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 № 2

Выполнил студент группы КС-36 Меркушов Даниил Игоревич

Ссылка на репозиторий: https://github.com/VooDooCry/Algorithms\_KS36.git

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 17.02.2025

Оглавление

[Описание задачи. 2](#_Toc63548272)

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

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

Необходимо реализовать алгоритм пирамидальной сортировки, который упорядочивает элементы массива целых чисел в порядке возрастания.

# Описание метода/модели.

# Пирамидальная сортировка (или сортировка кучей, HeapSort) — это метод сортировки сравнением, основанный на такой структуре данных как двоичная куча. Она похожа на сортировку выбором, где мы сначала ищем максимальный элемент и помещаем его в конец. Далее мы повторяем ту же операцию для оставшихся элементов.

# Выполнение задачи.

Язык программирования: **Golang**, **Python**

Входные данные / тесты: (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000)

1. Генерация массива:

Функция **generate\_array** создаёт массив заданного размера со случайными числами от -1 до 1.

1. Пирамидальная сортировка:

**heapify** преобразует поддерево в кучу, рекурсивно вызывая себя для дочерних узлов. Подсчитывает общее количество вызовов и внутренние рекурсивные вызовы.

**build\_heap** строит кучу из массива, вызывая **heapify** для каждого узла, начиная с середины.

**heap\_sort** выполняет сортировку, извлекая элементы из кучи и перестраивая её.

1. Тестирование:

Для каждого элемента массива из списка m\_values проводится 20 замеров.

Каждый замер включает генерацию нового массива, копирование его для сохранения исходных данных, выполнение сортировки с подсчётом времени и вызовов функций.

Результаты сохраняются в виде списка словарей, содержащих время выполнения и счётчики вызовов для каждой попытки.

1. Графики:

**analyze\_and\_plot** собирает результаты и выводит через matplotlib графики, заданные заданием.

# Заключение.

Основные выводы:

1. Пирамидальная сортировка является надежным выбором для задач, требующих стабильной производительности на больших объемах данных
2. Разница между средним, наилучшим и наихудшим временем выполнения не превышает 15-20%, что подтверждает предсказуемость алгоритма даже для больших данных.
3. График наихудшего времени выполнения демонстрирует рост, близкий к O (n log n).

**Графики и результаты**:









