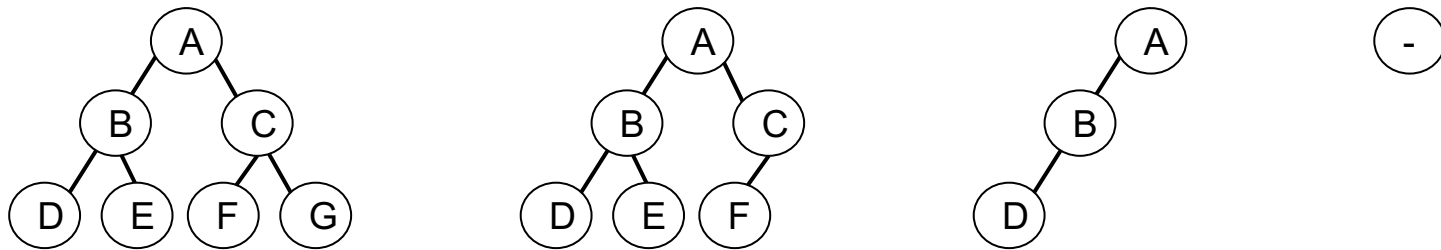


# Árbol Binario

# Características

- Conceptos: camino, profundidad, altura
- Estructura

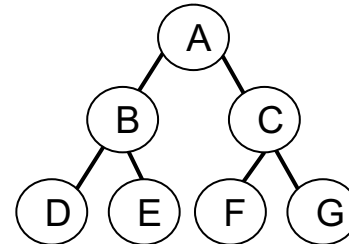


- No hay relación entre la altura y la cantidad de nodos

...**A MENOS** que se indique alguna característica,

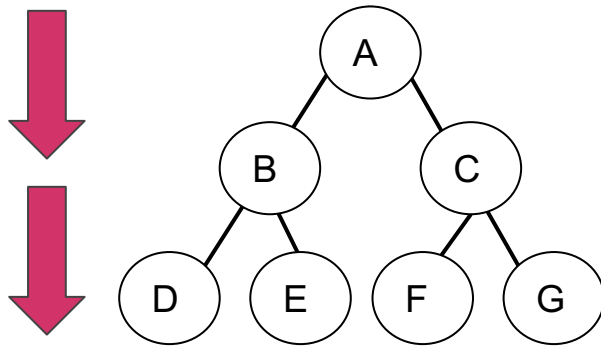
ej. que el árbol sea “**lleno**”

$$\# \text{ total de nodos} = 2^{h+1} - 1$$

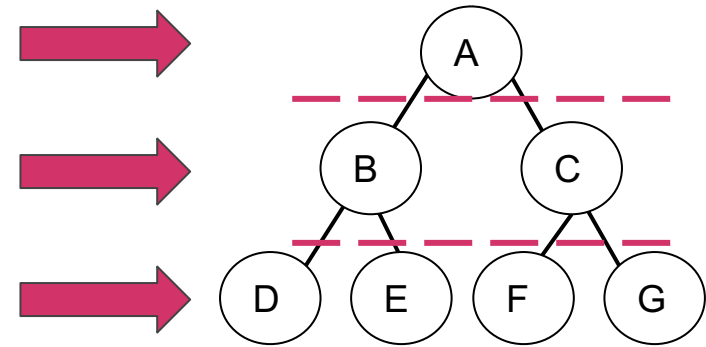


# ¿Cómo se recorre un árbol?

**En profundidad**

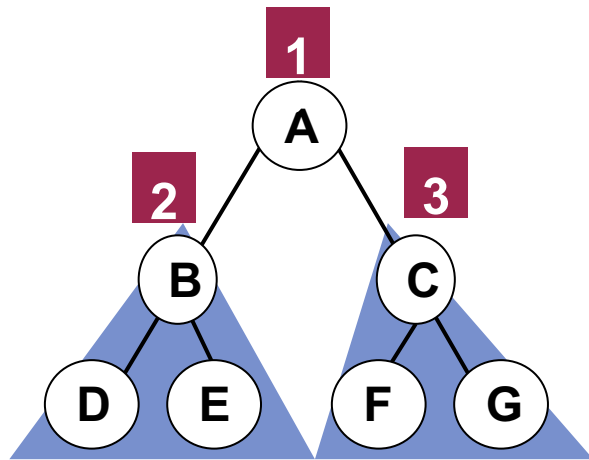


**Por niveles**



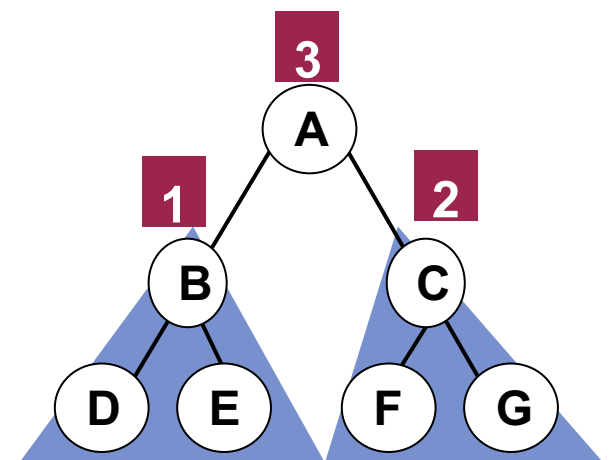
# Recorridos en profundidad

**pre-orden**



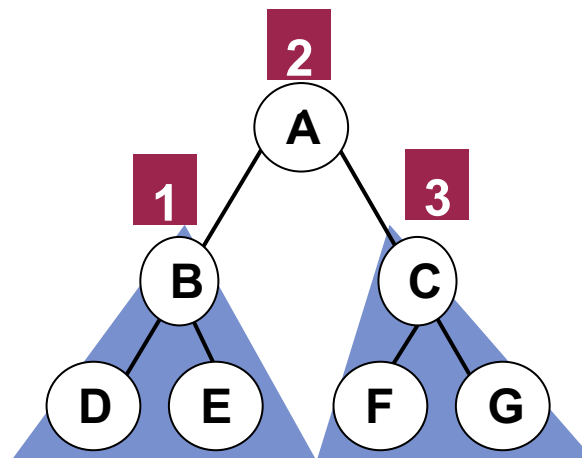
**A B D E C F G**

**post-orden**



**D E B F G C A**

**in-orden**



**D B E A F C G**

# Implementación en Java

| 🕒 BinaryTree<T>  |
|--|
| <ul style="list-style-type: none"><li>❑ data: T</li><li>❑ leftChild: BinaryTree&lt;T&gt;</li><li>❑ rightChild: BinaryTree&lt;T&gt;</li></ul>   |
| <ul style="list-style-type: none"><li>● BinaryTree(): void</li><li>● BinaryTree(T): void</li><li>● getData(): T</li><li>● setData(T): void</li><li>● getLeftChild(): BinaryTree&lt;T&gt;</li><li>● getRightChild(): BinaryTree&lt;T&gt;</li><li>● addLeftChild(BinaryTree&lt;T&gt;): void</li><li>● addRightChild(BinaryTree&lt;T&gt;): void</li><li>● removeLeftChild(): void</li><li>● removeRightChild(): void</li><li>● isEmpty(): boolean</li><li>● isLeaf(): boolean</li><li>● hasLeftChild(): boolean</li><li>● hasRightChild(): boolean</li><li>● toString(): String</li><li>● contarHojas(): int</li><li>● espejo(): BinaryTree&lt;T&gt;</li><li>● entreNiveles(int, int): void</li></ul> |

# Recorrido pre-orden - versión 1

```
public void imprimirPreOrder(ArbolBinario aBinario)
{
    System.out.println(aBinario.getData());
    imprimirPreOrder(aBinario.getLeftChild());
    imprimirPreOrder(aBinario.getRightChild());
}
```



¿es correcta la solución?

## Recorrido pre-orden - versión 2

```
public void imprimirPreOrder(ArbolBinario aBinario)
{
    System.out.println(aBinario.getData());

    if (aBinario.hasLeftChild())
        imprimirPreOrder(aBinario.getLeftChild());

    if (aBinario.hasRightChild())
        imprimirPreOrder(aBinario.getRightChild());
}
```



¿es correcta la solución?

## Recorrido pre-orden - versión 3

```
public void imprimirPreOrder(ArbolBinario aBinario) {  
  
    if (aBinario!=null && !aBinario.isEmpty()) {  
        System.out.println(aBinario.getData());  
  
        if (aBinario.hasLeftChild())  
            imprimirPreOrder(aBinario.getLeftChild());  
  
        if (aBinario.hasRightChild())  
            imprimirPreOrder(aBinario.getRightChild());  
        ;  
    }  
  
}
```



¿es correcta la solución?



# Ejercicio

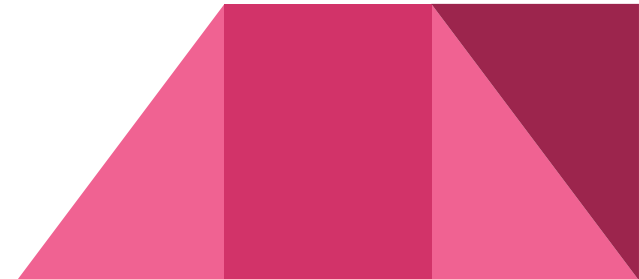
Dado un árbol que contiene valores enteros, **devolver la suma** de todos los elementos.



## Discusión:

¿cómo debo recorrer el árbol?

¿existe una solución más eficiente que otra?



# Solución

```
public class TestAB {  
  
    private int sumar(ArbolBinario<Integer> aBinario) {  
        int suma = 0;  
        if (aBinario!=null && !aBinario.isEmpty()) {  
            suma = aBinario.getData();  
            if (aBinario.hasLeftChild())  
                suma += this.sumar(aBinario.getLeftChild());  
            if (aBinario.hasRightChild())  
                suma += this.sumar(aBinario.getRightChild());  
        }  
        return suma;  
    }  
}
```

# Solución

```
public class TestAB {  
  
    private int sumar(ArbolBinario<Integer> aBinario) {  
        if (aBinario == null || aBinario.isEmpty())  
            return 0;  
        else {  
            if (aBinario.isLeaf())  
                return aBinario.getData();  
            else  
                return aBinario.getData() +  
                       this.sumar(aBinario.getLeftChild()) +  
                       this.sumar(aBinario.getRightChild());  
        }  
    }  
}
```

