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 № 7

Выполнил студент группы КС-36 Акулинин Андрей Игоревич

Ссылка на репозиторий: <https://github.com/MUCTR-IKT-CPP/AkulininAI_36/tree/main/algorithms>

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 14.04.2025

Оглавление

[Описание задачи. 2](#_Toc63548272)

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

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы была поставлена цель — изучить, реализовать и сравнить два различных вида сбалансированных деревьев поиска: **Декартово дерево (Treap)** и **Красно-черное дерево (Red-Black Tree)**. В качестве эталонной реализации использовалось уже реализованное **AVL-дерево**.

Для сравнения структур необходимо было провести ряд тестов на производительность (время вставки, удаления и поиска), а также проанализировать их структуру (высоты, распределения веток). Особое внимание уделялось различиям поведения структур при работе с случайными и отсортированными входными данными.

# Описание метода/модели.

**AVL-дерево**

AVL — самобалансирующееся бинарное дерево поиска, которое поддерживает балансировку при каждой вставке или удалении узла. Разбалансировка фиксируется поворотами дерева. Гарантированная высота дерева: **O(log N)**.

* Вставка, удаление, поиск — O(log N)
* Память: O(N)
* Преимущества: строгая балансировка
* Недостатки: более частые повороты

**Декартово дерево (Treap)**

Treap объединяет свойства бинарного дерева поиска и кучи. Каждому ключу сопоставляется приоритет, и структура сохраняет свойства:

* По значению — как BST
* По приоритету — как куча (heap)
* Ожидаемая высота: O(log N) (при равномерных приоритетах)
* Вставка, удаление, поиск — O(log N) в среднем
* Преимущества: простая реализация, хорошее поведение на рандомных данных
* Недостатки: нет строгих гарантий в худшем случае

**Красно-черное дерево (RBT)**

RBT — бинарное дерево поиска с добавлением битов цвета для контроля баланса. Оно гарантирует **максимальную высоту в O(log N)**.

* Вставка, удаление, поиск — O(log N)
* Менее строгое сбалансированное дерево по сравнению с AVL, но требует меньше перестроений
* Плюсы: стабильное поведение, хорошо подходит для стандартных библиотек
* Минусы: сложнее реализации, не идеально сбалансировано

# Выполнение задачи.

**Язык и инструменты**

* Язык программирования: **C++17**
* Используемые библиотеки: <vector>, <chrono>, <fstream>, <random>, <cmath>, <algorithm>
* Визуализация графиков: **Python (Pandas + Matplotlib)**

**Организация программы**

* Для каждого дерева реализованы классы с методами вставки, удаления, поиска и анализа структуры
* Все данные логгируются в .csv файлы
* Для анализа реализованы Python-скрипты, автоматически строящие графики

**Проведенные тесты**

* Для N = 2^i, где i от 10 до 18
* 50 повторений на каждой размерности
* Генерация случайных и отсортированных входных данных
* Измерения:
  + Время вставки (1000 операций)
  + Время удаления (1000 операций)
  + Время поиска (1000 операций)
  + Максимальная высота дерева
  + Распределение высот веток

Изображение выглядит как текст, линия, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, линия, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

# Изображение выглядит как снимок экрана, линия, Параллельный, Красочность Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Красочность, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, График, линия, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, число, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, График, линия, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

# Заключение.

**Выводы:**

* **AVL-дерево** демонстрирует лучшую сбалансированность, но требует больше поворотов.
* **Treap** прост в реализации и показывает хорошие средние результаты, но чувствителен к случайности.
* **Красно-черное дерево** — лучший компромисс между простотой и стабильной производительностью.

**Впечатления:**

* Реализация структуры данных на C++ требует высокой точности в обработке указателей и рекурсивных операций.
* Было полезно сравнить подходы и увидеть, как влияет структура дерева на практическую производительность.

**Задания со звездочкой:**

* Проведены аналогичные тесты на отсортированных данных
* Реализовано и протестировано красно-черное дерево