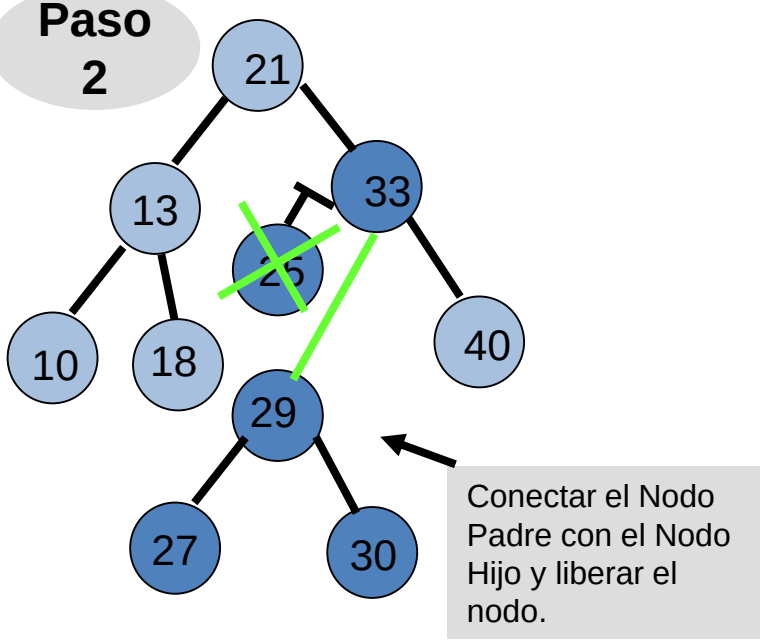
Caso: Eliminar Nodo con un hijo

Eliminar el valor 25

Paso 1



Paso 2

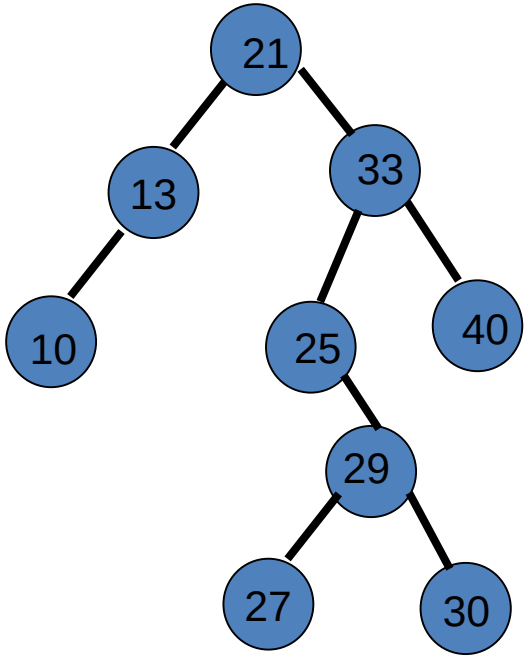


Caso: Eliminar nodo con dos hijos

1. Localizar el nodo predecesor o sucesor del nodo a borrar.
   - El PREDECESOR es “el Mayor de los Menores”.
   - El SUCESOR es “el Menor de los Mayores”.
   - Para la implementación es igual de eficiente programar la búsqueda del predecesor que del sucesor.
2. El valor del Predecedor (o sucesor) se copia al nodo a borrar.
3. Eliminar el nodo del predecesor (o sucesor según sea el caso).

Predecesor

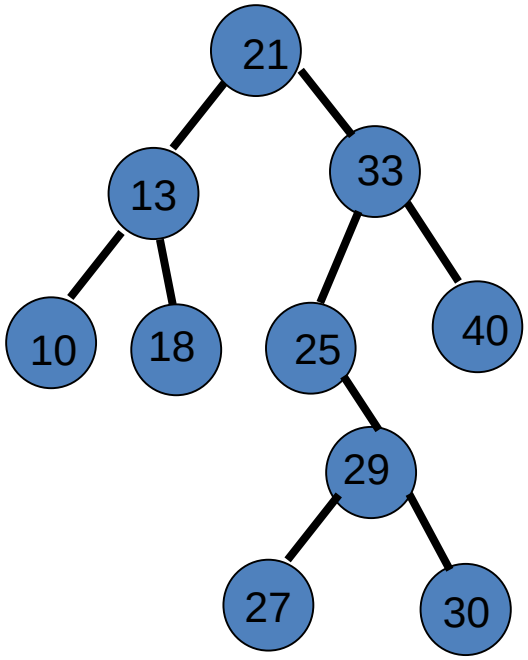
Uno a la IZQUIERDA y todo a la DERECHA



El predecesor de:	Es:
33	30
21	13
29	27

Sucesor

Uno a la DERECHA y todo a la IZQUIERDA



El sucesor de:	Es:
21	25
33	40
29	30

Implementación del....

PREDECESOR

```
P = actual -> izq;
while( p -> der != NULL)
   p=p->der;
return  p;
```

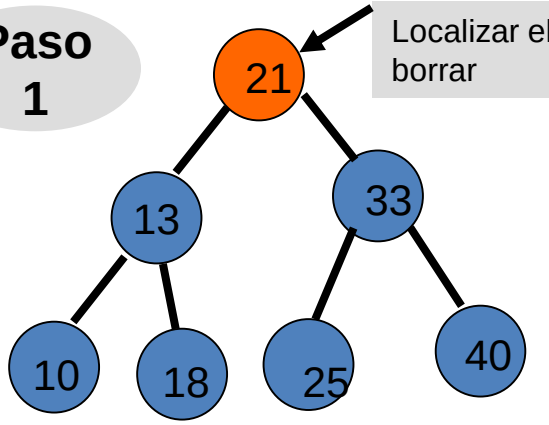
actual apunta al nodo a borrar

SUCESOR

```
P = actual -> der;
While (p -> izq != NULL )
   p=p->izq;
return p;
```

Caso: Eliminar Nodo con dos hijos

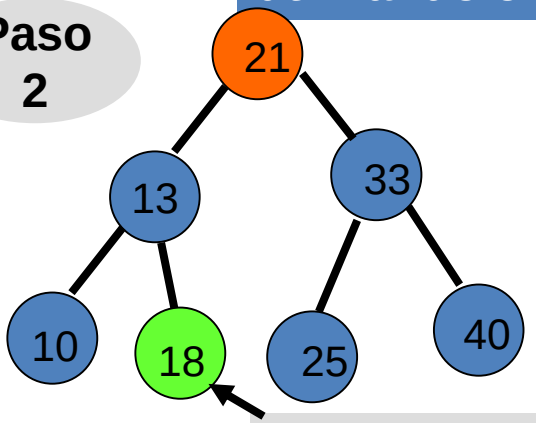
Paso 1



Localizar el valor a borrar

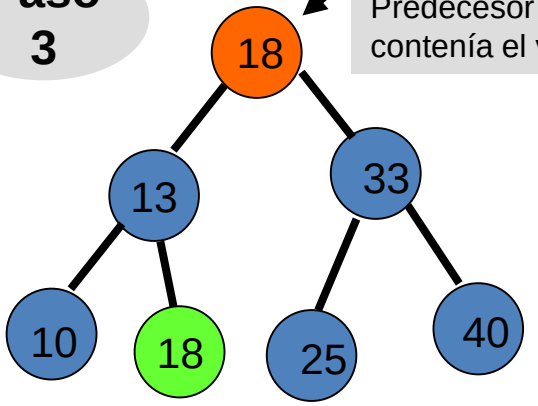
Eliminar el valor 21 utilizando el predecesor

Paso 2



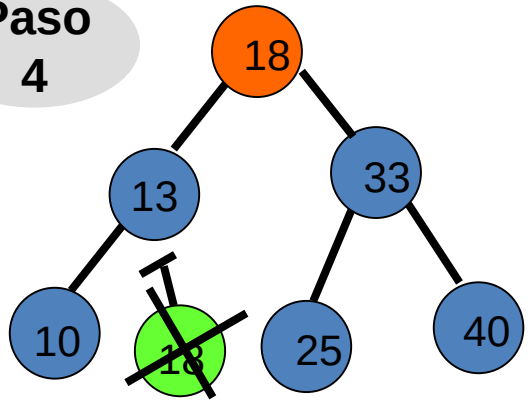
Localizar el Predecesor

Paso 3



Copiar el valor del Predecesor al nodo que contenía el valor a borrar

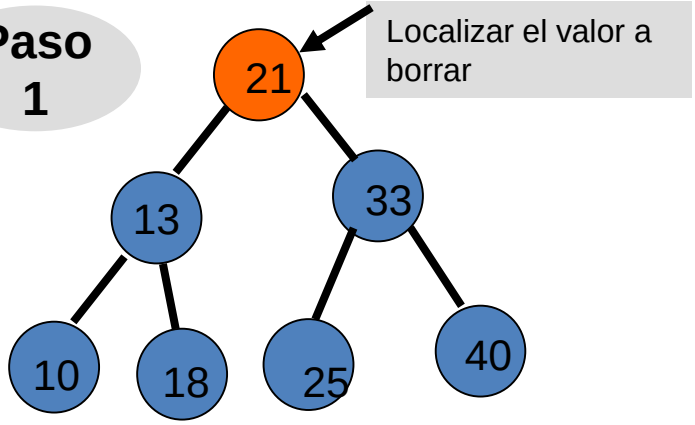
Paso 4



Desconectar y liberar el nodo del Predecesor

Caso: Eliminar Nodo con dos hijos

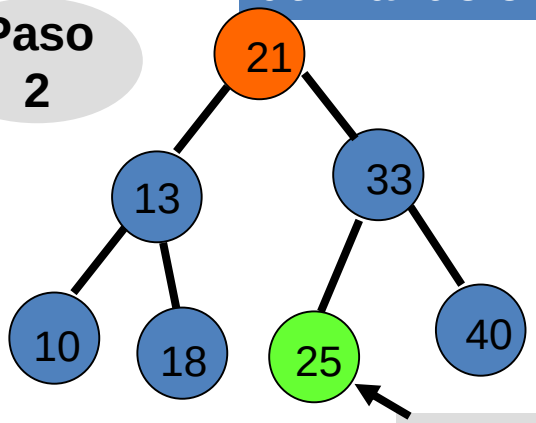
Paso 1



Localizar el valor a borrar

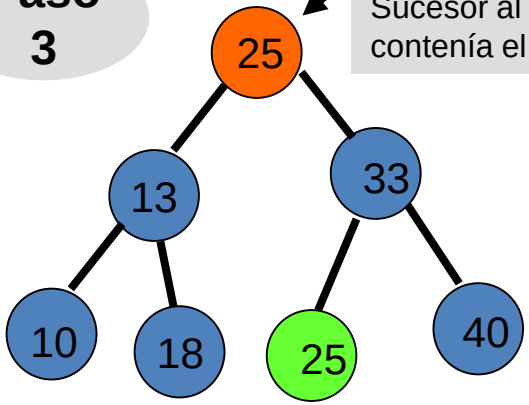
Eliminar el valor 21 utilizando el Sucesor

Paso 2



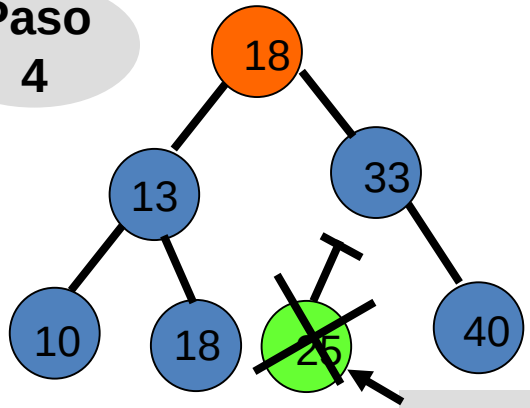
Localizar el Sucesor

Paso 3



Copiar el valor del Sucesor al nodo que contenía el valor a borrar

Paso 4



Desconectar y liberar el nodo del Sucesor