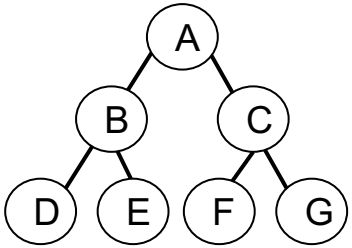
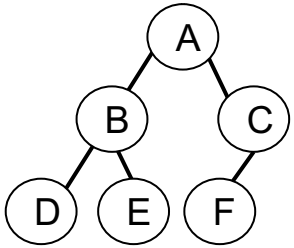
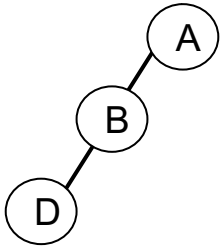
Árbol Binario

Características

- Conceptos: camino, profundidad, altura
- Estructura





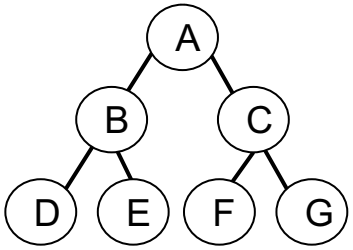


- No hay relación entre la altura y la cantidad de nodos

  ...A MENOS que se indique alguna característica,

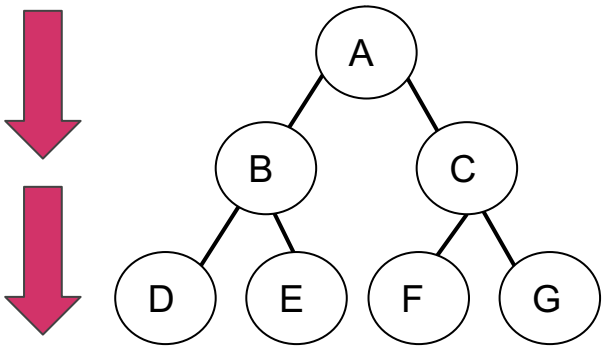
  ej. que el árbol sea "**lleno**"

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

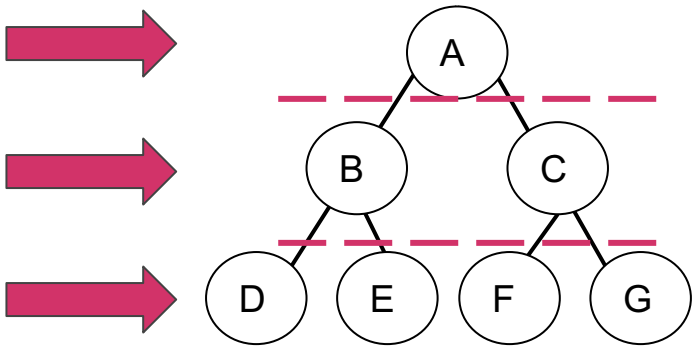


¿Cómo se recorre un árbol?

En profundidad

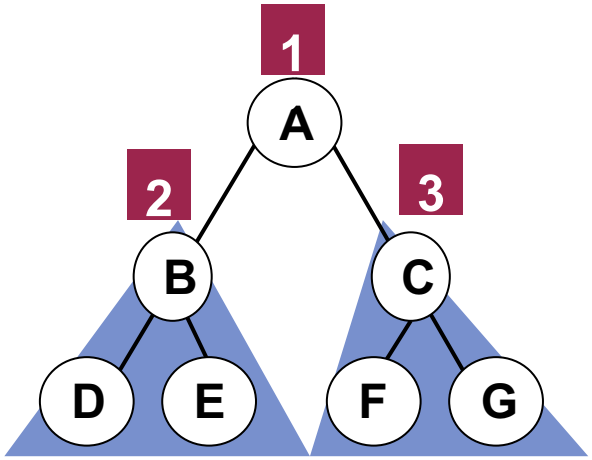


Por niveles



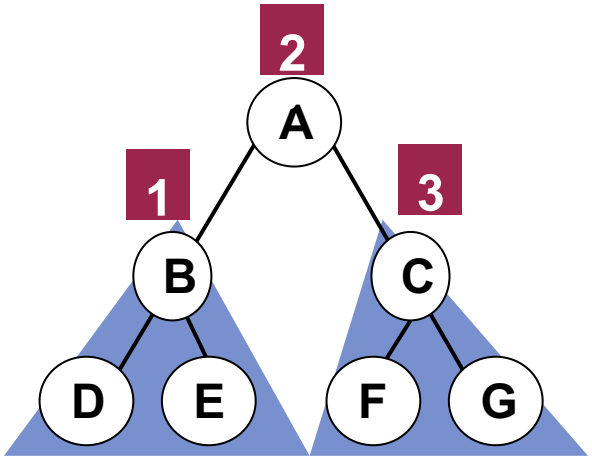
Recorridos en profundidad

pre-orden



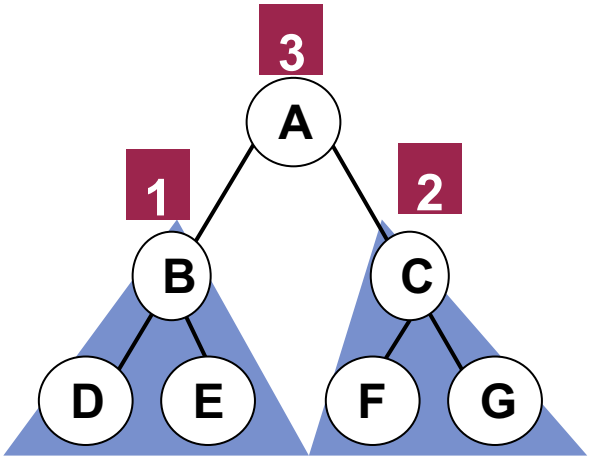
A B D E C F G

in-orden



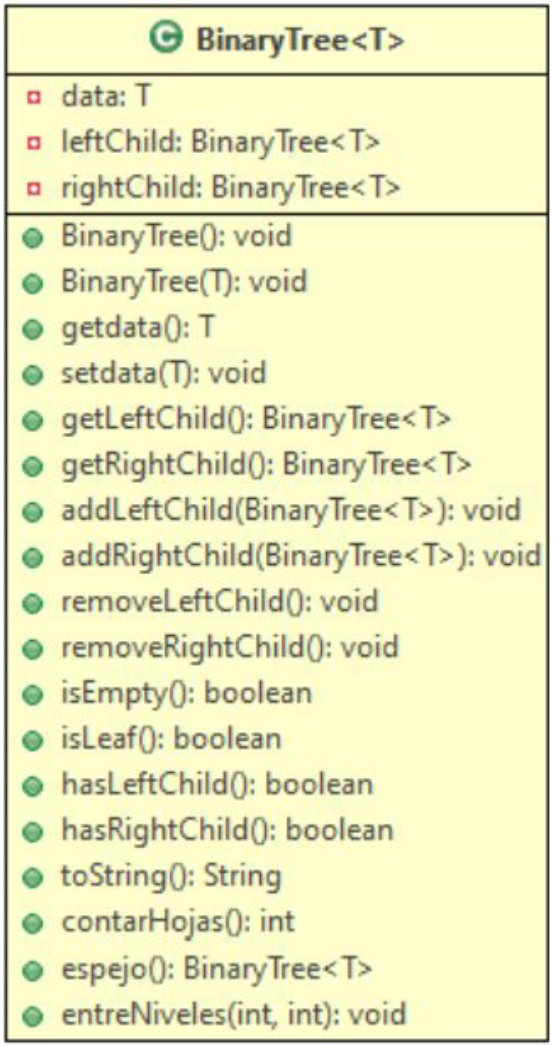
D B E A F C G

post-orden



D E B F G C A

Implementación en Java



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());
        }
    }


}
```