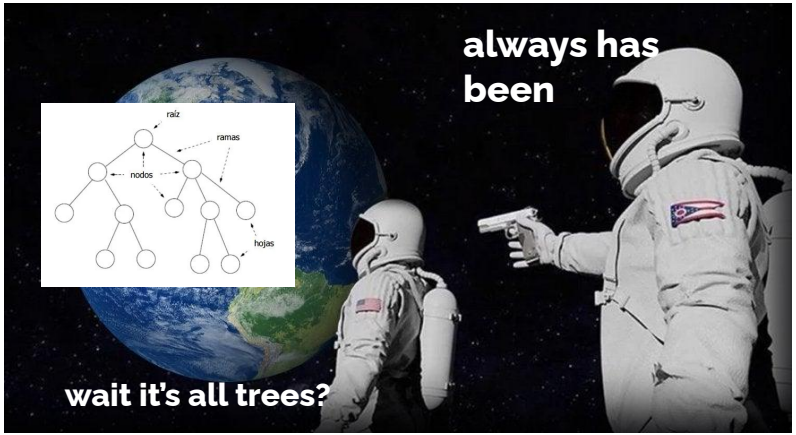
ÁRBOLES

Árboles y Árboles Binarios

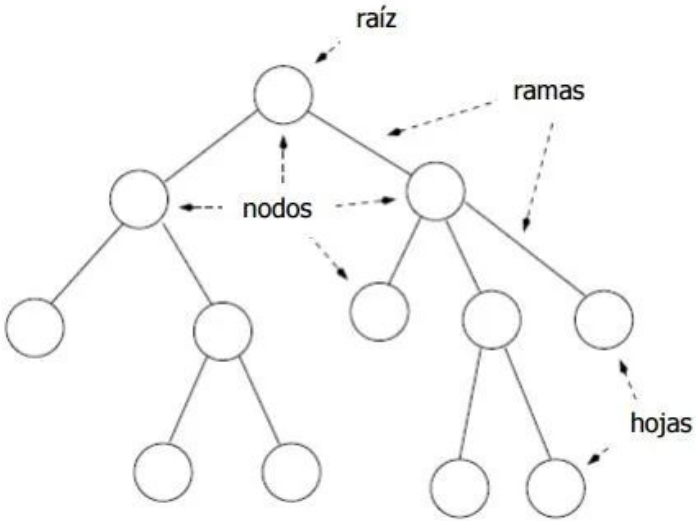
Francisca Aguilera
Anita Marti

Contenidos a revisar

BST - AVL



Que es un árbol



- Estructura de datos con nodos
- Tiene una raíz
- Cada nodo puede tener Cero o más hijos
- Cada nodo tiene un padre (menos el nodo raíz)
- No hay ciclos
- Se arman con normas pre-establecidas

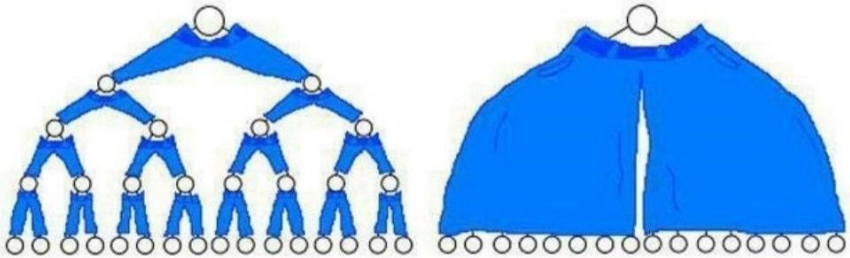
Binary Search Tree (BST)

Propiedades:

- Cada nodo tiene a lo más 2 hijos.
- Altura mínima de la rama más larga es O(log(n))
- Altura máxima de O(n), puede no estar balanceado
- Los hijos a la derecha de un nodo son todos mayores a dicho nodo mientras que los de la izquierda son todos menores

If a binary tree wore pants would he wear them

like this or like this?

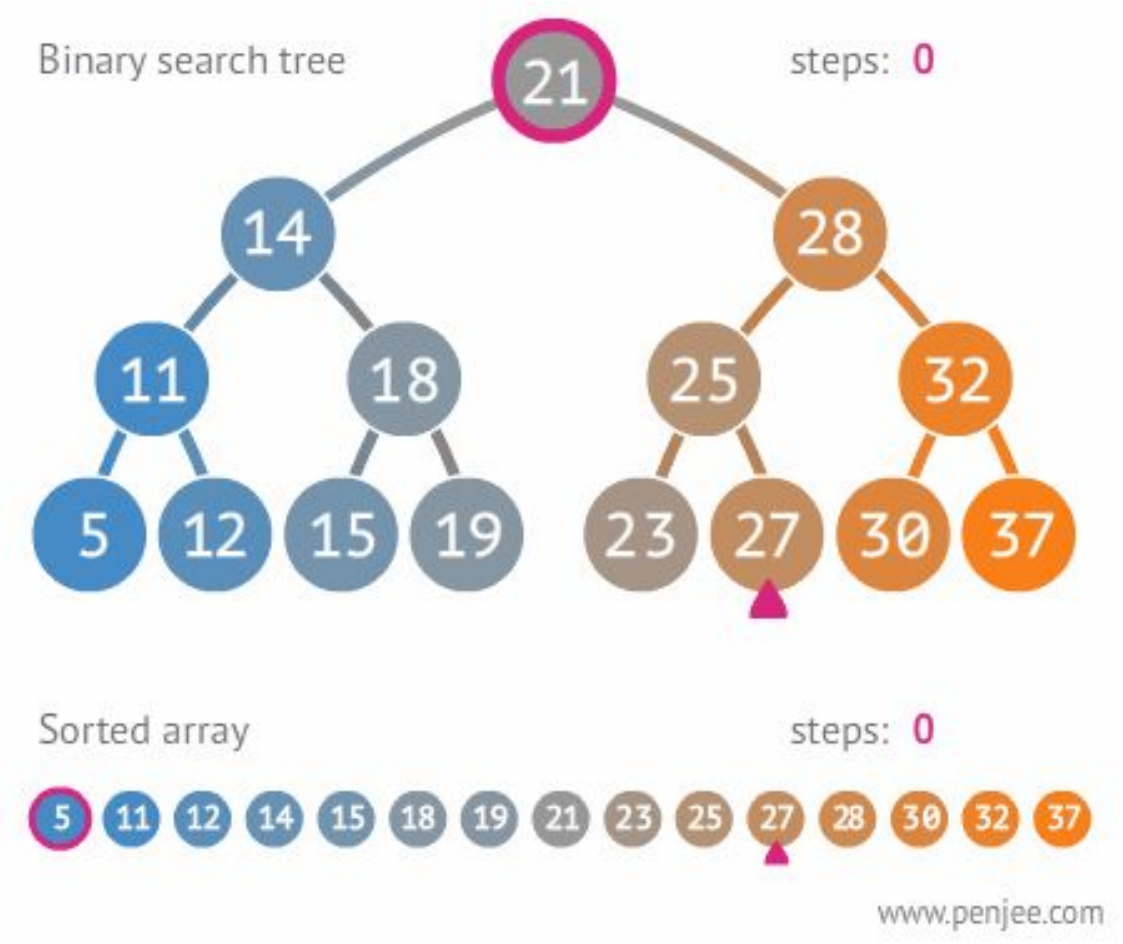


Árbol AVL

Propiedades:

- Es un tipo de binary search tree (BST)
- Está balanceado
- Las alturas de los hijos de la raíz difieren a lo más de 1 entre ellos
- Cada hijo es AVL
- Altura es log(n)

Árbol AVL

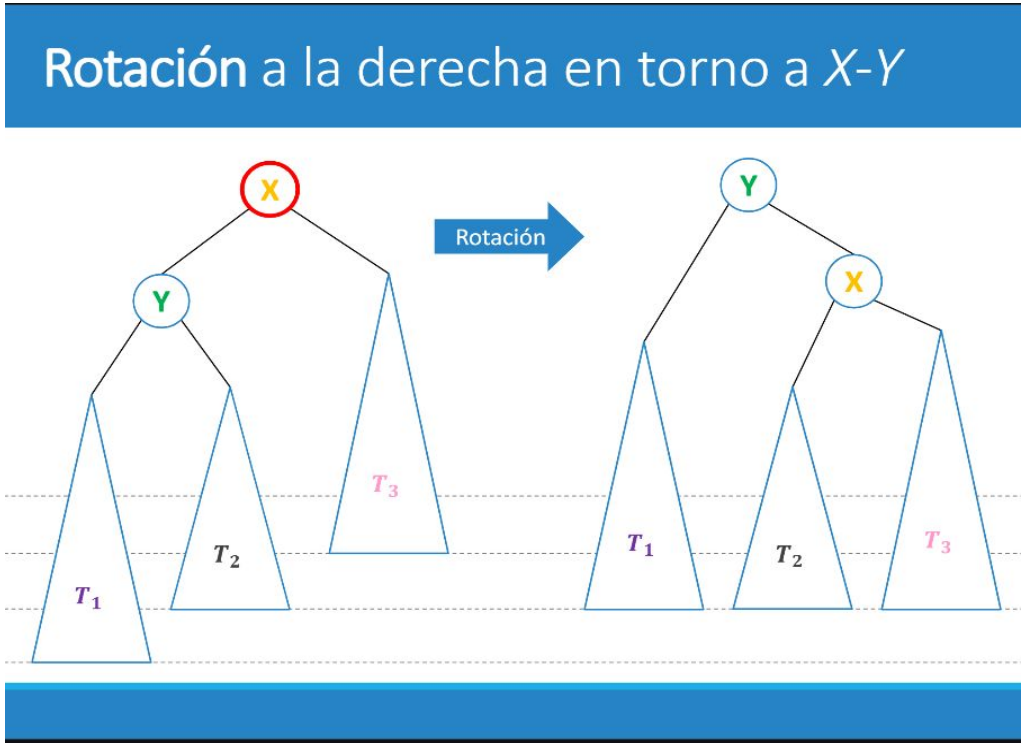

Binary search tree
steps: 0
21
14
28
11
18
25
32
5
12
15
19
23
27
30
37
Sorted array
steps: 0
5 11 12 14 15 18 19 21 23 25 27 28 30 32 37
www.penjee.com

Problema 1

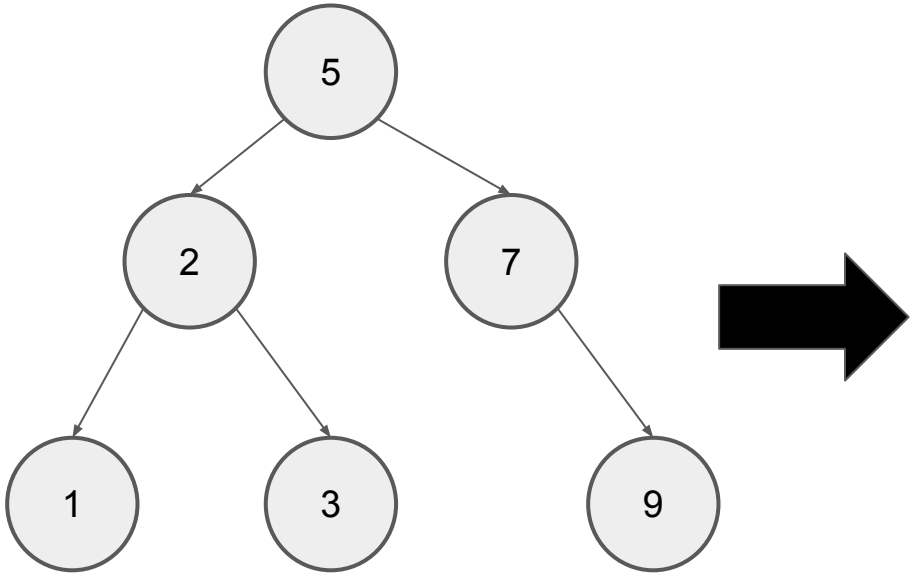
- Encuentre un forma de pasar de un árbol **BST** a otro árbol **BST** (que tengan los mismo elementos), por medio de rotaciones en O(n).

- Encuentre una forma de convertir un árbol BST a un array ordenado en O(n).

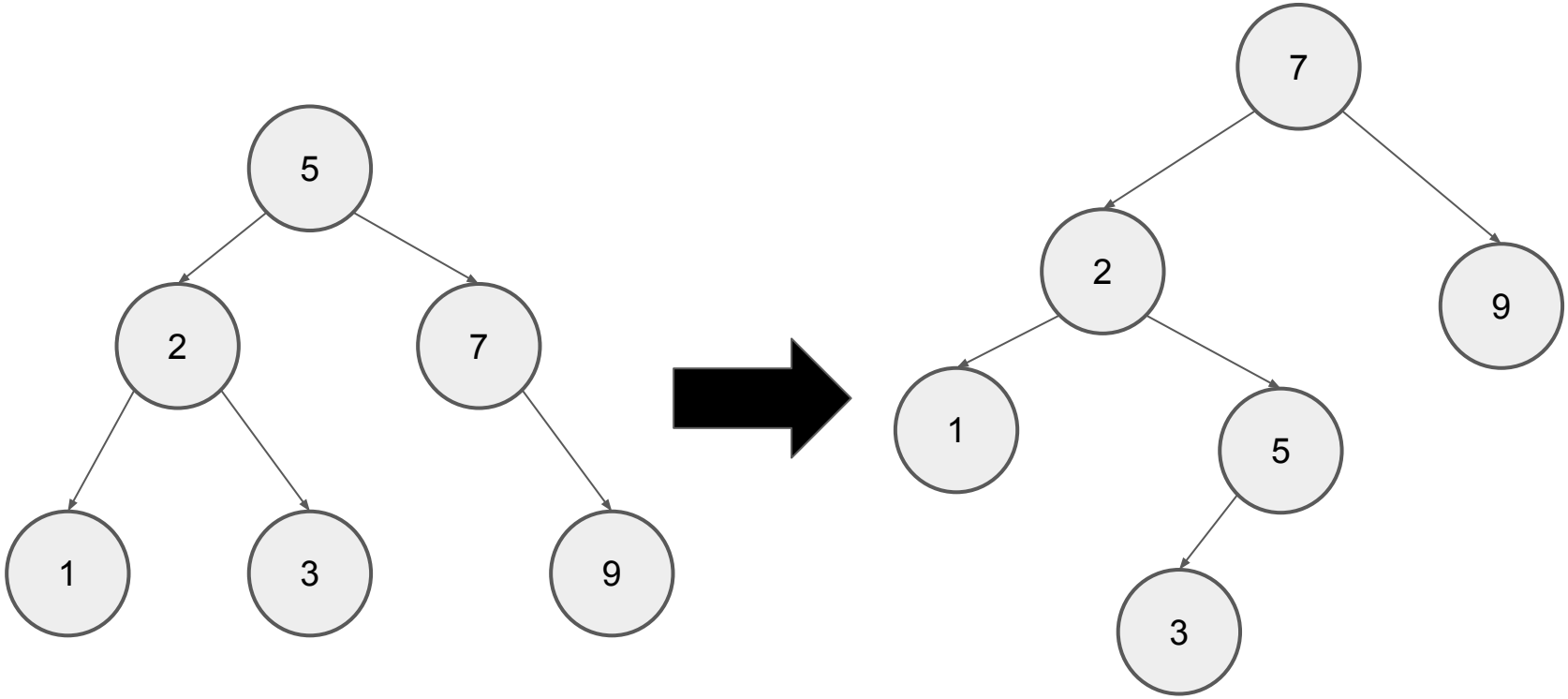
Recordemos la rotación



Problema 1



Problema 1



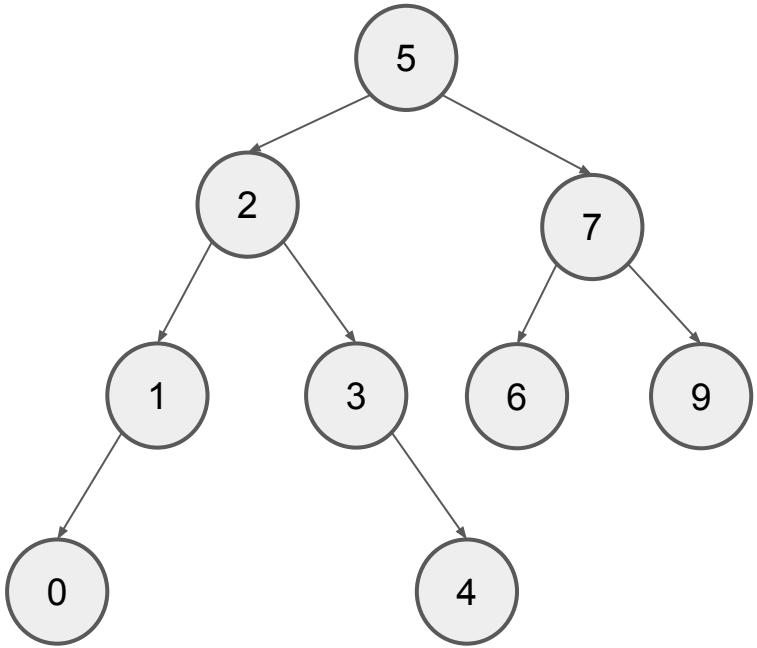
Problema 1

- Encuentre un forma de pasar de un árbol BST a otro árbol BST (que tengan los mismo elementos), por medio de rotaciones en O(n).

- Encuentre una forma de convertir un árbol BST a un array ordenado en O(n).

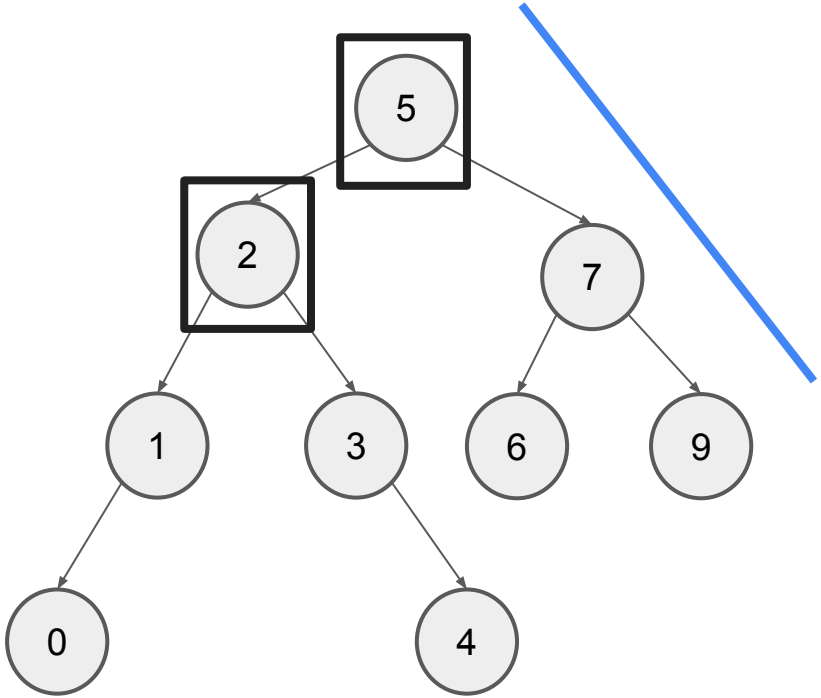
  ¿Por qué resolver el primer problema me permite resolver el otro?

Problema 1

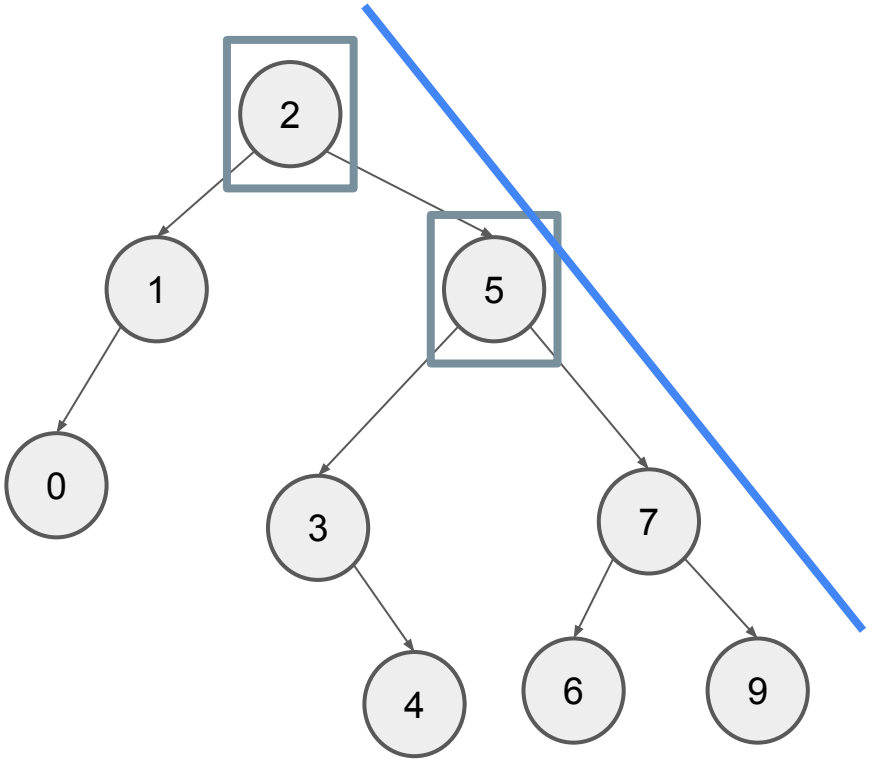


Problema 1

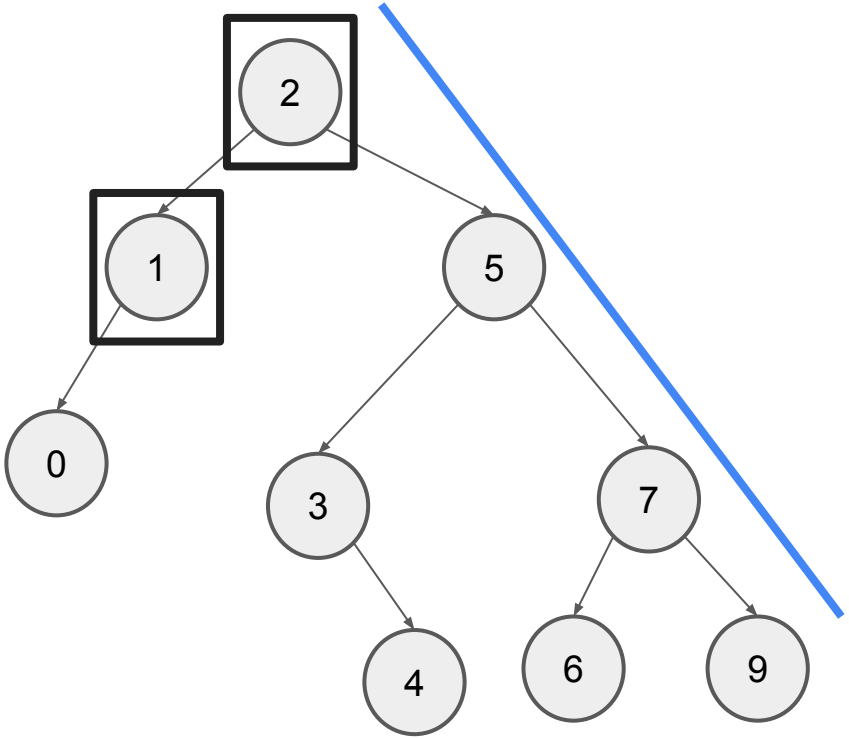
Vamos a hacer rotaciones siempre en una misma dirección desde el lugar más alto dentro del árbol (donde sea posible hacer una rotación)



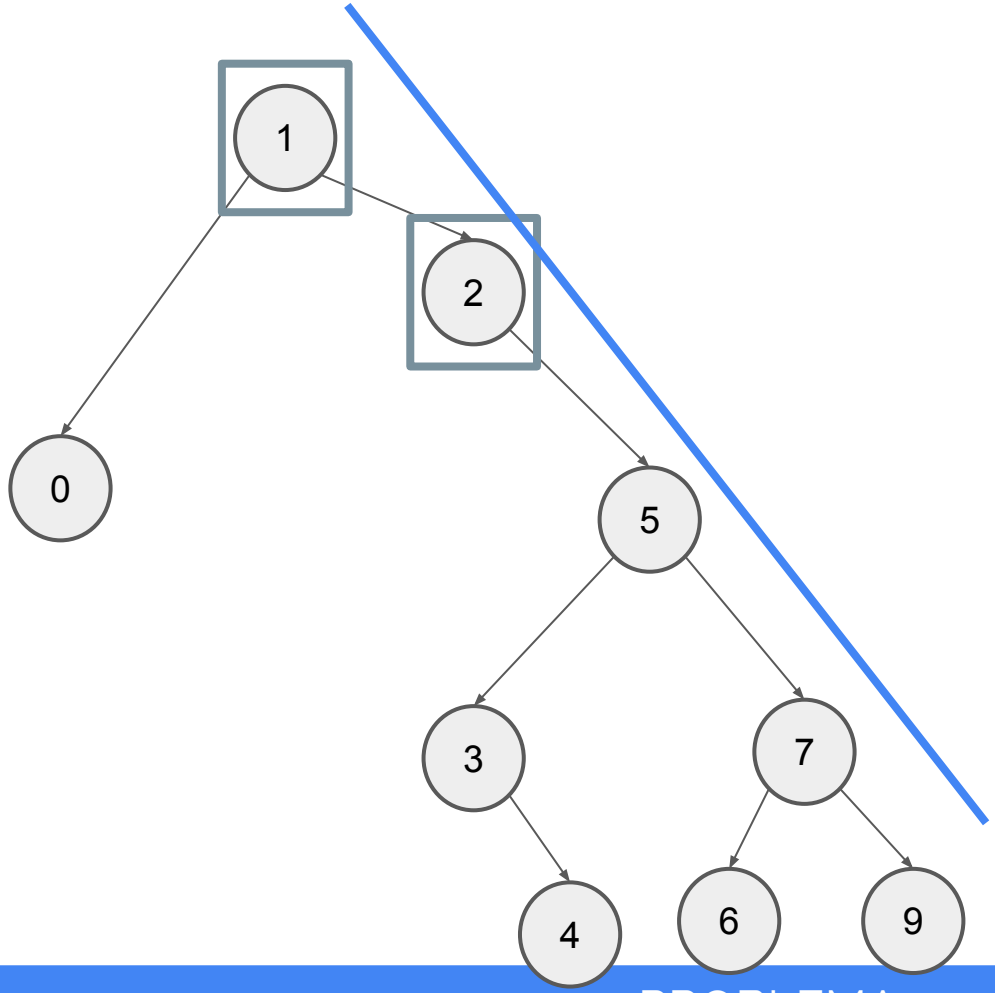
Problema 1



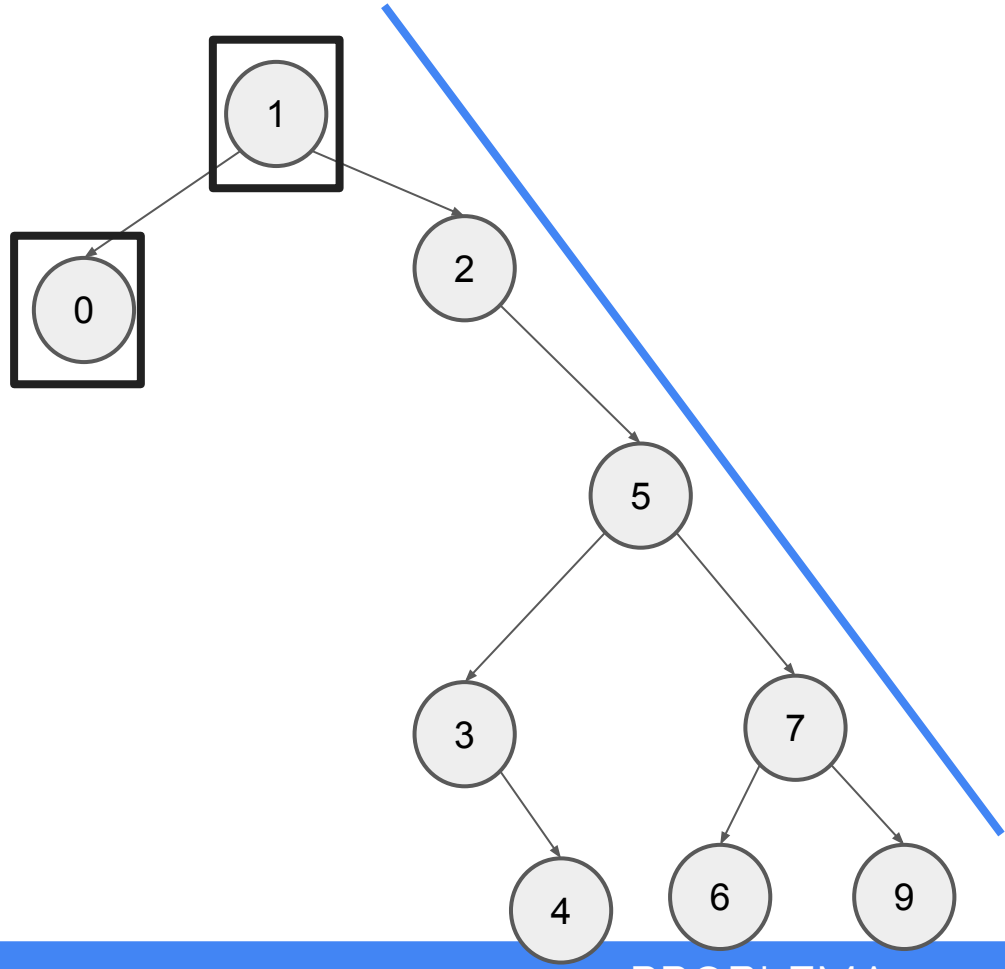
Problema 1



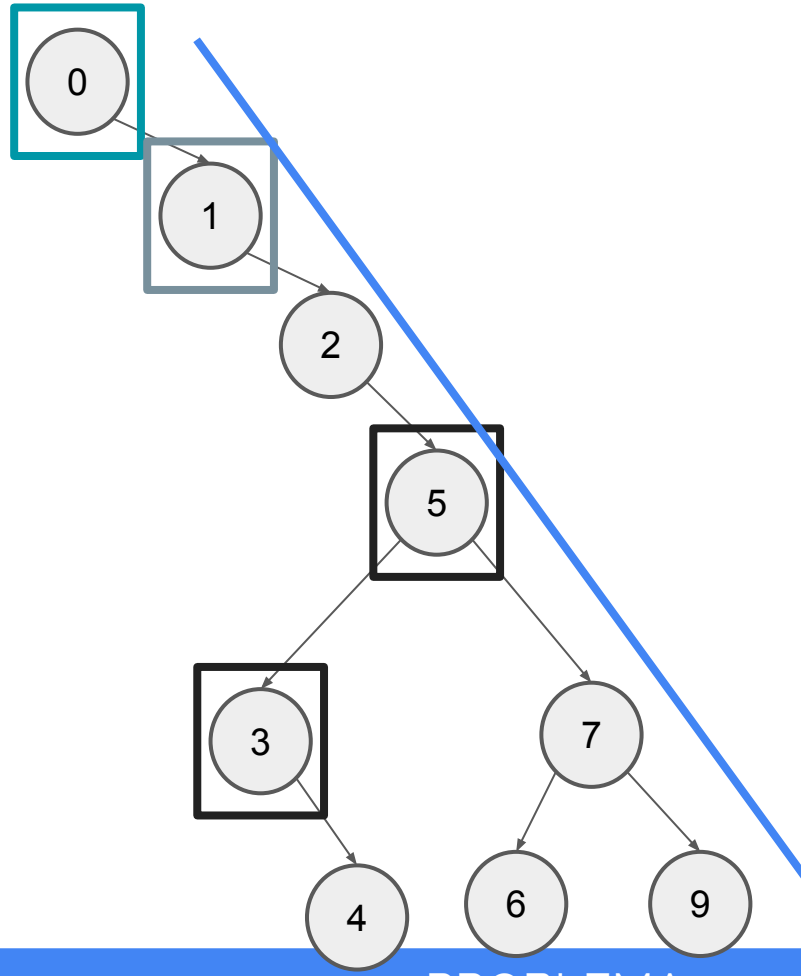
Problema 1



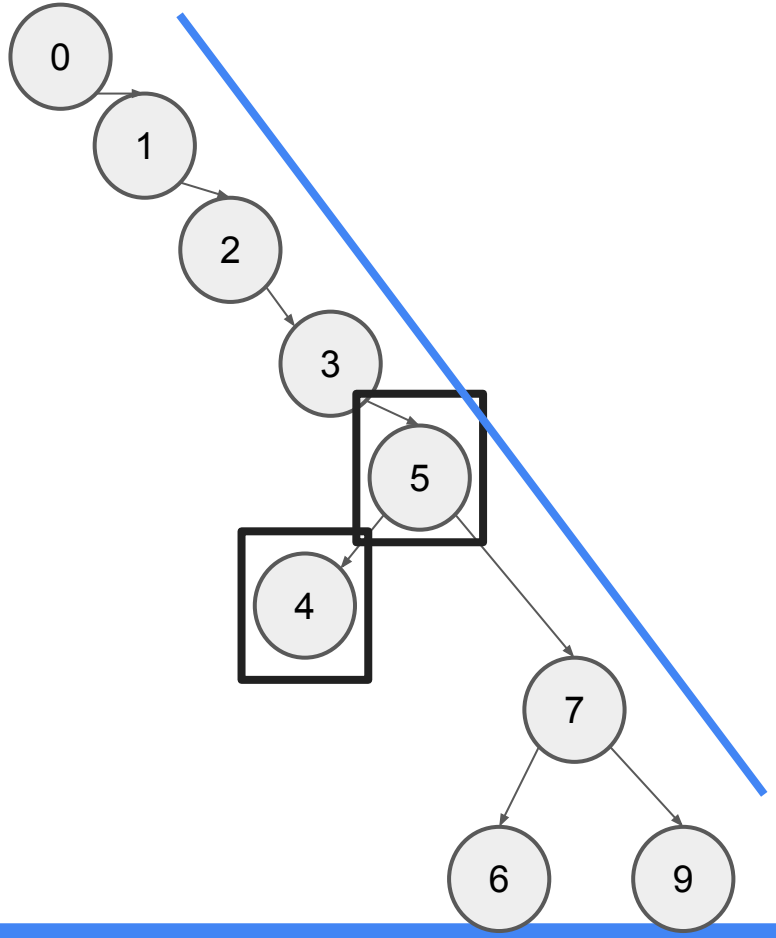
Problema 1



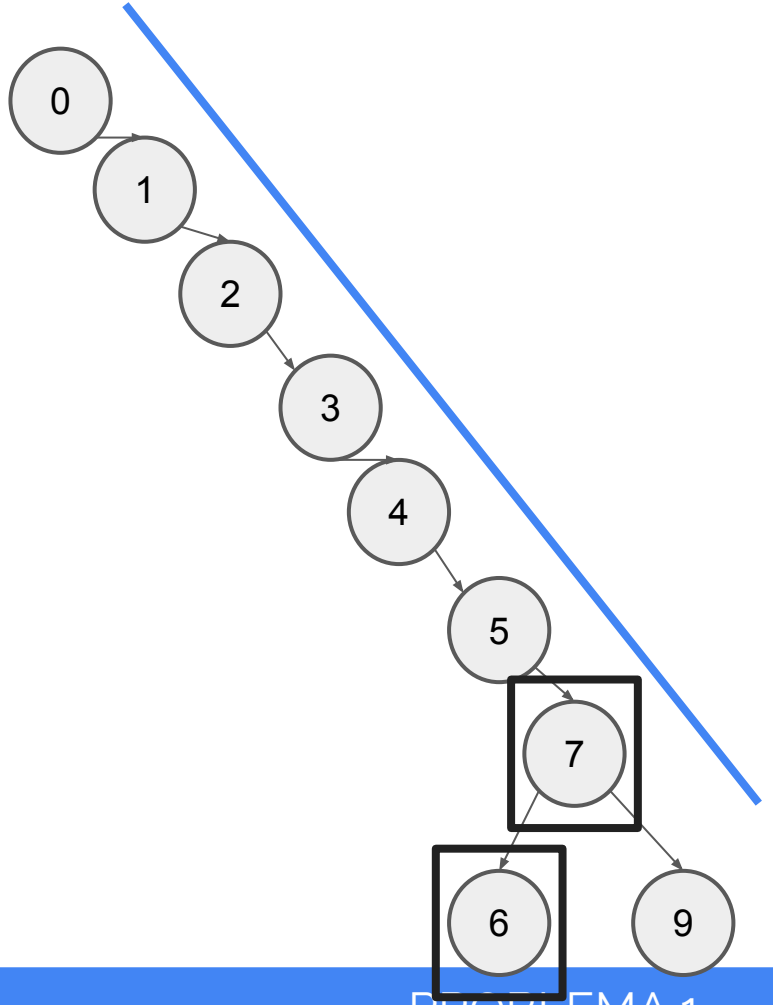
Problema 1



Problema 1



Problema 1



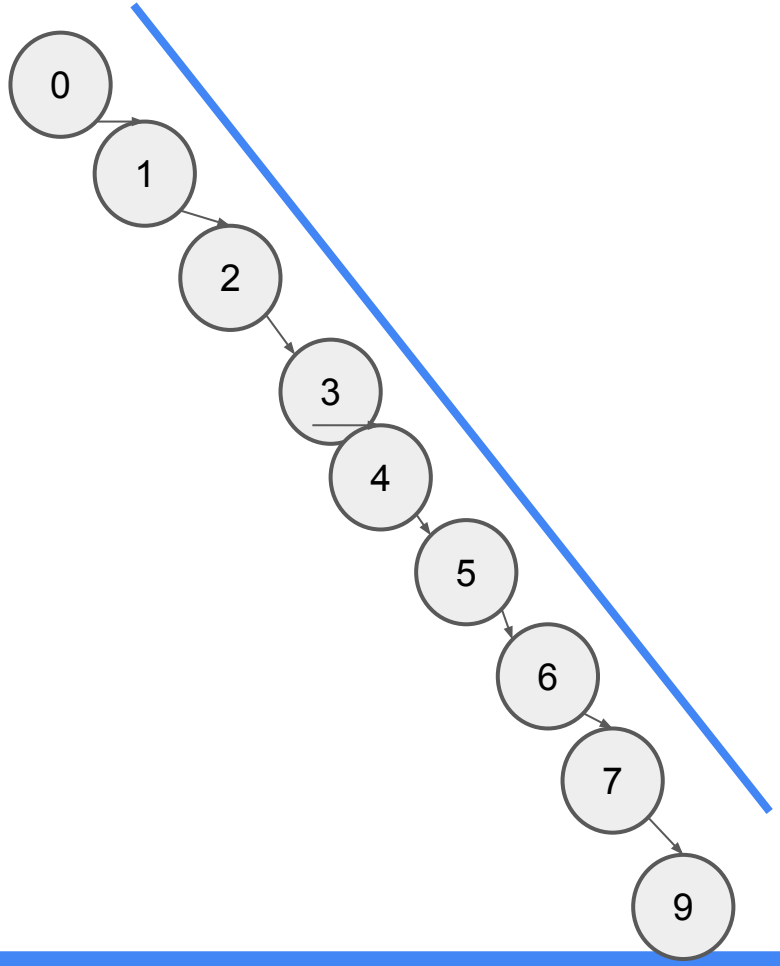
Problema 1

Lo logramos!

¿Por qué funciona?

¿Por qué funciona en O(n)?

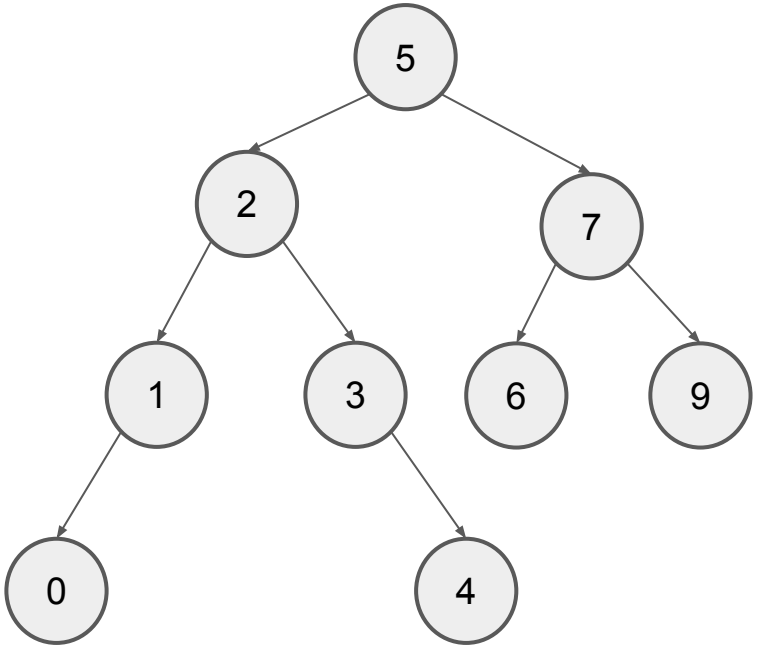
¿Como nos permite concluir el problema?



Formas de recorrer un árbol

InOrder
Izquierdo
raíz
derecho

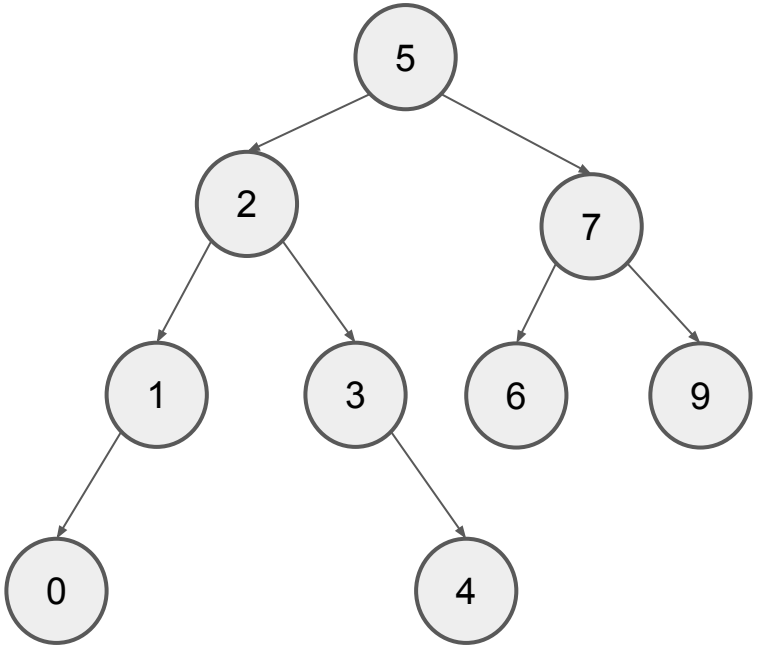
0 1 2 3 4 5 6 7 9



Formas de recorrer un árbol

Preorder
raíz
izquierdo
derecho

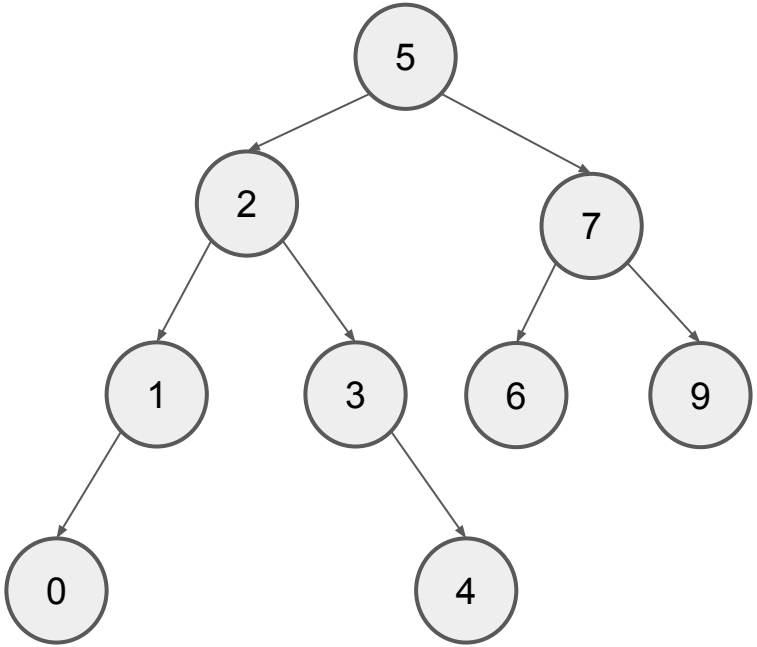
5 2 1 0 3 4 7 6 9



Formas de recorrer un árbol

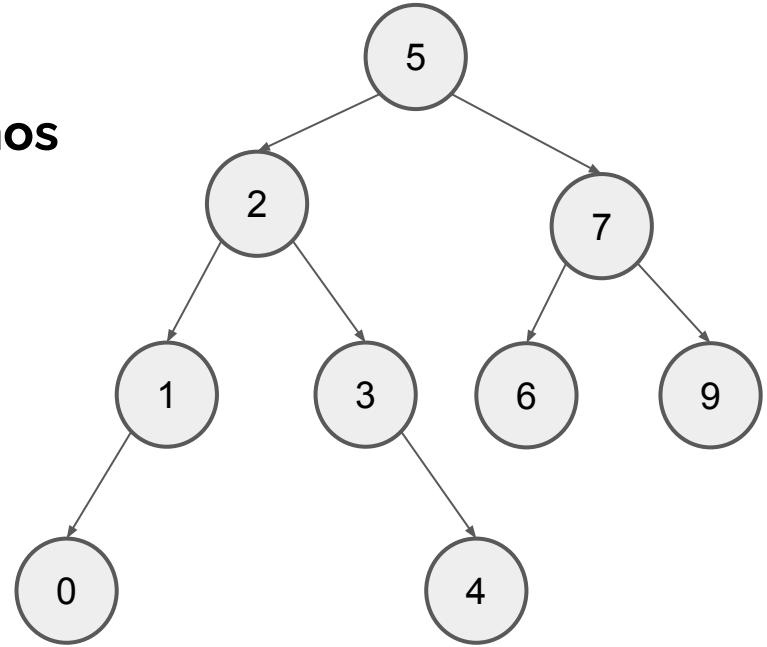
PostOrder
izquierdo
derecho
raíz

0 1 4 3 2 6 9 7 5



Problema 2

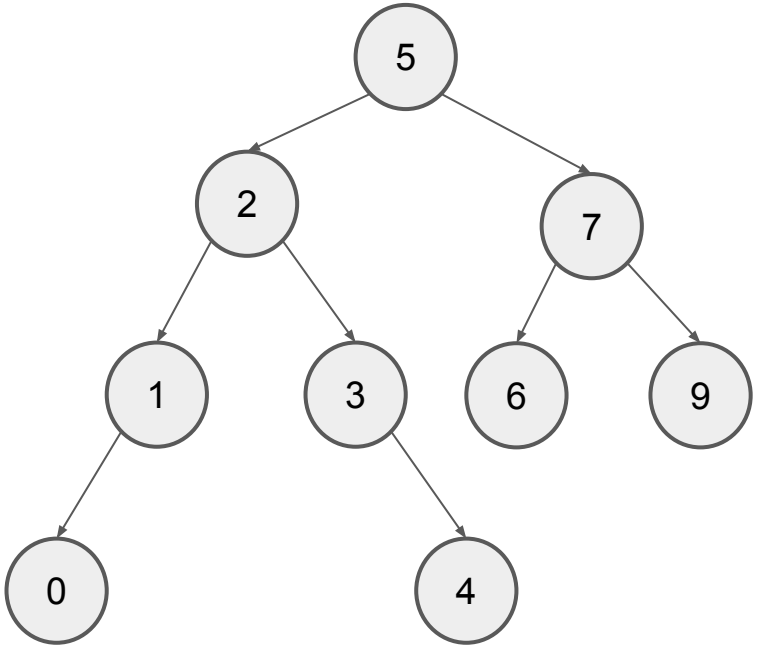
¿Cómo puedo saber la suma de los K elementos más grandes o más pequeños en un BST? (tiempo menor a O(n))



Problema 2

En el caso de necesitar los k menores elementos

- Desde el menor elemento, realizar un recorrido in-order
- establecer un contador



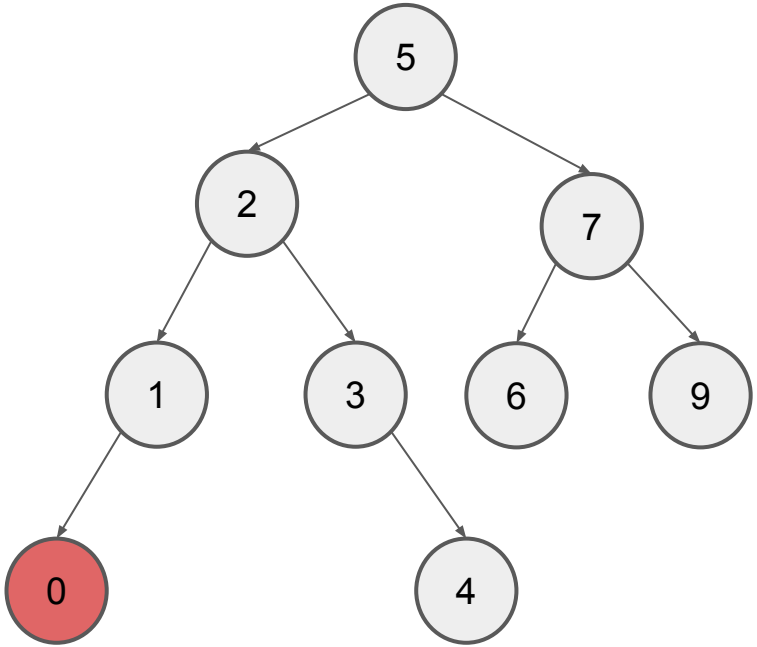
Problema 2

Para los 6 menores elementos

- **nodo izquierdo**
- nodo raíz
- nodo derecho

Suma = 0

Contador = 1



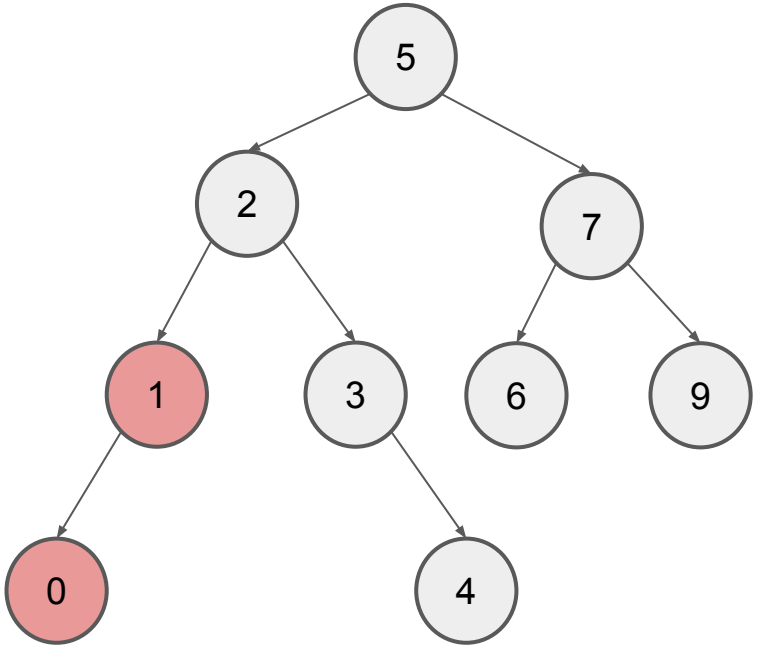
Problema 2

Para los 6 menores elementos

- nodo izquierdo
- **nodo raíz**
- nodo derecho

Suma = 1

Contador = 2



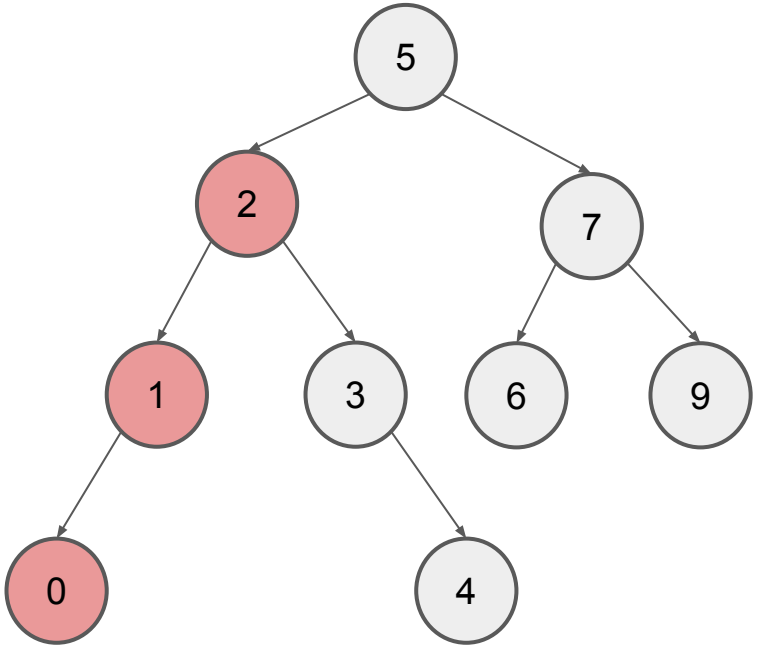
Problema 2

Para los 6 menores elementos

- nodo izquierdo
- **nodo raíz**
- nodo derecho

Suma = 3

Contador = 3



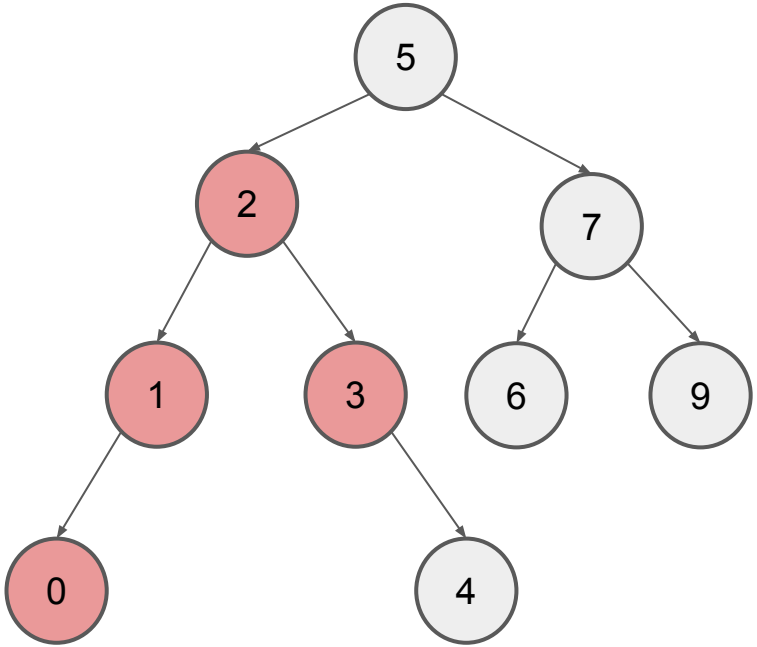
Problema 2

Para los 6 menores elementos

- nodo izquierdo
- nodo raíz
- **nodo derecho**

Suma = 6

Contador = 4



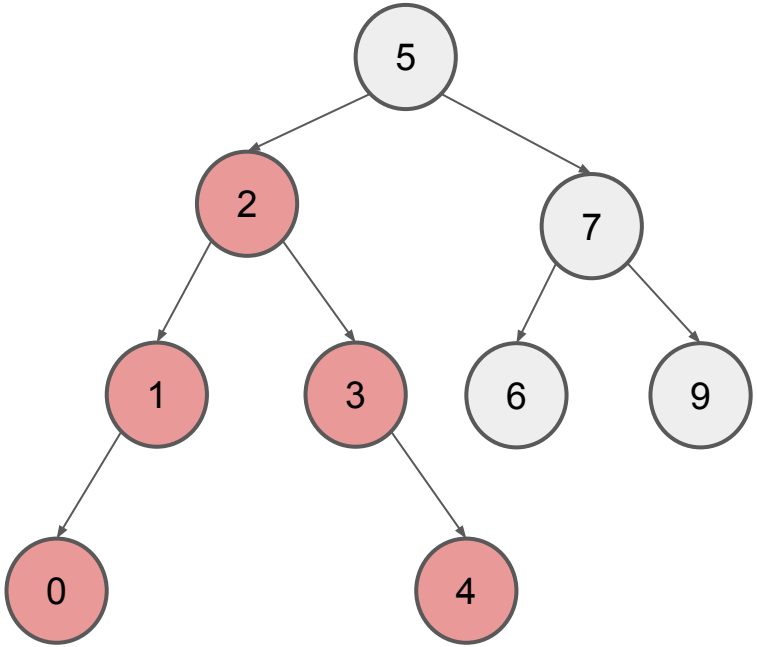
Problema 2

Para los 6 menores elementos

- nodo izquierdo
- nodo raíz
- **nodo derecho**

Suma = 10

Contador = 5



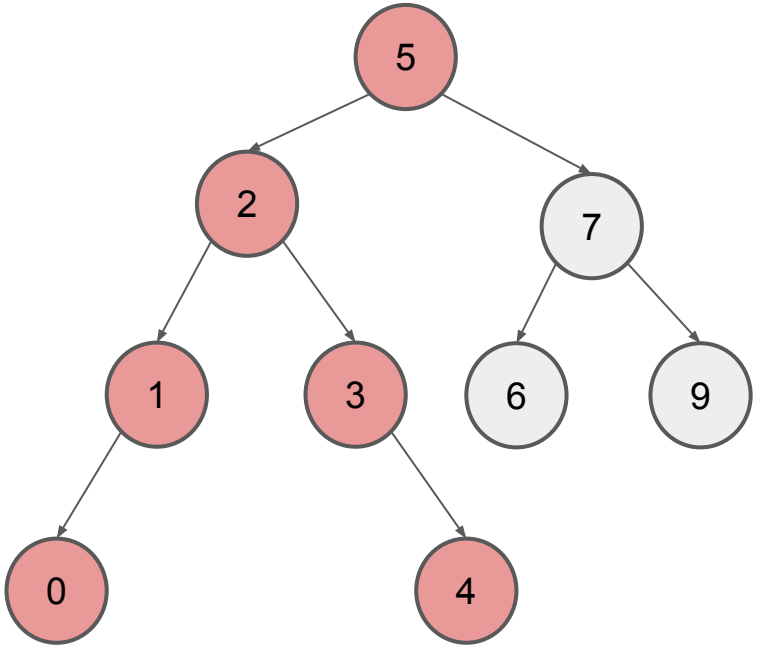
Problema 2

Para los 6 menores elementos

- nodo izquierdo
- **nodo raíz**
- nodo derecho

Suma = 15

Contador = 6



Problema 3

Muestre la secuencia de árbol **AVL** que se forma al insertar las claves 3, 2, 1, 4, 5, 6, 7, 16, 15 y 14, en este orden, en un árbol AVL inicialmente vacío.

Problema 3

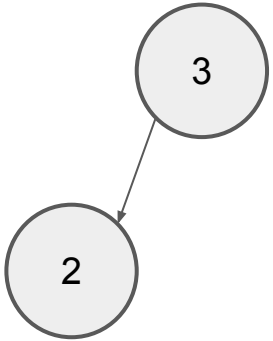
[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]

Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]

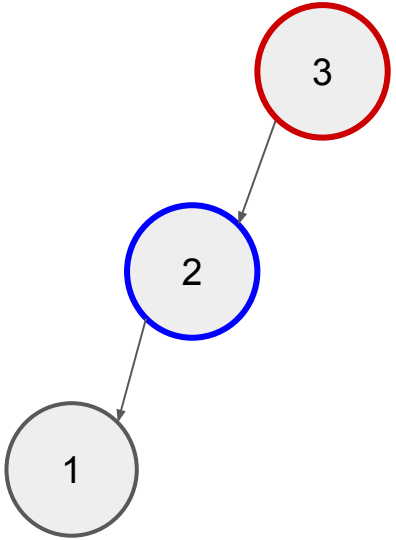
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



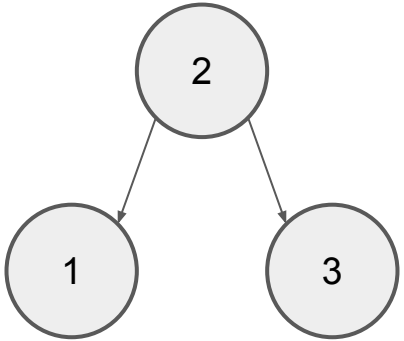
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



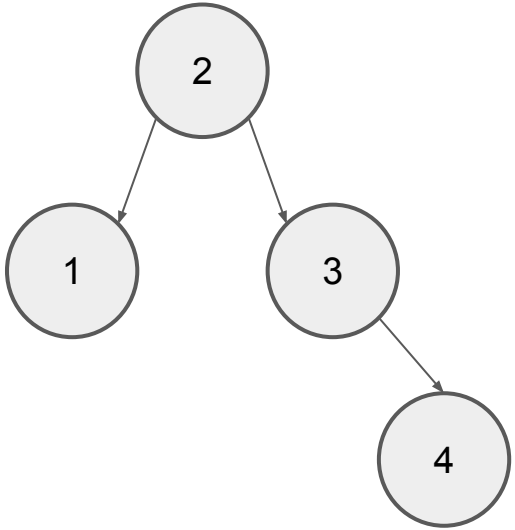
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



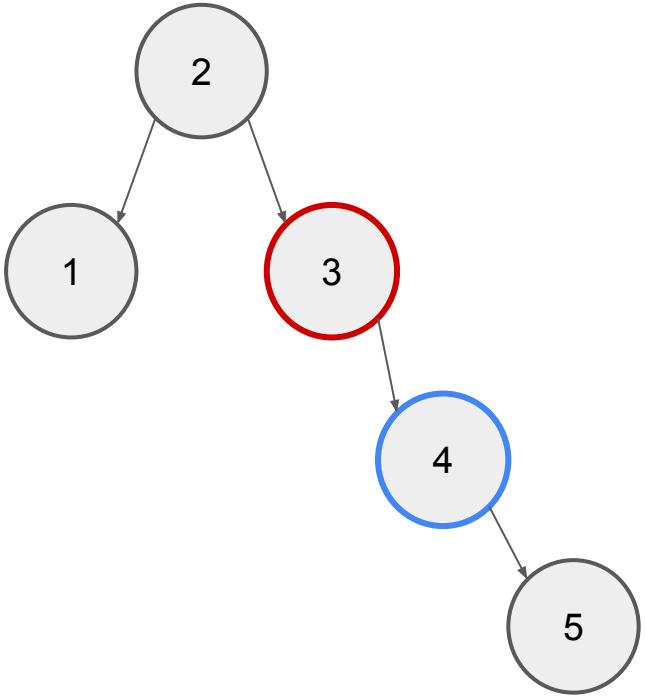
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



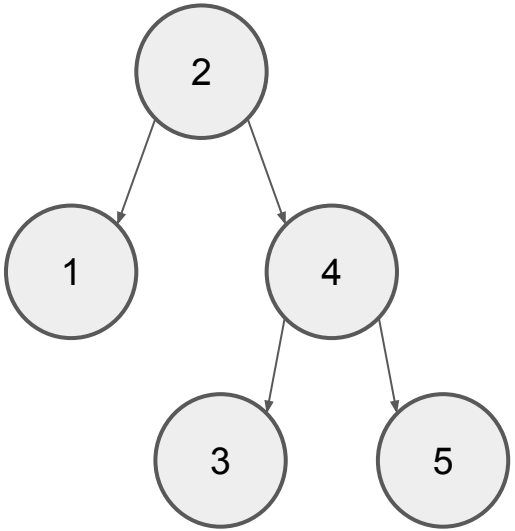
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



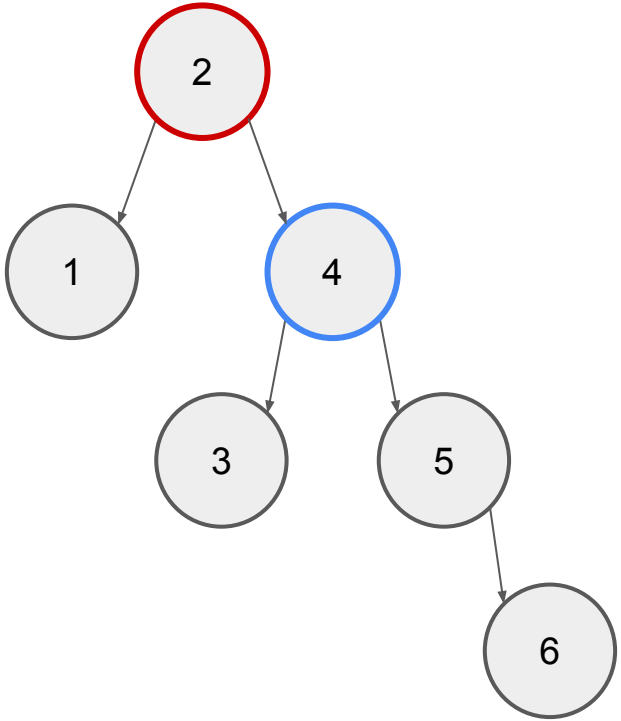
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



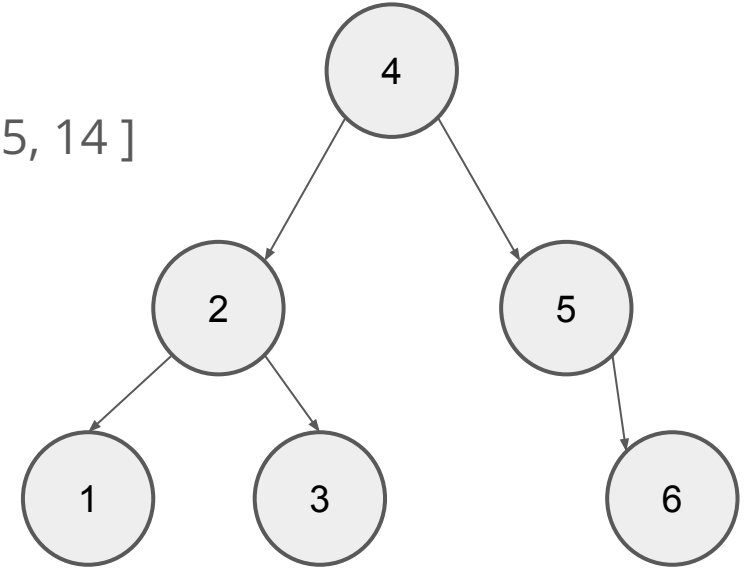
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



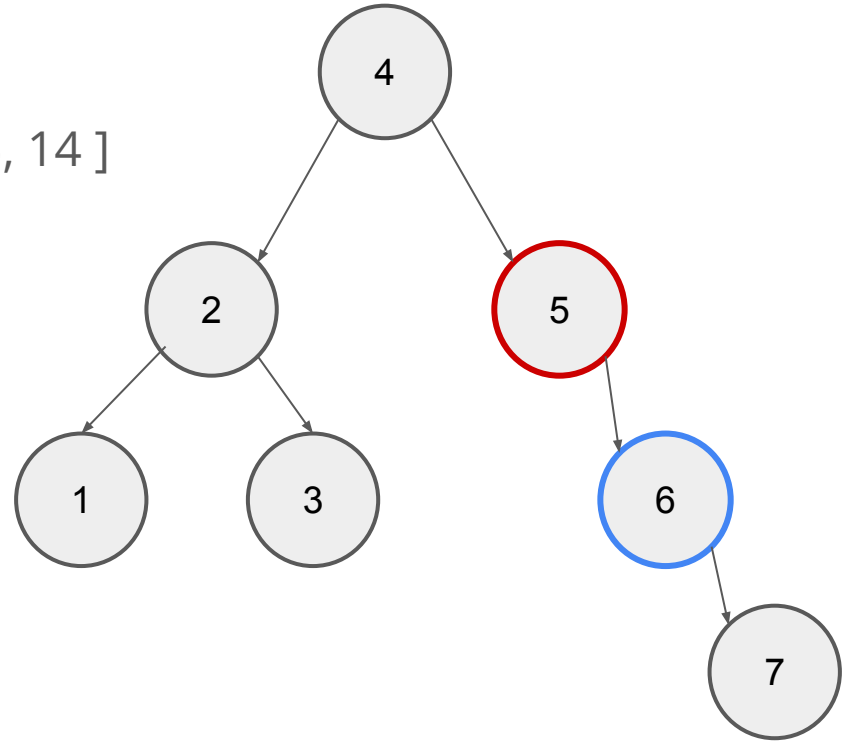
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



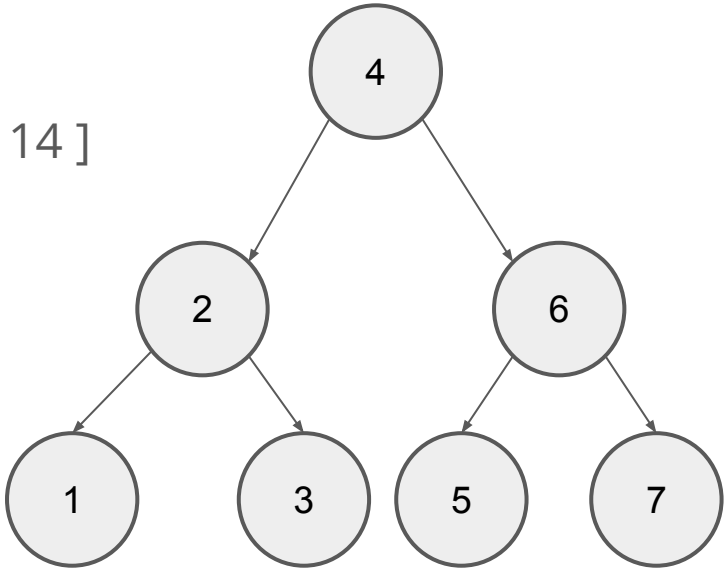
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



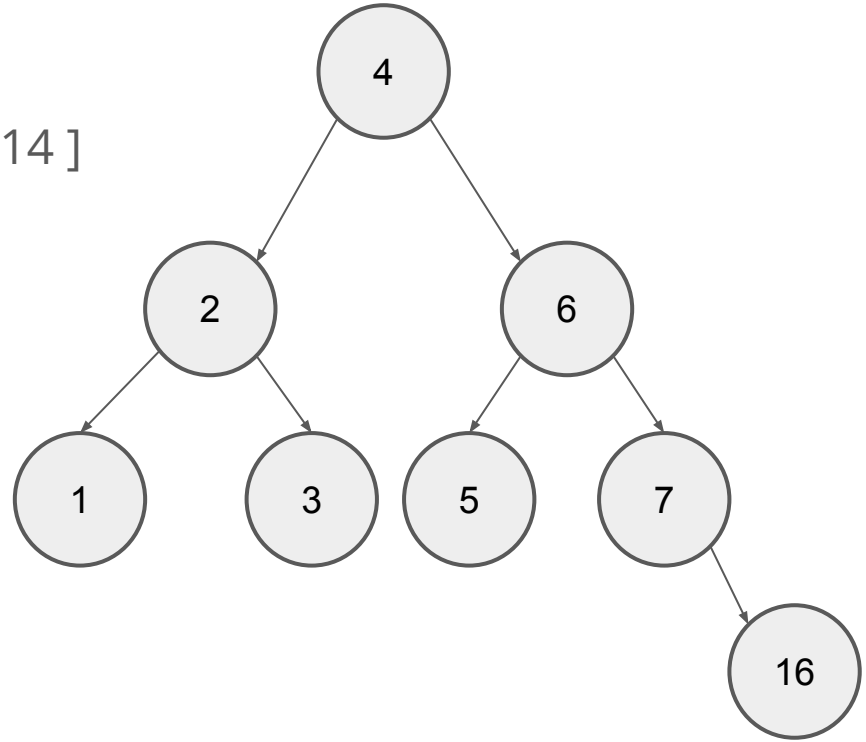
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



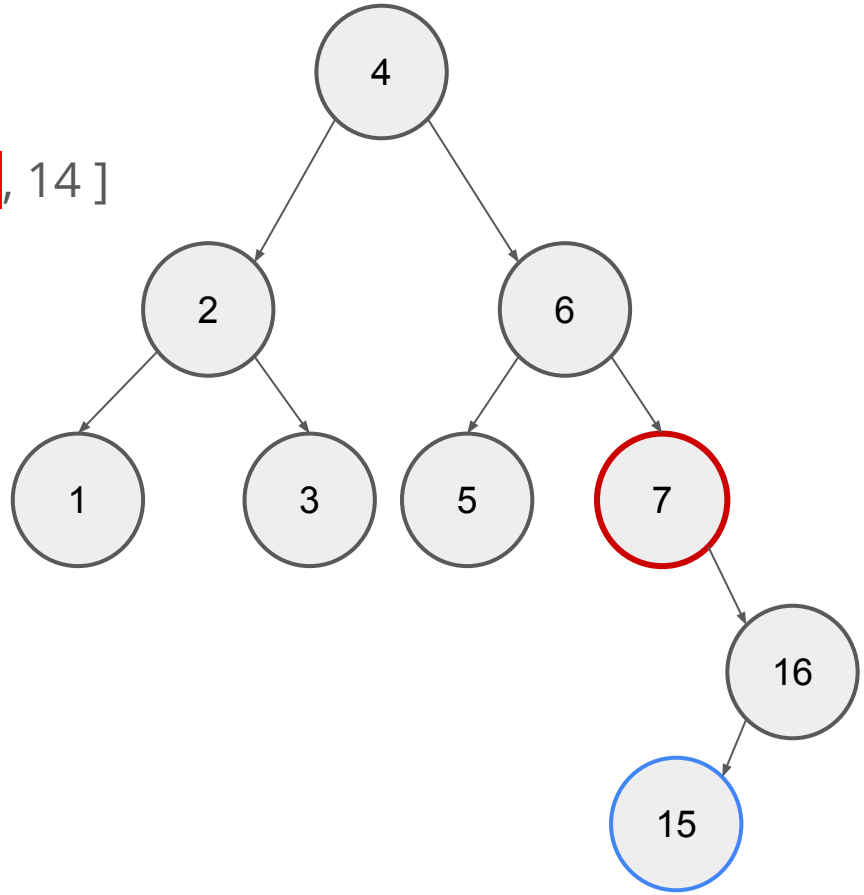
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



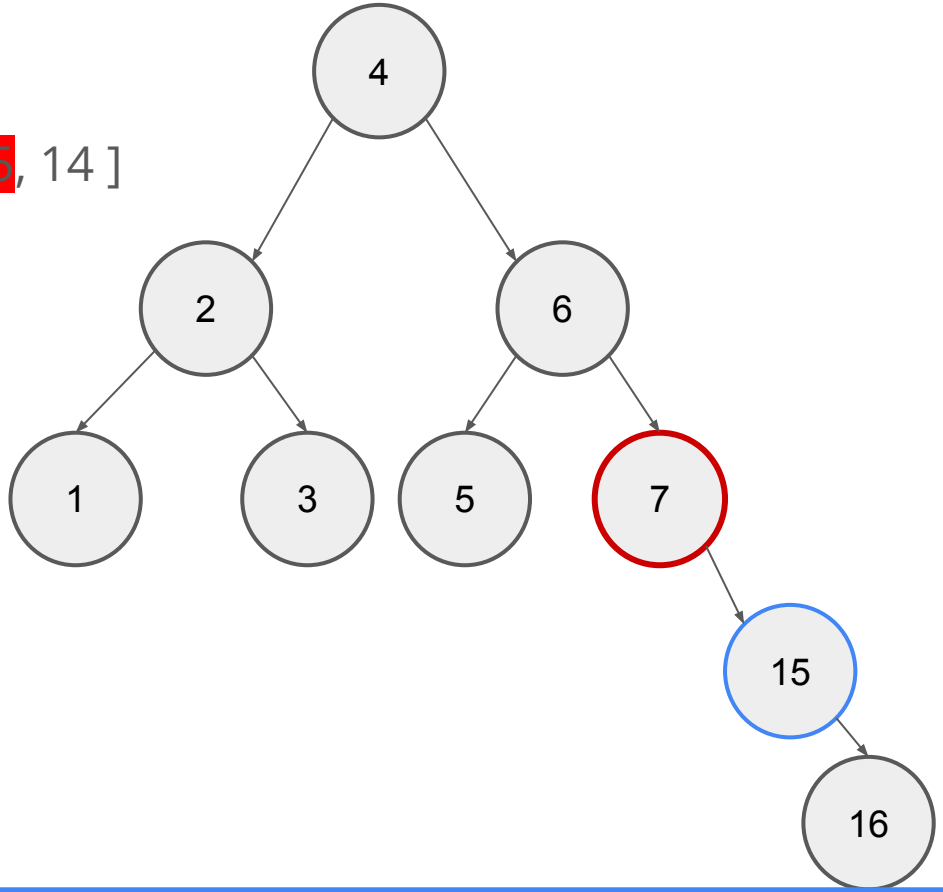
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



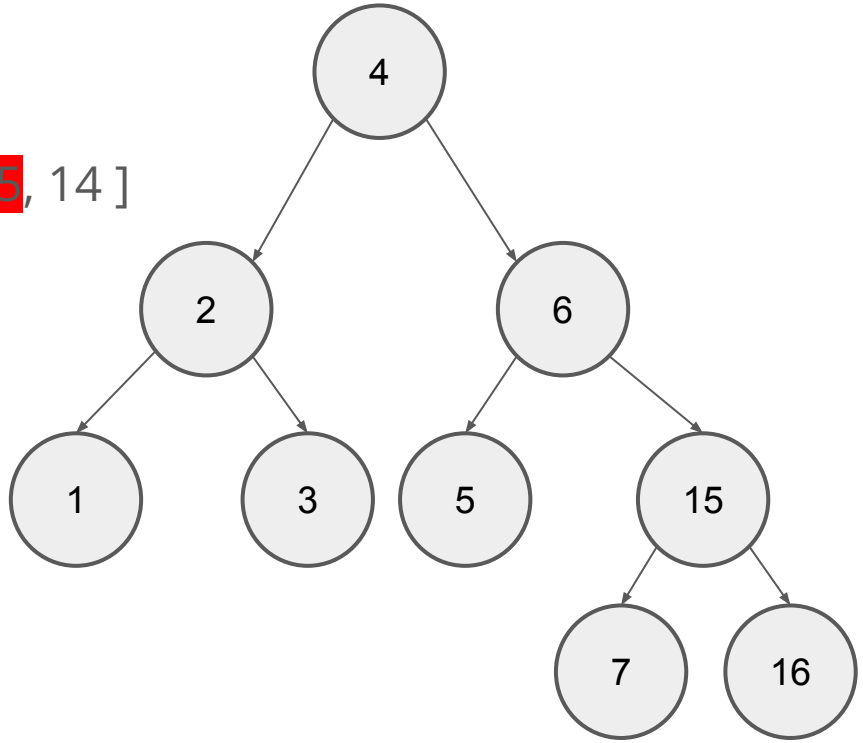
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



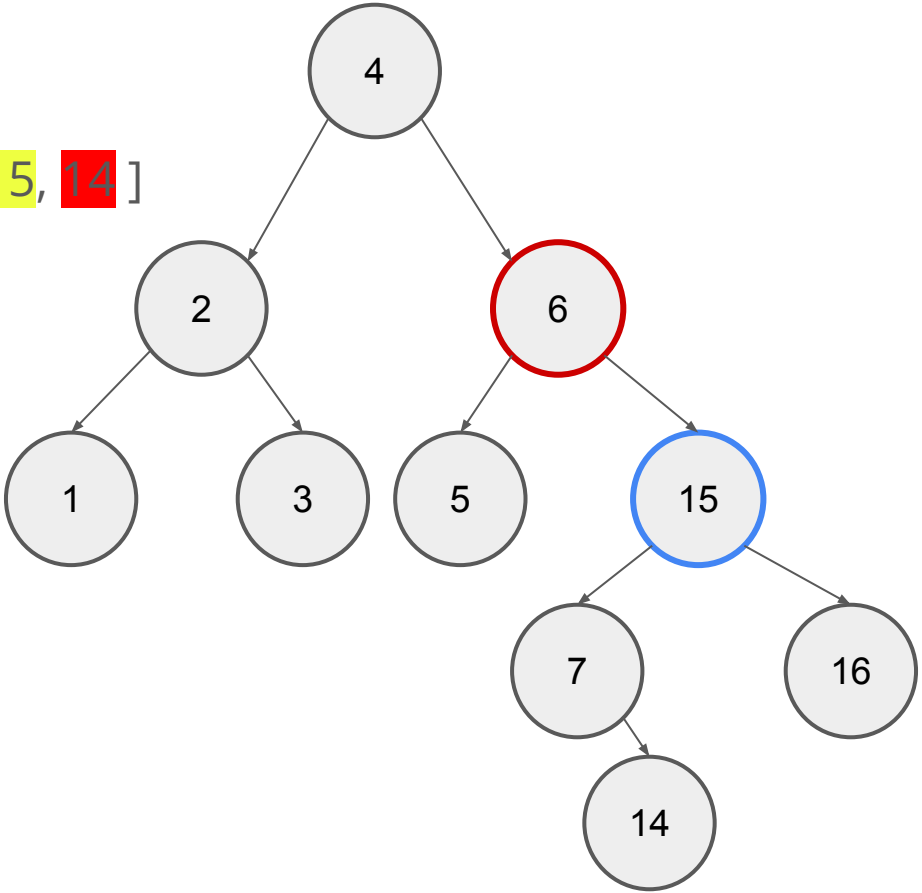
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



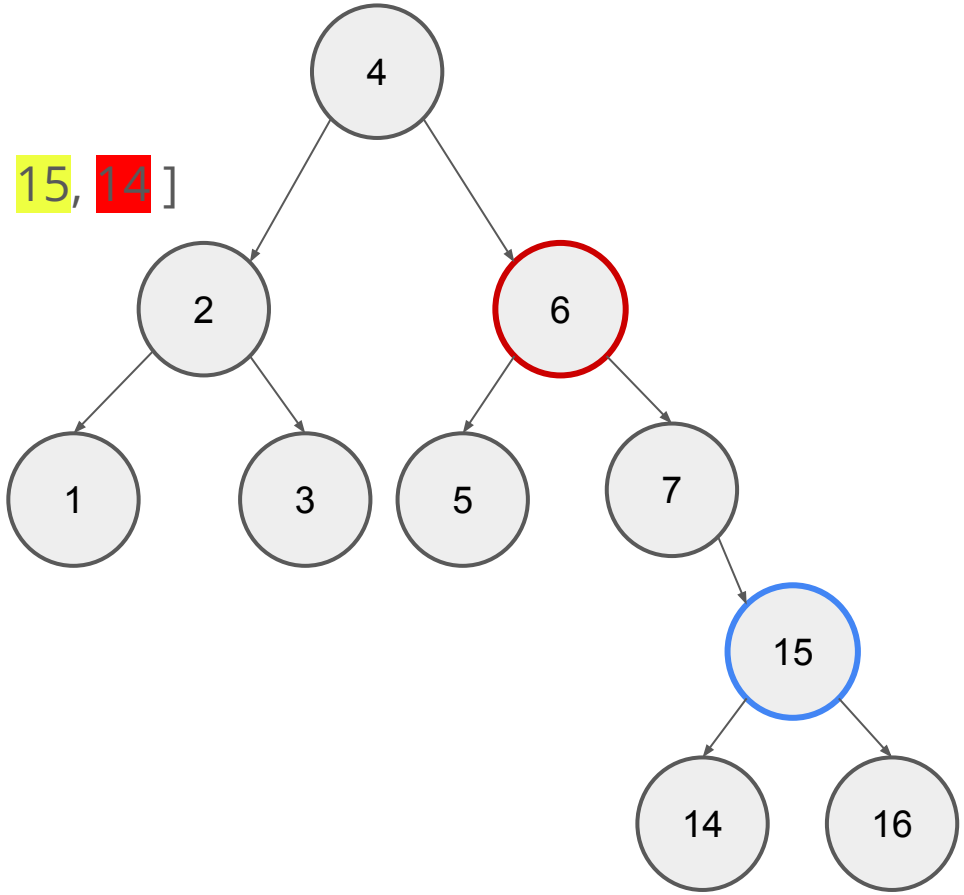
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



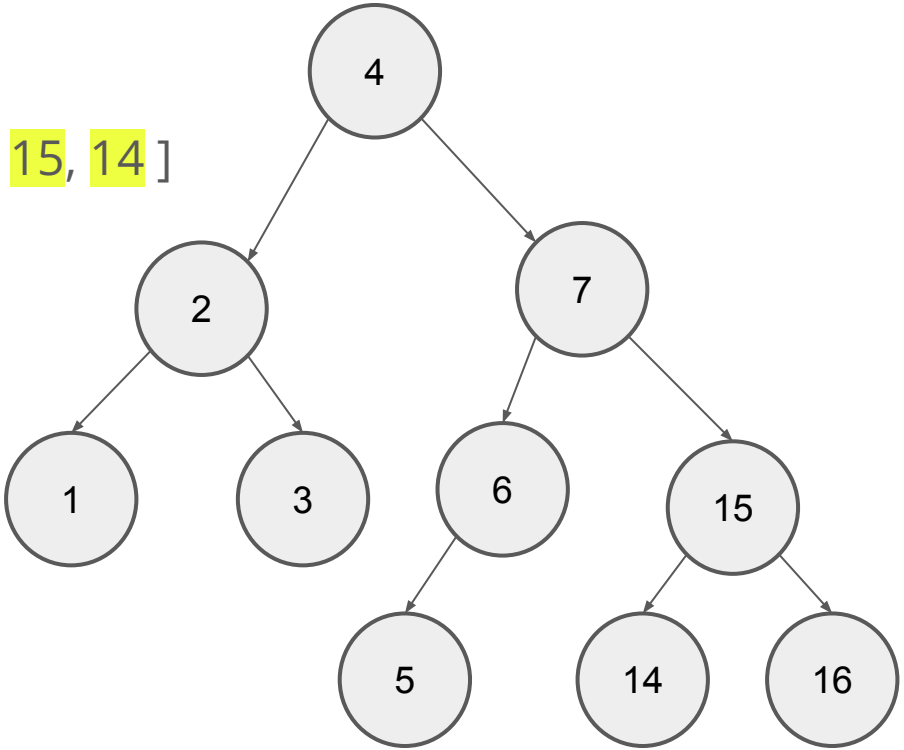
Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]

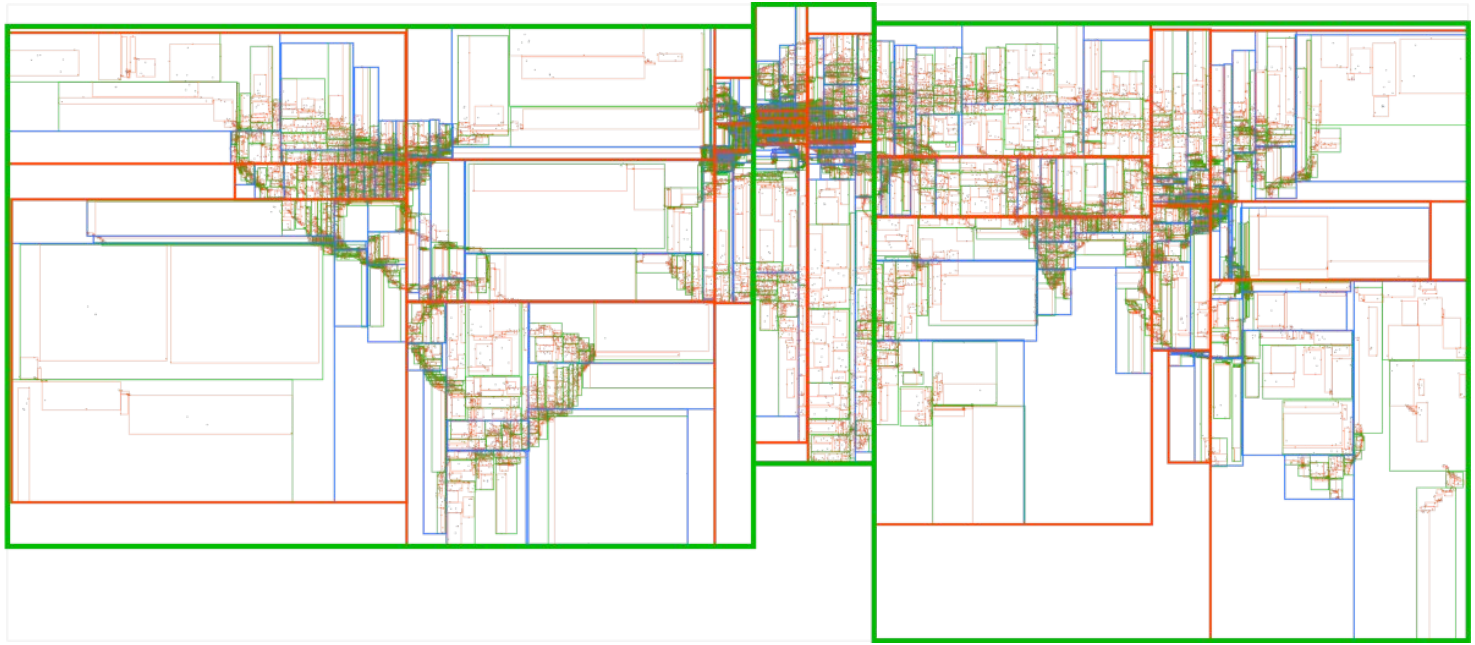


Problema 3

[3, 2, 1, 4, 5, 6, 7, 16, 15, 14]



Tip de tarea: Un ejemplo Gráfico, KD-Tree



http://bm-w.github.io/kd-tree-visualization/