Estructuras de Datos no Lineales

1.5.2. Árboles binarios de búsqueda equilibrados

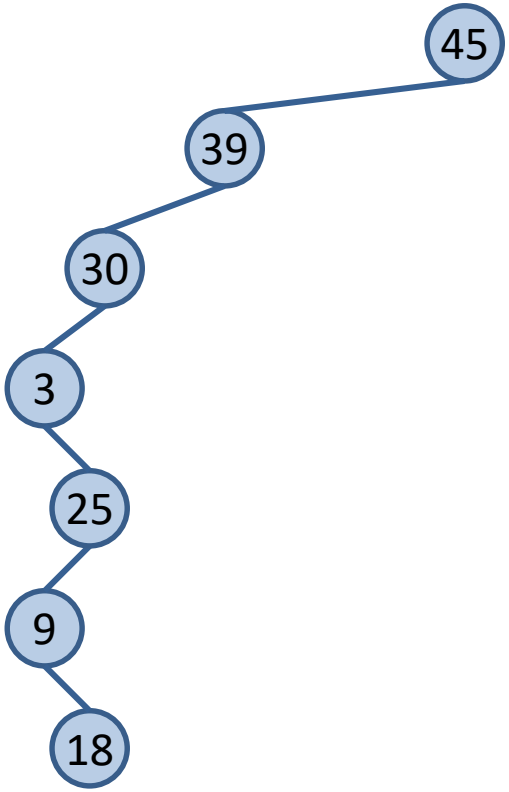
José Fidel Argudo Argudo
José Antonio Alonso de la Huerta
Mª Teresa García Horcajadas

Versión 3.0

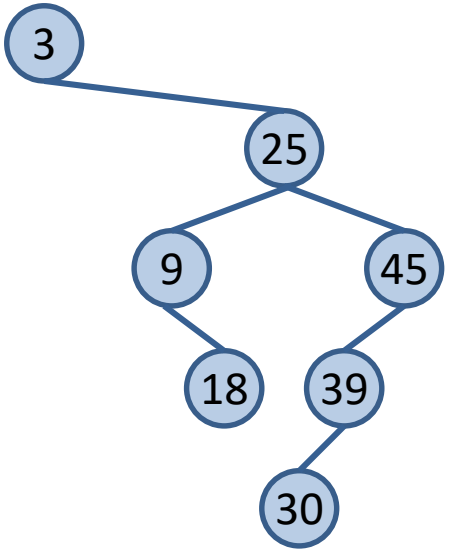
Desequilibrio de un ABB

El orden de inserción de los elementos en un ABB determina el grado de equilibrio del árbol.

45, 39, 30, 3, 25, 9, 18



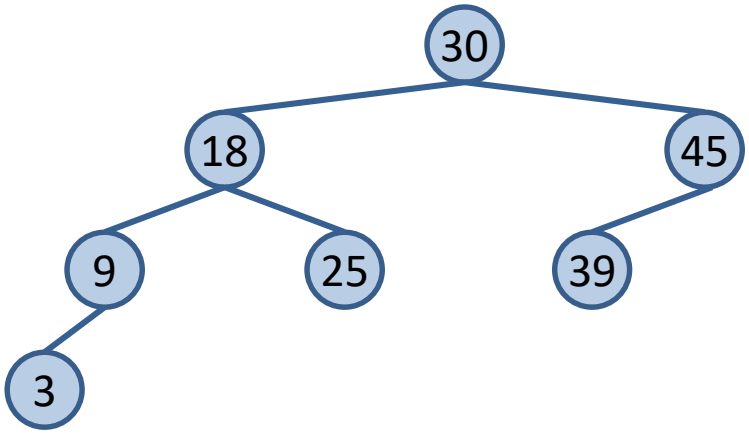
3, 25, 45, 39, 9, 30, 18



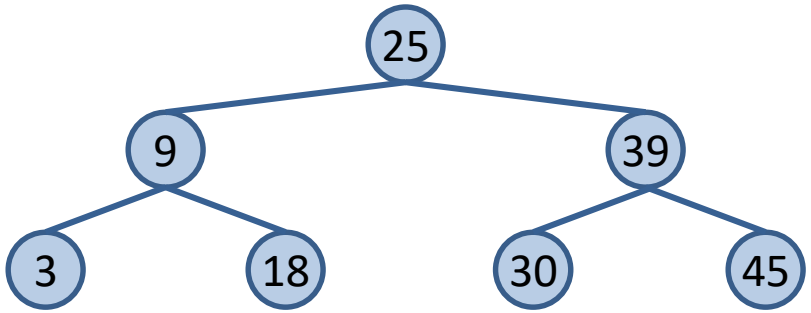
Desequilibrio de un ABB

El orden de inserción de los elementos en un ABB determina el grado de equilibrio del árbol.

30, 45, 39, 18, 9, 25, 3



25, 9, 18, 39, 30, 3, 45



Desequilibrio de un ABB

- Las sucesivas inserciones y eliminaciones en un ABB pueden alterar el grado de equilibrio del árbol.

- El tiempo de las operaciones sobre un ABB (búsqueda, inserción y eliminación)
  - depende de la altura, por tanto, del grado de equilibrio del árbol, y
  - puede llegar a ser $O(n)$ en el caso más desfavorable (árbol degenerado en una lista).

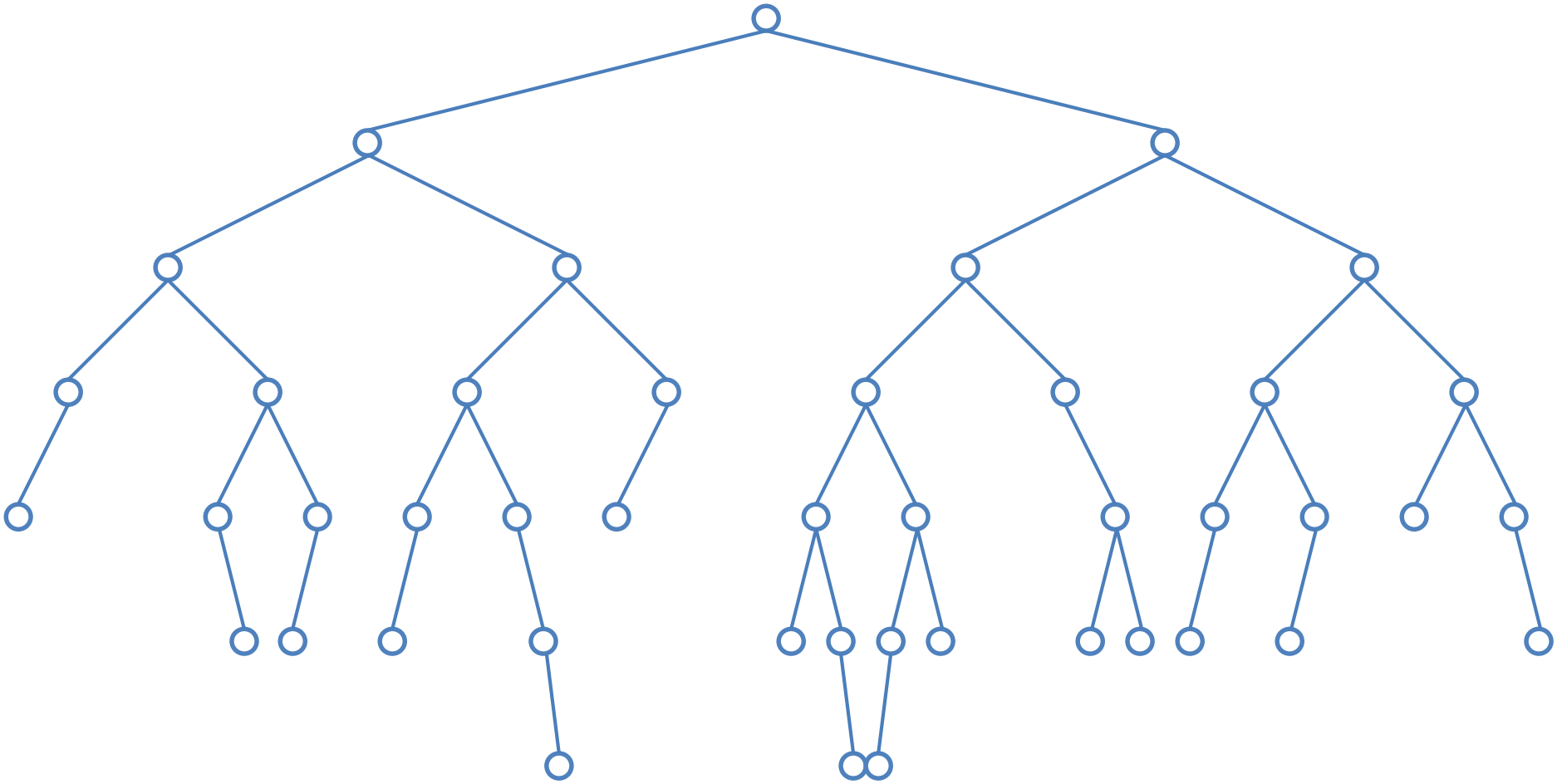
- Para garantizar un tiempo proporcional a la mínima altura posible, o sea $O(\log_2 n)$, es necesario mantener el árbol tan equilibrado como sea posible, después de cada inserción o eliminación.

Árbol AVL (ABB equilibrado)

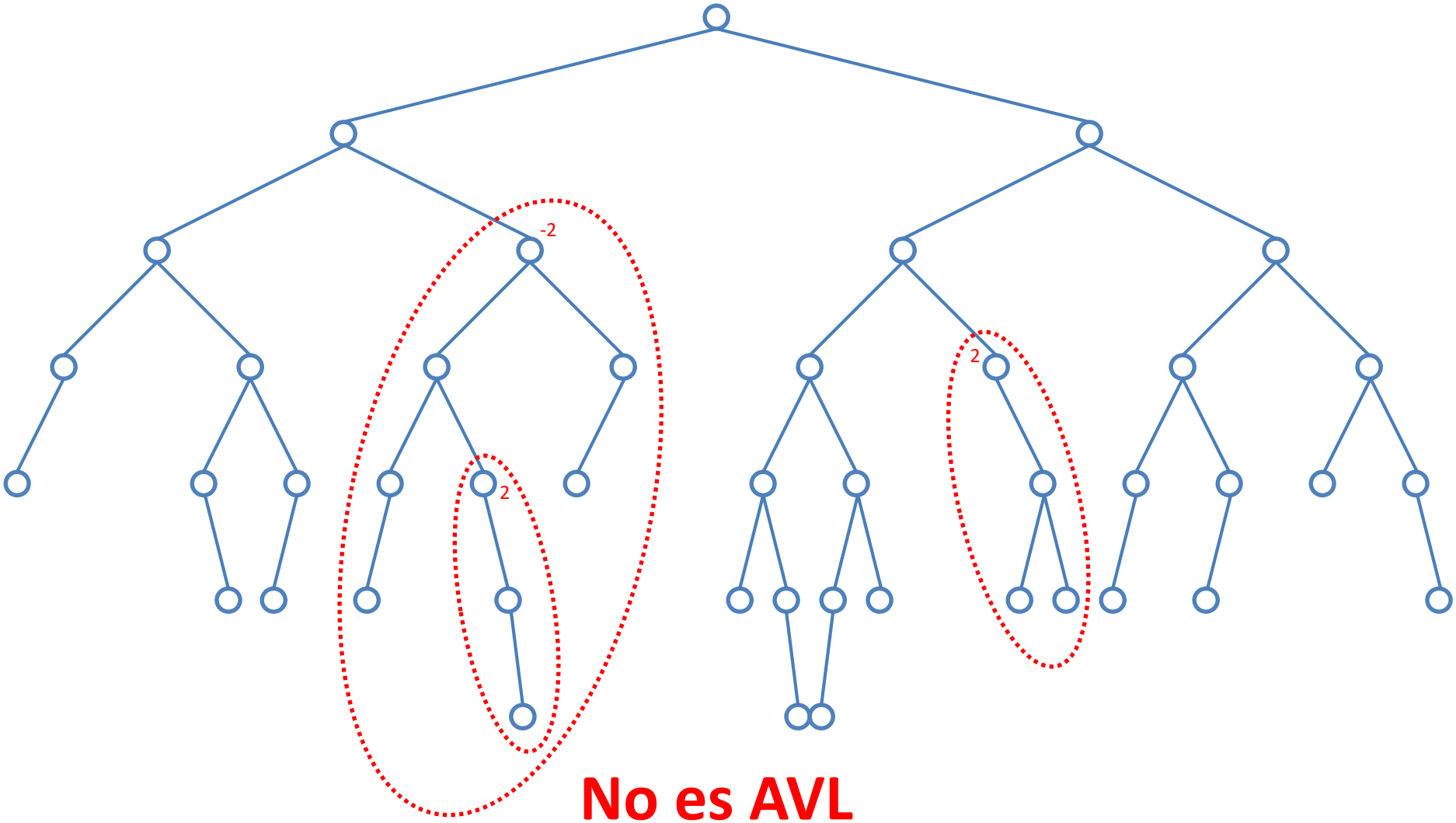
Un **árbol AVL** (**A**delson-**V**elskii & **L**andis, 1962) es un tipo de árbol binario de búsqueda equilibrado.

- Factor de equilibrio de un nodo: altura del subárbol derecho menos altura del subárbol izquierdo del nodo.
- Árbol binario equilibrado: aquél en el que el factor de equilibrio de todos los nodos es -1, 0 o 1.
- La propiedad de equilibrio garantiza que la altura de un AVL es de orden logarítmico.
- Los algoritmos de inserción y eliminación en un AVL pueden verificar la condición de equilibrio del árbol y, si es necesario, reequilibrarlo mediante rotaciones de sus nodos, en un tiempo proporcional a su altura.
- En consecuencia, los tiempos de búsqueda, inserción y eliminación en un árbol AVL están en $O(\log n)$, en el peor caso.

Suponiendo que los valores almacenados en los nodos cumplen la propiedad de orden de búsqueda, ¿este árbol es un AVL?

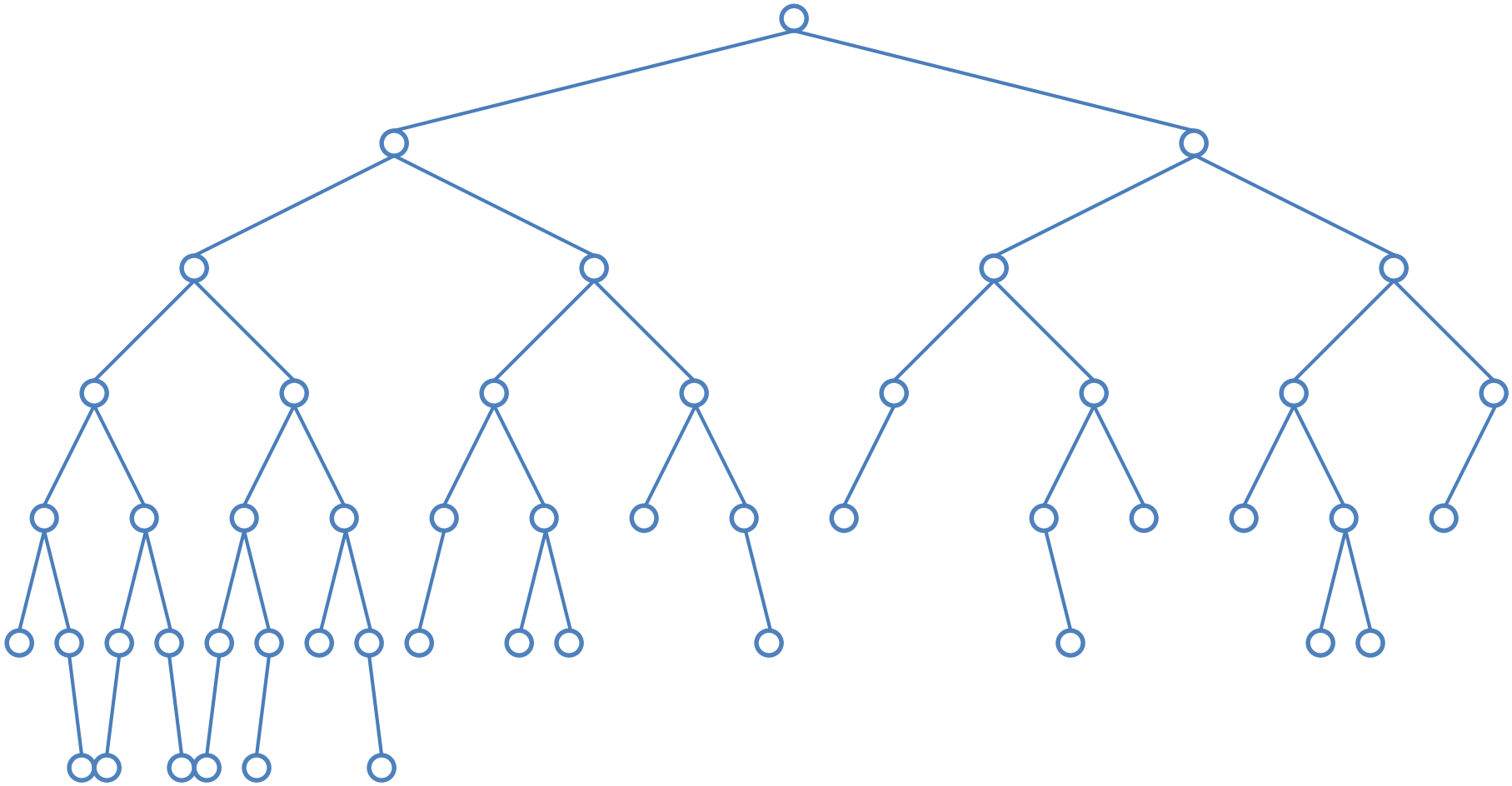


Suponiendo que los valores almacenados en los nodos cumplen la propiedad de orden de búsqueda, ¿este árbol es un AVL?

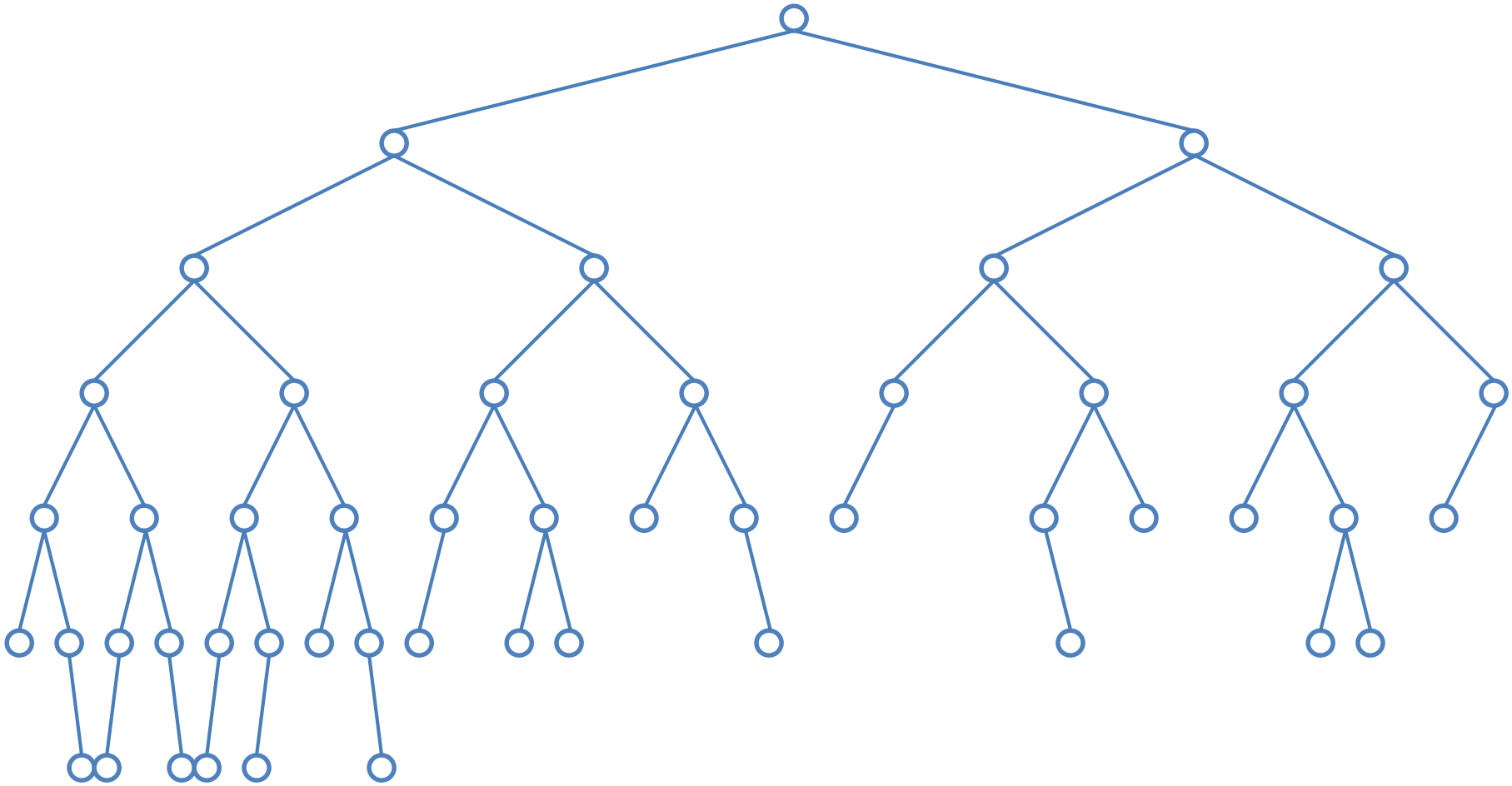


No es AVL

Suponiendo que los valores almacenados en los nodos cumplen la propiedad de orden de búsqueda, ¿este árbol es un AVL?

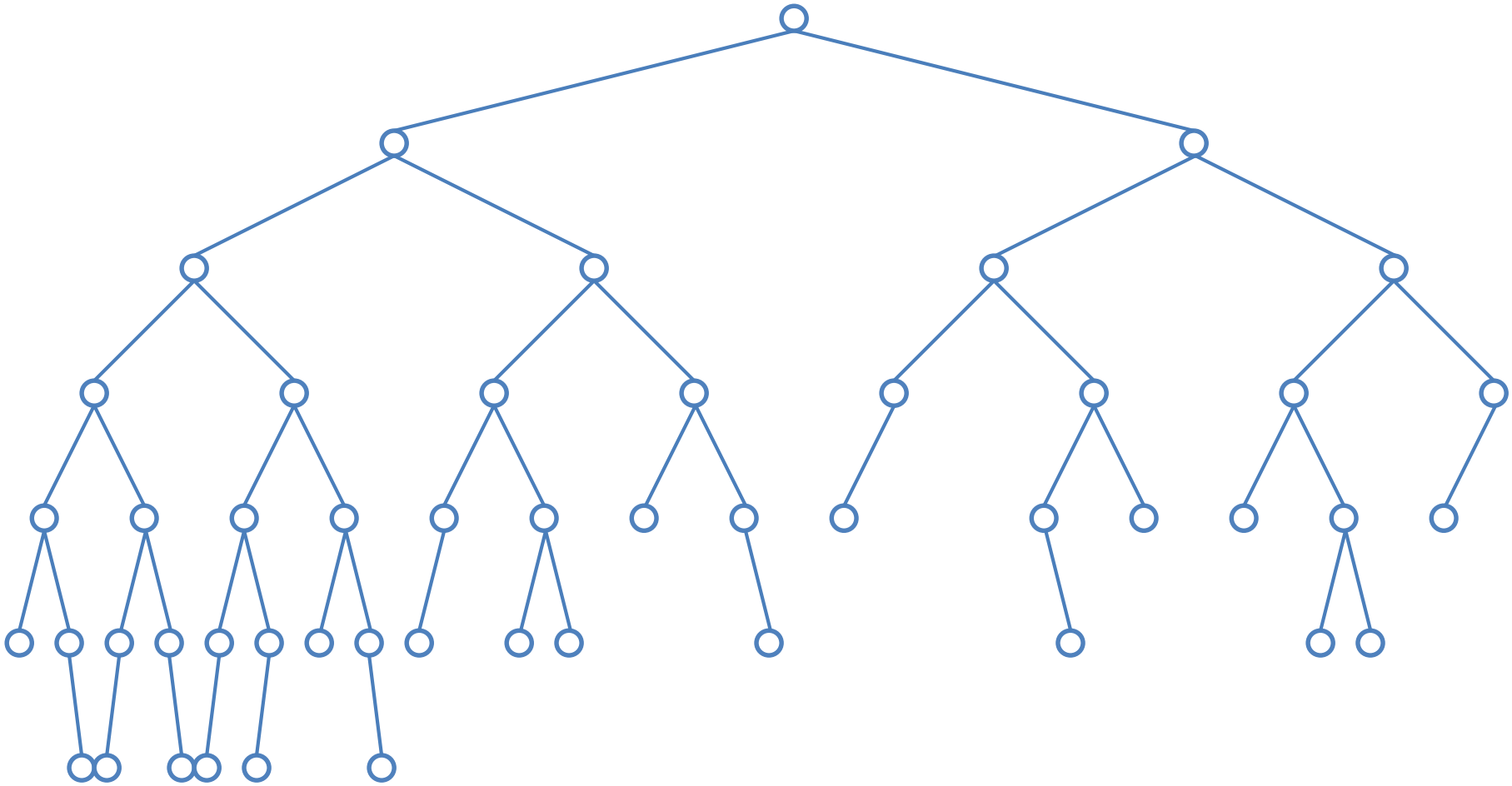


Suponiendo que los valores almacenados en los nodos cumplen la propiedad de orden de búsqueda, ¿este árbol es un AVL?



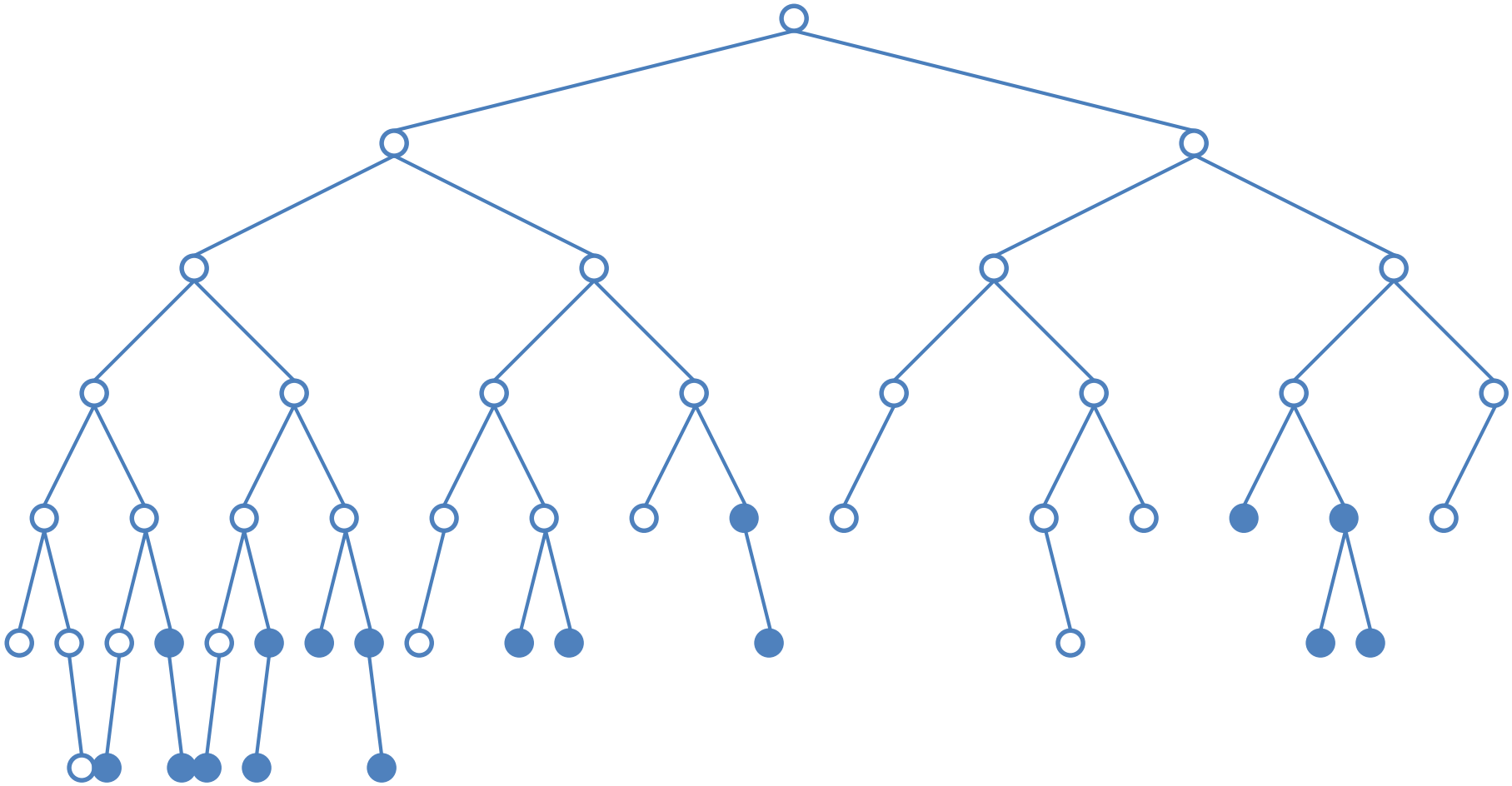
Es AVL

¿Cuáles son los nodos que se pueden suprimir sin que el árbol pierda el equilibrio y manteniendo su altura?



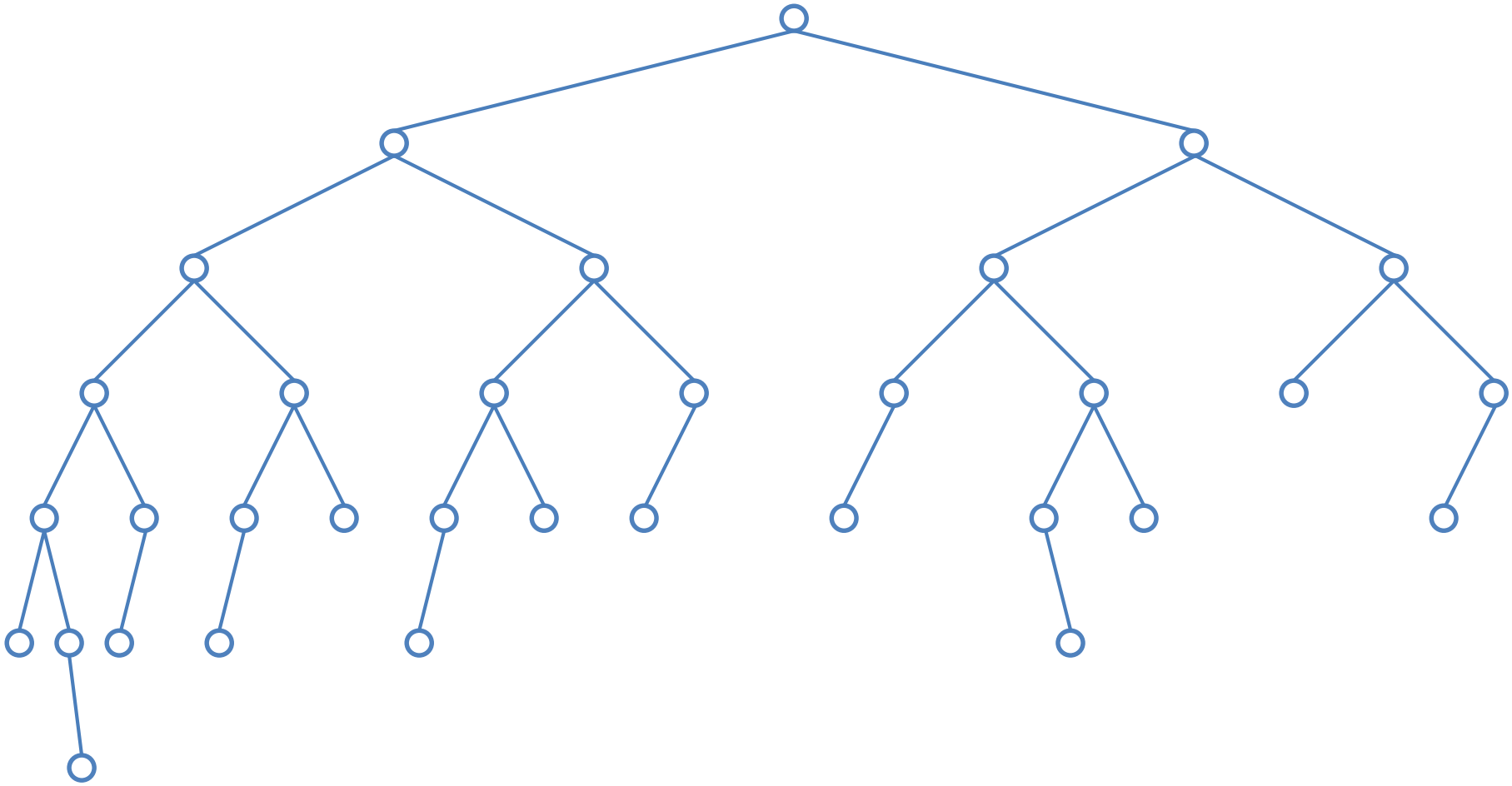
Es AVL

¿Cuáles son los nodos que se pueden suprimir sin que el árbol pierda el equilibrio y manteniendo su altura?



Es AVL

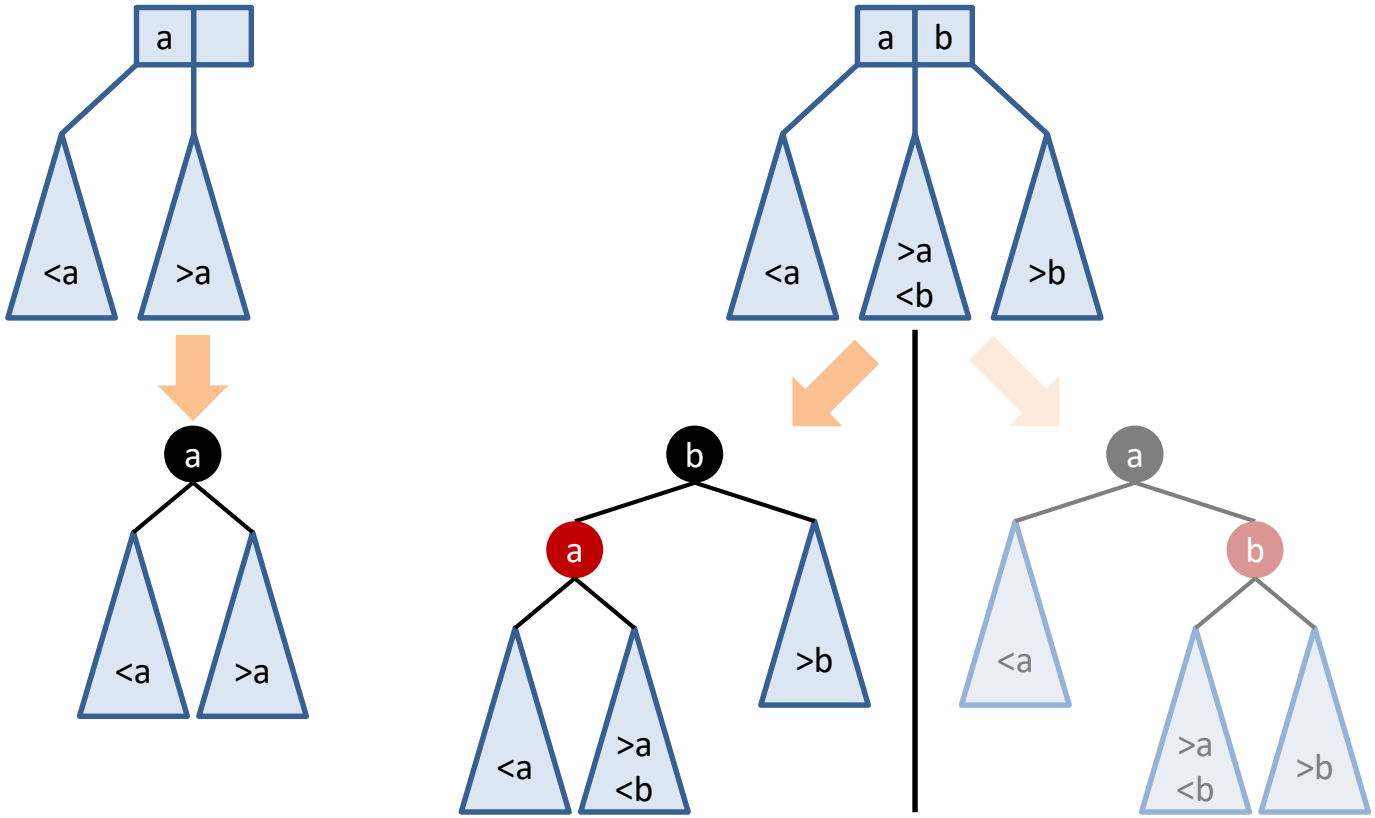
¿Cuáles son los nodos que se pueden suprimir sin que el árbol pierda el equilibrio y manteniendo su altura?



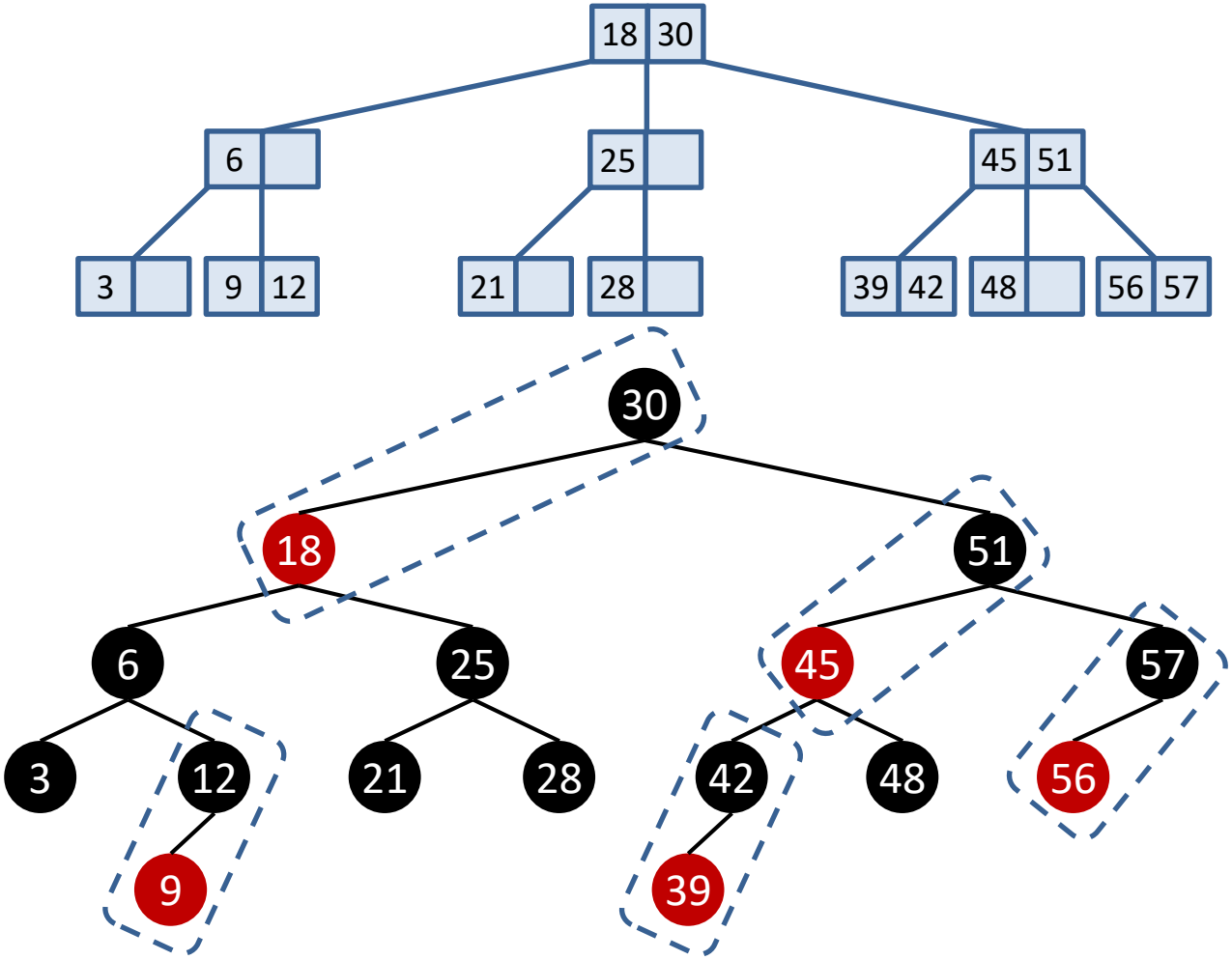
Es AVL

Árbol rojinegro (ABB equilibrado)

Un **árbol rojinegro** (ARN) es un ABB que representa un *árbol B de orden 3* (árbol 2-3), el cual está equilibrado (todas las hojas están en el mismo nivel). Esta representación permite una implementación más sencilla de las inserciones y eliminaciones y, también evita el sobrecoste de manejar una estructura de nodo más compleja, con más claves y punteros.



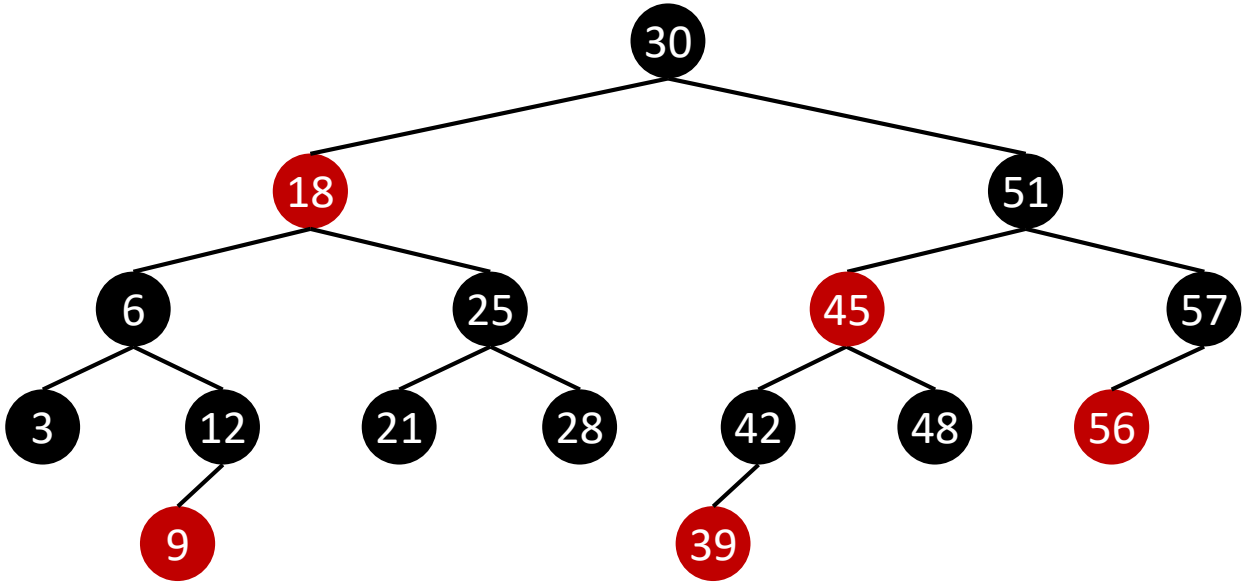
Árbol rojinegro (ABB equilibrado)



Árbol rojinegro (ABB equilibrado)

Árbol rojinegro (ARN): ABB con nodos rojos y negros, que cumple tres condiciones:

1. Cualquier nodo rojo es hijo izquierdo (implica que la raíz es negra).
2. Todo nodo rojo tiene sus hijos negros.
3. Toda rama desde la raíz tiene el mismo número de nodos negros.



Árbol rojinegro (ABB equilibrado)

Inserción y eliminación: se realizan como en un ABB y a continuación se retrocede por el camino de búsqueda para restaurar, si es necesario, las propiedades de ARN (debe representar correctamente el árbol 2-3 subyacente) mediante una combinación de un número limitado de *cambios de color* y *rotaciones*. Por lo tanto, como en un ABB, los tiempos de estas operaciones son del orden de la altura del árbol.

Operaciones internas:

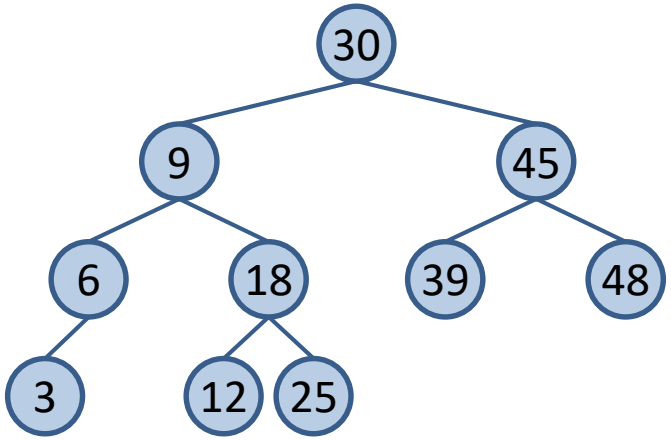
- **Rotaciones**: recolocan los nodos del árbol conservando la propiedad de orden/búsqueda.
- **Repintado**: cambia el color de algunos nodos.

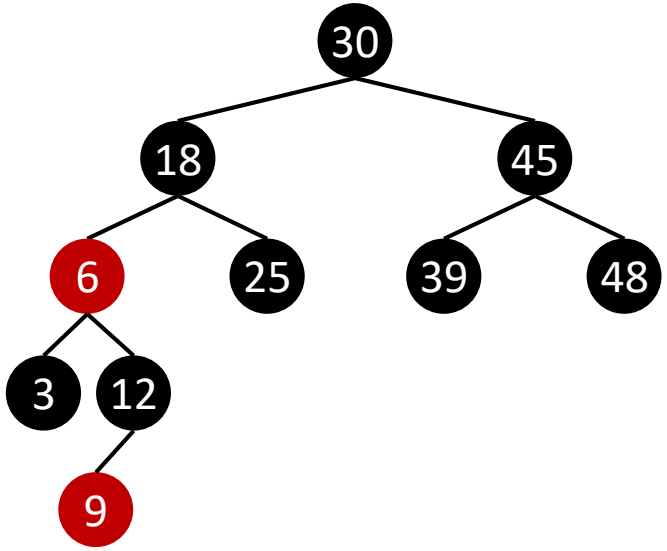
Eficiencia: Un árbol rojinegro con *n* nodos tiene una altura $h \leq 2\log_2 n$ y, dado que las operaciones de búsqueda, inserción y eliminación (incluyendo la restauración de ARN) tienen un peor tiempo de ejecución proporcional a la altura del árbol, entonces tales operaciones son de $O(\log n)$.

Se puede optar por representar árboles 2-3-4 (árboles B de orden 4) y usar versiones más complicadas de los algoritmos básicos que requieren menos rotaciones (como máximo una rotación por inserción y tres por eliminación).

AVL vs. ARN

Los AVL están *«más equilibrados»*, pero de forma más rígida, por lo que la búsqueda es un poco más rápida (la altura del árbol suele ser menor), pero las inserciones y eliminaciones son ligeramente más lentas. Es debido a que al insertar hay que actualizar los FE de los ascendientes del nodo insertado y, si es necesario, realizar una rotación; en la eliminación también hay que actualizar los FE, pero además, pueden necesitarse varias rotaciones para reequilibrar un AVL. En la práctica, la diferencia entre AVL y ARN respecto al rendimiento global es poco apreciable.





AVL y ARN con los mismos valores insertados en el mismo orden.