МИНОБРНАУКИ РОССИИ

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

Кафедра САПР

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: «Самобалансирующиеся двоичные деревья»

Студент гр. 2302		Фролов А. Э.
Преподаватель:		Пестерев Д. О.

Санкт-Петербург

2023

1. Постановка задачи

Реализовать двоичное дерево поиска, красно-черное дерево и АВЛ-дерево. Сравнить высоты деревьев на случайном наборе входных данных, распределенных случайно. Сравнить временные затраты на балансировку для красно-черного и АВЛ-дерева при удалении элементов, при вставке элементов.

Отдельно реализовать функции обхода по дереву: в глубину, в ширину, прямого (preorder), обратного (postorder), симметричный (inorder).

2. Сравнение высот деревьев

Очевидно, что высота дерева зависит от его сбалансированности. Согласно этому, высота бинарного дерева будет наибольшей, а высота красно-черного – наименьшей. При этом входные данные не гарантируют, что в какой-то момент времени высоты некоторых деревьев могут быть равны.

Поэтому теоретическая оценка будет следующая:

$$\boldsymbol{height(AVL) \leq height(RBT) \leq height(BST)} \qquad \textbf{(2.1)}$$

За счет того, что АВЛ-дерево является сбалансированным (высоты правого и левого поддеревьев различаются не более, чем на 1), его высота оценивается следующим образом:

$$\boldsymbol{h_{AVL} = O(\log(n))} \qquad \textbf{(2.2)}$$

Аналогичным образом за счет частичной сбалансированности красно-черного дерева (высоты левого и правого поддеревьев различаются не более, чем на 1), его оценка будет следующая:

$$\boldsymbol{h_{RBT} = O(\log(n))} \qquad \textbf{(2.3)}$$

Высоту бинарного дерева однозначно оценить не получится, т.к. в худшем случае она может оцениваться как O(n), а в лучшем – O(log(n)). Однако при большом количестве случайных входных данных ширина дерева будет постепенно расти, многократно опережая рост высоты, а значит, теоретическая оценка в этом случае будет следующей:

$$\boldsymbol{h_{BST} = O(\log(n))} \qquad \textbf{(2.4)}$$

В то же время, при строго отсортированном наборе выходных данных очевидно, что высота будет оцениваться следующим образом:

$$\boldsymbol{h_{RBT} = \theta(n)} \qquad \textbf{(2.5)}$$

Дополнительно рассмотрим точную оценку высот всех деревьев. За высоту дерева будем считать максимальный путь от корня до крайнего элемента. В сбалансированном бинарном дереве имеет место равенство:

$$\boldsymbol{N = 2^{h+1} - 1}$$

Отсюда высота сбалансированного бинарного дерева:

$$\boldsymbol{h_{BST} = [log(N + 1)] - 1}$$

, где [] означают округление вверх. Тогда для любого бинарного дерева имеет место неравенство:

$$[\boldsymbol{log}(\boldsymbol{N}+\mathbf{1})]-\mathbf{1} \leq \ \boldsymbol{h}_{\boldsymbol{BST}} \ \leq \boldsymbol{N} \tag{2.6}$$

Для красно-черного дерева, в свою очередь, имеют место неравенства:

$$\boldsymbol{bh} \geq \frac{\boldsymbol{h}}{\mathbf{2}}$$

$$\boldsymbol{N} \geq \ \mathbf{2}^{(\boldsymbol{bh}+\mathbf{1})}-\mathbf{1}$$

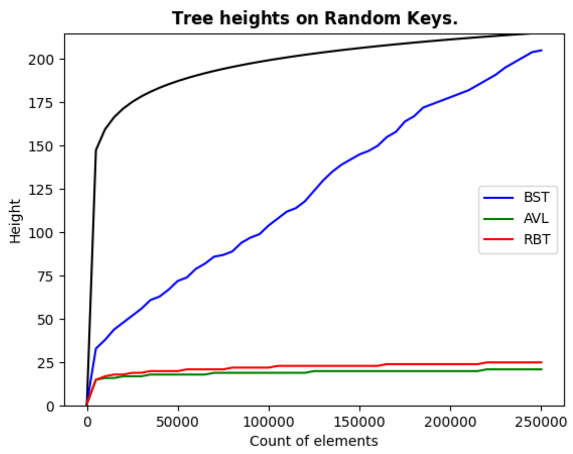
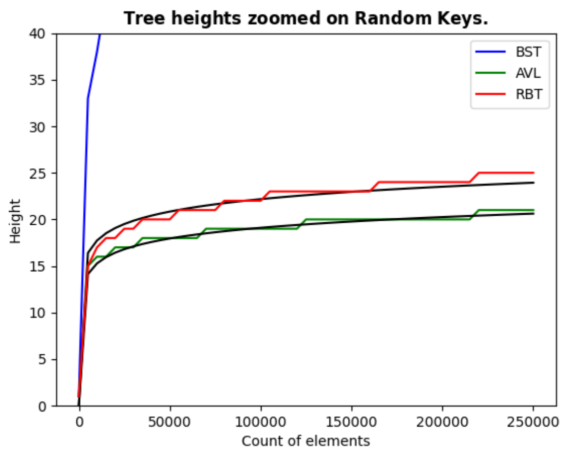
Выразим отсюда bh и сравним с h, после чего получим:

$$[\boldsymbol{log}(\boldsymbol{N}+\mathbf{1})]-\mathbf{1} \leq \ \boldsymbol{h}_{\boldsymbol{RBT}} \ <= \mathbf{2} * [\boldsymbol{log}(\boldsymbol{N}+\mathbf{1})-\mathbf{1}] \tag{2.7}$$

Для АВЛ дерева существует следующая формула (доказать, к сожалению, не смог):

$$[\boldsymbol{log}(\boldsymbol{N}+\mathbf{1})]-\mathbf{1} \ \leq \ \boldsymbol{h}_{\boldsymbol{AVL}} <= \mathbf{1.45} * \ \boldsymbol{log}(\boldsymbol{N}+\mathbf{2}) \tag{2.8}$$

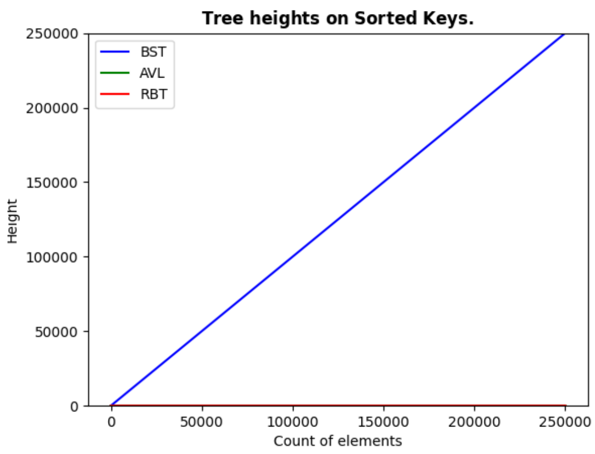
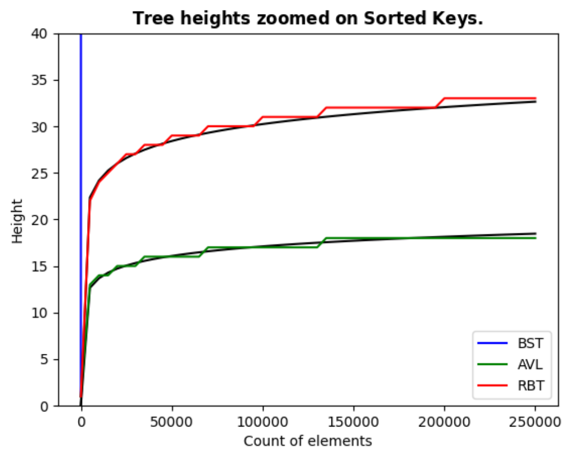
2.1. На случайном наборе входных данных

Общий график	Приближенный график
	
Черная линия задает функцию вида k*log(n), чтобы одноверенно и доказать теоретическую оценку 2.4, и показать что зависимость высоты бинарного дерева не иммет логарифмический вид.	Черные линии задают регрессию вида k * log(n) для высот АВЛ-дерева и КЧ-дерева.

Из графиков очевидно, что бинарное дерево имеет наибольшую высоту при текущих входных данных, а АВЛ-дерево – минимальную.

Экспериментальный результат подтвердил теоретические оценки (2.2, 2.3, 2.4).

2.2. На строго возрастающем наборе входных данных

Общий график	Приближенный график
	
Регрессионная функция не добавлена, т.к. график полностью соответсвует функции y = n	Черные линии задают регрессию вида k * log(n) для высот АВЛ-дерева и КЧ-дерева.

Из графиков очевидно, что бинарное дерево имеет наибольшую высоту при текущих входных данных, а АВЛ-дерево – минимальную.

Экспериментальный результат подтвердил теоретические оценки (2.2, 2.3, 2.5).

3. Сравнение времени балансировки для вставки и удаления

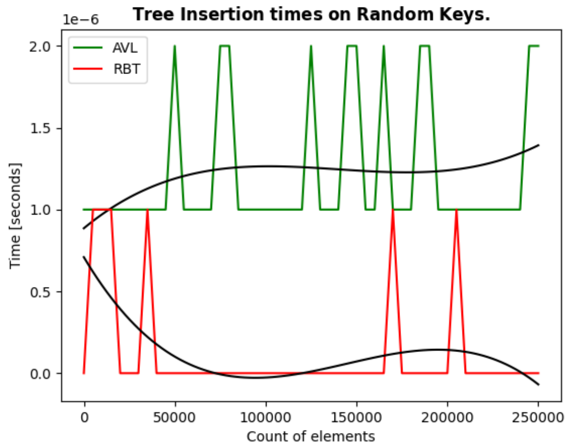
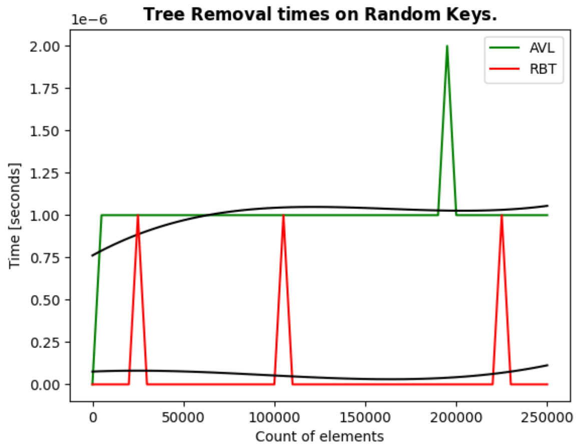
За счет того, что АВЛ-дерево является сбалансированным (высоты правого и левого поддеревьев различаются не более, чем на 1), время, затраченное на вставку и удаление будет оцениваться следующим образом (исходя из количества вызова рекурсии):

$$\boldsymbol{T_{AVL\_insert}} = \boldsymbol{T_{AVL\_remove}} = \boldsymbol{O}(\mathbf{log}(\boldsymbol{n})) \tag{3.1}$$

Аналогичным образом оценивается и красно-черное дерево, т.к. оно является почти сбалансированным (высоты левого и правого поддеревьев различаются не более, чем в два раза), т.к. также учитывается количество встречаемых элементов на пути к текущему узлу:

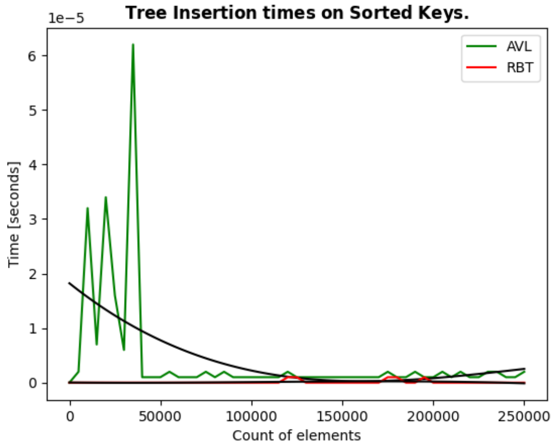
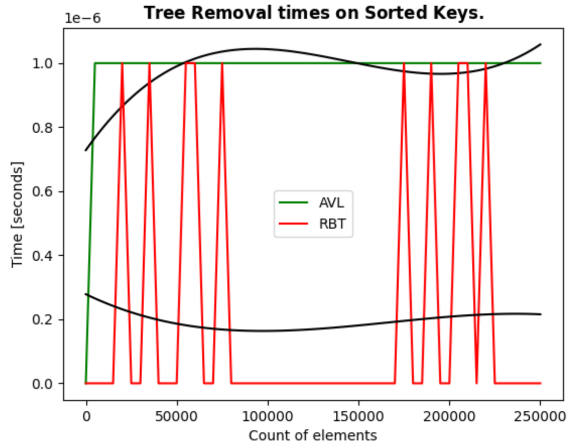
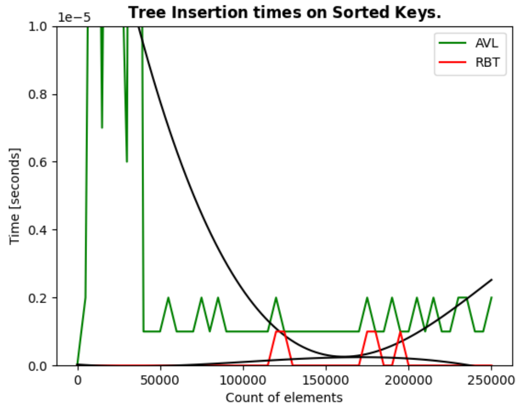
$$\boldsymbol{T_{RBT\_insert}} = \boldsymbol{T_{RBT\_remove}} = \boldsymbol{O}(\mathbf{log}(\boldsymbol{n})) \tag{3.2}$$

3.1. На случайном наборе входных данных:

Вставка	Удаление
	
Черные линии задают регрессию для времени АВЛ-дерева и КЧ-дерева. Зависимость не имеет теоретический вид (3.1, 3.2) из-за очень малого времени работы программы.	Черные линии задают регрессию для времени АВЛ-дерева и КЧ-дерева. Зависимость не имеет теоретический вид (3.1, 3.2) из-за очень малого времени работы программы.

Из графиков видно, что красно-черное дерево эффективней для вставки и удаления элементов. Обосновывается это тем, что **балансировка красно-черного дерева вызывается единожды** после вставки/удаления, а **балансировка АВЛ-дерева вызывается для каждого элемента на пути к текущему узлу**, чтобы поддерживать текущую высоту каждого узла и гарантировать сбалансированность.

3.2. На строго возрастающем наборе входных данных

<table>
<tr><th>Вставка</th><th>Удаление</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>Черные линии задают регрессию для времени АВЛ-дерева и КЧ-дерева. Зависимость не имеет теоретический вид (3.1, 3.2) из-за очень малого времени работы программы.</td><td>Черные линии задают регрессию для времени АВЛ-дерева и КЧ-дерева. Зависимость не имеет теоретический вид (3.1, 3.2) из-за очень малого времени работы программы.</td></tr>
<tr><th>Приближенный результат</th><th></th></tr>
<tr><td>
</td><td></td></tr>
</table>

Из графиков видно, что красно-черное дерево эффективней для вставки и удаления элементов. Обосновывается это тем, что **балансировка красно-черного дерева вызывается единожды** после вставки/удаления, а **балансировка АВЛ-дерева вызывается для каждого элемента на пути к текущему узлу**, чтобы поддерживать текущую высоту каждого узла и гарантировать сбалансированность.

Также можно заметить, что вставка элемента в АВЛ-дерево занимает гораздо больше времени в начале, чем при вставке случайных входных данных. Объяснить это можно тем, что при вставке строго отсортированных элементов балансировка будет вызываться для всех узлов, находящихся на пути к текущему, в то время как при вставке случайного элемента балансировка вызывается чаще всего для одного-двух узлов на пути к текущему. С ростом количества элементов время нормализируется, вероятно потому, что дерево становится достаточно большим для того, чтобы текущая сбалансированность позволяла не вызывать балансировку так часто.

4. Ссылка на GitHub

https://github.com/afrlfff/university-student-tasks/tree/master/algoritms-and-data-structures/lab3-binary-trees