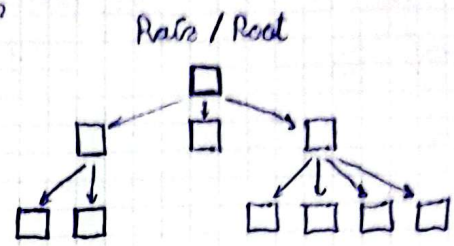
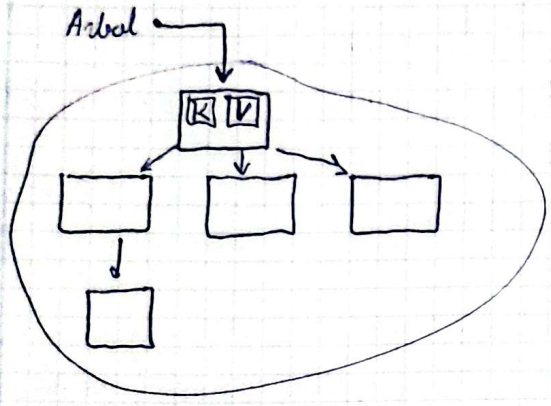
(Estructuras de Datos)

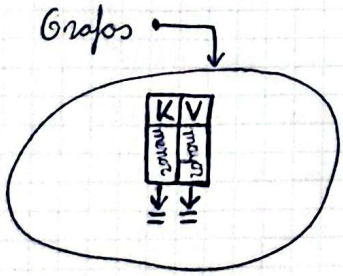
Árboles → Genéricos → De Búsqueda [Binarios] ← sirven para establecer un orden
↳ Equilibrados






```
class Arbol Generico <K,V> {
    Nodo Generico <K,V> raiz;
```

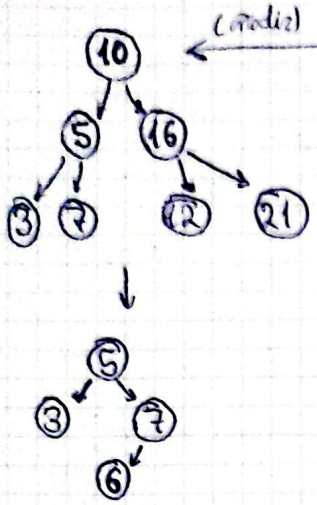
```
class Nodo Generico <K,V> {
    K clave
    V valor
    Lista <Nodo Generico <K,V>> hijos = ...
```




```
class Nodo ABB <K,V> {
    K clave
    V valor
    Nodo ABB<K,V> menor;
    Nodo ABB<K,V> mayor;
    Nodo ABB (K clave, V valor){
        this.clave = clave
        this.valor = valor
    }
```

```
class Arbol Binario de Busqueda <K,V>{
    Nodo ABB <K,V> raiz

    boolean add Nodo ABB (K clave , V valor ) {
        Nodo ABB <K,V> n = new Nodo ABB(K,V)
        this.raiz = n
        return true
    }
```

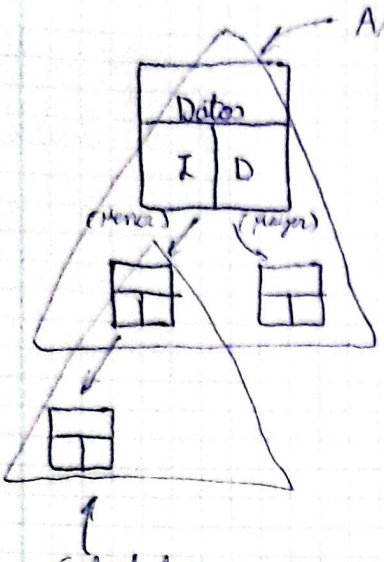


[6] Nodo ABB<K,V> buscarNodoAInsertar(K cBuscada, NodoABB<K,V> nodo)

```
if (nodo.getClave() == cBuscada)
    return null;
if (nodo.getClave() > cBuscada)
    if (nodo.getMenor() != null)
        return buscarNodoAInsertar(cBuscada, nodo.getMenor());
    else
        return nodo;
else
    if (nodo.getMayor() != null)
        return buscarNodoAInsertar(cBuscada, nodo.getMayor());
    else
        return nodo;
```

```
class ArbolBinarioDeBusqueda<K,V>{
    NodoABB<K,V> raiz;

    protected boolean addNodoRaiz(K clave, V valor) {...}
    protected boolean addOtrosNodos(K clave, V valor) {
        NodoABB candidato = buscarNodoAInsertar(clave, this.raiz)
        if(candidato == null)
            return false;
        NodoABB<K,V> n = new NodoABB(K,V);
        if(candidato.getClave() > clave)
            candidato.setMenor(n);
        else
            candidato.setMayor(n);
        return true;
    }
}
```



Subarbol

(Recorridos)


```
Void Preorden(N){
  if(N!=null){
      System.out.println(N.data);
      Preorden(N.getMenor());
      Preorden(N.getMayor());
  }
}
```

→ 10, 3, 5, 7, 15, 11, 19

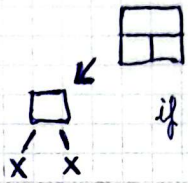
```
Void Central(N){
  if(N!=null){
     Central(N.getMenor());
     System.out.println(N.data);
     Central(N.getMayor());
  }
}
```

→ 3, 5, 7, 10, 11, 15, 19

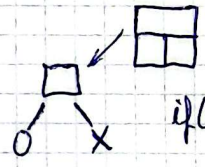
```
Void PostOrden(N){
  if(N!=null){
     PostOrden(N.getMenor());
     PostOrden(N.getMayor());
     System.out.println(N.data);
  }
}
```

→ 3, 7, 5, 11, 19, 15, 10

(Borrado)




```
if(N.padre.getMenor()==N)
    N.padre.setMenor(null);
else
    Npadre.setMayor(null);
```




```
if(N.getMenor()!==null)
    hijo=N.getMenor();
else
    hijo=N.getMayor();
```


```
if(N.padre.getMenor()==N)
    Npadre.setMenor(hijo)
else
    Npadre.setMayor(hijo)
hijo.setPadre(N.padre)
```

(Arboles AVL)

Estos arboles "balancean" el arbol al realizar comandos como añadir o eliminar nodos del arbol. Rango de balanceo $x = 0, \pm 1$. Metodos de balanceo:

- Rotar Derecha
- Rotar Izquierda
- Rotar Derecho - Izquierda
- Rotar Izquierda - Derecha

7, 9, 11, 8, 10, 12, 5, 14