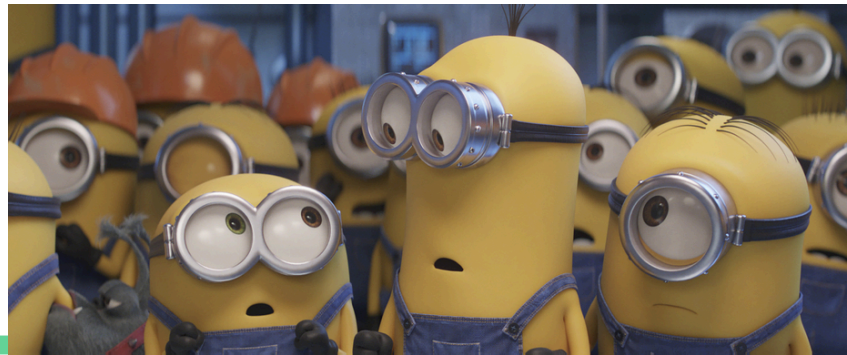
# Ayudantía ABB -AVL y 2-3



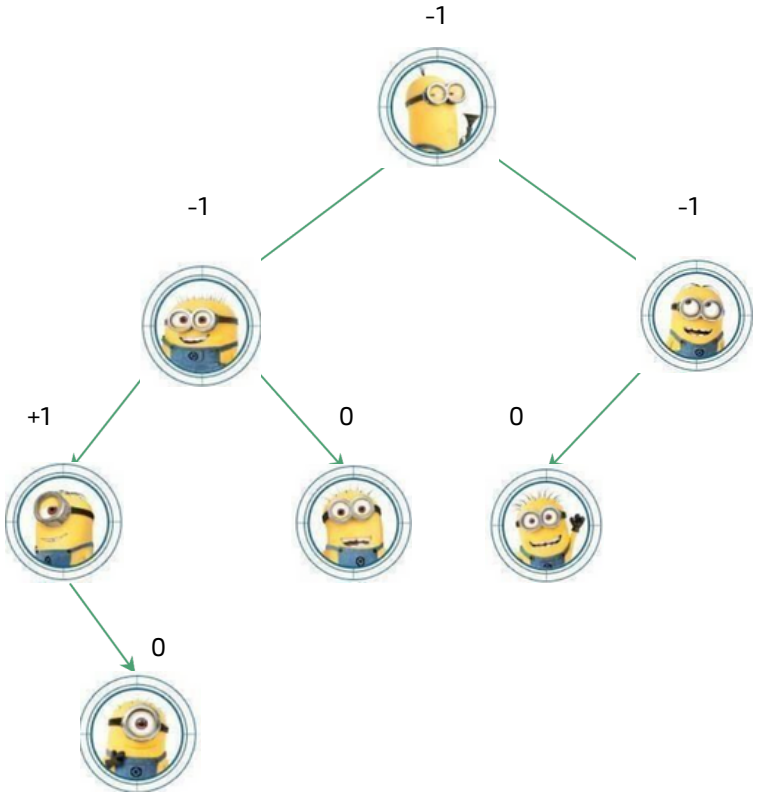
# ¿Que es un arbol AVL?

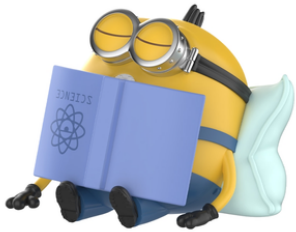
- Es un tipo de ABB
- Se encuentra balanceado, entonces altura O(log(n))
- La diferencia máxima de sus hijos es de 1
- Cada hijo es AVL (definición recursiva)
- Su nombre es por los científicos soviéticos

Georgii **A**delson-**V**elskii y Yevgeniy **L**andis.

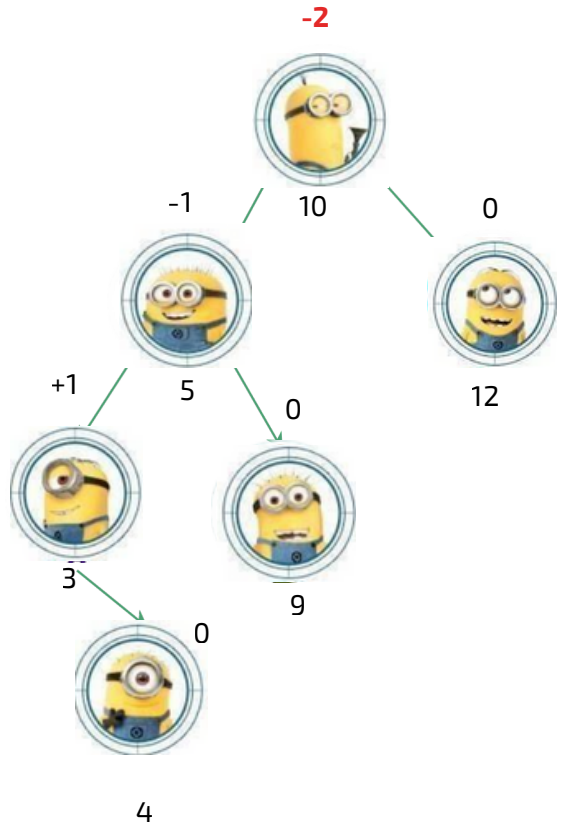


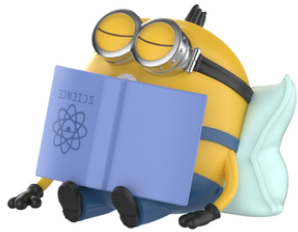
# ¿Que significa que difiera a lo más de 1?



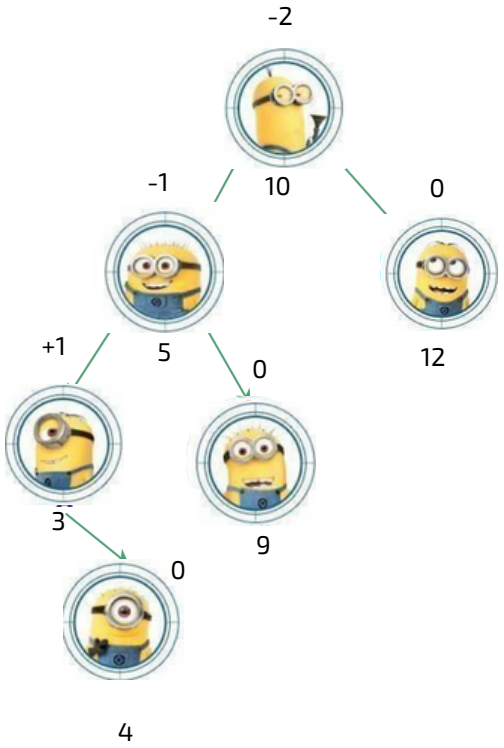
# ¿Que significa que difiera a lo más de 1?



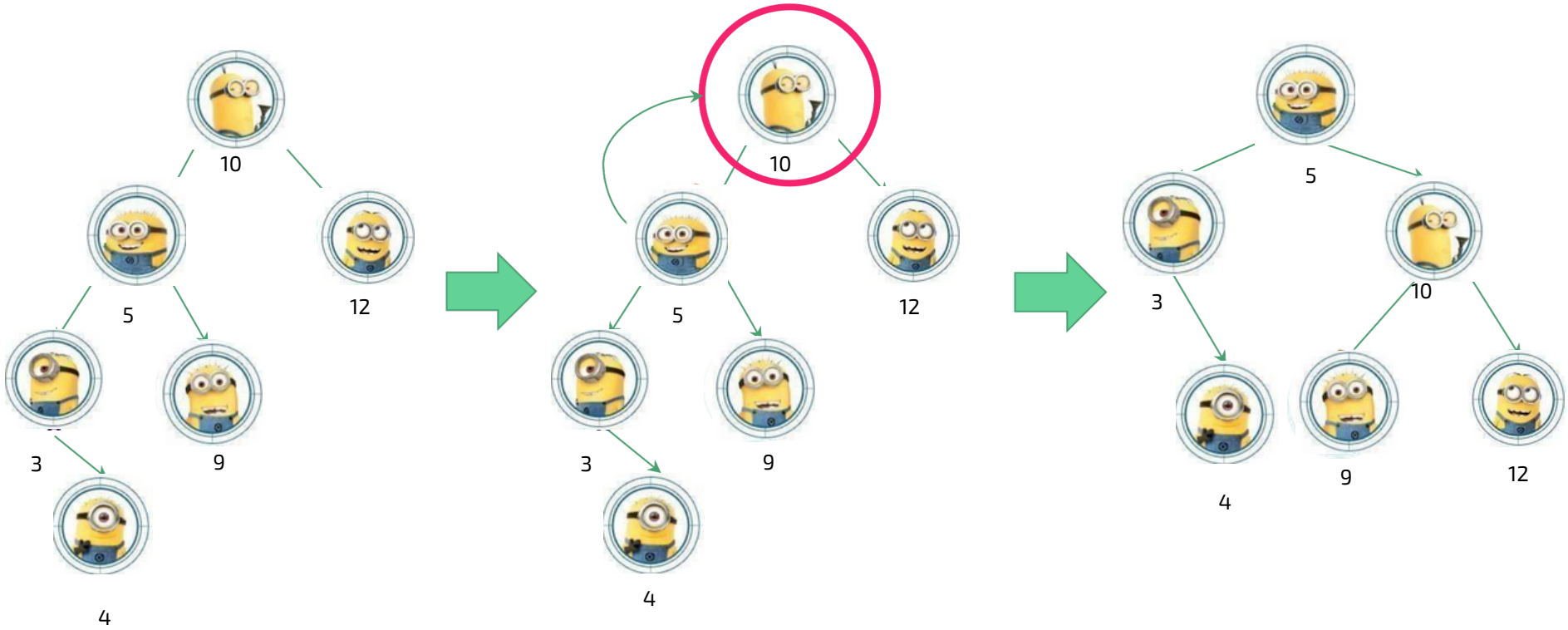



# ¿Cómo lo arreglamos?

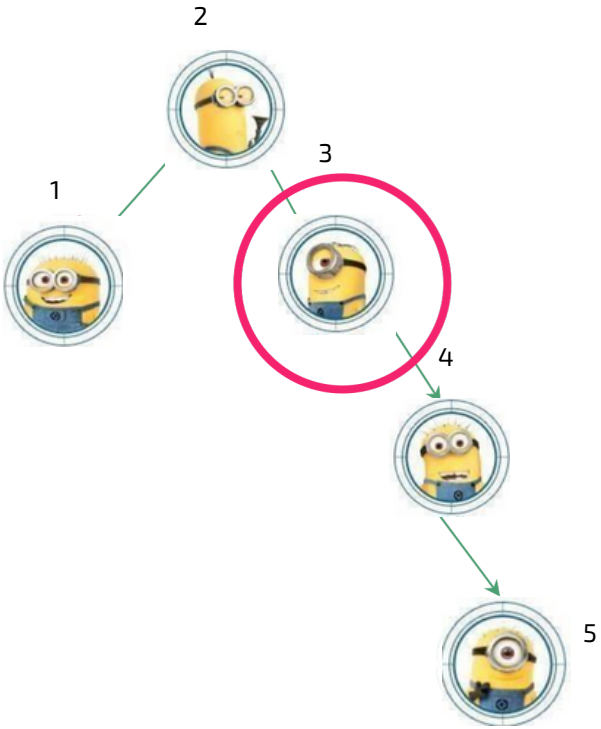
- Identificar primer nodo desbalanceado de forma ascendente
- Identificar rotación
- Rotar
- Evaluar balance

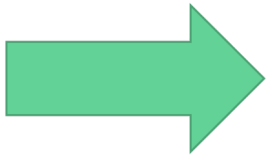


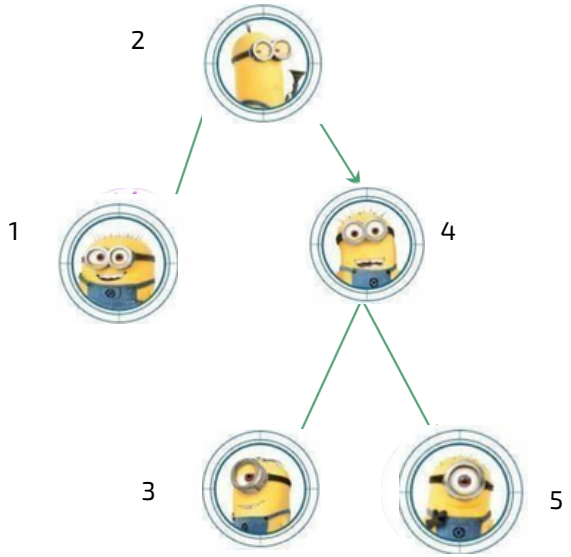
# Rotación Simple



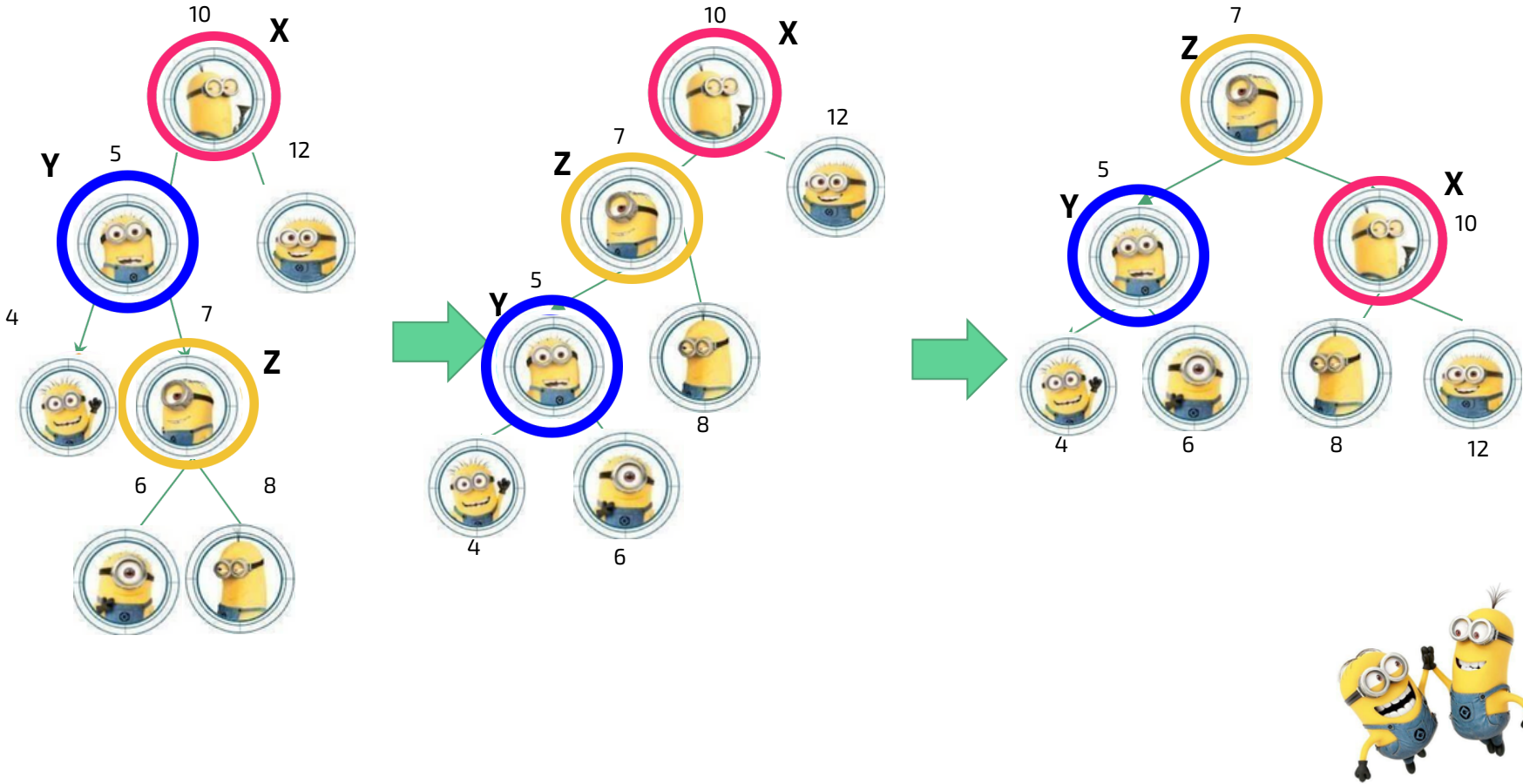
# Rotación Simple



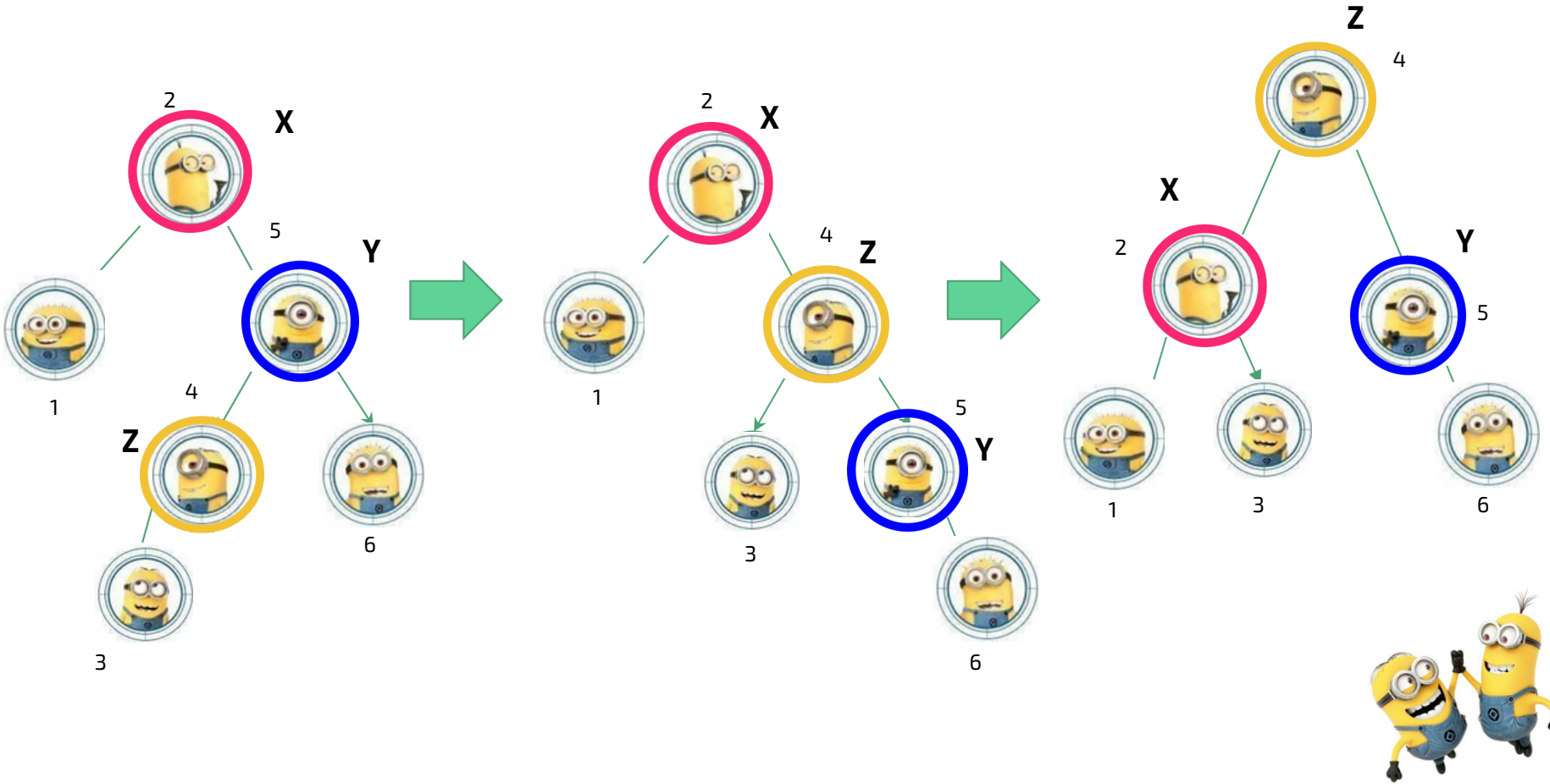





# Rotación Doble



# Rotación Doble



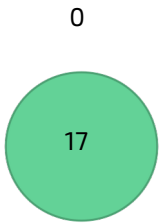
# Ejercicio rotación AVL

Considera que tienes un AVL vacío y una lista desordenada:

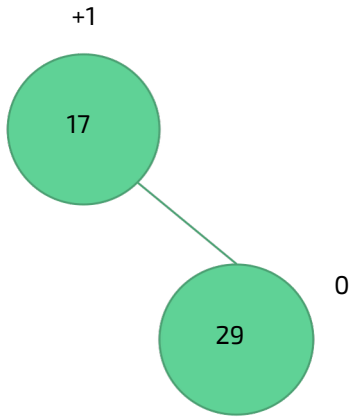
17, 29, 53, 61, 73, 37

Inserta los nodos realizando las rotaciones diferentes

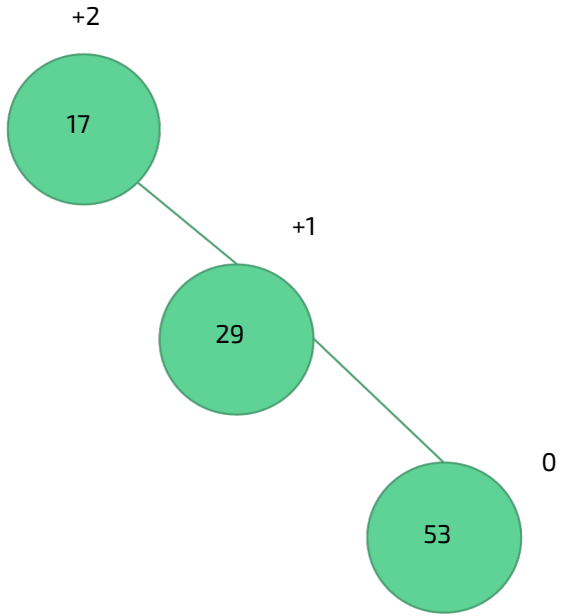
# Insertar 17



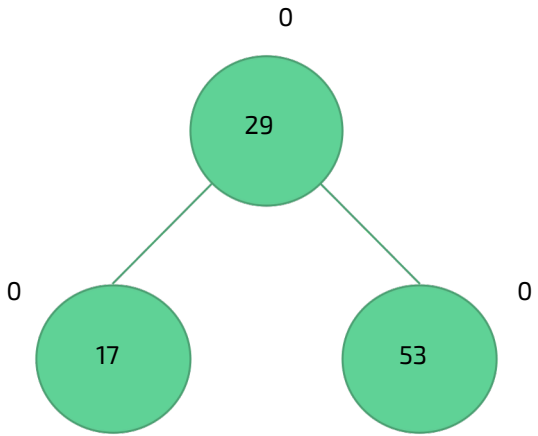
# Insertar 29



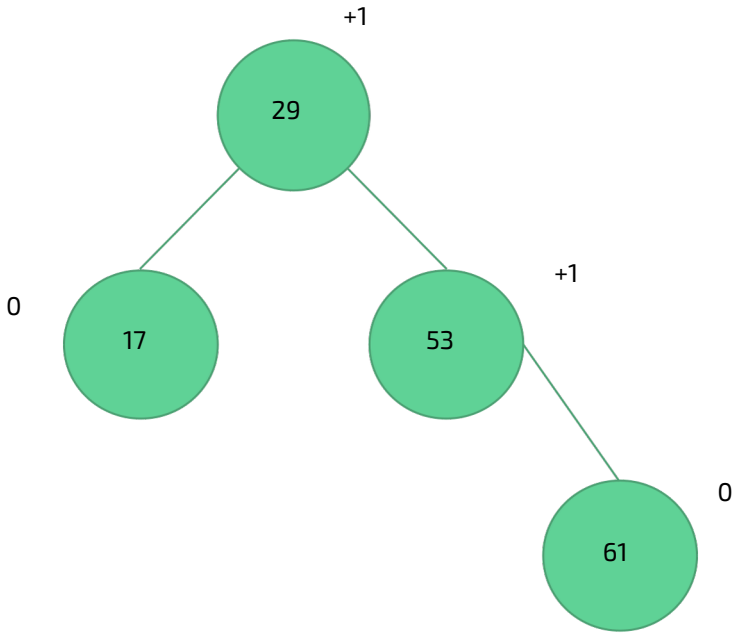
# Insertar 53



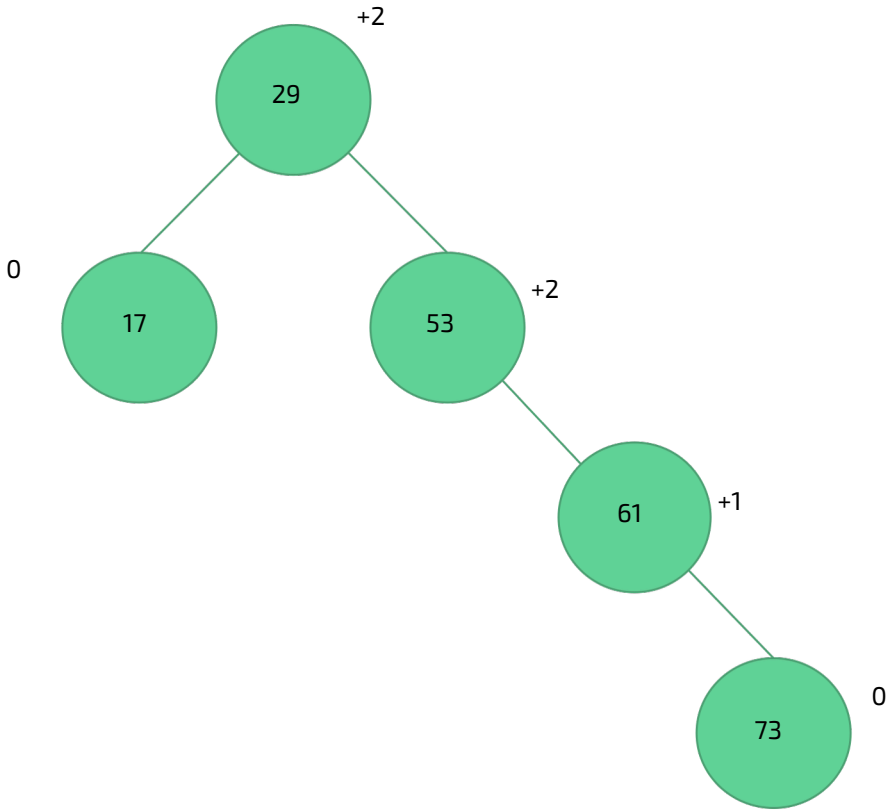
# Rotación Simple



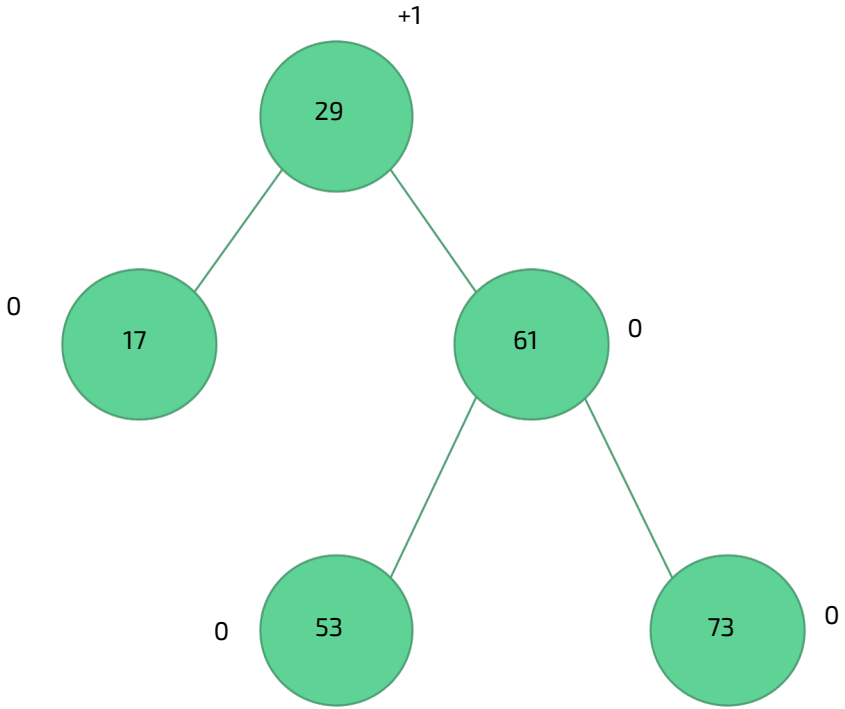
# Insertar 61



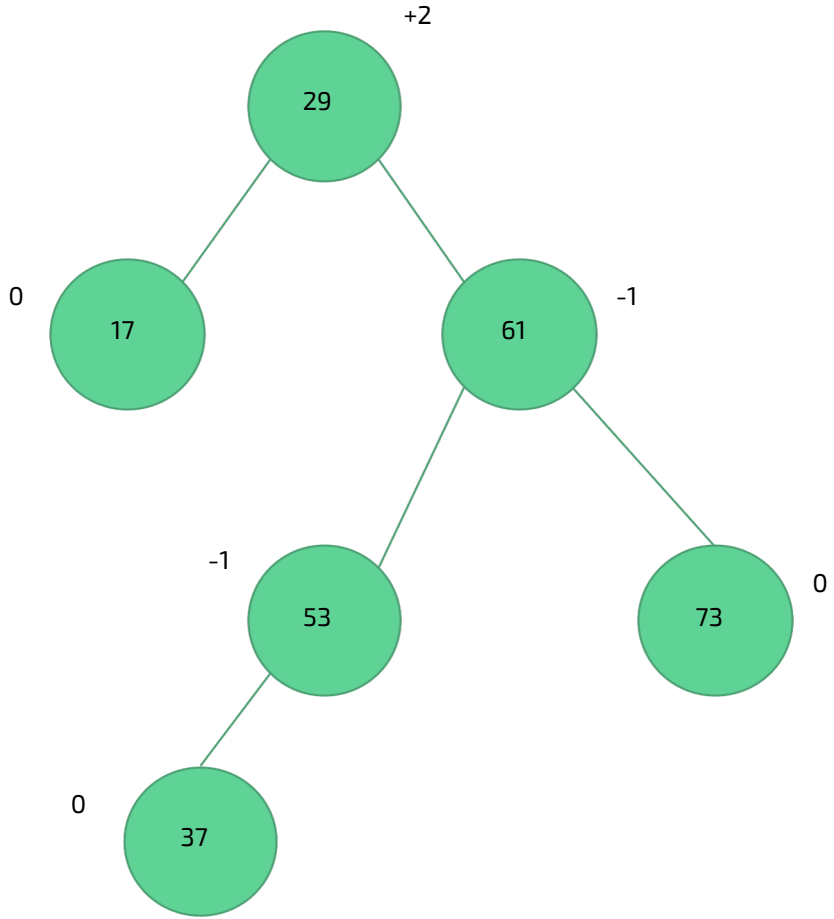
# Insertar 73



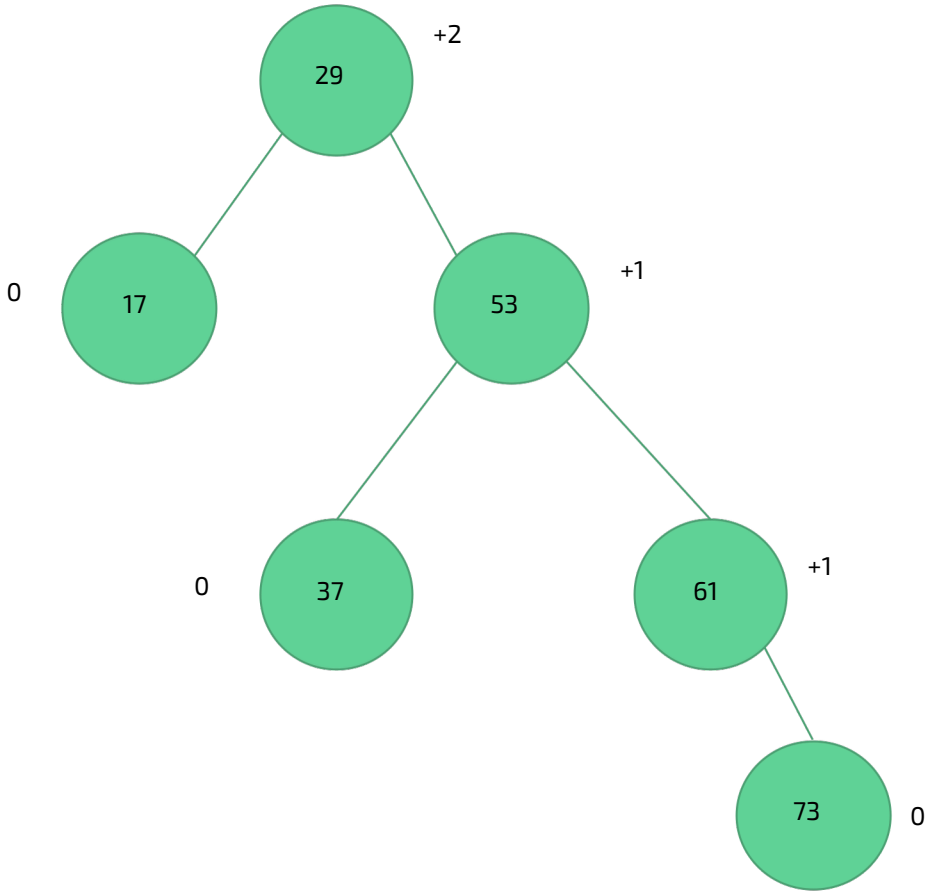
# Rotación Simple



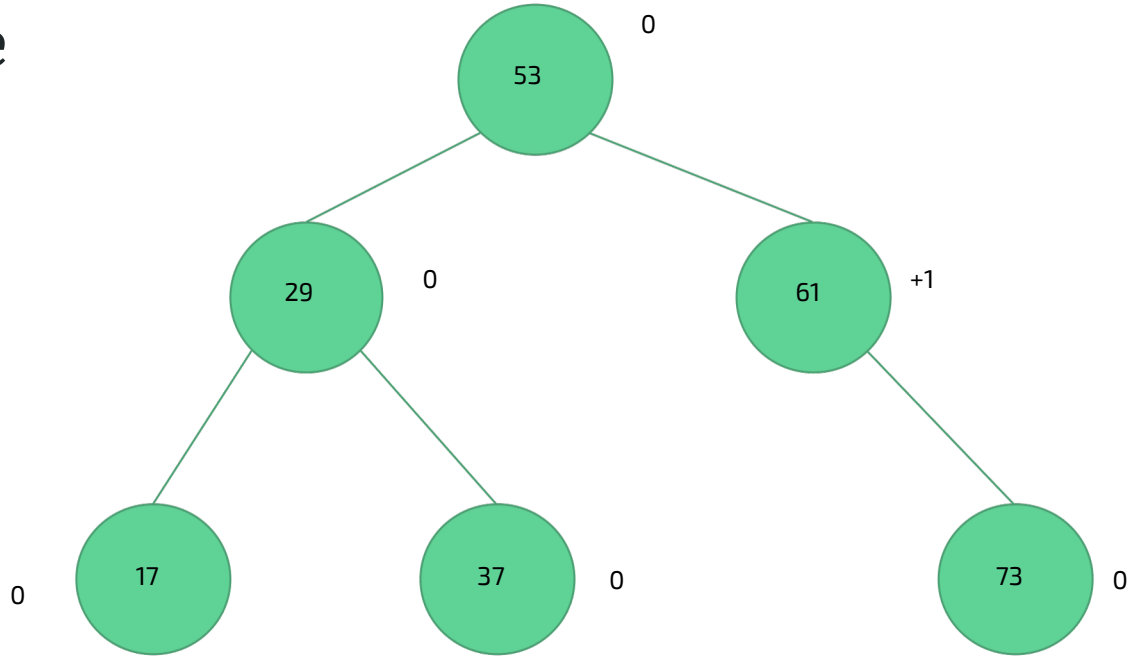
# Insertar 37



# Rotación doble (parte 1)



# Rotación doble (parte 2)



# Árboles 2-3

# Árboles 2-3

● Queremos tener las hojas del árbol a la misma profundidad → garantizamos profundidad O(log(n)) para operaciones de búsqueda

● Tendremos 2 tipos de **nodos** :

- Cada **nodo** esta compuesto por 1 o 2 **valores** ordenados.
- Cada nodo puede tener exacta- y respectivamente 2 o 3 hijos(o 0 en caso de ser un nodo hoja).

  Si el nodo tiene 2 hijos, es un 2-nodo. Si tiene 3 hijos, es un 3-nodo.
- En un 2-nodo, que tiene un valor k:

  [] El hijo izquierda cumplira con ser menor que k.

  [] El hijo derecho cumplira con ser mayor que k.
- En un 3-nodo, que tiene dos valores ordenados k1 y k2:

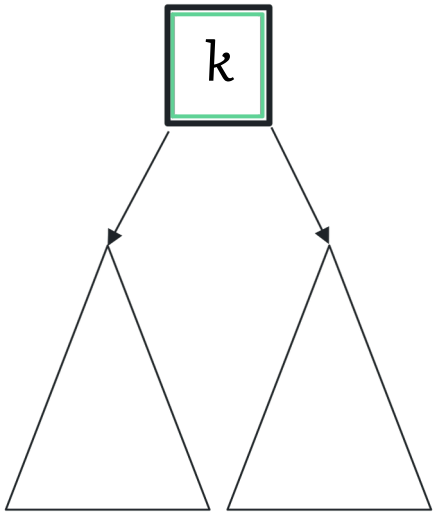
  [] El hijo izquierdo cumplira con ser menor que k1.

  [] El hijo central estara entre k1 y k2.

  [] El hijo derecho cumplira con ser mayor que k2.

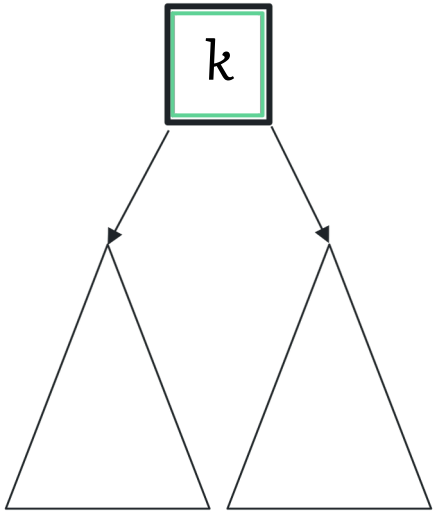
# Árboles 2-3: Ejemplos

**2-Nodo**

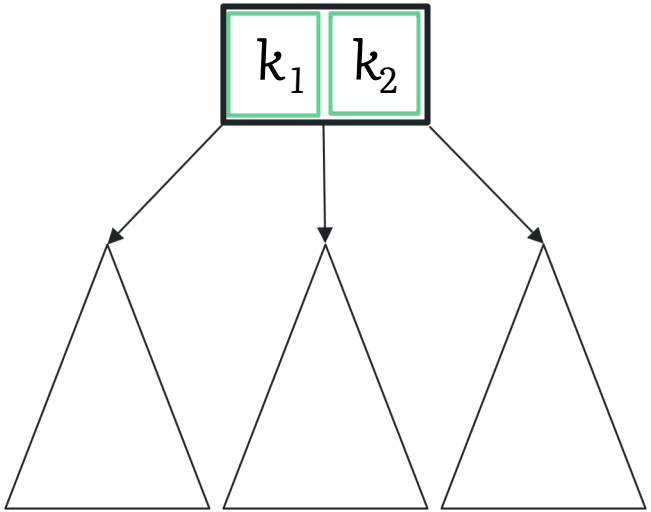


# Árboles 2-3: Ejemplos

**2-Nodo**

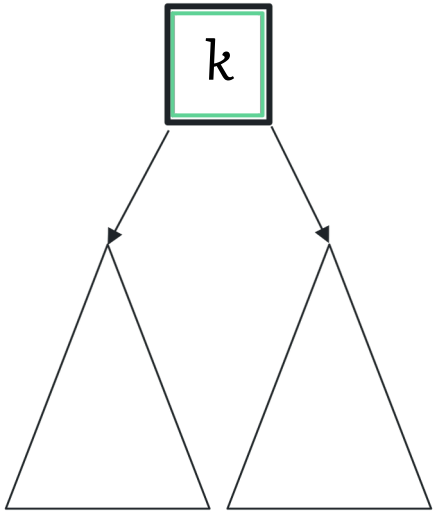



**3-Nodo**

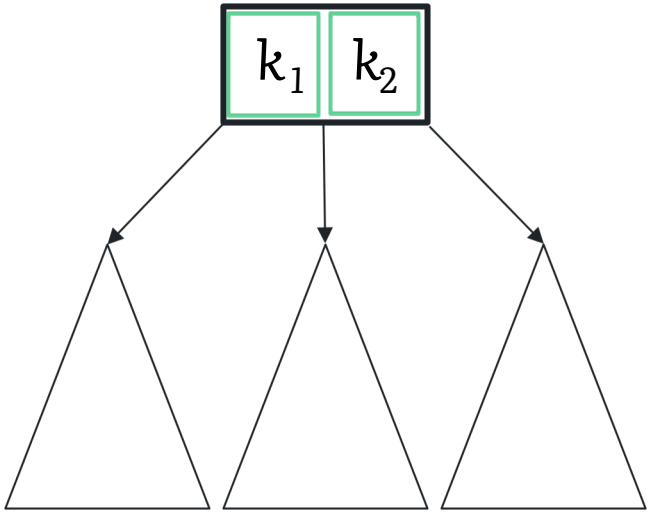


# Árboles 2-3: Ejemplos

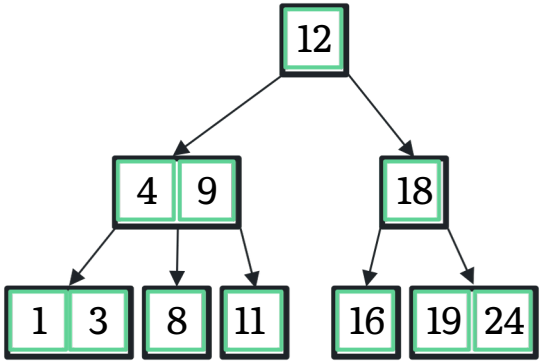
**2-Nodo**




**3-Nodo**




**Árbol 2-3**



# Construyendo el árbol 2-3

- La construccion se hace en base a inserciones
- La idea es construir el arbol de “abajo hacia arriba”
- La secuencia de pasos es:
  1. Insertar el valor en una hoja existente.
  2. Si el nodo hoja queda como 3-nodo, terminamos.
  3. Si el nodo hoja queda como 4-nodo (con 3 valores), tenemos que subir el valor del medio al siguiente nivel y crear dos 2-nodos con los valores que quedan en el nivel. Cada uno de estos sera apuntado por el lado izquierdo y derecho del valor que subio segun corresponda (ver ejemplo).
  4. Se repite la modificacion de forma recursiva hacia la raiz.
- Para saber en que hoja insertar, buscamos como si fuese un ABB (vamos descendiendo por el arbol)

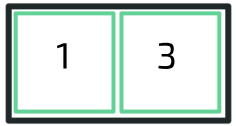
## Construyendo el árbol 2-3

# Crearemos un árbol 2-3 con los siguientes valores:
# [3, 1, 7, 8, 9, 2, 10, 0, 4, 5, 6, -1]

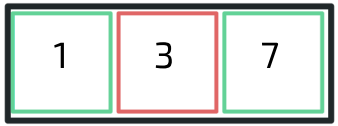
## Valores restantes: [3, 1, 7, 8, 9, 2, 10, 0, 4, 5, 6, -1]



## Valores restantes: [1, 7, 8, 9, 2, 10, 0, 4, 5, 6, -1]

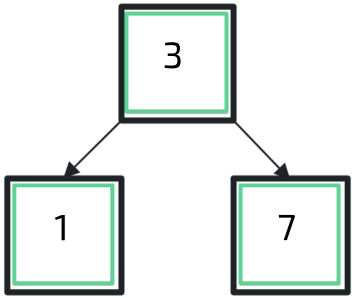


# Valores restantes: [7, 8, 9, 2, 10, 0, 4, 5, 6, -1]

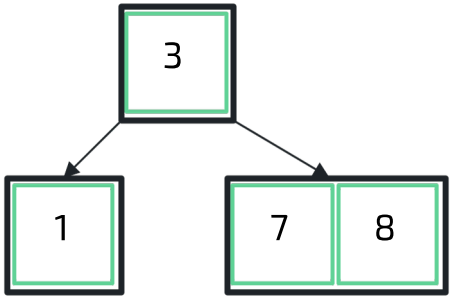



Toca subir el valor del centro

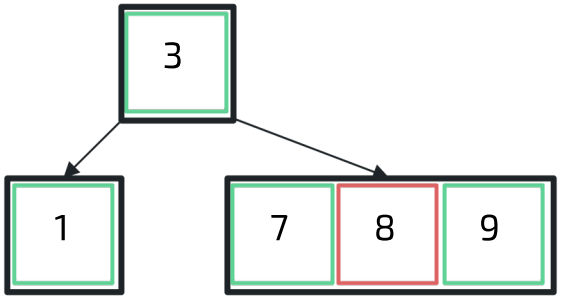
# Valores restantes: [7, 8, 9, 2, 10, 0, 4, 5, 6, -1]



# Valores restantes: [8, 9, 2, 10, 0, 4, 5, 6, -1]

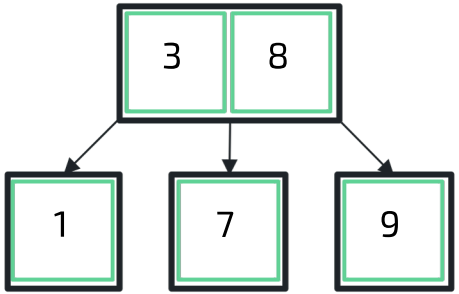


# Valores restantes: [9, 2, 10, 0, 4, 5, 6, -1]

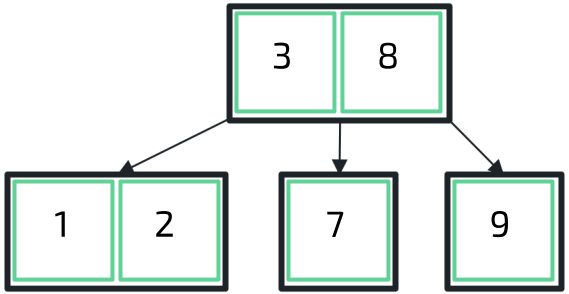



Toca subir el valor del centro

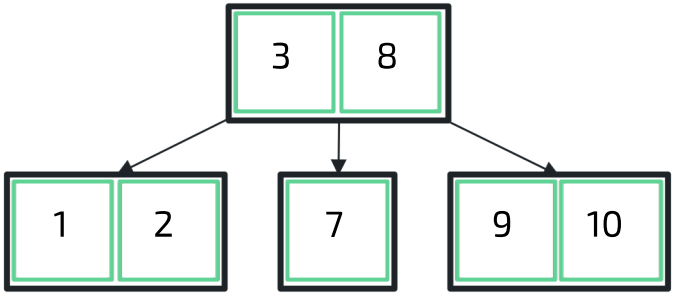
# Valores restantes: [9, 2, 10, 0, 4, 5, 6, -1]



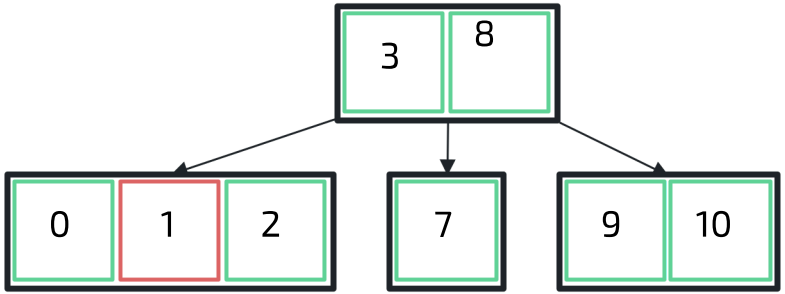
# Valores restantes: [2, 10, 0, 4, 5, 6, -1]



# Valores restantes: [10, 0, 4, 5, 6, -1]

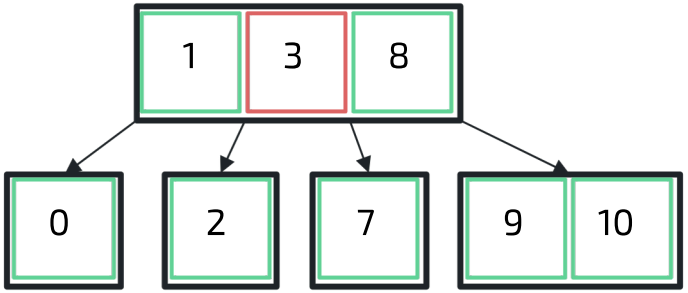


# Valores restantes: [0, 4, 5, 6, -1]



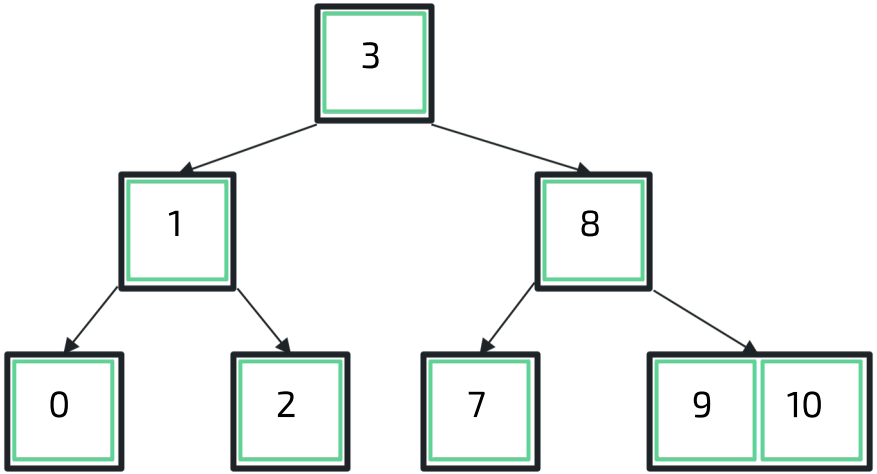

Toca subir el valor del centro

# Valores restantes: [0, 4, 5, 6, -1]

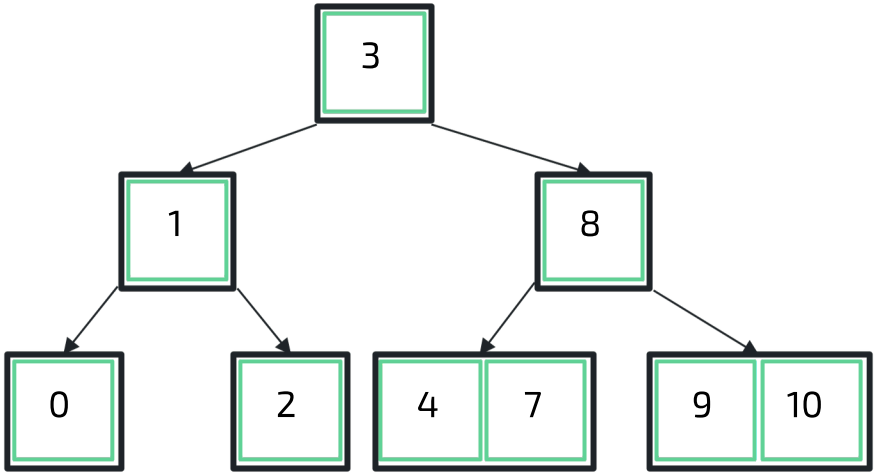



Toca subir el valor del centro

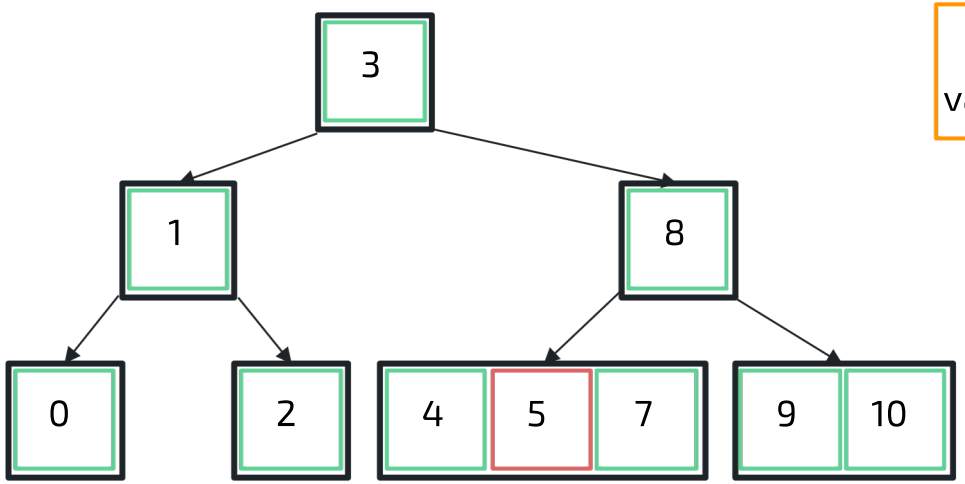
# Valores restantes: [0, 4, 5, 6, -1]



# Valores restantes: [4, 5, 6, -1]

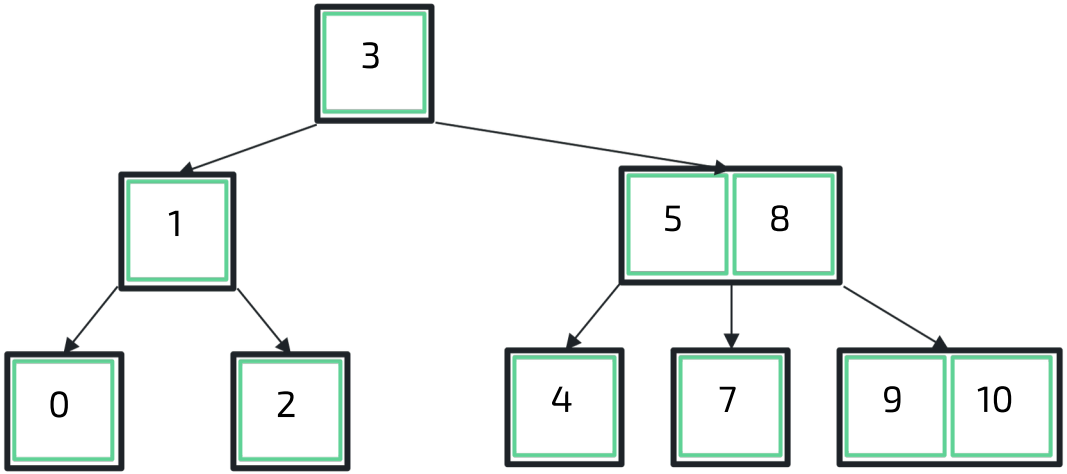


# Valores restantes: [5, 6, -1]

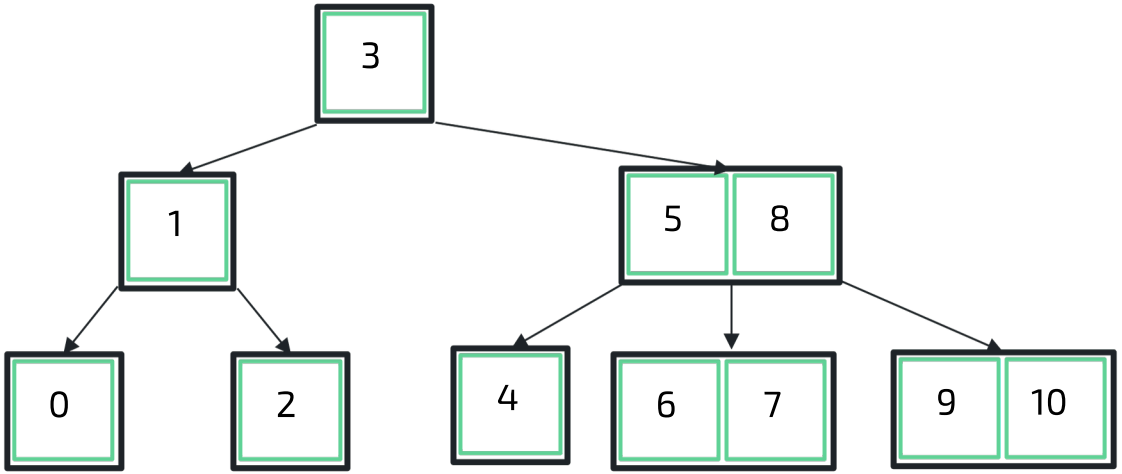



Toca subir el valor del centro

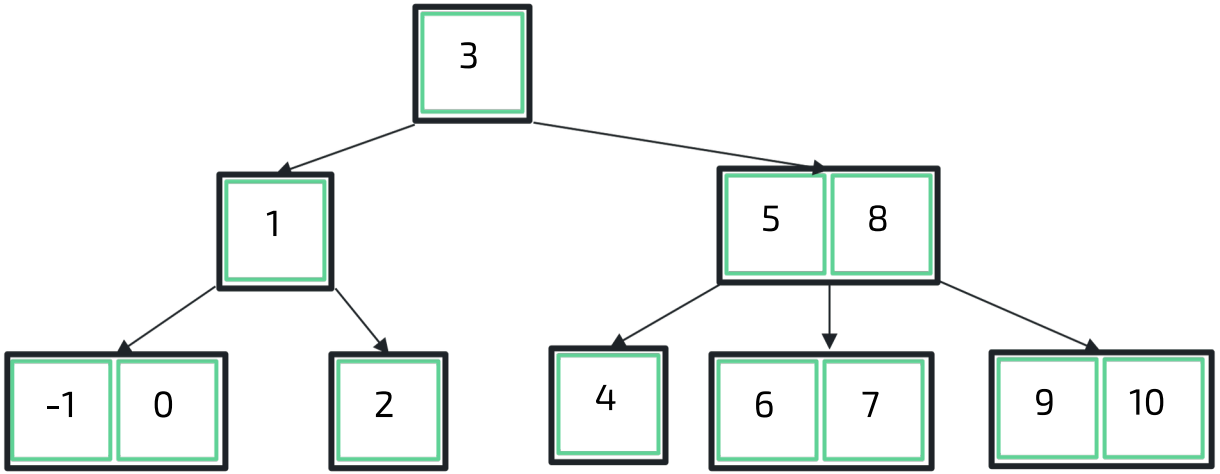
# Valores restantes: [5, 6, -1]



# Valores restantes: [6, -1]

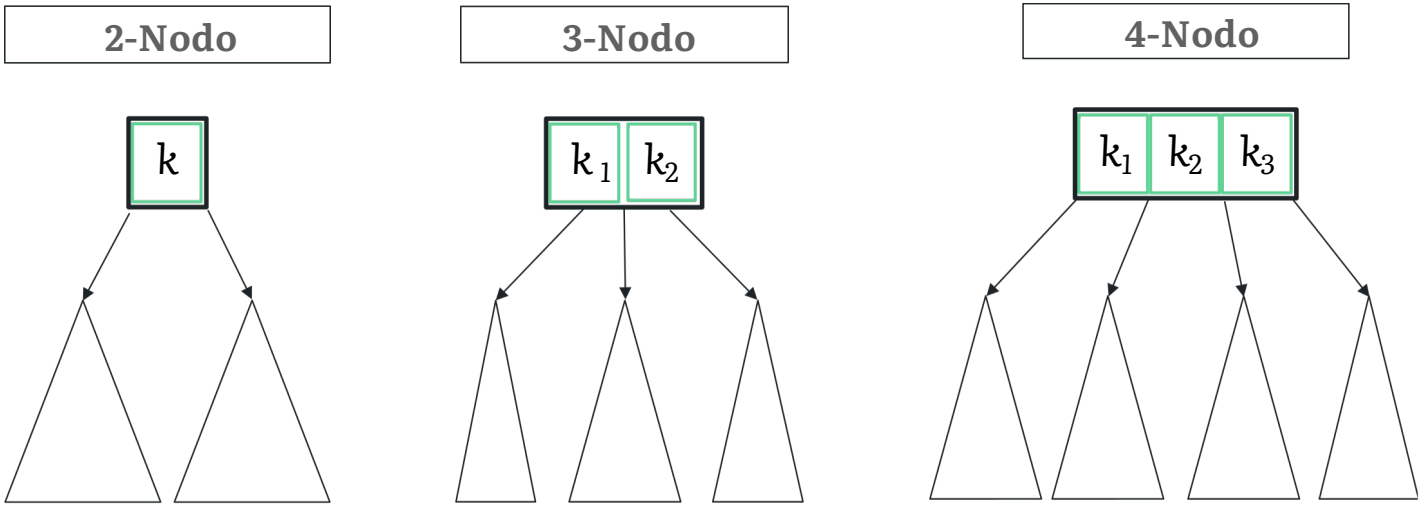


# Valores restantes: [-1]



# Árboles 2-4

- Igual que un árbol 2-3, pero que ahora permite 4-nodos.
- Tendremos 3 tipos de **nodos**:
  - Los mismos de antes(2-nodos y 3-nodos)
  - 4-nodo: está compuesto por 3 **valores** ordenados y tiene 4 hijos (salvo que el nodo sea hoja).



# Ejercicios



# Ejercicio 1



# I1 2025-1 P2 ABB, AVL, 2-3

a) (2pts) Considere un árbol AVL inicialmente vacío:

   (i) Proponga una secuencia de inserciones de claves tal que requiera lo antes posible una rotación simple para mantener el balance del árbol.

# I1 2025-1 P2 ABB, AVL, 2-3

a) (2pts) Considere un árbol AVL inicialmente vacío:

(i) Proponga una secuencia de inserciones de claves tal que requiera lo antes posible una rotación simple para mantener el balance del árbol.

**Solucion:** Un ejemplo podria ser 3, 2, 1; o tambien 1, 2, 3... en suma, cualquier secuencia creciente o decreciente de tres claves. Ejemplo en la diapo.

# I1 2025-1 P2 ABB, AVL, 2-3

a) (2pts) Considere un árbol AVL inicialmente vacío:

(I) Proponga una secuencia de inserciones de claves tal que requiera lo antes posible una rotación simple para mantener el balance del árbol.

**Solucion:** Un ejemplo podria ser 3, 2, 1; o tambien 1, 2, 3... en suma, cualquier secuencia creciente o decreciente de tres claves. Ejemplo en la diapo.

(II) Proponga una secuencia de inserciones de claves tal que requiera lo antes posible una rotación doble para mantener el balance del árbol. Dibuja paso a paso el proceso de inserciones.

# I1 2025-1 P2 ABB, AVL, 2-3

a) (2pts) Considere un árbol AVL inicialmente vacío:

(I) Proponga una secuencia de inserciones de claves tal que requiera lo antes posible una rotación simple para mantener el balance del árbol.

**Solucion:** Un ejemplo podria ser 3, 2, 1; o tambien 1, 2, 3... en suma, cualquier secuencia creciente o decreciente de tres claves. Ejemplo en la diapo.

(II) Proponga una secuencia de inserciones de claves tal que requiera lo antes posible una rotación doble para mantener el balance del árbol. Dibuja paso a paso el proceso de inserciones.

**Solucion:** Tambien basado en la diapo: 4, 2, 3; o 4, 6, 5; es decir, cualquier secuencia de tres claves qu produzca un quiebre en la direccion de la insercion.

# I1 2025-1 P2 ABB, AVL, 2-3

(III) Cuál es la menor secuencia de inserciones en un árbol 2-3 inicialmente vacío vacío que produzca un árbol 2-3 con 4 nodos tipo 3

# I1 2025-1 P2 ABB, AVL, 2-3

(III) Cuál es la menor secuencia de inserciones en un árbol 2-3 inicialmente vacío vacío que produzca un árbol 2-3 con 4 nodos tipo 3

**Solución:** 4 nodos tipo 3 son 8 claves: dos en la raíz y dos en cada uno de los 3 hijos.
P.ej., las primeras 3 claves pueden ser 10, 20 y 30; y luego 5 y 40 una raíz tipo 2 (20) y dos hijos tipo 3: (5/10) y (30/40). Luego insertamos 50, que va a parar inicialmente a (30/40, dejando momentáneamente (30/40/50), que hace que 40 suba a la raíz, convirtiéndola en un nodo tipo 3 (20/40), y que 30 y 50 queden en nodos tipo 2. Finalmente, insertamos 35 y 55 (o cualesquiera dos claves que vayan a parar a (30) y (50), respectivamente)

**Otra solución: 2,3,6,7,8,9,5,1**

# I1 2025-1 P2 ABB, AVL, 2-3

b) (2 pts) El Profesor Eterovic indica que ha descubierto una propiedad notable de los ABB. Suponga que la búsqueda de la clave k en un ABB termina en una hoja. Consideremos tres conjuntos: A, las claves a la izquierda de la ruta de búsqueda; B, las claves en la ruta de búsqueda; y C, las claves a la derecha de la ruta de búsqueda. El profesor Eterovic afirma que tres claves cualesquiera $a \in A, b \in B$ y $c \in C$ deben satisfacer $a \leq b \leq c$. ¿Será cierto? En caso afirmativo, demuestre formalmente esta propiedad; en caso negativo, muestre con un ejemplo que la propiedad no se cumple.

# I1 2025-1 P2 ABB, AVL, 2-3

b) (2 pts) El Profesor Eterovic indica que ha descubierto una propiedad notable de los ABB. Suponga que la búsqueda de la clave k en un ABB termina en una hoja. Consideremos tres conjuntos: A, las claves a la izquierda de la ruta de búsqueda; B, las claves en la ruta de búsqueda; y C, las claves a la derecha de la ruta de búsqueda. El profesor Eterovic afirma que tres claves cualesquiera $a \in A, b \in B$ y $c \in C$ deben satisfacer $a \leq b \leq c$. ¿Será cierto? En caso afirmativo, demuestre formalmente esta propiedad; en caso negativo, muestre con un ejemplo que la propiedad no se cumple.

**Solución:** La propiedad no es cierta, por lo que basta mostrar un contraejemplo: considera el ABB con raíz 4 e hijos 3 y 6, en que 6 tiene a su vez hijos 5 y 7. Considera la búsqueda del 7: ruta 4, 6, 7 $\Rightarrow$ a la izquierda quedan las claves 3 y 5, en que $5 > 4$.

# I1 2025-1 P2 ABB, AVL, 2-3

c) (2 pts) Escriba el algoritmo `CheckAVL`$(T)$, tal que dado un árbol binario de búsqueda $T$ de $n$ nodos (representado por un puntero a su raíz), entregue como resultado el par $(v, h)$ tal que:

- $v =$ `true` si el ABB cumple con las propiedades de un AVL, $v =$ `false` en otro caso.
- $h$ es la altura del ABB dado.

El árbol $T$ es un ABB, es decir, no contiene información de balance almacenada en los nodos.

*Hint:* Quizás le sea de utilidad aprovechar la definición recursiva de un ABB para pensar en un algoritmo recursivo.

# I1 2025-1 P2 ABB, AVL, 2-3

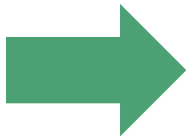
c) (2 pts) Escriba el algoritmo `CheckAVL`$(T)$, tal que dado un árbol binario de búsqueda $T$ de $n$ nodos (representado por un puntero a su raíz), entregue como resultado el par $(v, h)$ tal que:

- $v =$ `true` si el ABB cumple con las propiedades de un AVL, $v =$ `false` en otro caso.
- $h$ es la altura del ABB dado.

El árbol $T$ es un ABB, es decir, no contiene información de balance almacenada en los nodos.

*Hint:* Quizás le sea de utilidad aprovechar la definición recursiva de un ABB para pensar en un algoritmo recursivo.

**Solución:** el algoritmo es el siguiente



# I1 2025-1 P2 ABB, AVL, 2-3

**Algorithm 3:** `CheckAVL`$(T)$

```
1  begin
2      if T = ∅ then
3          return (true, 0)
4      else
5          (v1, h1) ← CheckAVL(T.izq)
6          (v2, h2) ← CheckAVL(T.der)
7          h ← 1 + máx {h1, h2}
8          if v1 AND v2 AND |h1 − h2| ≤ 1 then
9              return (true, h)
10         else
11             return (false, h)
```

Tener en cuenta que debe calcular la altura del árbol e indicar si es AVL o no. En la evaluación, dar un peso de 50 % a cada cosa.



# Ejercicio 2

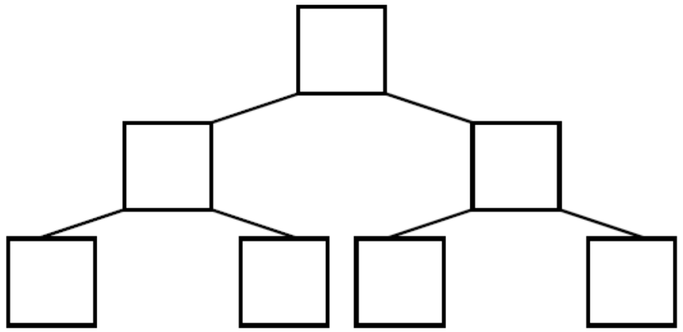
# I2 2020-1 Árbol 2-3

Los árboles 2-3 son árboles de búsqueda en que los nodos tienen ya sea una clave y dos hijos, o bien dos claves y tres hijos; y todas las hojas del árbol (que se exceptúan de la regla anterior porque no tienen hijos) están a la misma profundidad. Teniendo presente estas propiedades, responde:

a. ¿Cuál es la **altura máxima** que puede tener un árbol 2-3 con n elementos? ¿Y la **mínima**? ¿Cómo es la **estructura del árbol** cuando ocurre cada uno de estos casos?

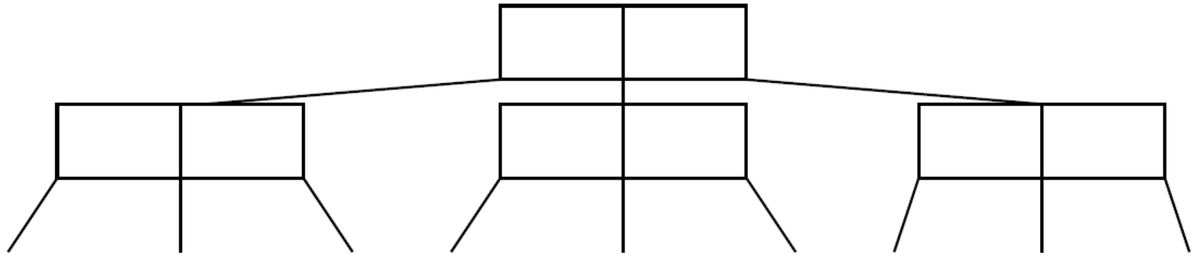
a. **Altura máxima** que puede tener un árbol 2-3 con n elementos
La altura será mayor mientras menos elementos tenga el árbol por nivel, y viceversa.

(a) Sólo nodos 2



Caso altura máxima

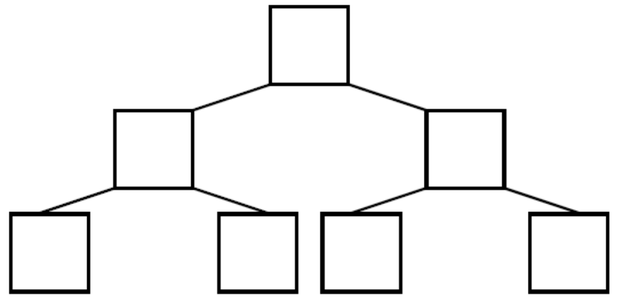
(b) Sólo nodos 3



Caso altura mínima

## Caso altura máxima

(a) Sólo nodos 2



- Cantidad elementos para el k-ésimo nivel:

$$2^{k-1}$$

- Cantidad elementos **acumulados** hasta el k-ésimo nivel:

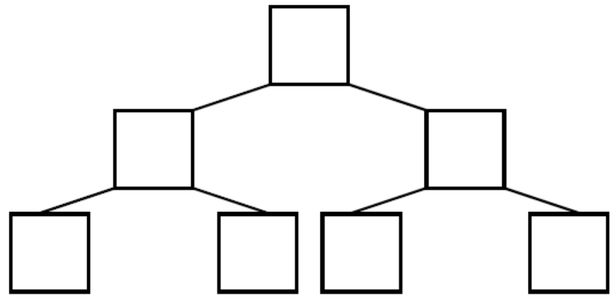
$$2^k - 1$$

Supongamos que nuestro árbol tiene n nodos y una altura h. Cuando el árbol esté completamente lleno tendremos:

$$n = 2^h - 1 \longrightarrow h = \log_2(n+1)$$

## Caso altura máxima

(a) Sólo nodos 2



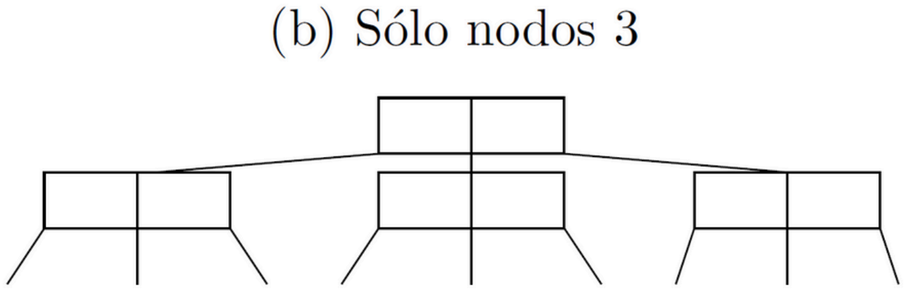
Cuando el árbol esté completamente lleno tendremos:

$$h = \log_2(n+1)$$

Como las hojas deben estar a la misma altura, al añadir un valor extra, el árbol se rebalancea manteniendo su altura, quedando:

$$\lfloor h = \log_2(n+1) \rfloor$$

## Caso altura mínima




- Cantidad elementos para el k-ésimo nivel:

$$2 \cdot 3^{k-1}$$

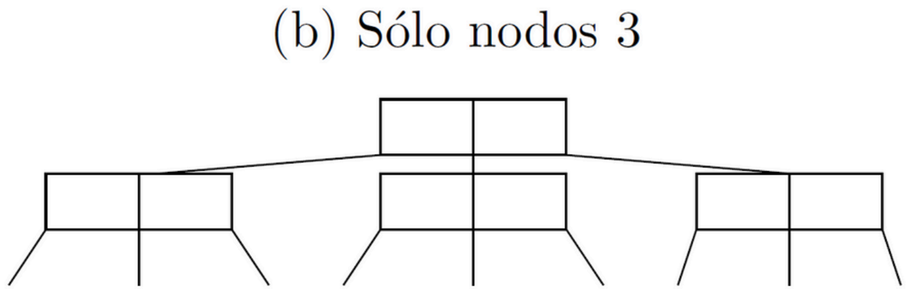
- Cantidad elementos acumulados hasta el k-ésimo nivel:

$$3^k - 1$$

Supongamos que nuestro árbol tiene n nodos y una altura h. Cuando el árbol esté completamente lleno tendremos:

$$n = 3^h - 1 \longrightarrow \mathrm{h} = \log_3(n+1)$$

**Caso altura mínima**




Cuando el árbol esté completamente lleno tendremos:

$$h = \log_3(n+1)$$

Para alcanzar un nivel “k” de altura, debe exceder necesariamente los
k - 1 niveles anteriores, y no superar los k niveles. Por lo tanto:

$$\lceil h = \log_3(n+1) \rceil$$

# I2 2020-1 Árbol 2-3

Los árboles 2-3 son árboles de búsqueda en que los nodos tienen ya sea una clave y dos hijos, o bien dos claves y tres hijos; y todas las hojas del árbol (que se exceptúan de la regla anterior porque no tienen hijos) están a la misma profundidad. Teniendo presente estas propiedades, responde:

a. Lista ✓
b. Queremos **insertar una clave X** en un **árbol 2-3** $T$ **de altura h**, que tiene **n claves**. ¿Qué debe cumplirse para que esta inserción aumente la altura de $T$? ¿Para qué **valores de n** está garantizado que sí **aumentará la altura**? ¿Para qué valores de n está garantizado que **no aumentará la altura**?

# I2 2020-1 Árbol 2-3 b)

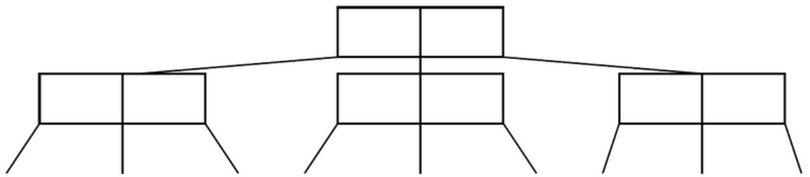
Queremos que al insertar la clave X aumente la altura del árbol 2 - 3.

- “**camino**” de un árbol T: nodos que hay que recorrer desde la raíz para llegar a un nodo hoja

  De esta forma, para que una **inserción provoque el aumento de altura** de un árbol T se debe cumplir que, en el **camino a la hoja** donde se insertará la nueva clave, **todos los nodos sean nodos 3** (es decir, que tengan 2 claves). Así, se haría un **split desde la hoja donde insertamos hasta la raíz**, aumentando finalmente la altura

**Vamos a analizar 2 casos: (altura h con n claves)**

(b) Sólo nodos 3



1. **Garantía de que aumenta la altura:**
   El árbol $T$ debe tener **solo nodos 3.** Notemos que si pasa lo contrario
   (al menos uno no es un nodo 3), entonces podemos buscar el camino en donde podamos insertar de tal forma que este nodo pase a ser nodo 3, lo que haría que no aumente la altura. Teniendo esto, la cantidad de clave
   función de la altura h es: $claves = \sum_{i=1}^{h} 2 \cdot (3^{i-1})$

en un nivel $i$ hay $3^{i-1}$ nodos, con cada uno 2 claves

$$claves = 3^h - 1$$

## 2. Garantía de que no aumenta la altura:

El árbol T **no debe tener ningún camino que tenga únicamente nodos 3**. Notemos por la definición del comienzo que la condición para que aumente la altura es que agreguemos una clave en la hoja que tenga un camino con sólo nodos 3, por lo que debemos asegurarnos que no exista tal camino. Luego, para **garantizar esto debemos contar la cantidad de claves para que no exista un árbol con dicho camino**.

## 2. Garantía de que no aumenta la altura:

La menor cantidad de claves con la que podría aumentar la altura es tal que todos sean nodos 2 excepto un camino en donde sean nodos 3. Teniendo esto, si tomamos esa cantidad de claves - 1 no podremos tener un camino con nodos 3, y garantizaremos desde esa cantidad hacia abajo que no aumente la altura.

- $\text{nivel } 1 \rightarrow 2 \text{ claves } (\text{nodo } 3)$
- $\text{nivel } 2 \rightarrow 4 \text{ claves } (\text{nodo } 3 + 2 \text{ nodos } 2)$
- $\text{nivel } 3 \rightarrow 8 \text{ claves } (\text{nodo } 3 + 6 \text{ nodos } 2)$

$$\vdots$$

- $\text{nivel } h \rightarrow 2^h \text{ claves } (\text{nodo } 3 + 2^h - 2 \text{ nodos } 2)$

$$\longrightarrow \quad claves = \sum_{i=1}^{h} 2^i - 1$$

## 2. Garantía de que no aumenta la altura:

$$claves = \sum_{i=1}^{h} 2^i - 1$$

$$claves = 2^{h+1} - 2$$

Luego, para garantizar que la inserción en el árbol T no tendrá efectos en la altura h, la cantidad de claves debe ser menor a $2^{h+1} - 2$

Además, para la factibilidad de un árbol 2-3, se debe cumplir que las claves sean mayores a $2^h - 1$ que es la cantidad mínima de claves que un árbol 2-3 de altura h puede tener.

