

Algorithmen für die Speicherhierarchie

Sortieren und Permutieren: Obere und untere Schranken [Aggarwal, Vitter 1988]

Riko Jacob

Lehrstuhl für Effiziente Algorithmen
Fakultät für Informatik
Technische Universität München

Vorlesung 22. Oktober 2007



Technische Universität München



k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$



k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$



k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$



k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

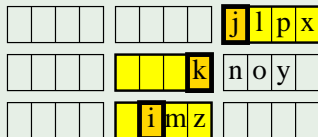


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

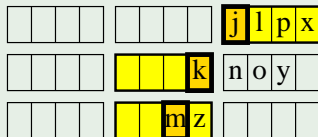


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

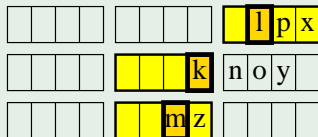


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

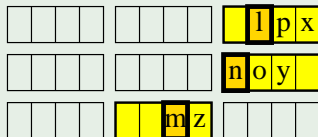


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

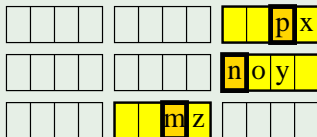


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

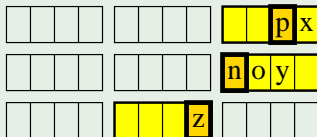


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$

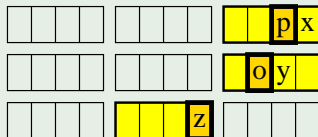


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$



a b e f | g h i j | k l m n | [] [] [] []

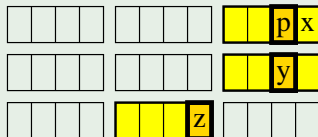


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$



a b e f | g h i j | k l m n | o

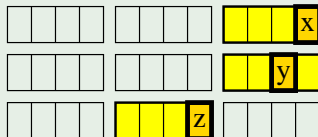


k-Wege Merge

Verschmelzen und I/O

Beispiel

$$k = 3, B = 4, M = 4B$$



a b e f g h i j k l m n o p



Analyse k -Wege MergeSort

klassisches Divide-and-Conquer

Theorem

MergeSort nutzt $\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right)$ I/Os

Basis:

Je M Elemente

Laden; Sortieren; Speichern;

I/Os

Runs

$\mathcal{O}(N/B)$

N/M

Verschmelzen:

je $k = M/B - 1$ sortierte Listen

$\mathcal{O}(N/B)$

$\times 1/k$

Verschmelzungstiefe: $\mathcal{O}\left(\log_{\frac{M}{B}} N/M\right)$



I/O-Bedarf Sortieren

Theorem

Sortieren benötigt höchstens

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

I/O-Operationen (für $N > M^2/B$).

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:

$M/B = 50.000$, $N/M = 40.000$;

einmal scannen um Runs zu erzeugen, einmal Verschmelzen



I/O-Bedarf Sortieren

Theorem

Sortieren benötigt höchstens

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

I/O-Operationen (für $N > M^2/B$).

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:

$M/B = 50.000$, $N/M = 40.000$;

einmal scannen um Runs zu erzeugen, einmal Verschmelzen



I/O-Bedarf Sortieren

Theorem

Sortieren benötigt höchstens

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

I/O-Operationen (für $N > M^2/B$).

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:

$M/B = 50.000$, $N/M = 40.000$;

einmal scannen um Runs zu erzeugen, einmal Verschmelzen

Geht es besser?



Technische Universität München



Sortieren: unter Schranke

Annahmen

- Vergleichsbasierter Algorithmus
- Elemente können nur bewegt, kopiert, gelöscht, und verglichen werden
- Eingabe in den ersten N/B Blöcken der Platte
- $M \geq 2B$

Gegenspieler

- Gegenspieler gibt Antworten auf Vergleiche
- Konsistenz: Wiederholung, Transitivität
- “Reagiert” auf Aktivitäten des Algorithmus

unter Schranke: Strategie des Gegenspielers

Strategie des Gegenspielers

- S_ℓ Menge von Permutationen, die nach ℓ I/O-Operationen noch konsistent mit bisherigen Antworten ist (weitere Antworten verkleinern diese Menge)
- Maximiere in jedem Schritt die Größe von S_ℓ (Kardinalität)

Argument

Wenn der Gegenspieler X Möglichkeiten hat, dann muss es eine geben, die mindestens $|S_\ell|/X$ Permutationen weiterhin erlaubt.

(Großes X schwächt dieses Argument für den Gegenspieler)



Zählen

Definition von X

X ist die Anzahl von verschiedenen Antworten wie **alle** Elemente im Hauptspeicher angeordnet sind.

Operation	lese frischen Block ¹	lese Block	schreibe Block
X	$\binom{M}{B} B!$	$\binom{M}{B}$	1

(1): höchstens N/B solche Operationen

$$S_0 = N!$$

$$S_\ell \leq \frac{N!}{\binom{M}{B}^\ell (B!)^{\frac{N}{B}}}$$

$$\text{Am Ende } S_\ell = 1$$

$$1 \geq \frac{N!}{\binom{M}{B}^\ell (B!)^{\frac{N}{B}}}$$

Zählen

Definition von X

X ist die Anzahl von verschiedenen Antworten wie **alle** Elemente im Hauptspeicher angeordnet sind.

Operation	lese frischen Block ¹	lese Block	schreibe Block
X	$\binom{M}{B} B!$	$\binom{M}{B}$	1

(1): höchstens N/B solche Operationen

$$S_0 = N!$$

$$S_\ell \leq \frac{N!}{\binom{M}{B}^\ell (B!)^{\frac{N}{B}}}$$

$$\text{Am Ende } S_\ell = 1$$

$$1 \geq \frac{N!}{\binom{M}{B}^\ell (B!)^{\frac{N}{B}}}$$

Auflösen nach ℓ

$$\binom{M}{B}^\ell (B!)^{\frac{N}{B}} \geq N!$$

$$\ell \log \binom{M}{B} + \frac{N}{B} \log(B!) \geq \log(N!)$$

$$3\ell B \log(M/B) + N \log B \geq N(\log N - \log e)$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{B \log(M/B)}$$

Lemma

- 1 $\log(x!) \geq x(\log x - \log e)$
- 2 $\log(x!) \leq x \log x$
- 3 $\log \binom{x}{y} \leq 3y \log \frac{x}{y}$ für $x \geq 2y$

Resultat

$$\ell = \Omega \left(\frac{N}{B} \log \frac{M}{B} \right)$$



Auflösen nach ℓ

$$\binom{M}{B}^\ell (B!)^{\frac{N}{B}} \geq N!$$

$$\ell \log \binom{M}{B} + \frac{N}{B} \log(B!) \geq \log(N!)$$

$$3\ell B \log(M/B) + N \log B \geq N(\log N - \log e)$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{B \log(M/B)}$$

Lemma

- 1 $\log(x!) \geq x(\log x - \log e)$
- 2 $\log(x!) \leq x \log x$
- 3 $\log \binom{x}{y} \leq 3y \log \frac{x}{y}$ für $x \geq 2y$

Resultat

$$\ell = \Omega \left(\frac{N}{B} \log \frac{M}{B} \right)$$



Auflösen nach ℓ

$$\binom{M}{B}^\ell (B!)^{\frac{N}{B}} \geq N!$$

$$\ell \log \binom{M}{B} + \frac{N}{B} \log(B!) \geq \log(N!)$$

$$3\ell B \log(M/B) + N \log B \geq N(\log N - \log e)$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{B \log(M/B)}$$

Lemma

- 1 $\log(x!) \geq x(\log x - \log e)$
- 2 $\log(x!) \leq x \log x$
- 3 $\log \binom{x}{y} \leq 3y \log \frac{x}{y}$ für $x \geq 2y$

Resultat

$$\ell = \Omega \left(\frac{N}{B} \log \frac{M}{B} \frac{N}{B} \right)$$

Lemma (Stirling)

Lemma

- 1 $\log(x!) \geq x(\log x - \log e)$
- 2 $\log(x!) \leq x \log x$
- 3 $\log \binom{x}{y} \leq 3y \log \frac{x}{y}$ für $x \geq 2y$

Stirling

$$x! = \sqrt{2\pi x} \cdot (x/e)^x \cdot (1 + \mathcal{O}(1/12x))$$

Beweis

- 1 $\log(x!) \geq \log \sqrt{2\pi x} + x(\log x - \log e) + o(1)$
- 2 $\log(x!) \leq \log(x^x) = x \log x$
- 3 $\log \binom{x}{y} \leq \log \frac{x^y}{(y/e)^y} = y(\log(x/y) + \log e)$



Lemma (Stirling)

Lemma

- 1 $\log(x!) \geq x(\log x - \log e)$
- 2 $\log(x!) \leq x \log x$
- 3 $\log \binom{x}{y} \leq 3y \log \frac{x}{y}$ für $x \geq 2y$

Stirling

$$x! = \sqrt{2\pi x} \cdot (x/e)^x \cdot (1 + \mathcal{O}(1/12x))$$

Beweis

- 1 $\log(x!) \geq \log \sqrt{2\pi x} + x(\log x - \log e) + o(1)$
- 2 $\log(x!) \leq \log(x^x) = x \log x$
- 3 $\log \binom{x}{y} \leq \log \frac{x^y}{(y/e)^y} = y(\log(x/y) + \log e)$



Lemma (Stirling)

Lemma

- 1 $\log(x!) \geq x(\log x - \log e)$
- 2 $\log(x!) \leq x \log x$
- 3 $\log \binom{x}{y} \leq 3y \log \frac{x}{y}$ für $x \geq 2y$

Stirling

$$x! = \sqrt{2\pi x} \cdot (x/e)^x \cdot (1 + \mathcal{O}(1/12x))$$

Beweis

- 1 $\log(x!) \geq \log \sqrt{2\pi x} + x(\log x - \log e) + o(1)$
- 2 $\log(x!) \leq \log(x^x) = x \log x$
- 3 $\log \binom{x}{y} \leq \log \frac{x^y}{(y/e)^y} = y(\log(x/y) + \log e)$



Zusammenfassung Sortieren

Definiere

$$n = N/B$$
$$m = M/B$$

Benutze

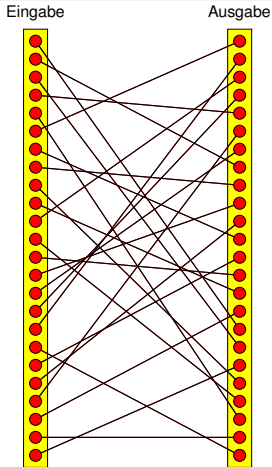
$$1 + \log_{\frac{M}{B}}(x) = \log_{\frac{M}{B}}\left(x \frac{M}{B}\right)$$

Komplexität Sortieren

$$\Theta(N/B \log_{M/B}(N/B))$$
$$= \Theta(n \log_m(n))$$
$$=: \text{sort}(N)$$



Aufgabe Permutieren



- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen

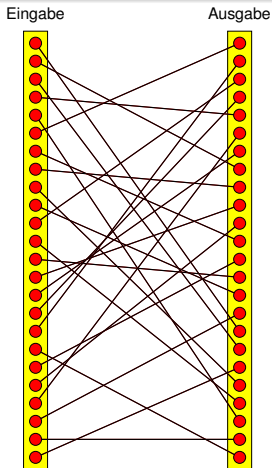
- $\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right)$ I/O-Ops.

durch MergeSort

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:
 $M/B = 50.000$, $N/M = 40.000$;
zweimal scannen
Faktor 500.

Aufgabe Permutieren



- $\mathcal{O}(N)$ CPU-Operationen

- $\mathcal{O}(N)$ I/O-Operationen

-

$$\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:

$M/B = 50.000$, $N/M = 40.000$;

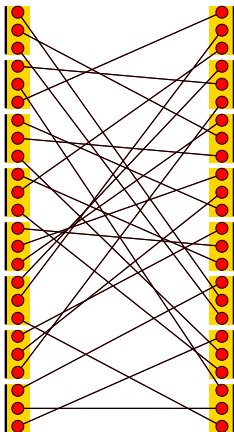
zweimal scannen

Faktor 500.

Aufgabe Permutieren

Eingabe

Ausgabe



- $\mathcal{O}(N)$ CPU-Operationen

- $\mathcal{O}(N)$ I/O-Operationen

-

$$\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Beispiel

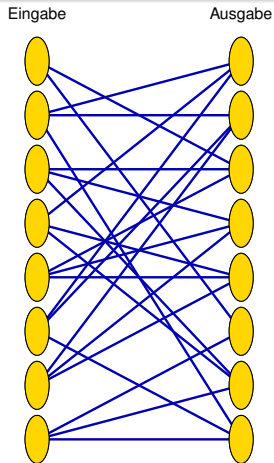
$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:

$M/B = 50.000$, $N/M = 40.000$;

zweimal scannen

Faktor 500.

Aufgabe Permutieren



- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen



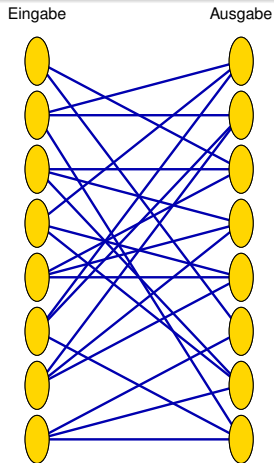
$$\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:
 $M/B = 50.000$, $N/M = 40.000$;
zweimal scannen
Faktor 500.

Aufgabe Permutieren



- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen
-

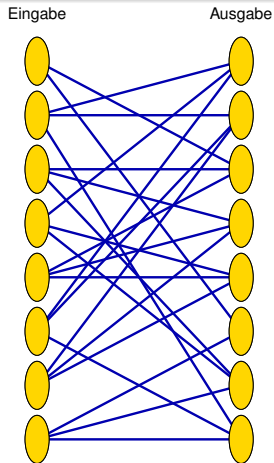
$$\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:
 $M/B = 50.000$, $N/M = 40.000$;
zweimal scannen
Faktor 500.

Aufgabe Permutieren



- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen
-

$$\mathcal{O}\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Beispiel

$M = 50\text{Mega}$, $B = 1.000$, $N = 200\text{Giga}$:
 $M/B = 50.000$, $N/M = 40.000$;
zweimal scannen
Faktor 500.

Untere Schranke Permutieren

Theorem ([Aggarwal, Vitter 1988])

*Angenommen, Elemente werden nur
kopiert, bewegt, oder gelöscht,
dann gibt es eine Permutation die*

$$\Omega \left(\min \left\{ N, \frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M} \right\} \right)$$

I/O-Operationen benötigt.

*Eine zufällig gewählte Permutation (uniform) benötigt mit hoher
Wahrscheinlichkeit diese Anzahl von I/O-Operationen*



Untere Schranke Permutieren

Theorem ([Aggarwal, Vitter 1988])

*Angenommen, Elemente werden nur
kopierte, bewegt, oder gelöscht,
dann gibt es eine Permutation die*

$$\Omega \left(\min \left\{ N, \frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M} \right\} \right)$$

I/O-Operationen benötigt.

*Eine zufällig gewählte Permutation (uniform) benötigt mit hoher
Wahrscheinlichkeit diese Anzahl von I/O-Operationen*



Kopieren und Löschen unnötig

Programmanalyse

Die betrachteten Programme sind deterministisch. Daher ist klar, wann und wie ein geschriebenes Element wieder verwendet wird.

Transformation

Arbeite nur mit der einen Kopie eines Elements, die letztendlich in der Ausgabe verwendet wird.
⇒ kein kopieren oder löschen.

Spur

Falls nur bewegt wird, beschreiben Hauptspeicher und Festplatte genau eine Permutation der Elemente.

Kopieren und Löschen unnötig

Programmanalyse

Die betrachteten Programme sind deterministisch. Daher ist klar, wann und wie ein geschriebenes Element wieder verwendet wird.

Transformation

Arbeite nur mit der einen Kopie eines Elements, die letztendlich in der Ausgabe verwendet wird.
⇒ kein kopieren oder löschen.

Spur

Falls nur bewegt wird, beschreiben Hauptspeicher und Festplatte genau eine Permutation der Elemente.

Kopieren und Löschen unnötig

Programmanalyse

Die betrachteten Programme sind deterministisch. Daher ist klar, wann und wie ein geschriebenes Element wieder verwendet wird.

Transformation

Arbeite nur mit der einen Kopie eines Elements, die letztendlich in der Ausgabe verwendet wird.
⇒ kein kopieren oder löschen.

Spur

Falls nur bewegt wird, beschreiben Hauptspeicher und Festplatte genau eine Permutation der Elemente.

Beweis: untere Schranke Permutieren

Normalisierung

Hauptspeicher in endgültiger (relativer) Reihenfolge

$S_\ell :=$ Anzahl der Permutationen die mit ℓ I/O-Operationen erreicht werden können

Anzahl X neuer Permutationen

Operation	lese frischen Block ¹	lese Block	schreibe Block
X	$N \binom{M}{B} B!$	$N \binom{M}{B}$	N

untere Schranke

$$\left(\ell \binom{M+B}{B} \right)^\ell B! \frac{N}{B} \geq N!$$

Beweis: untere Schranke Permutieren

Normalisierung

Hauptspeicher in endgültiger (relativer) Reihenfolge

$S_\ell :=$ Anzahl der Permutationen die mit ℓ I/O-Operationen erreicht werden können

Anzahl X neuer Permutationen

Operation	lese frischen Block ¹	lese Block	schreibe Block
X	$N \binom{M}{B} B!$	$N \binom{M}{B}$	N

untere Schranke

$$\left(\ell \binom{M+B}{B} \right)^\ell B! \frac{N}{B} \geq N!$$

Beweis: untere Schranke Permutieren

Normalisierung

Hauptspeicher in endgültiger (relativer) Reihenfolge

$S_\ell :=$ Anzahl der Permutationen die mit ℓ I/O-Operationen erreicht werden können

Anzahl X neuer Permutationen

Operation	lese frischen Block ¹	lese Block	schreibe Block
X	$N \binom{M}{B} B!$	$N \binom{M}{B}$	N

untere Schranke

$$\left(\ell \binom{M+B}{B} \right)^\ell B! \frac{N}{B} \geq N!$$

Beweis: untere Schranke Permutieren

Normalisierung

Hauptspeicher in endgültiger (relativer) Reihenfolge

$S_\ell :=$ Anzahl der Permutationen die mit ℓ I/O-Operationen erreicht werden können

Anzahl X neuer Permutationen

Operation	lese frischen Block ¹	lese Block	schreibe Block
X	$N \binom{M}{B} B!$	$N \binom{M}{B}$	N

untere Schranke

$$\left(\ell \binom{M+B}{B} \right)^\ell B! \frac{N}{B} \geq N!$$

Auflösen nach ℓ

$$\left(\binom{M}{B} \cdot \ell \right)^\ell (B!)^{\frac{N}{B}} \geq N!$$

$$\ell \log \left(\binom{M}{B} 2N \right) + \frac{N}{B} \log(B!) \geq \log(N!)$$

$$3\ell (B \log(M/B) + \log 2N) + N \log B \geq N(\log N - \log e)$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{B \log(M/B) + \log 2N}$$

$$B \log(M/B) \geq \log 2N$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{2B \log(M/B)}$$

$$\ell = \Omega(\text{sort}(N))$$

$$B \log(M/B) \leq \log 2N$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{2 \log 2N}$$

$$\ell = \Omega(N) \text{ wegen } B < \log N$$

Auflösen nach ℓ

$$\left(\binom{M}{B} \cdot \ell \right)^\ell (B!)^{\frac{N}{B}} \geq N!$$

$$\ell \log \left(\binom{M}{B} 2N \right) + \frac{N}{B} \log(B!) \geq \log(N!)$$

$$3\ell (B \log(M/B) + \log 2N) + N \log B \geq N(\log N - \log e)$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{B \log(M/B) + \log 2N}$$

$$B \log(M/B) \geq \log 2N$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{2B \log(M/B)}$$

$$\ell = \Omega(\text{sort}(N))$$

$$B \log(M/B) \leq \log 2N$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{2 \log 2N}$$

$$\ell = \Omega(N) \text{ wegen } B < \log N$$

Auflösen nach ℓ

$$\left(\binom{M}{B} \cdot \ell \right)^\ell (B!)^{\frac{N}{B}} \geq N!$$

$$\ell \log \left(\binom{M}{B} 2N \right) + \frac{N}{B} \log(B!) \geq \log(N!)$$

$$3\ell (B \log(M/B) + \log 2N) + N \log B \geq N(\log N - \log e)$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{B \log(M/B) + \log 2N}$$

$$B \log(M/B) \geq \log 2N$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{2B \log(M/B)}$$

$$\ell = \Omega(\text{sort}(N))$$

$$B \log(M/B) \leq \log 2N$$

$$3\ell \geq \frac{N(\log N - \log e - \log B)}{2 \log 2N}$$

$$\ell = \Omega(N) \text{ wegen } B < \log N$$

Komplexität Permutieren

Theorem ([Aggarwal, Vitter 1988])

*Angenommen, Elemente werden nur
kopiert, bewegt, oder gelöscht,
dann gibt es eine Permutation die*

$$\Omega \left(\min \left\{ N, \frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M} \right\} \right)$$

I/O-Operationen benötigt.

*Eine zufällig gewählte Permutation (uniform) benötigt mit hoher
Wahrscheinlichkeit diese Anzahl von I/O-Operationen*



Komplexität Permutieren

Theorem ([Aggarwal, Vitter 1988])

*Angenommen, Elemente werden nur
kopiert, bewegt, oder gelöscht,
dann gibt es eine Permutation die*

$$\Omega \left(\min \left\{ N, \frac{N}{B} \log_{\frac{M}{B}} \frac{N}{M} \right\} \right)$$

I/O-Operationen benötigt.

*Eine zufällig gewählte Permutation (uniform) benötigt mit hoher
Wahrscheinlichkeit diese Anzahl von I/O-Operationen*

