Тема 06: Кучи (Heaps)

Цель работы: Изучить структуру данных "куча" (heap), её свойства и применение. Освоить основные операции с кучей (добавление, извлечение корня) и алгоритм её построения. Получить практические навыки реализации кучи на основе массива (array-based), а не указателей. Исследовать эффективность основных операций и применение кучи для сортировки и реализации приоритетной очереди.

Теория (кратко):

- **Куча (Heap):** Специализированная древовидная структура данных, удовлетворяющая свойству кучи. Является **полным бинарным деревом** (все уровни заполнены, кроме последнего, который заполняется слева направо).
- **Свойство кучи:**
  - **Min-Heap:** Значение в любом узле **меньше или равно** значениям его потомков. Корень — минимальный элемент.
  - **Max-Heap:** Значение в любом узле **больше или равно** значениям его потомков. Корень — максимальный элемент.
- **Реализация:** Куча эффективно реализуется на основе **массива**. Для узла с индексом `i`:
  - Индекс родителя: `(i-1)//2`
  - Индекс левого потомка: `2*i + 1`
  - Индекс правого потомка: `2*i + 2`
- **Основные операции:**
  - **Вставка (Insert):** Элемент добавляется в конец массива и "всплывает" (sift-up) до восстановления свойства кучи. Сложность: O(log n).
  - **Извлечение корня (Extract):** Корень (элемент `[0]`) извлекается, последний элемент ставится на его место и "погружается" (sift-down) до восстановления свойства кучи. Сложность: O(log n).
  - **Построение кучи (Heapify):** Преобразование произвольного массива в кучу. Может быть выполнено алгоритмом со сложностью O(n).
- **Применение:**
  - Сортировка кучей (Heapsort).
  - Реализация приоритетной очереди.
  - Алгоритм Дейкстры.

Практика (подробно):

Задание:

1. Реализовать структуру данных "куча" (min-heap и max-heap) на основе массива.
2. Реализовать основные операции и алгоритм построения кучи из массива.
3. Реализовать алгоритм сортировки кучей (Heapsort).
4. Провести анализ сложности операций.
5. Сравнить производительность сортировки кучей с другими алгоритмами.

Шаги выполнения:

1. **Создание проекта:** Создать файлы `heap.py`, `heapsort.py`, `priority_queue.py`.
2. **Реализация кучи (в `heap.py`):**

- Реализовать класс `MinHeap` (или универсальный класс `Heap` с параметром `is_min`).
- Реализовать внутренние методы:
  - `_sift_up(index)`: Всплытие элемента.
  - `_sift_down(index)`: Погружение элемента.
- Реализовать основные методы:
  - `insert(value)`: Вставка элемента.
  - `extract()`: Извлечение корня.
  - `peek()`: Просмотр корня.
  - `build_heap(array)`: Построение кучи из произвольного массива.
- **После каждого метода указать его временную сложность.**

3. **Реализация Heapsort (в `heapsort.py`):**
   - Реализовать функцию `heapsort(array)`, использующую кучу для сортировки.
   - Реализовать in-place версию Heapsort, которая не использует дополнительную память под кучу, а преобразует исходный массив.
4. **Реализация приоритетной очереди (в `priority_queue.py`):**
   - На основе кучи реализовать класс `PriorityQueue` с методами `enqueue(item, priority)` и `dequeue()`.
5. **Тестирование:**
   - Написать unit-тесты для проверки корректности работы кучи, Heapsort и приоритетной очереди.
   - Проверить свойство кучи после операций вставки и извлечения.
6. **Экспериментальное исследование:**
   - Замерить время построения кучи разными методами (последовательная вставка vs алгоритм `build_heap`).
   - Сравнить время работы Heapsort с быстрой сортировкой и сортировкой слиянием на случайных данных.
   - **ВАЖНО: Все замеры проводить на одной вычислительной машине.**
7. **Визуализация:**
   - Реализовать вывод кучи в виде дерева (текстовый или графический) для небольших размеров.
   - Построить графики зависимости времени операций от количества элементов.
8. **Анализ результатов:**
   - Сравнить практическую и теоретическую сложность операций.
   - Объяснить разницу во времени между двумя методами построения кучи.
   - Проанализировать эффективность Heapsort.
9. **Оформление отчета:** Результаты оформить в файле `README.md`. Код должен соответствовать PEP8.
10. **Контроль версий:** Стратегия ветвления – GitHub Flow.

Критерии оценки:

- **Оценка «3» (удовлетворительно):**
  - Реализована куча (min-heap) с операциями `insert` и `extract`.
  - В коде присутствуют комментарии с оценкой сложности.
  - Проведены базовые замеры времени операций.
- **Оценка «4» (хорошо):**

- Выполнены все критерии на «3».
- Реализован метод `build_heap` для построения кучи из массива.
- Реализована сортировка Heapsort.
- Код хорошо отформатирован и полностью прокомментирован.
- Проведены замеры времени для разных методов построения кучи.
- Построены графики зависимости времени от количества элементов.

- **Оценка «5» (отлично):**

  - Выполнены все критерии на «4».
  - Приведены характеристики ПК для тестирования.
  - Реализована приоритетная очередь на основе кучи.
  - Реализована in-place версия Heapsort.
  - Проведен сравнительный анализ Heapsort и других алгоритмов сортировки (QuickSort, MergeSort).
  - В отчете присутствует детальный анализ с выводами о применении кучи и эффективности операций.

Рекомендованная литература

1. **Юрий Петров: "Программирование на Python"** — онлайн-курс и учебные материалы.

   - *Ссылка для изучения:* https://www.yuripetrov.ru/edu/python/index.html

2. **Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К.** Алгоритмы: построение и анализ, 3-е издание. — М.: Вильямс, 2022. — 1328 с.

   - *(Оригинальное название: Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. Introduction to Algorithms, 3rd Edition)*

3. **Скиена, С.** Алгоритмы. Руководство по разработке, 3-е издание. — СПб.: БХВ-Петербург, 2022. — 720 с.

   - *(Оригинальное название: Skiena, Steven S. The Algorithm Design Manual, 3rd ed.)*