External memory: k-way merge sort

Хованский Виктор Сергеевич, M4136c

Задача состоит в том, чтобы отсортировать большой файл, который не умещается целиком в оперативной памяти.

Условия

Пусть исходный файл *INPUT* занимает **T** байт на жестком диске.
Нужно его отсортировать, используя **M** байт в оперативной памяти.
Результат должен быть записан в файл *OUTPUT*.

Описание алгоритма

В первой фазе алгоритма будем считывать блоки размера **M** в оперативную память и сортировать их. Далее после сдвига указателя в файле на позицию **qM**, где **q** - номер блока. Можно записать отсортированный блок в файл.

Во второй фазе алгоритма заведем в памяти **k=[T/M]** ленивых узлов, каждый из которых будет хранить следующие поля:

- Локальный индекс в буфере *bufferIndex*;
- Доступная длина буфера *bufferAvailable*;
- Глобальный индекс в блоке *blockIndex*;
- Доступная длина блока *blockAvailable*;

Каждый узел представим в виде итератора. Тогда условием обновления данных в буфере будет:

```
if (bufferIndex == bufferAvailable && blockIndex < blockAvailable) { /**/ }
```

И при истинности данного выражения нужно загрузить в буфер данные с позиции

```
q * blockSize + blockIndex
```

blockAvailable для первых **k - 1** буферов будет равен **[M / k]**. Для **k** буфера его размер может быть меньше. Таким образом, для произведения одновременного слияния **k** ленивых узлов необходимо, чтобы *blockSize*(*blockSize* - 1) > **T**.

Данные в блоке были отсортированы в первой фазе алгоритма, поэтому на позиции *bufferIndex* будет всегда наименьший элемент (считаем, что сортировка по возрастанию) в блоке.

Далее заводим приоритетную очередь из узлов, где приоритетом будет выступать наименьший элемент из блока узла. Теперь же можно последовательно извлекать узлы-итераторы из очереди и записывать очередной элемент итератора в выходной файл *OUTPUT*. Если после записи очередного элемента в файл, итератор не пуст, то добавляем его обратно в очередь. Так продолжается до тех пор, пока очередь не станет пустой.

Тестирование

Для тестирования используется генерация случайного равновероятного набора элементов, которые записываются в массив.

Далее данный массив записывается в файл *INPUT* и сортируется в оперативной памяти для сравнения.

После этого сортируется файл *INPUT*, описанным выше алгоритмом, и проверяется корректность сортировки.

Результаты тестирования

Номер эксперимента	T	M	External	RAM
1	1000	32	64	0
2	1000	320	1	0
3	10000	101	50	0
4	10000	1010	5	0
5	100000	317	163	1
6	100000	3170	16	1
7	100000	31700	6	1
8	1000000	1001	1334	3
9	1000000	10010	131	3
10	1000000	100100	96	5
11	10000000	3163	15122	27
12	10000000	31630	1901	28
13	10000000	316300	1453	24
14	10000000	3163000	480	25

External – время выполнения алгоритма сортировки с использованием внешней памяти.
RAM – время выполнение алгоритма сортировки без использования внешней памяти (QuickSort).