МИНОБРНАУКИ РОССИИ САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**ОТЧЕТ**

по лабораторной работе № 2

дисциплина «Алгоритмы и Структуры данных»

Тема: «Самобалансирующие двоичные деревья поиска»

Студент гр. 3351 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Фабер К.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пестерев Д.О.

Санкт-Петербург

2024

**Цель лабораторной работы: реализация самобалансирующихся деревьев поиска и экспериментальная проверка оценок высоты данных деревьев.**

**Теоретическая часть.**

**АВЛ дерево.**

АВЛ дерево – двоичное дерево поиска, у которого для любого узла X: , где ( – высота левого поддерева X, – высота правого поддерева X).

Алгоритм вставки в АВЛ дерево: сравниваем значение ключа с ключом текущего узла, начиная с корня, в зависимости от результата сравнения идем либо в правый узел, либо в левый (если значение ключа нового элемента больше ключа текущего узла – вправо, иначе – влево), после полного прохождения дерева, когда текущей узел указывает на нулевой указатель, вставляем новый элемент и привязываем его к родителю левым или правым ребенком, в зависимости от значения ключа, далее, для того чтобы сохранялись свойства АВЛ дерева, необходимо произвести балансировку: берем самый нижний затронутый узел, вызываем от него один из поворотов при необходимости и рекурсивно повторяем от родителя.

Алгоритм удаления в АВЛ дереве: путём сравнения ключей, приходим к узлу дерева, который нужно удалить, после нахождения данного узла может быть 3 случая:

1. Найденный элемент не имеет детей (лист)
2. Найденный элемент имеет одного ребенка
3. Найденный элемент имеет двух детей

В первом случае необходимо просто удалить узел, во втором – перед удалением нужно произвести замену указателей: ребенок найденного узла становится ребенком родителя найденного узла, в 3 случае – ищем наименьший элемент больше данного (т.е. идем один раз вправо и влево до конца), в результате этот элемент имеет не больше одного поддерева, меняем местами данный элемент с удаляемым и вызываем функцию удаления для правого поддерева текущего элемента и получаем первый, либо второй случай. После удаления также необходимо произвести балансировку.

Верхняя оценка высоты АВЛ дерева:

Пусть - минимальное количество узлов в АВЛ дереве высоты h, тогда из определения АВЛ дерева можно получить, что ;

Можно заметить, что , где – h + 2 число Фибоначчи

*,* где *,* тогда *,*

Следовательно

где k – количество элементов, тогда

**Красно-черное дерево.**

Красно-черное дерево – это двоичное дерево поиска, с разностью высот дочерних поддеревьев всех узлов 1, а также каждый узел имеет свой цвет: красный или черный, определяемый свойствами красно-черного дерева.

Свойства красно-черного дерева:

1. Корень всегда черный
2. Потомки красного узла не могут быть красными
3. Существуют фиктивные узлы – черные листья
4. Путь из корня до любого листа содержит одинаковое количество черных вершин

**Практическая часть.**

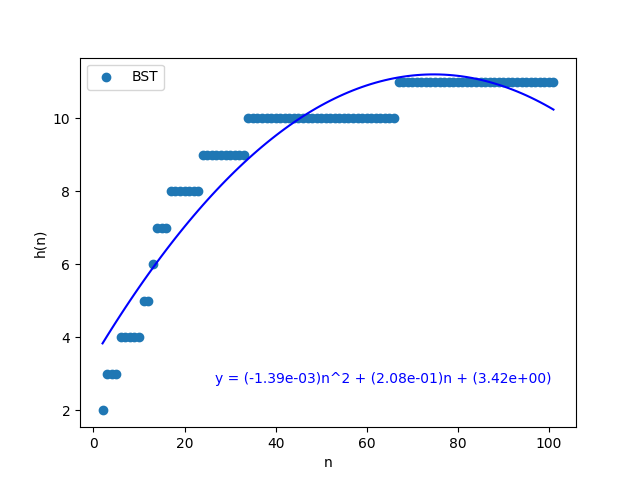
****

Рисунок 1 – График зависимости высоты h от количества элементов n для бинарного дерева поиска

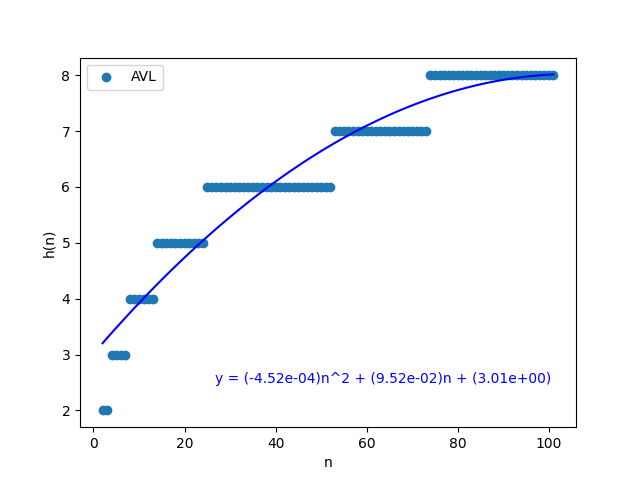
****

Рисунок 2 – График зависимости высоты h от количества элементов n для АВЛ дерева

Рисунок 3 – График зависимости высоты h от количества элементов n для красно-черного дерева

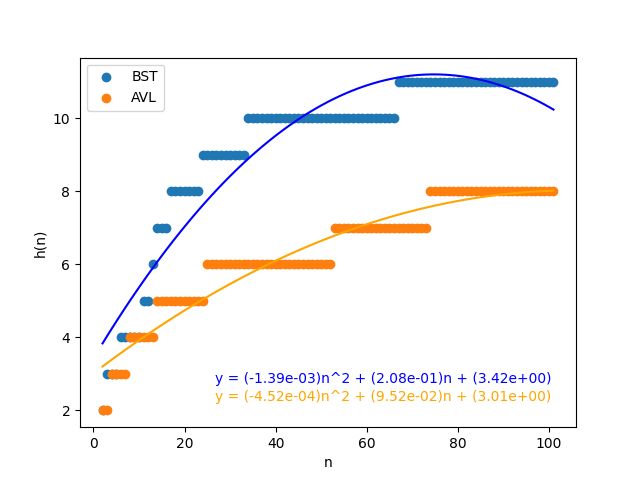
****

Рисунок 4 – График зависимости высоты h от количества элементов n для всех деревьев

Сравнивая теоретические данные, с практическими, можно заметить, что асимптотика высоты действительно логарифмическая. Высота бинарного дерева поиска растет быстрее, это объясняется тем, что бинарное дерево поиска никак не балансируется при вставке нового ключа, в отличии от АВЛ дерева и красно-черного дерева

**Ссылки:**

Репозиторий GitHub: <https://github.com/KIRILLFABER/AICD2>

**Код:**

**main.cpp**

**Data.cpp**

**Data.h**

**BST.cpp**

**BST.h**

**AVL.cpp**

**AVL.h**

**RB.cpp**

**RB.h**