Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
УПРАВЛЕНИЯ» Институт заочного образования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Выполнил: Васюков Владислав Станиславович

Общие замечания:

- **Больше конкретики:** Текст часто очень общий. Добавьте конкретики, чтобы сделать его более убедительным и полезным. Например, приведите примеры входных и выходных данных, укажите, как именно выбирается опорный элемент, и более подробно опишите результаты тестирования.
- **Последовательность:** Соблюдайте последовательность в терминологии. Иногда используется термин “массив”, иногда “список”. Выберите один и используйте его. Я буду использовать “список” в своих предложениях, так как это более естественно для Python.
- **Структура:** Подумайте о том, чтобы немного изменить структуру, чтобы сделать ее более логичной. Например, объедините описание алгоритма с реализацией программы.
- **Орфография и грамматика:** Внимательно проверьте текст на опечатки и грамматические ошибки.

По разделам:

1. Введение

- **Цель работы:** Отлично!
- **Задачи работы:** Отлично!

2. Описание алгоритма

- **“Наиболее эффективных методов сортировки”**: Это заявление требует оговорки. QuickSort эффективен *в среднем*, но в худшем случае может быть O(n^2). Важно это отметить. Кроме того, сравните его с другими алгоритмами, такими как Merge Sort, который имеет гарантированную сложность O(n log n).
- **“Выбирается элемент массива, называемый опорным элементом (pivot).”**: Укажите *как* выбирается опорный элемент. Обычно это первый элемент, последний элемент, случайный элемент или медиана из трех. Выбор влияет на производительность.
- **“Массив разбивается на две части: элементы, меньшие опорного, и элементы, большие или равные ему.”**: Это хорошее описание.
- **“Процесс повторяется рекурсивно”**: Отлично.
- **Добавьте пример:** Добавьте короткий пример, чтобы проиллюстрировать алгоритм. Например:

  “Предположим, у нас есть список [5, 2, 8, 1, 9, 4]. Если мы выберем 5 в качестве опорного элемента, разделение приведет к [2, 1, 4] и [8, 9]. Затем мы рекурсивно сортируем каждый из этих подсписков.”

3. Реализация программы

- **“Программа написана на языке Python и состоит из двух основных функций”**: Хорошо.
- **Больше деталей:** Добавьте больше деталей о том, *что* делают функции. Например:

  “Функция partition принимает список и опорный элемент и переставляет элементы списка так, чтобы все элементы, меньшие опорного элемента, находились слева от него, а все элементы, большие или равные ему, - справа. Она возвращает индекс опорного элемента после разделения.”

  “Функция quicksort принимает список и рекурсивно сортирует его, вызывая функцию partition для разделения списка и затем рекурсивно вызывая себя для сортировки подсписков.”

4. Входные и выходные данные

- **“Список чисел, введенных пользователем через консоль”**: Хорошо.
- **Обработка ошибок:** Упомяните обработку ошибок. Что происходит, если пользователь вводит нечисловые данные? Будет ли программа корректно обрабатывать это?
- **Пример:** Дайте конкретный пример.

  “Например, если пользователь введет ‘3 1 4 1 5’, программа выведет ‘[1, 1, 3, 4, 5]’.”

5. Инструкции по использованию

- **Точность:** Убедитесь, что имя файла программы правильное (quicksort.py).
- **Требования:** Укажите, требуются ли какие-либо зависимости (например, конкретная версия Python).

6. Результаты тестирования

- **“Программа была протестирована на нескольких наборах данных различной длины”**: Это слишком расплывчато. Укажите конкретные примеры тестовых данных. Например:

  *“Программа была протестирована на следующих наборах данных: * Пустой список: []. * Уже отсортированный список: [1, 2, 3, 4, 5]. * Список в обратном порядке: [5, 4, 3, 2, 1]. * Список со случайными числами: [3, 1, 4, 1, 5, 9, 2, 6].”*

- **“Все тесты прошли успешно”**: Что это значит? Какие критерии использовались для определения успеха? Укажите, что программа возвращала корректно отсортированный список для каждого тестового случая.

- **Производительность:** Включите информацию о производительности (время выполнения) для разных размеров входных данных. Даже просто укажите, что тестирование показало, что алгоритм работает достаточно быстро для разумных размеров списков.

7. Заключение

- **“Программа соответствует всем требованиям”**: Какие это требования? Четко перечислите их.
- **Будущая работа:** Укажите возможные направления для дальнейшего развития. Например, можно добавить различные стратегии выбора опорного элемента, чтобы улучшить производительность в худшем случае. Или, можно реализовать гибридный алгоритм, который использует QuickSort для больших списков и Insertion Sort для небольших подсписков (так как Insertion Sort часто быстрее для небольших списков).