Отчет по курсовой работе.
Алгоритм внешней сортировки.

Задача:

Есть некий файл с данными, размер которых превышает объем оперативной памяти. Необходимо отсортировать строки в этом файле.

Алгоритм:

1. Считываем данные с файла в память, пока хватает памяти на хранение и сортировку этих данных.
2. Сортируем часть данных и записываем их в новый файл.
3. Повторяем шаги 1-2 пока не все данные считаны из исходного файла
4. Считываем из каждого файла строку, выбираем из всех строк минимальную и записываем в результирующий файл.
5. Для файла, строка которого оказалась наименьшей, считываем следующую, если такая есть.
6. Повторяем шаги 4-5 пока есть строки, не записанные в ответ

Шаг 4 можно ускорить, если для поиска минимума использовать структуру данных - приоритетную очередь.

Шаг 5 также можно ускорить, если для каждого файла хранить небольшой буфер с данными, чтобы не считывать новую строку каждый раз.

Тестирование:

В качестве тестового примера был сгенерирован файл с 100 тысячами случайных строк, длинами от 200 до 500 символов. Размер памяти для хранения данных для простоты задавался количеством строк, которое можно одновременно держать в памяти. Результаты приведены в таблице 1

В таблице записаны данные, усредненные за 3 запуска программы. Время измерялось простой разницей времени перед запуском и после, так как приложение работает в одном потоке. Память измерялась с помощью утилит для слежения использования памяти процессами.

Из таблицы видно что при маленьком количество памяти для хранения строк в памяти, появляются большие накладные расходы на хранения большого количество одновременно открытых файлов. Этот расход можно уменьшить, если проводить процедуру слияния не сразу всех файлов, а постепенно, например за раз объединять не более 100 файлов.

	Размер памяти	Время сортировки, с.	Потребляемая память, MB
in-memory sort	-	1.3 ± 0.1	46
external sort	100000	5 ± 0.1	48
external sort	10000	3.5 ± 0.1	15
external sort	1000	4 ± 0.1	12

Таблица 1. Сравнение алгоритмов