External Merge Sort with K-Buffering

In class, we have studied double buffering. This mechanism assigns tow pages for each participating run and two pages for the output buffer. We want to generalize this by assigning K pages per participating run and at least K pages for output buffer.

Consider a data file of size D pages.

Total available number of main memory pages are P.

Pass 1: At the end of this pass we will create total ceiling(D/P) runs. Each run will be of size P pages, except last that might have fewer or equal pages.

Pass 2: Out of total P pages, we have reserve at least K pages for the output buffer. Therefore we can merge floor((P-K)/K) runs simultaneously. If (P-K) is not divisible by K then remaining extra pages are allocated to output buffer. Therefore, the output buffer might get more than K pages. If R is the number of runs at the end of previous pass, the number of runs at the end of this pass will be

R' = ceiling (R*K/(P-K))

If S was the size of runs in the previous pass, then the size of runs after this pass will be

S' = S*(floor((P-K)/K))

Only the last run might have smaller or equal size.

Keep on doing more passes till you get a single run that is the sorted output.

Your code should read from standard input and write to standard output.

Submit single file <roll_number>_kmergesort.cpp

Consider the give input file.

Line 1 describes the page size in terms of number of entries per page. Value 3 indicates that 3 entries fit per page.

Line 2 described total number of available main memory pages. For this input, we have 13 pages of main memory available.

Line 3 describes the value of K for K-Buffering. Value 3 indicates, we have to allocate three pages per run while merging.

Line 4 describes total number of entries present in the data file. We have 1000 entries to sort.

We can compute following parameters

D= ceiling(number of entries/page size) = ceiling(1000/3) = 334 (First 333 pages will be completely filled. The last page will have only one entry.)

P=13

K=3

How many runs we can merge simultaneously? Floor((13-3)/3)

After Pass 1, we will have runs of the size 13 pages. Total number of runs: 26 (25 of size 13 pages and 1 of size 9 pages)

After Pass 2, we will have runs of the size 39 pages. Total number of runs: 9 (8 of size 39 and 1 of size 22 pages)

After pass 3, we will have runs of the size 117. Total number of runs: 3 (2 of size 117 and 1 of size 100 pages)

After pass 4, we will have a single run of size 1000 pages.

Lines 5 through 1004 describe each entry of the data file. First three entries 1000, 999, 998 will form the page 1 of the data file. Similarly, last entry 1 will for the page 334 of the data file. Now perform the external merge sort with K-buffering. Preserve the state of data file after each pass.

Line 1005 and 1006 ask you to print a particular page: Pass 1, Page 2. Your output should be

Pass 1, Page 2: 965, 966, 967

Lines 1007 and 1008 ask you to print a particular page: Pass 2, Page 335. Your output should be

Pass 2, Page 335: Invalid page number

Lines 1009 and 1010 ask you to print a particular page: Pass 2, Page 334

Pass 2, Page 334: 64

Lines 1011 and 1012 ask you to print a particular page: Pass 4, Page 1

Pass 4, Page 1: 1 2 3

Lines 1013 and 1014 ask you to print a particular page: Pass 5, Page 1

Pass 5, Page 1: Invalid pass number

Line 1015 have -1. This indicates termination of the program.