```
class AVLTree {
  public root: AVLNode | null;
  constructor() {
    this.root = null;
  }
  private getHeight(node: AVLNode | null): number {
    return node ? node.height : 0;
  }
  private updateHeight(node: AVLNode): void {
    node.height =
      1 + Math.max(this.getHeight(node.left),
this.getHeight(node.left));
  }
  private getBalanceFactor(node: AVLNode): number {
    return this.getHeight(node.left) - this.getHeight(node.right);
  }
  public insert(key: number): void {
    this.root = this.insertData(this.root, key);
  }
  private insertData(node: AVLNode | null, key: number): AVLNode {
    if (!node) {
      return new AVLNode(key);
    } else if (key < node.key) {
      node.left = this.insertData(node.left, key);
      node;
    } else if (key > node.key) {
      node.right = this.insertData(node.right, key);
      node;
    } else {
      return node;
    }
    this.updateHeight(node);
    let balance: number = this.getBalanceFactor(node);
    if (balance > 1) {
```

```
    let select = node.left as AVLNode;

    if (key < select.key) {
      return this.rightRotate(node);
    } else {
      node.left = this.leftRotate(node.left as AVLNode);
      return this.rightRotate(node);
    }
  } else if (balance < -1) {
    let select = node.left as AVLNode;

    if (key > select.key) {
      return this.leftRotate(node);
    } else {
      node.right = this.rightRotate(node.left as AVLNode);
      return this.leftRotate(node);
    }
  }
  return node;
}

private rightRotate(node: AVLNode): AVLNode {
  let x: AVLNode = node.left as AVLNode;
  let T2 = x.right as AVLNode;
  x.right = node;
  node.left = T2;
  this.updateHeight(node);
  this.updateHeight(x);
  return x;
}
private leftRotate(node: AVLNode): AVLNode {
  let x: AVLNode = node.right as AVLNode;
  let T2 = x.left as AVLNode;
  x.right = node;
```

```
    node.left = T2;
    this.updateHeight(node);
    this.updateHeight(x);
    return x;
  }
  public inOrderTraversal(node: AVLNode | null): void {
    if (node) {
      this.inOrderTraversal(node.left);
      console.log(node.key);
      this.inOrderTraversal(node.right);
    }
  }
}

class AVLNode
{
    key: number;
    left:AVLNode | null ;
    right:AVLNode | null;
    height : number;
    constructor(key :number)
    {
        this.key=key;
        this.left=null;
        this.right=null;
        this.height=1;
    }
}
```