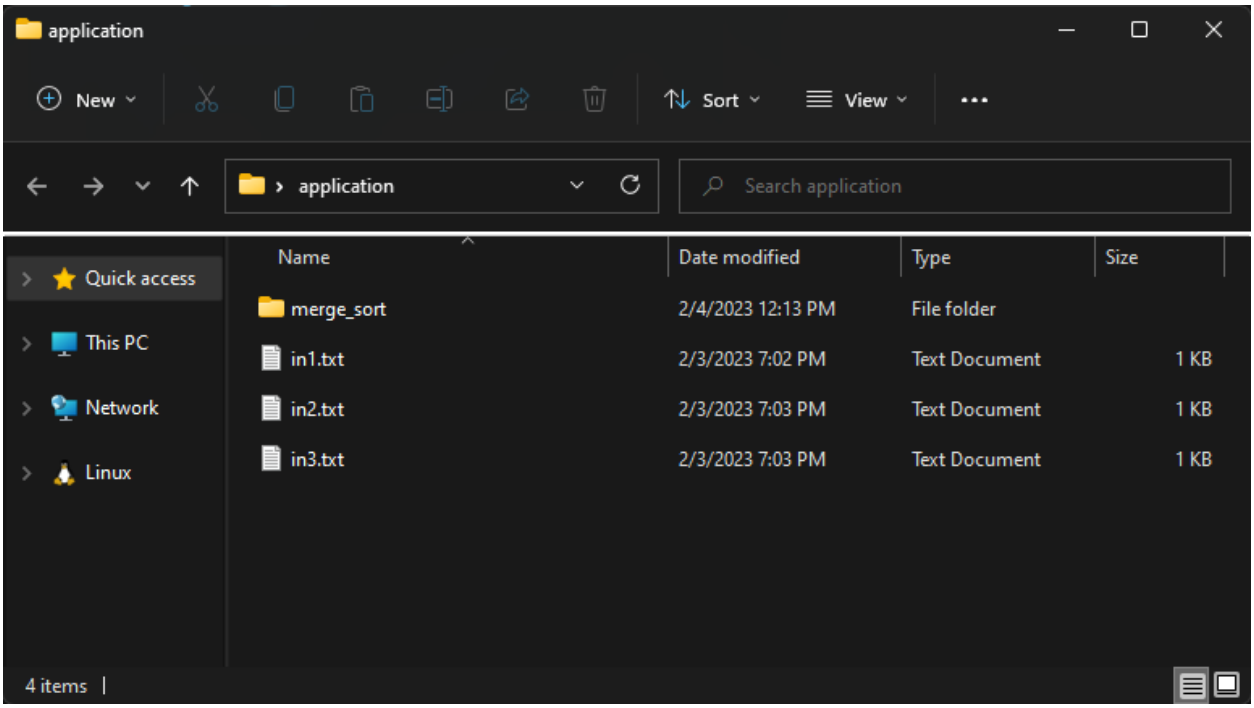
Инструкция по запуску

Основное:

Для компиляции я использовал `openjdk-19`. В проекте не использовались система сборки и сторонние библиотеки.

Проект находится в пакете `merge_sort`, для запуска рекомендую создать директорию и переместить в него пакет и входные файлы, например так:



`main` находится в `merge_sort.App`

Компиляция:

Unix: `javac merge_sort/App.java`

Windows: `javac .\merge_sort\App.java`

Запуск:

Unix: `java merge_sort.App -a -i out.txt in1.txt in2.txt in3.txt`

Windows: `java .\merge_sort.App -a -i out.txt in1.txt in2.txt in3.txt`

Пример компиляции и запуска программы:

```
shift@cft MINGW64 ~/OneDrive/Desktop/application
$ javac merge_sort/App.java

shift@cft MINGW64 ~/OneDrive/Desktop/application
$ cat in*.txt
1
4
9
1
8
27
1
2
3
shift@cft MINGW64 ~/OneDrive/Desktop/application
$ java merge_sort.App -a -i out.txt in1.txt in2.txt in3.txt
Buffer size: 87362062 bytes.
Success! See result in: "out.txt"

shift@cft MINGW64 ~/OneDrive/Desktop/application
$ cat out.txt
1
1
1
2
3
4
8
9
27

shift@cft MINGW64 ~/OneDrive/Desktop/application
$
```

Важно:

Программе запрещено перезаписывать файлы, поэтому перед запуском необходимо вручную удалить файл с именем выходного файла. (В примере out.txt)

// Комментарий:

Флаги в аргументах программы можно объединять (например -ai), причем оба флага являются опциональными (по умолчанию используются ascending и integer).

`App::main()` есть переменнная `IGNORE_ERRORS`, которая позволяет игнорировать ошибки и пропускать некорректные данные (по умолчанию true).

Валидация данных в программе принудительная, поэтому проверка типа и сортировка входных файлов происходит всегда.

Входные файлы считываются в буфер (некорректные отбрасываются), буфер сортируется и записывается во временный файл “.temp*” (если при попытке удаления временных файлов произошла ошибка, то программа сообщит какие файлы удалить не удалось) .

После валидации всех данных программа использует ту же идею, что и при слиянии в алгоритме сортировки, но на вход уже подаются отсортированные временные файлы.

Сортировка просиходит на 1 потоке.

При тесте на входном файле в 11 Гб целых чисел профайлер показал, что сортировка занимет только 57% времени:

Сам тест с профайлером занял почти 14 минут. (на i7-12650H с 16GB озу, у jvm стандартные настройки)

Тест с файлом 1.1 Gb длится около 75 секунд (тоже с профайлером).

Со строками длительность теста немного возрастает.

Выделение памяти для внутреннего буфера происходит по данным из Runtime, поэтому, если при выполнении возникнет `OutOfMemoryError`, то в `merge_sort.config.RuntimeHelper::freeMemory()`

нужно поменять значение `return` на константу (в файле поменять комментарии местами).