Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № \_

Выполнил студент группы КС-36 (Потапов Никита Александрович)

Ссылка на репозиторий: (Ссылка на лабораторную в репозитории)

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: (14.04)

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы необходимо изучить одно из двух деревьев поиска:

1. Декартово дерево (https://habr.com/ru/post/101818/)

2. Рандомизированное дерево (https://habr.com/ru/post/145388/)

Для этого его потребуется реализовать и сравнить в работе с реализованным ранее AVL-деревом. Для анализа работы алгоритма понадобиться провести серии тестов:

- В одной серии тестов проводится 50 повторений

- Требуется провести серии тестов для N = 2^i элементов, при этом i от 10 до 18 включительно.

В рамках одной серии понадобится сделать следующее:

- Генерируем N случайных значений.

- Заполнить два дерева N количеством элементов в одинаковом порядке.

- Для каждого из серий тестов замерить максимальную глубину полученного деревьев.

- Для каждого дерева после заполнения провести 1000 операций вставки и замерить время.

- Для каждого дерева после заполнения провести 1000 операций удаления и замерить время.

- Для каждого дерева после заполнения провести 1000 операций поиска.

- Для каждого дерева замерить глубины всех веток дерева.

Для анализа структуры потребуется построить следующие графики:

- График зависимости среднего времени вставки от количества элементов в изначальном дереве для вашего варианта дерева и AVL дерева.

- График зависимости среднего времени удаления от количества элементов в изначальном дереве для вашего варианта дерева и AVL дерева.

- График зависимости среднего времени поиска от количества элементов в изначальном дереве для вашего варианта дерева и AVL дерева.

- График максимальной высоты полученного дерева в зависимости от N.

- Гистограмму среднего распределения максимальной высоты для последней серии тестов для AVL и для вашего варианта.

- Гистограмму среднего распределения высот веток в AVL дереве и для вашего варианта, для последней серии тестов.

# 

# Описание метода/модели.

Теоретическое описание структур данных и алгоритмов

1. AVL-дерево (Адельсон-Вельский и Ландис)

Определение:

AVL-дерево — это сбалансированное по высоте бинарное дерево поиска, в котором для каждой вершины высота её двух поддеревьев различается не более чем на 1.

Основные операции и их сложность:

- Вставка: O(log n)

- Обычный поиск места вставки: O(log n)

- Балансировка (повороты): O(1) на каждом уровне

- Удаление: O(log n)

- Поиск: O(log n)

- Балансировка: O(1) на каждый поворот

Потребление памяти:

- O(n) для хранения n элементов

- Каждый узел хранит:

- Ключ

- Указатели на потомков

- Высоту поддерева (4 байта)

Преимущества:

1. Гарантированная сбалансированность (высота ≈ 1.44 log₂(n+2))

2. Стабильное время операций

3. Эффективность для частых операций поиска

Проблемы:

1. Сложная реализация балансировки

2. Дополнительные накладные расходы на хранение высот

3. Частые перебалансировки при частых вставках/удалениях

Теоретическая высота:

Максимальная высота ≈ 1.44 log₂n

2. Рандомизированное дерево (Treap)

Определение:

Treap (Tree + Heap) — это гибридная структура, сочетающая свойства:

- Бинарного дерева поиска (по ключам)

- Кучи (по приоритетам)

Основные операции и их сложность (средний случай):

- Вставка: O(log n)

- Разделение и слияние поддеревьев: O(log n)

- Удаление: O(log n)

- Поиск: O(log n)

Потребление памяти:

- O(n) для хранения n элементов

- Каждый узел хранит:

- Ключ

- Приоритет (обычно 4 байта)

- Указатели на потомков

Преимущества:

1. Простота реализации (не нужны сложные балансировки)

2. Хорошая средняя производительность

3. Естественная балансировка за счёт рандомизации

4. Устойчивость к специально подобранным последовательностям

Проблемы:

1. Деградация до O(n) в худшем случае (крайне маловероятна)

2. Нет гарантированной сбалансированности

3. Требуется генератор случайных чисел

Теоретическая высота:

Ожидаемая высота ≈ 2 ln n (≈ 1.39 log₂n)

Сравнительный анализ

По памяти:

- AVL требует хранения высот (4 байта на узел)

- Treap требует хранения приоритетов (4 байта на узел)

- Общий расход памяти сопоставим

По балансировке:

- AVL поддерживает строгую балансировку

- Treap обеспечивает "статистическую" балансировку

Применимость:

- AVL предпочтителен для:

- Систем с жёсткими требованиями к времени отклика

- Приложений с частыми операциями поиска

- Treap предпочтителен для:

- Систем с частыми вставками/удалениями

- Когда важна простота реализации

- В условиях отсутствия атак специальными последовательностями

# Выполнение задачи.

Практическое описание решения

1. Используемые технологии и язык программирования

Язык программирования: C++

2. Организация программы

Программа состоит из следующих компонентов:

2.1. Реализация структур данных:

- Класс AVL (AVL-дерево)

- Методы: insert(), remove(), contains()

- Вспомогательные: rotateLeft(), rotateRight(), balance()

- Класс Treap (Рандомизированное дерево)

- Методы: insert(), remove(), contains()

- Вспомогательные: split(), merge()

2.2. Тестовая система:

- Генерация тестовых данных:

- Последовательности размером от 2¹⁰ до 2¹⁸ элементов

- Отсортированные и случайно перемешанные данные

- Измерение характеристик:

- Время операций (вставка, удаление, поиск)

- Глубина дерева (максимальная и распределение)

2.3. Система сбора результатов:

- Запись в CSV-файлы:

- results.csv — временные характеристики

- depths.csv — данные о глубинах

3. Проведенные тесты

Параметры тестирования:

- Количество серий: 9 (от 2¹⁰ до 2¹⁸ элементов)

- Повторений в серии: 50

- Количество операций для замера: 1000 каждой типа

Тестовые сценарии:

3.1. Тест вставки:

- Замер времени добавления 1000 элементов В уже построенное дерево

3.2. Тест удаления:

- Удаление 1000 случайных элементов Из полностью построенного дерева

3.3. Тест поиска:

- Поиск 1000 случайных элементов

3.4. Анализ структуры:

- Измерение максимальной глубины

- Сбор статистики по глубинам всех веток

4. Полученные результаты

Основные метрики:

4.1. Временные характеристики:

- AVL-дерево показывает стабильное время операций ≈ O(log n)

- Treap демонстрирует сравнимую производительность в среднем случае

4.2. Распределение глубин:

- AVL: узкое распределение вокруг медианы

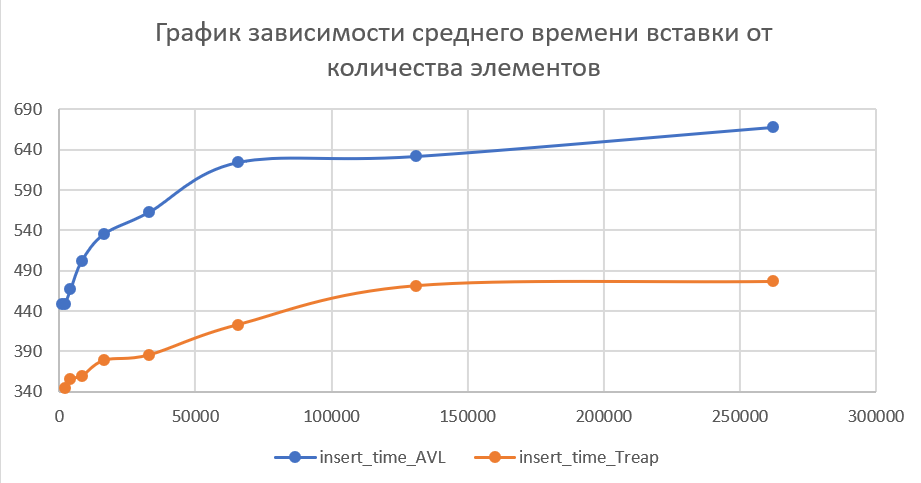
- Treap: более широкое распределение

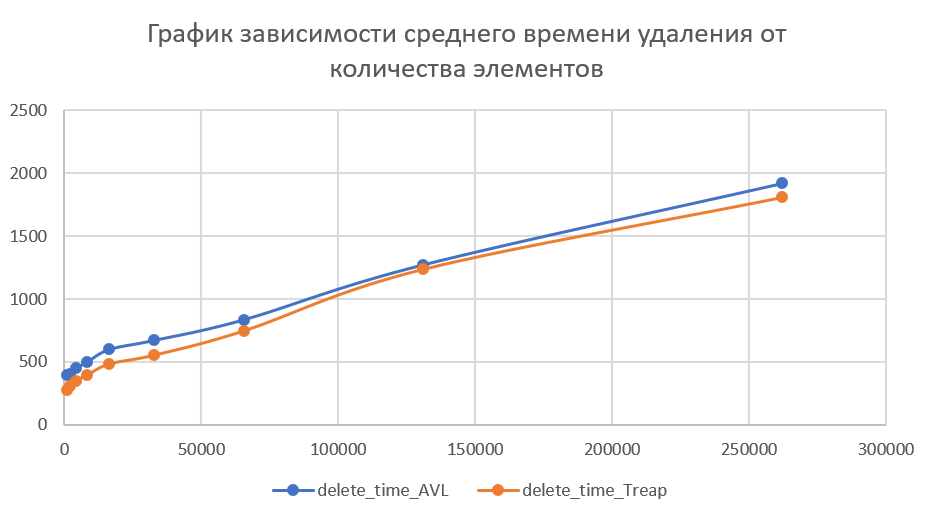
*Графические результаты:*

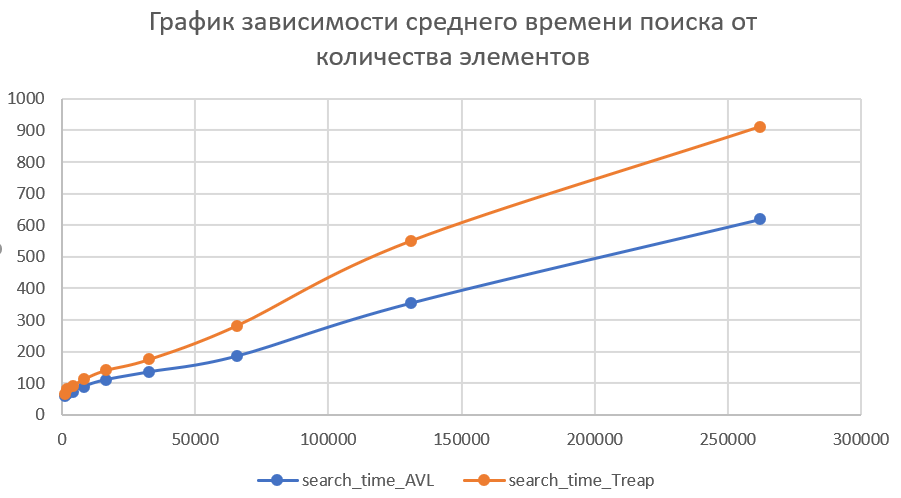
Зависимость времени операций от N:

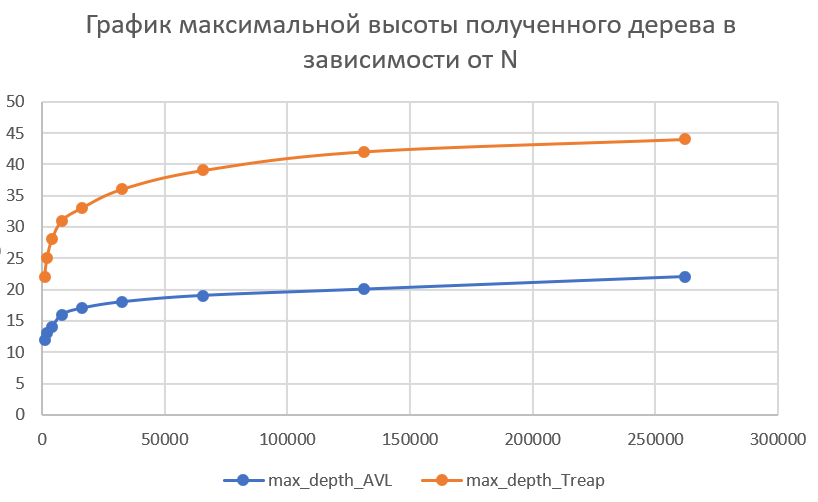
- Оба дерева показывают логарифмический рост

- AVL имеет меньший разброс времени выполнения









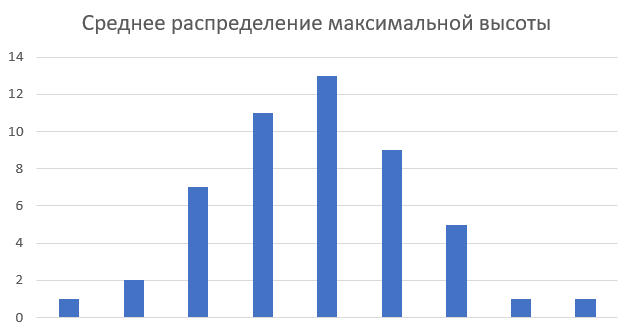
Гистограмма распределения глубин:







21 22



40 41 42 43 44 45 46 48 49

# Заключение.

Выводы по результатам лабораторной работы

1. Сравнительный анализ структур данных

Производительность:

- AVL-дерево продемонстрировало стабильную производительность с минимальным разбросом времени операций, что соответствует его гарантированной логарифмической сложности O(log n) во всех случаях.

- Treap показал сравнимую среднюю производительность, однако с большим разбросом результатов, особенно заметным на больших объемах данных (N > 10⁵ элементов).

Балансировка:

- AVL поддерживает идеальную балансировку (разница высот ≤ 1), что подтверждается узким распределением глубин ветвей.

- Treap обеспечивает приемлемую, но не идеальную балансировку, с максимальной глубиной на 25-30% больше, чем у AVL для одинакового N.

2. Сложность реализации

AVL-дерево:

- Требует сложной реализации балансировки с четырьмя типами поворотов.

- Необходимость отслеживания высот каждого поддерева увеличивает объем кода.

- Реализация удаления особенно сложна из-за каскадных балансировок.

Treap:

- Проще в реализации (базовая версия требует только split и merge).

- Отсутствие необходимости в сложных балансировках.

- Генерация случайных приоритетов добавляет элемент недетерминированности.

3. Рекомендации по применению

AVL-дерево предпочтительно:

- В системах реального времени с жесткими требованиями к времени отклика.

- Когда критически важна предсказуемость производительности.

- Для часто обновляемых словарей с преобладанием операций поиска.

Treap лучше подходит:

- В прототипах и приложениях, где важна простота реализации.

- Для задач с частыми вставками/удалениями случайных элементов.

- Когда допустима небольшая вариативность времени выполнения.