Кейс 7. Задание по работе с файлами

На диске лежит большой текстовый файл объёмом 100 Гб, который содержит длинную строку чисел. Нужно отсортировать числа по возрастанию. Опишите алгоритм решения такой задачи.

## Разбор кейса

Сложность выполнения задания в том, что файл размером 100 Гб нельзя прочитать в коллекцию или список и отсортировать, потому что обычно объём оперативной памяти компьютеров и даже серверов значительно меньше 100 Гб.

Оптимальным решением будет следующий алгоритм:

1. Начать читать числа из файла и складывать их в массив, пока он не достигнет размера, близкого к максимально возможному: доступному для виртуальной машины Java или равному величине оперативной памяти.
2. Отсортировать массив (например, методом sort класса Arrays) и записать в отдельный файл.
3. Повторить операции 1 и 2 с остальной частью исходного большого файла таким образом, чтобы в результате на диске были сохранены отдельные файлы с отсортированными в них числами.
4. Брать пары файлов и сливать их между собой, формируя файлы большего размера. Файлы при этом необходимо читать не целиком, а по одному числу и, сравнивая числа между собой, записывать отсортированную последовательность в итоговый файл так, как работает алгоритм сортировки слиянием — merge sort.

Пример реализации сортировки для двух массивов с числами:

| **int[] values1 = new int[200];**  **int[] values2 = new int[200];**  ***fillSortedArray*(values1);**  ***fillSortedArray*(values2);**  **int[] newValues = new int[values1.length + values2.length];**  **int index = 0;**  **int pointer1 = 0;**  **int pointer2 = 0;**  **while(true) {**  **int value1 = pointer1 < values1.length ?**  **values1[pointer1] : Integer.*MAX\_VALUE*;**  **int value2 = pointer2 < values2.length ?**  **values2[pointer2] : Integer.*MAX\_VALUE*;**  **if (value1 <= value2) {**  **newValues[index++] = value1;**  **pointer1++;**  **} else {**  **newValues[index++] = value2;**  **pointer2++;**  **}**  **if (index == newValues.length) {**  **break;**  **}**  **}** |
| --- |

В случае с файлами слияние может происходить так:

| **BufferedReader reader1 = Files.*newBufferedReader*(file1);**  **BufferedReader reader2 = Files.*newBufferedReader*(file2);**  **String resultFile = "merged.tmp";**  **FileWriter writer = new FileWriter(resultFile);**  **String line1 = reader1.readLine();**  **String line2 = reader2.readLine();**  **while(true) {**  **int value1 = line1 != null ?**  **Integer.*parseInt*(line1) : Integer.*MAX\_VALUE*;**  **int value2 = line2 != null ?**  **Integer.*parseInt*(line2) : Integer.*MAX\_VALUE*;**  **if (value1 <= value2) {**  **writer.write(Integer.*toString*(value1));**  **line1 = reader1.readLine();**  **} else {**  **writer.write(Integer.*toString*(value2));**  **line2 = reader2.readLine();**  **}**  **writer.write("\n");**  **if (line1 == null && line2 == null) {**  **break;**  **}**  **}**  **writer.flush();**  **writer.close();**  **reader1.close();**  **reader2.close();** |
| --- |

1. Чтобы программа работала быстрее, обрабатывать отдельные пары файлов лучше в многопоточном режиме, в каждом потоке по два файла.