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*. Разрешается создавать временные файлы.

Описание алгоритма

В первой фазе алгоритма будем считывать блоки размера **M** в оперативную память и сортировать их в ней. Отсортированный блок размера **M** записывается в файл *OUTPUT_i*, где **i** – номер итерации.

Во второй фазе алгоритма заводим приоритетную очередь из ленивых узлов, где приоритетом будет выступать наименьший элемент из узла. Каждый узел будет хранить следующие поля:

- Указатель на временный файл *OUTPUT_i*;
- Локальное смещение в буфере *bufferOffset*;
- Доступная длина буфера *bufferAvailable*;
- Глобальное смещение в файле *fileOffset*;
- Доступная длина файла *fileAvailable*;

Узлы представим в виде итераторов. Тогда условием обновления данных в буфере узла будет следующим:

```
if (bufferOffset == bufferAvailable && fileOffset < fileAvailable) { /**/ }
```

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

На очередной итерации фазы слияния в приоритетную очередь кладутся **k** узлов, которые соответствуют **k** наиболее старым файлам (с наименьшим номером **i**).

Теперь можно последовательно извлекать узлы-итераторы из очереди и записывать очередной элемент итератора в выходной файл *OUTPUT_(i+1)*.

Данные в каждом временном файле были отсортированы в первой фазе алгоритма или на предыдущих итерациях слияния файлов, поэтому на позиции *bufferOffset* будет всегда наименьший элемент (считаем, что сортировка по возрастанию).

Если после записи очередного элемента в файл, итератор не пуст, то добавляем его обратно в очередь. Иначе, узел можно удалить вместе с его временным файлом. Таким образом, суммарный размер временных файлов не будет превышать **T**. Цикл продолжается до тех пор, пока очередь не станет пустой. Далее итерация начинается заново. Последующие итерации продолжаются до тех пор, пока не останется только один ленивых-узел с одним временным файлом. Этот файл можно записать в *OUTPUT* как результат алгоритма.

Тестирование

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

Далее данный массив записывается в файл *INPUT* и после этого сортируется, описанным выше алгоритмом, и проверяется корректность сортировки.

Результаты тестирования

Номер эксперимента	T	M	k	External, ms
1	100000	317	2	208
2	100000	317	4	210
3	100000	317	8	91
4	100000	317	16	94
5	100000	317	32	118
6	100000	3170	2	23
7	100000	3170	4	32
8	100000	3170	8	21
9	100000	3170	16	25
10	100000	3170	32	27
11	100000	31700	2	7
12	100000	31700	4	7
13	100000	31700	8	8
14	100000	31700	16	7
15	100000	31700	32	8
16	1000000	1001	2	671
17	1000000	1001	4	553
18	1000000	1001	8	528
19	1000000	1001	16	520
20	1000000	1001	32	576
21	1000000	10010	2	328
22	1000000	10010	4	285
23	1000000	10010	8	293
24	1000000	10010	16	281
25	1000000	10010	32	253
26	1000000	100100	2	154
27	1000000	100100	4	141
28	1000000	100100	8	158
29	1000000	100100	16	119
30	1000000	100100	32	103
31	10000000	3163	2	5587
32	10000000	3163	4	5327
33	10000000	3163	8	4887
34	10000000	3163	16	4111
35	10000000	3163	32	4573
36	10000000	31630	2	3425
37	10000000	31630	4	3813
38	10000000	31630	8	3704
39	10000000	31630	16	3146

Номер эксперимента	T	M	k	External, ms
40	10000000	31630	32	2933
41	10000000	316300	2	2153
42	10000000	316300	4	2115
43	10000000	316300	8	1945
44	10000000	316300	16	1786
45	10000000	316300	32	1446
46	10000000	3163000	2	593
47	10000000	3163000	4	519
48	10000000	3163000	8	557
49	10000000	3163000	16	505
50	10000000	3163000	32	530

k – количество одновременно сливаемых файлов.
External – время выполнения алгоритма сортировки с использованием внешней памяти.