Mini-rapport : BST vs AVL

Dans ChargeCraft, il y a eu 2 choix possibles pour indexer les stations par **station_id**

- BST (Binary Search Tree) : simple arbre binaire de recherche
- AVL (Adelson-Velsky & Landis) : BST auto-équilibré

Mais qu'est-ce qu'un BST et un AVL ?

Un BST (Binary Search Tree) est un arbre binaire où chaque nœud a un enfant gauche plus petit et un enfant droit plus grand, utilisé pour stocker des données triées.

Un AVL est un BST auto-équilibré, qui ajuste sa structure avec des rotations pour que la hauteur reste minimale, garantissant des accès rapides.

Alors pourquoi ne pas toujours prendre un AVL ?

Parce que les AVL demandent plus de rotations et de gestion à l'insertion/suppression, ce qui peut ralentir légèrement les opérations si l'équilibrage n'est pas nécessaire.

Et dans notre cas ? Avons nous besoin d'un AVL ou d'un BST ?

On utilise un **AVL** pour garantir que les recherches, insertions et mises à jour de stations restent **rapides (O(log n))** même si les IDs arrivent triés ou dans un ordre défavorable.

Un simple BST pourrait devenir déséquilibré et ralentir les accès à O(n).

Si les **station_id** arrivent dans un **ordre croissant**, le BST pourrait devenir déséquilibré avec une **hauteur** de **1000 pour 1000 stations,** ce qui donne une complexité O(n). Avec un AVL, l'arbre reste équilibré, sa hauteur serait ≈ 10, ce qui rend les recherches et insertions beaucoup plus rapides.

Dans notre projet les fonctions **rotL()**, **rotR()** et **rebalance()** assurent cet équilibrage.