a- Usar un ABB es más conveniente que un AVL es más conveniente cuando:

* La inserción/eliminación es más frecuente que la búsqueda.
* No se requiere mantener el árbol perfectamente balanceado.
* El árbol no crece mucho.
* Se desea una implementación más simple (el ABB no necesita rotaciones).

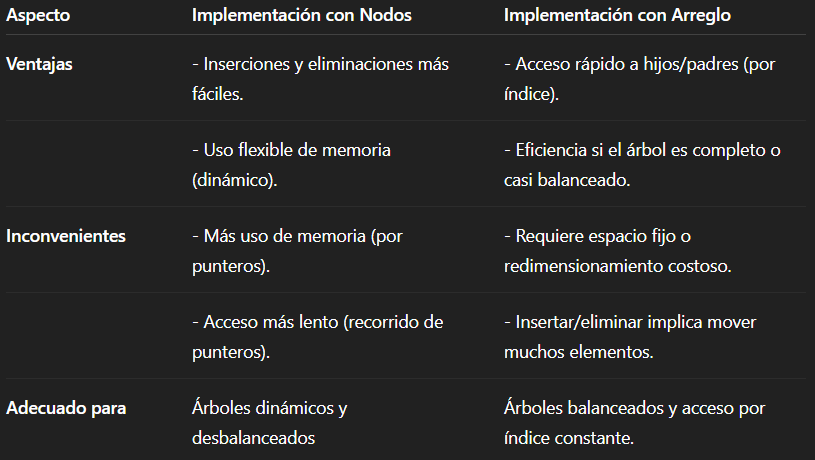
El ABB es más fácil de implementar y modificar. Sin embargo, si los datos se insertan en orden o muy desbalanceados, el árbol puede degenerarse a una lista (con eficiencia O(n)).  
No es conveniente si:

* Se necesita operaciones de búsqueda rápidas y constantes.
* Se va a trabajar con grandes volúmenes de datos.
* El orden de inserción de datos es aleatorio o no controlado.

Se usa un AVL cuando:

* Se necesita alta eficiencia en búsquedas (logarítmica garantizada).
* El conjunto de datos puede crecer mucho o llegar en orden aleatorio.

b-



Los nodos son mejores para estructuras dinámicas, y los arreglos para estructuras estáticas o balanceadas (como un heap o árbol binario completo).

c- La altura de un árbol es la longitud del camino más largo desde la raíz a una hoja.

Relación clave: Cuanto menor es la altura, más eficientes son las operaciones de búsqueda, inserción y eliminación.  
Estas operaciones dependen de recorrer desde la raíz hasta el nodo deseado, lo que en árboles binarios toma un tiempo proporcional a la altura.



AVL mantiene la altura mínima posible, garantizando eficiencia.  
ABB puede degenerarse si no se balancea, afectando el rendimiento.