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 № 8

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 требовалось:

1. Реализовать **бинарную кучу** (min-heap или max-heap).
2. Реализовать одну из альтернативных структур куч:
   * **Биномиальную кучу** (выбранный вариант).
3. Провести тестирование производительности:
   * Наполнить кучу **N** элементов (N = 10³, 10⁴, 10⁵, 10⁶, 10⁷).
   * После заполнения выполнить:
     + 1000 операций поиска минимума/максимума.
     + 1000 операций удаления минимума/максимума.
     + 1000 операций вставки нового элемента.
   * Замерить **среднее** и **максимальное** время выполнения операций.
4. Построить графики на основе полученных данных.

# Описание метода/модели.

**Бинарная куча**

* **Структура:** Полное бинарное дерево, где каждый родитель меньше (min-heap) или больше (max-heap) своих потомков.
* **Реализация:** Обычно на основе массива.
* **Сложность операций:**
  + Вставка (insert): **O(log n)**
  + Поиск минимума (getMin): **O(1)**
  + Удаление минимума (extractMin): **O(log n)**
* **Преимущества:**
  + Простота реализации.
  + Эффективность для приоритетных очередей.
* **Недостатки:**
  + Медленное слияние (**O(n)**).

**Биномиальная куча**

* **Структура:** Набор биномиальных деревьев разного порядка.
* **Сложность операций:**
  + Вставка (insert): **O(1)** (амортизированная)
  + Поиск минимума (getMin): **O(log n)**
  + Удаление минимума (extractMin): **O(log n)**
  + Слияние (merge): **O(log n)**
* **Преимущества:**
  + Эффективное слияние (**O(log n)**).
  + Амортизированно быстрая вставка.
* **Недостатки:**
  + Более сложная реализация.
  + Хуже кэш-локальность из-за указателей.

# Выполнение задачи.

**Реализация**

* **Язык программирования:** C++ (тесты), Python (графики).
* **Структуры данных:**
  + BinaryHeap (на массиве).
  + BinomialHeap (на основе связанных деревьев).
* **Тестирование:**
  + Замер времени выполнения каждой операции **индивидуально** (без пакетной обработки).
  + Расчет **среднего** и **максимального** времени.

**Графики**

1. **Среднее время операций**
   * Бинарная куча быстрее в **поиске** и **удалении**.
   * Биномиальная куча имеет **амортизированно быструю вставку**.
2. **Максимальное время операций**
   * У биномиальной кучи наблюдаются **пиковые задержки** из-за перестроения структуры.

Изображение выглядит как текст, диаграмма, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

# Заключение.

**Выводы**

1. **Бинарная куча** показала **лучшую производительность** в синхронных операциях (поиск, удаление).
2. **Биномиальная куча** эффективна при **частых вставках** и **слияниях**, но имеет **нестабильное время выполнения**.
3. **Практическая применимость:**
   * **Бинарная куча** — лучший выбор для задач с частыми запросами минимума.
   * **Биномиальная куча** — полезна в алгоритмах, требующих слияния куч (например, алгоритм Дейкстры с уменьшением ключа).

**Рекомендации**

* Для **однопоточных** задач с приоритетной очередью — **бинарная куча**.
* Для **многопоточных** или **динамически изменяемых** структур — **биномиальная/фибоначчиева куча**.