Тема 01: Введение в алгоритмы. Сложность. Поиск.

Цель работы: Освоить понятие вычислительной сложности алгоритма. Получить практические навыки реализации и анализа линейного и бинарного поиска. Научиться экспериментально подтверждать теоретические оценки сложности O(n) и O(log n).

Теория (кратко):

- **Сложность алгоритма:** Характеризует количество ресурсов (времени и памяти), необходимых алгоритму для обработки входных данных объема *n*.
- **Асимптотический анализ:** Анализ поведения алгоритма при стремлении *n* к бесконечности. Позволяет абстрагироваться от констант и аппаратных особенностей.
- **O-нотация («O-большое»):** Верхняя асимптотическая оценка роста функции. Определяет наихудший сценарий работы алгоритма.
- **Линейный поиск (Linear Search):** Последовательный перебор всех элементов массива. Сложность: **O(n)**.
- **Бинарный поиск (Binary Search):** Поиск в отсортированном массиве путем многократного деления интервала поиска пополам. Сложность: **O(log n)**. Требует предварительной сортировки (O(n log n)).

Практика (подробно):

Задание:

1. Реализовать функцию линейного поиска элемента в массиве.
2. Реализовать функцию бинарного поиска элемента в отсортированном массиве.
3. Провести теоретический анализ сложности обоих алгоритмов.
4. Экспериментально сравнить время выполнения алгоритмов на массивах разного размера.
5. Визуализировать результаты, подтвердив асимптотику O(n) и O(log n).

Шаги выполнения:

1. **Создание проекта:** Создать файл `search_comparison.py`.
2. **Реализация алгоритмов:**
   - Реализовать функцию `linear_search(arr, target)`.
   - Реализовать функцию `binary_search(arr, target)`.
   - **После каждой строки кода в комментарии указать её асимптотическую сложность.**
   - **В конце каждой функции в комментарии указать общую сложность алгоритма.**
3. **Подготовка данных:** Сгенерировать отсортированные массивы целых чисел разного размера (напр., `[1000, 2000, 5000, ..., 1000000]`). Для каждого размера выбрать целевой элемент (напр., первый, последний, средний, отсутствующий).
4. **Эмпирический анализ производительности:**
   - Написать функцию для замера среднего времени выполнения.
   - Для каждого размера массива и каждого алгоритма провести серию замеров времени поиска.
   - **ВАЖНО:** Все замеры проводить на одной и той же вычислительной машине. Убедиться, что массив для бинарного поиска отсортирован.

5. **Визуализация:**
    - Построить на одном графике зависимости времени выполнения от размера массива для обоих алгоритмов.
    - Построить второй график в логарифмическом масштабе по оси *y* (для наглядного отображения O(log n)) или по обеим осям (log-log scale).
6. **Анализ результатов:** Сравнить теоретические предсказания с практическими результатами. Объяснить расхождения, если они есть.
7. **Оформление отчета:** Результаты оформить в файле `README.md` в корне директории с кодом. Отчет должен содержать цель, теорию, графики, анализ и выводы. Код должен соответствовать PEP8 и требованиям из `00_lab00-Требования к коду.pdf`.
8. **Контроль версий:** Стратегия ветвления - GitHub Flow.

Критерии оценки:

- **Оценка «3» (удовлетворительно):**
    - Реализованы и корректно работают обе функции поиска.
    - В коде присутствуют комментарии с оценкой сложности для ключевых операций.
    - Проведены базовые замеры времени для 2-3 размеров массивов.
- **Оценка «4» (хорошо):**
    - Выполнены все критерии на «3».
    - Код хорошо отформатирован, читаем и полностью прокомментирован.
    - Замеры времени проведены для 5+ размеров массивов с усреднением результатов.
    - Построен график в линейном масштабе, наглядно демонстрирующий разницу в росте времени выполнения.
- **Оценка «5» (отлично):**
    - Выполнены все критерии на «4».
    - Приведены характеристики ПК для тестирования.
    - Построен график в логарифмическом масштабе, убедительно подтверждающий логарифмический рост времени выполнения бинарного поиска.
    - В отчете присутствует детальный анализ результатов, включая сравнение теоретической и практической сложности, объяснение выбора тестовых данных (поиск разных элементов).
    - Программа структурирована, содержит функции для генерации данных, замера времени и построения графиков.

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.)*