

## 1 General External Merge Sort

**Pass 0** – Use  $B$  buffer pages. Produce  $N/B$  sorted runs of  $B$  pages each.

**Further passes** – Merge  $B-1$  runs.

- (a) You have 4 buffer pages and your file has a total of 108 pages of records to sort. How many passes would it take to sort the file?

Pass 0 -  $\text{ceil}(108/4) = 27$  sorted runs of 4 pages each

Pass 1 -  $\text{ceil}(27/3) = 9$  sorted runs of 12 pages each

Pass 2 -  $\text{ceil}(9/3) = 3$  sorted runs of 36 pages each

Pass 3 - Sorted file (1 run)

Total = 4 passes

- (b) How many runs would each pass produce?

Pass 0 - 27 sorted runs (of 4 pages each)

Pass 1 - 9 sorted runs (of 12 pages each)

Pass 2 - 3 sorted runs (of 36 pages each)

Pass 3 - 1 sorted run (of 108 pages)

- (c) What is the total cost for this sort process in terms of I/O?

4 passes \* 2 (read + write per pass) \* 108 (pages in the file) = 864 I/Os

- (d) If the pages were already sorted individually, how many passes would it take to sort the file and how many IOs would it be instead?

Sorted pages does not change IO cost! Since Pass 0 is still going to produce  $\text{ceil}(N/B)$  sorted runs of  $B$  pages each, you would still require 4 passes and 864 IOs.

- (e) Given an arbitrary  $N$  number of buffer pages, if we could sort the file in 3 passes, what order of magnitude is the size of the file in terms of number of pages?

$N^3$

Every extra pass means an extra magnitude of order of the number of pages to sort.

## 2 Hashing

- (a) If we had records that we wanted to hash where the key being used were random integer values distributed uniformly, would the length of the integer be a good hash key? Why?

No. Integer length is heavily skewed - # of possible values for each key is not anywhere near evenly distributed.

- (b) What are some of the use-cases of hashing is preferred over sorting?

Removing duplicates, when partition phase can be omitted or shortened. Operations that require only data rendezvous (matching data must be together) and no order requirements - such as GROUP BY without ORDER BY.

- (c) We can process  $B \times (B-1)$  pages of data with external hashing in two passes. For this case, fill in the blanks with the appropriate number of pages, where we have B pages of available RAM (buffer pages).

  1   input buffer(s)

  B-1   partitions after pass 0

  B   pages per partition

- (d) If you are processing exactly  $B \times (B-1)$  pages of data with external hashing, is it likely that you'll have to perform recursive external hashing? Why or why not?

Yes. To avoid additional recursive external hashing, you would have to have an absolutely perfect hash function that evenly distributes records into the B - 1 partitions. This is almost impossible in practice. We should expect that some partitions may have more than B pages after partition hashing.

- (e) Can we use the same hash function for partitioning (divide) and rehashing (conquer)? How about for the recursive partitioning?

No. We use the hash function to create the partitions, so every record in a partition has the same hash value. Therefore, applying the hash function again in an attempt to rehash the data does nothing.