Algorithms With Java

Week 2 Day 0: [Не]сбалансированное бинарное дерево

Академия Ковалевского

Содержание

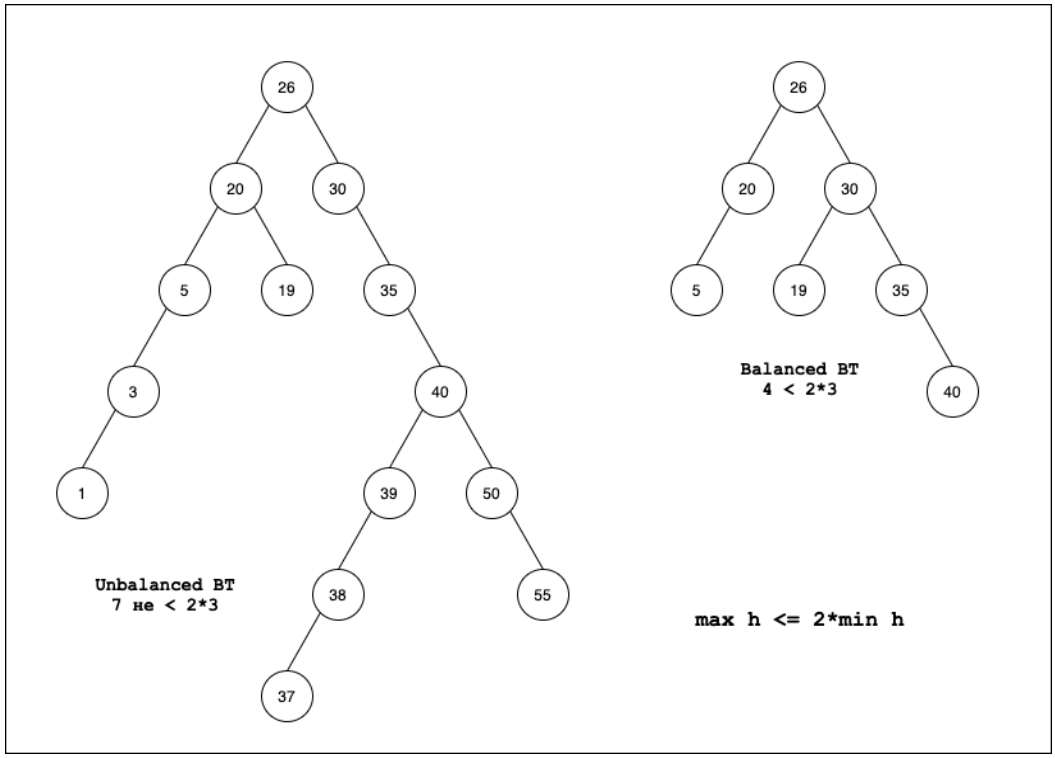
1. Теория

[Не]сбалансированное бинарное дерево

Сбалансированное бинарное дерево в основном используется в запросах поиска данных. Такое дерево максимально широкое и низкое. Дерево **идеально** сбалансировано, если для каждого его узла количество узлов в левом и правом поддереве различается не более чем на 1.

Сбалансированное бинарное дерево поиска — это бинарное дерево поиска с логарифмической высотой. Данное определение скорее идейное, чем строгое. Строгое определение оперирует разницей глубины самого глубокого и самого неглубокого листа (в AVL-деревьях) или отношением глубины самого глубокого и самого неглубокого листа (в красно-черных деревьях). В сбалансированном бинарном дереве поиска операции поиска, вставки и удаления выполняются за логарифмическое время (так как путь к любому листу от корня не более логарифма).

Важно! В данном дне мы будем понимать под сбалансированным дерево такое дерево при котором длина самого длинного пути от вершины дерева не превышает удвоенного минимального расстояния:



Ссылки:

- Как определить, что дерево сбалансированное

2. Практическая работа

IntTreeNode

Этот класс описывает каждую вершину нашего дерева. Он содержит значение в текущей ноде, сумму всех значений в нодах ниже текущей, количество всех нод внизу этой вершины включая саму ноду, минимальную и максимальную глубину от этой ноды, и ссылки на левую и правую ноды.

```
package academy.kovalevskyi.algorithms.week2.day0;

public record IntTreeNode(
        int value,
        int sum,
        int count,
        int maxDepth,
```

```
        int minDepth,
        IntTreeNode left,
        IntTreeNode right) {
}
```

Идея в том что храня в ноде все эти данные мы можем со сложностью O(1) выполнять операции:

- Посчитать сумму
- Посчитать количество
- Проверить является ли дерево сбалансированным или нет

При этом на сложность добавления новых нодов это не влияет!

IntTreeHelper

Это класс, который содержит разные методы по манипуляции графом. В нем нужно реализовать:

1. Метод createNode() — создаёт новую ноду с заданным значением.
2. Метод addNode() — добавляет значение в дерево.
3. Метод needBalancing() — говорит, нужна ли балансировка дереву или нет.
4. Метод getSortedList() — делает из дерева отсортированный список.
5. Метод generateBalanceTree() — делает из дерева сбалансированное дерево.
6. Метод hasValue() — смотрит, если ли в дереве такое значение или нет.

```
package academy.kovalevskyi.algorithms.week2.day0;

import java.util.ArrayList;
import java.util.List;

public class IntTreeHelper {
  public static IntTreeNode createNode(int value) {
    // TODO
  }

  public static IntTreeNode addNode(IntTreeNode root, int value) {
    // TODO
  }
  public static boolean needBalancing(IntTreeNode root) {
    // TODO
  }
  public static List<Integer> getSortedList(IntTreeNode root) {
    // TODO
  }

  public static IntTreeNode generateBalanceTree(IntTreeNode root) {
    // TODO
  }

  public static boolean hasValue(IntTreeNode root, int value) {
    // TODO
}
```