

Übung 6 – Algorithmen II

Sebastian Lamm, Demian Hespe – lamm@kit.edu, hespe@kit.edu
http://algo2.iti.kit.edu/AlgorithmenII_WS18.php

Institut für Theoretische Informatik - Algorithmik II

```
    result = current_weight;  
    return true;  
}  
  
for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){  
    const Edge & edge = graph.getEdge( eid );  
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ); )  
    if( edge.forward ){  
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ); )  
        Weight new_weight = edge.weight + current_weight;  
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );  
        if( !priority_queue.isReached( edge.target ) ){  
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ); )  
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ); )  
            priority_queue.push( edge.target, new_weight );  
        } else {  
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){  
                COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_NODES ); )  
                priority_queue.decreaseKey( edge.target, new_weight );  
            }  
        }  
    }  
}
```

Randomisierte Algorithmen

- Grundlagen
- Verifikation von Matrix-Matrix Multiplikation
- Coupon Collector Problem

Speichermodell

- Parallel Disk Model
- Speicherlatenzen
- Blockgrößen

Ergänzendes Material: I/O-effizientes Design

- Basistechniken
- *externe* Prioritätslisten
- *externe* Sortieren

Randomisierte Algorithmen

■ *Las Vegas Algorithmus*

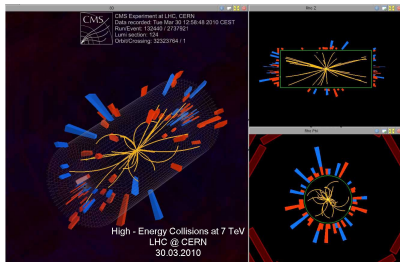
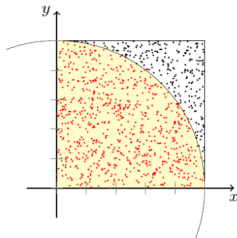
- immer korrekte/optimale Lösung
- Laufzeit ist Zufallsvariable
→ erwartete Laufzeit $\mathbb{E}[T]$
- Bsp.: Quicksort

■ *Monte Carlo Algorithmus*

- falsche/suboptimale Lösung möglich
→ mit Wahrscheinlichkeit p
- deterministische Laufzeit
- Bsp.: Miller-Rabin Primzahltest,
nicht vorbereiteter Student beim *multiple-choice* Test

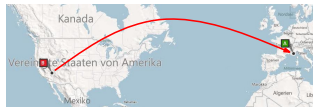
■ Monte Carlo Simulation

- nicht zu verwechseln mit *Monte Carlo Algorithmus*!
- Zufallsexperimente sehr oft ausführen (Gesetz der großen Zahlen)
- je länger / öfter ausgeführt, desto bessere Ergebnisse
 - Alternative zur analytischen Lösung
 - Verteilungsfunktion unbekannter Zufallsvariablen
 - Nachbildung komplexer Prozesse



Randomisierte Algorithmen

Las Vegas $\rightarrow$ Monte Carlo



geg: Las Vegas Algorithmus mit erwarteter Laufzeit $\mathbb{E}[T] = f(n)$

ges: Monte Carlo Algorithmus mit Laufzeit $\mathcal{O}(f(n))$, Fehlerrate p

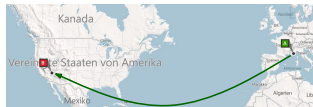
Idee: Abbruch nach Zeit $\alpha f(n)$

- Ausgabe FALSCH, wenn Algorithmus abgebrochen wurde
- $\mathbb{P}[T > \alpha f(n)] \leq 1/\alpha$ (Markov Ungleichung)

$\rightarrow$ Monte Carlo Algorithmus mit Laufzeit $\alpha f(n)$ und Fehlerrate $p = 1/\alpha$

Randomisierte Algorithmen

Monte Carlo → Las Vegas



geg: Monte Carlo Algorithmus mit Laufzeit $\mathcal{O}(f(n))$, Fehlerrate p ,
Korrektheit in $\mathcal{O}(g(n))$ prüfbar

ges: Las Vegas Algorithmus mit erwarteter Laufzeit $\mathbb{E}[T]$

Idee: Wiederhole MC bis korrekte Ergebnis gefunden

■ Laufzeit $T \leq i \cdot \mathcal{O}(f(n) + g(n))$ (i Schritte benötigt)

■ $\mathbb{E}[T] \leq \mathbb{E}[i] \cdot \mathcal{O}(f(n) + g(n))$

$$\begin{aligned} \blacksquare \mathbb{E}[i] &= \sum_{k=1}^{\infty} k \cdot p^{k-1} \cdot (1-p) = \frac{1}{1-p} \\ &\text{(nach } \sum_{k=1}^{\infty} kx^k = x/(1-x)^2, x < 1) \end{aligned}$$

→ Las Vegas Algorithmus mit erwarteter Laufzeit $\mathbb{E}[T] \leq \frac{\mathcal{O}(f(n)+g(n))}{1-p}$

Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

- deterministisch:

- berechne $X \cdot Y$ und vergleiche mit Z
→ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

- randomisiert:

- Wähle 0-1 Vektor $r = (r_1, \dots, r_n)$ zufällig
- Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH
→ Laufzeit $\mathcal{O}(n^2)$

Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

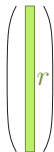
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

■ deterministisch:

- berechne $X \cdot Y$ und vergleiche mit Z
→ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

■ randomisiert:

- Wähle 0-1 Vektor $r = (r_1, \dots, r_n)$ zufällig
- Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH
→ Laufzeit $\mathcal{O}(n^2)$



Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

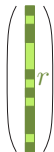
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

■ deterministisch:

- berechne $X \cdot Y$ und vergleiche mit Z
→ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

■ randomisiert:

- Wähle 0-1 Vektor $r = (r_1, \dots, r_n)$ zufällig
- Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH
→ Laufzeit $\mathcal{O}(n^2)$



Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

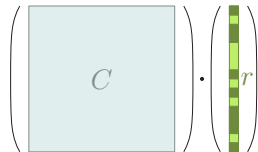
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

■ deterministisch:

- berechne $X \cdot Y$ und vergleiche mit Z
→ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

■ randomisiert:

- Wähle 0-1 Vektor $r = (r_1, \dots, r_n)$ zufällig
- Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH
→ Laufzeit $\mathcal{O}(n^2)$


$$\begin{pmatrix} C \end{pmatrix} \cdot \begin{pmatrix} r \end{pmatrix}$$

Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

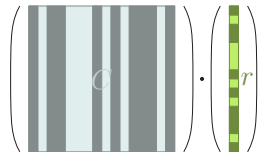
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

■ deterministisch:

- berechne $X \cdot Y$ und vergleiche mit Z
→ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

■ randomisiert:

- Wähle 0-1 Vektor $r = (r_1, \dots, r_n)$ zufällig
- Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH
→ Laufzeit $\mathcal{O}(n^2)$



Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

Definitionen:

■ $A := XY, B := Z$

Annahme: $A \neq B$; Wann ist dann $Ar = Br$?

■ $\exists i, j : A_{i,j} \neq B_{i,j}$

■ Sei $a := A_i$ und $b := B_i$ (A_i : i'te Zeile von A)

$$\alpha := \sum_{k \neq j} a_k r_k \text{ und } \beta := \sum_{k \neq j} b_k r_k$$

$$\Rightarrow ar = \alpha + a_j r_j \text{ und } br = \alpha + b_j r_j$$

$$\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j)r_j$$

■ Annahme: Werte für $r_{1 \dots j-1, j+1 \dots n}$ bereits festgelegt

→ noch 2 Möglichkeiten für r_j , Gleichheit bei maximal einer

$$\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\alpha - \beta}{a_j - b_j}] \leq \frac{1}{2}$$

→ Fehlerrate $p \leq 0.5$

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

Definitionen:

■ $A := XY, B := Z$

Annahme: $A \neq B$; Wann ist dann $Ar = Br$?

■ $\exists i, j : A_{i,j} \neq B_{i,j}$

■ Sei $a := A_i$ und $b := B_i$ (A_i : i'te Zeile von A)

$$\alpha := \sum_{k \neq j} a_k r_k \text{ und } \beta := \sum_{k \neq j} b_k r_k$$

$$\Rightarrow ar = \alpha + a_j r_j \text{ und } br = \alpha + b_j r_j$$

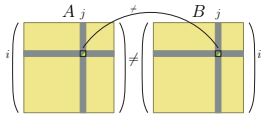
$$\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j)r_j$$

■ Annahme: Werte für $r_{1 \dots j-1, j+1 \dots n}$ bereits festgelegt

→ noch 2 Möglichkeiten für r_j , Gleichheit bei maximal einer

$$\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\alpha - \beta}{a_j - b_j}] \leq \frac{1}{2}$$

→ Fehlerrate $p \leq 0.5$



Randomisierte Algorithmen

Matrix-Matrix Multiplikation

Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

Definitionen:

■ $A := XY, B := Z$

Annahme: $A \neq B$; Wann ist dann $Ar = Br$?

■ $\exists i, j : A_{i,j} \neq B_{i,j}$

■ Sei $a := A_i$ und $b := B_i$ (A_i : i'te Zeile von A)

$$\alpha := \sum_{k \neq j} a_k r_k \text{ und } \beta := \sum_{k \neq j} b_k r_k$$

$$\Rightarrow ar = \alpha + a_j r_j \text{ und } br = \alpha + b_j r_j$$

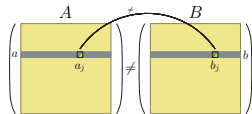
$$\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j)r_j$$

■ Annahme: Werte für $r_{1 \dots j-1, j+1 \dots n}$ bereits festgelegt

→ noch 2 Möglichkeiten für r_j , Gleichheit bei maximal einer

$$\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\alpha - \beta}{a_j - b_j}] \leq \frac{1}{2}$$

→ Fehlerrate $p \leq 0.5$



Randomisierte Algorithmen

Matrix-Matrix Multiplikation

Behauptung: Wenn $XY \neq Z$ dann $\mathbb{P}[XYr = Zr] \leq 0.5$

Definitionen:

■ $A := XY, B := Z$

Annahme: $A \neq B$; Wann ist dann $Ar = Br$?

■ $\exists i, j : A_{i,j} \neq B_{i,j}$

■ Sei $a := A_i$ und $b := B_i$ (A_i : i'te Zeile von A)

$$\alpha := \sum_{k \neq j} a_k r_k \text{ und } \beta := \sum_{k \neq j} b_k r_k$$

$$\Rightarrow ar = \alpha + a_j r_j \text{ und } br = \alpha + b_j r_j$$

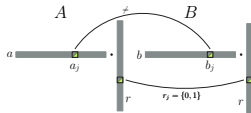
$$\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j)r_j$$

■ Annahme: Werte für $r_1 \dots r_{j-1}, r_{j+1} \dots r_n$ bereits festgelegt

→ noch 2 Möglichkeiten für r_j , Gleichheit bei maximal einer

$$\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\alpha - \beta}{a_j - b_j}] \leq \frac{1}{2}$$

→ Fehlerrate $p \leq 0.5$



Beschleunigung durch *probability boosting*

(nur bei $p \leq 0.5$ schnelle Konvergenz)

- Wiederhole Test k mal mit unterschiedlicher Wahl von r

- Ein Test liefert FALSCH

- $AB \neq C$, fertig

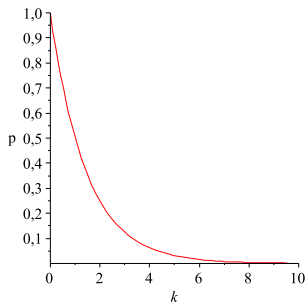
- Alle Tests liefern KORREKT

- *false positive* mit Wahrscheinlichkeit

- $\mathbb{P}[ABr = Cr] \leq 0.5^k$

→ Laufzeit $\mathcal{O}(kn^2)$

(linear längere Laufzeit bei exponentiell weniger Fehler)



Müslipackungen enthalten jeweils eine von n verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

- $X = \#$ Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$, mit $X_i = \#$ Packungen während ich $i - 1$ Karten hatte
 - X_i sind geometrische Zufallsvariablen mit $p_i = 1 - \frac{i-1}{n}$
 - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$ (Linearität des Erwartungswertes)
- $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i}$

Müslipackungen enthalten jeweils eine von n verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

- $X = \#$ Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$, mit $X_i = \#$ Packungen während ich $i - 1$ Karten hatte
 - X_i sind geometrische Zufallsvariablen mit $p_i = 1 - \frac{i-1}{n}$
 - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$ (Linearität des Erwartungswertes)
- $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i}$

Müslipackungen enthalten jeweils eine von n verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

- $X = \#$ Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$, mit $X_i = \#$ Packungen während ich $i - 1$ Karten hatte
 - X_i sind geometrische Zufallsvariablen mit $p_i = 1 - \frac{i-1}{n}$
 - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$ (Linearität des Erwartungswertes)
- $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i}$

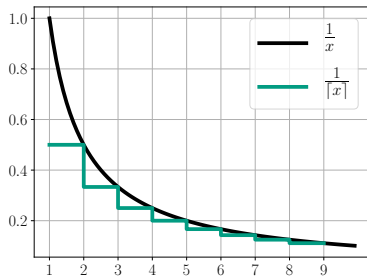
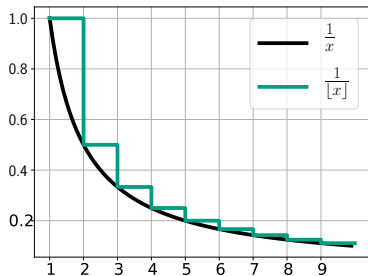
Müslipackungen enthalten jeweils eine von n verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

- $X = \#$ Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$, mit $X_i = \#$ Packungen während ich $i - 1$ Karten hatte
 - X_i sind geometrische Zufallsvariablen mit $p_i = 1 - \frac{i-1}{n}$
 - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$ (Linearität des Erwartungswertes)
- $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i}$

$$\blacksquare H_n = \sum_{i=1}^n \frac{1}{i}$$

$$\ln n = \int_{x=1}^n \frac{1}{x} dx \leq \sum_{i=1}^n \frac{1}{i}$$

$$\sum_{i=2}^n \frac{1}{i} \leq \int_{x=1}^n \frac{1}{x} dx = \ln n$$



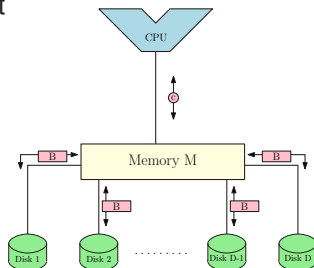
$$\blacksquare \ln n \leq H_n \leq \ln n + 1$$

Müslipackungen enthalten jeweils eine von n verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

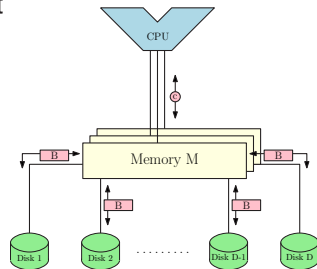
- $X = \#$ Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$, mit $X_i = \#$ Packungen während ich $i - 1$ Karten hatte
 - X_i sind geometrische Zufallsvariablen mit $p_i = 1 - \frac{i-1}{n}$
 - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$ (Linearität des Erwartungswertes)
- $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i} = nH_n \leq n \ln n + n$

Speichermodell

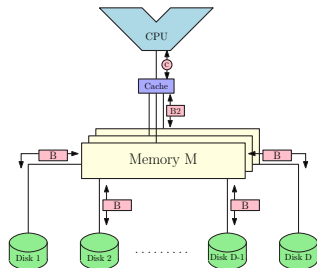
- Vitter und Shriver:
 - Parallel Disk Model (PDM)
- Speicherzugriffe in Blöcken
- Blockzugriffe minimieren → Datenlokalität
- Muster wiederholt sich in Speicherhierarchie immer wieder



- Vitter und Shriver:
 - Parallel Disk Model (PDM)
- Speicherzugriffe in Blöcken
- Blockzugriffe minimieren → Datenlokalität
- Muster wiederholt sich in Speicherhierarchie immer wieder



- Vitter und Shriver:
 - **Parallel Disk Model** (PDM)
- Speicherzugriffe in Blöcken
- Blockzugriffe minimieren → Datenlokalität
- Muster wiederholt sich in Speicherhierarchie immer wieder



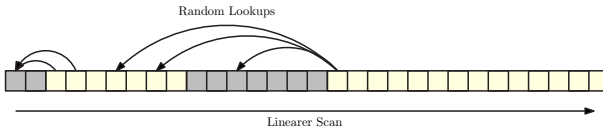
Speichermodell

Latenzen

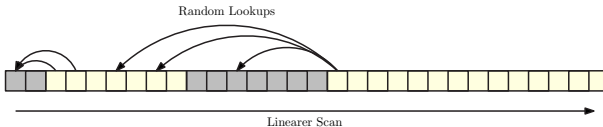
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access	120 ns	6 min
Solid-state disk I/O	50-150 μ s	2-6 days
Rotational disk I/O	1-10 ms	1-12 months
Internet: SF to NYC	40 ms	4 years
Internet: SF to UK	81 ms	8 years
Internet: SF to Australia	183 ms	19 years
OS virtualization reboot	4 s	423 years
SCSI command time-out	30 s	3000 years
Hardware virtualization reboot	40 s	4000 years
Physical system reboot	5 m	32 millenia

- nicht nur relevant bei Disk-I/O
- Beispiel: kürzeste Wege-Bäume auf großen Straßennetzen
(z.B. Europa)
 - Dijkstra (annähernd linear): $\approx 3 - 5$ Sek.
 - Breitensuche (untere Schranke?): ≈ 2 Sek.

- nicht nur relevant bei Disk-I/O
- Beispiel: kürzeste Wege-Bäume auf großen Straßennetzen
(z.B. Europa)
 - Dijkstra (annähernd linear): $\approx 3 - 5$ Sek.
 - Breitensuche (untere Schranke?): ≈ 2 Sek.
 - PHAST (linearer Scan): < 0.2 Sek.



- nicht nur relevant bei Disk-I/O
- Beispiel: kürzeste Wege-Bäume auf großen Straßennetzen
(z.B. Europa)
 - Dijkstra (annähernd linear): $\approx 3 - 5$ Sek.
 - Breitensuche (untere Schranke?): ≈ 2 Sek.
 - PHAST (linearer Scan): < 0.2 Sek.



- **Strukturierter Zugriff** als wichtiges Designprinzip

Einflussfaktoren

- T_{seek} : Positionierungszeit
- W_{max} : maximale Bandbreite

→ Lesedauer: $T = T_{seek} + B/W_{max}$

Optimale Blockgröße

- Beispiel: RAM
 - $T_{seek} = 30$ ns (Zykluszeit)
 - $W_{max} \approx 25$ GB/s
 - Anzahl Blöcke pro Zeile: $R = 128$
 - Ziel: 95% Auslastung der Bandbreite bei sequenziellem Lesen
 - $W = \frac{B}{T} = B / \left(\frac{1}{128} T_{seek} + B/W_{max} \right) \stackrel{!}{=} 0.95 \cdot W_{max}$
 - $B = 0.15 \cdot W_{max} \cdot T_{seek} = 111$ Byte!

Ende!



Feierabend!

Ergänzendes Material

I/O-effizientes Design

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

■ Stack / Queue

■ Zugriffsmuster

- Random Access erwartet $\mathcal{O}(n)$ I/Os
- Linearer Scan $\mathcal{O}(n/B)$ I/Os

■ Sortieren

- $\mathcal{O}\left(\frac{2n}{B} \left(1 + \lceil \log_{M/B} \frac{n}{M} \rceil\right)\right)$ I/Os
- lokale Kriterien
- oft vorbereitend für linearen Scan

■ Prioritätswarteschlangen

- $\approx \text{sort}(n)$ I/Os
- nutzbar als Warteliste: „Speicherzugriff auf später verschieben“

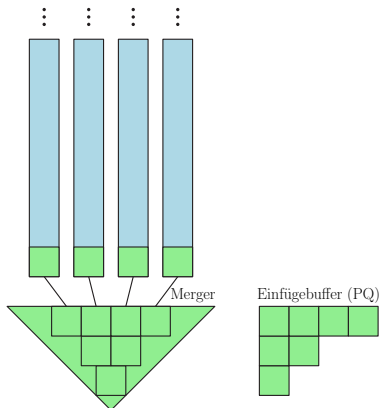
■ Stack / Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



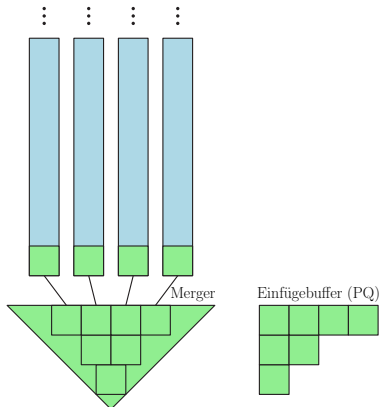
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



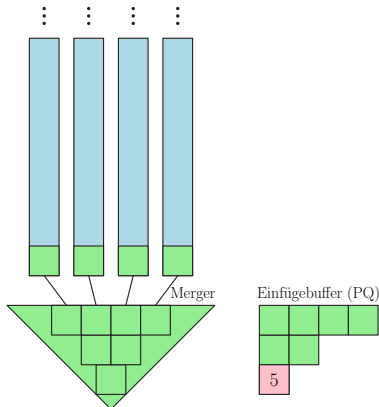
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



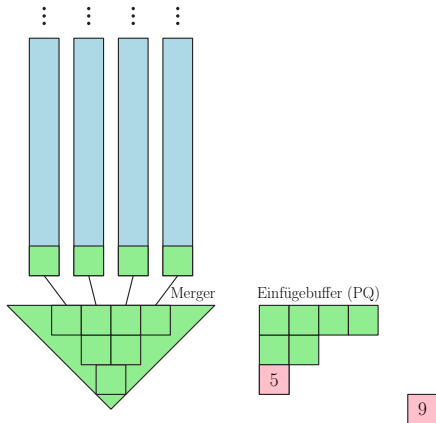
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



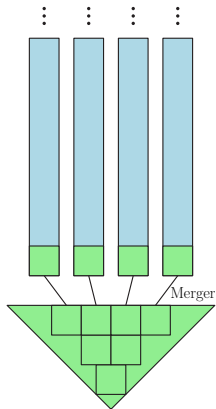
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

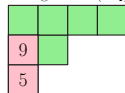
■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



Einfügepuffer (PQ)



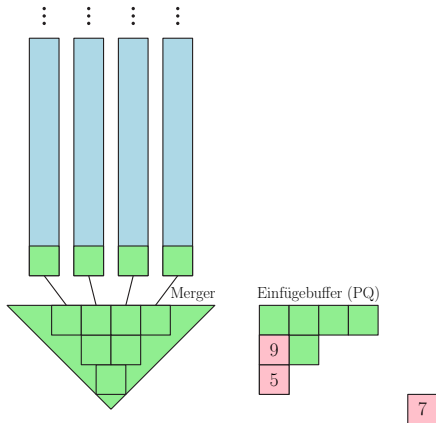
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



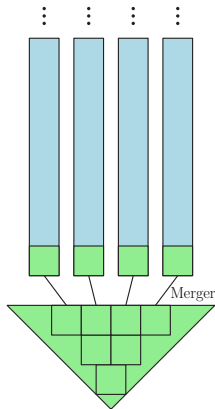
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

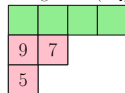
■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



Einfügepuffer (PQ)



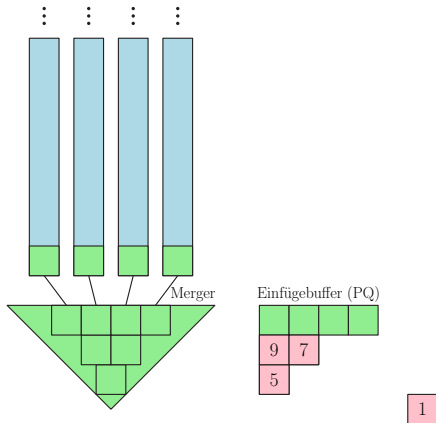
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



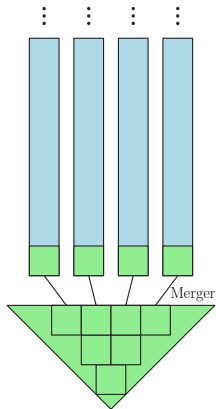
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



Einfügepuffer (PQ)

1			
9	7		
5			

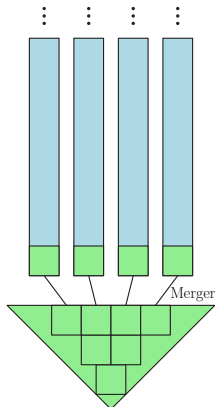
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



Einfügepuffer (PQ)

9			
1	7		
5			

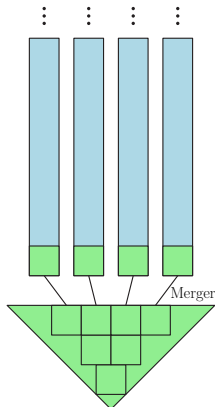
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



Einfügepuffer (PQ)

9			
5	7		
1			

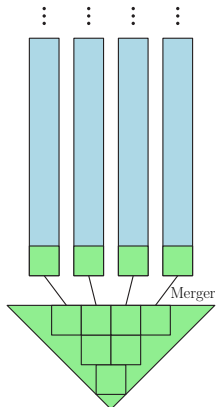
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



Einfügepuffer (PQ)

9	11	8	10
5	7		
1			

2

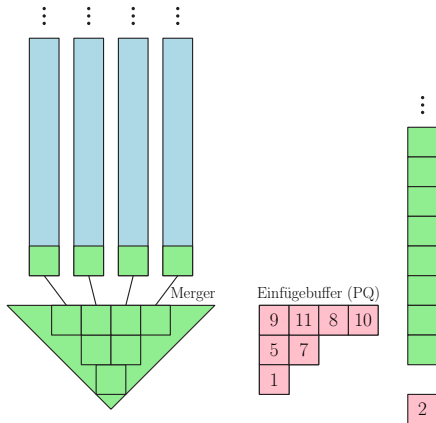
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



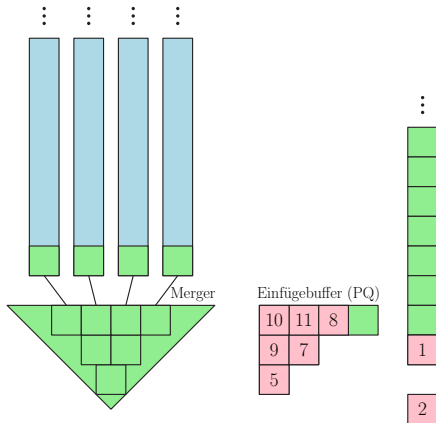
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



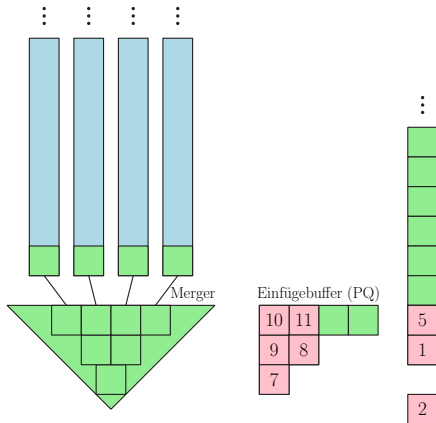
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



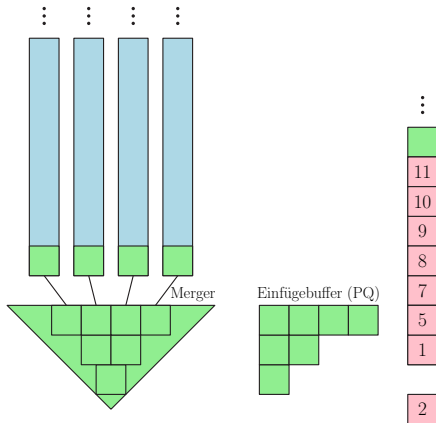
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?

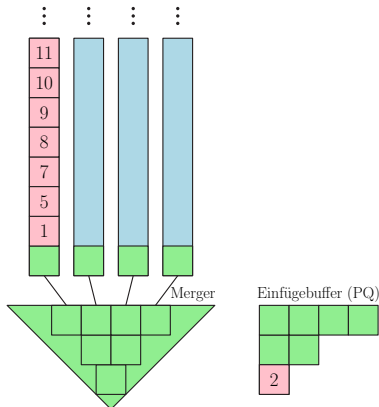


- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



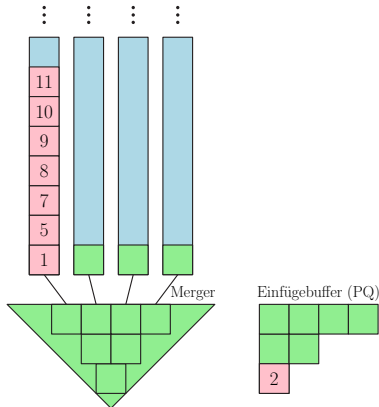
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



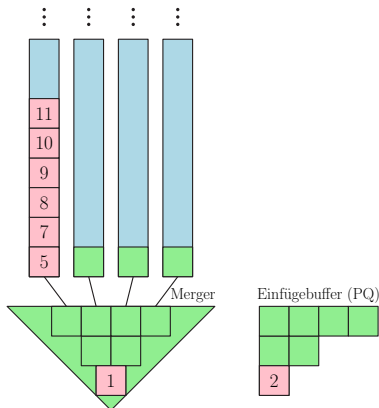
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



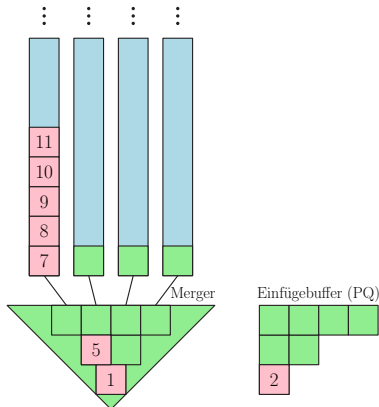
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



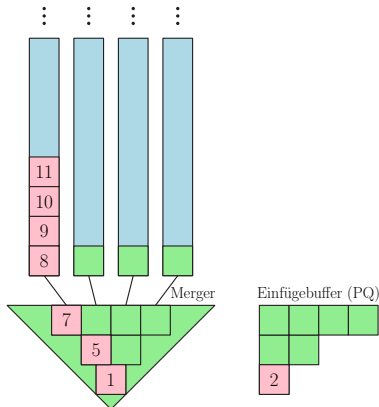
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

■ natürliches Limit:

$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

■ Was tun bei mehr Elementen?



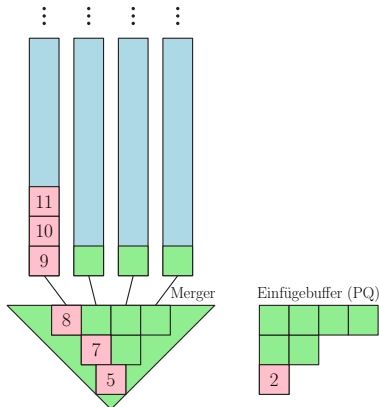
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin

- natürliches Limit:

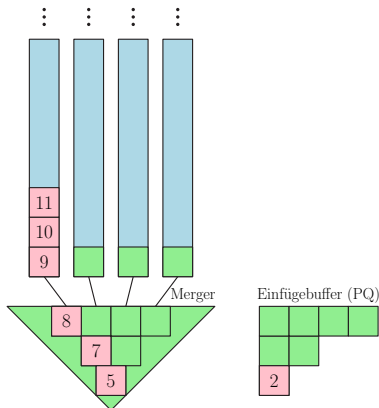
$$\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$$

- Was tun bei mehr Elementen?



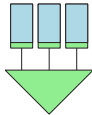
Externe Priority Queue

- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
 - Insert
 - DeleteMin
- natürliches Limit:
 - $\# \text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$
- Was tun bei mehr Elementen?



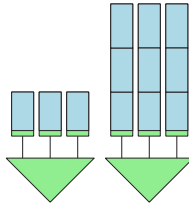
Externe Priority Queue

für sehr große Datenmengen



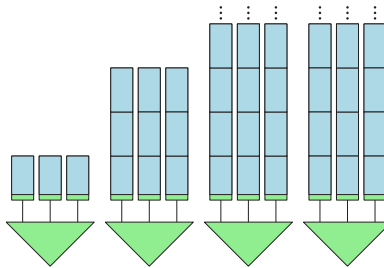
Externe Priority Queue

für sehr große Datenmengen



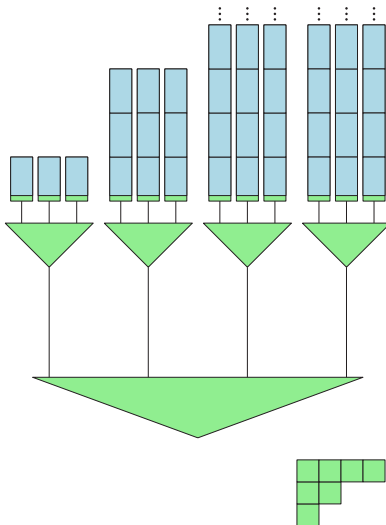
Externe Priority Queue

für sehr große Datenmengen



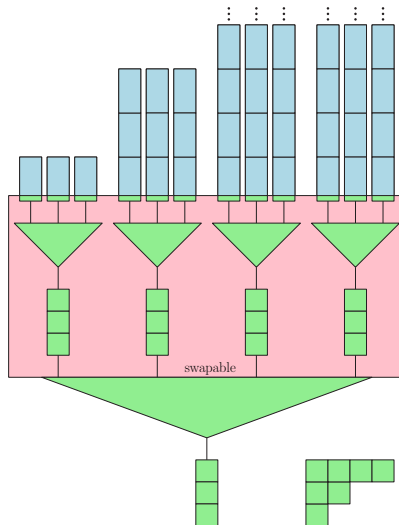
Externe Priority Queue

für sehr große Datenmengen



Externe Priority Queue

für sehr große Datenmengen



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ **Phasen**

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- alle Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

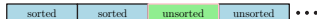
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

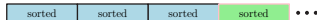
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

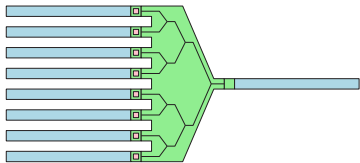
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



Externes Sortieren

Zwei-Phasen Algorithmus

■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

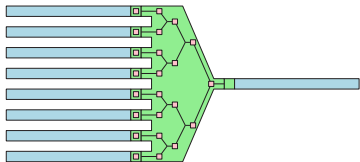
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

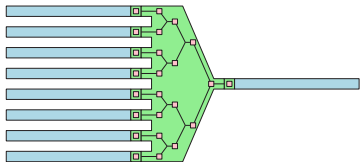
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



Externes Sortieren

Zwei-Phasen Algorithmus

■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

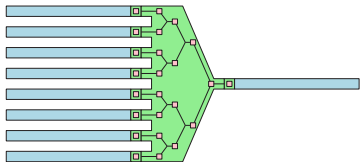
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



Externes Sortieren

Zwei-Phasen Algorithmus

■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

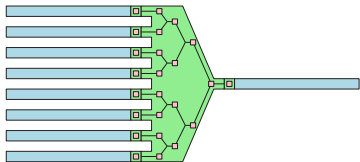
- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$



Externes Sortieren

Zwei-Phasen Algorithmus

■ Run Formation

- Run entspricht einem Teilbereich zu sortierender Daten
- $\lceil \frac{n}{M} \rceil$ Stück, Größe M
- jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
- **alle** Daten einmal lesen + schreiben

■ Multiway Merge

- jede Mischphase liest und schreibt **alle** Daten $\rightarrow \frac{2n}{B}$ I/Os
- Hilfsmittel: Interne PQ für $\frac{M}{B}$ Eingabeströme
- Pro Phase: Gruppen von $\frac{M}{B}$ Runs zu einem Run mergen
 $\rightarrow \lceil \log_{M/B} \frac{n}{M} \rceil$ Phasen

■ Innere Arbeit:

$$\mathcal{O}\left(\frac{n}{M} \cdot M \log M + n \log \frac{M}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

■ I/O Operationen:

$$\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$$

