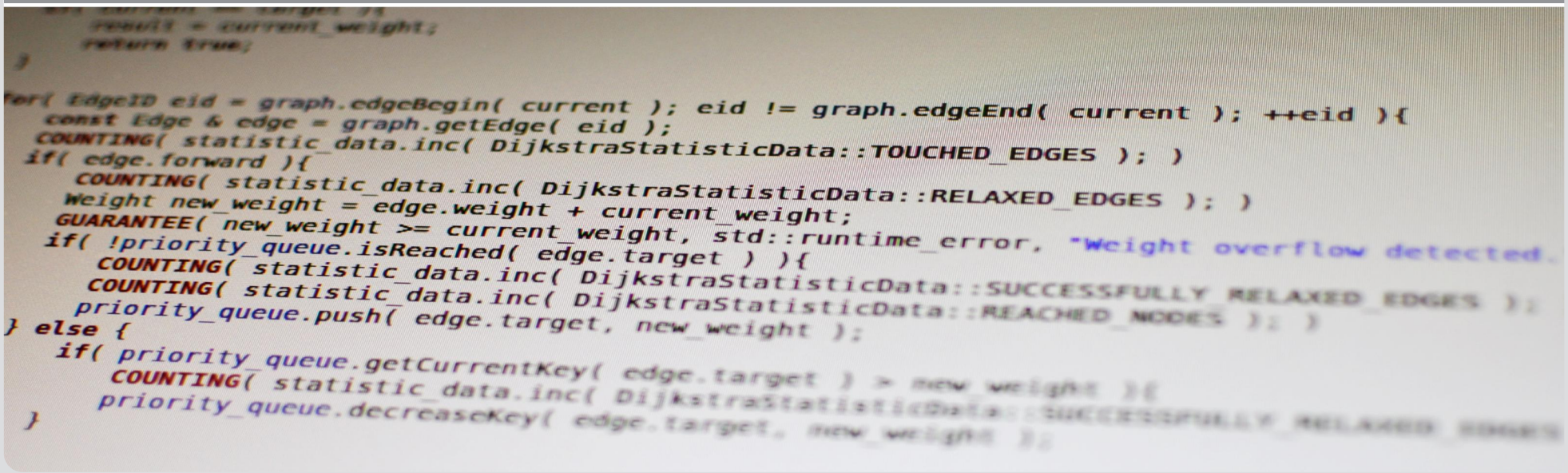
Karlsruher Institut für Technologie

Übung 6 – Algorithmen II

Tobias Heuer, Sebastian Lamm – *tobias.heuer@kit.edu, lamm@kit.edu*

`http://algo2.iti.kit.edu/AlgorithmenII_WS19.php`

Institut für Theoretische Informatik - Algorithmik II



Themenübersicht

Karlsruher Institut für Technologie

Randomisierte Algorithmen

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

Speichermodell

- Speicherlatenzen
- Parallel Disk Model
- Blockgrößen

I/O-effizientes Design

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

Randomisierte Algorithmen

Randomisierte Algorithmen

Übersicht

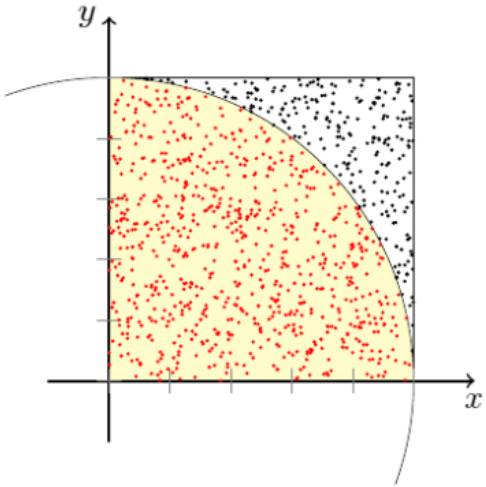
- *Las Vegas Algorithmus*
  - immer korrekte/optimale Lösung
  - Laufzeit ist Zufallsvariable
    $\rightarrow$ erwartete Laufzeit $\mathbb{E}[T]$
  - *Bsp.:* Quicksort

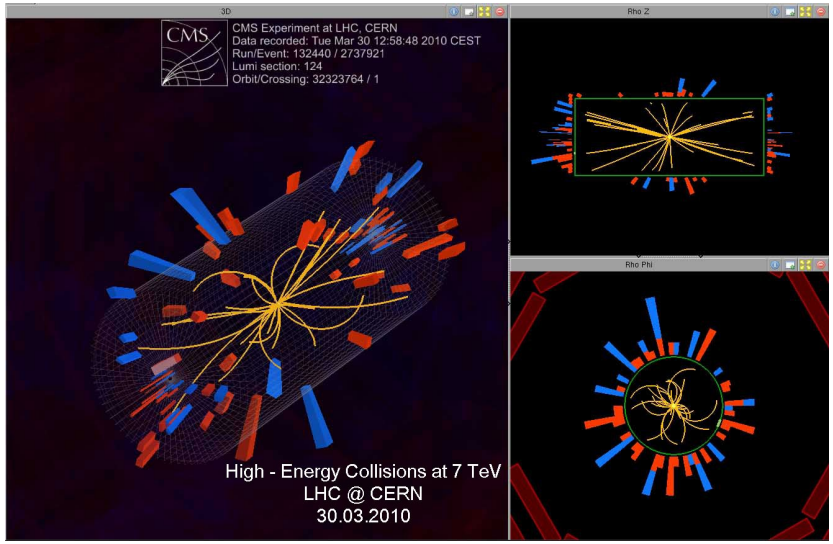
- *Monte Carlo Algorithmus*
  - falsche/suboptimale Lösung möglich
    $\rightarrow$ mit Wahrscheinlichkeit p
  - deterministische Laufzeit
  - *Bsp.:* Miller-Rabin Primzahltest,

Randomisierte Algorithmen

Monte Carlo Simulation

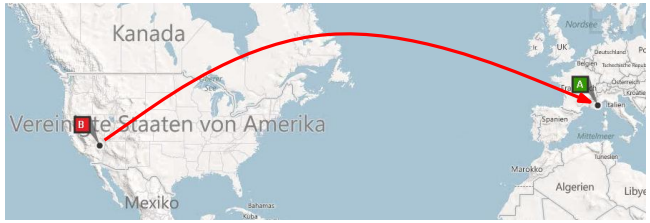
- *Monte Carlo Simulation*
  - nicht zu verwechseln mit *Monte Carlo Algorithmus*!
  - Zufallsexperimente sehr oft ausführen (Gesetz der großen Zahlen)
  - je länger / öfter ausgeführt, desto bessere Ergebnisse
    - Alternative zur analytischen Lösung
    - Verteilungsfunktion unbekannter Zufallsvariablen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

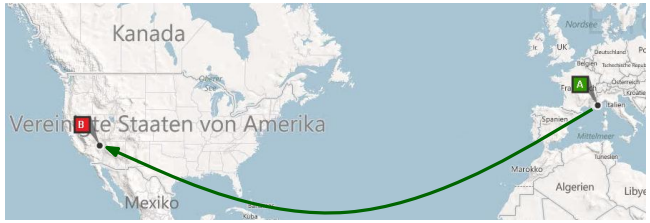
Idee: Abbruch nach Zeit $\alpha f(n)$

- 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)
- $\mathbb{E}[T] \leq \mathbb{E}[i] \cdot \mathcal{O}(f(n) + g(n))$
  - $\mathbb{E}[i] = \sum_{k=1}^{\infty} k \cdot p^{k-1} \cdot (1-p) = \frac{1}{1-p}$
    (nach $\sum_{k=1}^{\infty} kx^k = x/(1-x)^2,\ x < 1$)

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

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

  $\rightarrow$ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

- randomisiert:
  - Wähle 0-1 Vektor $r = (r_1, \ldots, r_n)$ zufällig
  - Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH

  $\rightarrow$ Laufzeit $\mathcal{O}(n^2)$

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

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

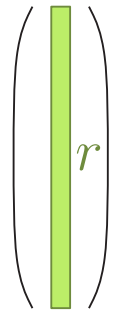
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

- deterministisch:
  - berechne $X \cdot Y$ und vergleiche mit Z

  $\rightarrow$ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

- randomisiert:
  - Wähle 0-1 Vektor $r = (r_1, \ldots, r_n)$ zufällig
  - Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH

  $\rightarrow$ Laufzeit $\mathcal{O}(n^2)$



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

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

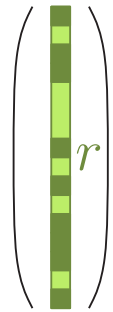
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

- deterministisch:
  - berechne $X \cdot Y$ und vergleiche mit Z

  $\rightarrow$ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

- randomisiert:
  - Wähle 0-1 Vektor $r = (r_1, \ldots, r_n)$ zufällig
  - Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH

  $\rightarrow$ Laufzeit $\mathcal{O}(n^2)$



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

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

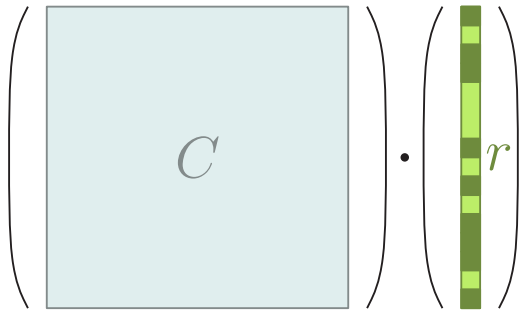
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

- deterministisch:
  - berechne $X \cdot Y$ und vergleiche mit Z

  $\rightarrow$ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

- randomisiert:
  - Wähle 0-1 Vektor $r = (r_1, \ldots, r_n)$ zufällig
  - Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH

  $\rightarrow$ Laufzeit $\mathcal{O}(n^2)$



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

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

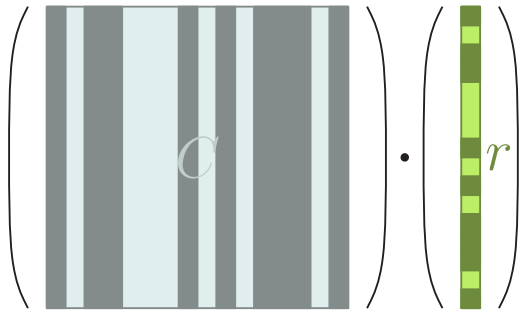
Aufgabe: Überprüfe, ob $X \cdot Y = Z$ für Matrizen X, Y, Z

- deterministisch:
  - berechne $X \cdot Y$ und vergleiche mit Z

  $\rightarrow$ Laufzeit $\mathcal{O}(n^3)$ (naiv), $\mathcal{O}(n^{2.37})$ (best)

- randomisiert:
  - Wähle 0-1 Vektor $r = (r_1, \ldots, r_n)$ zufällig
  - Wenn $X(Yr) = Zr$ dann KORREKT, sonst FALSCH

  $\rightarrow$ 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)
  $\alpha := \sum_{k \neq j} a_k r_k$ und $\beta := \sum_{k \neq j} b_k r_k$
  $\Rightarrow ar = \alpha + a_j r_j$ und $br = \beta + b_j r_j$
  $\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j) r_j$
- Annahme: Werte für $r_{1 \ldots j-1, j+1 \ldots n}$ bereits festgelegt
  $\rightarrow$ noch 2 Möglichkeiten für r_j, Gleichheit bei maximal einer
  $\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\beta - \alpha}{a_j - b_j}] \leq \frac{1}{2}$

$\rightarrow$ Fehlerrate $p \leq 0.5$

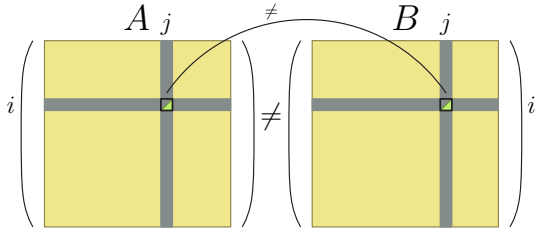
Randomisierte Algorithmen

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)
  $\alpha := \sum_{k \neq j} a_k r_k$ und $\beta := \sum_{k \neq j} b_k r_k$
  $\Rightarrow ar = \alpha + a_j r_j$ und $br = \beta + b_j r_j$
  $\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j) r_j$
- Annahme: Werte für $r_{1 \ldots j-1, j+1 \ldots n}$ bereits festgelegt
  $\rightarrow$ noch 2 Möglichkeiten für r_j, Gleichheit bei maximal einer
  $\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\beta - \alpha}{a_j - b_j}] \leq \frac{1}{2}$

$\rightarrow$ Fehlerrate $p \leq 0.5$

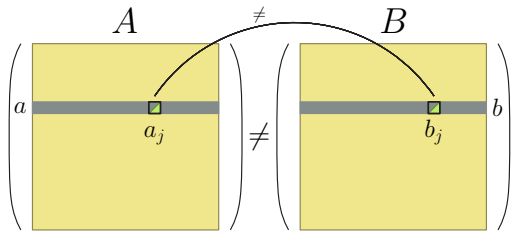
Randomisierte Algorithmen

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)
  $\alpha := \sum_{k \neq j} a_k r_k$ und $\beta := \sum_{k \neq j} b_k r_k$
  $\Rightarrow ar = \alpha + a_j r_j$ und $br = \beta + b_j r_j$
  $\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j) r_j$
- Annahme: Werte für $r_{1 \ldots j-1, j+1 \ldots n}$ bereits festgelegt
  $\rightarrow$ noch 2 Möglichkeiten für r_j, Gleichheit bei maximal einer
  $\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\beta - \alpha}{a_j - b_j}] \leq \frac{1}{2}$

$\rightarrow$ Fehlerrate $p \leq 0.5$

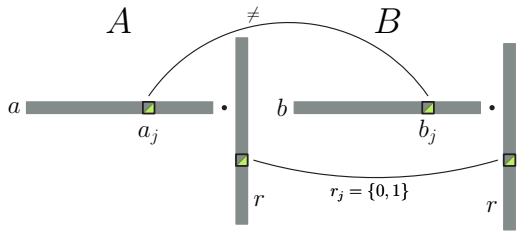
Randomisierte Algorithmen

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)
  $\alpha := \sum_{k \neq j} a_k r_k$ und $\beta := \sum_{k \neq j} b_k r_k$
  $\Rightarrow ar = \alpha + a_j r_j$ und $br = \beta + b_j r_j$
  $\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j) r_j$
- Annahme: Werte für $r_{1 \ldots j-1, j+1 \ldots n}$ bereits festgelegt
  $\rightarrow$ noch 2 Möglichkeiten für r_j, Gleichheit bei maximal einer
  $\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\beta - \alpha}{a_j - b_j}] \leq \frac{1}{2}$

$\rightarrow$ Fehlerrate $p \leq 0.5$

Randomisierte Algorithmen

Matrix-Matrix Multiplikation

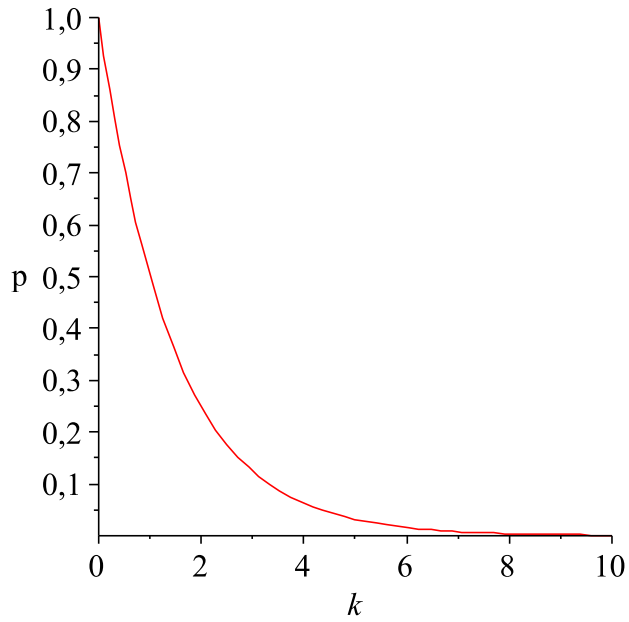
Beschleunigung durch *probability boosting*

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

- Wiederhole Test k mal mit unterschiedlicher Wahl von r
  - Ein Test liefert FALSCH
    $\rightarrow AB \neq C$, fertig
  - Alle Tests liefern KORREKT
    $\rightarrow$ *false positive* mit Wahrscheinlichkeit
    $\mathbb{P}[ABr = Cr] \leq 0.5^k$

$\rightarrow$ Laufzeit $\mathcal{O}(kn^2)$

(linear längere Laufzeit bei exponentiell weniger Fehler)



Coupon Collector

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)
- $= \sum_{i=1}^{n} \frac{n}{n-i+1} = n \sum_{i=1}^{n} \frac{1}{i}$

Coupon Collector

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)
- $= \sum_{i=1}^{n} \frac{n}{n-i+1} = n \sum_{i=1}^{n} \frac{1}{i}$

Coupon Collector

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)
- $= \sum_{i=1}^{n} \frac{n}{n-i+1} = n \sum_{i=1}^{n} \frac{1}{i}$

Coupon Collector

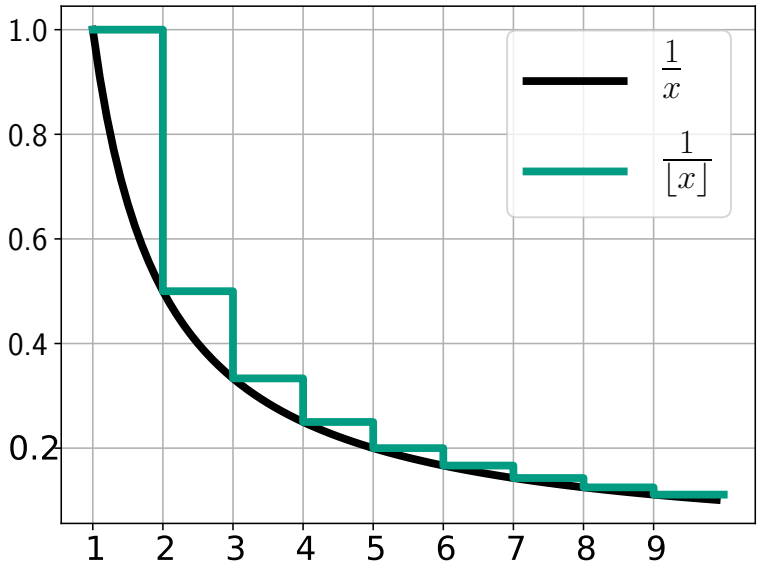
Müslipackungen enthalten jeweils eine von n verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

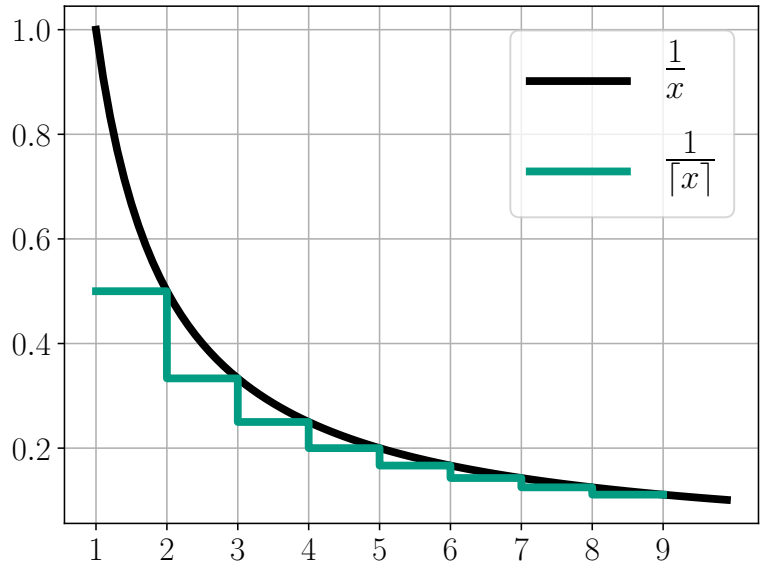
- X = # Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^{n} X_i$, mit X_i = # Packungen während ich $i-1$ Karten hatte
  - X_i sind geometrische Zufallsvariablen mit $p_i = 1 - \frac{i-1}{n}$
  - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^{n}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

Harmonische Zahlen

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

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





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

Coupon Collector

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)
- $= \sum_{i=1}^{n} \frac{n}{n-i+1} = n \sum_{i=1}^{n} \frac{1}{i} = nH_n \leq n \ln n + n$

External Memory
Speichermodell

Speichermodell

Latenzen

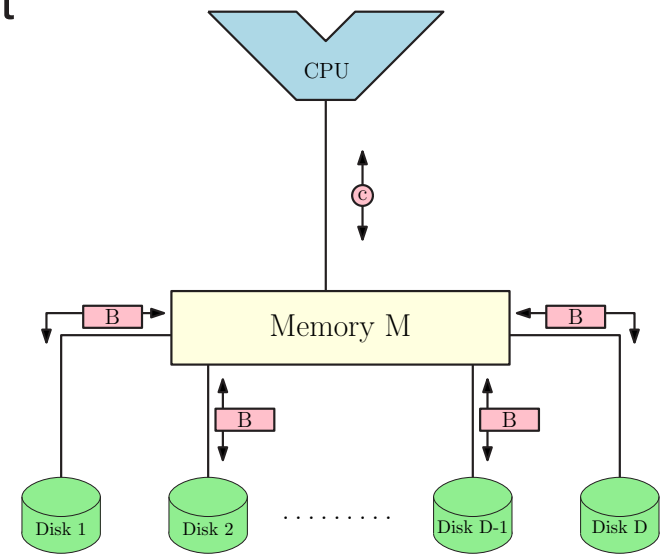
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access	120 ns	6 min
Solid-state disk I/O	50-150 µs	2-6 days
Rotational disk I/O	1-10 ms	1-12 months
Internet: SF to NYC	40 ms	4 years
Internet: SF to UK	81 ms	8 years
Internet: 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

Speichermodell

- Vitter und Shriver:
  - Parallel Disk Model (PDM)

- Speicherzugriffe in Blöcken
- Blockzugriffe minimieren $\rightarrow$ Datenlokalität

- Muster wiederholt sich in Speicherhierarchie immer wieder

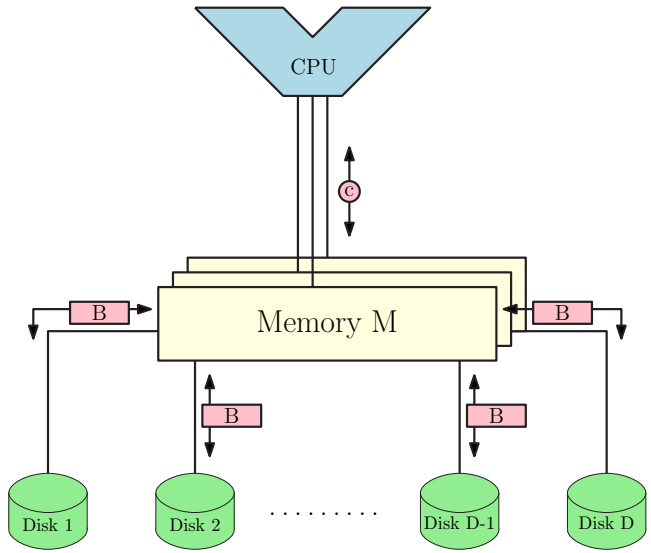


Speichermodell

- Vitter und Shriver:
  - Parallel Disk Model (PDM)

- Speicherzugriffe in Blöcken
- Blockzugriffe minimieren $\rightarrow$ Datenlokalität

- Muster wiederholt sich in Speicherhierarchie immer wieder

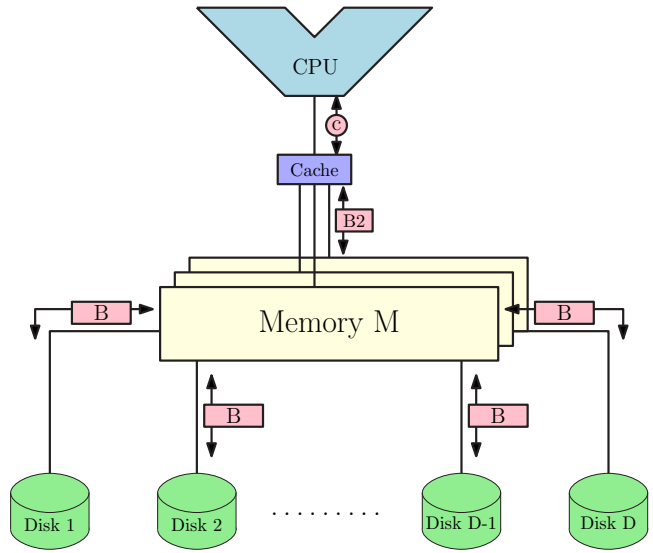


Speichermodell

- Vitter und Shriver:
  - Parallel Disk Model (PDM)

- Speicherzugriffe in Blöcken
- Blockzugriffe minimieren $\rightarrow$ Datenlokalität

- Muster wiederholt sich in Speicherhierarchie immer wieder



Blockgrößen

Einflussfaktoren

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

$\rightarrow$ 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

    $\rightarrow W = \frac{B}{T} = B / \left(\frac{1}{128} T_{seek} + B / W_{max} \right) \stackrel{!}{=} 0.95 \cdot W_{max}$

    $\rightarrow B = 0.15 \cdot W_{max} \cdot T_{seek} = 111$ Byte!

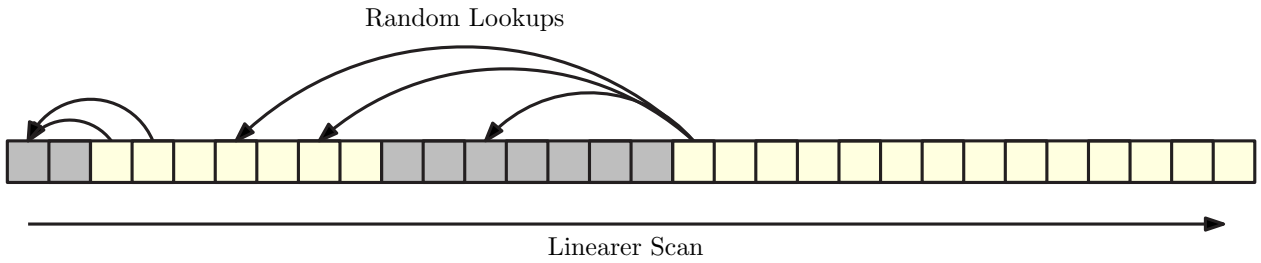
I/O-effizientes Design

I/O-effizientes Design

- 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.

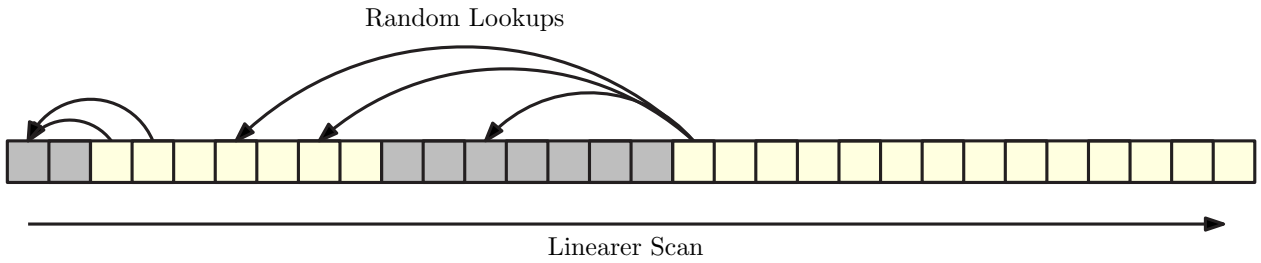
I/O-effizientes Design

- 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.



I/O-effizientes Design

- 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

I/O-effizientes Design

Grundlegende Techniken

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

I/O-effizientes Design

Grundlegende Techniken

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

I/O-effizientes Design

Grundlegende Techniken

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

I/O-effizientes Design

Grundlegende Techniken

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

I/O-effizientes Design

Grundlegende Techniken

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

I/O-effizientes Design

Grundlegende Techniken

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

I/O-effizientes Design

Grundlegende Techniken

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

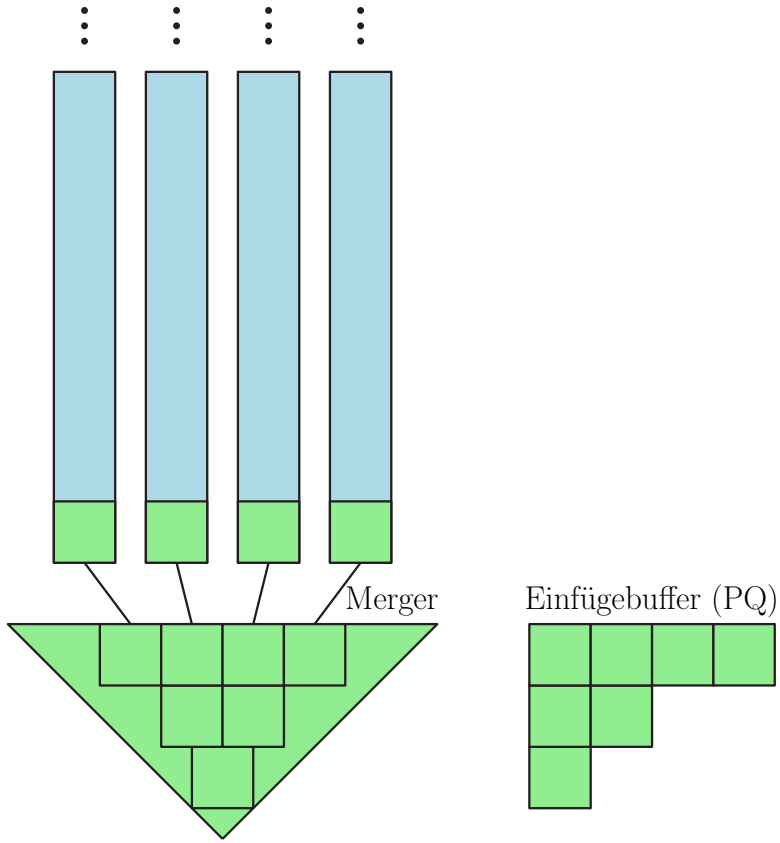
I/O-effizientes Design

Grundlegende Techniken

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

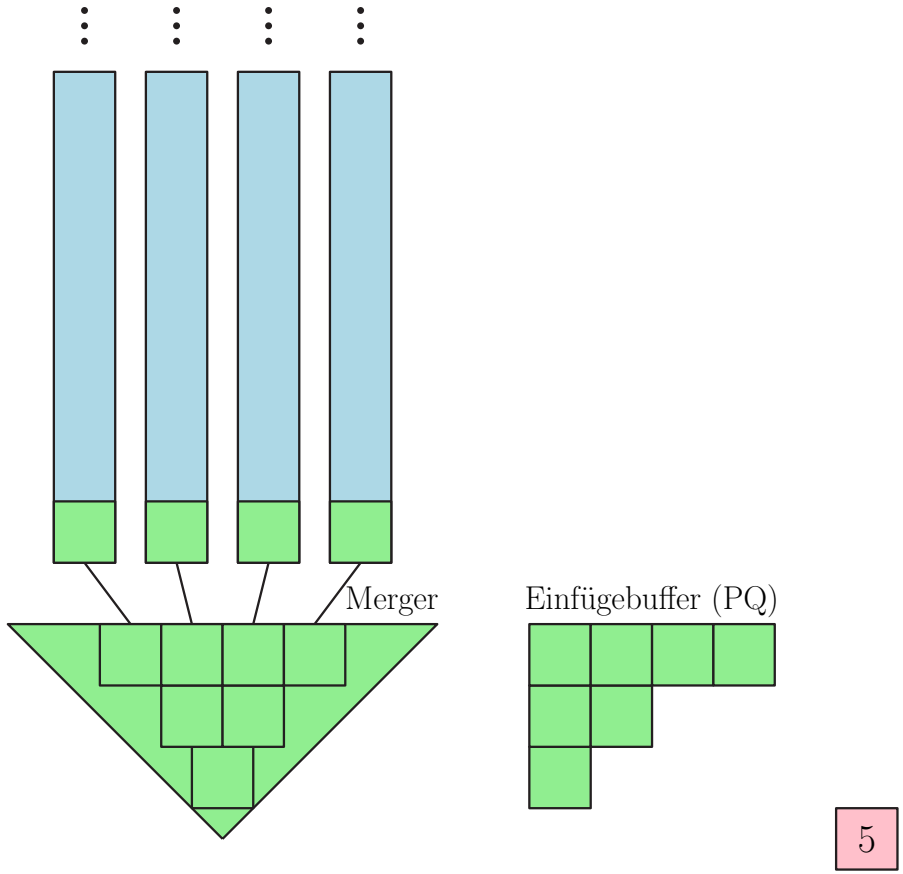
Externe Priority Queue

- 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?



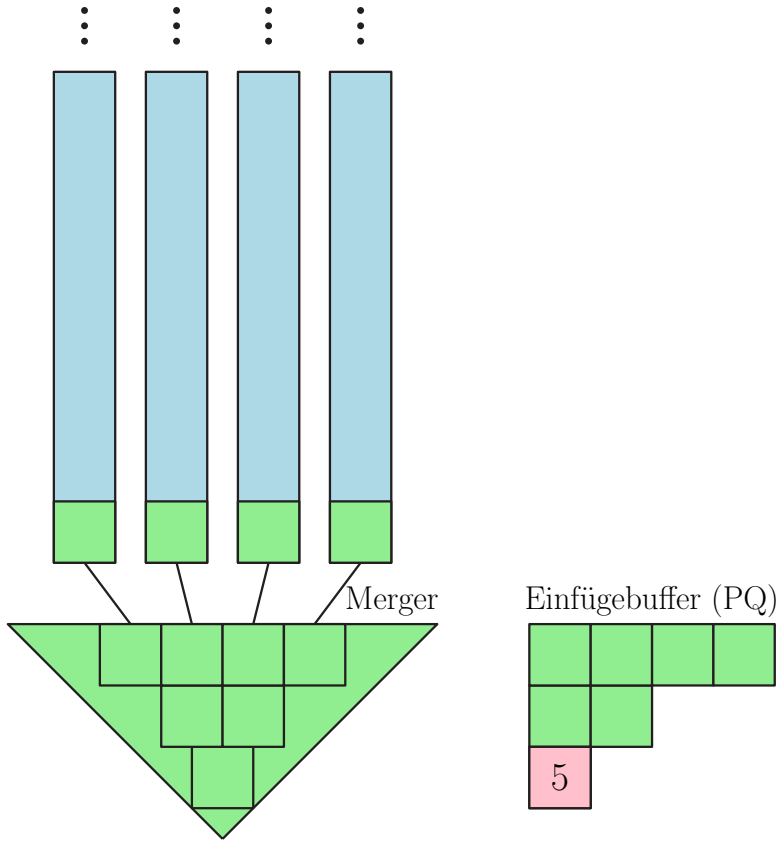
Externe Priority Queue

- 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?



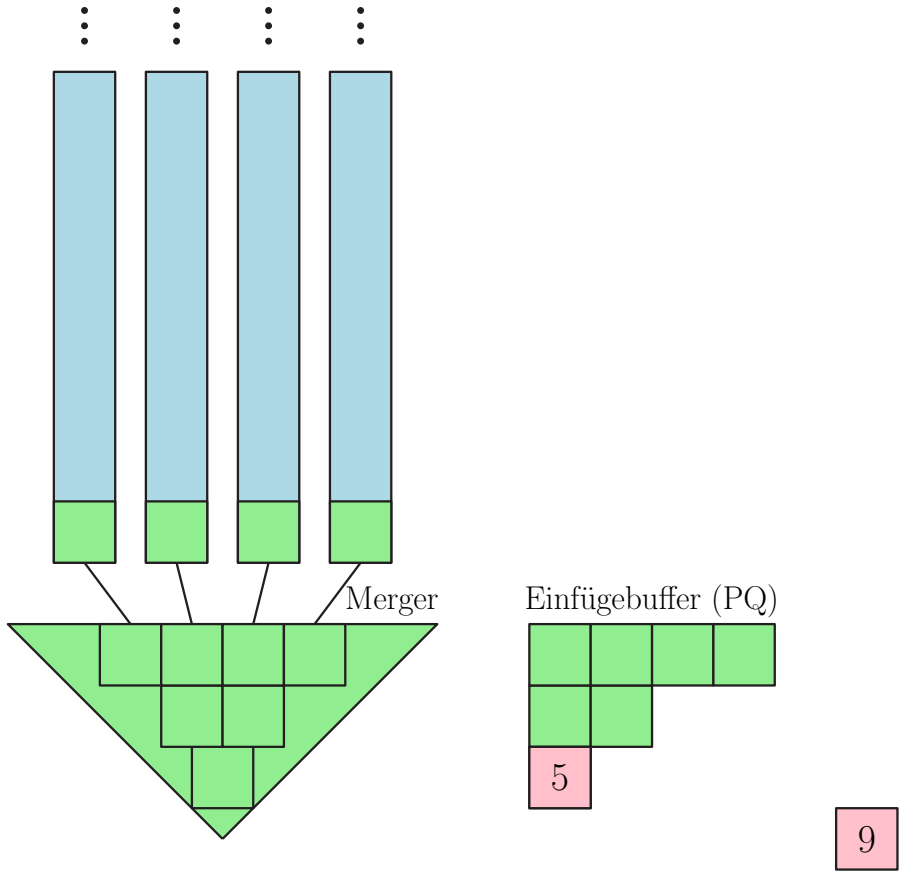
Externe Priority Queue

- 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?



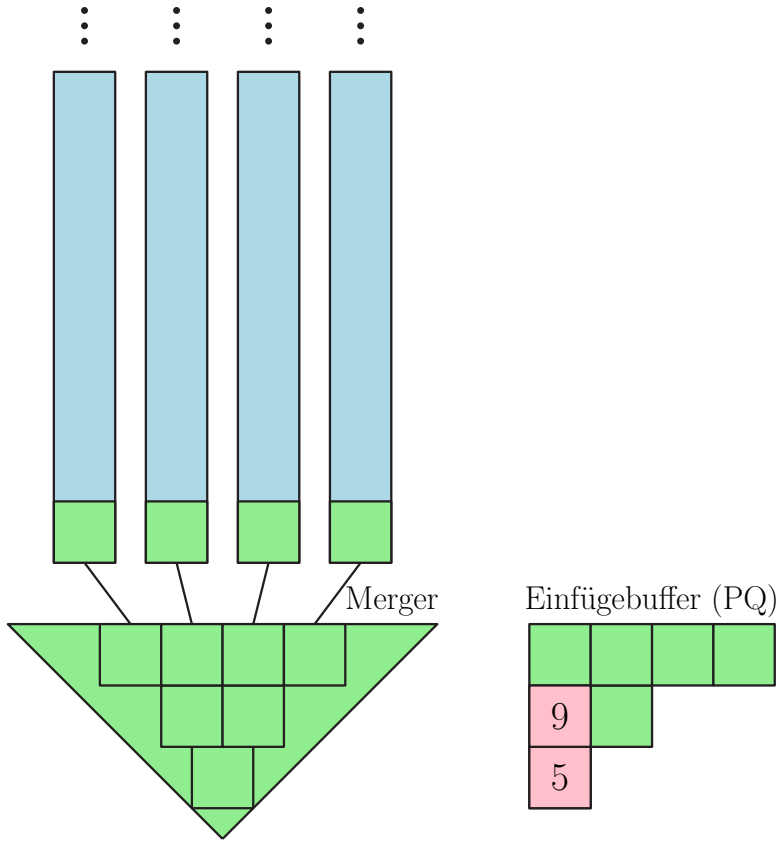
Externe Priority Queue

- 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?



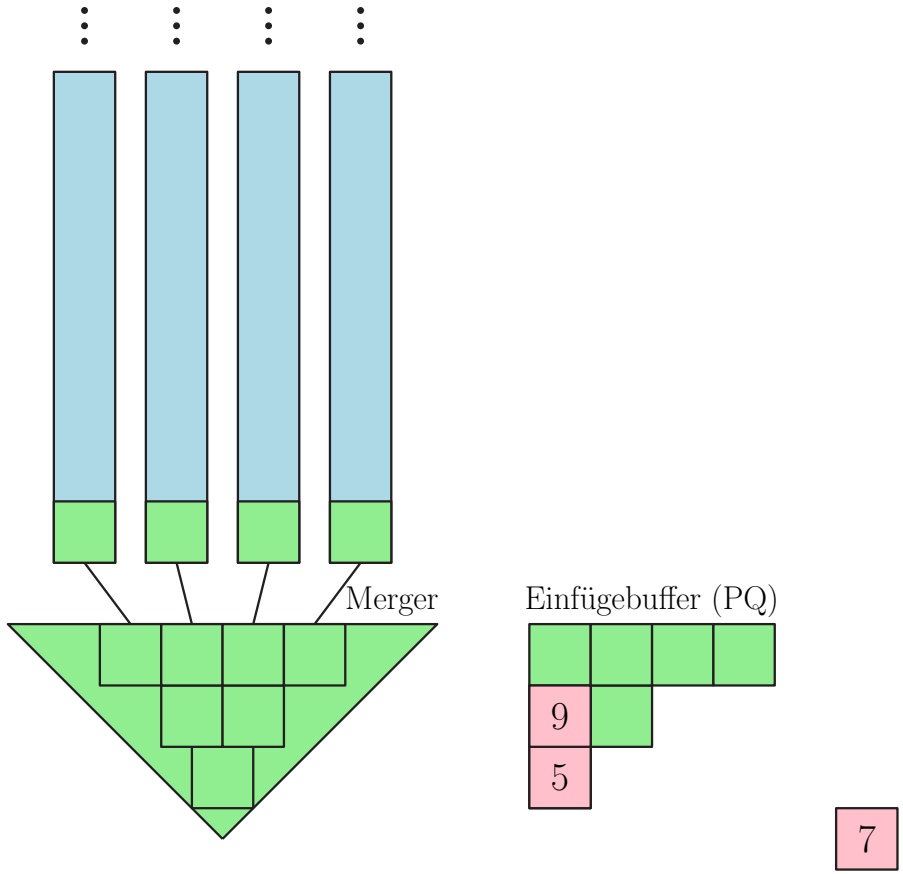
Externe Priority Queue

- 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?



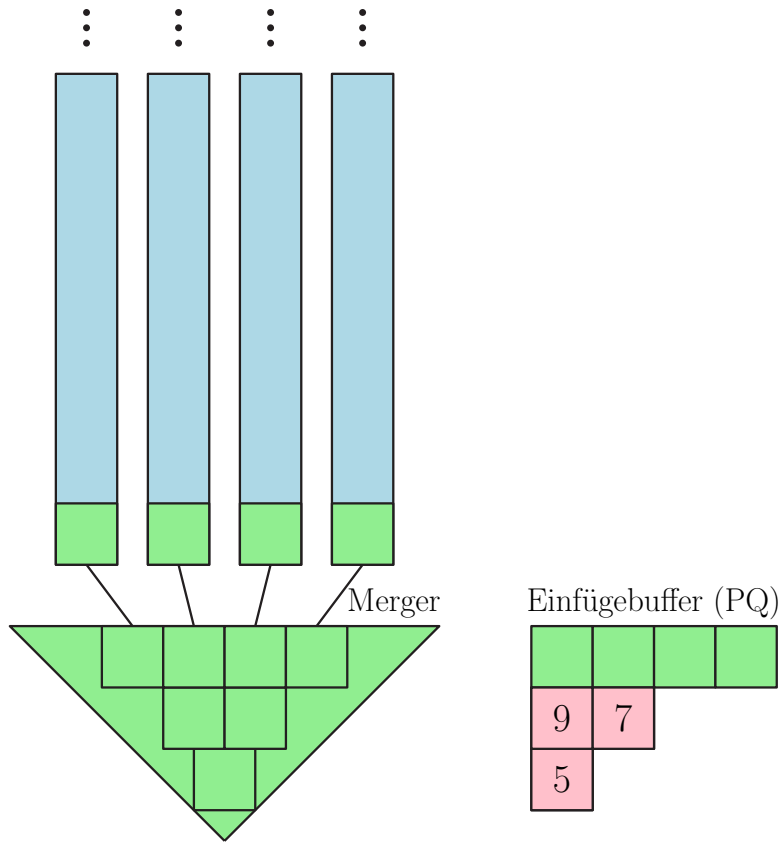
Externe Priority Queue

- 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?



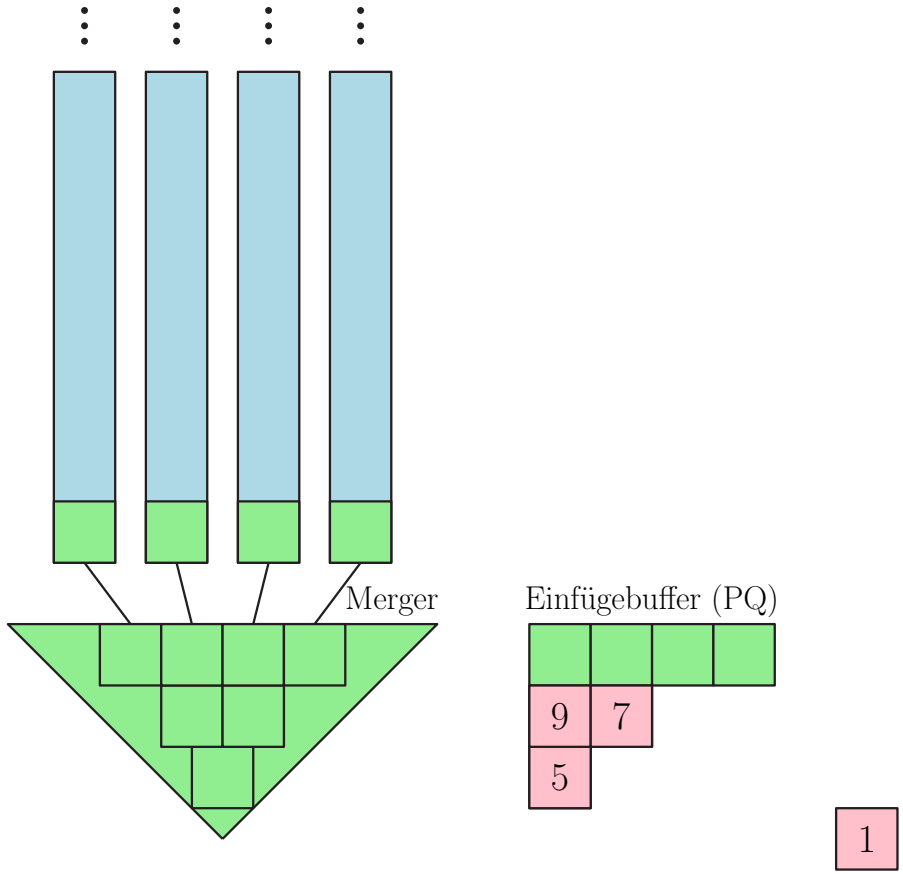
Externe Priority Queue

- 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?



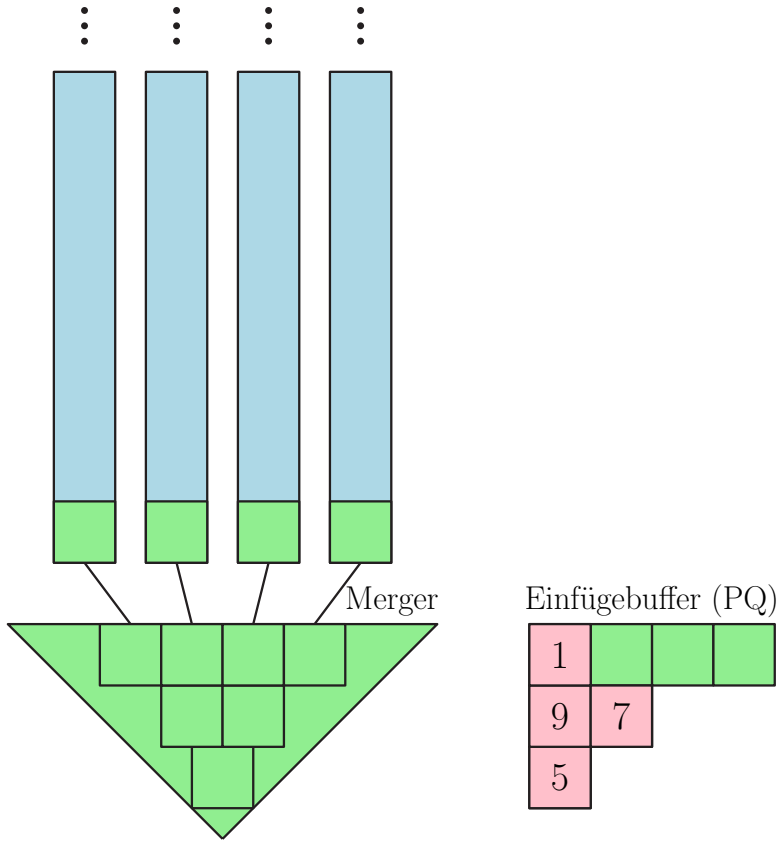
Externe Priority Queue

- 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?



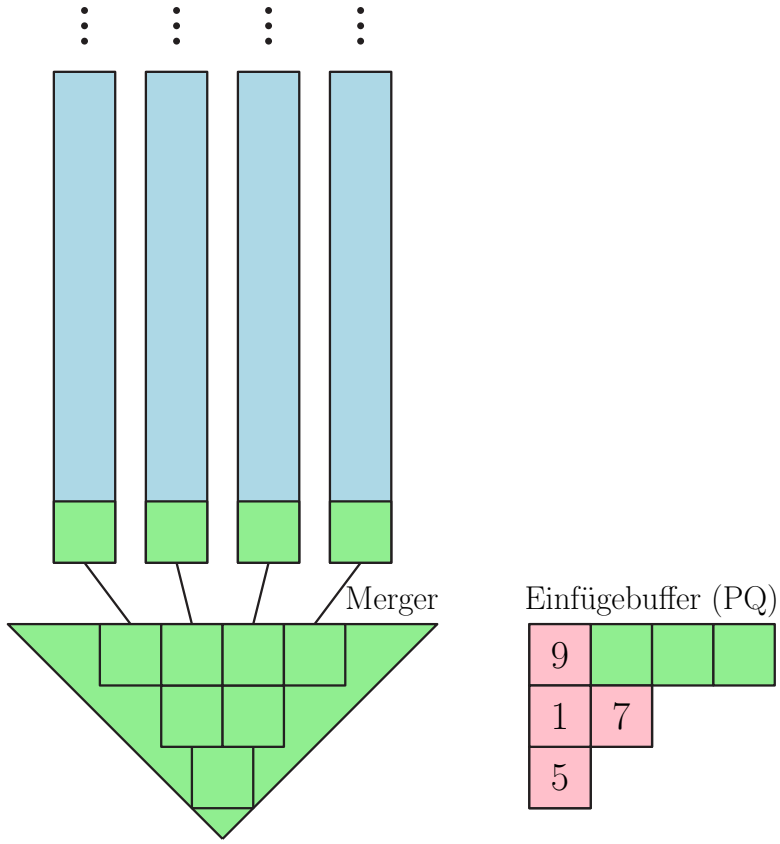
Externe Priority Queue

- 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?



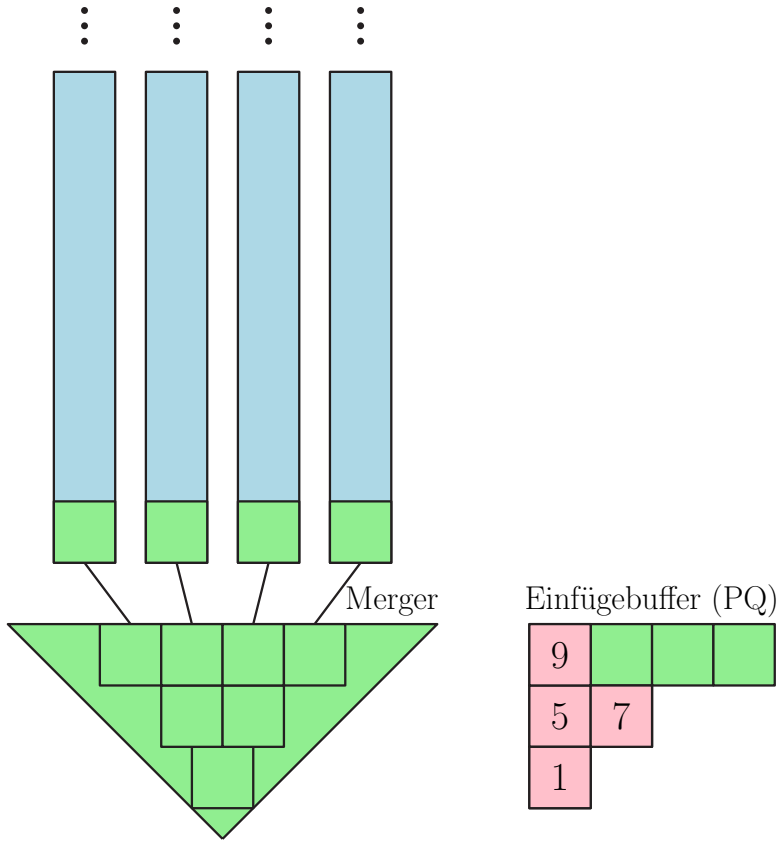
Externe Priority Queue

- 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?



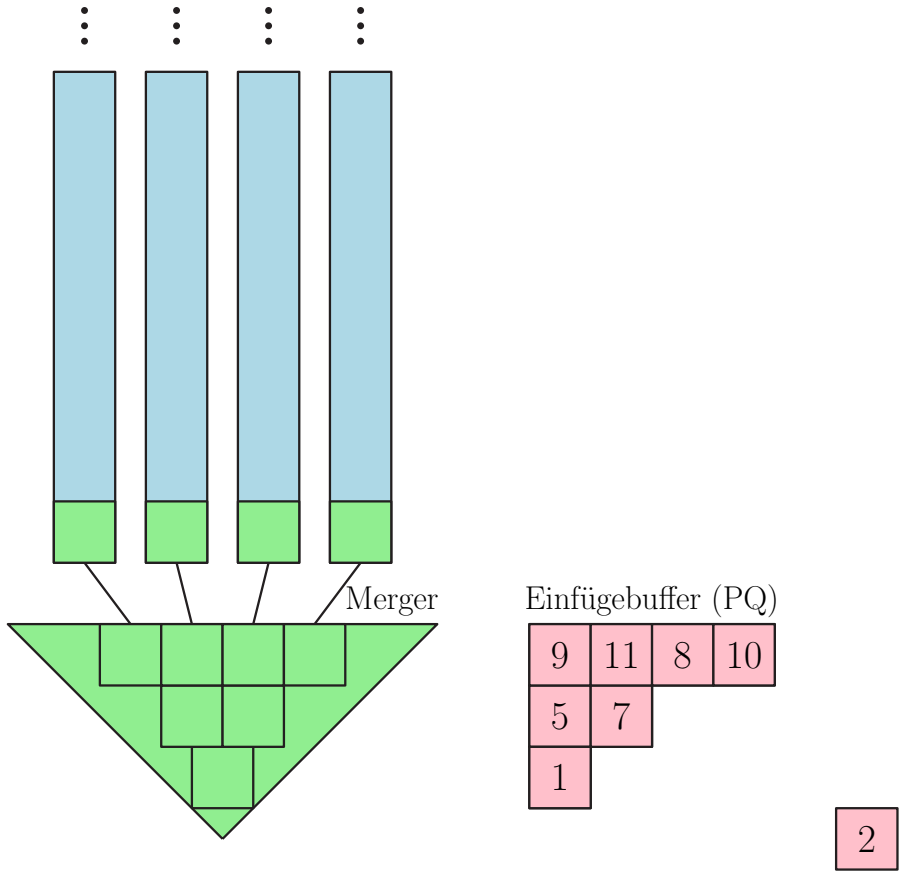
Externe Priority Queue

- 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?



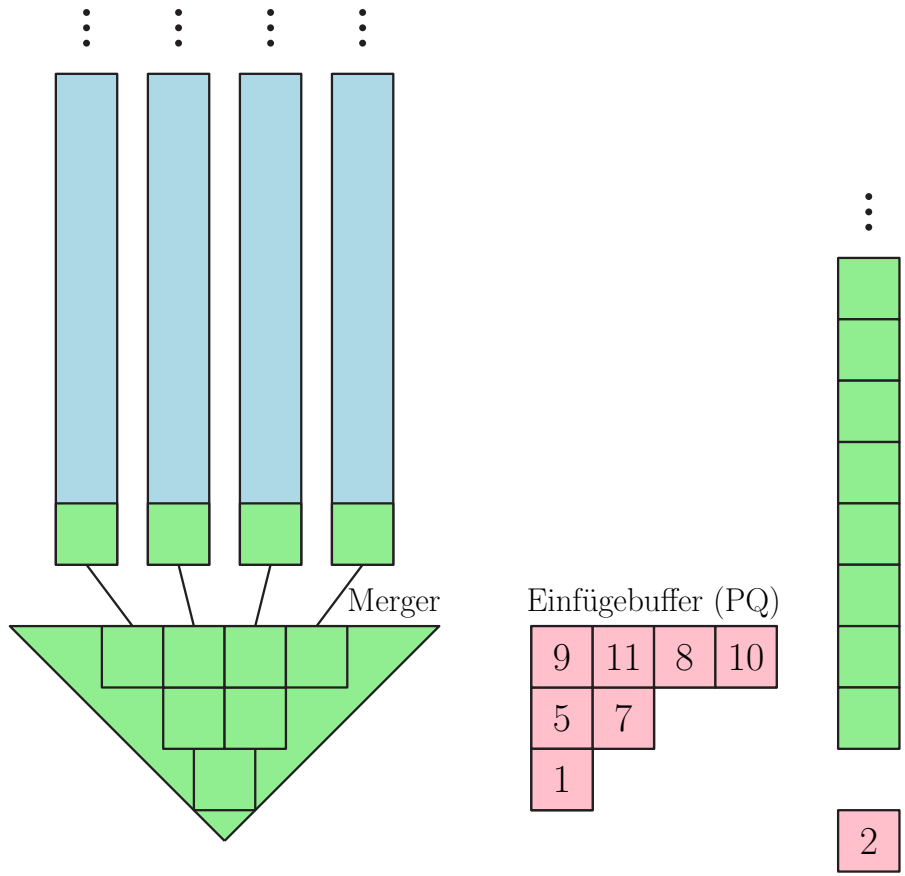
Externe Priority Queue

- 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?



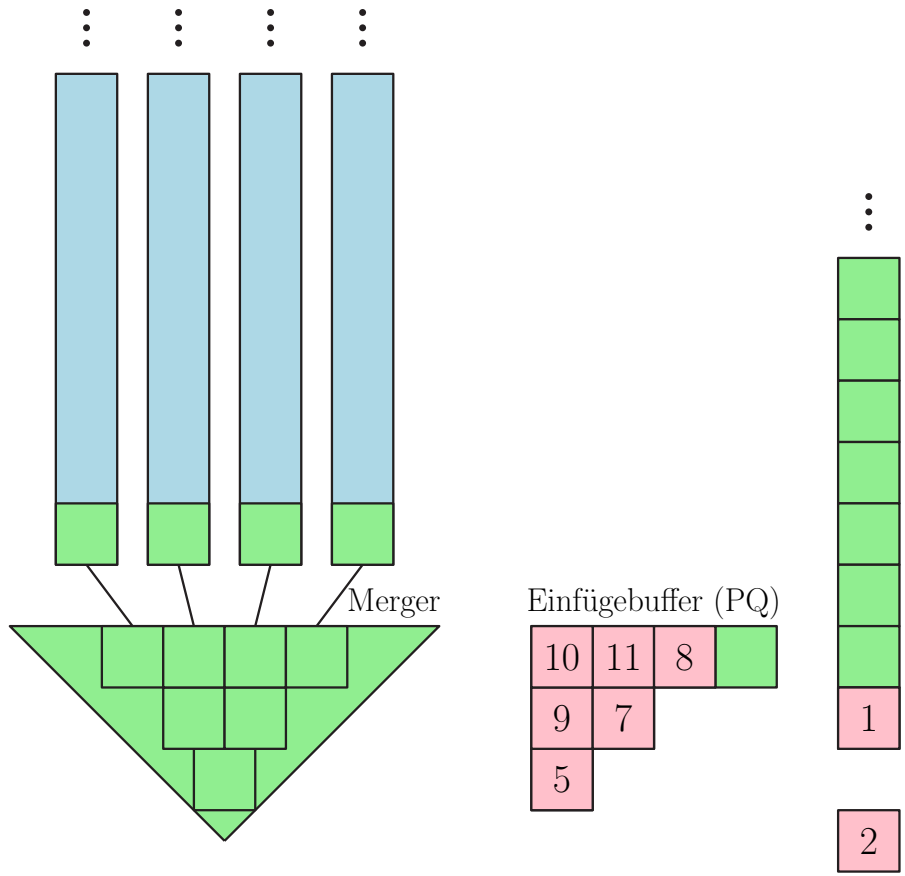
Externe Priority Queue

- 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?



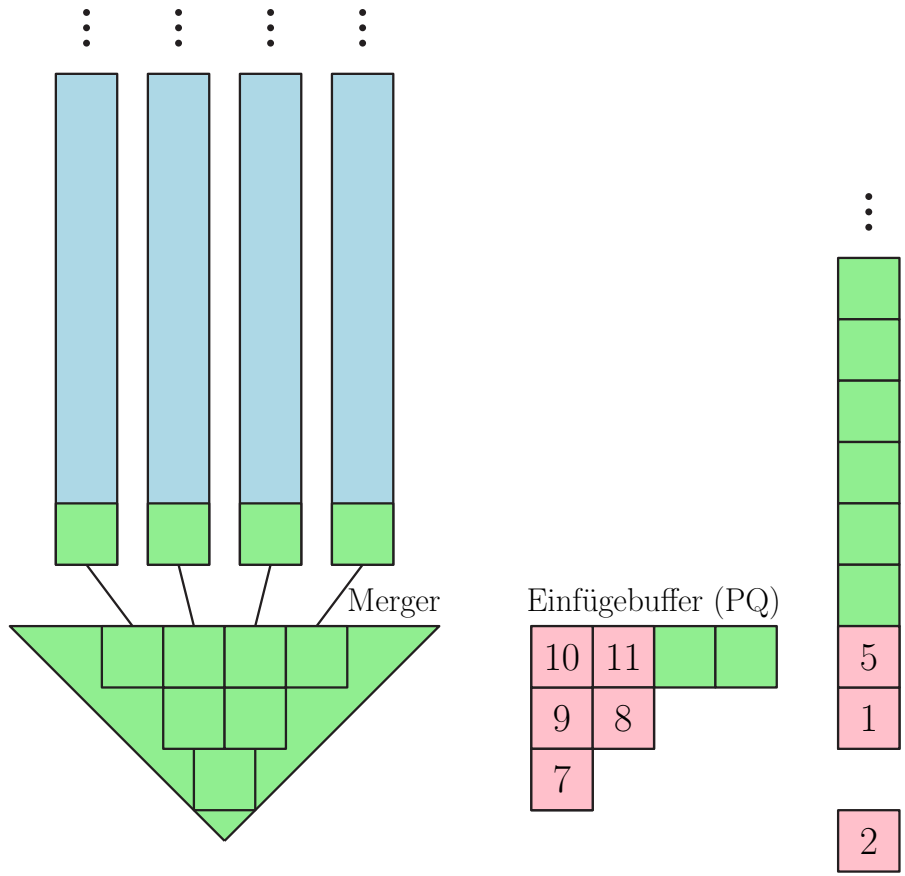
Externe Priority Queue

- 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?



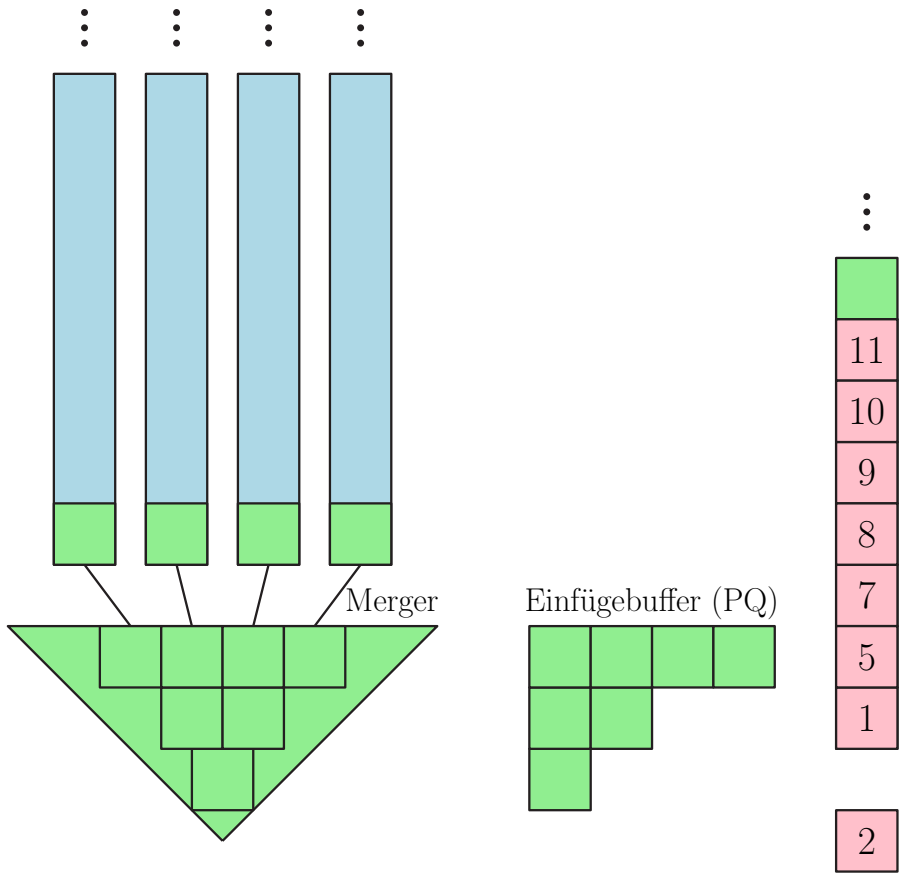
Externe Priority Queue

- 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?



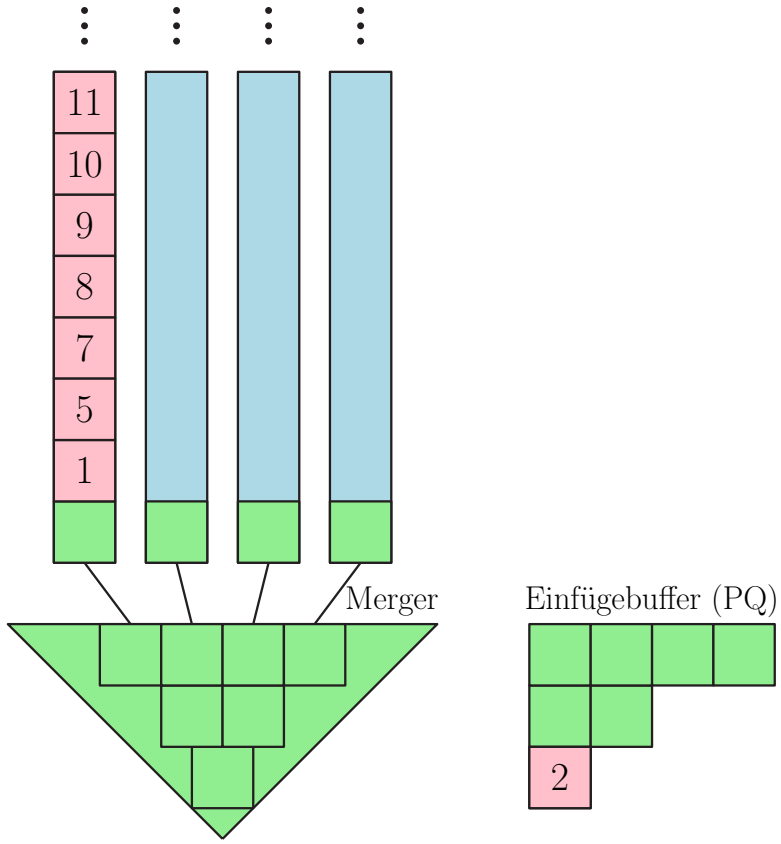
Externe Priority Queue

- 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?



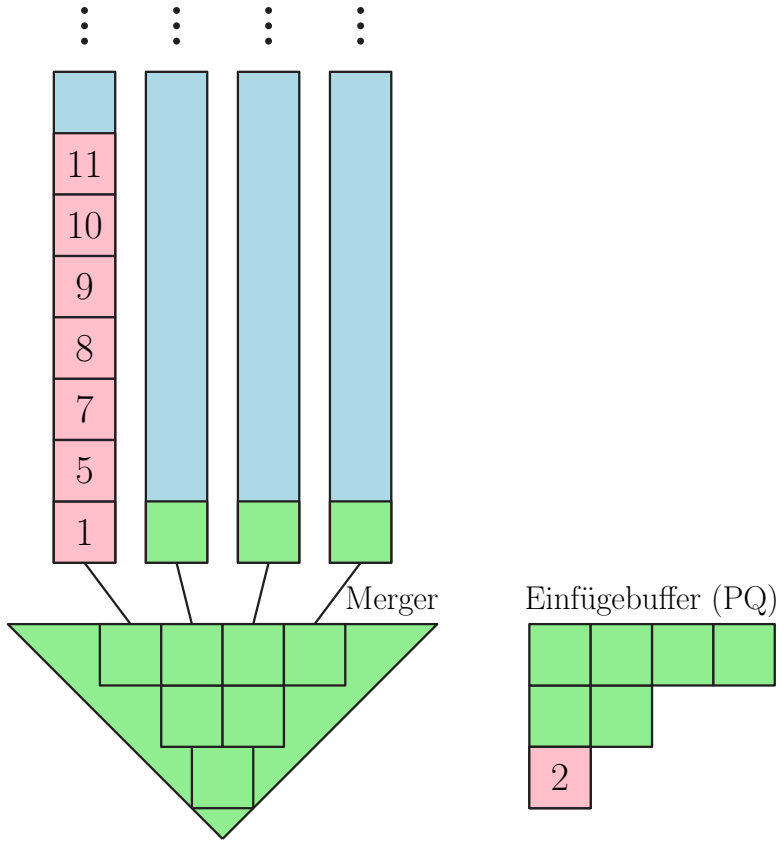
Externe Priority Queue

- 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?



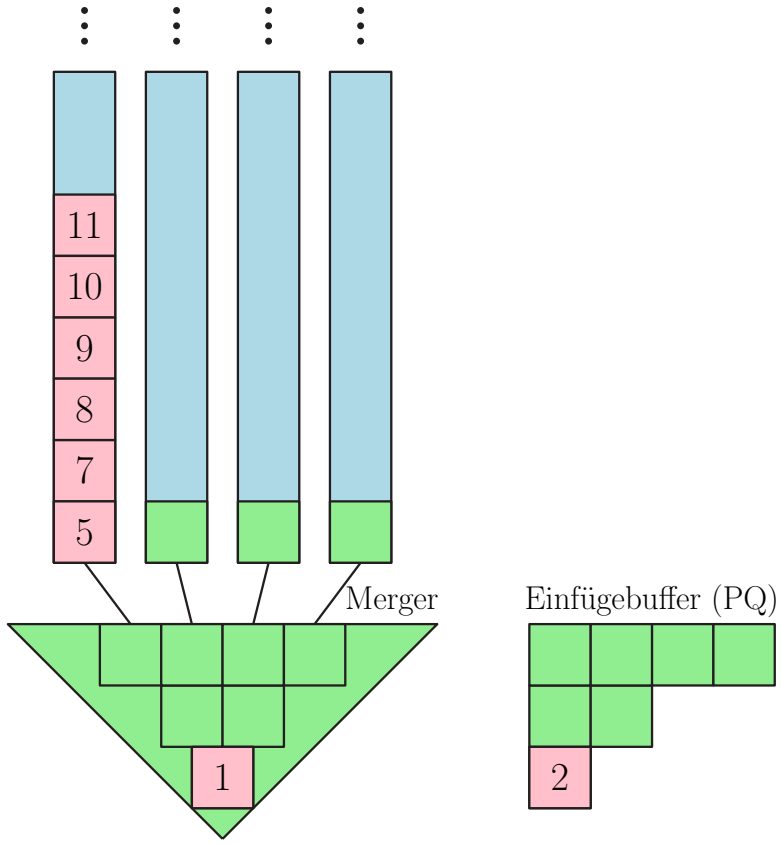
Externe Priority Queue

- 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?



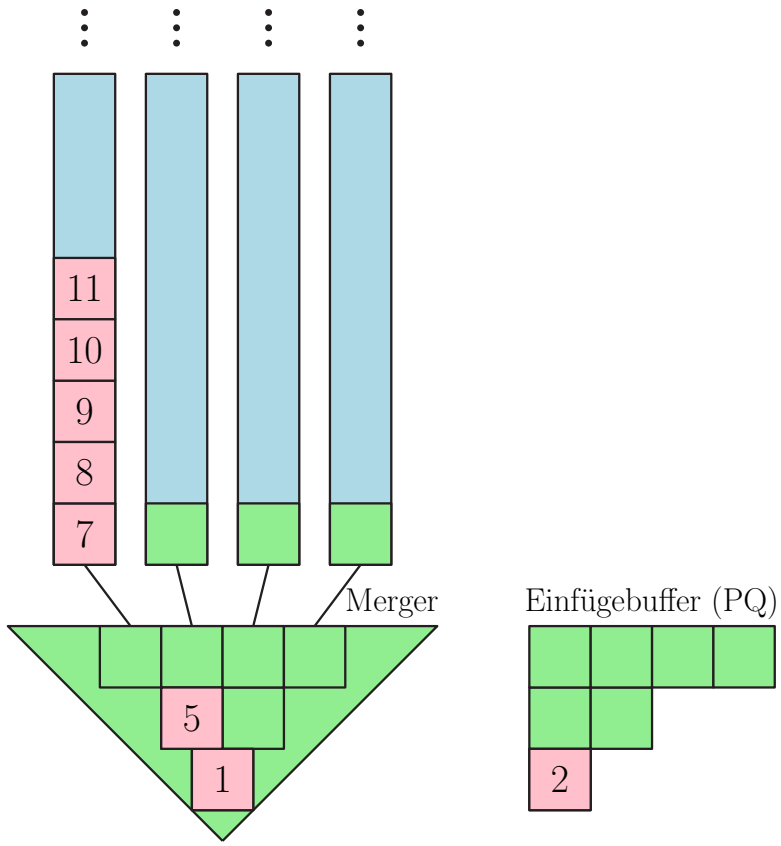
Externe Priority Queue

- 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?



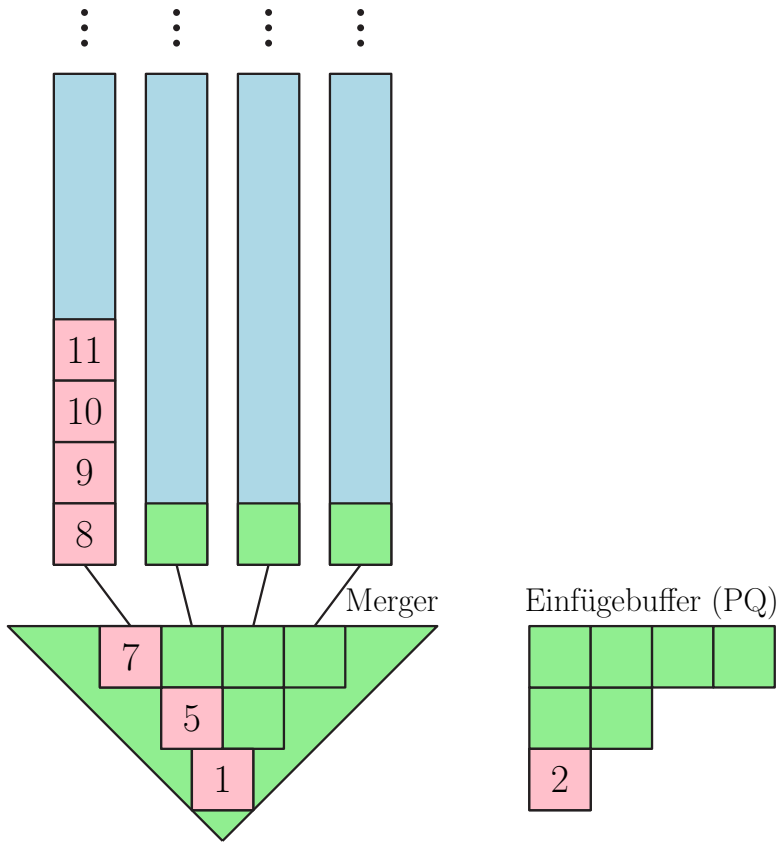
Externe Priority Queue

- 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?



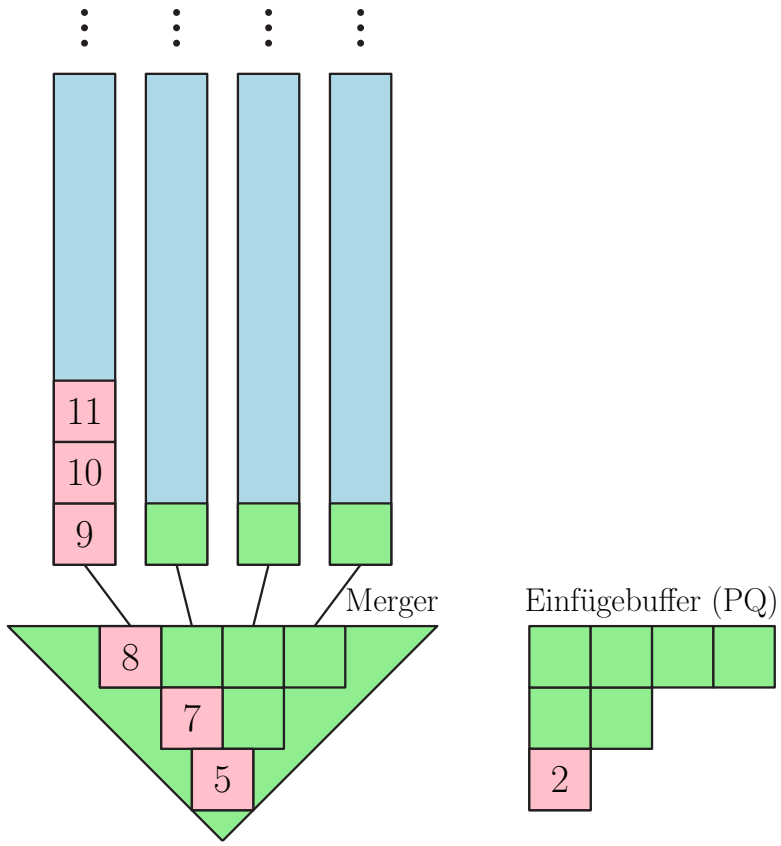
Externe Priority Queue

- 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?



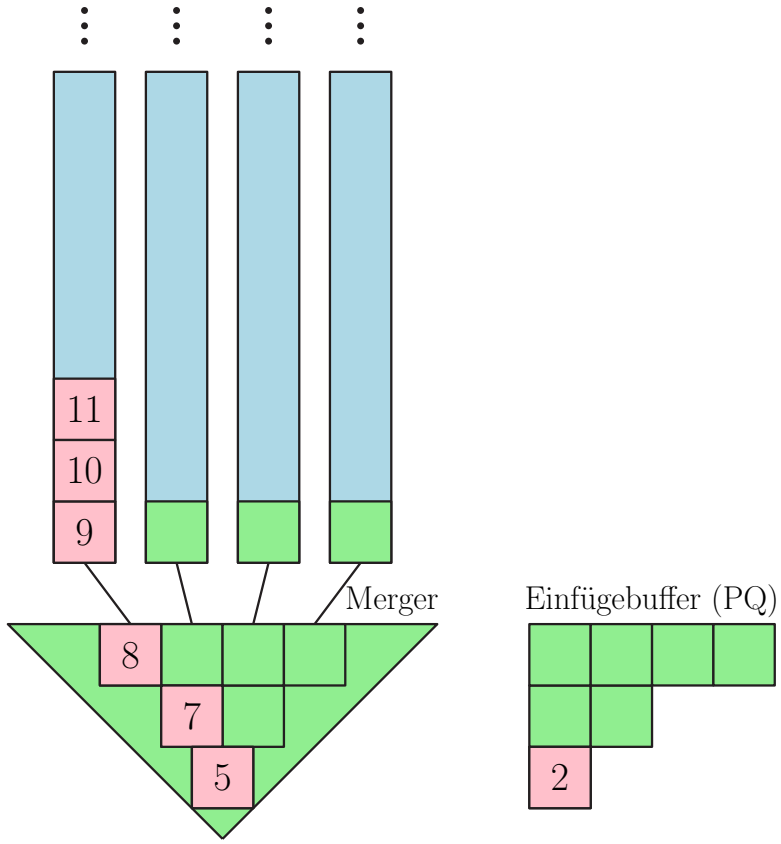
Externe Priority Queue

- 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?



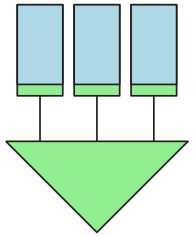
Externe Priority Queue

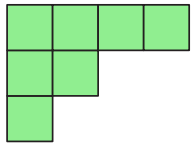
- sortierte Teilsequenzen
- Einfügepuffer (interne PQ)
- Merger (interne PQ)
- Operationen:
  - `Insert`
  - `DeleteMin`
- natürliches Limit:
  - $\#\text{Eingefügte Elemente} \leq \frac{1}{8} \cdot \frac{M^2}{B}$
- Was tun bei mehr Elementen?



Externe Priority Queue

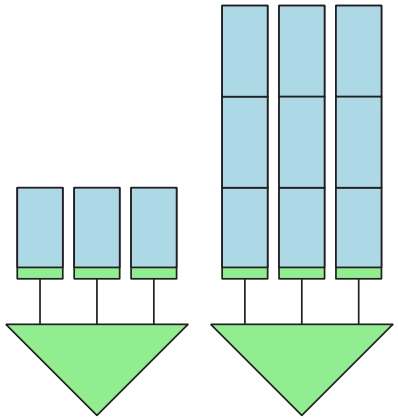
für sehr große Datenmengen

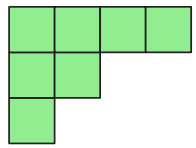




Externe Priority Queue

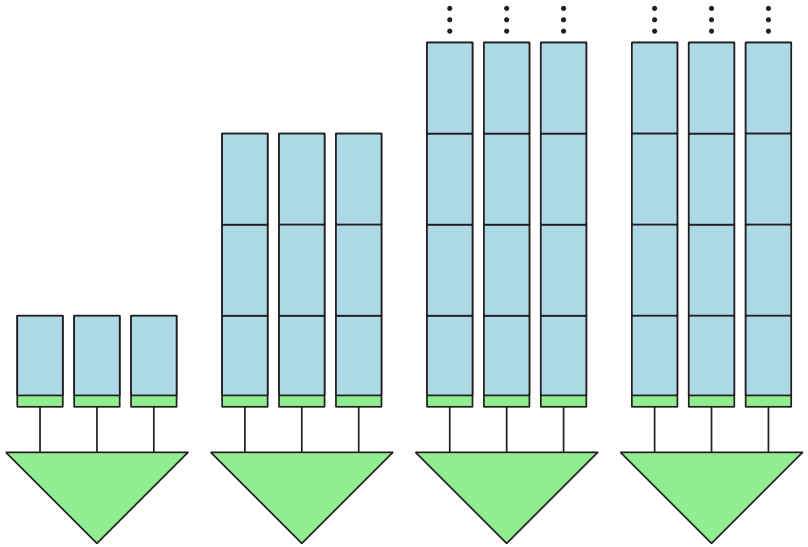
für sehr große Datenmengen

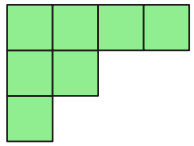




Externe Priority Queue

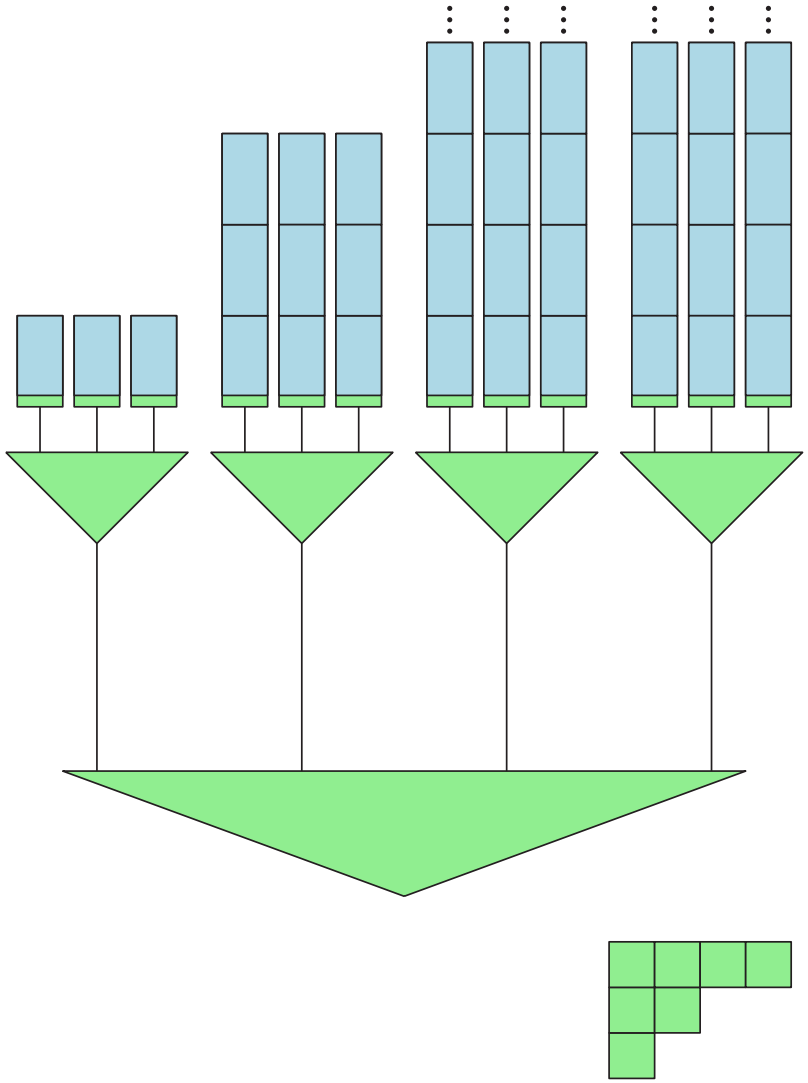
für sehr große Datenmengen





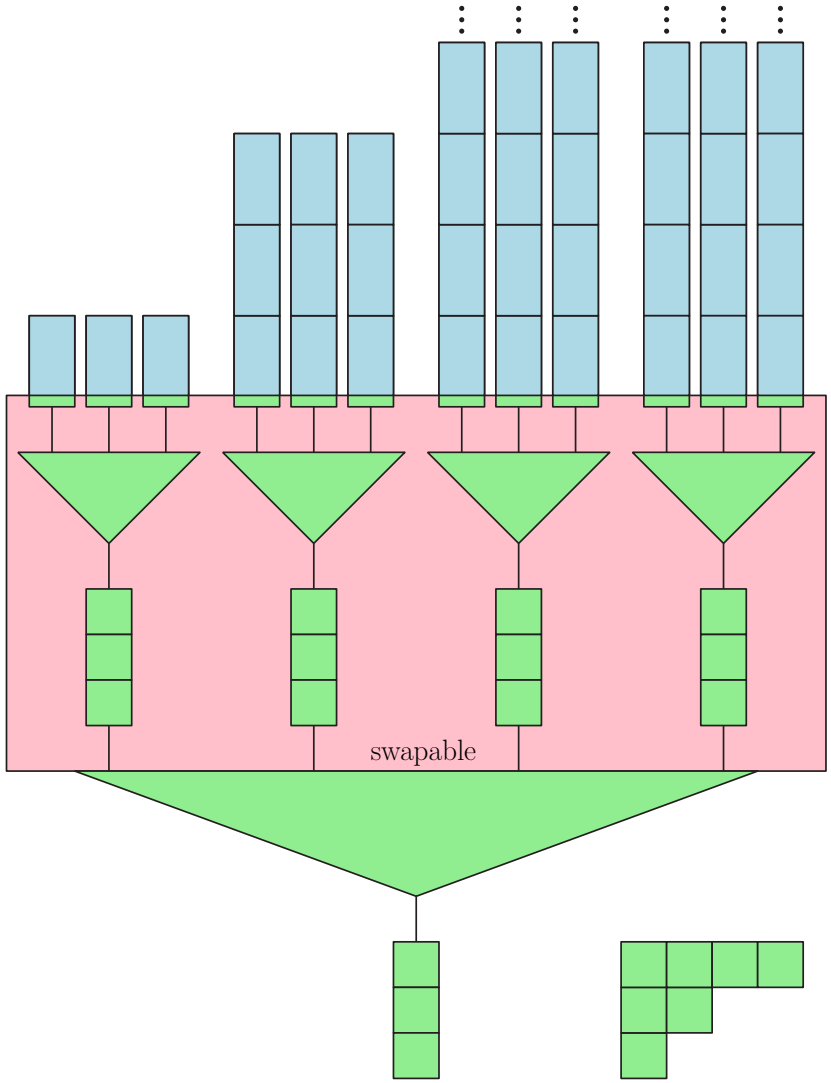
Externe Priority Queue

für sehr große Datenmengen



Externe Priority Queue

für sehr große Datenmengen



Externes Sortieren

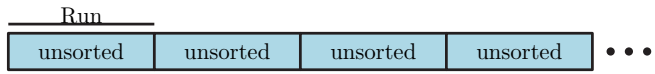
Zwei-Phasen Algorithmus

- *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

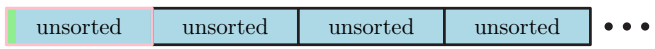
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

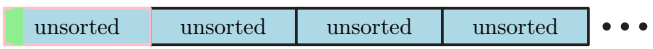
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Externes Sortieren

Zwei-Phasen Algorithmus

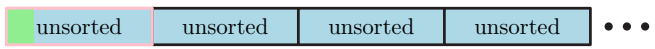
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

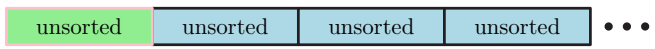
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

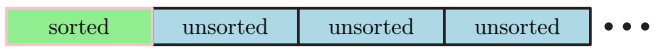
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Externes Sortieren

Zwei-Phasen Algorithmus

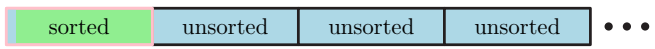
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

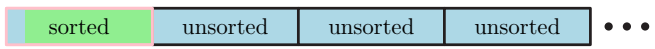
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

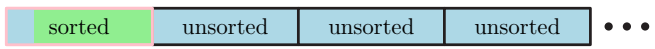
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Externes Sortieren

Zwei-Phasen Algorithmus

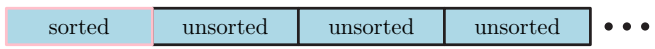
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

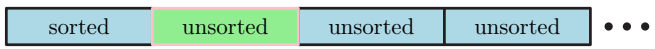
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

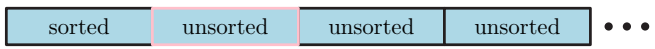
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Externes Sortieren

Zwei-Phasen Algorithmus

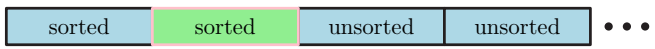
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

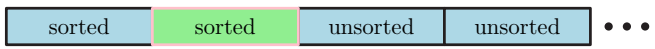
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

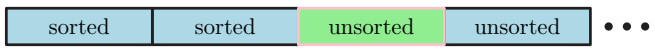
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



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

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

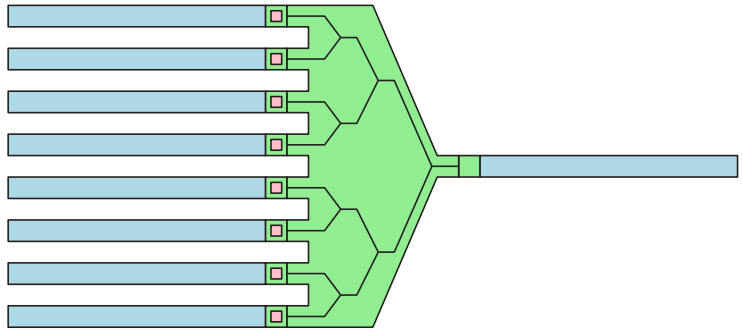
Zwei-Phasen Algorithmus

- *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

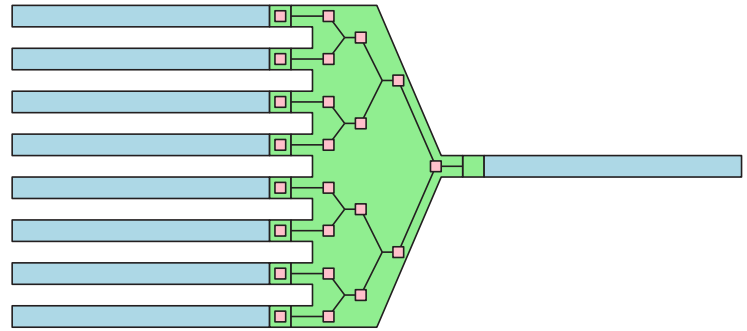
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Externes Sortieren

Zwei-Phasen Algorithmus

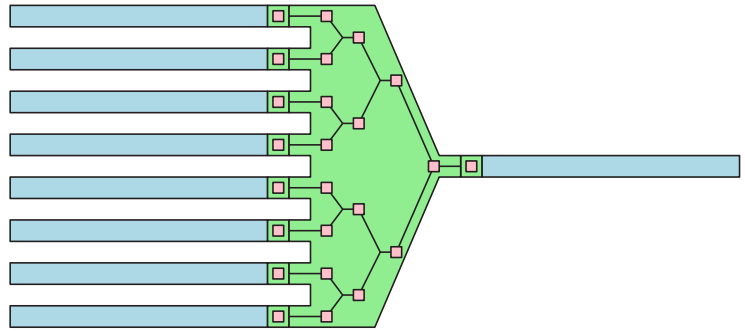
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

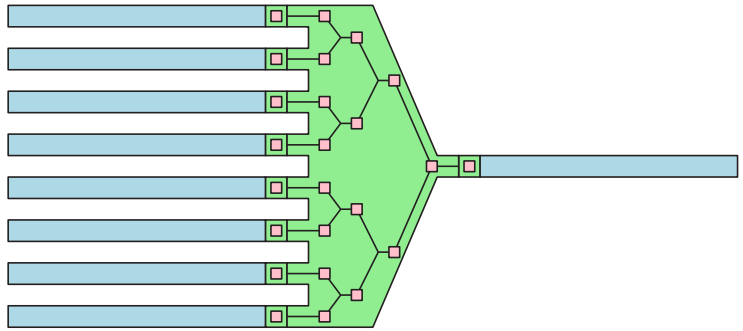
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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

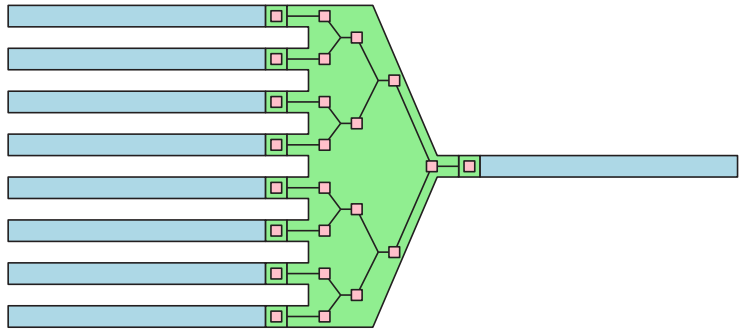
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Externes Sortieren

Zwei-Phasen Algorithmus

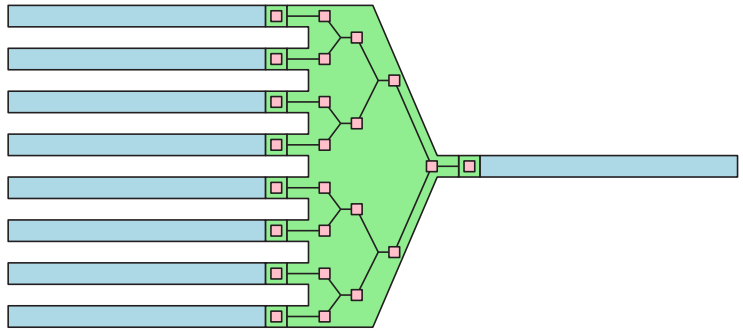
- *Run Formation*
  - Run entspricht einem Teilbereich zu sortierender Daten
  - $\lceil \frac{n}{M} \rceil$ Stück, Größe M
  - jeweils $\mathcal{O}(M \log M)$ Arbeit (Sortieren)
  - alle Daten einmal lesen + schreiben
- *Multiway Merge*
  - jede Mischphase liest und schreibt alle Daten $\rightarrow \frac{2n}{B}$ 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:
  $\mathcal{O}\left(\frac{2n}{B} + \frac{2n}{B} \cdot \lceil \log_{M/B} \frac{n}{M} \rceil\right)$



Ende!

Feierabend!