Stack, binary tree, BST, BBST (balanced and AVL balanced tree)

Stack

- A LIFO data structure (Last in First out)
- Adding and removing (push and pop) takes $O(1)$

Balanced BST

- The difference between the height of the left and the right subtree for each node is either 0 or 1.
- It ensures that time complexity of searching is $O(\log n)$
- AVL tress are an implementation of BBST

AVL tree

- Is a specific implementation of a BBST
- Rebalancing an AVL tree is performed after every deletion/insertion.
- Balance factor is a property of every node, it is equal to height of the left subtree - height of the right subtree. The balance factor should always be -1, 0, or 1.
- The procedure for insertion is:
    - The new node always gets the balance factor equal to 0.
    - Find the correct location of the new node and insert it there (the same algorithm as for normal BST)
    - Update the balance factor of every node

- Then starting from the parent of the new node perform those operations:
    - If its balance factor > 1:
        - If the balance factor of the left child is < 0: do the left-right rotation
        - Else do the right rotation
    - If its balance factor < -1:
        - If the balance factor of the right child is > 0: do right-left rotation
        - Else do left rotation