Finding Approximate Partitions and Splitters in External Memory*

Xiaocheng Hu† Yufei Tao† Yi Yang‡ Shuigeng Zhou‡

†Chinese University of Hong Kong
{xchu, taoyf}@cse.cuhk.edu.hk

‡Fudan University
{yyang1, sgzhou}@fudan.edu.cn

ABSTRACT

This paper studies two fundamental problems both of which are defined on a set $\mathcal{S}$ of elements drawn from an ordered domain. In the first problem—called *approximate K-partitioning*—we want to divide $\mathcal{S}$ into K disjoint partitions $P_1, ..., P_K$ such that (i) every element in P_i is smaller than all the elements in P_j for any i, j satisfying $1 \leq i < j \leq K$, and (ii) the size of each P_i ($1 \leq i \leq K$) falls in a given range $[a, b]$. In the second problem—called *approximate K-splitters*—we want to find $K-1$ elements $s_1, ..., s_{K-1}$ from $\mathcal{S}$, such that the size of $\mathcal{S} \cap (s_i, s_{i-1}]$ falls in a given range $[a, b]$ (define dummy $s_0 = -\infty$ and $s_K = \infty$).

We present I/O-efficient comparison-based algorithms for solving these problems, and establish their optimality by proving matching lower bounds. Our results reveal that the two problems are separated in terms of I/O complexity when K is small, but have the same hardness when K is large.

Categories and Subject Descriptors

F.2.2 [**Analysis of algorithms and problem complexity**]: Nonnumerical Algorithms and Problems

Keywords

Approximate partitioning, approximate splitters, external memory, lower bound

1. INTRODUCTION

Let $\mathcal{S}$ be a set of N elements drawn from an ordered domain. Given two elements e_1, e_2, we use the natural notations $e_1 < e_2$ and $e_1 > e_2$ to indicate that e_1 is before and after e_2 in their domain, respectively. Furthermore, in the former (latter) case, we say that e_1 is *smaller* (*larger*) than e_2. Also, by $[e_1, e_2]$, we refer to the set of elements e in the underlying domain such that $e_1 \leq e \leq e_2$. Notations $(e_1, e_2]$ and $[e_1, e_2)$ are defined analogously (i.e., exclusive at e_1 and e_2, respectively). Denote by $-\infty$ and ∞ two special elements that are smaller and larger than all the elements in $\mathcal{S}$, respectively.

We study two problems defined on $\mathcal{S}$. In the first problem—called *approximate K-partitioning*—we are given (i) an integer K of which N is a multiple and (ii) an integer interval $[a, b]$, and want to divide $\mathcal{S}$ into K disjoint partitions $P_1, ..., P_K$ such that both of the following hold:

- Every element in P_i is smaller than all the elements in P_j for any i, j satisfying $1 \leq i < j \leq K$.
- $a \leq |P_i| \leq b$ for all $i \in [1, K]$.

The algorithm is required to output $P_1, ..., P_K$ in a linked list, where the elements of P_1 precede those of P_2, followed by those of P_3, and so on. The relative positions of the elements in the *same* partition are not important.

In the second problem—called *approximate K-splitters*—it is not necessary to actually perform the partitioning; instead, the goal is to indicate *where to*. Formally, we are given the same parameters as in approximate K-partitioning, but want to find $K-1$ elements $s_1, ..., s_{K-1}$ from $\mathcal{S}$ called *splitters* with the property below:

- Suppose $s_1 < ... < s_{K-1}$, and let $s_0 = -\infty$ and $s_K = \infty$. Then, $a \leq |\mathcal{S} \cap (s_{i-1}, s_i]| \leq b$ for every $i \in [1, K]$.

An algorithm can output the splitters in any order.

Motivation. Both the above problems are frequently encountered in manipulating ordered elements. Partitioning naturally arises, for example, in distributing $\mathcal{S}$ onto a number K of machines for parallel processing. Achieving a perfectly balanced load (where each machine is responsible for N/K elements) is a special instance of approximate K-partitioning with $a = b = N/K$. As we will see, interestingly, the cost of partitioning can be reduced if one is satisfied with a roughly balanced distribution where each machine is allocated at least a but at most b elements (with $a \neq b$). Splitters, on the other hand, have been very useful in building statistical profiles of $\mathcal{S}$. For example, the bucket boundaries of an *equi-depth* histogram of K buckets (also known as a $(1/K)$-*quantile*) correspond to the output of the approximate K-splitters problem with $a = b = N/K$. If one can accept a *nearly equi-depth* histogram where each bucket covers at least a but at most b elements, then the bucket boundaries

*Xiaocheng Hu and Yufei Tao were supported in part by projects GRF 4165/11, 4164/12, and 4168/13 from HKRGC. Yi Yang and Shuigeng Zhou were supported in part by the Research Innovation Program of Shanghai Municipal Education Commission under grant No. 13ZZ003.

SPAA'14, June 23–25, 2014, Prague, Czech Republic.

Http://dx.doi.org/10.1145/2612669.2612691.

can be found in less—sometimes even sublinear—time, as we will show in this paper.

Math Conventions and Computation Model. We are interested in solving the two problems in the *external memory* (EM) model [1]. In this model, a machine is equipped with memory of size M words, and a disk that has been formatted into *blocks* of size B words. It holds that $M \geq 2B$. An I/O operation either reads a block of data from the disk to memory, or writes B words in memory to a disk block. The cost of an algorithm is measured by the number of I/Os performed. CPU calculation is free. All the algorithms discussed in this paper are comparison based, and adhere to the *indivisibility assumption* that each data element is always stored as a whole.

We define $\lg_x y = \max\{1, \log_x y\}$. The base x equals 2 if omitted. *Linear cost* refers to $O(N/B)$ when the problem at hand has an input size of N.

1.1 Parameter Ranges and Companion Problems

It is easy to see that the parameters a, b and K must satisfy the conditions

$$a \leq N/K \text{ and } b \geq N/K;$$

otherwise, the approximate K-partitioning and K-splitters problems both have no solution.

Let us quickly get rid of the case $K = N$. In such a scenario, approximate K-partitioning degenerates into *sorting*, while for approximate K-splitters an algorithm can simply return the input $\mathcal{S}$ directly. Henceforth, we will consider $K \leq N/2$.

When $a = 0$, the approximate K-partitioning/splitters problem is said to be *left-grounded*. Similarly, when $b = N$, the problem is *right-grounded*. When $a \neq 0$ and $b \neq N$, the problem is said to be *two-sided*.

The problems we study are closely related to two other important problems:

- *Multi-Partition*: Besides $\mathcal{S}$, we are given $K-1$ integers $\sigma_1, ..., \sigma_{K-1}$, and need to partition $\mathcal{S}$ into $P_1, ..., P_K$ such that (i) $|P_i| = \sigma_i$ for each $i \in [1, K-1]$, and (ii) all elements in P_i are smaller than those in P_j for any i, j satisfying $1 \leq i < j \leq K$.
- *Multi-Selection*: Besides $\mathcal{S}$, we are given K ranks[1] $r_1, ..., r_K$, and need to report K elements $e_1, ..., e_K$ in $\mathcal{S}$ such that e_i $(1 \leq i \leq K)$ has rank r_i in $\mathcal{S}$.

1.2 Previous Results

It is clear that all the above problems can be trivially solved by sorting in $O(\frac{N}{B} \lg_{M/B} \frac{N}{B})$ I/Os. The interesting question is when it is possible to do better.

The multi-partition problem is already well understood. Aggarwal and Vitter [1] gave an algorithm that performs $O(\frac{N}{B} \lg_{M/B} K)$ I/Os, which is optimal (we suspect that the optimality may be folklore, but we are not aware of a published proof of the lower bound; thus, we give one in the appendix—see the proof of Lemma 5).

Multi-selection can be also be solved in $O(\frac{N}{B} \lg_{M/B} K)$ I/Os by first doing a multi-partition, and then returning the largest element of each partition. In internal memory, the problem requires $\Theta(N \lg K)$ comparisons [7]. Combining the internal-memory lower bound with a general result of Arge, Knudsen and Larsen [2] gives a lower bound $\Omega(\frac{N}{B} \lg_{M/B} \frac{K}{B})$ in EM. Therefore, the aforementioned approach (by resorting to multi-partition) is not optimal when K is small. It remains open to close this gap.

A multi-partition algorithm can be directly applied to solve the approximate K-partitioning problem: simply divide $\mathcal{S}$ into K partitions of equal size. Therefore, regardless of a and b, approximate K-partitioning can always be settled in $O(\frac{N}{B} \lg_{M/B} K)$ I/Os. This further implies that the same bound holds on the approximate K-splitters problem (first do approximate K-partitioning and then return each partition's max element).

Recently, Hu et al. [6] studied a special instance of the approximate K-splitters problem, where $K = M$, $a = c_1 N/M$, and $b = c_2 N/M$, with c_1 and c_2 being some constant. They gave an algorithm solving the problem in $O(N/B)$ I/Os. Unfortunately, their algorithm does not extend to the case where $K > M$, and/or arbitrary a and b.

1.3 Our Main Results

In this work, we present matching upper and lower bounds for both the approximate K-partitioning and splitters problems.

Lower Bounds. Concerning approximate K-splitters, our first result is:

THEOREM 1. *For any $a \in [2, N/K]$, any comparison-based algorithm solving the right-grounded approximate K-splitters problem must perform $\Omega((1 + \frac{aK}{B}) \lg_{M/B} \frac{K}{B})$ I/Os in the worst case.*

Note that when $a = o(N/(K \lg_{M/B} \frac{K}{B}))$, the lower bound is sublinear (as we will see, this bound is tight)! This is interesting because all the existing lower bound machineries [1, 2, 5] in EM are inherently designed to prove bounds at least linear, while our argument circumvents this obstacle.

Our second result on approximate K-splitters is:

THEOREM 2. *For any $b \in [N/K, N/2]$, any comparison-based algorithm solving the left-grounded approximate K-splitters problem must perform $\Omega(\frac{N}{B} \lg_{M/B} \frac{N}{bB})$ I/Os in the worst case.*

As a corollary, for any $a \in [2, N/K]$ and any $b \in [N/K, N/2]$, any algorithm solving the two-sided approximate K-splitters problem must incur

$$\Omega\left(\max\left\{\left(1 + \frac{aK}{B}\right) \lg_{M/B} \frac{K}{B}, \frac{N}{B} \lg_{M/B} \frac{N}{bB}\right\}\right)$$

I/Os in the worst case.

Concerning approximate K-partitioning, we prove:

THEOREM 3. *If $\lg N \leq B \lg \frac{M}{B}$, any comparison-based algorithm solving the approximate K-partitioning problem must perform $\Omega(\frac{N}{B} \lg_{M/B} \min\{\frac{N}{b}, \frac{N}{B}\})$ I/Os in the worst case.*

Interestingly, our proof of the above theorem does not use combinatorial arguments, but is instead derived from an elegant reduction from multi-partition to left-grounded approximate K-partitioning.

[1]We follow the convention that, in an ordered set, the element with rank i is the i-th smallest in the set.

		Lower bound	Upper bound	Remarks
K-splitters	right	$\Theta((1+\frac{aK}{B})\lg_{M/B}\frac{K}{B})$		Thm 1, 5
	left	$\Theta(\frac{N}{B}\lg_{M/B}\frac{N}{bB})$		Thm 2, 5
	2-sided	$\Theta\left(\left(1+\frac{aK}{B}\right)\lg_{M/B}\frac{K}{B}+\frac{N}{B}\lg_{M/B}\frac{N}{bB}\right)$		Thm 1, 2, 5
K-partitioning	right	$\Omega(N/B)$	$O(\frac{N}{B}+\frac{aK}{B}\lg_{M/B}\min\{K,\frac{aK}{B}\})$	Sec 3, Thm 6
	left	$\Theta(\frac{N}{B}\lg_{M/B}\min\{\frac{N}{b},\frac{N}{B}\})$		Thm 3, 6
	2-sided	$\Omega(\frac{N}{B}\lg_{M/B}\min\{\frac{N}{b},\frac{N}{B}\})$	$O(\frac{aK}{B}\lg_{M/B}\min\{K,\frac{aK}{B}\}+\frac{N}{B}\lg_{M/B}\min\{\frac{N}{b},\frac{N}{B}\})$	Sec 3, Thm 3, 6

Table 1: Summary of our results

Upper Bounds. In terms of algorithms, our main contribution is an optimal solution to multi-selection:

THEOREM 4. *There is an algorithm that solves the multi-selection problem in $O(\frac{N}{B}\lg_{M/B}\frac{K}{B})$ I/Os.*

This finally closes the gap between the upper and lower bounds on this problem. Equipped with this new weapon, we present algorithms for both the approximate K-splitters and approximate K-partitioning problems with optimal performance (except for a single case of approximate K-partitioning where the value of aK is close to N). Table 1 summarizes these results.

The establishment of Theorem 4 has another implication. As mentioned earlier, there is a lower bound of $\Omega(\frac{N}{B}\lg_{M/B}K)$ on the multi-partition problem. Hence, Theorem 4 formally separates multi-selection from multi-partition in terms of I/O-complexity. Notice that the separation occurs only for small K, whereas for large K, the two problems have the same hardness. This phenomenon is interesting because in internal memory the two problems have exactly the same complexity: both demand $\Theta(N\lg K)$ comparisons.

2. LOWER BOUNDS: APPROXIMATE K-SPLITTERS

In this section, we establish our lower bounds for the approximate K-splitters problem. As before, let $s_1, ..., s_{K-1}$ in ascending order be the splitters returned by an algorithm. Define dummy $s_0 = -\infty$ and $s_K = \infty$. Given a pair of consecutive splitters s_{i-1} and s_i ($1 \le i \le K$), we say that they *induce* a partition P on the dataset $\mathcal{S}$ where $P = \mathcal{S} \cap (s_{i-1}, s_i]$. Recall that $|P|$ must fall between a and b.

2.1 Right-Grounded

This subsection serves as a proof for Theorem 1 (i.e., $b = N$). Let us first discuss the simple case $K < \alpha M$—where α is a constant to be determined later—under which the target lower bound is $\Omega(1+\frac{aK}{B})$. Consider an algorithm that sees $N_0 \in [0, N]$ elements $\mathcal{S}$ during its execution. In the K partitions induced by the returned splitters, there must exist a partition P containing at most N_0/K seen elements. Since all the other $N - N_0$ elements have not been seen, they can have any possible ranks in $\mathcal{S}$, so it is possible that none of them is in P, i.e., $|P| \le N_0/K$. By problem definition, $|P| \ge a$, which gives $N_0/K \ge a$. Hence, $N_0 \ge aK$, which means the algorithm must spend $\Omega(1+\frac{N_0}{B}) = \Omega(1+\frac{aK}{B})$ I/Os.

The rest of the subsection assumes $K \ge \alpha M$, where our target lower bound becomes $\Omega(\frac{aK}{B}\lg_{M/B}\frac{K}{B})$. We define a family Π_{hard} of hard permutations of $\mathcal{S}$ as follows. First recall that $\mathcal{S}$ is initially stored in N/B input blocks, each with B elements. For each $i \in [1, B]$, let $\mathcal{S}_i$ be the set consisting of the i-th element of every input block. Then, Π_{hard} consists of all those permutations where for any i, j satisfying $1 \le i < j \le B$, every element in $\mathcal{S}_i$ is smaller than all elements in $\mathcal{S}_j$. Clearly, $|\Pi_{hard}| = ((N/B)!)^B$.

Consider an algorithm that solves the problem in H I/Os. Let Π be the set of permutations in Π_{hard} that are consistent with all the comparisons performed by the algorithm. A standard argument (see appendix) shows that:

LEMMA 1. $|\Pi| \ge ((N/B)!)^B / \binom{M}{B}^H$.

Next, we will derive an upper bound on $|\Pi|$ (which will then yield a lower bound on H together with Lemma 1). For this purpose, we will analyze $|\Pi|$ by resorting to order theory. Let us first define some general concepts concerning partial orders. Let $\prec$ be a partial order defined on some domain. Consider a set X of elements from this domain, and π a permutation of X. We say that π is *consistent* with $\prec$ if, for any $x, y \in X$ such that $x \prec y$, x precedes y in π. Let $\mathrm{CP}(\prec, X)$ be the set of permutations of X consistent with $\prec$.

Now we are ready to proceed with our analysis on $|\Pi|$. First, define a partial order $\prec_\star$ over $\mathcal{S}$ as follows: given two elements $x, y \in \mathcal{S}$, $x \prec_\star y$ if and only if x precedes y in *all* the permutations in Π. We prove the following intuitive fact in the appendix:

FACT 1. $\Pi = \mathrm{CP}(\prec_\star, \mathcal{S})$.

Let x, y be two different elements in $\mathcal{S}$. If either $x \prec_\star y$ or $y \prec_\star x$ holds, we say that they are *comparable*; otherwise, they are *incomparable*. We observe:

FACT 2. *For any i, j satisfying $1 \le i < j \le K-1$, s_i and s_j are comparable.*

PROOF. Suppose that s_i and s_j are incomparable. Then we can find a permutation $\pi \in \mathrm{CP}(\prec_\star, \mathcal{S})$ in which s_i precedes s_j without any element between them. By Fact 1, $\pi \in \Pi$, so the algorithm has to be correct on π. However, the partition induced on π between s_i and s_j has only one element s_j, contradicting the requirement that each partition has size at least $a \ge 2$ (as is a condition of Theorem 1). □

By Fact 2, we have that $s_1 \prec_\star ... \prec_\star s_{K-1}$. For each $i \in [2, K-1]$, define $T_i = \{x \in \mathcal{S} \mid s_{i-1} \prec_\star x \prec_\star s_i\}$. Also define $T_1 = \{x \in \mathcal{S} \mid x \prec_\star s_1\}$ and $T_K = \{x \in \mathcal{S} \mid s_{K-1} \prec_\star x\}$.

FACT 3. $|T_i| \ge a-1$ *for each* $i \in [1, K-1]$, *and* $|T_K| \ge a$.

PROOF. For any $i \in [2, K-1]$, there exists a permutation $\pi \in \mathrm{CP}(\prec_*, \mathcal{S}) = \Pi$ in which only elements in T_i are between s_{i-1} and s_i. Therefore, the partition induced on π between the two splitters is exactly $T_i \cup \{s_i\}$, implying that $|T_i| \geq a-1$. $|T_1| \geq a-1$ and $|T_K| \geq a$ can be proved similarly. □

We also need the following basic facts from order theory:

FACT 4. *Let $\prec$ be a partial order over a finite set X. If X can be divided into disjoint partitions X_1 and X_2, such that $x \prec y$ holds for any $x \in X_1$ and $y \in X_2$, then* $|\mathrm{CP}(\prec, X)| = |\mathrm{CP}(\prec, X_1)| \cdot |\mathrm{CP}(\prec, X_2)|$.

FACT 5. *Let $\prec$ be a partial order over a finite set X. For any subset $Y \subseteq X$, it holds that* $|\mathrm{CP}(\prec, X)| \leq |\mathrm{CP}(\prec, Y)| \cdot |\mathrm{CP}(\prec, X \setminus Y)| \cdot \binom{|X|}{|Y|}$.

Now we are ready to give an upper bound of $|\Pi| = |\mathrm{CP}(\prec_\star, \mathcal{S})|$, as promised earlier:

LEMMA 2.

$$\begin{aligned}\lg |\mathrm{CP}(\prec_\star, \mathcal{S})| &\leq B \lg((N/B)!) - aK \lg(K/B) \\ &\quad + O(K \lg a)\end{aligned}$$

PROOF. In this proof, we abbreviate $\mathrm{CP}(\prec_\star, X)$ into $\mathrm{CP}(X)$ for any $X \subseteq \mathcal{S}$. Recall that the elements in $\mathcal{S}$ come from B disjoint sets $\mathcal{S}_1, ..., \mathcal{S}_B$, where each $\mathcal{S}_i$ $(1 \leq i \leq B)$ takes the i-th element of every input block. In any permutation $\pi \in \Pi$, for any $x \in \mathcal{S}_i$ and $y \in \mathcal{S}_j$ $(1 \leq i < j \leq B)$, we have that x precedes y in π because $\Pi \subseteq \Pi_{hard}$. Therefore, $x \prec_\star y$ by the definition of $\prec_\star$.

For each $i \in [1, K-1]$, let A_i be a set consisting of s_i and $a-1$ arbitrary elements from T_i. Also let A_K be a set consisting of a arbitrary elements in T_k. By Fact 3, $A_1, ..., A_K$ exist. By definition, for any $x \in A_i$ and $y \in A_j$ $(1 \leq i < j \leq K)$, we have that $x \prec y$. Let $A = A_1 \cup ... \cup A_K$. Therefore,

$$\begin{aligned}
& |\mathrm{CP}(\mathcal{S})| \\
= & \prod_{i=1}^{B} |\mathrm{CP}(\mathcal{S}_i)| \quad \text{(by Fact 4)} \\
\leq & \prod_{i=1}^{B} \left(\binom{|\mathcal{S}_i|}{|\mathcal{S}_i \setminus A|} \cdot |\mathrm{CP}(\mathcal{S}_i \setminus A)| \cdot |\mathrm{CP}(\mathcal{S}_i \cap A)| \right) \\
& \text{(by Fact 5)} \\
= & \prod_{i=1}^{B} \left(\binom{|\mathcal{S}_i|}{|\mathcal{S}_i \setminus A|} \cdot |\mathrm{CP}(\mathcal{S}_i \setminus A)| \cdot \prod_{j=1}^{K} |\mathrm{CP}(\mathcal{S}_i \cap A_j)| \right) \\
& \text{(by Fact 4)} \\
\leq & \prod_{i=1}^{B} \left(\binom{|\mathcal{S}_i|}{|\mathcal{S}_i \setminus A|} \cdot |\mathcal{S}_i \setminus A|! \cdot \prod_{j=1}^{K} |\mathcal{S}_i \cap A_j|! \right). \qquad (1)
\end{aligned}$$

Then the lemma follows from simplification of (1), which can be found in the appendix. □

Let β be the hidden constant in the term $O(K \lg a)$ in 2. Now we fix $\alpha = 2^{2\beta}$. Lemmas 1 and 2 give

$$\begin{aligned}
& B \lg((N/B)!) - H \lg \binom{M}{B} \\
\leq & B \lg((N/B)!) - aK \lg(K/B) + \beta K \lg a.
\end{aligned}$$

Hence:

$$\begin{aligned}
H \lg \binom{M}{B} &\geq aK \lg(K/B) - \beta K \lg a \\
\Rightarrow \quad H \cdot B \lg \frac{M}{B} &= \Omega\left(aK \lg(K/B) - \beta K \lg a\right) \\
\Rightarrow \quad H &= \Omega((aK/B) \lg_{M/B}(K/B))
\end{aligned}$$

as needed, where the last step used the fact that $a \lg(K/B) \geq a \lg(\alpha M/B) \geq 2a\beta$.

2.2 Left-Grounded

This subsection will prove Theorem 2 (i.e., $a = 0$). Let us first start with the simple case $N/b < \alpha M$—where α is a constant to be determined later—under which our target lower bound becomes $\Omega(N/B)$. Consider an algorithm that sees $N_0 \in [0, N]$ elements in $\mathcal{S}$ during its execution. Then, we can construct a permutation, such that all the $N - N_0$ unseen elements are in the same partition induced by the splitters. By problem definition, $N - N_0 \leq b$, which together with the condition $b \leq N/2$ (of Theorem 2) implies that $N_0 \geq N/2$. Hence, the algorithm must perform $\Omega(N/B)$ I/Os.

The rest of the subsection assumes that $N/b \geq \alpha M$. Let Π_{hard} and H be defined in the same way as in Section 2.1. Lemma 1 still holds, giving a lower bound on $|\Pi|$.

To prove an upper bound on $|\Pi|$, we consider the same partial order $\prec_\star$ defined in Section 2.1. As before, given elements $x, y \in \mathcal{S}$, they are *comparable* if either $x \prec_\star y$ or $y \prec_\star x$ holds; otherwise, they are *incomparable*. Fact 1 still holds.

Define $T_i = \mathcal{S}_i \setminus \{s_1, ..., s_{K-1}\}$ for each $i \in [1, B]$, i.e., T_i is the set of non-splitter elements in $\mathcal{S}_i$. Let $T = T_1 \cup ... \cup T_B$, i.e., T is the set of all the non-splitter elements. Note that $|T| \geq N/2$ because $K \leq N/2$.

FACT 6. *For any $i \in [1, B]$, let A be a subset of T_i, such that all the elements in A are pairwise incomparable. Then* $|A| \leq b$.

PROOF. Since the elements in A are pairwise incomparable, there exists a permutation $\pi \in \mathrm{CP}(\prec_*, \mathcal{S}) = \Pi$, such that all the elements of A appear consecutively in π. Furthermore, $A \subseteq T_i$ implies that A does not contain any splitter. Therefore, there exists a partition induced on π by the splitters which contains all the elements in A. The size of the partition must be at most b, so $|A| \leq b$. □

In the appendix, we prove a general lemma in order theory:

LEMMA 3. *Consider a partial order $\prec$ over a finite set X of n elements. Any set of pairwise incomparable elements from X has size at most w. Then,* $\lg |\mathrm{CP}(\prec, X)| \leq n \lg w + O(\lg n)$.

Combining Fact 6 and Lemma 3, we know that, for each $i \in [1, B]$:

$$\lg |\mathrm{CP}(\prec_\star, T_i)| \leq |T_i| \lg b + O(\lg |T_i|). \qquad (2)$$

Now we show an upper bound on $|\mathrm{CP}(\prec_\star, \mathcal{S})|$:

LEMMA 4.

$$\begin{aligned}\lg |\mathrm{CP}(\prec_\star, \mathcal{S})| &\leq B \lg((N/B)!) - |T| \lg(|T|/(bB)) \\ &\quad + O(|T|)\end{aligned}$$

PROOF. In this proof, we abbreviate $\text{CP}(\prec_\star, X)$ into $\text{CP}(X)$ for any $X \subseteq \mathcal{S}$. Recall that for any $1 \le i < j \le B$, $x \in \mathcal{S}_i$ and $y \in \mathcal{S}_j$, it holds that $x \prec_\star y$. Therefore,

$$\begin{aligned}
& |\text{CP}(\mathcal{S})| \\
= & \prod_{i=1}^{B} |\text{CP}(\mathcal{S}_i)| \quad \text{(by Fact 4)} \\
\le & \prod_{i=1}^{B} \binom{|\mathcal{S}_i|}{|T_i|} \cdot |\text{CP}(\mathcal{S}_i \setminus T_i)| \cdot |\text{CP}(T_i)| \quad \text{(by Fact 5)} \\
\le & \prod_{i=1}^{B} \binom{|\mathcal{S}_i|}{|T_i|} \cdot |\mathcal{S}_i \setminus T_i|! \cdot |\text{CP}(T_i)|. \qquad (3)
\end{aligned}$$

Then the lemma follows from combining and simplifying (2) and (3). See appendix for details. □

Let β be the hidden constant in the term $O(|T|)$ in Lemma 4. Now we set constant α to be $2^{2\beta}$. From Lemmas 1 and 4, we have

$$\begin{aligned}
& B \lg((N/B)!) - H \lg \binom{M}{B} \\
\le & B \lg((N/B)!) - |T| \lg(|T|/(bB)) + \beta|T|.
\end{aligned}$$

Therefore,

$$\begin{aligned}
H \lg \binom{M}{B} &\ge |T| \lg(|T|/(bB)) - \beta|T| \\
\Rightarrow H \cdot B \lg(M/B) &= \Omega(|T| \lg(|T|/(bB)) - \beta|T|) \\
\Rightarrow H &= \Omega((N/B) \lg_{M/B}(N/(bB)))
\end{aligned}$$

as needed, where the last step used the fact that $\lg \frac{|T|}{bB} \ge \lg \frac{N}{2bB} \ge \lg \frac{\alpha M}{2B} \ge 2\beta$.

3. LOWER BOUNDS: APPROXIMATE K-PARTITIONING

In this section, we will prove the lower bound in Theorem 3 for the approximate K-partitioning problem. Recall that the goal is to divide the dataset $\mathcal{S}$ into K partitions by respecting the ordering, such that the size of each partition falls in $[a, b]$. As in Section 2, we will discuss the left- and right-grounded versions of the problem separately. Theorem 3 will then follow from our results for these versions.

Right-Grounded. When $b = N$, any algorithm must see all the elements at least once, as long as $a \ge 1$ and $K \ge 2$. To understand this, suppose that the algorithm terminates without seeing all elements. Consider an arbitrary seen element e and an arbitrary unseen element e'. Regardless of whether the algorithm puts e, e' in the same partition, the adversary can always manipulate the value of e' to call the algorithm wrong. Therefore, any algorithm must perform $\Omega(N/B)$ I/Os.

Left-Grounded. Now we will prove a lower bound for the case where $a = 0$. For this purpose, let us define the *precise K-partitioning problem* as the special instance of the multi-partition problem with $\sigma_1 = ... = \sigma_{K-1} = N/K$. We will first present a reduction from precise K-partitioning to left-grounded approximate K-partitioning. Then, we will prove a lower bound for the former problem.

Our reduction works as follows. Suppose that there is an algorithm solving the left-grounded problem in $F(N, K, b)$ I/Os (recall that $b \ge N/K$ needs to hold). The same argument we gave for the right-grounded case implies that $F(N, K, b) = \Omega(N/B)$ when $b < N$ and $K \ge 2$. Assuming that N is a multiple of b, we can perform precise (N/b)-partitioning as follows:

1. Solve an approximate K-partitioning problem to divide $\mathcal{S}$ into partitions $P_1, ..., P_K$ (in this order), where each partition has size at most b.
2. Let R be an initially empty set. We process $P_1, ..., P_K$ in turn. At P_i ($i \in [1, K]$), we first append the entire P_i to R. If $|R| > b$, divide R into disjoint partitions R_1 and R_2 such that every element in R_1 is smaller than all elements in R_2, and $|R_1| = b$. Then R_1 is returned as the next partition in the precise (N/b)-partitioning, and R_2 replaces R as we proceed to process P_{i+1}.

The first step requires $F(N, K, b)$ I/Os, while the second step requires $O(N/B)$ I/Os in total. Therefore, we can solve the precise (N/b)-partitioning in $F(N, K, b) + O(N/B) = O(F(N, K, b))$ I/Os. However, in the appendix, we prove:

LEMMA 5. *If $K \ge 2$ and $\lg N \le B \lg \frac{M}{B}$, a comparison-based algorithm solving the precise K-partitioning problem must perform $\Omega(\frac{N}{B} \lg_{M/B} \min\{K, \frac{N}{B}\})$ I/Os in the worst case.*

The lemma then implies

$$F(N, K, b) = \Omega((N/B) \lg_{M/B} \min\{N/b, N/B\}).$$

Note that K has no effect on the lower bound. We thus complete the proof of Theorem 3.

4. MULTI-SELECTION

This section develops a new algorithm for the multi-selection problem. Towards this purpose, we will first solve a relevant problem called *L-intermixed selection* in Section 4.1, and then leverage our solution to attack multi-selection in Section 4.2.

4.1 L-Intermixed Selection

Set $m = cM$ for some sufficiently small constant c. Given an integer $L \in [1, m]$, the *L-intermixed selection* is defined as follows. The input consists of:

- A set $\mathcal{D}$ with each element being a pair $e = (k_e, g_e)$, where k_e is drawn from an ordered domain, and g_e is an integer in $[1, L]$. We refer to k_e (g_e) as the *key* (*group id*) of e. For each $i \in [1, L]$, denote by $\mathcal{D}_i$ the set of elements in $\mathcal{D}$ with group id i. Each of $\mathcal{D}_1, ..., \mathcal{D}_L$ is called a *group*.
- L integers $t_1, ..., t_L$, where $1 \le t_i \le |\mathcal{D}_i|$ for each $i \in [1, L]$.

The objective is to report, for each $i \in [1, L]$, the element e_i with the t_i-th smallest key in $\mathcal{D}_i$. Intuitively, we want to solve L instances of rank selection, but the L datasets are intermixed.

Algorithm. Whenever we compare two elements e and e' from the same group, we are comparing their keys k_e and $k_{e'}$.

Similarly, when we say e is the median of a subset S of some group, we mean that e has the median key in S. If $|\mathcal{D}| \leq M/3$, the problem can be solved trivially by loading $\mathcal{D}$ and $t_1, ..., t_L$ entirely into memory. The subsequent discussion assumes $|\mathcal{D}| > M/3$.

Our algorithm can be thought of as concurrently running L threads of the "median-of-medians" selection algorithm [3]. However, doing so naively would demand a block of memory dedicated to each thread. This is an issue because it will allow us to do only $O(M/B)$ threads at a time, whereas $L = \Omega(M)$. Next we show how to overcome the obstacle by using only $O(1)$ words of memory for each thread.

In the first step, by scanning $\mathcal{D}$ only once, we can divide each group arbitrarily into *subgroups* of size at most 5, and collect the median of each subgroup into a set Σ. To do so, maintain L sets $S_1, ..., S_L$ in memory, which are initially empty. For each element $e = (k_e, g_e)$ in $\mathcal{D}$, we first insert e into S_{g_e}. If $|S_{g_e}| = 5$, then S_{g_e} makes a subgroup, so we add the median of S_{g_e} to Σ, and then clear the contents of S_{g_e}. After all elements have been scanned, if any S_i $(1 \leq i \leq L)$ is non-empty, then S_i is the last subgroup of $\mathcal{D}_i$; its median is added to Σ.

Now Σ consists of the medians of all the subgroups. For each $i \in [1, L]$, denote by Σ_i the set of elements with group id i in Σ. By recursion, we can find the medians of $\Sigma_1, ..., \Sigma_L$, denoted as $\mu_1, ..., \mu_L$, respectively. With another scan of $\mathcal{D}$, we can obtain the rank θ_i of μ_i in $\mathcal{D}_i$ for each $i \in [1, L]$.

In the last step, we construct another (smaller) instance of L-intermixed selection, with $\mathcal{D}' = \mathcal{D}'_1 \cup ... \cup \mathcal{D}'_L$ and $t'_1, ..., t'_L$ as the input. For each $i \in [1, L]$, if $t_i \leq \theta_i$, then $\mathcal{D}'_i = \mathcal{D}_i \cap (-\infty, \mu_i]$ and $t'_i = t_i$; otherwise, $\mathcal{D}'_i = \mathcal{D}_i \cap (\mu_i, \infty)$ and $t'_i = t_i - \theta_i$. Clearly, by solving this instance, we also solve the original L-intermixed selection problem. To create the instance, we keep in memory $t_1, ..., t_L$, $\mu_1, ..., \mu_L$, $\theta_1, ..., \theta_L$. Then, $t'_1, ..., t'_L$ can be computed in memory. $\mathcal{D}'$ can be generated with another scan of $\mathcal{D}$ in $O(|\mathcal{D}|/B)$ I/Os as follows. For each element $e \in \mathcal{D}$, let $i \in [1, L]$ be its group id (i.e., $g_e = i$). By comparing e with μ_i and t_i with θ_i, we know whether $e \in \mathcal{D}'_i$; if so, add e to $\mathcal{D}'$.

Analysis. We now prove the efficiency of our algorithm:

LEMMA 6. *The above algorithm solves the L-intermixed selection problem in $O(|\mathcal{D}|/B)$ I/Os.*

PROOF. Let $F(\mathcal{D})$ be the cost of our algorithm on dataset $\mathcal{D}$. When $|\mathcal{D}| \leq M/3$, the algorithm simply solves the problem in memory. When $|\mathcal{D}| > M/3$, the algorithm recurses on Σ and $\mathcal{D}'$, respectively, and scans $\mathcal{D}$ for a constant number of times. Therefore,

$$F(\mathcal{D}) = \begin{cases} O(|\mathcal{D}|/B), & \text{if } |\mathcal{D}| \leq M/3 \\ O(|\mathcal{D}|/B) + F(\Sigma) + F(\mathcal{D}'), & \text{if } |\mathcal{D}| > M/3 \end{cases}.$$

Clearly, $|\Sigma| = \sum_{i=1}^{L} \lceil |\mathcal{D}_i|/5 \rceil \leq |\mathcal{D}|/5 + L$. By the analysis in [3], we can show that $|\mathcal{D}'_i| \leq \frac{7}{10}|\mathcal{D}_i| + 3$ for each $i \in [1, L]$. Hence, $|\mathcal{D}'| \leq \frac{7}{10}|\mathcal{D}| + 3L$. So we have

$$|\Sigma| + |\mathcal{D}'| \leq \frac{9}{10}|\mathcal{D}| + 4L \leq \frac{9}{10}|\mathcal{D}| + 4cM \leq \left(\frac{9}{10} + 12c\right)|\mathcal{D}|,$$

where the last inequality follows from $|\mathcal{D}| > M/3$. By setting c sufficiently small, $|\Sigma| + |\mathcal{D}'|$ is at most $\frac{19}{20}|\mathcal{D}|$. Then by standard analysis, the recursion is solved as $F(\mathcal{D}) = O(|\mathcal{D}|/B)$. □

4.2 Solving Multi-Selection

Recall that in the multi-selection problem, we need to find elements $e_1, ..., e_K$ from $\mathcal{S}$, such that each e_i $(1 \leq i \leq K)$ has rank r_i in $\mathcal{S}$. Let m be as defined in Section 4.1. We first show that the in the *base case* where $K \leq m$, the problem can be solved in linear I/Os. After that, we solve the *general case* where $K > m$ by decomposing it into base cases.

Base Case $K \leq m$. As mentioned in Section 1.3, Hu et al. [6] has solved the special case $K = M, a = c_1 N/M$ and $b = c_2 N/M$ of approximate K-splitters in linear I/Os for some constants c_1 and c_2. Let $s_1 < ... < s_{M-1}$ be the splitters returned by their algorithm on $\mathcal{S}$. Define $P_i = \mathcal{S} \cap (s_{i-1}, s_i]$ for each $i \in [1, M]$ (define dummy $s_0 = -\infty$ and $s_M = \infty$). Note that $|P_i| = \Theta(N/M)$.

Consider the following instance of the K-intermixed selection problem. For each $i \in [1, K]$, let P_j $(1 \leq j \leq M)$ be the partition such that e_i is in P_j. Define group $\mathcal{D}_i = \{(e, i) \mid e \in P_j\}$, and rank $t_i = r_i - (|P_1| + ... + |P_{j-1}|)$. Then, $\mathcal{D} = \mathcal{D}_1 \cup ... \cup \mathcal{D}_K$. The output of this instance is exactly the set of elements $e_1, ..., e_K$ we want for the original multi-selection problem. The instance can be solved in $O(|\mathcal{D}|/B)$ I/Os by Lemma 6, which is $O(N/B)$ because each group has size $\Theta(N/M)$ and there are $K < M$ groups.

It suffices to show that the instance can be constructed in $O(N/B)$ I/Os. To do so, we first obtain $|P_1|, ..., |P_M|$ with one scan of $\mathcal{S}$, while keeping $s_1, ..., s_{M-1}$ memory resident. After this, $t_1, ..., t_K$ can be calculated in memory. We scan $\mathcal{S}$ one more time to construct $\mathcal{D}$ as follows. For each element $e \in \mathcal{S}$, by comparing it with the splitters, we get the partition id $j \in [1, M]$ such that $e \in P_j$. Then, by from the ranks $r_1, ..., r_K$ and the partition sizes $|P_1|, ..., |P_M|$, we can find (in memory) those group ids $i \in [1, K]$ such that $\mathcal{D}_i$ should contain pair (e, i): specifically, find every i that $|P_1| + ... + |P_{j-1}| < r_i \leq |P_1| + ... + |P_j|$. Once such an i is found, insert pair (e, i) into $\mathcal{D}$. In this manner, we generate $\mathcal{D}$ in $O(N/B)$ I/Os.

General Case: $K > m$. Set $g = \lceil K/m \rceil$. Perform multi-partition (using an algorithm of [1], which is reviewed in Section 1.2) to divide $\mathcal{S}$ at ranks $r_m, r_{2m}, ..., r_{(g-1)m}$ into g partitions $P_1, ..., P_g$. Now for each $i \in [1, g-1]$, the elements of ranks $r_{(i-1)m+1}, ..., r_{im}$ must be in P_i, and the elements of ranks $r_{(g-1)m+1}, ..., r_K$ must be in P_g. Since at most m elements need to be selected from each partition, it suffices to solve a base case on each of them.

Generating the partitions takes $O((N/B) \lg_{M/B} g) = O((N/B) \lg_{M/B}(K/B))$ I/Os, while solving all the base cases requires altogether $O(\sum_{i=1}^{g} |P_i|/B) = O(N/B)$ I/Os. We thus have completed the proof of Theorem 4.

5. ALGORITHMS FOR APPROXIMATE K-SPLITTERS AND K-PARTITIONING

This section presents algorithms for the approximate K-splitters and K-partitioning problems.

5.1 Approximate K-Splitters

Right-Grounded. Take a set $\mathcal{S}'$ of aK arbitrary elements in $\mathcal{S}$ in $O(1 + aK/B)$ I/Os. Then, simply return the $\frac{1}{K}$-quantile of $\mathcal{S}'$ as the splitters $s_1, ..., s_{K-1}$, namely, s_i $(1 \leq i \leq K-1)$ has rank ia in $\mathcal{S}'$. The cost is $O((1 + \frac{aK}{B}) \lg_{M/B} \frac{K}{B})$ by Theorem 4.

Left-Grounded. Set $K' = \lceil N/b \rceil$. Pick splitters $s_1, ..., s_{K'-1}$ from $\mathcal{S}$ such that s_i $(1 \leq i \leq K'-1)$ has rank ib in $\mathcal{S}$. This demands $O(\frac{N}{B} \lg_{M/B} \frac{K'}{B}) = O(\frac{N}{B} \lg_{M/B} \frac{N}{bB})$ I/Os by Theorem 4. After that, if $K' < K$, we select $s_{K'}, ..., s_{K-1}$ as $K - K'$ arbitrary distinct elements in $\mathcal{S} \setminus \{s_1, ..., s_{K'-1}\}$. The total cost is $O(\frac{N}{B} \lg_{M/B} \frac{N}{bB})$.

Two-Sided. We first get rid of the scenario where $a \geq N/2K$ or $b \leq 2N/K$. In this case, we simply return the $\frac{1}{K}$-quantile of $\mathcal{S}$ as the splitters (namely, splitter s_i has rank iN/K in $\mathcal{S}$, where $1 \leq i \leq K-1$). The cost is $O(\frac{N}{B} \lg_{M/B} \frac{K}{B}) = O(\max\{\frac{aK}{B} \lg_{M/B} \frac{K}{B}, \frac{N}{B} \lg_{M/B} \frac{N}{bB}\})$.

Henceforth, we assume $a < N/2K$ and $b > 2N/K$. Set $K' = \lfloor \frac{bK-N}{b-a} \rfloor$. It is easy to verify that $K' \in [1, K-1]$. Our algorithm partitions $\mathcal{S}$ into $\mathcal{S}_{low}$ and $\mathcal{S}_{high}$, such that $\mathcal{S}_{low}$ consists of the aK' smallest elements of $\mathcal{S}$, and $\mathcal{S}_{high}$ the remaining ones. After that, we determine the splitters $s_1, ..., s_{K-1}$ of $\mathcal{S}$ as follows:

- $s_{K'}$ is the largest element of $\mathcal{S}_{low}$;
- $s_1, ..., s_{K'-1}$ constitute the $\frac{1}{K'}$-quantile of $\mathcal{S}_{low}$;
- $s_{K'+1}, ..., s_{K-1}$ constitute the $\frac{1}{K-K'}$-quantile of $\mathcal{S}_{high}$.

To see the correctness of our algorithm, consider the K partitions induced by $s_1, ..., s_{K-1}$. Among them, we get K' even partitions of $\mathcal{S}_{low}$, and $K - K'$ even partitions of $\mathcal{S}_{high}$. Since $|\mathcal{S}_{low}| = aK'$, each of the partitions from $\mathcal{S}_{low}$ has exactly a elements. To show that the partitions from $\mathcal{S}_{high}$ have legal sizes, it suffices to prove that $|\mathcal{S}_{high}| = N - aK'$ is in the range $[a(K-K'), b(K-K')]$. This is true by our choice of K' and the facts that $a < N/2K$ and $b > 2N/K$.

It takes $O(N/B)$ I/Os to obtain $\mathcal{S}_{low}$, $\mathcal{S}_{high}$ and $s_{K'}$. By Theorem 4, it takes $O((|\mathcal{S}_{low}|/B) \lg_{M/B}(K'/B))$ I/Os to get $s_1, ..., s_{K'-1}$ from $\mathcal{S}_{low}$, and $O((|\mathcal{S}_{high}|/B) \lg_{M/B}(K - K')/B)$ I/Os to get $s_{K'+1}, ..., s_{K-1}$ from $\mathcal{S}_{high}$. As $K - K' = \Theta(\frac{N-aK}{b-a}) = \Theta(N/b)$, $|\mathcal{S}_{low}| = aK' < aK$, and $|\mathcal{S}_{high}| < N$, we know that the total cost is bounded by $O(\frac{aK}{B} \lg_{M/B} \frac{K}{B} + \frac{N}{B} \lg_{M/B} \frac{N}{bB})$.

THEOREM 5. *For the approximate K-splitters problem, there is an algorithm that solves*

- *the right-grounded version in $O((1 + \frac{aK}{B}) \lg_{M/B} \frac{K}{B})$ I/Os;*
- *the left-grounded version in $O(\frac{N}{B} \lg_{M/B} \frac{N}{bB})$ I/Os;*
- *the two-sided version in $O(\frac{aK}{B} \lg_{M/B} \frac{K}{B} + \frac{N}{B} \lg_{M/B} \frac{N}{bB})$ I/Os.*

The cost for all three versions is optimal according to the lower bounds in Theorems 1 and 2.

5.2 Approximate K-Partitioning

Right-Grounded. Take the set $\mathcal{S}'$ of the $a(K-1)$ smallest elements of $\mathcal{S}$ in $O(N/B)$ I/Os. Then, divide $\mathcal{S}'$ into $K-1$ partitions of size a using the multi-partition algorithm of [1] in $O((1+\frac{aK}{B}) \lg_{M/B} \min\{K, \frac{aK}{B}\})$ I/Os. Simply treat $\mathcal{S} \setminus \mathcal{S}'$ as the K-th partition (whose size is $N - a(K-1) \geq a$).

Left-Grounded. Let $K' = \lceil N/b \rceil$. Perform multi-partition to divide $\mathcal{S}$ into K' partitions of size at most b. If $K' < K$, simply add $K - K'$ empty partitions. The total cost is $O(\frac{N}{B} \lg_{M/B} \min\{\frac{N}{b}, \frac{N}{B}\})$.

Two-Sided. The algorithm is analogous to our two-sided approximate K-splitters algorithm, by simply replacing multi-selection with multi-partition. More specifically, we generate $\mathcal{S}_{low}$ and $\mathcal{S}_{high}$ using the same K'. Then, we divide $\mathcal{S}_{low}$ into K' partitions of the same size, and $\mathcal{S}_{high}$ into $K - K'$ partitions of the same size, both respecting the ordering. The total cost is $O(\frac{aK}{B} \lg_{M/B} \min\{K, \frac{aK}{B}\} + \frac{N}{B} \lg_{M/B} \min\{\frac{N}{b}, \frac{N}{B}\})$.

THEOREM 6. *For the approximate K-partitioning problem, there is an algorithm that solves*

- *the right-grounded version in $O(\frac{N}{B} + \frac{aK}{B} \lg_{M/B} \min\{K, \frac{aK}{B}\})$ I/Os;*
- *the left-grounded version in $O(\frac{N}{B} \lg_{M/B} \min\{\frac{N}{b}, \frac{N}{B}\})$ I/Os;*
- *the two-sided version in $O(\frac{aK}{B} \lg_{M/B} \min\{K, \frac{aK}{B}\} + \frac{N}{B} \lg_{M/B} \min\{\frac{N}{b}, \frac{N}{B}\})$ I/Os.*

The cost for all three versions is optimal according to the discussion in Section 3. In particular, we have matching upper and lower bounds whenever $\log N < B \log(M/B)$ (which is the condition of Theorem 3) and $a \leq N/(K \lg_{M/B} K)$.

6. REFERENCES

[1] Alok Aggarwal and Jeffrey Scott Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM (CACM)*, 31(9):1116–1127, 1988.

[2] Lars Arge, Mikael Knudsen, and Kirsten Larsen. A general lower bound on the I/O-complexity of comparison-based algorithms. In *Algorithms and Data Structures Workshop (WADS)*, pages 83–94, 1993.

[3] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *Journal of Computer and System Sciences (JCSS)*, 7(4):448–461, 1973.

[4] Robert P. Dilworth. A decomposition theorem for partially ordered sets. *The Annals of Mathematics*, 51(1):161–166, 1950.

[5] Jeff Erickson. Lower bounds for external algebraic decision trees. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 755–761, 2005.

[6] Xiaocheng Hu, Cheng Sheng, Yufei Tao, Yi Yang, and Shuigeng Zhou. Output-sensitive skyline algorithms in external memory. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 887–900, 2013.

[7] Kanela Kaligosi, Kurt Mehlhorn, J. Ian Munro, and Peter Sanders. Towards optimal multiple selection. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 103–114, 2005.

APPENDIX

Proof of Lemma 1

In our context, the algorithm only needs to focus on the permutations in Π_{hard}. Therefore, prior to reading a block,

the algorithm must have already known the ordering of the elements there, regardless of whether the block had been written by the algorithm before. Thus, the algorithm can be described with a decision tree with fanout at most $\binom{M}{B}$. The lemma follows from the fact that $|\Pi_{hard}| = ((N/B)!)^B$.

Proof of Fact 1

We first show $\Pi \subseteq \mathrm{CP}(\prec_\star, \mathcal{S})$. Suppose that there is a permutation $\pi \in \Pi$ but $\pi \notin \mathrm{CP}(\prec_\star, \mathcal{S})$. This means that there exist elements x, y such that $x \prec_\star y$ but y precedes x in π. This is a contradiction by the definition of $\prec_\star$.

Next we show $\mathrm{CP}(\prec_\star, \mathcal{S}) \subseteq \Pi$. Suppose that there is a permutation $\pi \in \mathrm{CP}(\prec_\star, \mathcal{S})$ but $\pi \notin \Pi$. There must be elements x, y such that x precedes y in π, but the algorithm can infer $y < x$ from the comparisons it has performed. But $y < x$ implies that $y \prec_\star x$, which contradicts the assumption that $\pi \in \mathrm{CP}(\prec_\star, \mathcal{S})$.

Simplification of (1)

For each $i \in [1, B]$ and $j \in [1, K]$, let $a_{ij} = |\mathcal{S}_i \cap A_j|$. Also let $a_i = |\mathcal{S}_i \cap A| = \sum_{j=1}^{K} a_{ij}$ for each $i \in [1, B]$. By plugging in $|\mathcal{S}_1| = ... = |\mathcal{S}_B| = N/B$, we have

$$\begin{aligned}
|\mathrm{CP}(\mathcal{S})| &\le (1) \\
&= \left(\prod_{i=1}^{B} \binom{N/B}{a_i} \cdot (N/B - a_i)!\right) \cdot \left(\prod_{j=1}^{K}\prod_{i=1}^{B} a_{ij}\right)! \\
&\le \left(\prod_{i=1}^{B} \frac{(N/B)!}{a_i!}\right) \cdot \left(\prod_{j=1}^{K} \left(\sum_{i=1}^{B} a_{ij}\right)!\right) \\
&\quad \text{(in general, } x!y! \le (x+y)!\text{)} \\
&= ((N/B)!)^B \cdot (a!)^K / \prod_{i=1}^{B} a_i! \\
&\quad \text{(for each } j \in [1, K], \sum_{i=1}^{B} a_{ij} = |A_j| = a\text{)}.
\end{aligned}$$

This leads to:

$$\begin{aligned}
&\lg |\mathrm{CP}(\mathcal{S})| \\
&\le B \lg((N/B)!) + K \lg(a!) - \sum_{i=1}^{B} \lg(a_i!) \\
&\le B \lg((N/B)!) + aK \lg a - \sum_{i=1}^{B} a_i \lg a_i + O(K \lg a) \\
&\quad \text{(by Stirling's formula)} \\
&\le B \lg((N/B)!) + aK \lg a \\
&\quad - \left(\sum_{i=1}^{B} a_i\right) \lg\left(\frac{1}{B} \cdot \sum_{i=1}^{B} a_i\right) + O(K \lg a) \\
&\quad \text{(by convexity of function } x \lg x\text{)} \\
&\le B \lg((N/B)!) + aK \lg a - aK \lg(aK/B) + O(K \lg a) \\
&\quad \text{(by } \sum_{i=1}^{B} a_i = |A| = aK\text{)} \\
&= B \lg((N/B)!) - aK \lg(K/B) + O(K \lg a).
\end{aligned}$$

Proof of Lemma 3

We will need:

THEOREM 7 (DILWORTH'S THEOREM [4]). *Let X be a set of n elements, and $\prec$ a partial order over X. If X contains at most w pairwise incomparable elements, then X can be divided into w disjoint partitions $X_1, ..., X_w$, such that elements in each X_i $(1 \le i \le w)$ are pairwise comparable.*

In the sequel, we abbreviate $\mathrm{CP}(\prec_\star, X)$ into $\mathrm{CP}(X)$ for any $X \subseteq \mathcal{S}$. Let $X_1, ..., X_w$ be as defined in the above theorem, and $n_i = |X_i|$ $(1 \le i \le n)$. By Fact 5,

$$|\mathrm{CP}(X)| \le \frac{n!}{\prod_{i=1}^{w} n_i!} \cdot \prod_{i=1}^{w} |\mathrm{CP}(X_i)|.$$

For each $i \in [1, w]$, since the elements in X_i are pairwise comparable, there is only one permutation of X_i that is consistent with $\prec$. Therefore, $|\mathrm{CP}(X_i)| = 1$. Hence,

$$|\mathrm{CP}(X)| \le \frac{n!}{\prod_{i=1}^{w} n_i!}.$$

Taking logarithms at both sides gives

$$\begin{aligned}
&\lg |\mathrm{CP}(X)| \\
&\le \lg(n!) - \sum_{i=1}^{w} \lg(n_i!) \\
&\le n \lg n - \sum_{i=1}^{w} n_i \lg n_i + O(\lg n) \\
&\quad \text{(by Stirling's formula)} \\
&\le n \lg n - n \lg(n/w) + O(\lg n) \\
&\quad \text{(by convexity of function } x \lg x\text{)} \\
&= n \lg w + O(\lg n).
\end{aligned}$$

Simplification of (3)

By substitution of $|\mathcal{S}_1| = ... = |\mathcal{S}_B| = N/B$, we have

$$\begin{aligned}
|\mathrm{CP}(\mathcal{S})| \le (3) &\le \prod_{i=1}^{B} \binom{N/B}{|T_i|} \cdot (N/B - |T_i|)! \cdot |\mathrm{CP}(T_i)| \\
&= \prod_{i=1}^{B} (N/B)! \cdot |\mathrm{CP}(T_i)| / |T_i|!.
\end{aligned}$$

Taking logarithms at both sides gives

$$\begin{aligned}
&\lg |\mathrm{CP}(\mathcal{S})| \\
&\le B \lg((N/B)!) - \sum_{i=1}^{B} (\lg(|T_i|!) - \lg |\mathrm{CP}(T_i)|) \\
&\le B \lg((N/B)!) - \sum_{i=1}^{B} (\lg(|T_i|!) - |T_i| \lg b - O(\lg |T_i|)) \\
&\quad \text{(by (2))} \\
&\le B \lg((N/B)!) - \sum_{i=1}^{B} (|T_i| \lg |T_i| - |T_i| \lg b - O(|T_i|)) \\
&\quad \text{(by Stirling's formula)} \\
&\le B \lg((N/B)!) - (|T| \lg(|T|/B) - |T| \lg b - O(|T|)) \\
&\quad \text{(by convexity of function } x \lg x\text{)} \\
&= B \lg((N/B)!) - |T| \lg(|T|/(bB)) + O(|T|).
\end{aligned}$$

Proof of Lemma 5

For $K \geq 2$, any algorithm must spend $\Omega(N/B)$ I/Os reading the entire dataset (by our right-grounded argument in Section 3). Hence, the lemma holds for $K \leq 16M/B$, in which case our target lower bound is $\Omega(N/B)$.

Let us also get rid of another simple case $K > N/B$. Under this condition, we can sort $\mathcal{S}$ by (i) running a precise K-partitioning algorithm on $\mathcal{S}$, and (ii) sorting the elements inside each partition. Since the size of each partition is $N/K \leq B$, step (ii) takes only $O(N/B)$ I/Os; and the running time of the algorithm is dominated by step (i). Consequently, the time for precise K-partitioning cannot be smaller than the lower bound $\Omega(\frac{N}{B} \lg_{M/B} \frac{N}{B})$ of sorting.

The rest of the proof focuses on $K \in [16M/B, N/B]$, where our target lower bound becomes $\Omega(\frac{N}{B} \lg_{M/B} K)$. We also assume that the algorithm uses at most $N \lg N$ blocks (otherwise, the algorithm has already written more blocks than the target lower bound). We consider the memory $\mathcal{M}$ as a multiset of size M, and the i-th block $\mathcal{B}_i$ as a multiset of size B for each $i \in [1, N \lg N]$. Each element in $\mathcal{M}$ or $\mathcal{B}_i$ is either an element of $\mathcal{S}$ or *nil*.

At any moment during the execution of an algorithm, the *machine state* can be represented as a sequence $\mathcal{M}, \mathcal{B}_1, ..., \mathcal{B}_{N \lg N}$. For each $t \geq 0$, let $\mathrm{MS}(t)$ be the set of all the possible machine states that can be generated by the algorithm after t I/Os. Using a standard argument (see [1]), we know:

LEMMA 7. *For any $t \geq 0$, $|\mathrm{MS}(t)| \leq (2N \lg N \cdot \binom{M}{B})^t$.*

Let H be the worst-case I/O cost of the algorithm. By Lemma 7, during the execution of the algorithm, at most $|\mathrm{MS}(H)| \leq \left(2N \lg N \cdot \binom{M}{B}\right)^H$ machine states can be generated. The following lemma shows that, the algorithm has to be able to generate a large number of machine states in order to ensure its correctness.

LEMMA 8. $|\mathrm{MS}(H)| \geq \frac{N!}{(N/K)!^K}$.

PROOF. Let $P_1, ..., P_K$ be the partitions output by the algorithm. Set $g = \lceil N/KB \rceil$, i.e., g is the minimum number of blocks required to store a partition. We can safely assume that, when the algorithm finishes, $P_1, ..., P_K$ are stored on the disk as follows: for each $i \in [1, K]$, blocks $\mathcal{B}_{(i-1)g+1}, ..., \mathcal{B}_{ig}$ store all and only the elements of P_i (any algorithm can be slightly modified to satisfy this assumption by spending additional $O(N/B)$ I/Os). Therefore, whenever the output partitions are different, the algorithm's final machine state is also different. There are $N!/(N/K)!^K$ different ways to divide $\mathcal{S}$ into K partitions of size N/K. This completes the proof. □

By combining Lemmas 7 and 8, we have

$$\left(2N \lg N \cdot \binom{M}{B}\right)^H \geq \frac{N!}{(N/K)!^K}.$$

Taking logarithms at both sides and then applying Stirling's formula yield

$$H\left(\lg N + B \lg \frac{M}{B}\right) = \Omega(N \lg K),$$

When $\lg N \leq B \lg \frac{M}{B}$, it follows that

$$H = \Omega\left(\frac{N}{B} \lg_{M/B} K\right).$$