Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

**ОТЧЕТ**

по лабораторной работе №2. «Двоичные деревья поиска» по дисциплине «Алгоритмы и структуры данных»

Выполнил студент 1 курса, группы К3140

Байков Иван

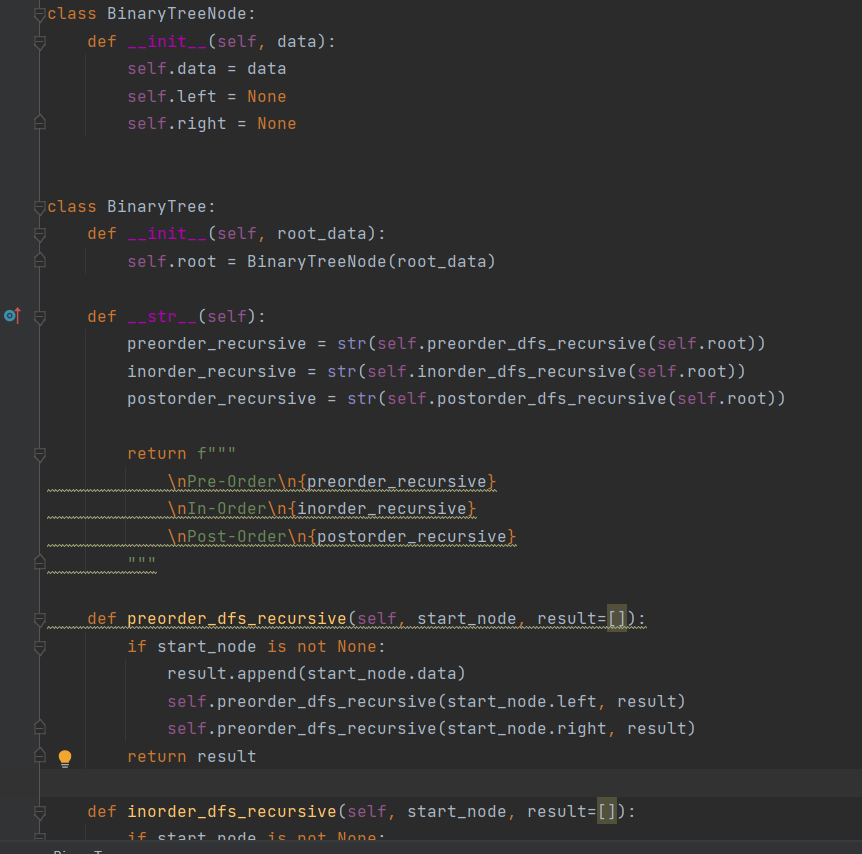
Преподаватель: Харьковская Татьяна Александровна

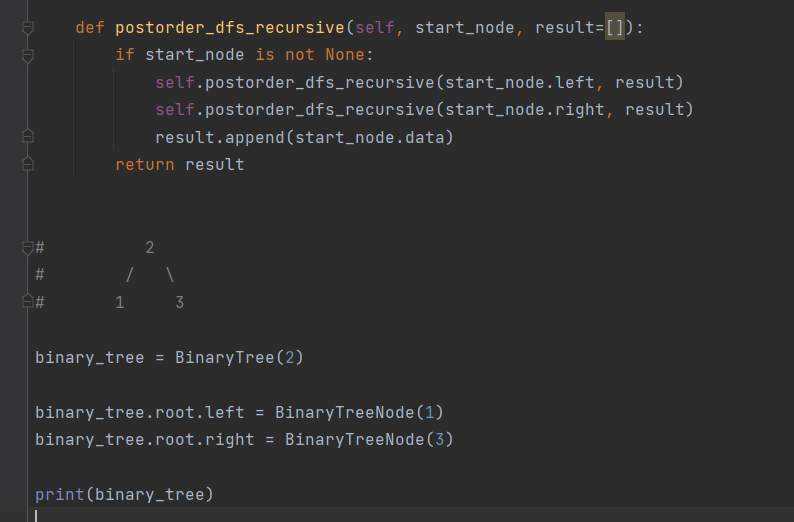
05 Июня, 2022 года, г. Санкт-Петербург

**Задание 1**

**Описание**

В этой задаче вы реализуете три основных способа обхода двоичного дерева «в глубину»: центрированный (in-order), прямой (pre-order) и обратный (post-order).

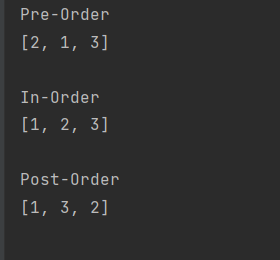
**Исходный код**



**Решение**

Мы обходим дерево по определенным порядкам, стартовой точкой (entry point) всегда является корень, но он не обязан быть первым в списке обхода. Обход реализован 3 строчками кода, где одна – это добавление результата, другие две – рекурсивный вызов левого или правого поддерева. В зависимости от типа обхода эти функции выполняются в разном порядке.

**Тесты**



Взял простое дерево [2,1,3], чтобы легче показать на примере.

**Вывод**

Было полезно попрактиковаться в реализации алгоритмов обхода, чтобы лучше понимать такую полезную структуру данных и как с ней взаимодействовать.

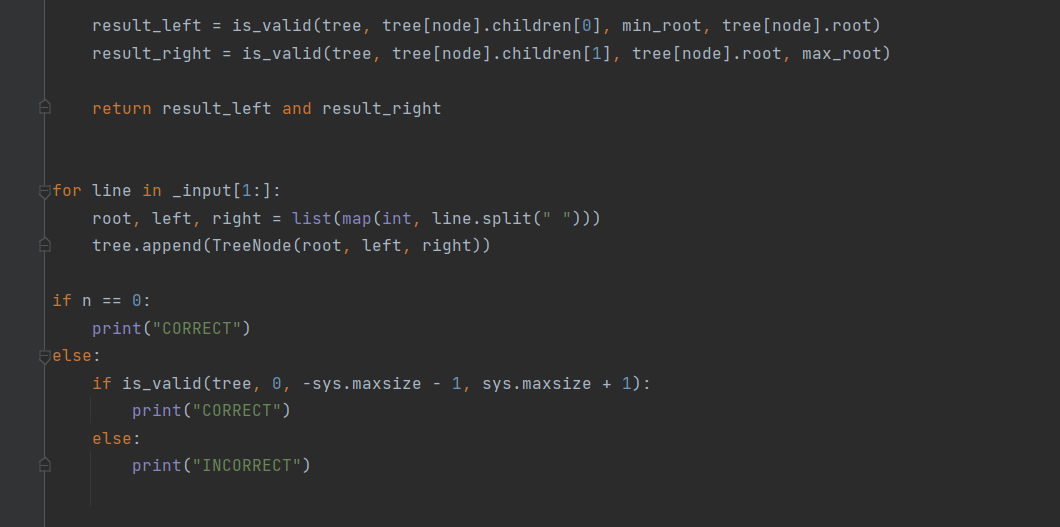
**Задание 6-7**

**Описание**

В данной задаче необходимо проверить, правильно ли реализована структура BST. В 7 задаче усложнение – деревья могут содержать равные ключи.

**Исходный код**





**Решение**

При выполнении In-order DFS каждый текущий узел во время обхода можно добавлять в массив, тогда в конце обхода можно наглядно убедиться, является ли данное дерево двоичным деревом поиском. Суть в том, что при таком обходе, каждый следующий элемент массива будет больше или равен (7) предыдущему, поэтому простым циклом можно это проверить.

**Тесты**

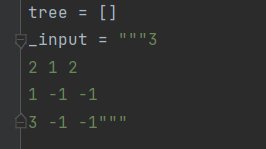


Рисунок 12

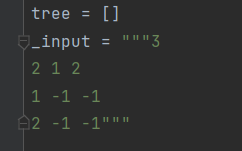


Рисунок 14

Были проверены два теста из условия, один на CORRECT, другой на INCORRECT.

**Вывод**

Не самая сложная задача, если решать через рекурсивный подход. Помогает детальнее изучить BST и узнать некоторые свойства. Асимптотика алгоритма – O(n), где n – количество узлов.

**Задание 11-15**

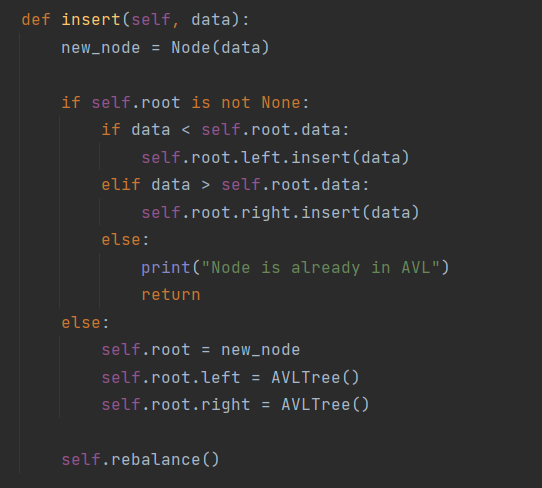
**Описание**

Если объединить все задания, то нужно реализовать AVL дерево, в котором можно добавлять/удалять узлы, искать высоту, след/пред узел, делать левый и правый повороты и т.д.

**Исходный код**

На гитхабе, некоторый функционал в решении.

**Решение**

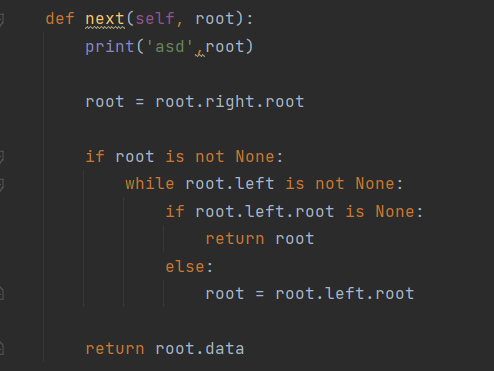


В первую очередь реализована вставка в дерево с автоматической ребалансировкой, выполняется с помощью массива:

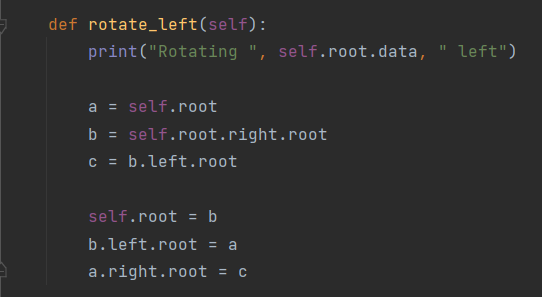




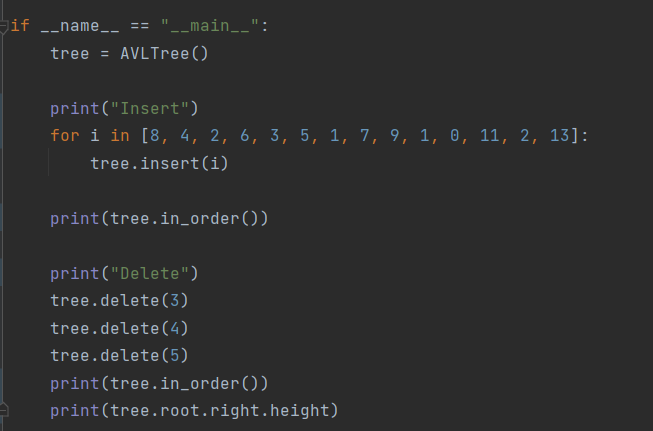
Для удаления сначала рекурсивно проходимся по дереву и ищем узел с нужным значением, затем заменяем узел на None либо заменяем (если есть дочерние узлы) его данные на дочерние. Для поиска дочерних используем метод next().

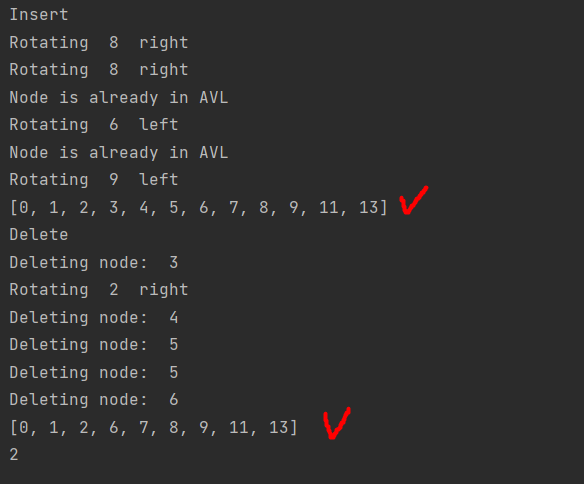


Для балансировки делаем левый или правый повороты.



**Тесты**





Взяты рандомные небольшие значения для лучшего понимания происходящего. Галочками отмечены состояния дерева во время тестов. Как видно они балансируются до BST.

**Вывод**

Проработан основной функционал AVL деревьев, получены практические знания о реализации и удобстве хранения данных в таком виде.

**Задание 17**

**Описание**

В этой задаче ваша цель – реализовать структуру данных для хранения набора целых чисел и быстрого вычисления суммы элементов в заданном диапазоне.

Методы: add, del, find, sum.

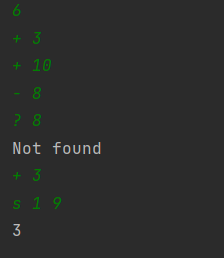
**Исходный код**

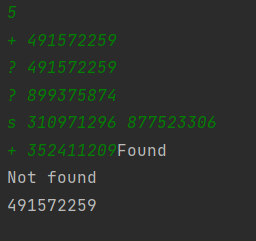
Так же на гитхабе.

**Решение**

Каждая вершина имеет ключ, представляющий целое число, сумму, равную сумме всех ключей в поддереве, указатель на его левую дочернюю вершину, правый указатель на его правую дочернюю вершину и родительский элемент, указывающий на родительскую вершину. Для удобства дерево делится на 2 поддерева используя указанный ключ.

**Тесты**





Тесты из условия + рандомные, проверяют весь функционал.

**Вывод**

Реализовано Splay дерево, позволяющее хранить и удаления данных с возможностью быстрого вычисления суммы элементов в заданном диапазоне. Непростое задание с основной сложность в виде реализации Splay дерева с ключами, split и merge методами.

.