Python ООП

Теория

Прочитать про алгоритм быстрой сортировки quicksort и сортировки слиянием mergesort (НЕ смотреть реализацию на Python)

Практика

Компании FICSIT, записывающей кодированную информацию в виде последовательности чисел, приходящих по внешнему конвееру, необходимы приложения, которые отсортируют все входящие числа на выходной конвеер. Проблема в том, что памяти в компьютере намного меньше, чем всего чисел в конвеере, поэтому загрузить все числа с конвеера в аппарат не получится. Надо решить проблему (не спрашивайте зачем, все равно не поймете)

FICSIT имеет две аппаратные машины, которые нужно программировать:

- Сортировщик блоков (FicsitBlockSorter)
- Соединитель сортированных блоков (FicsitBlockMerger)

Получив на вход бинарный файл с рандомными числами на Сортировщике блоков, нужно получить на выходе Соединителя один бинарный файл со всеми отсортированными числами с входного файла.

Описание решения

Входной конвеер можно представить в виде файла. Входной файл ДОЛЖЕН быть бинарным (FICSIT экономит место на конвеере) Что значит памяти меньше чем чисел в конвеере? Это значит, что мы можем прочитать из конвеера только процию чисел, поработать с ней, сохранить где-то и уже потом читать следующую порцию. Файл содержит от 0 до 8_000_000 чисел от -100_000 до 100_000, записанные в бинарном виде. FICSIT предоставляет скрипты для генерации файла и их перевода в текстовый формат. Количество чисел, которое влезет в память, равно 8000 Работа программ:

1. FicsitBlockSorter берет блок данных, сортирует его собственной сортировкой и записывает результат в бинарный файл по типу sorted_block_0.dat, после чего приступает к следующему блоку, пока не закончится входной файл
2. FicsitBlockMerger ждет от FicsitBlockSorter отсортированные файлики и соединяет их собственной сортировкой слиянием. Важно, что у него почти нет памяти, поэтому ему нельзя загружать из файла в память больше 400 чисел

Примечания

Важные пункты

- FicsitBlockSorter и FicsitBlockMerger это два разных приложения, и запущены они могут быть одновременно: python FicsitBlockSorter.py; python FicsitBlockMerger.py (в терминале VS Code)
- Выходной файл должен быть бинарным, соответственно его размер равен входному
- Если запускать сначала FicsitBlockSorter, а потом FicsitBlockMerger, то результат должен быть такой же, как и при одновременном запуске

- Повторяем, что функции сортировки должны быть свои, нельзя использовать sort() или что-то подобное
- FicsitBlockSorter и FicsitBlockMerger должны быть классами. Они должны иметь метод Sort(list) и НЕ должны заниматься чтением и записью бинарного файла (для этого можно написать свои классы, например PipelineIO, который будет иметь методы ReadBlock(), WriteBlock())