- AVL tree (**A**delson-**V**elsky and **L**andis), es un self-balancing binary search tree.
- Fue el primer tipo de self-balancing binary search tree
- En esta estructura los sub-trees de los hijos pueden variar máximo por una altura de 1. Si es que esto no se cumple, se tiene que seguir un método para balancear el árbol.
- Las operaciones básicas son de orfen O(log n)
- Las operaciones son:
    - Searching
        - El método de searching es igual en un árbol balanceado tanto como no balanceado por lo que el AVL tree utiliza una búsqueda como la de un binary search tree común.
        - O(log n)
    - Traversal
        - El método de traversal también funciona igual en un árbol no balanceado tanto como un árbol balanceado.
        - O(log n)
    - Insert
        1. Al inicio se inserta el dato en el árbol como un binary search tree no balanceado
        2. Después de esta inserción se sabe que solo los ancestros pueden estar no balanceados por lo que necesitar checar cada ancestro por un desbalance según las reglas de AVL. A esto se le llama “retracing”.
            - Para esto se considera el “Balance Factor” donde se consigue la diferencia de altura y si la diferencia es -1,0,1 el árbol esta balanceado. → si abs(right-left) <= 1 .O → el árbol varían por una altura de máximo 1
        3. Debido a que la altura solo puede cambiar máximo por uno, la diferencia de altura puede ser máximo de ±2, si la diferencia es de ±2 se hacen “rotations” para el rebalanceo el árbol
        4. Las rotaciones y el balanceo se hace en sentido de child->parent donde si de la leaf insertada se balancean los padres siguiendo la línea a la raíz, el arbol terminará balanceado
        - O(log n)
    - Delete
        1. Este método es similar al insertion donde al incio sigue el método de un binary search tree no balanceado
        2. Después de borrar se busca en los ancestros por nodos que tengan un balance factor diferente a -1,0,1
            - El balance factor solo puede ser máximo de abs(±2)
        3. Se hacen rotations en caso de que no este balanceado un nodo
        4. Se sigue del subtree en el que se borró a los ancestros
        - O(log n)
- Los tipos de abalanzamiento:
    - Hay 4 diferentes posibilidades para los balanceos que se pueden que hacer

        - [RR] El hijo desbalanceado se encuentra a la derecha y el balance factor del hijo desbalanceado (rightHeight-leftHeight) ≥ 0
            - Se necesita hacer rotate_left
        - [LL] El hijo desbalanceado se encuentra a la izquieda y el balance factor del hijo desbalanceado (rightHeight-leftHeight) ≤ 0
            - Se necesita hacer rotate_right
        - [RL] El hijo desbalanceado se encuentra a la derecha y el balance factor del hijo desbalanceado (rightHeight-leftHeight) < 0
            - Se necesita hacer rotate_RightLeft
        - [LR] El hijo desbalanceado se encuentra a la izquieda y el balance factor del hijo desbalanceado (rightHeight-leftHeight) > 0
            - Se necesita hacer rotate_LeftRight
        -
    - Simple rotation (right || left)
        - …
    - Double rotation (rightLeft || leftRight)
        - …