6

Lower Bounds on I/O

In this chapter, we prove the lower bounds from Theorems 5.1–5.4, including a careful derivation of the constants of proportionality in the permuting and sorting lower bounds. We also mention some related I/O lower bounds for the batched problems in computational geometry and graphs that we cover later in Chapters 8 and 9.

6.1 Permuting

The most trivial batched problem is that of scanning (a.k.a. streaming or touching) a file of N data items, which can be done in a linear number $O(N/DB) = O(n/D)$ of I/Os. Permuting is one of several simple problems that can be done in linear CPU time in the (internal memory) RAM model. But if we assume that the N items are indivisible and must be transferred as individual entities, permuting requires a nonlinear number of I/Os in PDM because of the locality constraints imposed by the block parameter B.

Our main result for parallel disk sorting is that we close the gap between the upper and lower bounds up to lower order terms. The lower bound from [23] left open the nature of the constant factor of

proportionality of the leading term; in particular, it was not clear what happens if the number of output steps and input steps differ.

Theorem 6.1 ([202]). Assuming that $m = M/B$ is an increasing function, the number of I/Os required to sort or permute n indivisible items, up to lower-order terms, is at least

$$\frac{2N}{D}\frac{\log n}{B\log m + 2\log N} \sim \begin{cases} \dfrac{2n}{D}\log_m n & \text{if } B\log m = \omega(\log N); \\ \dfrac{N}{D} & \text{if } B\log m = o(\log N). \end{cases} \tag{6.1}$$

The main case in Theorem 6.1 is the first one, and this theorem shows that the constant of proportionality in the $Sort(N)$ bound (5.1) of Theorem 5.1 is at least 2.

The second case in the theorem is the pathological case in which the block size B and internal memory size M are so small that the optimum way to permute the items is to move them one at a time in the naive manner, not making use of blocking.

We devote the rest of this section to a proof of Theorem 6.1. For the lower bound calculation, we can assume without loss of generality that there is only one disk, namely, $D = 1$. The I/O lower bound for general D follows by dividing the lower bound for one disk by D.

We call an input operation *simple* if each item that is transferred from the disk gets removed from the disk and deposited into an empty location in internal memory. Similarly, an output is *simple* if the transferred items are removed from internal memory and deposited into empty locations on disk.

Lemma 6.2 ([23]). For each computation that implements a permutation of the N items, there is a corresponding computation strategy involving only simple I/Os such that the total number of I/Os is no greater.

The lemma can be demonstrated easily by starting with a valid permutation computation and working backwards. At each I/O step,

in backwards order, we cancel the transfer of an item if its transfer is not needed for the final result; if it is needed, we make the transfer simple. The resulting I/O strategy has only simple I/Os.

For the lower bound, we use the basic approach of Aggarwal and Vitter [23] and bound the maximum number of permutations that can be produced by at most t I/Os. If we take the value of t for which the bound first reaches $N!$, we get a lower bound on the worst-case number of I/Os. In a similar way, we can get a lower bound on the average case by computing the value of t for which the bound first reaches $N!/2$.

In particular, we say that a permutation $\langle p_1, p_2, \ldots, p_N \rangle$ of the N items can be *produced after t_I input operations and t_O output operations* if there is some intermixed sequence of t_I input operations and t_O output operations so that the items end up in the permuted order $\langle p_1, p_2, \ldots, p_N \rangle$ in extended memory. (By extended memory we mean the memory locations of internal memory followed by the memory locations on disk, in sequential order.) The items do not have to be in contiguous positions in internal memory or on disk; there can be arbitrarily many empty locations between adjacent items.

As mentioned above, we can assume that I/Os are simple. Each I/O causes the transfer of exactly B items, although some of the items may be **nil**. In the PDM model, the I/Os obey block boundaries, in that all the non-**nil** items in a given I/O come from or go to the same block on disk.

Initially, before any I/Os are performed and the items reside on disk, the number of producible permutations is 1. Let us consider the effect of an output. There can be at most $N/B + o - 1$ nonempty blocks before the oth output operation, and thus the items in the oth output can go into one of $N/B + o$ places relative to the other blocks. Hence, the oth output boosts the number of producible permutations by a factor of at most $N/B + o$, which can be bounded trivially by

$$N(1 + \log N). \tag{6.2}$$

For the case of an input operation, we first consider an input I/O from a specific block on disk. If the b items involved in the input I/O were together in internal memory at some previous time (e.g., if the block was created by an earlier output operation), then the items could

have been arranged in an arbitrary order by the algorithm while they were in internal memory. Thus, the $b!$ possible orderings of the b input items relative to themselves could already have been produced before the input operation. This implies in a subtle way that rearranging the newly input items among the other $M - b$ items in internal memory can boost the number of producible permutations by a factor of at most $\binom{M}{b}$, which is the number of ways to intersperse b indistinguishable items within a group of size M.

The above analysis applies to input from a specific block. If the input was preceded by a total of o output operations, there are at most $N/B + o \leq N(1 + \log N)$ blocks to choose from for the I/O, so the number of producible permutations is boosted further by at most $N(1 + \log N)$. Therefore, assuming that at some prior time the b input items were together in internal memory, an input operation can boost the number of producible permutations by at most

$$N(1 + \log N)\binom{M}{b}. \tag{6.3}$$

Now let us consider an input operation in which some of the input items were not together previously in internal memory (e.g., the first time a block is input). By rearranging the relative order of the items in internal memory, we can increase the number of producible permutations by a factor of $B!$. Given that there are N/B full blocks initially, we get the maximum increase when all N/B blocks are input in full, which boosts the number of producible permutations by a factor of

$$(B!)^{N/B}. \tag{6.4}$$

Let I be the total number of input I/O operations. In the ith input operation, let b_i be the number of items brought into internal memory. By the simplicity property, some of the items in the block being accessed may not be brought into internal memory, but rather may be left on disk. In this case, b_i counts only the number of items that are removed from disk and put into internal memory. In particular, we have $0 \leq b_i \leq B$.

By the simplicity property, we need to make room in internal memory for the new items that arrive, and in the end all items are stored

back on disk. Therefore, we get the following lower bound on the number O of output operations:

$$O \geq \frac{1}{B}\Big(\sum_{1\leq i\leq I} b_i\Big). \tag{6.5}$$

Combining (6.2), (6.3), and (6.4), we find that

$$\big(N(1+\log N)\big)^{I+O} \prod_{1\leq i\leq I} \binom{M}{b_i} \geq \frac{N!}{(B!)^{N/B}}, \tag{6.6}$$

where O satisfies (6.5).

Let $\tilde{B} \leq B$ be the average number of items input during the I input operations. By a convexity argument, the left-hand side of (6.6) is maximized when each b_i has the same value, namely, $\tilde{B}$. We can rewrite (6.5) as $O \geq I\tilde{B}/B$, and thus we get $I \leq (I+O)/(1+\tilde{B}/B)$. Combining these facts with (6.6), we get

$$\big(N(1+\log N)\big)^{I+O} \binom{M}{\tilde{B}}^{I} \geq \frac{N!}{(B!)^{N/B}}; \tag{6.7}$$

$$\big(N(1+\log N)\big)^{I+O} \binom{M}{\tilde{B}}^{(I+O)/(1+\tilde{B}/B)} \geq \frac{N!}{(B!)^{N/B}}. \tag{6.8}$$

By assumption that M/B is an increasing function, the left-hand side of (6.8) is maximized when $\tilde{B} = B$, so we get

$$\big(N(1+\log N)\big)^{I+O} \binom{M}{B}^{(I+O)/2} \geq \frac{N!}{(B!)^{N/B}}. \tag{6.9}$$

The lower bound on $I+O$ for $D=1$ follows by taking logarithms of both sides of (6.9) and solving for $I+O$ using Stirling's formula. We get the general lower bound of Theorem 6.1 for D disks by dividing the result by D.

6.2 Lower Bounds for Sorting and Other Problems

Permuting is a special case of sorting, and hence, the permuting lower bound of Theorem 6.1 applies also to sorting. In the unlikely case that $B\log m = o(\log n)$, the permuting bound is only $\Omega(N/D)$, and we must