Repaso I2

Árboles - ABB - AVL - 2-3 - RN

Cristóbal Berríos - crisberrios@uc.cl Nicolás Santa María - n.santamaria@uc.cl

¿Por qué Árboles?

optimized	base
0.176	13.524
0.188	16.848
0.179	16.273
0.320	28.715
0.349	32.709
0.283	45.130
0.325	46.006

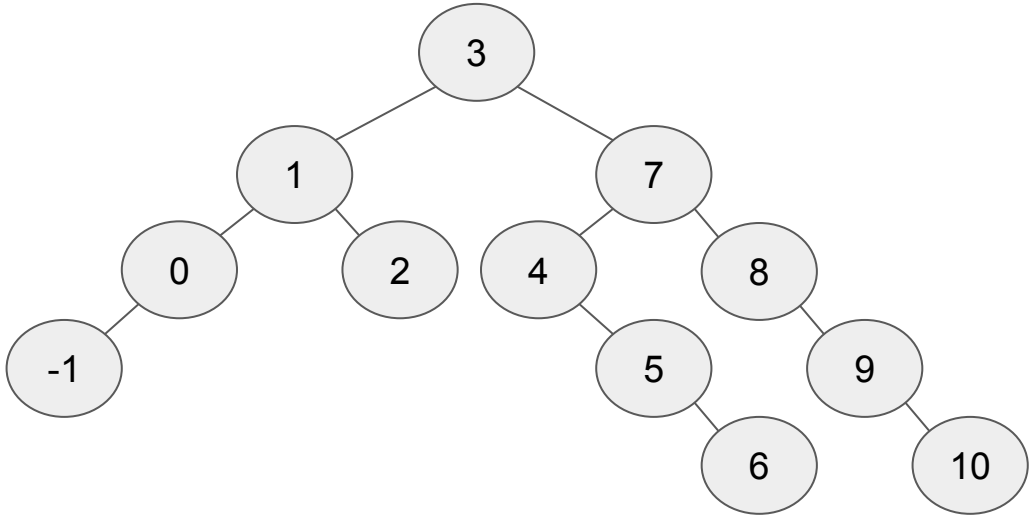
optimized	base
0.430	78.155
0.499	82.594
0.737	122.235
0.466	139.664
0.813	224.974
0.738	261.786
0.948	299.892

ABB

- Árbol Binario de Búsqueda
- Estructura de Datos para buscar valores
- Costo de la búsqueda depende de la altura del árbol
- En un nodo, el subárbol izquierdo son todos valores menores, y el subárbol derecho son todos valores mayores

Ejemplo ABB

3	1	7	8	9	2	10	0	4	5	6	-1



AVL

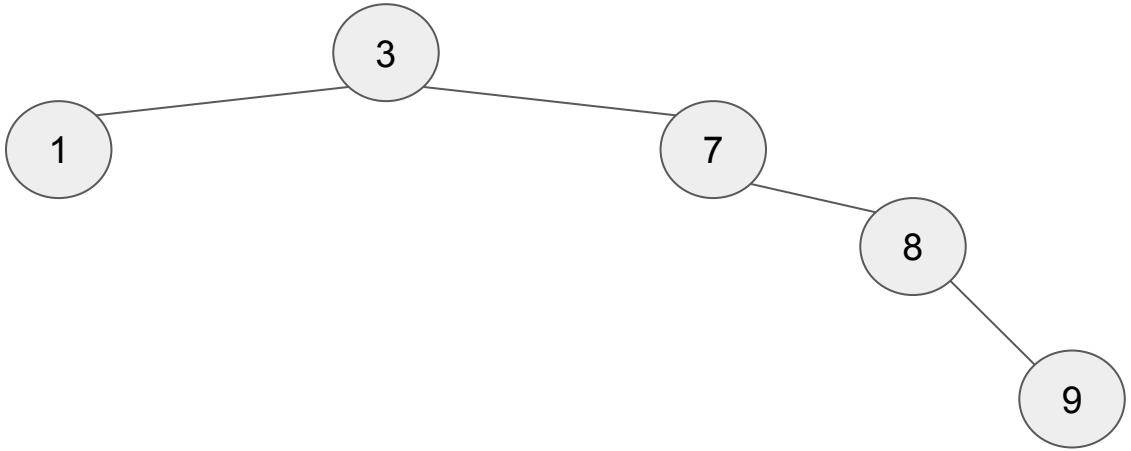
- Criterio de balance para ABB

- Pide dos condiciones:
  1) Las alturas de sus hijos difieren a lo más en 1
  2) Sus hijos están también AVL-balanceados

- Acota la altura del árbol a O(log n), y por lo tanto acota la búsqueda de elementos a O(log n)

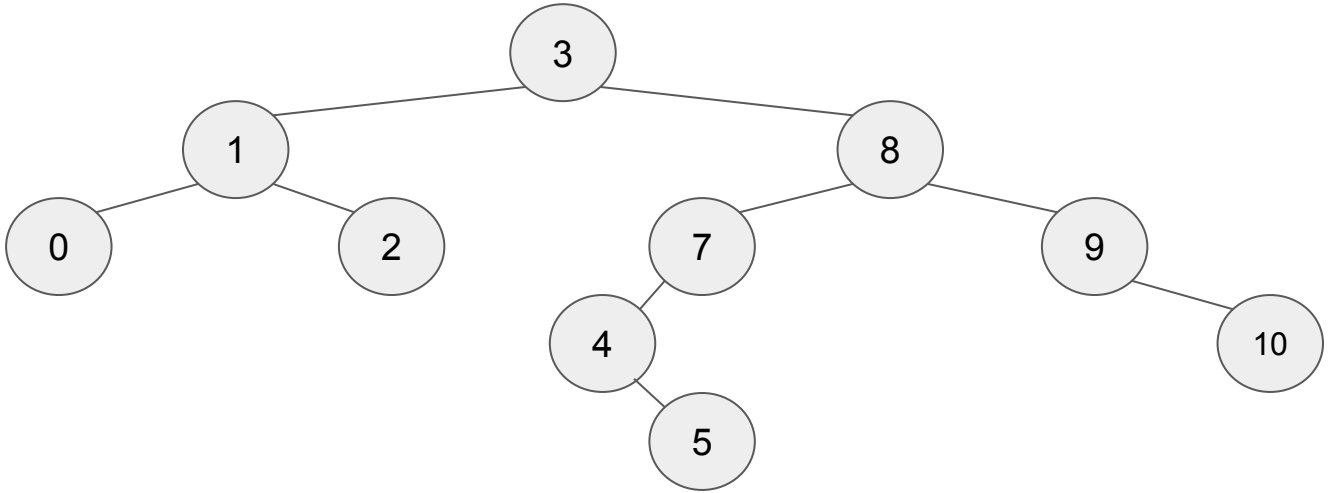
Ejemplo AVL

3	1	7	8	9	2	10	0	4	5	6	-1



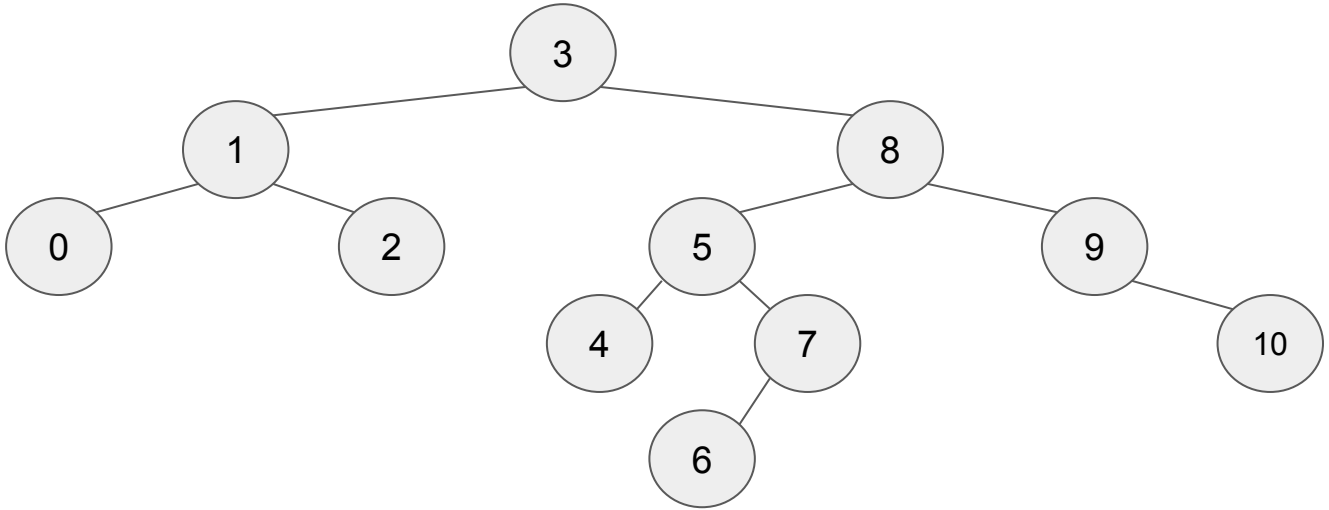
Ejemplo AVL

3	1	7	8	9	2	10	0	4	5	6	-1



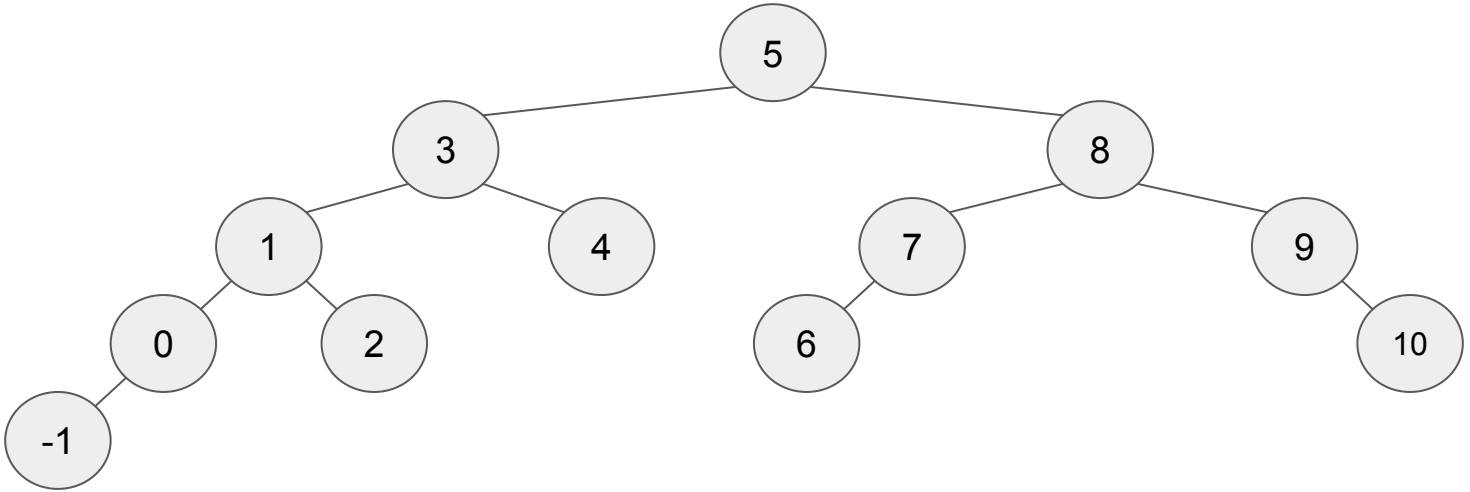
Ejemplo AVL

3	1	7	8	9	2	10	0	4	5	6	-1



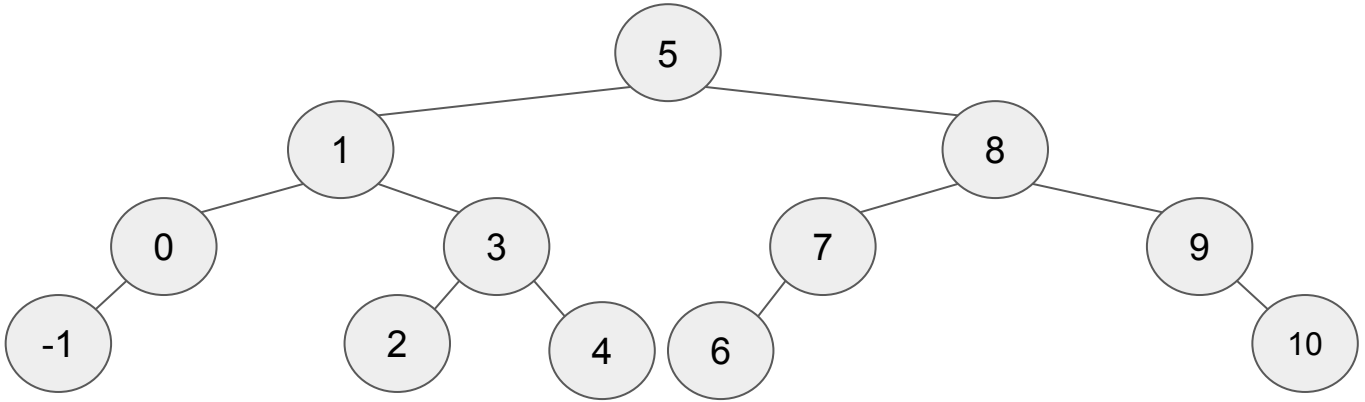
Ejemplo AVL

3	1	7	8	9	2	10	0	4	5	6	-1

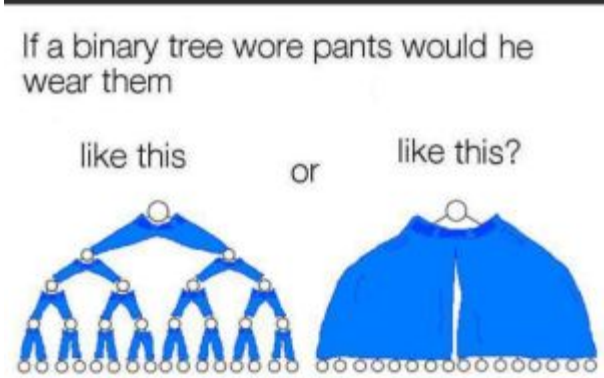


Ejemplo AVL

3	1	7	8	9	2	10	0	4	5	6	-1



2-3



- Tipo de árbol con condiciones para mantener balance, mantiene todas las hojas a la misma profundidad

- Tiene dos tipos de nodos
  1) Nodos 2: tienen una clave, y si no es hoja, 2 hijos
  2) Nodos 3: tienen dos claves, y si no es hoja, 3 hijos

- Se puede extender a 2-4, 2-5, etc

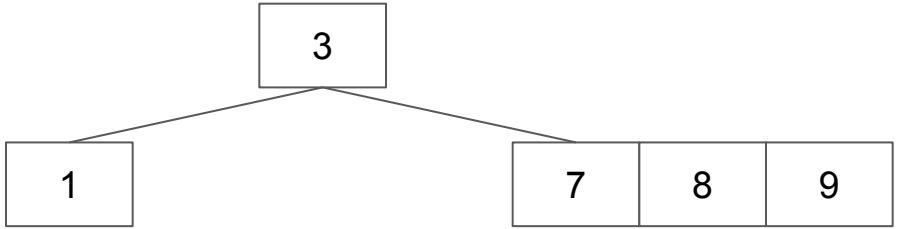
Ejemplo 2-3

3	1	7	8	9	2	10	0	4	5	6	-1

1	3	7

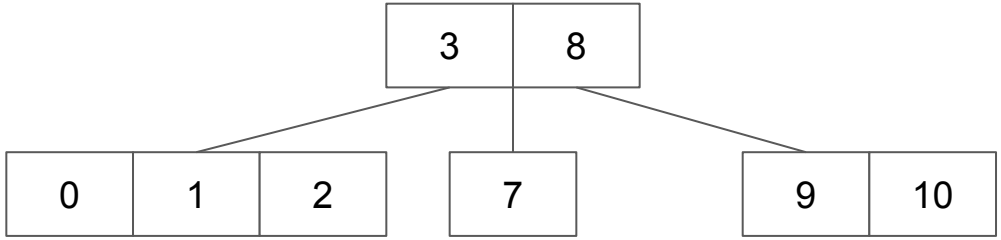
Ejemplo 2-3

3	1	7	8	9	2	10	0	4	5	6	-1



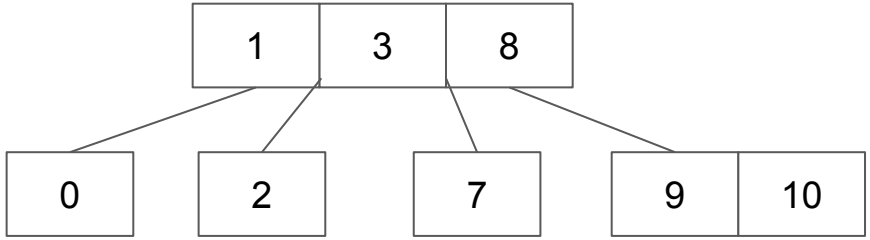
Ejemplo 2-3

3	1	7	8	9	2	10	0	4	5	6	-1



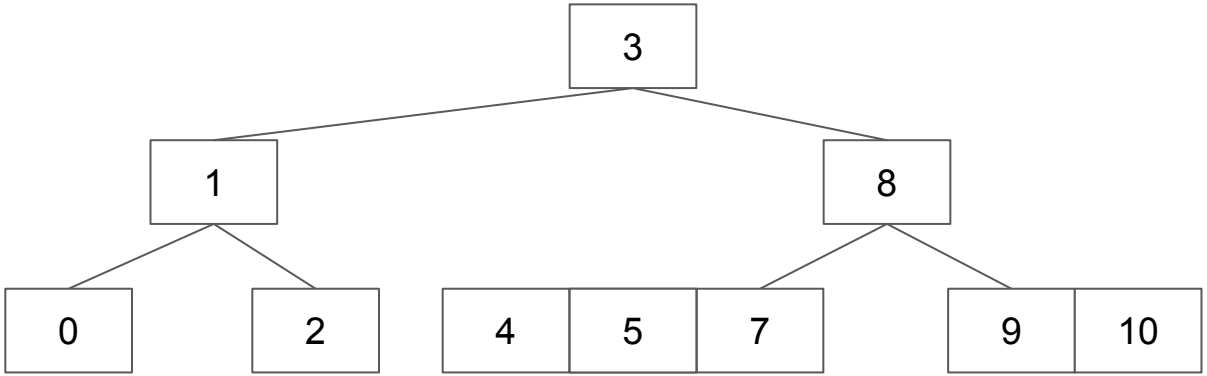
Ejemplo 2-3

3	1	7	8	9	2	10	0	4	5	6	-1



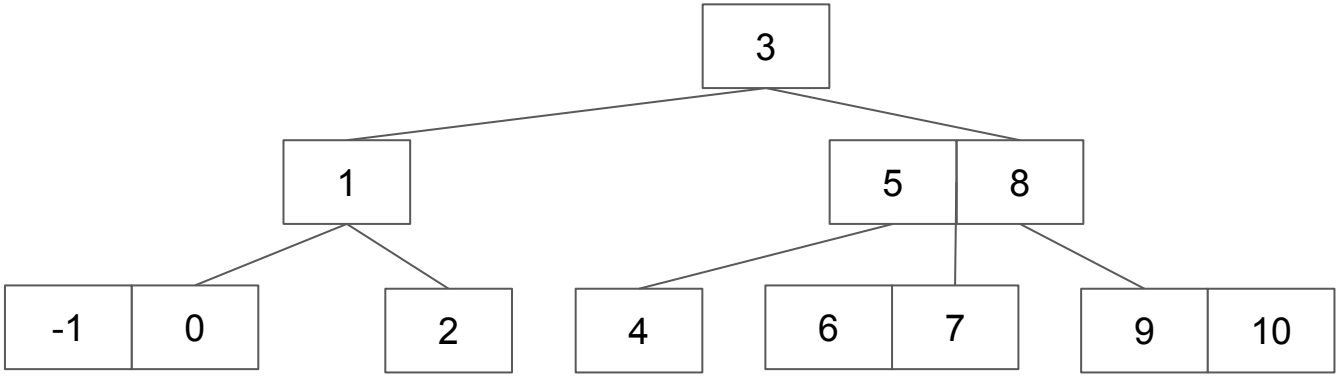
Ejemplo 2-3

3	1	7	8	9	2	10	0	4	5	6	-1



Ejemplo 2-3

3	1	7	8	9	2	10	0	4	5	6	-1



Rojo Negro

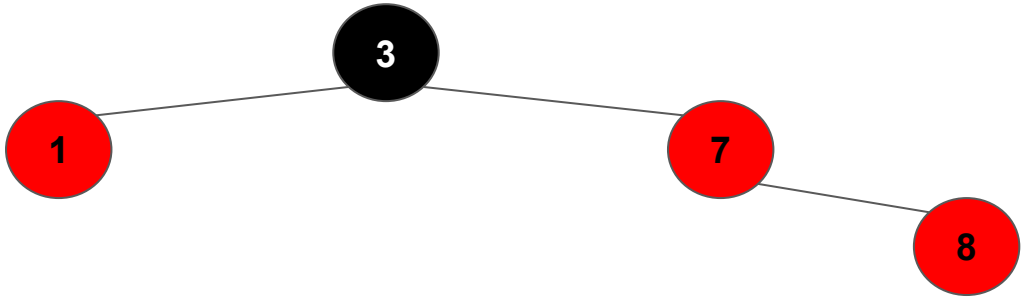
- Tipo de árbol que nace de intentar hacer el 2-3 como árbol binario

- Es equivalente a un 2-4

- Cumplen 4 condiciones:
  1) Cada nodo es **rojo** o **negro**
  2) La raíz del árbol es **negra**
  3) Si un nodo es **rojo**, sus hijos son **negros**
  4) Misma cantidad de nodos **negros** hasta las hojas

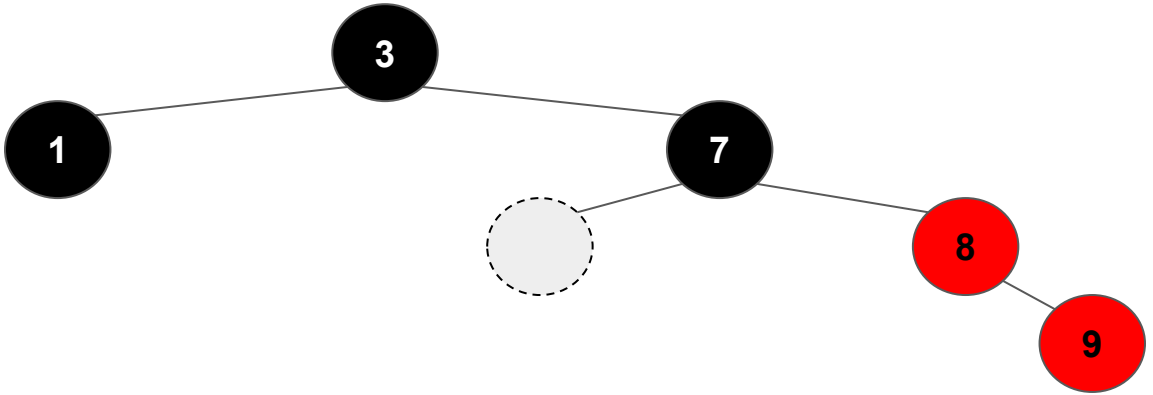
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



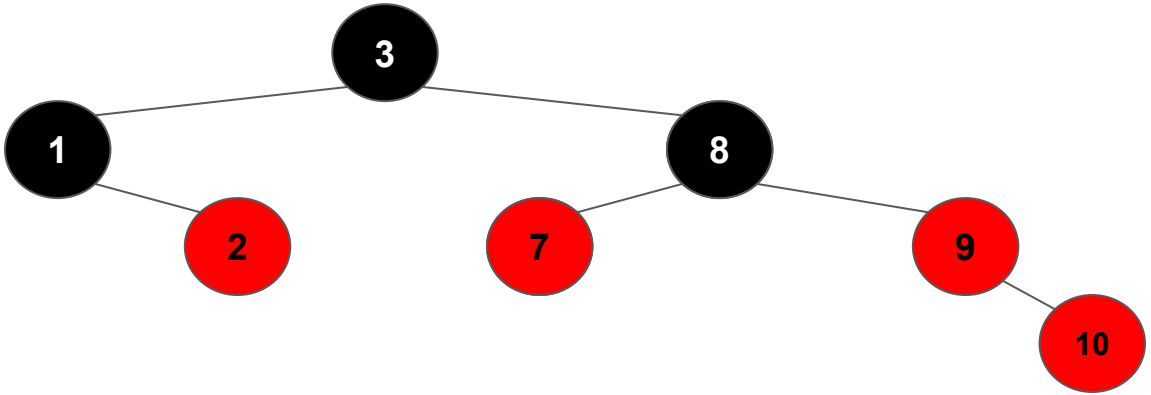
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



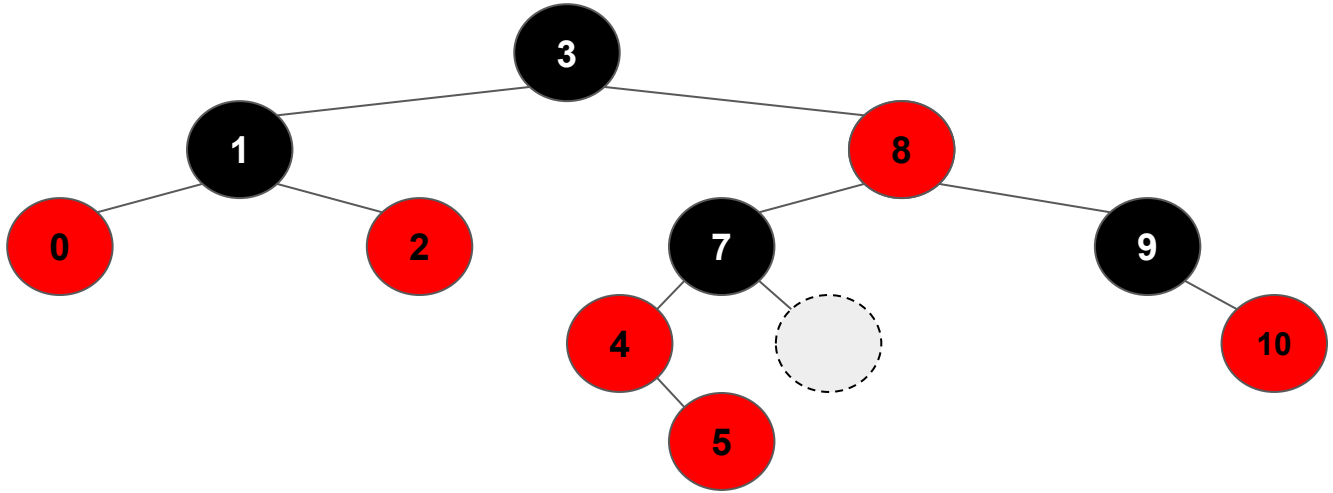
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



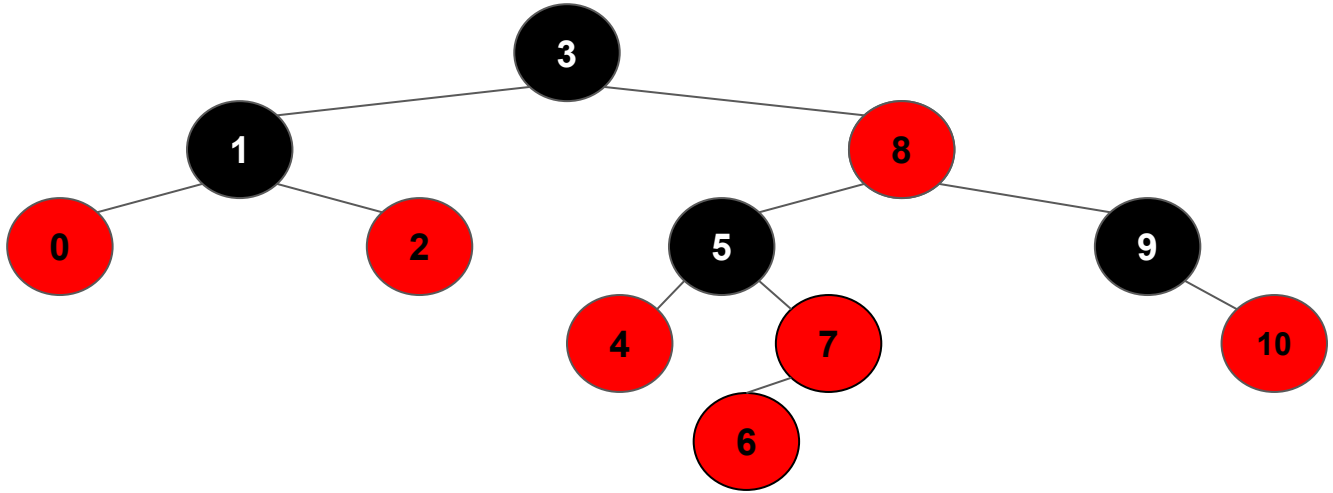
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



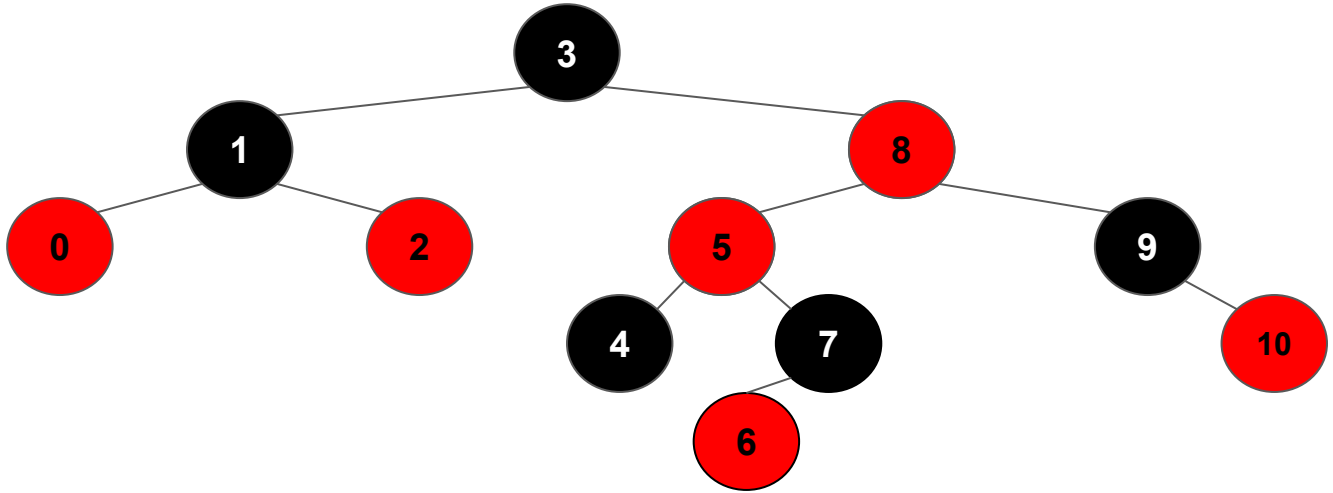
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



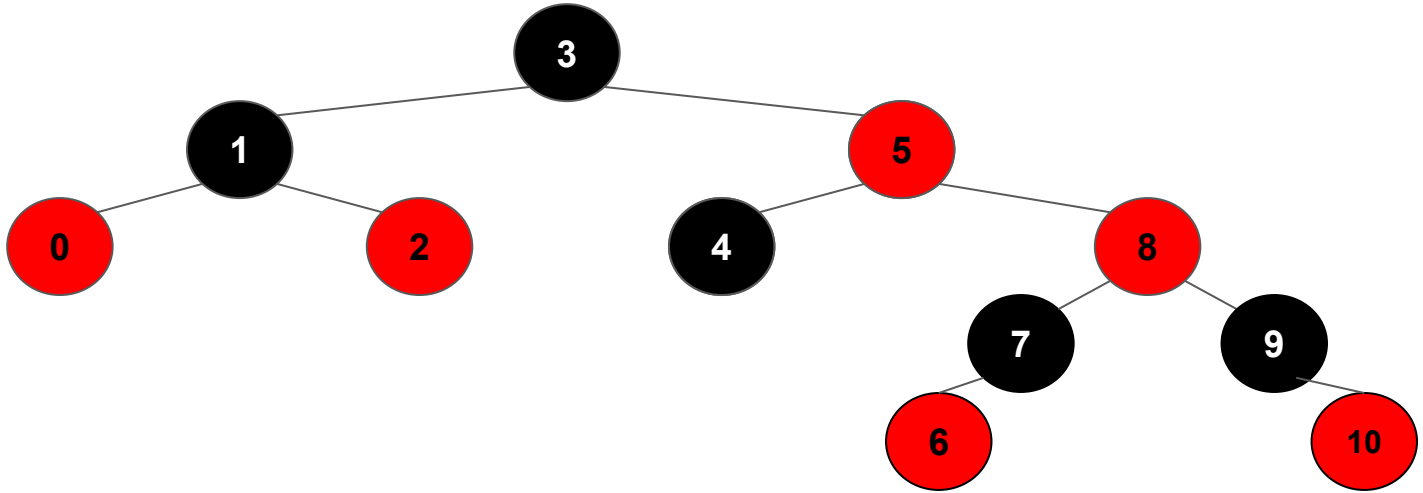
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



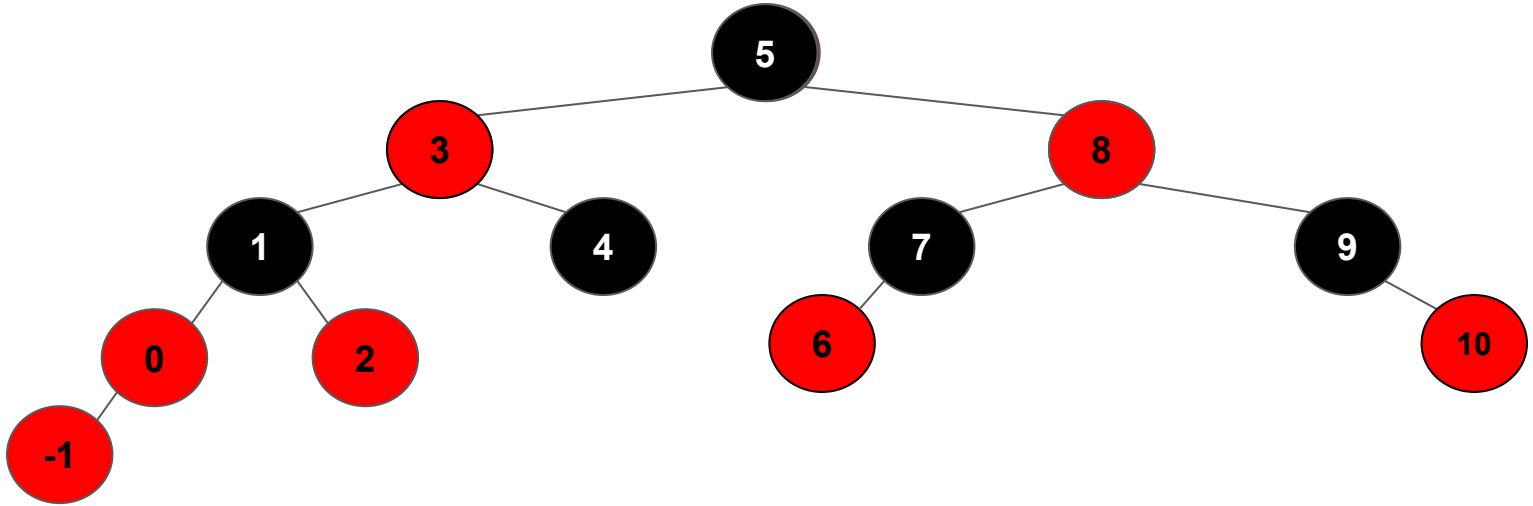
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



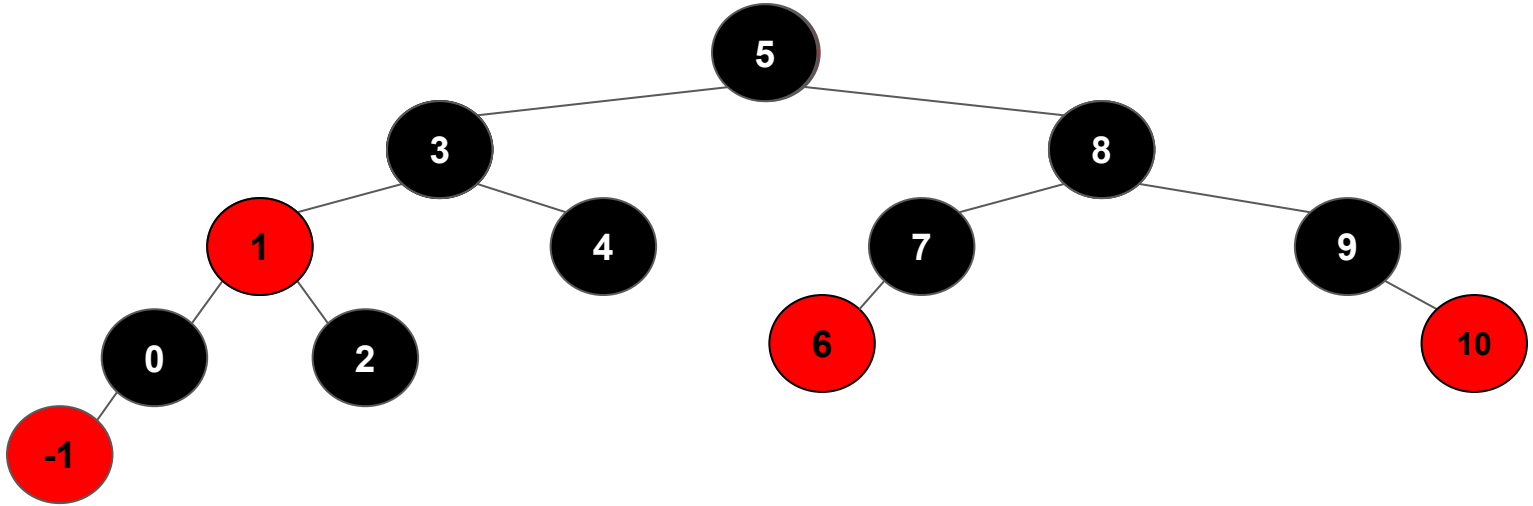
Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1



Ejemplo Rojo Negro

3	1	7	8	9	2	10	0	4	5	6	-1

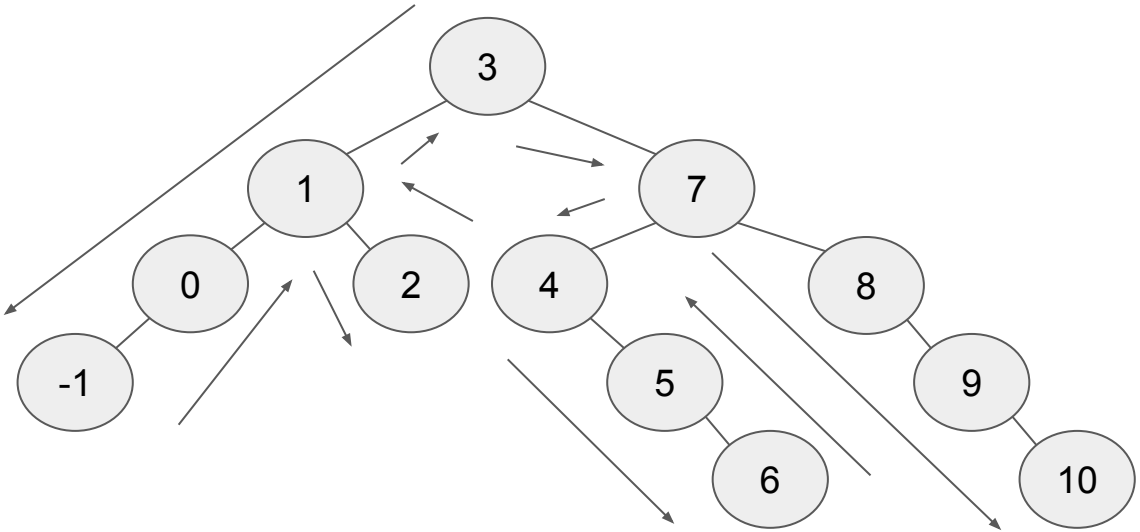


Pregunta 1

Considera que tienes un ABB T vacío sin auto-balance, y una lista desordenada L de números.

a) ¿Cómo se puede utilizar T para ordenar L?
b) ¿Cuál es la complejidad de esto en el mejor caso? ¿Y en el peor?

Pregunta 1



Recorre todos los nodos, por lo que tiene complejidad **O(n)**

Pregunta 1

La complejidad del recorrido in-order es **O(n)**

Pregunta 1

La complejidad del recorrido in-order es **O(n)**

La complejidad de armar el árbol, depende de los datos

Pregunta 1

La complejidad del recorrido in-order es **O(n)**

La complejidad de armar el árbol, depende de los datos

En el mejor caso, insertar un nodo es **O(log n)**, luego la complejidad de armar el árbol quedaría **O(n log n)**

Pregunta 1

La complejidad del recorrido in-order es **O(n)**

La complejidad de armar el árbol, depende de los datos

En el mejor caso, insertar un nodo es **O(log n)**, luego la complejidad de armar el árbol quedaría **O(n log n)**

En el peor caso, insertar un nodo es **O(n)**, luego la complejidad de armar el árbol queda **O(n^2)**

Pregunta 2

Se define B como el conjunto de nodos en la ruta de inserción de una llave x. Se define A como los nodos que están a la izquierda de la ruta de inserción, y C como los que están a la derecha.

a) ¿Siempre se cumple que $a \leq b \leq c$?
b) ¿Es posible que un nodo en C sea menor que un nodo en A?

Pregunta 2

Hay dos casos para a:

1) a es hijo izquierdo de un nodo de B. Por definición de ABB, a tiene que ser menor que todo el subárbol derecho de B. El nodo que se busca, x, si o sí está en el subárbol derecho de B. Luego, a < x

Pregunta 2

Hay dos casos para a:

2) a es un descendiente de otro nodo de A. Por definición de ABB, si un nodo es menor a un número, todos sus descendientes son menores a ese número. Como si o si el nodo a es descendiente de un nodo que cumple el primer caso, este nodo también cumple a < x.

Pregunta 2

Análogamente, podemos hacer el mismo desarrollo para C, llegando a que todo nodo en C cumple que $x < c$.

Luego, $a < x < c$, y demostramos que $a < c$.

Pregunta 3 - AVL

En clases vimos que para un árbol AVL de altura h, el número mínimo de nodos (claves), $m(h)$, cumple la recurrencia $m(h) = m(h-1) + m(h-2) + 1$, y resolvimos esta recurrencia apoyándonos en las propiedades de la secuencia de Fibonacci.

Pero también es posible resolver la recurrencia de otra forma: notamos que $m(h-1) > m(h-2)$; por lo tanto, $m(h) > 2m(h-2)$. Así, los primeros pasos del nuevo desarrollo son

$$m(h) > 2m(h-2) > 4m(h-4) > 8m(h-6) > \ldots$$

Termina de resolver la recurrencia y encuentra la relación entre la altura h, y el número mínimo de nodos $m(h)$, en notación $O(\ldots)$, en un árbol AVL.

Pregunta 3 - AVL

$$m(h) > 2m(h-2) > 4m(h-4) > \cdots > 2^k m(h-2k)$$

$$m(h) > 2^k m(h-2k)$$

$$h - 2k = 1 \rightarrow k = \frac{h-1}{2}$$

$$m(h) > 2^{\frac{h-1}{2}} \cdot m(1)$$

$$log(m(h)) > \frac{h-1}{2} \rightarrow h < 2log(m(h)) + 1$$

$$h \in \mathcal{O}(log(m(h)))$$

$$h \in \mathcal{O}(log(n))$$

Pregunta 4 - Rojo Negro

Considera un arbol Rojo-Negro formado por insertar **n** nodos. Argumente que si se tiene **n > 1**, el árbol tiene al menos un nodo rojo.