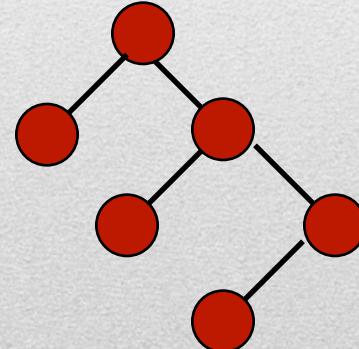
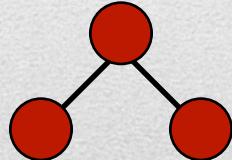
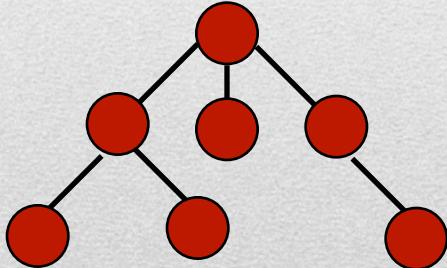


Árboles Binarios de Búsqueda (BST)

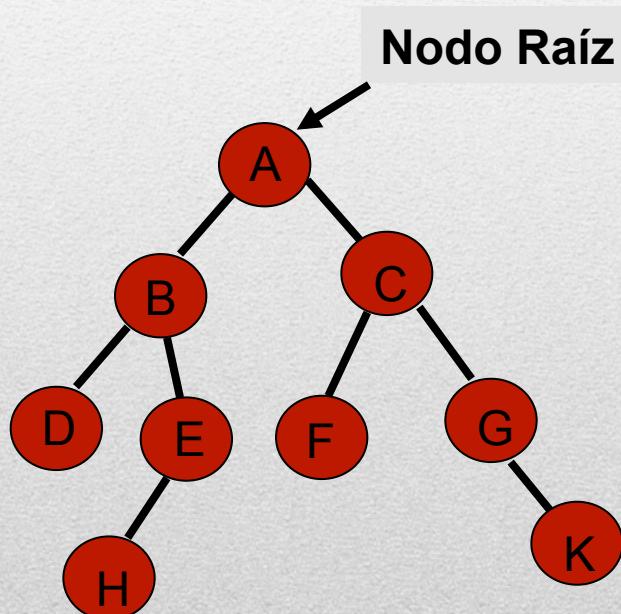
Ing. Luis Humberto González

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



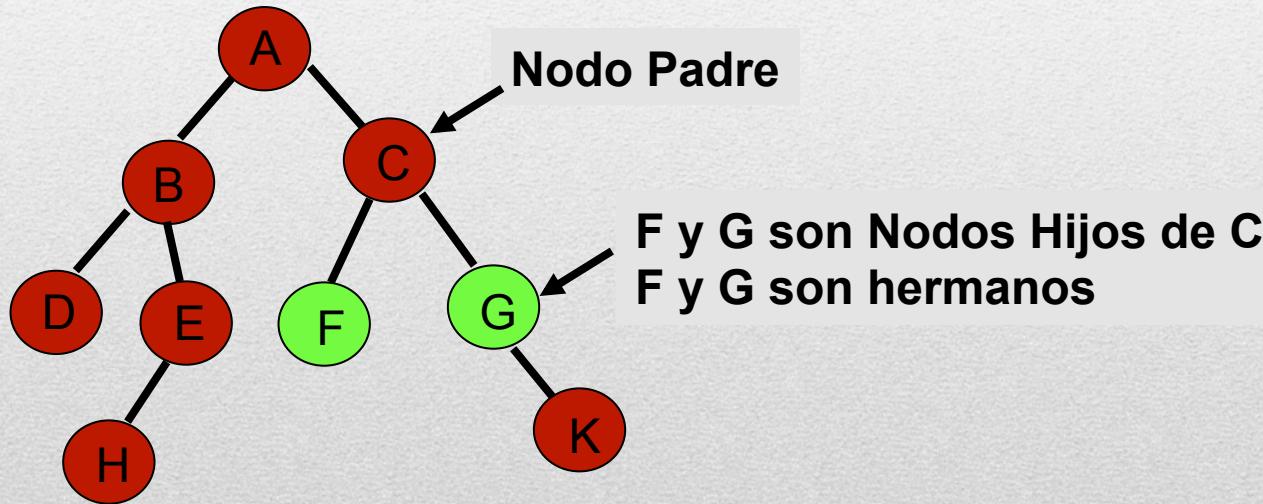
¿Qué es un Árbol?

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



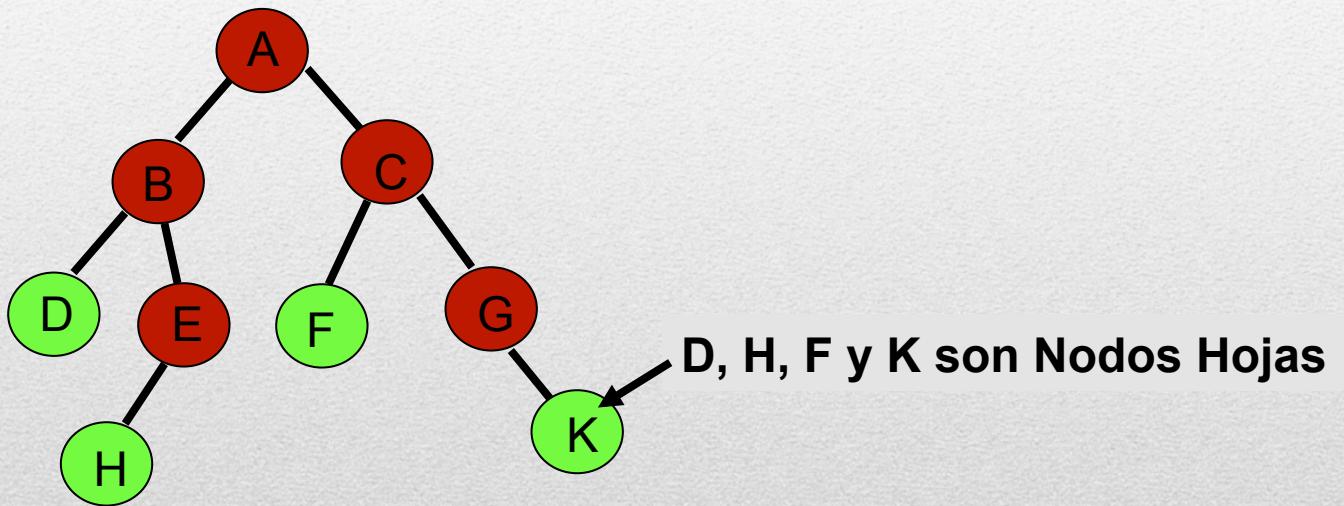
Terminología

- ✓ 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.



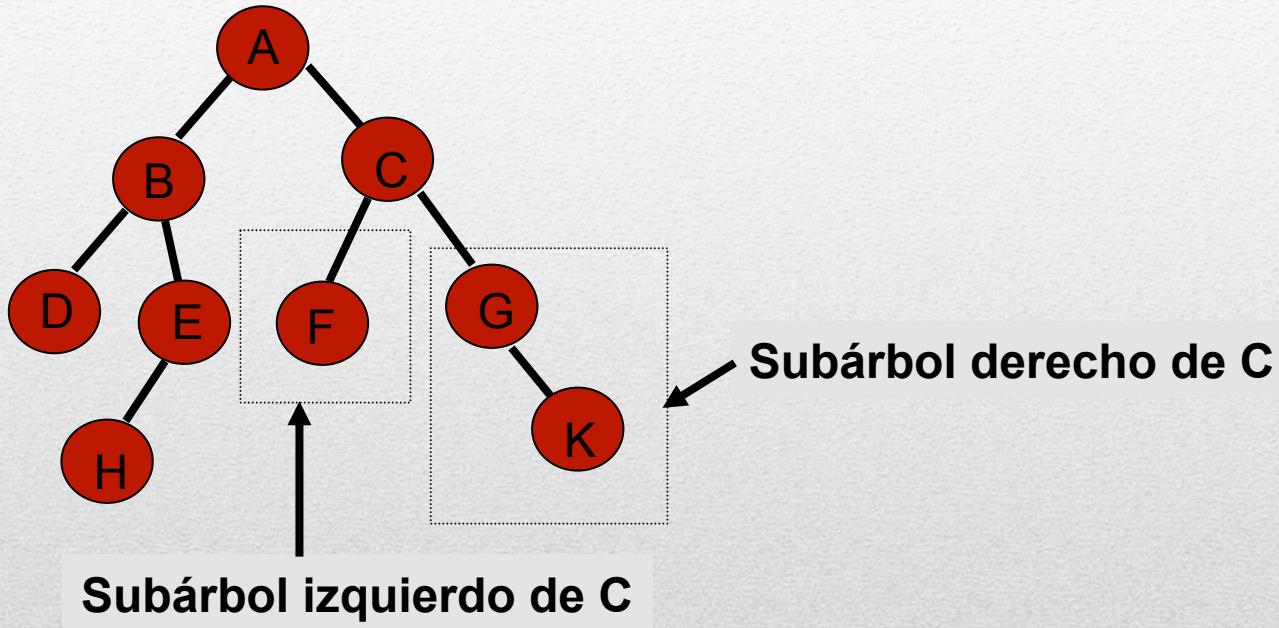
Más terminología

- ✓ Nodo Hoja: Aquel nodo que no tiene hijos.



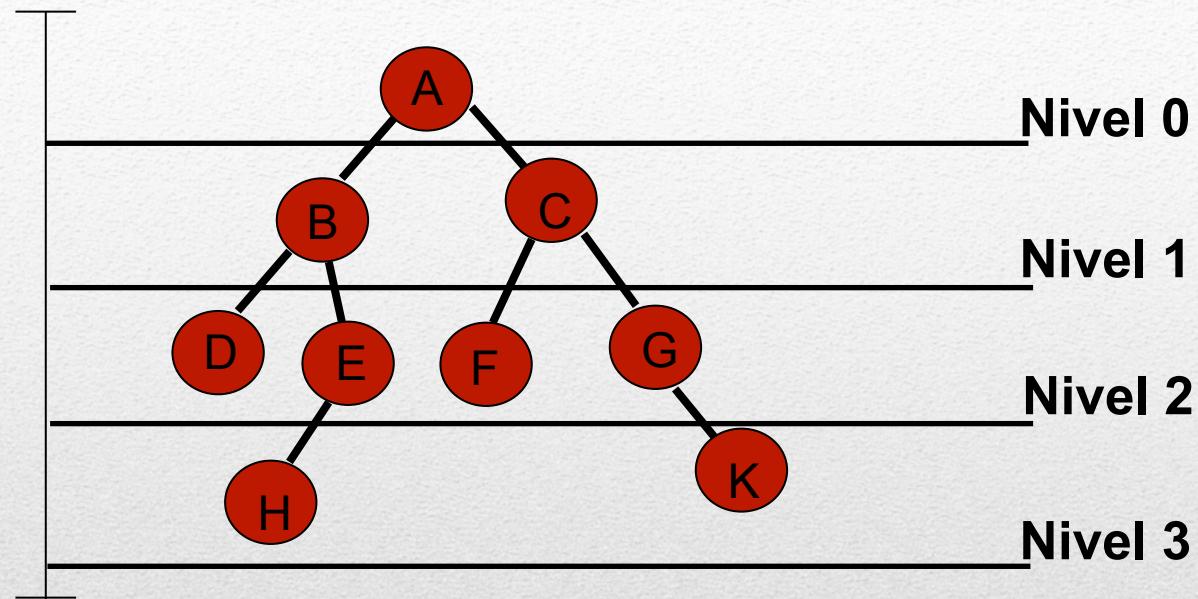
Más terminología

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



Más terminología

**Altura del
árbol = 4**



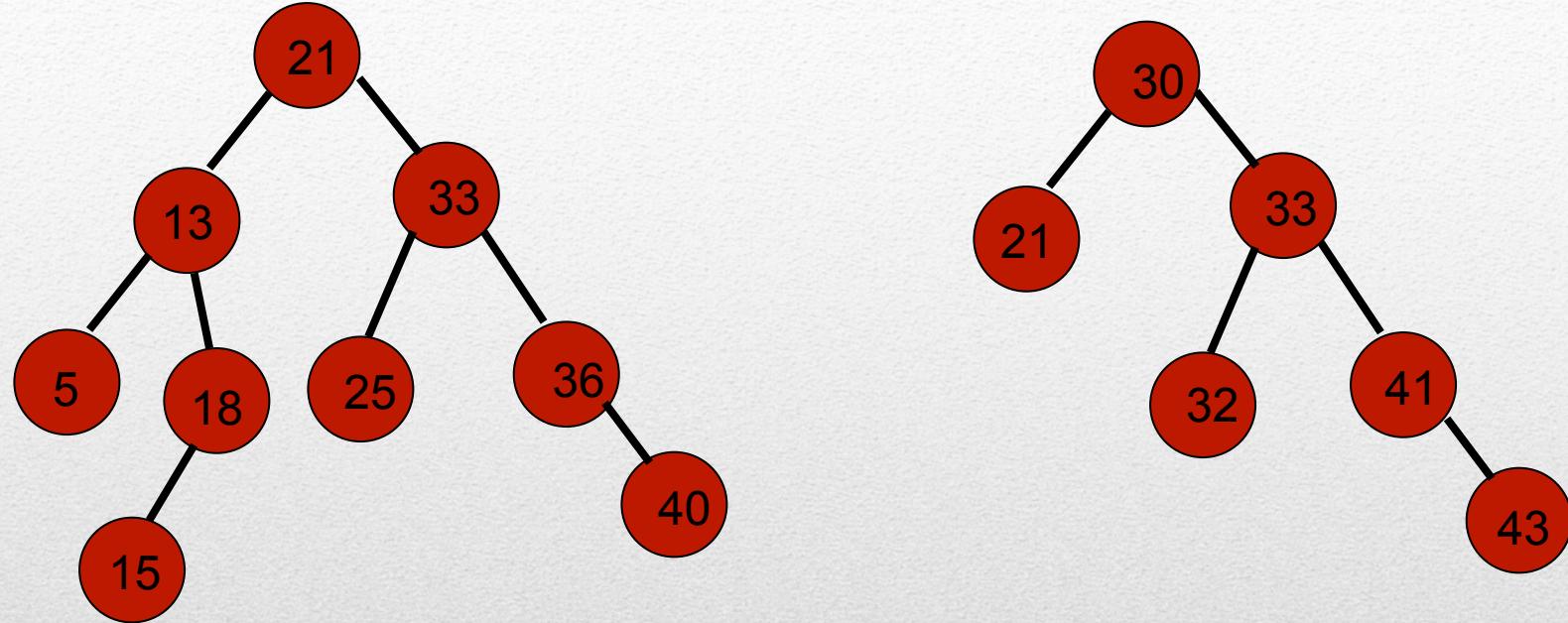
La Altura es la cantidad de niveles.

Altura y Niveles

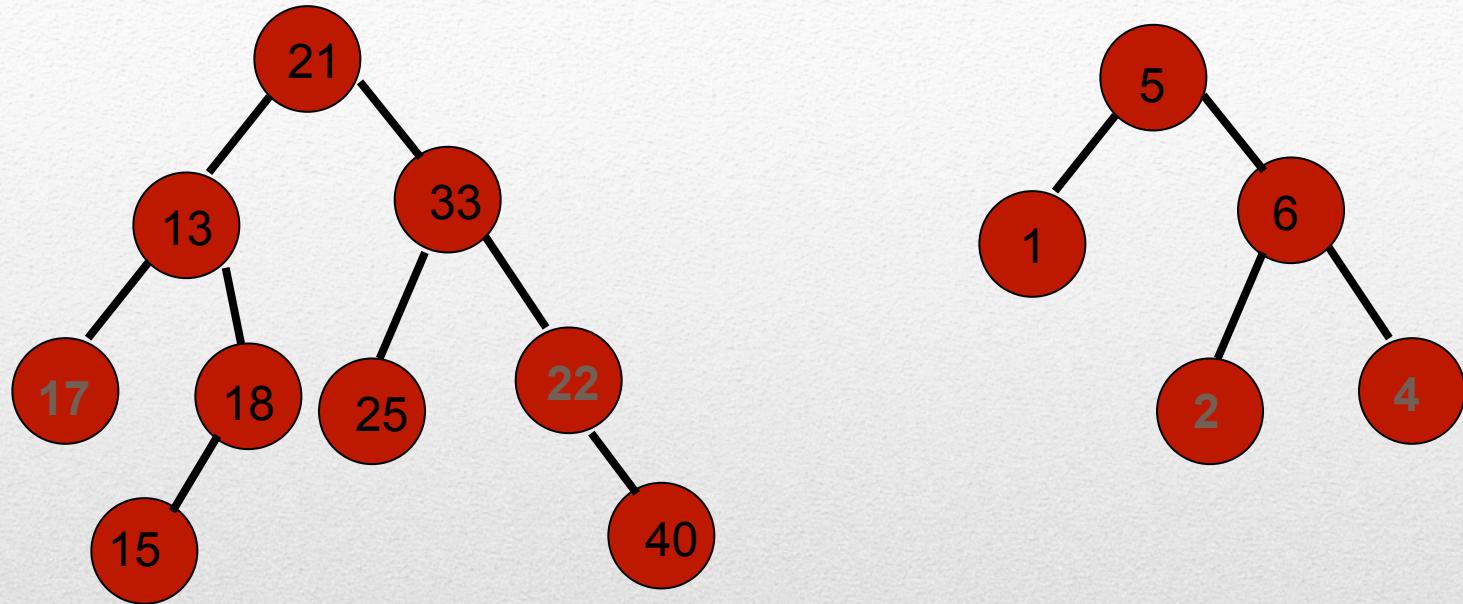
Este tipo de árbol permite almacenar información ordenada.
Reglas a cumplir:

- ✓ 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**.

Árbol Binario de Búsqueda (ABB)



Ejemplos de ABB...



¿Por qué **no** son ABB?

```
class NodeT
{
    private:
        int data;
        NodeT *left, *right;
    public:
        NodeT( );
        NodeT(int d);
}
NodeT() { left = right= NULL; }
NodeT(int d) { data = d; left = right= NULL; }
```

Implementación de un BST...

```
class BST
{
    private:
        NodeT *root;
    public:
        BST( );      // constructor
```

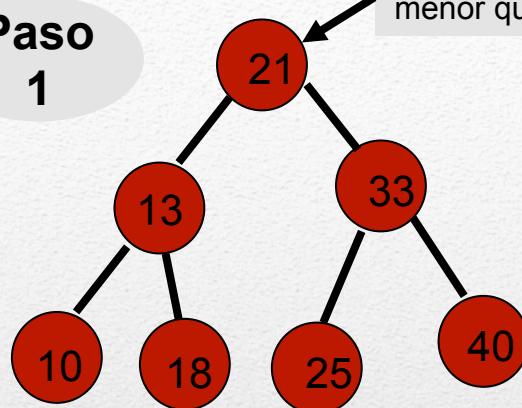
//otros métodos

}

Continuación...

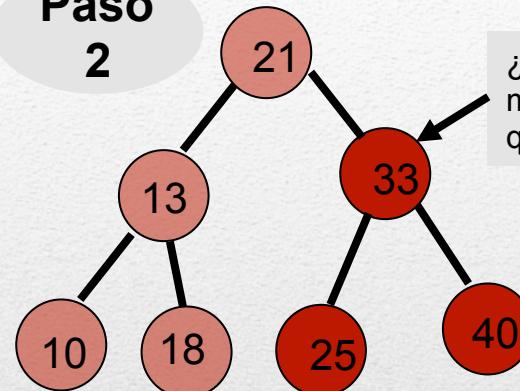
Buscar el 25

Paso
1



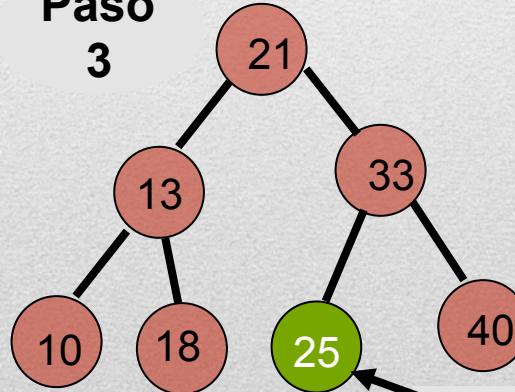
¿El 25 es mayor o menor que el 21?

Paso
2



¿El 25 es mayor o menor que el 33?

Paso
3



Encontrado

Proceso para buscar un nodo...

```
public bool BST::search(int valor){  
    p=root;  
    while (p != null)  
    { if (p.getData == valor)  
        return true; ← P contiene la dirección del nodo  
                      que tiene el valor buscado  
    else  
        p=(p.getData() > valor ? p.getLeft(): p.getRight()); ← Equivalente a:  
    }  
    return null; ← No se encontró el valor por lo que  
                  se regresa un null  
}
```

Equivalentе a:

```
if ( p.getData() > valor )  
    p = p.getLeft();  
else p = p.getRight();
```

Implementación de la búsqueda

Reglas:

- ✓ El valor a insertar no existe en el árbol.
- ✓ El nuevo nodo será un Nodo Hoja del árbol.

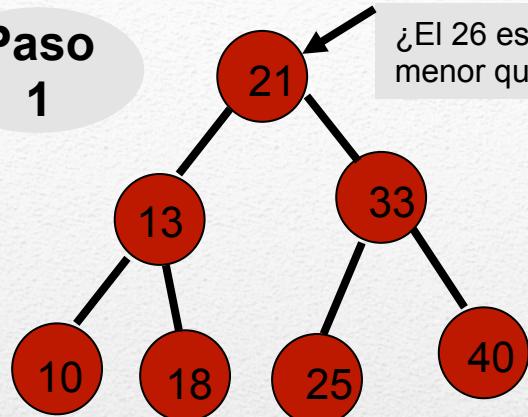
Procedimiento

1. Buscar el Nodo Padre del nodo a agregar.
2. Agregar el nodo.

Proceso para agregar nodos...

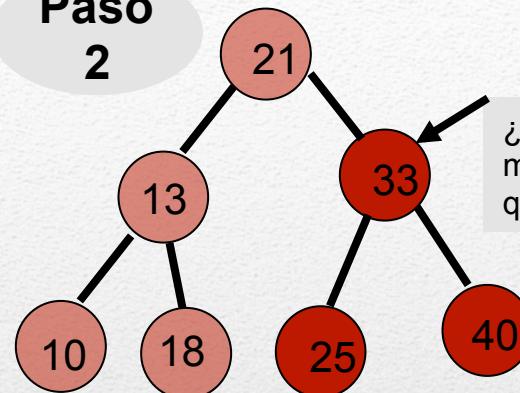
Agregar el valor 26

Paso
1



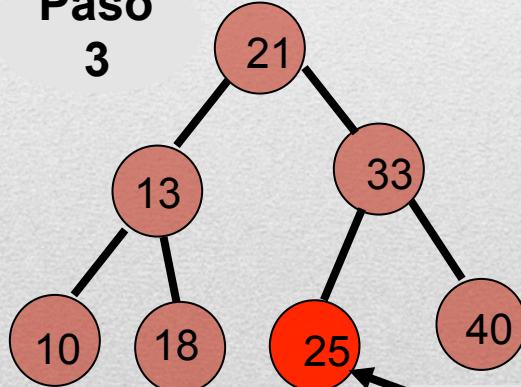
¿El 26 es mayor o menor que el 21?

Paso
2



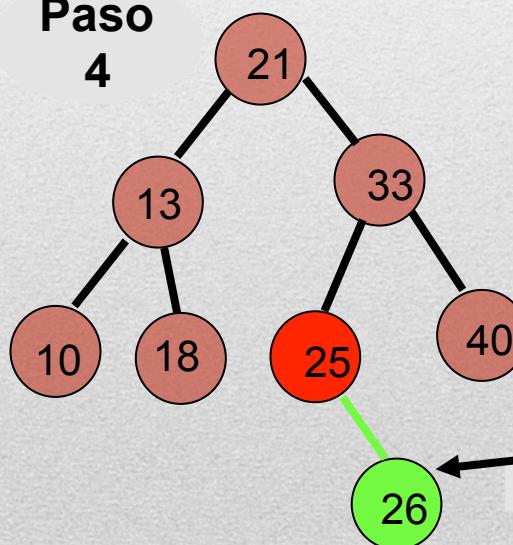
¿El 26 es
mayor o menor
que el 33?

Paso
3



Se encontró el Nodo
Padre

Paso
4

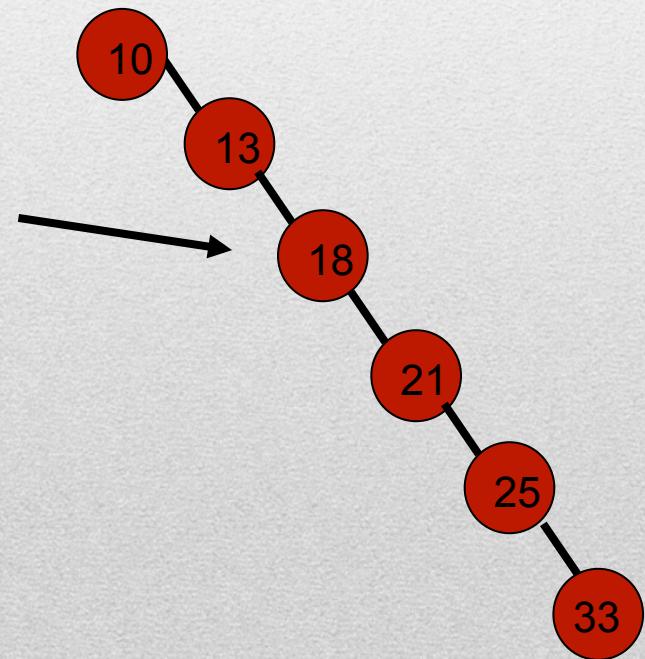


Agregar el nodo

Ejemplo

- ✓ 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.

Este árbol está desbalanceado porque los valores se agregaron en el siguiente orden:
10, 13, 18, 21, 25, 33



Comentarios importantes....

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.

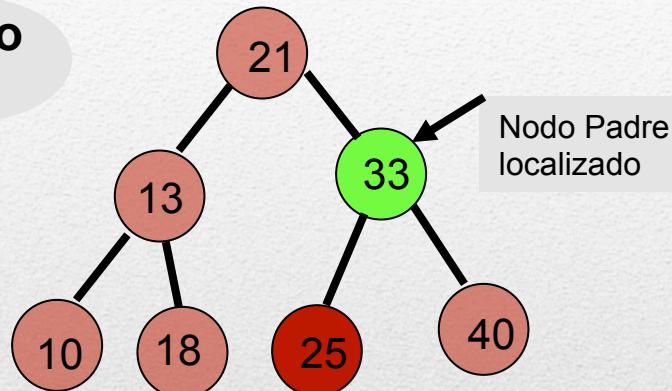
✓ Nodo con dos hijos

- Localizar el nodo predecesor o sucesor del nodo a borrar.
- Copiar la información.
- Eliminar el predecesor o sucesor según sea el caso.

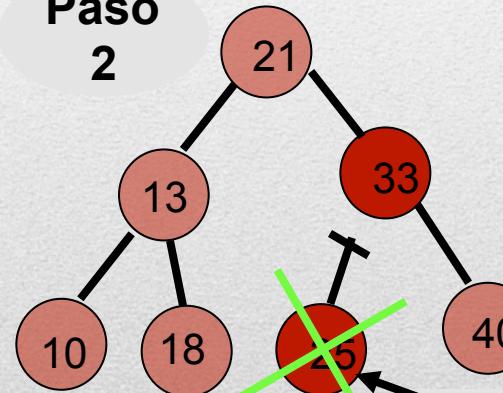
Proceso para eliminar un nodo

Eliminar el valor 25

**Paso
1**



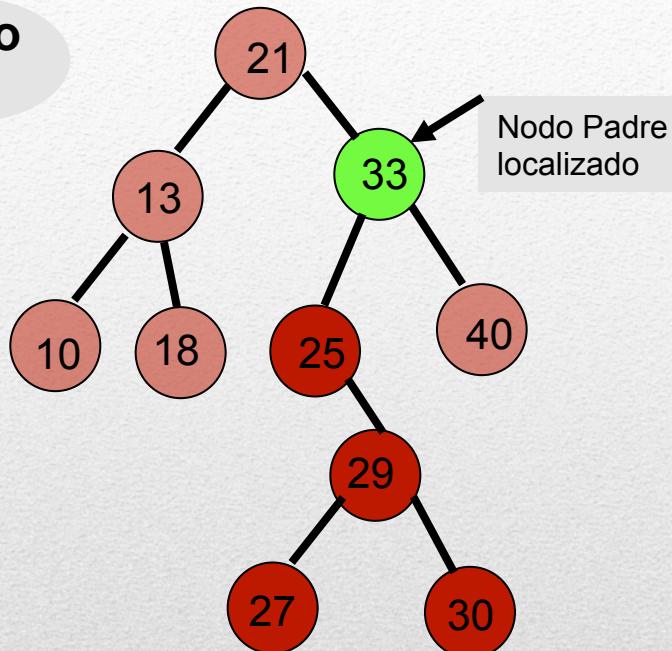
**Paso
2**



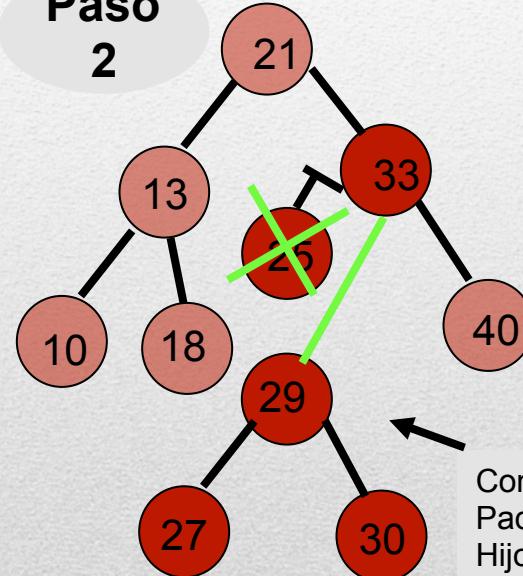
Caso: Eliminar Nodo hoja

Eliminar el valor 25

Paso
1



Paso
2



Conegar el Nodo
Padre con el Nodo
Hijo y liberar el
nodo.

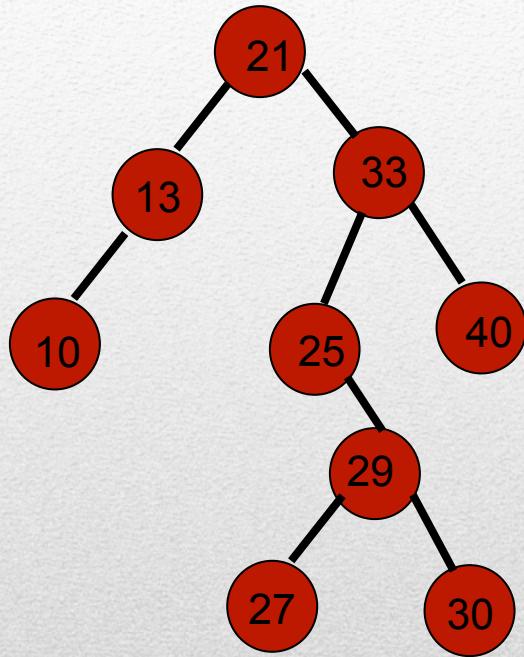
CASO: Eliminar Nodo con un hijo

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).

CASO: Eliminar nodo con dos hijos

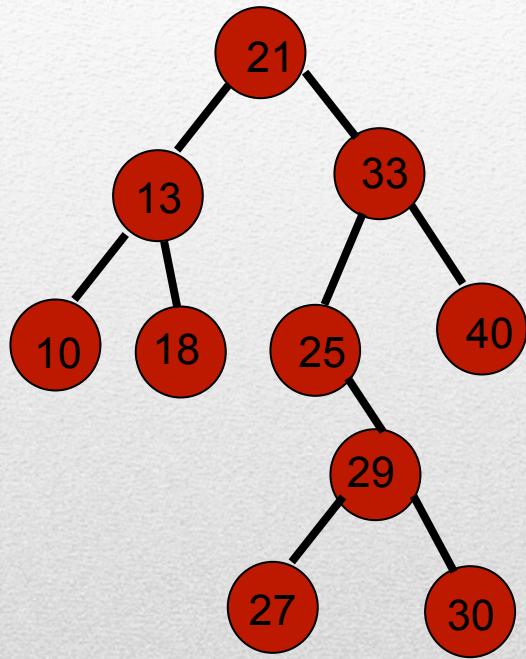
Uno a la IZQUIERDA y todo a la DERECHA



<i>El predecesor de:</i>	<i>es:</i>
33	30
21	13
29	27

Predcesor

Uno a la DERECHA y todo a la IZQUIERDA

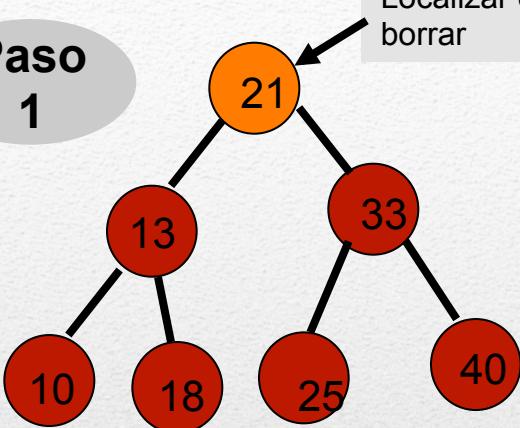


<i>El sucesor de:</i>	<i>es:</i>
21	25
33	40
29	30

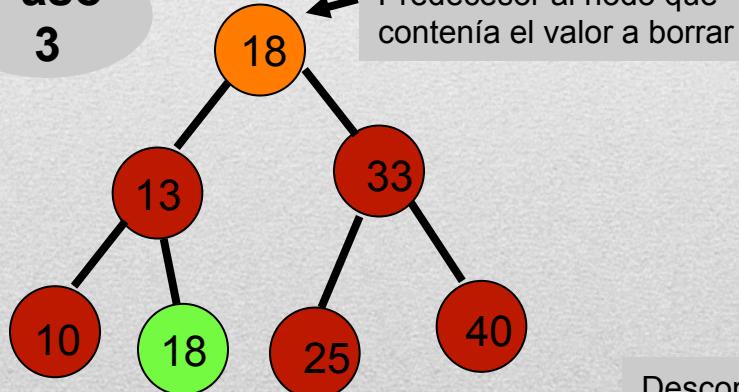
Sucesor

Eliminar el valor 21 utilizando el predecesor

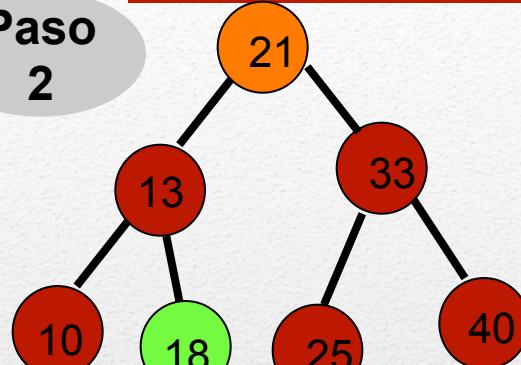
Paso 1



Paso 3

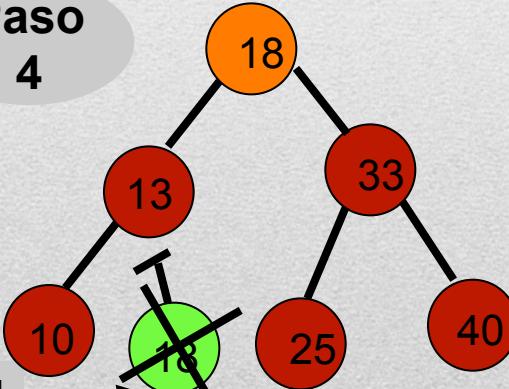


Paso 2



Paso 4

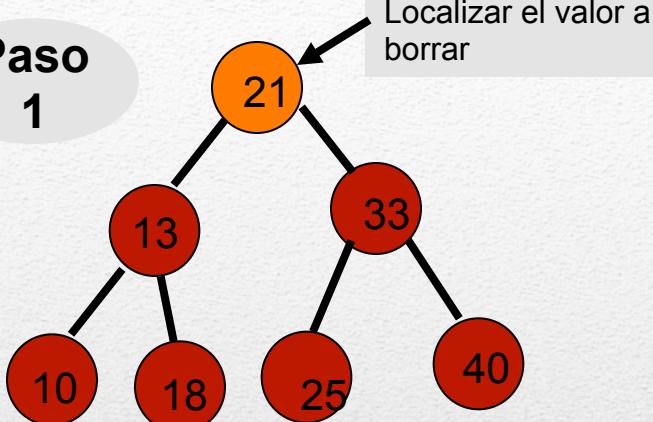
Desconectar y liberar el nodo del Predecesor



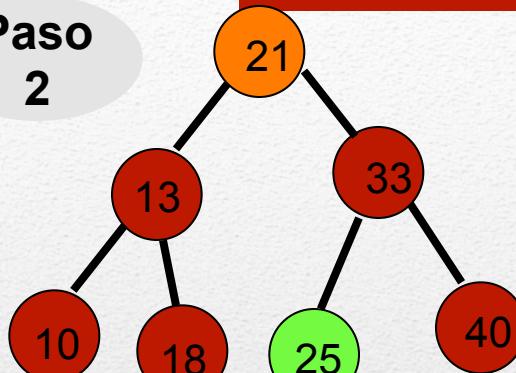
Caso: Eliminar Nodo con dos hijos

Eliminar el valor 21 utilizando el Sucesor

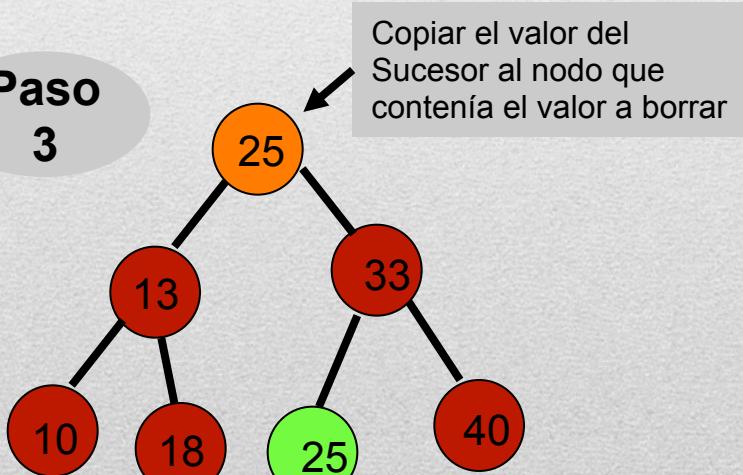
Paso 1



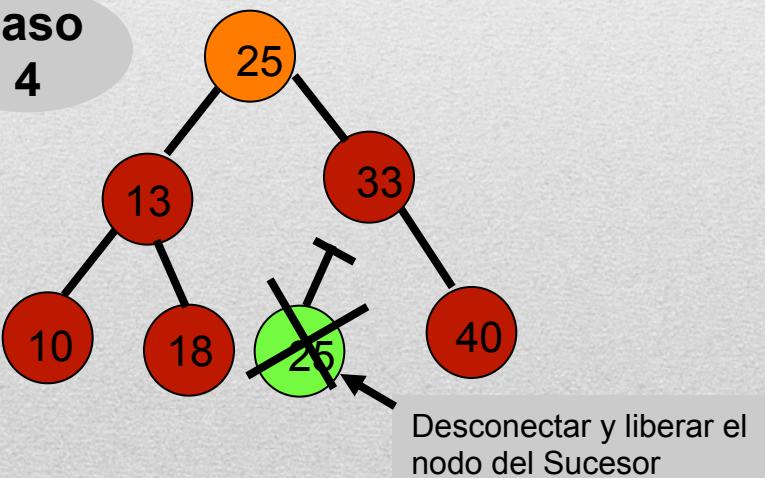
Paso 2



Paso 3



Paso 4



CASO: Eliminar Nodo con dos hijos