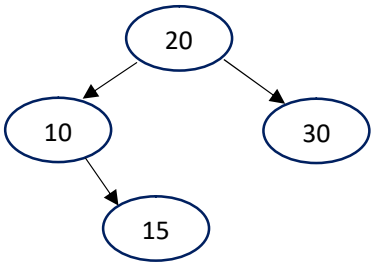
TP - Unidad 05-C
Árboles De Búsqueda. Caso de Uso: Soporte para Índices.

Ejercicio 1

Mostrar paso a paso, gráficamente, cómo queda el árbol AVL luego de realizar cada una de las siguientes operaciones de inserción: 1, 2, 4, 7, 15, 3, 10, 17, 19 y 16

Ejercicio 2

Partiendo del siguiente árbol AVL



Mostrar **gráficamente** cómo va quedando si se le aplican las operaciones solicitadas en secuencia.

Para cada operación se pide:

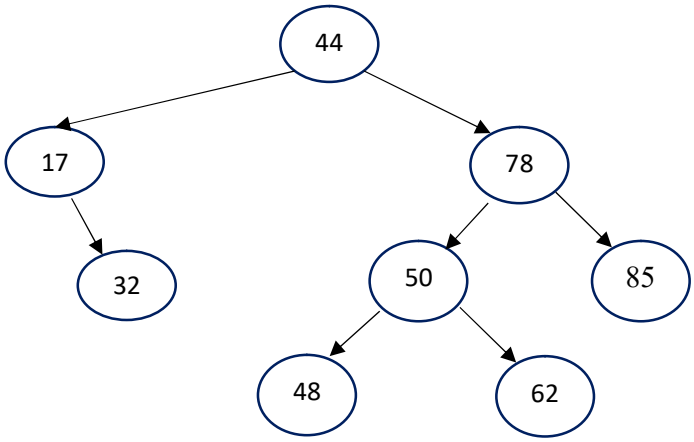
- **Mostrar primero gráficamente** dónde se inserta el valor.
- **Analizar si genera o no un desbalance.** Si genera un desbalanceo, indicar **cuál es el tipo de rotación** que lo soluciona y además **mostrar gráficamente cómo queda el árbol** luego de aplicar la rotación correspondiente.

Operaciones:

- Insertar valor 12
- Insertar valor 14
- Insertar valor 25
- Insertar valor 70
- Insertar valor 90
- Insertar valor 45
- Insertar valor 23
- Insertar valor 27
- Insertar valor 11

Ejercicio 3

Partiendo del siguiente árbol AVL



Mostrar **gráficamente** cómo va quedando si se le aplican las operaciones solicitadas en secuencia.

Para cada operación se pide:

- **Mostrar primero gráficamente** dónde se inserta el valor.
- **Analizar si genera o no un desbalance.** Si genera un desbalanceo, indicar **cuál es el tipo de rotación** que lo soluciona y además **mostrar gráficamente cómo queda el árbol** luego de aplicar la rotación correspondiente.

Operaciones:

- Insertar valor 46
- Insertar valor 66
- Insertar valor 30
- Insertar valor 25
- Insertar valor 42
- Insertar valor 20

Ejercicio 4

Implementar la clase AVL que permita insertar, buscar elementos e imprimirse.

```
public class AVL<T extends Comparable<? super T>> implements BSTreeInterface<T> {

…
}
```

Ejercicio 5

5.1) Dibujar el árbol de Fibonacci de orden 6. ¿Cuantos nodos tiene? Calcular su altura.

5.2) Cuál sería la altura de un completo (perfectamente balanceado) con la misma cantidad de nodos?

Ejercicio 6

Demostrar **por inducción** que para un árbol AVL no vacío de altura h, la altura está acodada superiormente por log(N) donde N es la cantidad de nodos de dicho AVL.

Tip:
Demostrarlo para arboles de Fibonacci de altura h se satisface que CantNodos(h) >= goldennumber^h
Para cualquier otro AVL de misma altura tendrán igual o más nodos, seguirá valiendo por cota superior, o sea, N >= goldennumber^h
De esa última formula surgirá que la cota superior para altura en cualquier AVL es log(N)

Ejercicio 7

Mostrar gráficamente paso a paso cómo queda el **red-black tree** después de cada inserción. Explicar las operaciones y chequeos que se realizan para garantizar que se mantienen las propiedades del red-black tree luego de cada inserción.

Las inserciones se realizan en el siguiente orden: 10, 18, 5, 15, 17, 29, 40, 91

Ejercicio 8

Mostrar gráficamente paso a paso cómo queda el **red-black tree** después de cada inserción. Explicar las operaciones y chequeos que se realizan para garantizar que se mantienen las propiedades del red-black tree luego de cada inserción.

Las inserciones se realizan en el siguiente orden: 15, 4, 2, 12, 20, 9, 13, 35

Ejercicio 9

Se quiere insertar en un **árbol B de orden 1**. Mostrar paso a paso gráficamente como queda el mismo al insertar las claves en secuencia: 100, 80, 40, 20 y 60

Ejercicio 10

Insertar las siguientes claves en un árbol B de orden 1: 10, 20, 30, 50, 70, 100, 150, 130, 120, 220, 180, 200, 240, 140, 160 en forma gráfica.

Ejercicio 11

Insertar en un árbol B de orden 2, las claves ordenadas del 0 al 19 inclusive en forma gráfica.

Ejercicio 12

Crear un Proyecto Java y usar la implementación publicada en:
https://github.com/phishman3579/java-algorithms-implementation/blob/master/src/com/jwetherell/algorithms/data_structures/BTree.java

https://github.com/phishman3579/java-algorithms-implementation/blob/master/src/com/jwetherell/algorithms/data_structures/interfaces/ITree.java

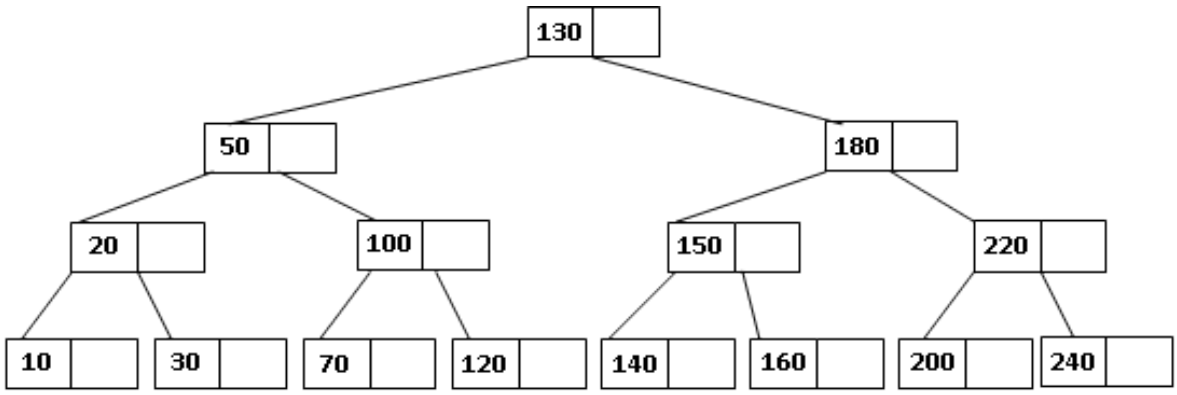
Chequear con el código que lo propuesto en el ejercicio 10 es correcto.

```
public class Test {
  public static void main(String[] args) {
   BTree<Integer> st = new BTree<>(2);

   for(int rec= 0; rec < 20; rec++)
   {
    st.add(rec);
    System.out.println( st.toString() );
   System.out.println("....");
   }
  }
}
```

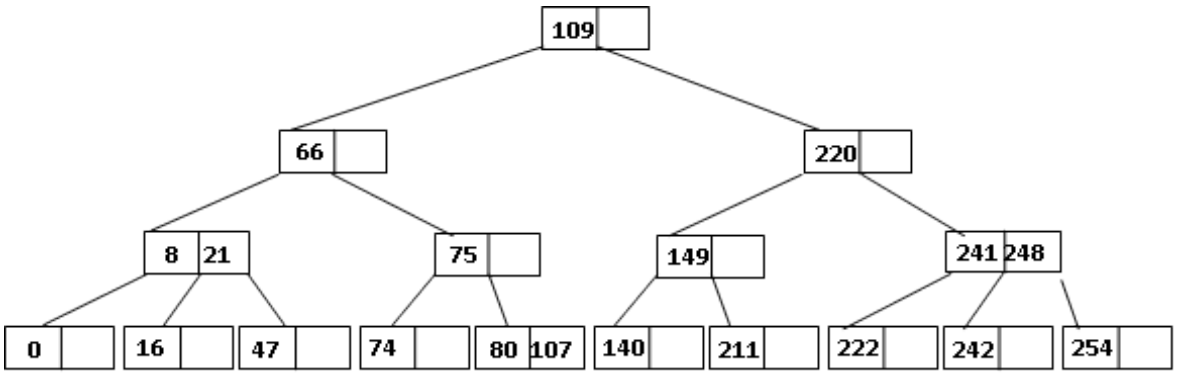
Ejercicio 13

A partir del siguiente árbol de Orden 1, eliminar gráficamente 200, 220, 50



Ejercicio 14

El siguiente árbol surgió de inserción en un árbol B de orden 1 las claves 0, 8, 109, 220, 222, 241, 149, 107, 75, 248, 254, 140, 16, 66, 74, 21, 211, 47, 80 y 242. **Eliminar gráficamente** las claves 66, 21, 109, 241, 149, 140, 211, 220 y 242.


109
66
220
8
21
75
149
241
248
0
16
47
74
80
107
140
211
222
242
254

Ejercicio 15

Modificar el código del árbol B para que en vez de usar el predecesor inorder use la otra opción. Chequear correctitud.

Ejercicio 16

Insertar y Borrar en un Arbol B de Orden 2. Verificarlo con el código.
Inserción
0, 8, 109, 220, 222, 241, 149, 107, 75, 248, 254, 140, 16, 66, 74, 21, 211, 47, 80 y 242.
Eliminación
66, 21, 109, 241, 149, 140, 211, 220 y 242.