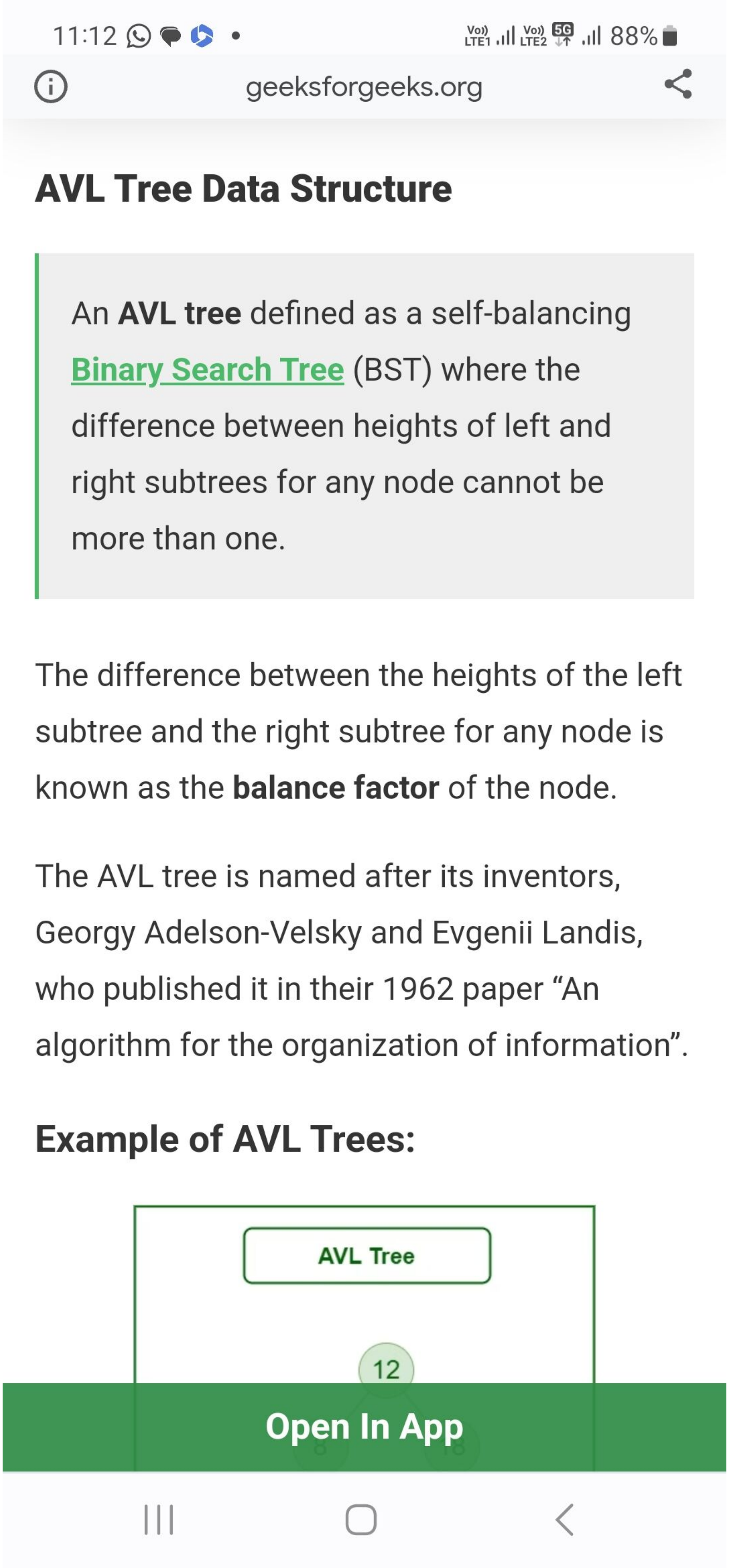


AVL Tree Data Structure

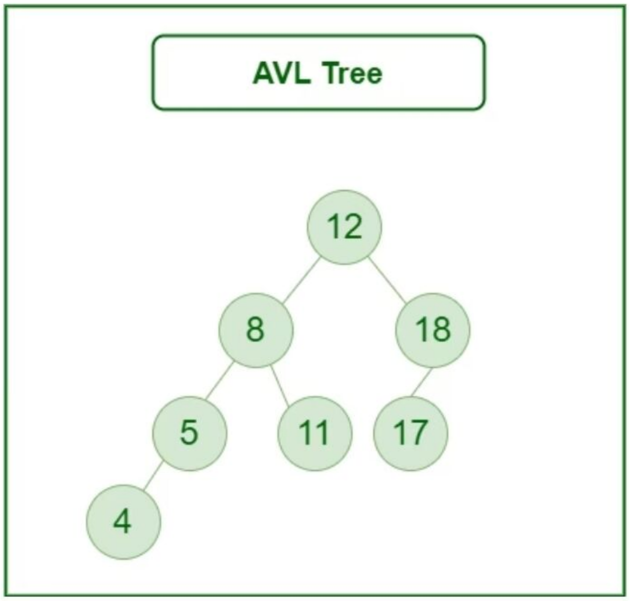
> An **AVL tree** defined as a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees for any node cannot be more than one.

The difference between the heights of the left subtree and the right subtree for any node is known as the **balance factor** of the node.

The AVL tree is named after its inventors, Georgy Adelson-Velsky and Evgenii Landis, who published it in their 1962 paper “An algorithm for the organization of information”.

Example of AVL Trees:

Example of AVL Trees:



AVL tree

The above tree is AVL because the differences between the heights of left and right subtrees for every node are less than or equal to 1.

Operations on an AVL Tree:

- Insertion
- Deletion
- Searching [It is similar to performing a search in BST]

Rotating the subtrees in an AVL Tree:

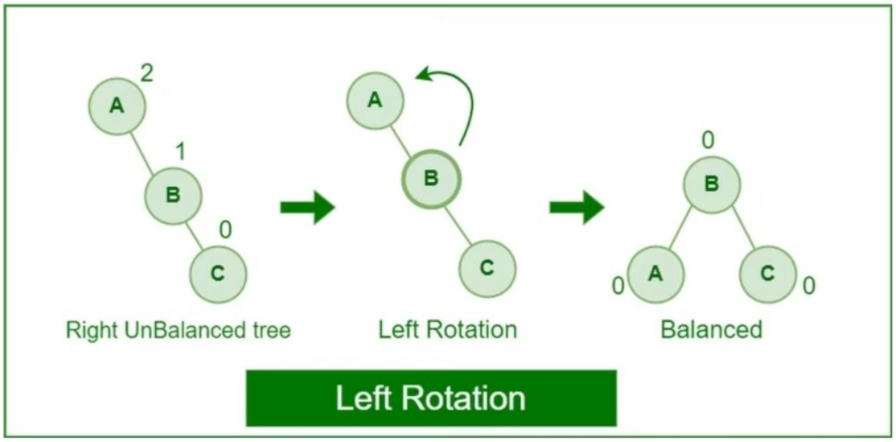
An AVL tree may rotate in one of the following

Rotating the subtrees in an AVL Tree:

An AVL tree may rotate in one of the following four ways to keep itself balanced:

Left Rotation:

When a node is added into the right subtree of the right subtree, if the tree gets out of balance, we do a single left rotation.



Left-Rotation in AVL tree

Right Rotation:

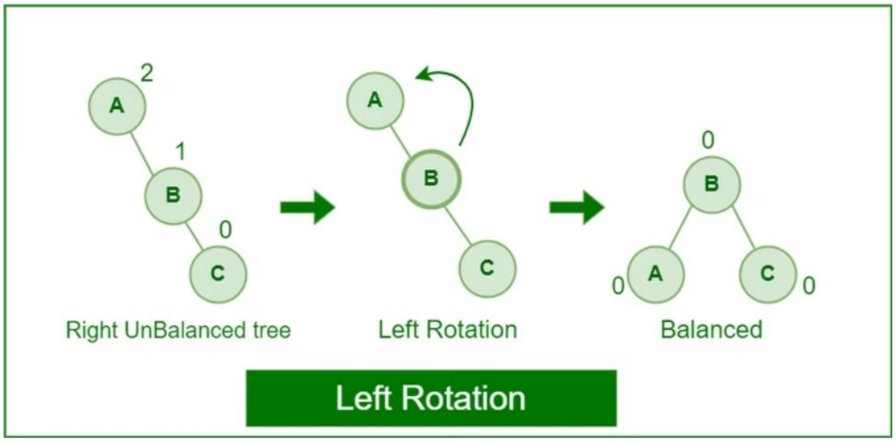
If a node is added to the left subtree of the left subtree, the AVL tree may get out of balance, we do a single right rotation.

Rotating the subtrees in an AVL Tree:

An AVL tree may rotate in one of the following four ways to keep itself balanced:

Left Rotation:

When a node is added into the right subtree of the right subtree, if the tree gets out of balance, we do a single left rotation.



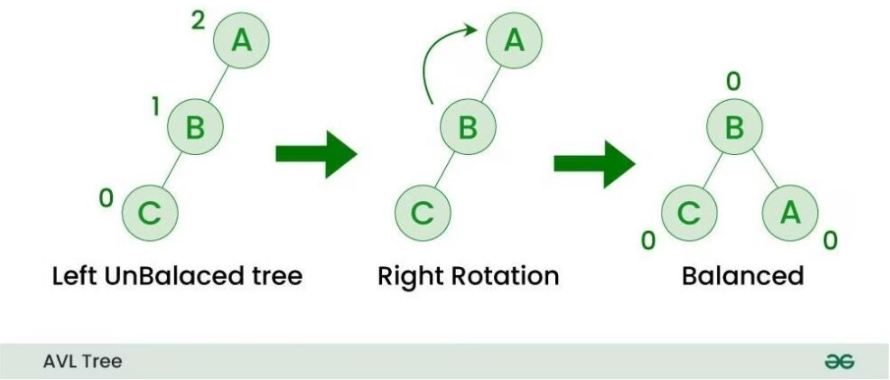
Left-Rotation in AVL tree

Right Rotation:

If a node is added to the left subtree of the left subtree, the AVL tree may get out of balance, we do a single right rotation.

Right Rotation:

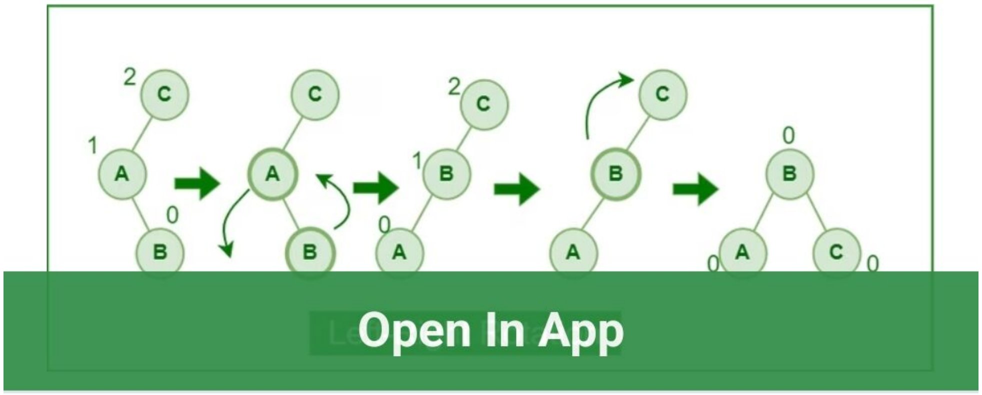
If a node is added to the left subtree of the left subtree, the AVL tree may get out of balance, we do a single right rotation.



Right-Rotation in AVL Tree

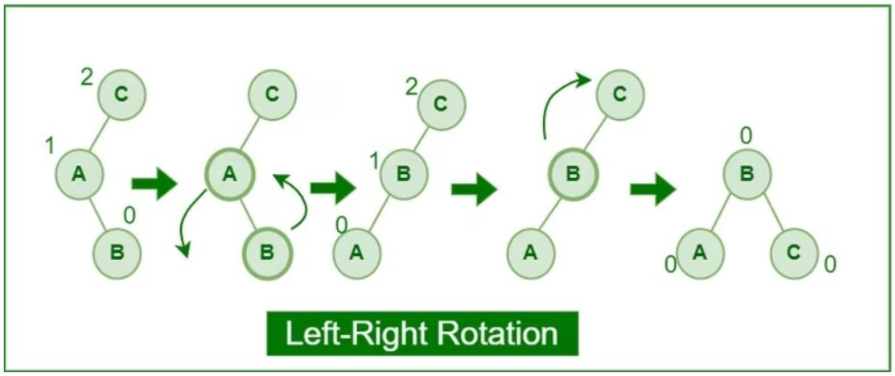
Left-Right Rotation:

A left-right rotation is a combination in which first left rotation takes place after that right rotation executes.



Left-Right Rotation:

A left-right rotation is a combination in which first left rotation takes place after that right rotation executes.



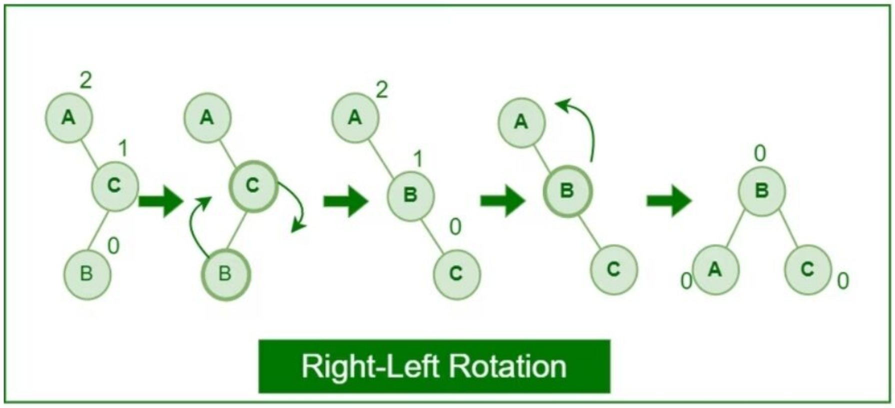
Left-Right Rotation in AVL tree

Right-Left Rotation:

A right-left rotation is a combination in which first right rotation takes place after that left rotation executes.

Right-Left Rotation:

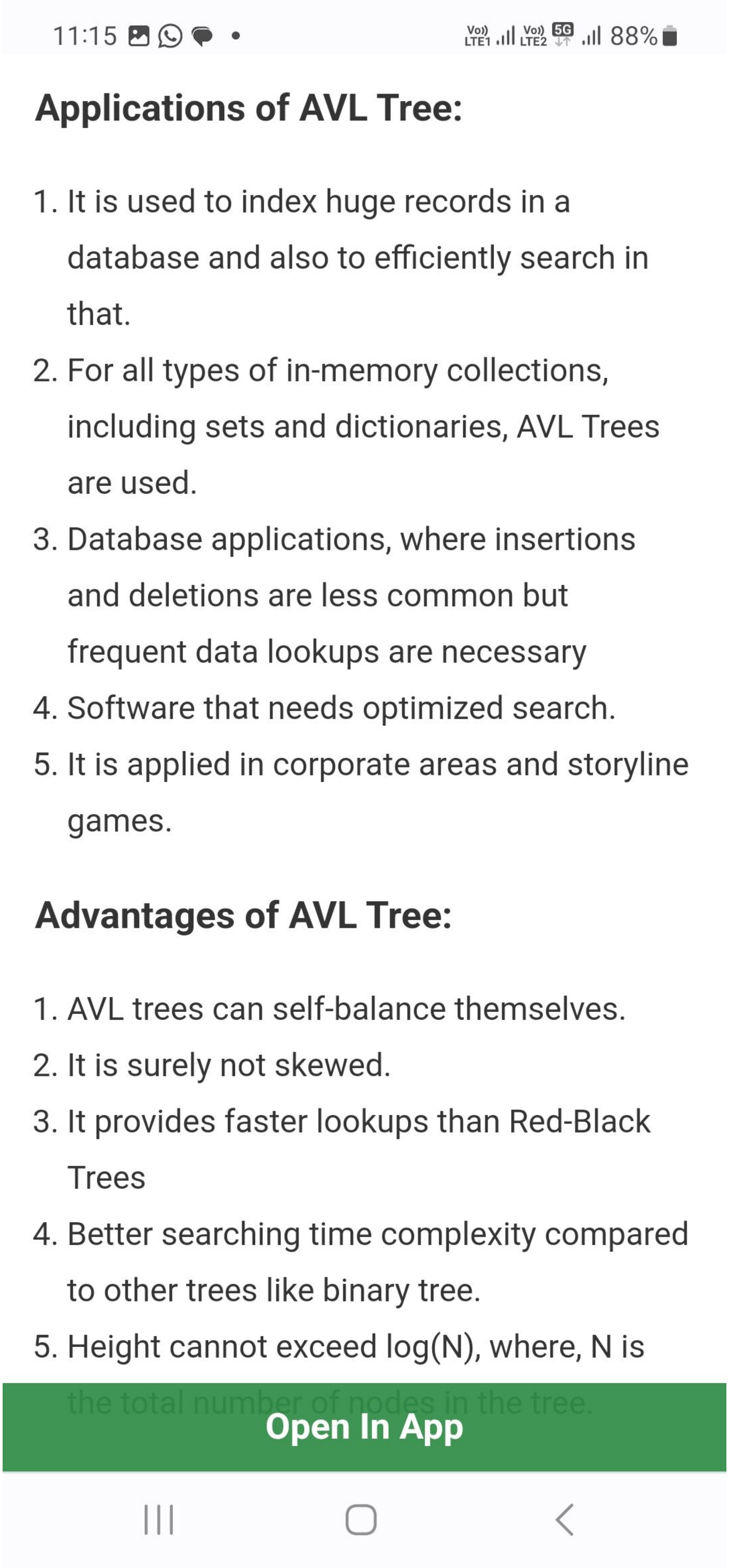
A right-left rotation is a combination in which first right rotation takes place after that left rotation executes.



Right-Left Rotation in AVL tree

Applications of AVL Tree:

1. It is used to index huge records in a database and also to efficiently search in that.
2. For all types of in-memory collections, including sets and dictionaries, AVL Trees are used.
3. Database applications, where insertions

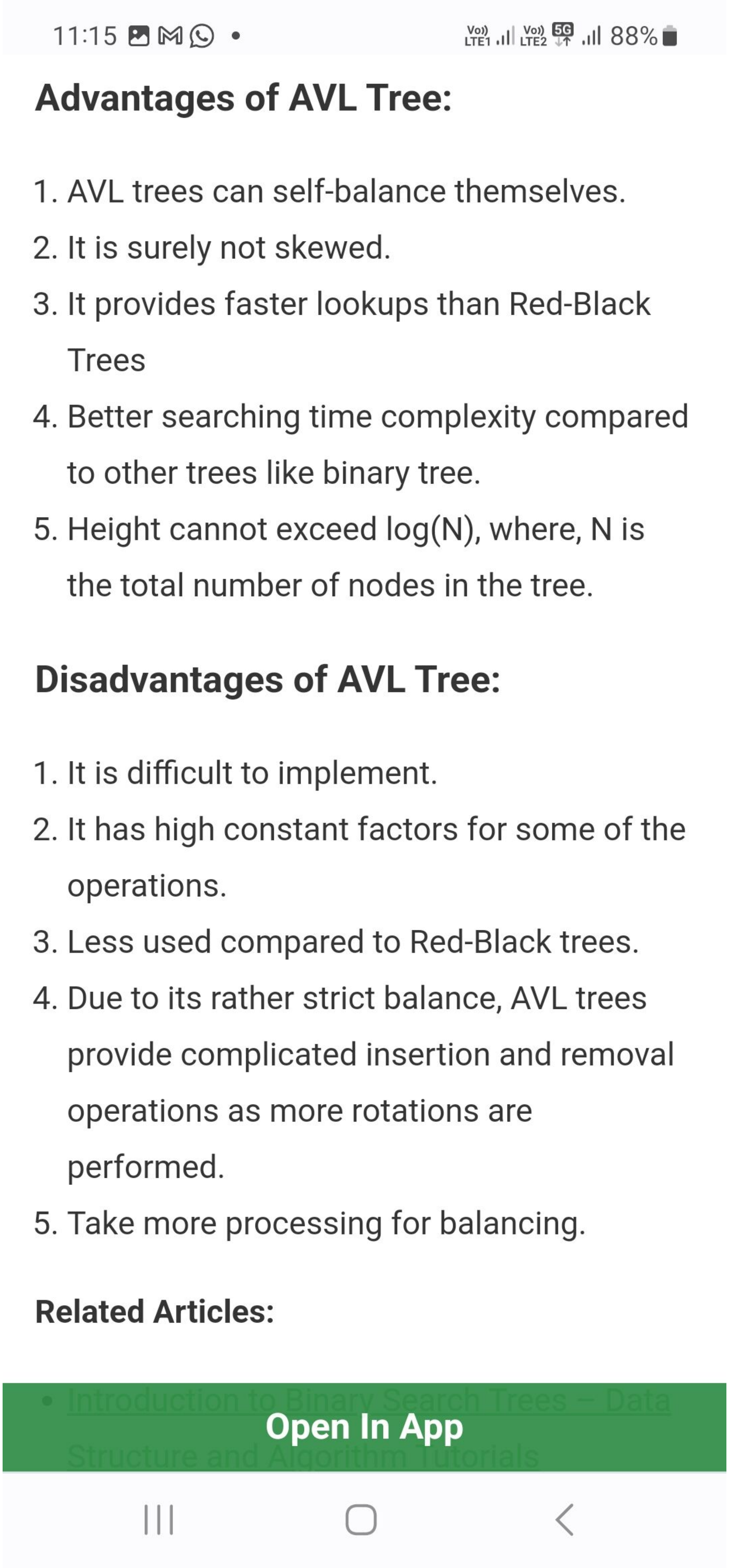


Applications of AVL Tree:

1. It is used to index huge records in a database and also to efficiently search in that.
2. For all types of in-memory collections, including sets and dictionaries, AVL Trees are used.
3. Database applications, where insertions and deletions are less common but frequent data lookups are necessary
4. Software that needs optimized search.
5. It is applied in corporate areas and storyline games.

Advantages of AVL Tree:

1. AVL trees can self-balance themselves.
2. It is surely not skewed.
3. It provides faster lookups than Red-Black Trees
4. Better searching time complexity compared to other trees like binary tree.
5. Height cannot exceed log(N), where, N is the total number of nodes in the tree.



Advantages of AVL Tree:

1. AVL trees can self-balance themselves.
2. It is surely not skewed.
3. It provides faster lookups than Red-Black Trees
4. Better searching time complexity compared to other trees like binary tree.
5. Height cannot exceed log(N), where, N is the total number of nodes in the tree.

Disadvantages of AVL Tree:

1. It is difficult to implement.
2. It has high constant factors for some of the operations.
3. Less used compared to Red-Black trees.
4. Due to its rather strict balance, AVL trees provide complicated insertion and removal operations as more rotations are performed.
5. Take more processing for balancing.

Related Articles:

- Introduction to Binary Search Trees – Data Structure and Algorithm Tutorials