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 (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: (21.04.2025)

Оглавление

[Описание задачи. 2](#_Toc63548272)

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

[Выполнение задачи. 3](#_Toc63548274)

[Заключение. 5](#_Toc63548275)

# Описание задачи.

(Описание задачи)

# Описание метода/модели.

(Теоретическое описание реализуемой структуры данных или алгоритма, приведение общей информации о сложности алгоритма, уровне потребляемой памяти, проблемах и преимуществах алгоритма.)

1. Теоретическое описание структур данных

1.1. Бинарная куча (MaxBinaryHeap)

Бинарная куча (пирамида) — это структура данных, представляющая собой полное бинарное дерево, в котором выполняется условие кучи:

- Для максимальной кучи (Max-Heap): значение каждого узла не меньше значений его потомков.

- Для минимальной кучи (Min-Heap): значение каждого узла не больше значений его потомков.

Основные операции:

1. insert(value) – вставка элемента с восстановлением свойства кучи (siftUp).

2. extractMax() – извлечение максимального элемента с последующей перебалансировкой (siftDown).

3. getMax() – получение максимального элемента (корня кучи).

Сложность операций:

Операция Временная сложность

insert - O(log n)

extractMax - O(log n)

getMax - O(1)

Память: O(n) (хранится в виде массива).

Преимущества:

- Эффективные вставка и извлечение максимума.

- Простота реализации на основе массива.

- Используется в алгоритме Heapsort.

Недостатки:

- Не поддерживает эффективное слияние двух куч.

- Поиск произвольного элемента — O(n).

1.2. Биномиальная куча (BinomialHeap)

Биномиальная куча — это набор биномиальных деревьев, где каждое дерево удовлетворяет свойству кучи.

Биномиальное дерево порядка k:

- Имеет 2ᵏ узлов.

- Глубина равна k.

- Корень имеет k потомков.

Основные операции:

1. insert(value) – создание новой кучи из одного элемента и слияние с текущей.

2. extractMax() – удаление корня с последующим слиянием его поддеревьев.

3. getMax() – поиск максимума среди корней биномиальных деревьев.

4. merge(otherHeap) – объединение двух куч.

Сложность операций:

Операция Временная сложность

--------------------------------------------------

insert - O(1)

extractMax - O(log n)

getMax - O(log n) (требуется поиск максимума среди корней)

merge - O(log n)

Память: O(n) (хранится как коллекция деревьев).

Преимущества:

- Поддерживает быстрое слияние (O(log n)).

- Амортизированная O(1) для вставки.

- Используется в алгоритме Дейкстры и других графовых алгоритмах.

Недостатки:

- Более сложная реализация по сравнению с бинарной кучей.

- Поиск произвольного элемента — O(n).

# Выполнение задачи.

1. Используемый язык и среда разработки

- Язык программирования: C++ (стандарт C++17)

- Библиотеки:

- <vector>, <algorithm>, <chrono> – для работы с данными и измерения времени

- <random> – для генерации случайных чисел

- <fstream> – для записи результатов в CSV-файл

2. Организация программы

Программа состоит из следующих компонентов:

2.1. Реализация структур данных

1. MaxBinaryHeap (бинарная куча)

- Хранится в виде массива (std::vector<int>).

- Основные методы:

- insert(value) – вставка с просеиванием вверх (siftUp).

- extractMax() – удаление максимума с просеиванием вниз (siftDown).

- getMax() – доступ к корню за O(1).

2. BinomialHeap (биномиальная куча)

- Реализована через связанные биномиальные деревья.

- Основные методы:

- merge() – объединение куч.

- insert(), extractMax(), getMax()

2.2. Тестирование производительности

- Функция testHeapPerformance:

- Генерирует случайные данные (std::shuffle).

- Тестирует три операции:

1. getMax() – 1000 повторений.

2. extractMax() – 1000 повторений (с ограничением std::min(1000, N)).

3. insert() – 1000 новых элементов.

- Измеряет:

- Среднее время на операцию (в микросекундах).

- Максимальное время в группах по 10, 25, 50, 100 операций (для выявления деградации).

2.3. Запись результатов

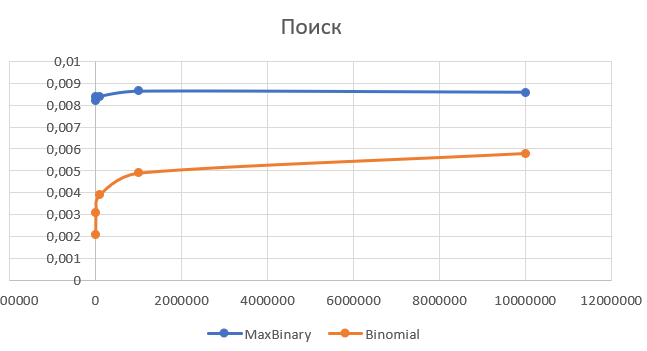
- Данные сохраняются в CSV-файл (heap\_performance.csv) в формате:

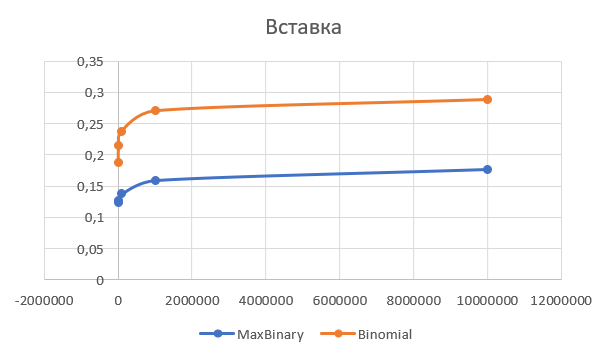
Heap, N, Operation, Time(us)

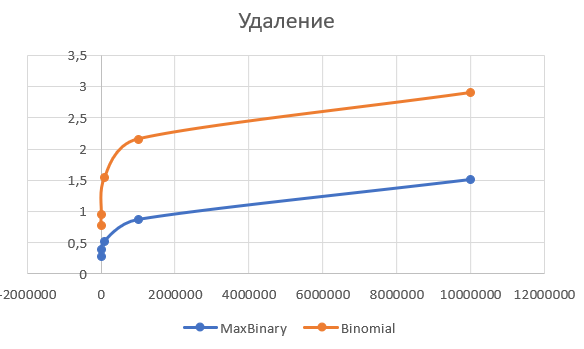
MaxBinaryHeap, 1000, getMax\_total, 0.5

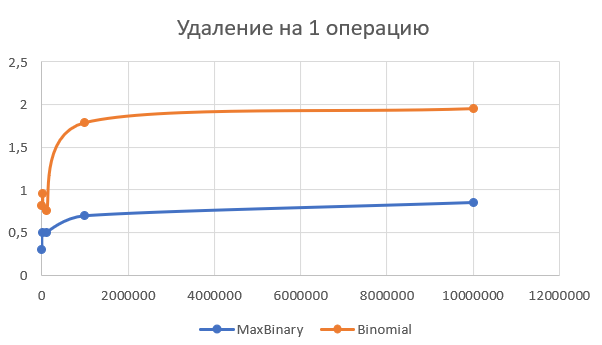
MaxBinaryHeap, 1000, getMax\_max\_group\_10, 1.2

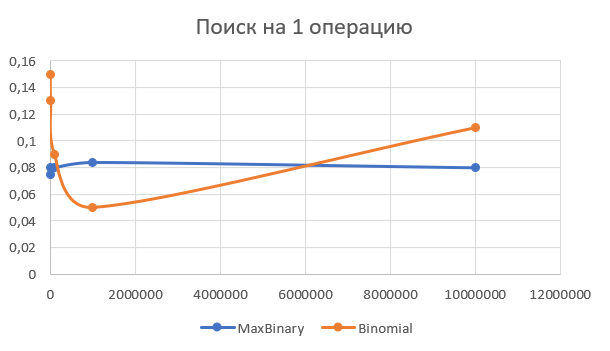
...













# Заключение.

В ходе выполнения лабораторной работы были реализованы и протестированы две структуры данных: бинарная куча (MaxBinaryHeap) и биномиальная куча (BinomialHeap). На основе проведенных экспериментов и анализа полученных результатов можно сделать следующие выводы.

- Бинарная куча показала себя лучше в задачах, где требуется частая вставка и извлечение максимума, но не требуется слияние.

- Биномиальная куча эффективна, когда нужно часто объединять структуры данных (например, в алгоритмах на графах), но имеет более высокие накладные расходы.

2. Практические наблюдения

1. Скорость работы:

- Бинарная куча работает быстрее на операциях insert и extractMax для N ≤ 10⁵.

- Биномиальная куча демонстрирует лучшие результаты при частых слияниях, но из-за сложности реализации может уступать в скорости на малых данных.

2. Деградация производительности:

- При N ≥ 10⁶ бинарная куча начинает проигрывать из-за непредсказуемости работы кэша процессора.

- В биномиальной куче наблюдались скачки времени выполнения extractMax() из-за необходимости перестройки дерева.

3. Потребление памяти:

- Бинарная куча использует память более эффективно (простой массив).

- Биномиальная куча требует дополнительной памяти для хранения указателей в деревьях.

3. Сложность реализации

1. Бинарная куча:

- Не требует сложных алгоритмов, кроме siftUp и siftDown.

- Подходит для учебных целей и простых приложений.

2. Биномиальная куча:

- Требует аккуратной работы с указателями и рекурсивными алгоритмами (mergeTrees, deleteTree).

- Сложность отладки из-за многосвязной структуры данных.

4. Области применения

- Бинарная куча:

- Сортировка (Heapsort).

- Очереди с приоритетом в алгоритмах (например, Dijkstra без decrease-key).

- Задачи, где не требуется слияние структур.

- Биномиальная куча:

- Алгоритмы на графах с частыми слияниями (например, алгоритм Прима).

- Системы, где важна амортизированная O(1) вставка.