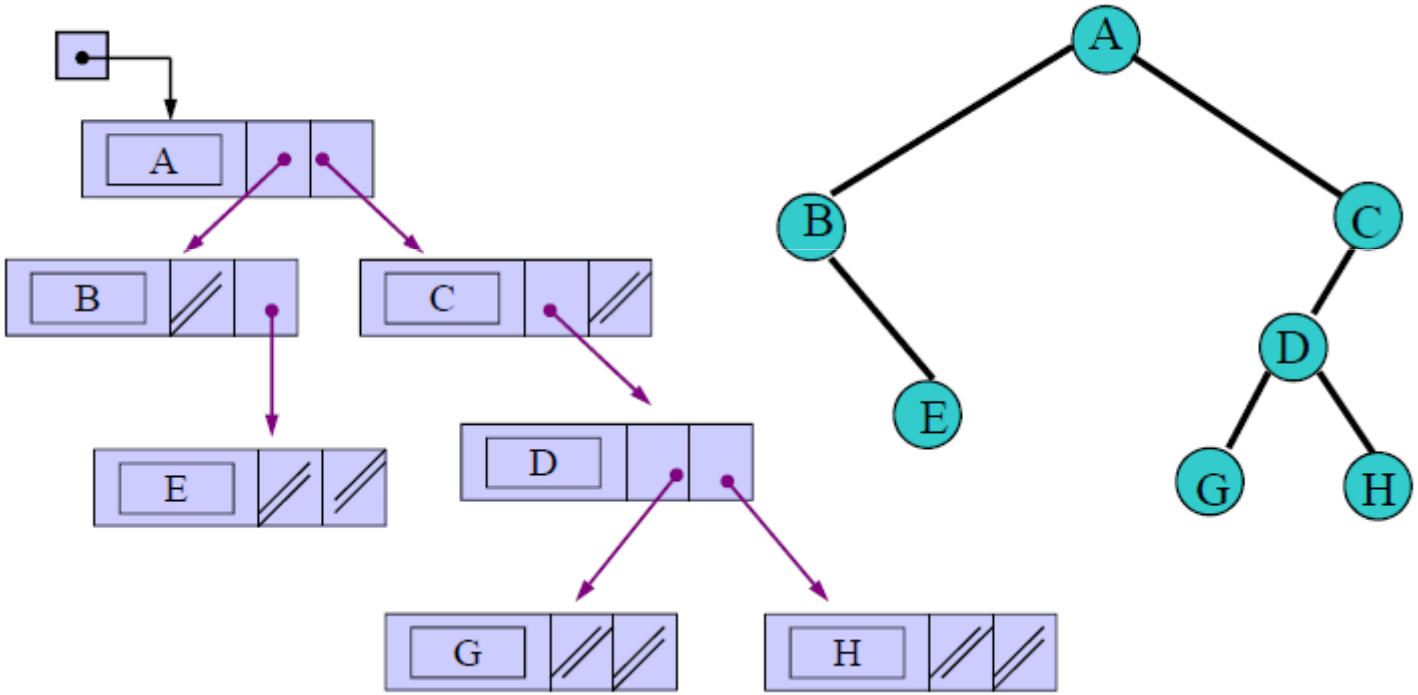
Algoritmos y Estructuras de Datos

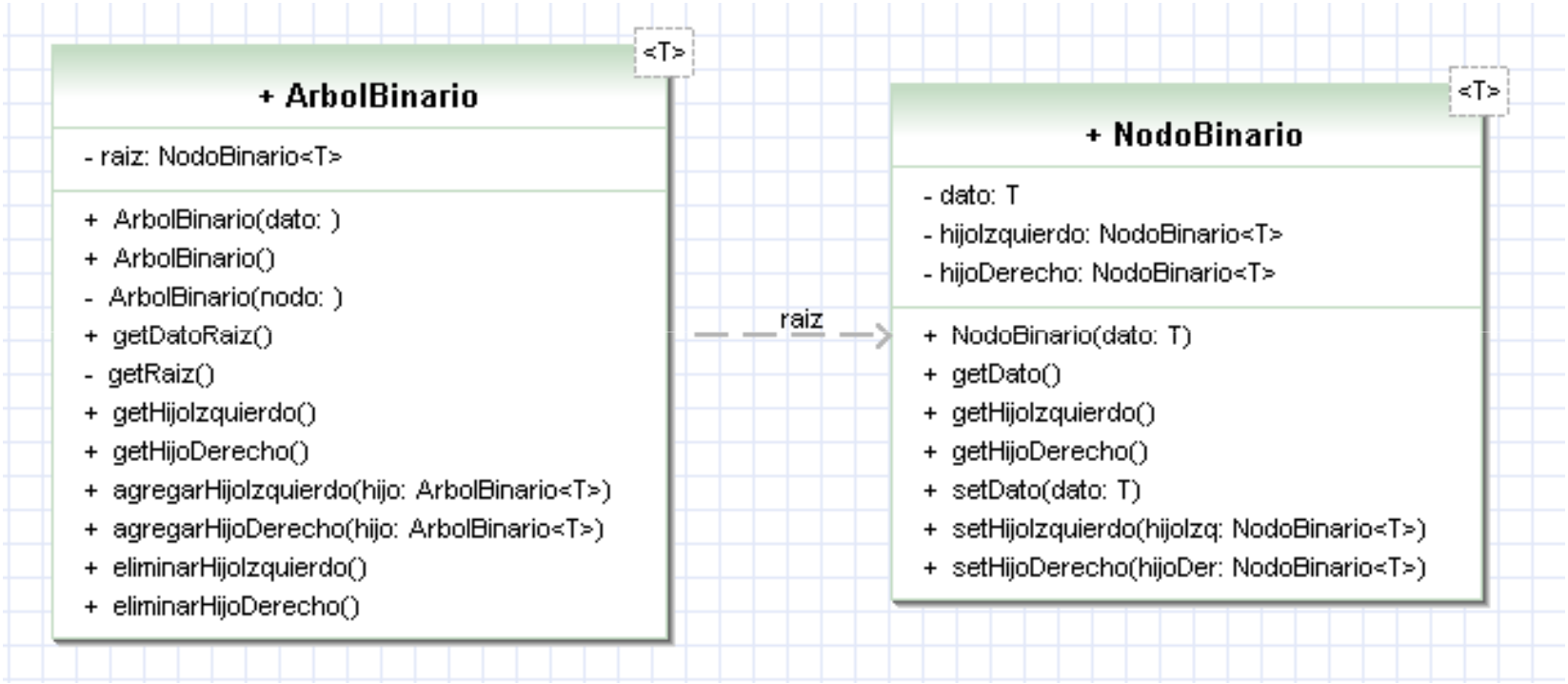
Árboles Binarios
Cursada 2011

Árbol Binario

- Representación Hijo Izquierdo – Hijo Derecho



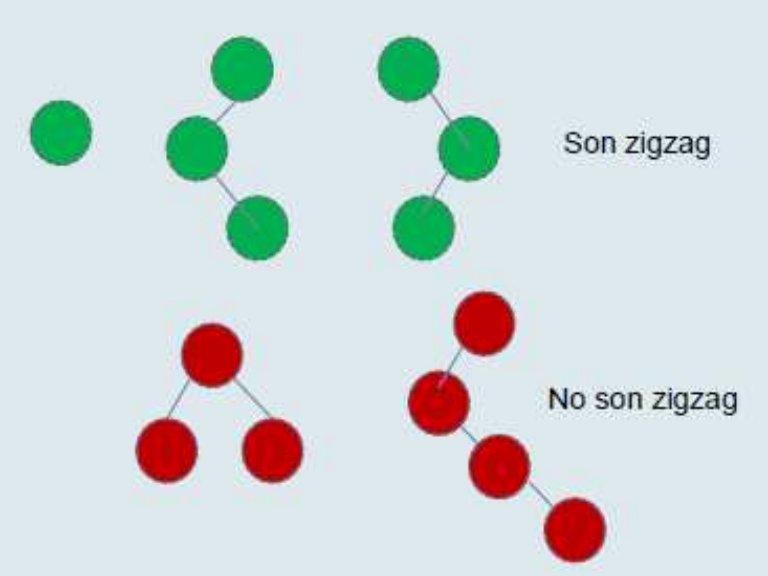
Árbol Binario



Árbol Binario

- **zigZag(): boolean**

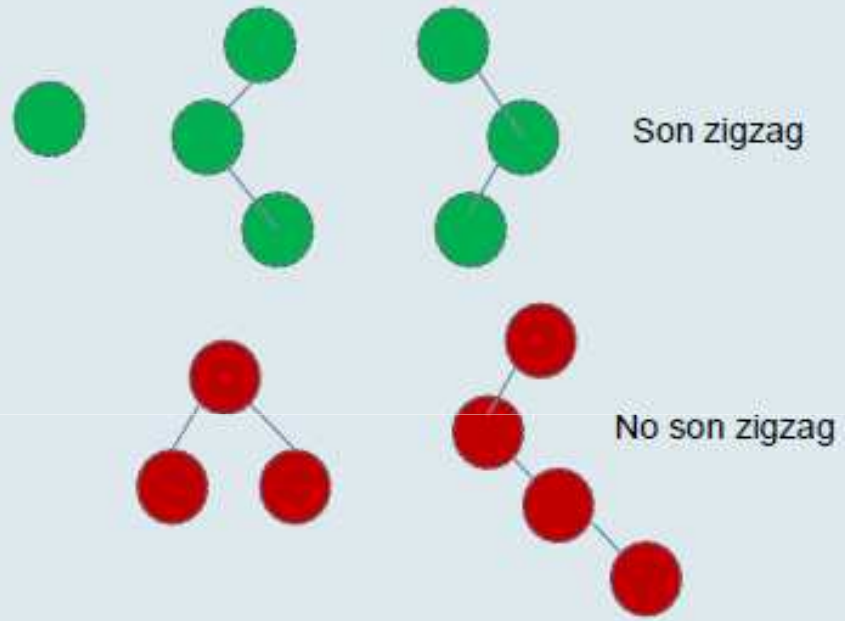
  Devuelve true si el árbol es degenerado con direcciones alternadas, esto es, si en lugar de ser una lista donde todos los hijos están en el lado izquierdo o derecho, se van alternando.



Árbol Binario - **zigZag(): boolean**

```
public boolean zigZag() {
    boolean ok = true;
    if ( ! this.esVacio() ) {
      ArbolBinario <T> hi = this.getHijoIzquierdo();
      ArbolBinario <T>hd = this.getHijoDerecho();
      if  (! this.isLeaf()) {
        if ((! hi.esVacio()) && (! hd.esVacio()))
            ok = false;
        else
            ok = (this.zigZagIzquierda() || this.zigZagDerecha());

      }
    }
    return ok;
}
```



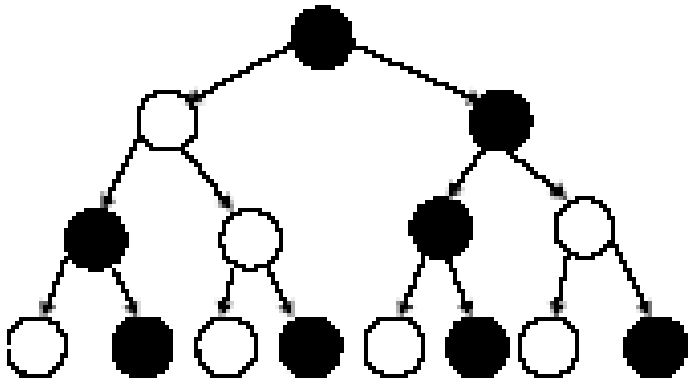

```
private boolean zigZagIzquierda() {
  boolean ok = true;
  if (! this.esVacio() && ! this.esHoja()) {
    if (this.tieneHijoIzquierdo() && ! this.tieneHijoDerecho())
       ok = this.getHijoIzquierdo().zigZagDerecha();
    else
       ok =false;
  }
  return ok;
}
```

```
private boolean zigZagDerecha() {
boolean ok = true;
if (! this.esVacio() && ! this.esHoja()) {
 if (! this.tieneHijoIzquierdo() && this.tieneHijoDerecho())
    ok = this.getHijoDerecho().zigZagIzquierda();
 else
    ok= false;
}
return ok;
}
```

Árbol Binario

- **colorear()**

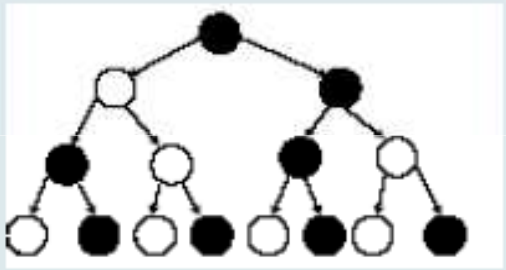
  Un árbol binario coloreado es un árbol binario lleno cuyos nodos tienen uno de dos colores posibles: negro o blanco. El color para la raíz del árbol es negro. Y para cada nivel los colores de los nodos se intercalan, como así también al comenzar cada nivel. Por ejemplo un árbol coloreado de altura 3 deberá ser:



Árbol Binario - **colorear() iterativo**

```
public void colorear() {
    Cola iterar = new Cola();
    int color, topeNivel, cont;
    ArbolBinario <T> arbol;
    color = 1;  topeNivel = 2;  cont = 0;
    if (! this.esVacio()) {
       this.getRaiz().setDato(color);
       iterar.push(this);
       color = 1 - color;
       while (! iterar.esVacia()) {
          arbol = (ArbolBinario<T>) iterar.pop();
          if (! arbol.getHijoIzquierdo().esVacio()){
             arbol.getHijoIzquierdo().getRaiz().setDato(color);
             iterar.push(arbol.getHijoIzquierdo());
             cont = cont + 1;  color = 1 - color;          // contabiliza el árbol del nivel que se encoló y alterna color
             arbol.getHijoDerecho().getRaiz().setDato(color);
             iterar.push(arbol.getHijoDerecho());
             cont = cont + 1; color = 1 - color;          // contabiliza el árbol del nivel que se encoló y alterna color
             if (cont == topenivel) {      // se acabaron los nodos del nivel
               topeNivel = topeNivel * 2;
               cont = 0;  color = 1 - color;       // inicializa las variables para le próximo nivel
             }
          }
       }
    }
}
```

- ✓ El árbol debe ser **lleno** y no vacio
- ✓ El color blanco se representa con el 0
- ✓ El color negro se representa con el 1

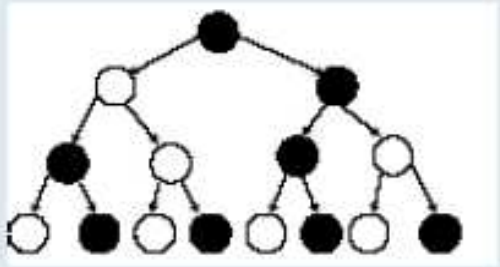


Árbol Binario - **colorear() recursivo**

```
public void colorear() {
    int color, nivel;
    if (! this.esVacio()) {
        color = 1;
        this.getRaiz().setDato(color);
        nivel = 0;
        this.colorear(color, nivel);
    }
}

private void colorear(int color, int nivel) {
    if ( ! this.esVacio()) {
        this.getRaiz().setDato(color);
        If ( (nivel % 2) == 0 )
            color = 0;
        else
            color =1;
        this.getHijoIzquierdo().colorear(color, nivel+1);
        this.getHijoDerecho().colorear(1- color, nivel+1);
    }
}
```

- ✓ El árbol debe ser **lleno** y no vacio
- ✓ El color blanco se representa con el 0
- ✓ El color negro se representa con el 1



En los niveles pares TODOS los nodos tienen que tener el hijo Izquierdo pintado de negro y el hijo derecho pintado de blanco. En los niveles impares es justamente al revés