The Input/Output Complexity of Sorting and Related Problems

Alok Aggarwal and Jeffery Scott Vitter

6.886

February 21, 2019
Overview

- I/O Model

- Tight bounds for worst and average case for the following problems
  1. Sorting
  2. Permuting
  3. FFT/permutation networks
  4. Matrix transposition

- Analysis of I/O bounds for algorithms
I/O Model

Memory model:

- Memory is divided into internal memory and secondary storage/disk.
I/O Model

Memory model:

- Memory is divided into internal memory and secondary storage/disk.
- *Time* is defined as the number of I/O operations. *Space* is measured as the amount of memory needed.
I/O Model

Memory model:

- Memory is divided into internal memory and secondary storage/disk.
- *Time* is defined as the number of I/O operations. *Space* is measured as the amount of memory needed.
- CPU calculation can be done only on data in the memory, but any such calculation is charged with no cost.
I/O Model

Memory model:

- Memory is divided into internal memory and secondary storage/disk.
- *Time* is defined as the number of I/O operations. *Space* is measured as the amount of memory needed.
- CPU calculation can be done only on data in the memory, but any such calculation is charged with no cost.
- Accessing data in the memory is also for free.
I/O Model

Memory model:

- Memory is divided into internal memory and secondary storage/disk.
- *Time* is defined as the number of I/O operations. *Space* is measured as the amount of memory needed.
- CPU calculation can be done only on data in the memory, but any such calculation is charged with no cost.
- Accessing data in the memory is also for free.
Parameters

- \( N \) = Number of records to sort

Some bounds:

\[ 1 \leq B \leq M < N \]
\[ 1 \leq P \leq \lfloor \frac{M}{B} \rfloor \]
Parameters

- \( N \) = Number of records to sort
- \( M \) = Number of records that fit in internal memory

Some bounds:

\[ 1 \leq B \leq M < N \]
\[ 1 \leq P \leq \lfloor M / B \rfloor \]
Parameters

- \( N \) = Number of records to sort
- \( M \) = Number of records that fit in internal memory
- \( B \) = Number of records that can be transferred in a single block

Some bounds:
\[ 1 \leq B \leq M < N \]
\[ 1 \leq P \leq \left\lfloor \frac{M}{B} \right\rfloor \]
Parameters

- \( N \) = Number of records to sort
- \( M \) = Number of records that fit in internal memory
- \( B \) = Number of records that can be transferred in a single block
- \( P \) = Number of blocks that can be transferred concurrently

Some bounds:

\[ 1 \leq B \leq M < N \]

\[ 1 \leq P \leq \lfloor \frac{M}{B} \rfloor \]
Parameters

- $N =$ Number of records to sort
- $M =$ Number of records that fit in internal memory
- $B =$ Number of records that can be transferred in a single block
- $P =$ Number of blocks that can be transferred concurrently

Some bounds:

$$1 \leq B \leq M < N$$
Parameters

- \( N \) = Number of records to sort
- \( M \) = Number of records that fit in internal memory
- \( B \) = Number of records that can be transferred in a single block
- \( P \) = Number of blocks that can be transferred concurrently

Some bounds:

\[
1 \leq B \leq M < N
\]

\[
1 \leq P \leq \lfloor M/B \rfloor
\]
Parameters

- **N** = Number of records to sort
- **M** = Number of records that fit in internal memory
- **B** = Number of records that can be transferred in a single block
- **P** = Number of blocks that can be transferred concurrently

Some bounds:

\[ 1 \leq B \leq M < N \]

\[ 1 \leq P \leq \left\lfloor \frac{M}{B} \right\rfloor \]
I/O Model

Memory model:

- Memory is divided into internal memory (holds $M$ records) and secondary storage/disk (>> $M$)
I/O Model

Memory model:

- Memory is divided into internal memory (holds $M$ records) and secondary storage/disk (>> $M$).
- We can think of both together as a single contiguous array, where internal memory goes from $x[1], x[2], \cdots, x[M]$ and secondary storage from $x[M + 1], x[M + 2], \cdots$. 
Sorting

- Problem: The internal memory is empty, and the $N$ records reside at the beginning of the disk.
- Goal: The internal memory is empty, and the $N$ records reside at the beginning of the disk in sorted nondecreasing order by their key values.
- Some notation: We denote the $N$ records as $R_1, R_2, \ldots, R_N$. At the start of the problem, $x[M + i] = R_i \ \forall 1 \leq i \leq N$. 
Permutation

- Problem: The internal memory is empty, and the N records reside at the beginning of the disk (same as sorting).
- Goal: The internal memory is empty, and the N records reside at the beginning of the disk. The key values of the N records form a permutation of \( \{1, 2, \ldots, N\} \).
- What is the relationship between sorting and permuting?
External Merge Sort

Assume $P = 1$, $3B \leq M$.

1. Start with internal memory empty, $N/B$ block in disk.
External Merge Sort

Assume $P = 1, 3B \leq M$.

1. Start with internal memory empty, $N/B$ block in disk.

2. For each block, load it into internal memory and sort the keys within the block. We now have $N/B$ partitions that are each internally sorted.
External Merge Sort

Assume $P = 1$, $3B \leq M$.

1. Start with internal memory empty, $N/B$ block in disk.
2. For each block, load it into internal memory and sort the keys within the block. We now have $N/B$ partitions that are each internally sorted.
3. Now we begin merging partitions
External Merge Sort

Assume $P = 1, 3B \leq M$.

1. Start with internal memory empty, $N/B$ block in disk.
2. For each block, load it into internal memory and sort the keys within the block. We now have $N/B$ partitions that are each internally sorted.
3. Now we begin merging partitions
   3.1 Assume we have $P_1$ and $P_2$. We want the get the $B$ first elements in $P_1 \cup P_2$
   3.2 This is clearly contained in $P_1[1 : B] \cup P_2[1 : B]$.
   3.3 How do we get the next $B$ elements?
Assume $P = 1$, $3B \leq M$.

- Runtime:
  \[ O((N/B)\log_{M/B}(N/B)) \]
External Merge Sort

Assume $P = 1$, $3B \leq M$.

- Runtime:
  
  \[ O((N/B)\log_{M/B}(N/B)) \]

- The total number of levels of merges is $O(\log_{M/B}(N/B))$. 
External Merge Sort

Assume $P = 1$, $3B \leq M$.

- Runtime:
  \[ O\left(\frac{N}{B}\log_{\frac{M}{B}}\left(\frac{N}{B}\right)\right) \]

- The total number of levels of merges is $O\left(\log_{\frac{M}{B}}\left(\frac{N}{B}\right)\right)$.

- Each level, we do $O\left(\frac{N}{B}\right)$ work (in terms of I/Os).
Assume $P = 1$, $3B \leq M$.

- Runtime:
  \[ O\left((N/B)\log_{M/B}(N/B)\right) \]

- The total number of levels of merges is $O\left(\log_{M/B}(N/B)\right)$.
- Each level, we do $O((N/B))$ work (in terms of I/Os).
Permuting: Two ways

- How do we permute elements that are all stored in RAM?
- What about with secondary storage?
- Approach 1: Reuse the algorithm used for the RAM model.
Permuting: Two ways

- How do we permute elements that are all stored in RAM?
- What about with secondary storage?
- Approach 1: Reuse the algorithm used for the RAM model. Number of I/Os $O(N)$
Permuting: Two ways

- How do we permute elements that are all stored in RAM?
- What about with secondary storage?
- Approach 1: Reuse the algorithm used for the RAM model. Number of I/Os \( O(N) \)
- Approach 2: External sort: I/O’s \( O\left(\frac{N}{B}\log_{M/B}\left(\frac{N}{B}\right)\right) \)
- Can we do \( O\left(\frac{N}{B}\right) \)?
I/O Model

A few assumptions about the I/O Model

▶ Records are indivisible (no bit manipulations)
I/O Model

A few assumptions about the I/O Model

- Records are indivisible (no bit manipulations)
- All I/Os are "simple": when transferring a record, it is written to an location, then deleted from the original location.
I/O Model

A few assumptions about the I/O Model

- Records are indivisible (no bit manipulations)
- All I/Os are "simple": when transferring a record, it is written to an location, then deleted from the original location.
- The disk is divided into blocks called "tracks": locations $x[M + (k - l)B + 1], x[M + (k - l)B + 2], ..., x[M + kB]$ comprise the $k$th track.
I/O Model

A few assumptions about the I/O Model

▶ Records are indivisible (no bit manipulations)
▶ All I/Os are ”simple”: when transferring a record, it is written to an location, then deleted from the original location.
▶ The disk is divided into blocks called ”tracks”: locations $x[M + (k - l)B + 1], x[M + (k - l)B + 2], \ldots, x[M + kB]$ comprise the $k$th track.
▶ Each I/O performed transfers $B$ records that come from the same track.
Main results - Sorting

Theorem

The average and worst case number of I/Os for sorting $N$ records is

$$\theta \left( \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right)$$

If $M = 2$, $B = P = 1$ we get the well known $O(N \log(N))$ bound on comparison sort.
Main results - Permutation

Theorem

The average and worst case number of I/Os for permuting $N$ records is

$$\theta \left( \min \left( \frac{N}{P}, \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right) \right)$$
Main results - Permutation

Theorem

The average and worst case number of I/Os for permuting $N$ records is

$$\theta \left( \min \left( \frac{N}{P}, \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right) \right)$$

- The second term is the same as the bound for sorting.
Main results - Permutation

Theorem

The average and worst case number of I/Os for permuting $N$ records is

$$\theta \left( \min \left( \frac{N}{P}, \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right) \right)$$

- The second term is the same as the bound for sorting.
- When $M$ and $B$ are small, we are essentially doing the naive permutation method described before.
Main results - Permutation Proof

Theorem

The average and worst case number of I/Os for permuting \( N \) records is

\[
\theta \left( \min \left( \frac{N}{P}, \frac{N}{PB \log(1 + N/B)} \right) \right)
\]

We say a permutation \( p_1, p_2, \ldots, p_N \) of the \( N \) records can be generated at time \( t \) if there is some sequence of \( t \) I/OS such that after the I/OS all records are in correct permuted order in disk:

\( x[i] = R_{p_k} \) and \( x[j] = R_{p_{k+1}} \) imply \( i < j \ \forall i, j, k \).
1. Strip out key values and sort in memory.
2. Permute records based off key order.
Main results - Permutation Proof

Theorem

The average and worst case number of I/Os for permuting \( N \) records is

\[
\theta \left( \min \left( \frac{N}{P}, \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right) \right)
\]

Proof Idea: Bound the number of possible permutations that can be generated by \( t \) I/Os. Choose smallest \( t \) such that the number of possible permutations is \( \geq N! \)
fft

Background: Fourier Series

\[
\hat{f}(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\theta) e^{i n \theta} d\theta =: \langle f, e_n \rangle.
\]

\[
f(\theta) \sim \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{i n \theta} = \sum_{n=-\infty}^{\infty} \langle f, e_n \rangle e_n(\theta).
\]
FFT

Background: Fourier Series

\[ \hat{f}(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\theta) e_n(\theta) \, d\theta =: \langle f, e_n \rangle. \]

\[ f(\theta) \approx \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{in\theta} = \sum_{n=-\infty}^{\infty} \langle f, e_n \rangle e_n(\theta). \]

DFT:

\[ \hat{v}(m) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} v(n) e^{-2\pi i mn/N}. \]
FFT

Background: Fourier Series

\[ \hat{f}(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\theta) \overline{e_n(\theta)} \, d\theta =: \langle f, e_n \rangle. \]

\[ f(\theta) \sim \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{i n \theta} = \sum_{n=-\infty}^{\infty} \langle f, e_n \rangle e_n(\theta). \]

DFT:

\[ \hat{v}(m) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} v(n) e^{-2\pi i m n / N}. \]

DFT in Matrix Form:

\[
\begin{pmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
\vdots \\
y_{n-1}
\end{pmatrix}
= 
\begin{pmatrix}
1 & 1 & 1 & 1 & \cdots & 1 \\
1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\
1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\
1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-1)}
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\vdots \\
a_{n-1}
\end{pmatrix}.
\]
FFT

Butterfly Diagram
FFT

Butterfly Diagram
Main results - FFT

Theorem

The average and worst case number of I/Os for computing the $N$-input FFT digraph is

$$\theta \left( \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right).$$
Matrix Transposition

Problem: A $p \times q$ matrix $A = (A_{i,j})$ of $N = pq$ records stored in row-major order on disk. The internal memory is empty.
Matrix Transposition

- Problem: A $p \times q$ matrix $A = (A_{i,j})$ of $N = pq$ records stored in row-major order on disk. The internal memory is empty.
- Goal: The internal memory is empty, and the transposed matrix $A^T$ resides on disk in row-major order.
Matrix Transposition

- Problem: A $p \times q$ matrix $A = (A_{i,j})$ of $N = pq$ records stored in row-major order on disk. The internal memory is empty.
- Goal: The internal memory is empty, and the transposed matrix $A^T$ resides on disk in row-major order.
- Reminder:

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\]
Main results - Matrix Transposition

Theorem

The number of I/OS required to transpose a $p \times q$ matrix stored in row-major order, is

$$\theta \left( \frac{N \log \left( \min 1 + N/B, M, 1 + \min(p, q) \right)}{PB} \frac{\log(1 + M/B)}{\log(1 + M/B)} \right).$$
Main results - Matrix Transposition

**Theorem**

*The number of I/OS required to transpose a $p \times q$ matrix stored in row-major order, is*

$$\theta \left( \frac{N \log (\min 1 + N/B, M, 1 + \min(p, q))}{PB} \right) \log(1 + M/B)$$

▷ Also a similar problem to permuting/sorting when $B$ is large.
Main results - Matrix Transposition

Theorem

The number of I/OS required to transpose a $p \times q$ matrix stored in row-major order, is

$$\theta \left( \frac{N \log \left( \min 1 + N/B, M, 1 + \min(p, q) \right)}{PB \log(1 + M/B)} \right)$$

Also a similar problem to permuting/sorting when $B$ is large.

$$\theta \left( \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right)$$
Algorithm: Merge Sort

Assume $P = 1$, $3B \leq M$. Reminder

- Runtime:

$$O((N/B) \log_{M/B}(N/B))$$
Algorithm: Merge Sort

Assume \( P = 1, \ 3B \leq M \). Reminder

- **Runtime:**
  \[
  O((N/B) \log_{M/B}(N/B))
  \]

- The total number of levels of merges is \( O(\log_{M/B}(N/B)) \).
Algorithm: Merge Sort

Assume $P = 1$, $3B \leq M$. Reminder

- Runtime:
  \[ O\left(\frac{N}{B}\log_{\frac{M}{B}}\left(\frac{N}{B}\right)\right) \]

- The total number of levels of merges is $O\left(\log_{\frac{M}{B}}\left(\frac{N}{B}\right)\right)$.
- Each level, we do $O\left(\frac{N}{B}\right)$ work (in terms of I/Os).
Algorithm: Merge Sort

Assume $P = 1$, $3B \leq M$. Reminder

- Runtime:
  \[ O((N/B)\log_{M/B}(N/B)) \]

- The total number of levels of merges is $O(\log_{M/B}(N/B))$.
- Each level, we do $O((N/B)$ work (in terms of I/Os).
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort.
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, \ldots, p_f$ such that there are $O(N/f)$ elements in each partition. **Takes $O(N/B)$**
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, \ldots, p_f$ such that there are $O(N/f)$ elements in each partition. **Takes $O(N/B)$**

2. Partition elements in $S$ into buckets based on pivots: $S_1, S_2, \ldots, S_f$
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, .., p_f$ such that there are $O(N/f)$ elements in each partition. **Takes $O(N/B)$**

2. Partition elements in $S$ into buckets based on pivots: $S_1, S_2, \cdots, S_f$

3. Recurse to sort within each bucket.
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, \ldots, p_f$ such that there are $O(N/f)$ elements in each partition. **Takes $O(N/B)$**

2. Partition elements in $S$ into buckets based on pivots: $S_1, S_2, \ldots, S_f$

3. Recurse to sort within each bucket.

4. If $S \leq B$, sort in internal memory.
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, \ldots, p_f$ such that there are $O(N/f)$ elements in each partition. **Takes $O(N/B)$**
2. Partition elements in $S$ into buckets based on pivots: $S_1, S_2, \ldots, S_f$
3. Recurse to sort within each bucket.
4. If $S \leq B$, sort in internal memory.

Recursion:

$$T(N) \leq \sum_{i=1}^{f} T(|S_i|) + O(N/B)$$
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, \ldots, p_f$ such that there are $O(N/f)$ elements in each partition. Takes $O(N/B)$
2. Partition elements in $S$ into buckets based on pivots: $S_1, S_2, \ldots, S_f$
3. Recurse to sort within each bucket.
4. If $S \leq B$, sort in internal memory.

Recursion:

$$T(N) \leq \sum_{i=1}^{f} T(|S_i|) + O(N/B)$$

Runtime:

$$O\left(\left(\frac{N}{B}\right)\log_{M/B}(N/B)\right)$$
Algorithm: Distribution Sort

Analogous to quick-sort. Let $S$ be the set of elements you wish to sort

1. Let $f = \sqrt{\frac{M}{B}}$. Find a set of pivots $p_1, p_2, \ldots, p_f$ such that there are $O(N/f)$ elements in each partition. Takes $O(N/B)$

2. Partition elements in $S$ into buckets based on pivots: $S_1, S_2, \ldots, S_f$

3. Recurse to sort within each bucket.

4. If $S \leq B$, sort in internal memory.

Recursion:

$$T(N) \leq \sum_{i=1}^{f} T(|S_i|) + O(N/B)$$

Runtime:

$$O \left( (N/B) \log_{M/B} (N/B) \right)$$

Compare with theoretical bound:

$$\theta \left( \frac{N \log(1 + N/B)}{B \log(1 + M/B)} \right)$$
Distribution Sort

How do we find our pivots in $O(N/B)$? Intuition: Median of Medians

1. Let $t = N/M$
Distribution Sort

How do we find our pivots in $O(N/B)$? Intuition: Median of Medians

1. Let $t = N/M$
2. Divide $S$ into $t$ groups: $G_1, \ldots, G_t$, each with $M$ elements.
Distribution Sort

How do we find our pivots in $O(N/B)$? Intuition: Median of Medians

1. Let $t = N/M$
2. Divide $S$ into $t$ groups: $G_1, \ldots, G_t$, each with $M$ elements.
3. Load each $G_i$ into memory + sort.
Distribution Sort

How do we find our pivots in \( O(N/B) \)? Intuition: Median of Medians

1. Let \( t = N/M \)
2. Divide \( S \) into \( t \) groups: \( G_1, \cdots, G_t \), each with \( M \) elements.
3. Load each \( G_i \) into memory + sort.
4. After sorting, collect one out of every \( f \) elements of \( G_i \). Call these your representatives.
Distribution Sort

How do we find our pivots in $O(N/B)$? Intuition: Median of Medians

1. Let $t = N/M$
2. Divide $S$ into $t$ groups: $G_1, \cdots, G_t$, each with $M$ elements.
3. Load each $G_i$ into memory + sort.
4. After sorting, collect one out of every $f$ elements of $G_i$. Call these your representatives.
5. Let $G$ be the set of representatives for every $G_i$. There are $O\left(\frac{M \cdot N}{f \cdot M}\right) = O\left(\frac{N}{f}\right)$ elements.
Distribution Sort

How do we find our pivots in $O(N/B)$? Intuition: Median of Medians

1. Let $t = N/M$
2. Divide $S$ into $t$ groups: $G_1, \ldots, G_t$, each with $M$ elements.
3. Load each $G_i$ into memory + sort.
4. After sorting, collect one out of every $f$ elements of $G_i$. Call these your representatives.
5. Let $G$ be the set of representatives for every $G_i$. There are $O\left(\frac{M}{f} \frac{N}{M}\right) = O\left(\frac{N}{f}\right)$ elements.
6. For $i \in [1, f]$ let $p_i$ be the $i\left\lceil \frac{N}{f^2} \right\rceil$ smallest element in $G$. 

How do we find that? $k$-selection! Takes $O\left(\frac{N}{B}\right)$.

Total cost of all $k$-selections is $O\left(\frac{N}{fB} f\right) = O\left(\frac{N}{B}\right)$.
How do we find our pivots in $O(N/B)$? Intuition: Median of Medians

1. Let $t = N/M$
2. Divide $S$ into $t$ groups: $G_1, \ldots, G_t$, each with $M$ elements.
3. Load each $G_i$ into memory + sort.
4. After sorting, collect one out of every $f$ elements of $G_i$. Call these your representatives.
5. Let $G$ be the set of representatives for every $G_i$. There are $O(\frac{M N}{f M}) = O(N/f)$ elements.
6. For $i \in [1, f]$ let $p_i$ be the $i\lceil \frac{N}{f^2} \rceil$ smallest element in $G$.
7. How do we find that?

k-selection! Takes $O(\frac{N}{B})$.

Total cost of all $k$-selections is $O(\frac{N f B}{f}) = O(\frac{N}{B})$. 
Distribution Sort

How do we find our pivots in $O(N/B)$? Inuition: Median of Medians

1. Let $t = N/M$

2. Divide $S$ into $t$ groups: $G_1, \cdot \cdot \cdot , G_t$, each with $M$ elements.

3. Load each $G_i$ into memory + sort.

4. After sorting, collect one out of every $f$ elements of $G_i$. Call these your representatives.

5. Let $G$ be the set of representatives for every $G_i$. There are $O\left(\frac{M}{f} \frac{N}{M}\right) = O\left(\frac{N}{f}\right)$ elements.

6. For $i \in [1, f]$ let $p_i$ be the $i\left\lceil \frac{N}{f^2} \right\rceil$ smallest element in $G$.

7. How do we find that? k-selection! Takes $O(N/B)$. 
Distribution Sort

How do we find our pivots in \( O(N/B) \)? Intuition: Median of Medians

1. Let \( t = N/M \)
2. Divide \( S \) into \( t \) groups: \( G_1, \ldots, G_t \), each with \( M \) elements.
3. Load each \( G_i \) into memory + sort.
4. After sorting, collect one out of every \( f \) elements of \( G_i \). Call these your representatives.
5. Let \( G \) be the set of representatives for every \( G_i \). There are \( O(M f N M) = O(N/f) \) elements.
6. For \( i \in [1, f] \) let \( p_i \) be the \( i\lceil \frac{N}{f^2} \rceil \) smallest element in \( G \).
7. How do we find that? k-selection! Takes \( O(N/B) \).
8. Total cost of all k-selections is \( O(\frac{N}{fB} f) = O(N/B) \).
Algorithm: Permuting

- Permuting is a special case of sorting.
- Unless $B$, $M$ is small: then use naive method.
Summary

- **Sorting**
  \[ \theta \left( \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right) \]

- **Permuting**
  \[ \theta \left( \min \left( \frac{N \log(1 + N/B)}{PB \log(1 + M/B)}, \frac{N}{P} \right) \right) \]

- **FFT**
  \[ \theta \left( \frac{N \log(1 + N/B)}{PB \log(1 + M/B)} \right) \]

- **Matrix Transposition**
  \[ \theta \left( \frac{N \log \left( \min 1 + N/B, M, 1 + \min(p, q) \right)}{PB \log(1 + M/B)} \right) \]