Algorithmen für die Speicherhierarchie

Einführung und Aufbau der Vorlesung

Riko Jacob

Lehrstuhl für Effiziente Algorithmen
Fakultät für Informatik
Technische Universität München

Vorlesung 22. April 2009

Technische Universität München

Akademischer Werdegang

Riko Jacob

1992-1997	Diplom in Würzburg
1997-1998	GRA in Los Alamos (TRANSIMS)
1998-2002	PhD in Aarhus, Dänemark (BRICS)
2002-2003	PostDoc an der LMU (Hans-Peter Kriegel)
2003-2007	Oberassistent an der ETH Zürich, Schweiz (Peter Widmayer)
2007-2012	Nachwuchsgruppenleiter an der TUM

Nachwuchsgruppe

Speicherhierarchie-optimale Matrixmultiplikations-Programme

- Behandelt eine Grundlegende Problemstellung
- Theoretische Fragestellungen
  - untere Schranken
  - Optimierungsproblem
  - Verbindung zu Parallelrechnern
- Algorithm Engineering
  - Evaluation der Algorithmen
  - Realistische Modellbildung

Für 5 Jahre im Rahmen des Emmy Noether Programms der Deutschen Forschungsgemeinschaft (DFG).

Technische Universität München

Realer Rechner

Beispiel

	Speedo	Memo
Name:		
CPU:	3 GHz	1 GHz
Speicher:	250 Mb	1000 Mb
Textverarbeitung:	++	+
Text+Mail+Firefox+PDF:	−−	o

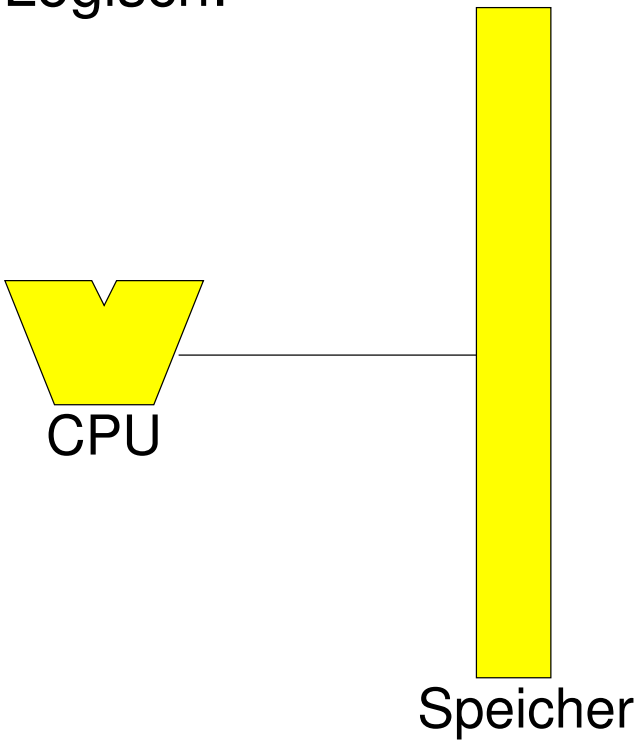
Erfahrung: Ausreichend Speicher ist Voraussetzung

Manche Anwendung: So viel Speicher kann man nicht einbauen (Satellitenbilder, . . .)

Technische Universität München

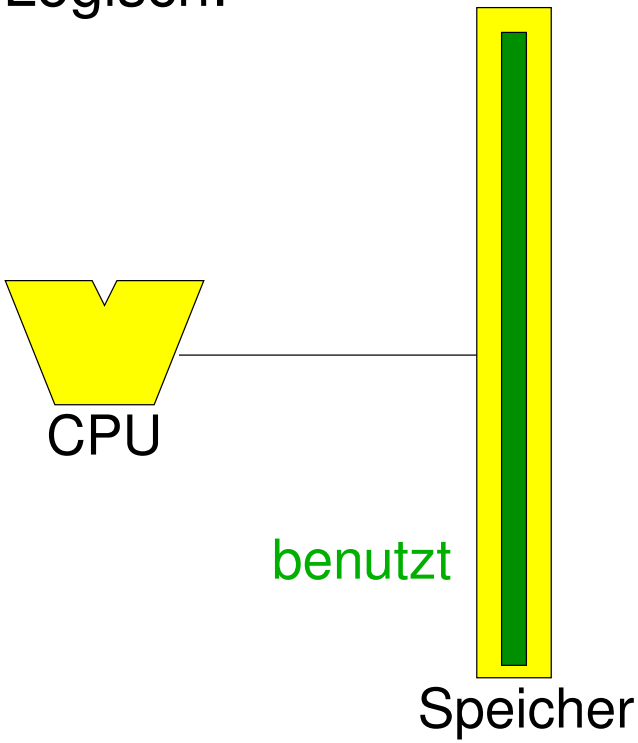
Speicherhierarchie

Logisch:

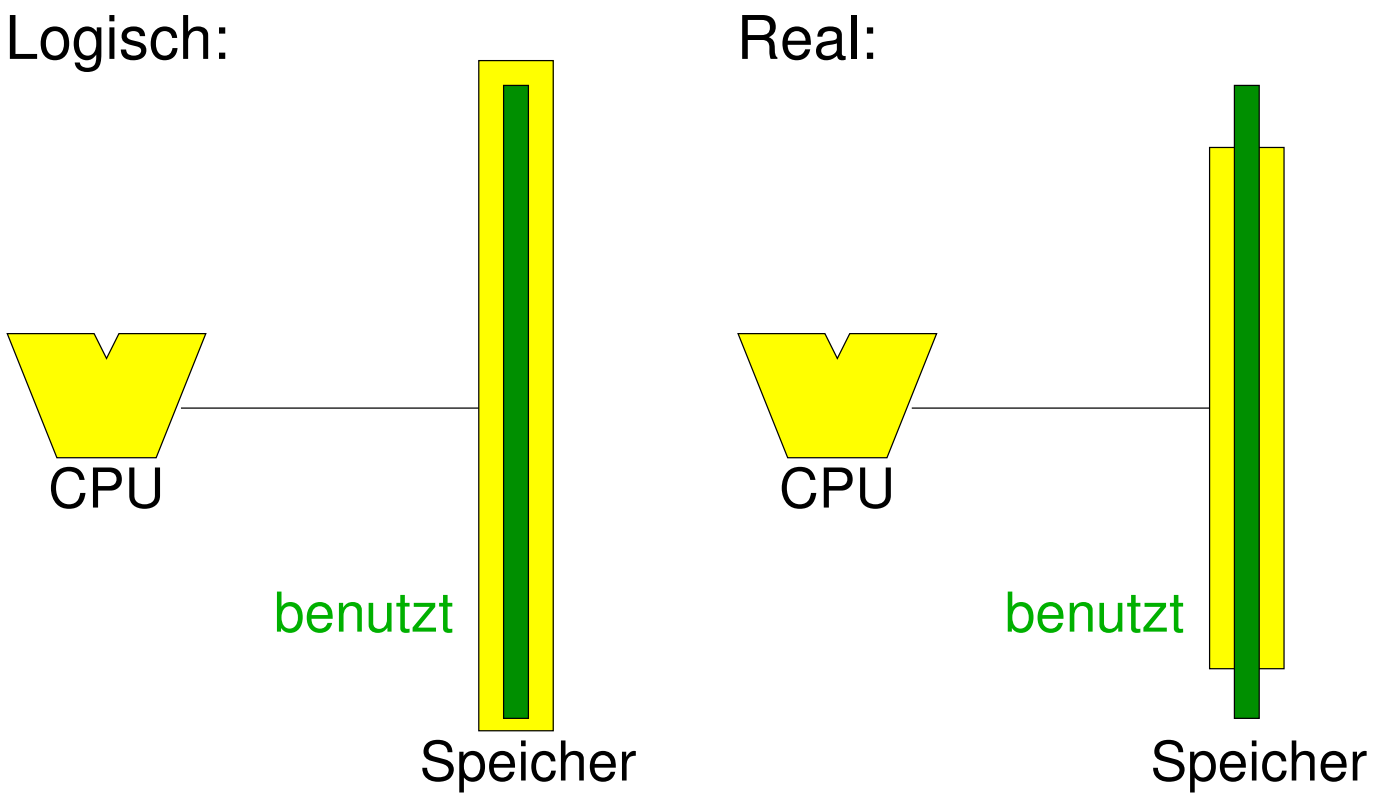


Speicherhierarchie

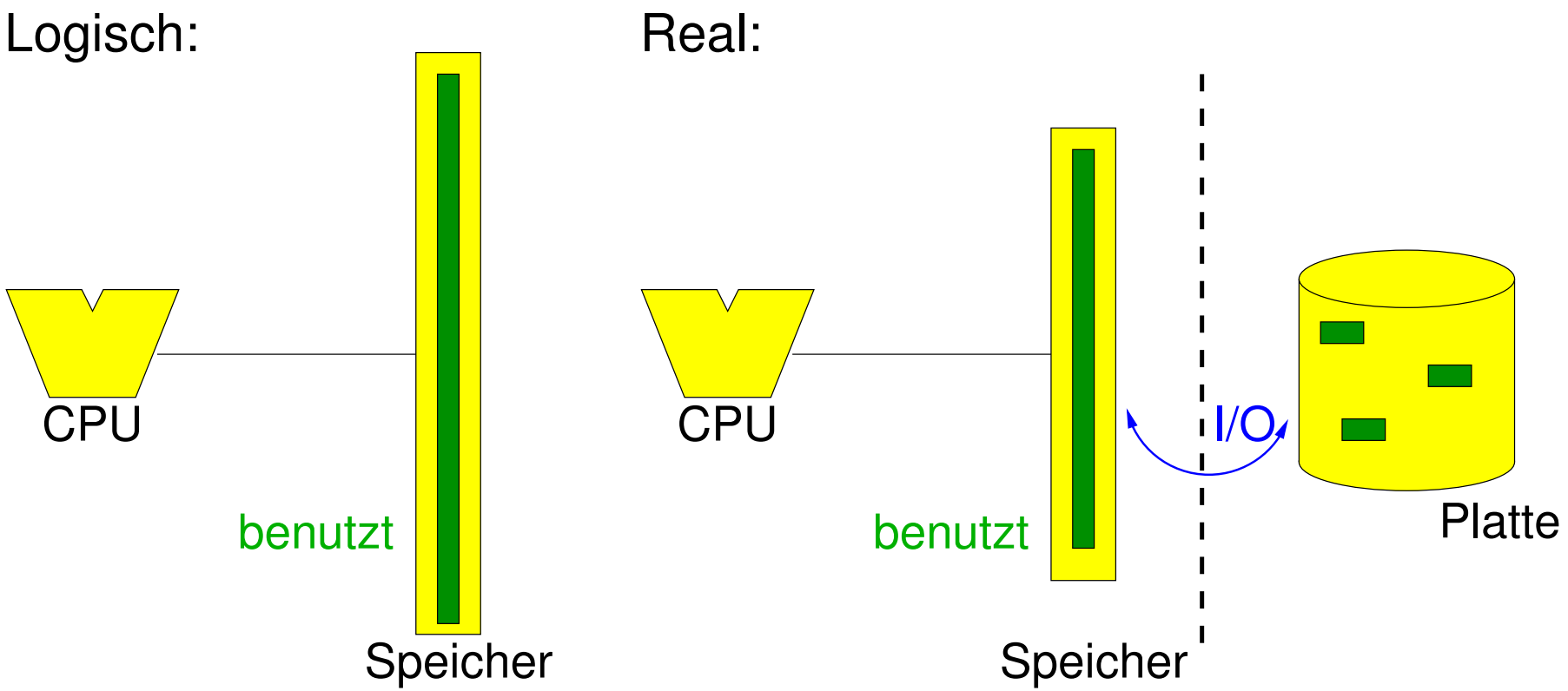
Logisch:



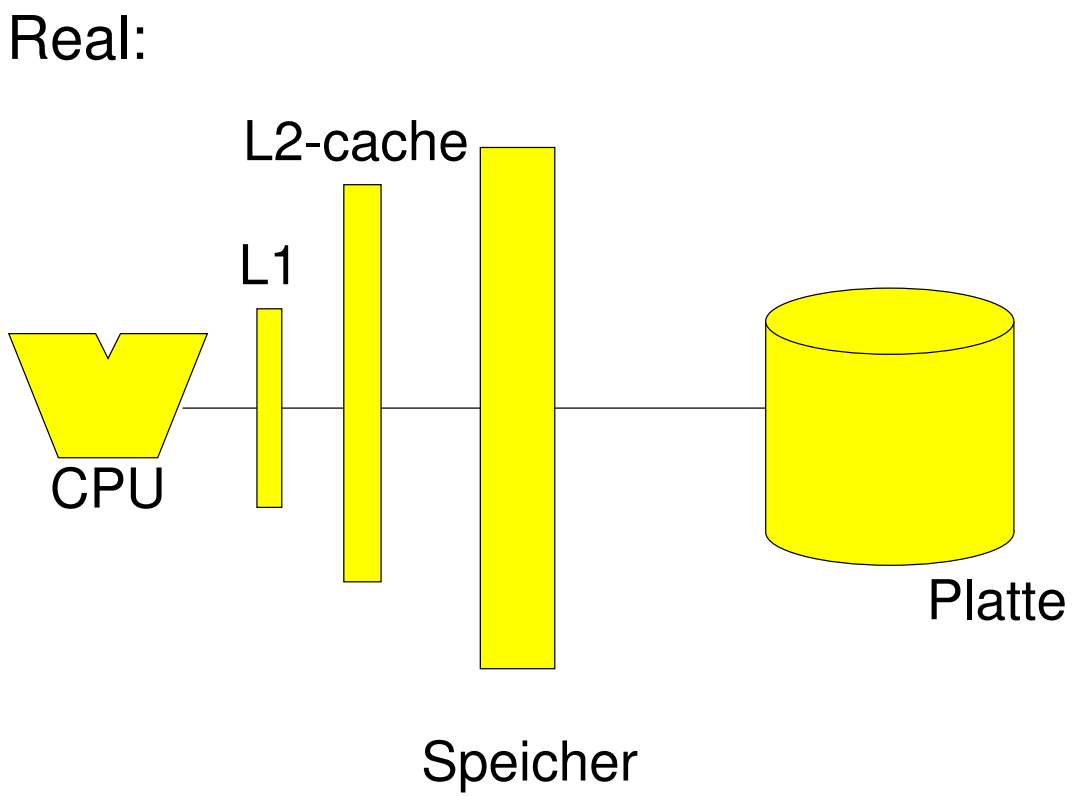
Speicherhierarchie



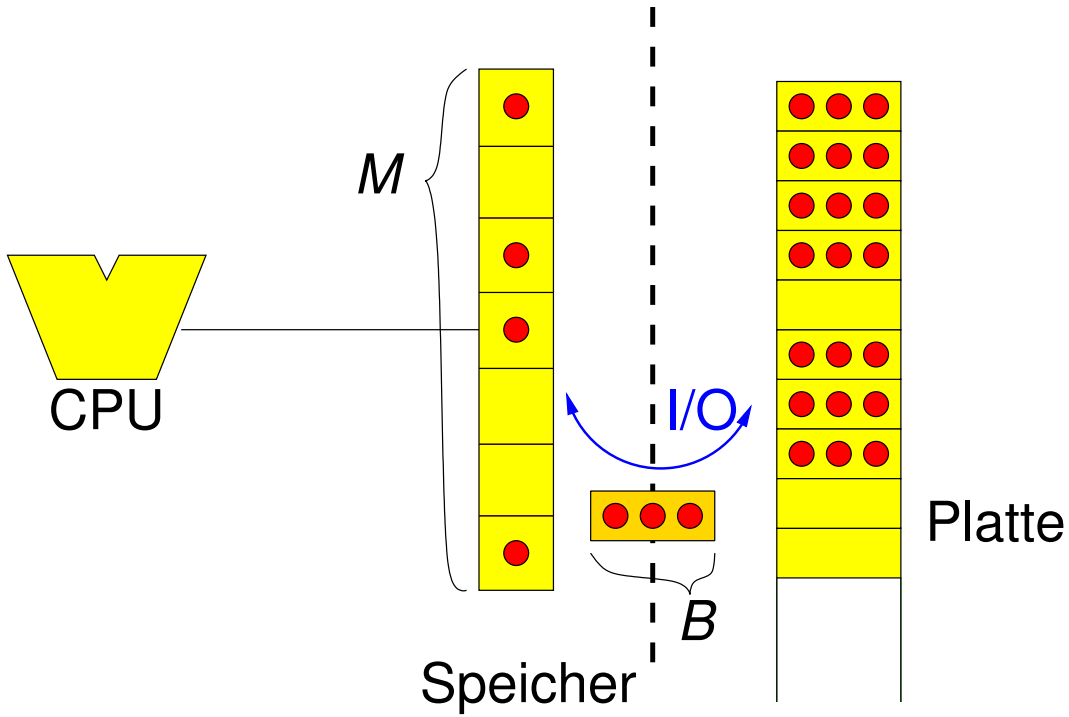
Speicherhierarchie



Speicherhierarchie



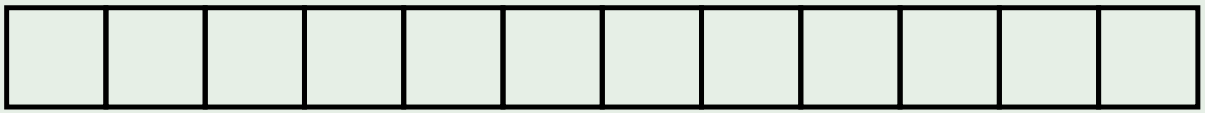
I/O-Modell



[Aggarwal, Vitter 1988]

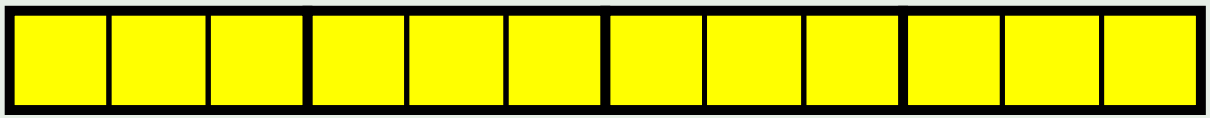
Beispiel: Zugriff

Beispiel



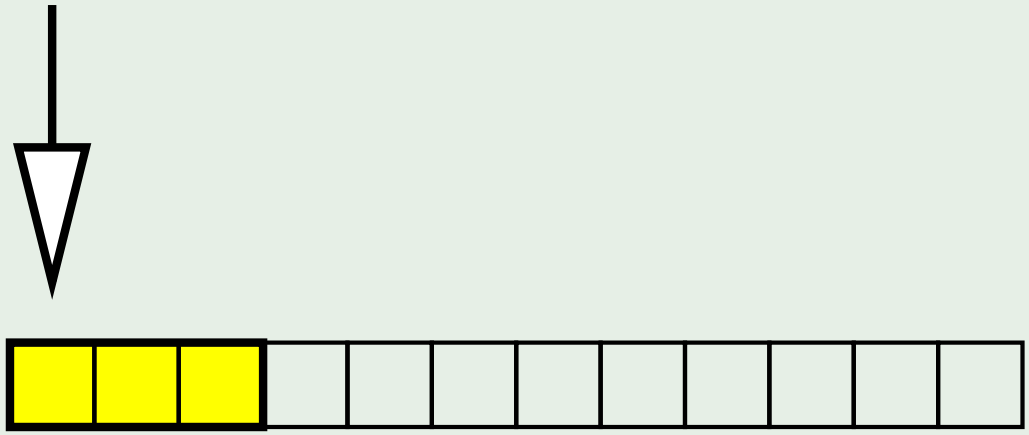
Beispiel: Zugriff

Beispiel



Beispiel: Zugriff

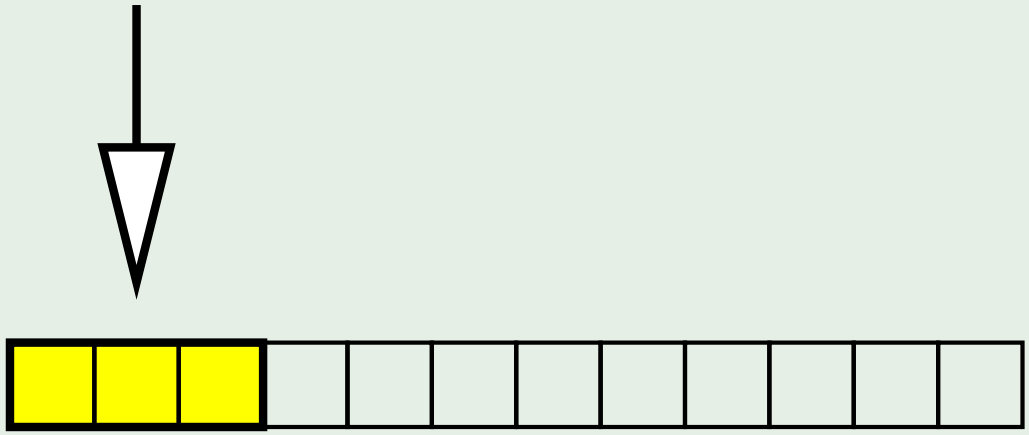
Beispiel



Scan

Beispiel: Zugriff

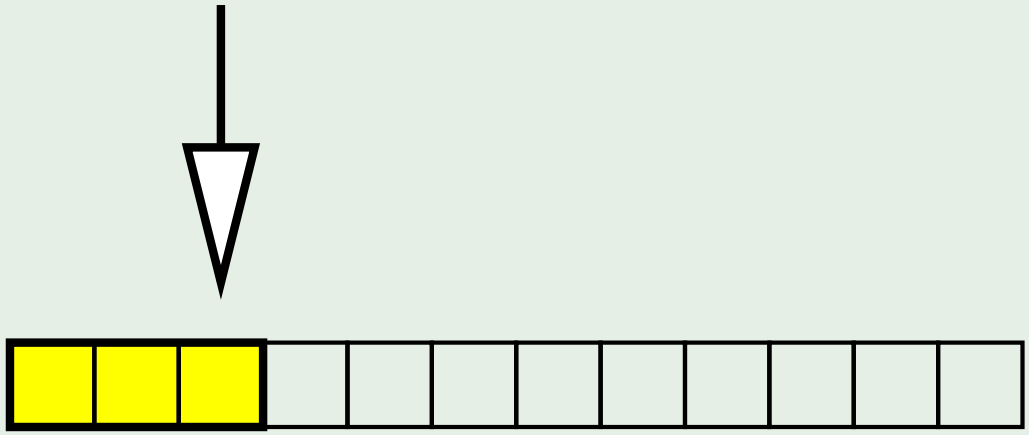
Beispiel



Scan

Beispiel: Zugriff

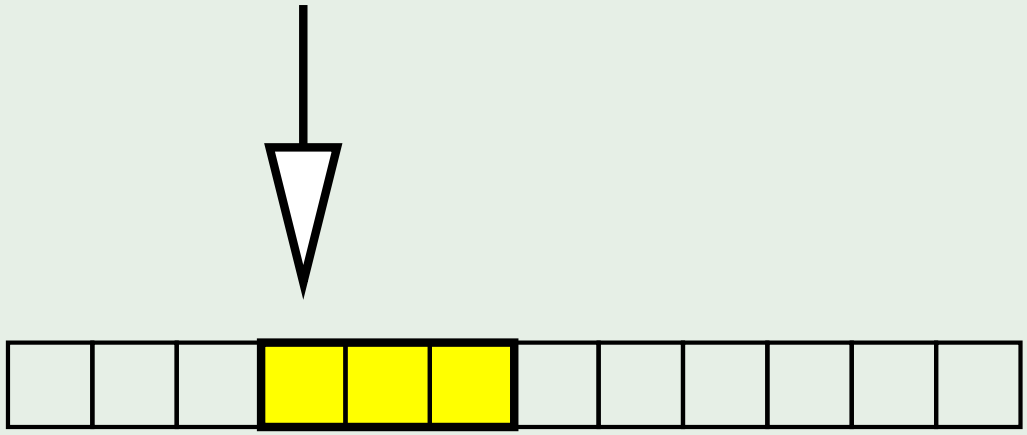
Beispiel



Scan

Beispiel: Zugriff

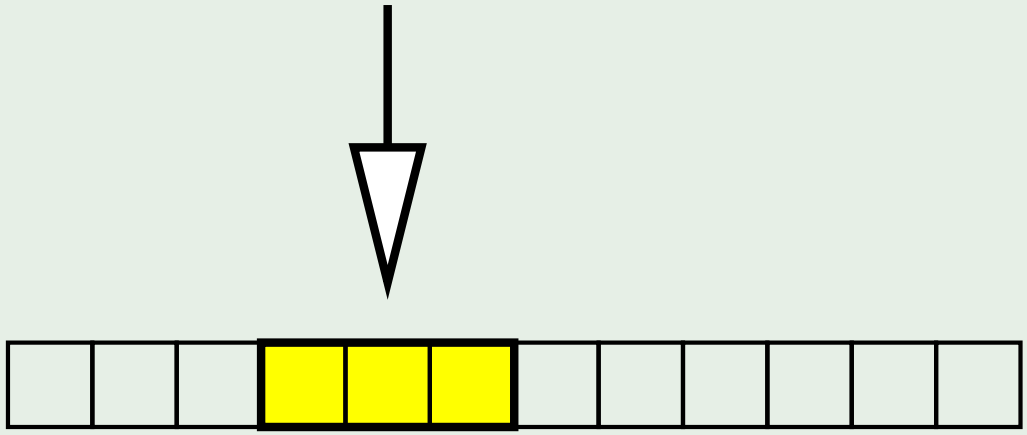
Beispiel



Scan

Beispiel: Zugriff

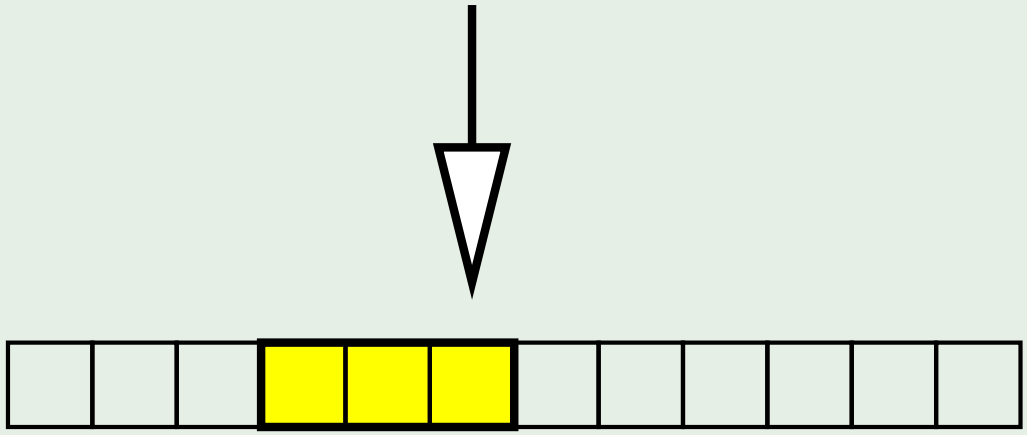
Beispiel



Scan

Beispiel: Zugriff

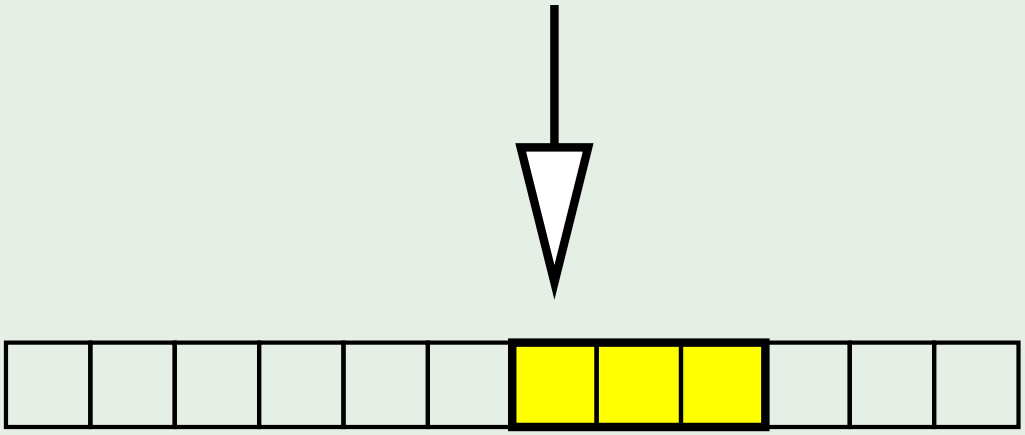
Beispiel



Scan

Beispiel: Zugriff

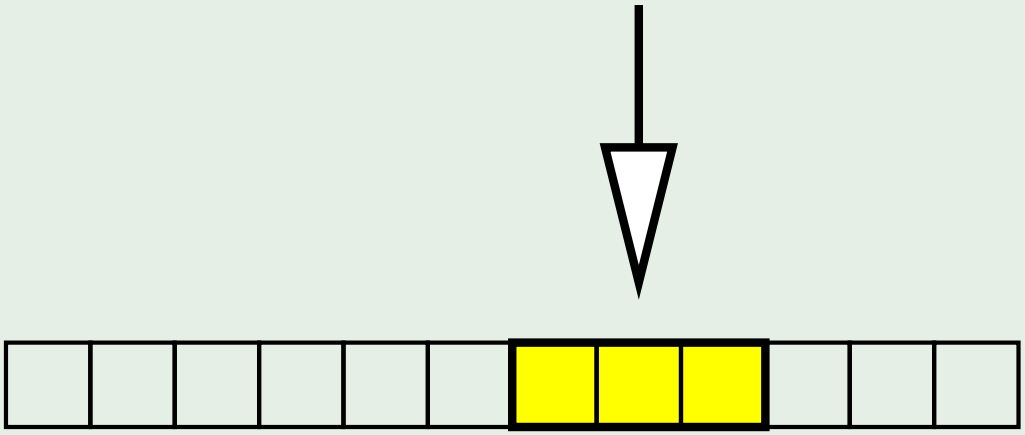
Beispiel



Scan

Beispiel: Zugriff

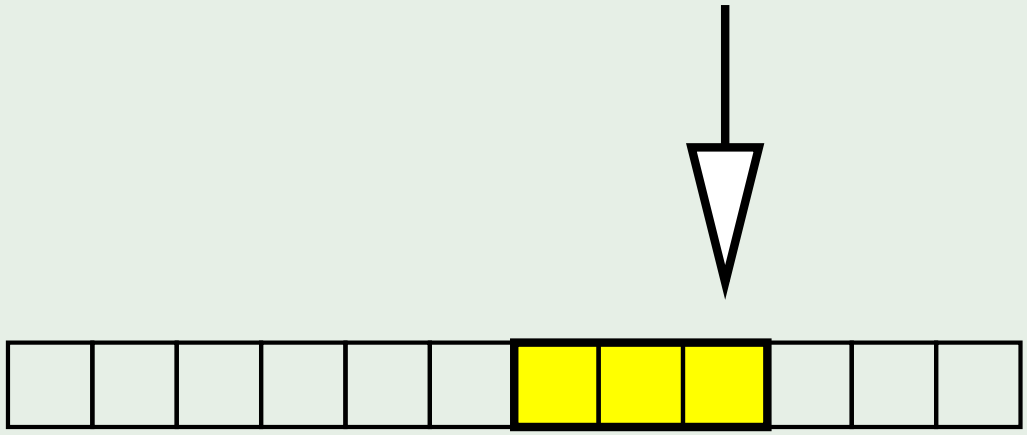
Beispiel



Scan

Beispiel: Zugriff

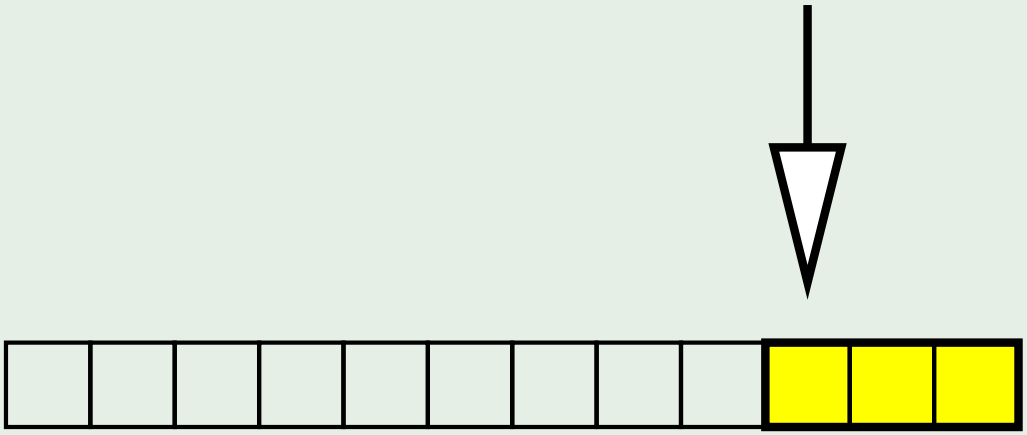
Beispiel



Scan

Beispiel: Zugriff

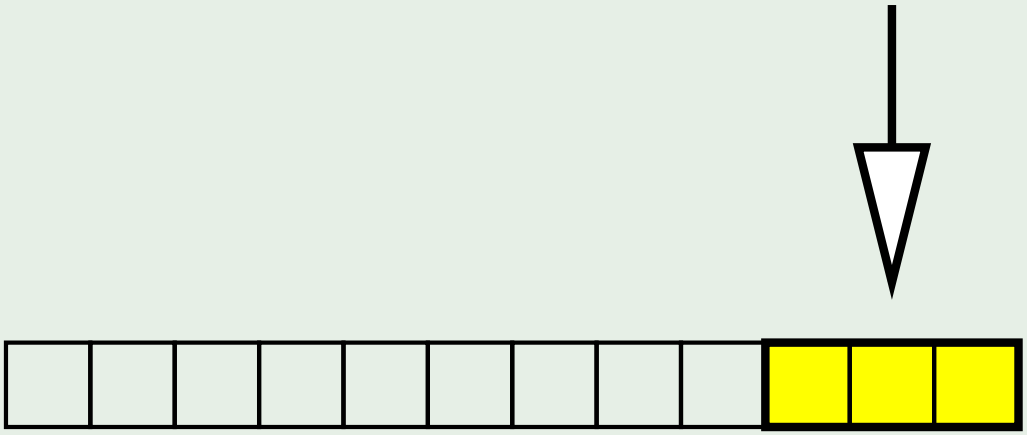
Beispiel



Scan

Beispiel: Zugriff

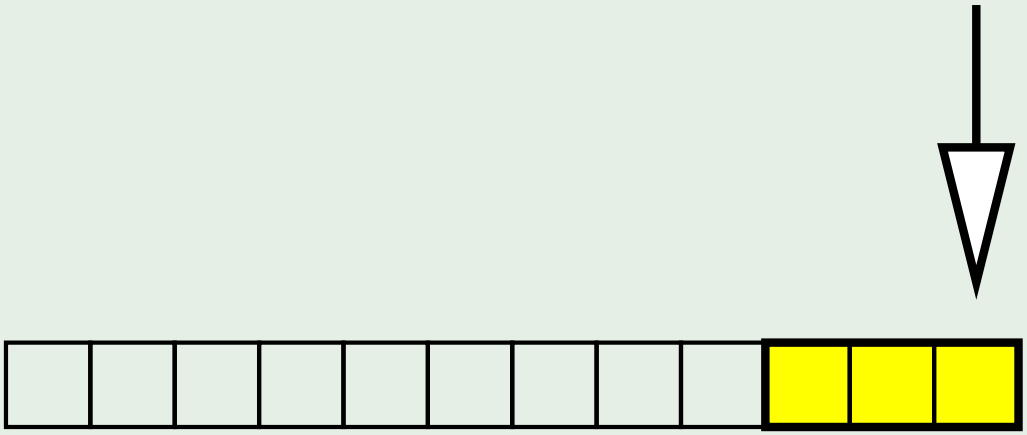
Beispiel



Scan

Beispiel: Zugriff

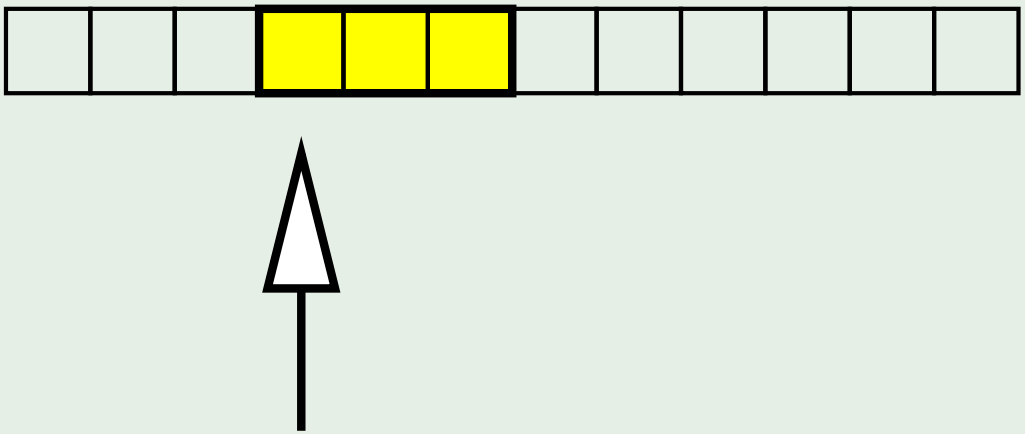
Beispiel



Scan

Beispiel: Zugriff

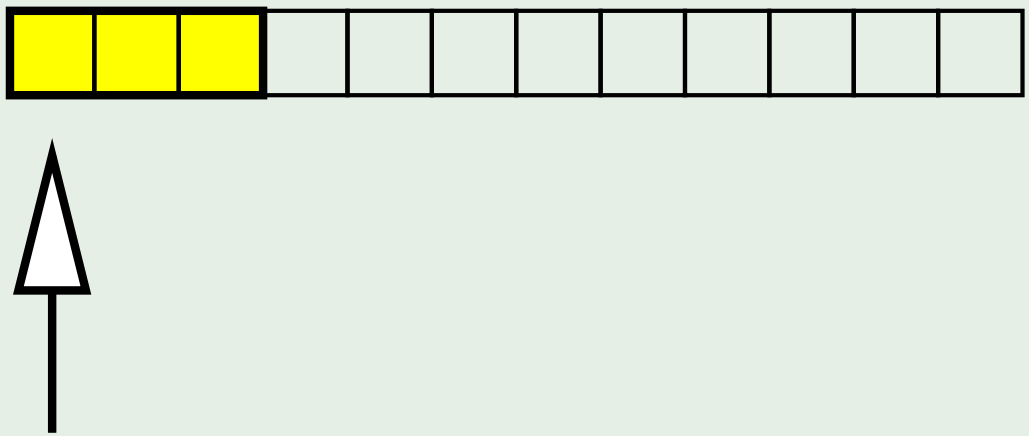
Beispiel



Random Access

Beispiel: Zugriff

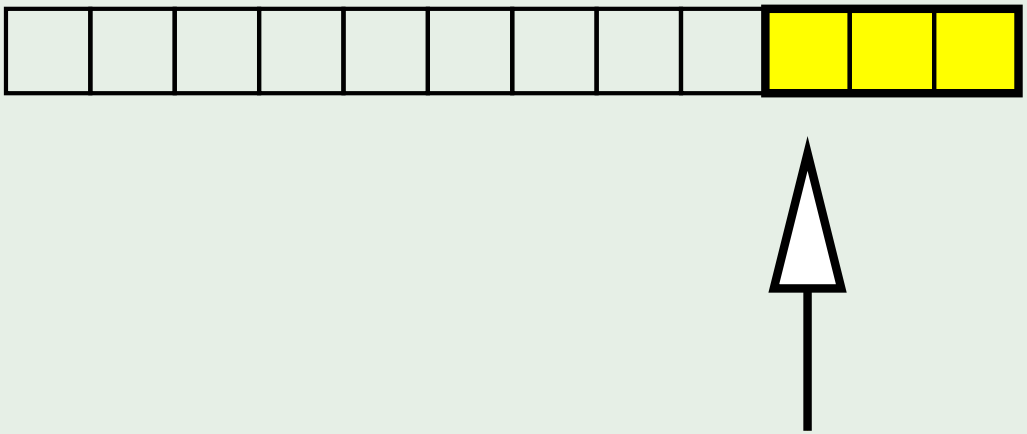
Beispiel



Random Access

Beispiel: Zugriff

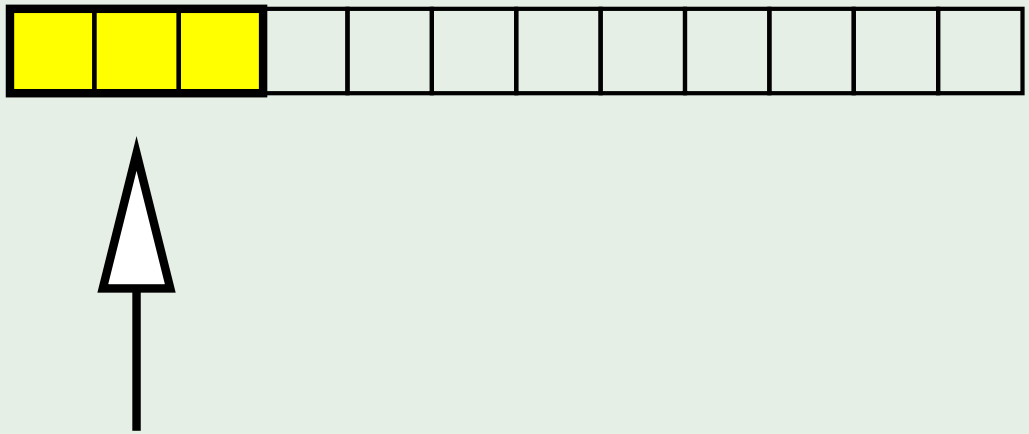
Beispiel



Random Access

Beispiel: Zugriff

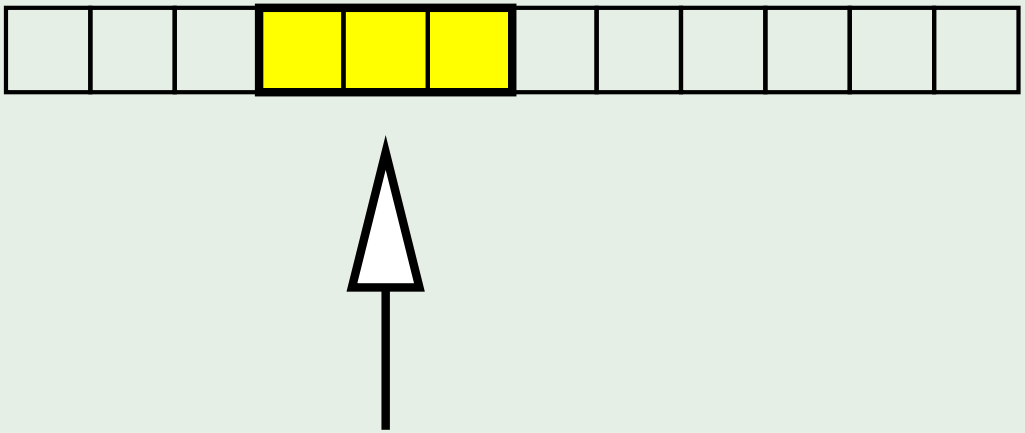
Beispiel



Random Access

Beispiel: Zugriff

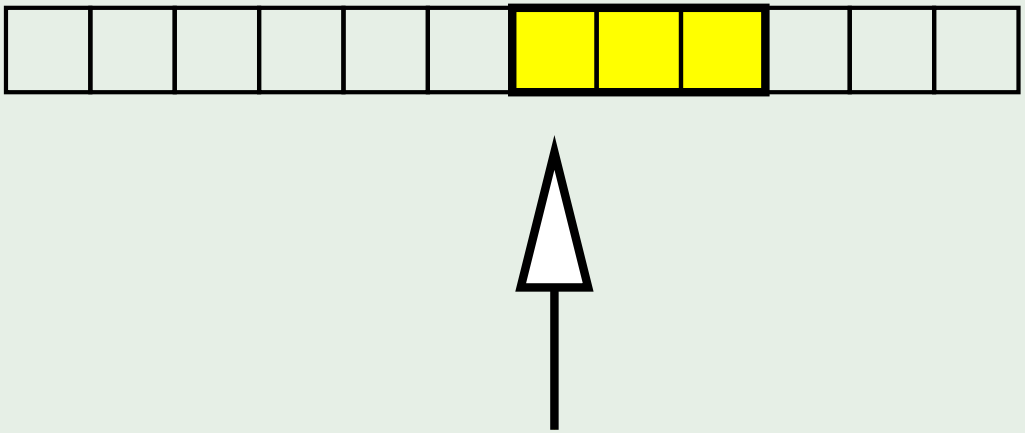
Beispiel



Random Access

Beispiel: Zugriff

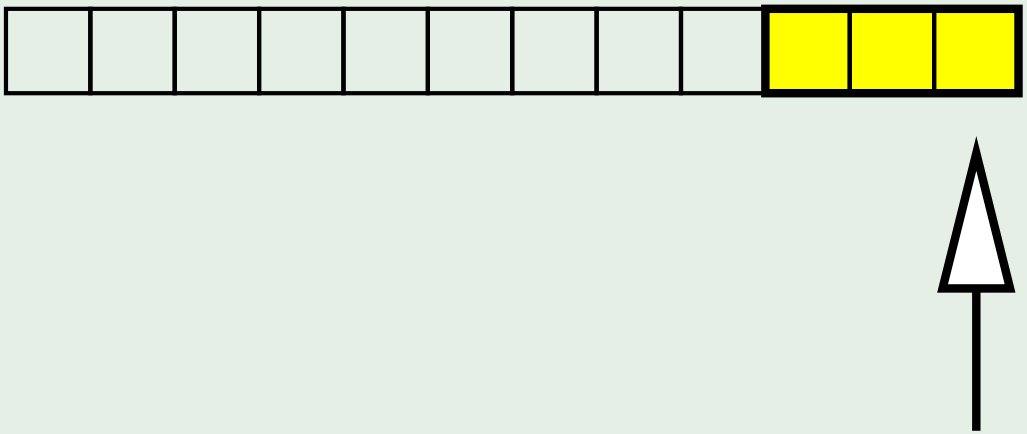
Beispiel



Random Access

Beispiel: Zugriff

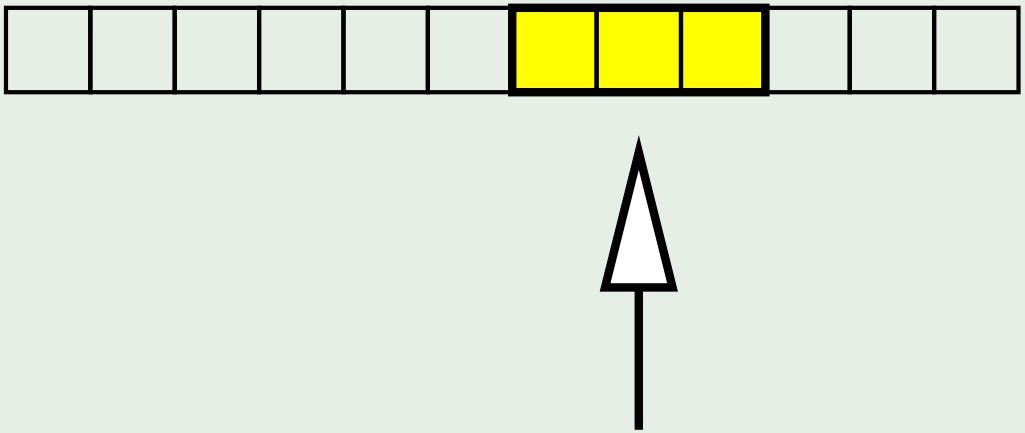
Beispiel



Random Access

Beispiel: Zugriff

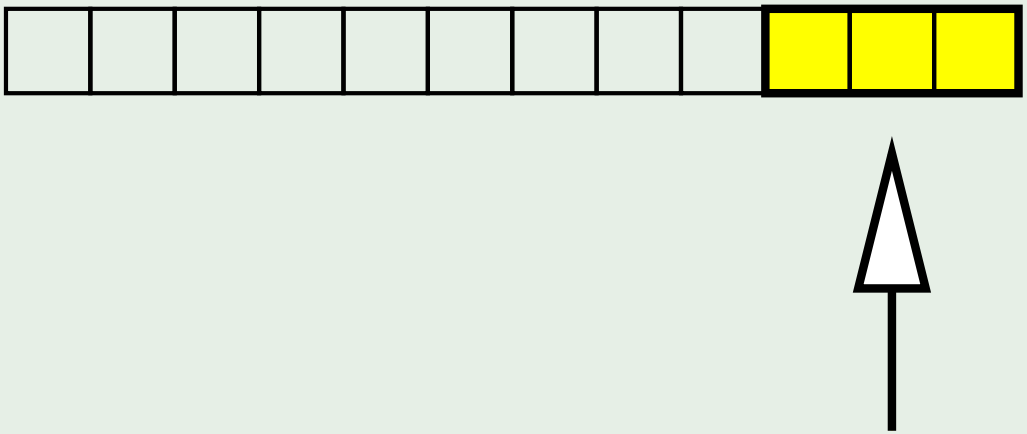
Beispiel



Random Access

Beispiel: Zugriff

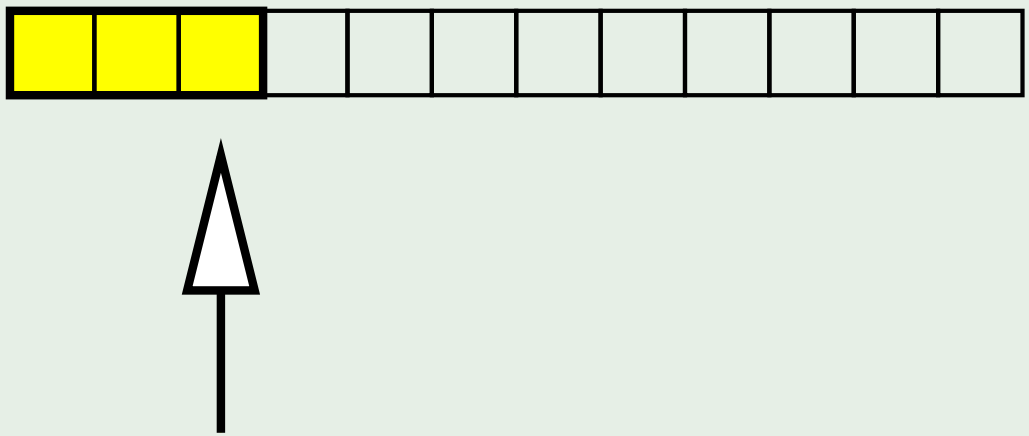
Beispiel



Random Access

Beispiel: Zugriff

Beispiel



Random Access

Inhalt der Vorlesung

- Modellbildung Externspeicher (I/O-model)
- Grundlegende Aufgaben:
  Algorithmen, Datenstrukturen, Untere Schranken
  - Sortieren
  - Permutieren
  - Suchen
- I/O-effiziente Algorithmen aus den Bereichen
  - Graphalgorithmen
  - Algorithmische Geometrie
  - Stringalgorithmen

Nebenbei Prinzipien des Designs I/O-effizienter Algorithmen

Darüberhinaus Programmierprojekte: Experimente und Details

Technische Universität München

Inhalt der Vorlesung

- Modellbildung Externspeicher (I/O-model)
- Grundlegende Aufgaben:
  Algorithmen, Datenstrukturen, Untere Schranken
  - Sortieren
  - Permutieren
  - Suchen
- I/O-effiziente Algorithmen aus den Bereichen
  - Graphalgorithmen
  - Algorithmische Geometrie
  - Stringalgorithmen

Nebenbei Prinzipien des Designs I/O-effizienter Algorithmen

Darüberhinaus Programmierprojekte: Experimente und Details

Technische Universität München

Inhalt der Vorlesung

- Modellbildung Externspeicher (I/O-model)
- Grundlegende Aufgaben:
  Algorithmen, Datenstrukturen, Untere Schranken
  - Sortieren
  - Permutieren
  - Suchen
- I/O-effiziente Algorithmen aus den Bereichen
  - Graphalgorithmen
  - Algorithmische Geometrie
  - Stringalgorithmen

Nebenbei Prinzipien des Designs I/O-effizienter Algorithmen

Darüberhinaus Programmierprojekte: Experimente und Details

Technische Universität München

Form der Vorlesung

- Fokus: Theoretischer Zugang / Verständnis
- Basierend auf
  - Grundlagen: LNCS-Sammelband (GI-Dagstuhl Seminar) “Algorithms for Memory Hierarchies”, Springer LNCS 2625.
  - Einzelne Themen: (aktuelle) Veröffentlichungen
  - Vorlesungen:
    Gerth Brodal und Rolf Fagerberg,
    Aarhus DK 1999, 2000, 2001, 2003
    zusammen mit Peter Widmayer, Zürich CH, 2004, 2005

Übung

Ab 1.5. Gero Greiner,
Raum und Zeitpunkt noch offen.

Technische Universität München

Form der Vorlesung

- Fokus: Theoretischer Zugang / Verständnis
- Basierend auf
  - Grundlagen: LNCS-Sammelband (GI-Dagstuhl Seminar) “Algorithms for Memory Hierarchies”, Springer LNCS 2625.
  - Einzelne Themen: (aktuelle) Veröffentlichungen
  - Vorlesungen:
    Gerth Brodal und Rolf Fagerberg,
    Aarhus DK 1999, 2000, 2001, 2003
    zusammen mit Peter Widmayer, Zürich CH, 2004, 2005

Übung

Ab 1.5. Gero Greiner,
Raum und Zeitpunkt noch offen.

Technische Universität München

Form der Vorlesung

- Fokus: Theoretischer Zugang / Verständnis
- Basierend auf
  - Grundlagen: LNCS-Sammelband (GI-Dagstuhl Seminar) “Algorithms for Memory Hierarchies”, Springer LNCS 2625.
  - Einzelne Themen: (aktuelle) Veröffentlichungen
  - Vorlesungen:
    Gerth Brodal und Rolf Fagerberg,
    Aarhus DK 1999, 2000, 2001, 2003
    zusammen mit Peter Widmayer, Zürich CH, 2004, 2005

Übung

Ab 1.5. Gero Greiner,
Raum und Zeitpunkt noch offen.

Technische Universität München

Form der Vorlesung

- Fokus: Theoretischer Zugang / Verständnis
- Basierend auf
  - Grundlagen: LNCS-Sammelband (GI-Dagstuhl Seminar) “Algorithms for Memory Hierarchies”, Springer LNCS 2625.
  - Einzelne Themen: (aktuelle) Veröffentlichungen
  - Vorlesungen:
    Gerth Brodal und Rolf Fagerberg,
    Aarhus DK 1999, 2000, 2001, 2003
    zusammen mit Peter Widmayer, Zürich CH, 2004, 2005

Übung

Ab 1.5. Gero Greiner,
Raum und Zeitpunkt noch offen.

Didaktische Form

- Verstehen durch Erarbeiten
  - Genauso viel Übung wie Vorlesung (wenn nötig mehr)
  - Selber Programmieren, Experimentieren
  - Klassische Übungen
- Lebt von Ihrer Mitarbeit
- offen für Ihre Anregungen und Ihre Kritik

Didaktische Form

- Verstehen durch Erarbeiten
  - Genauso viel Übung wie Vorlesung (wenn nötig mehr)
  - Selber Programmieren, Experimentieren
  - Klassische Übungen
- Lebt von Ihrer Mitarbeit
- offen für Ihre Anregungen und Ihre Kritik

Didaktische Form

- Verstehen durch Erarbeiten
  - Genauso viel Übung wie Vorlesung (wenn nötig mehr)
  - Selber Programmieren, Experimentieren
  - Klassische Übungen
- Lebt von Ihrer Mitarbeit
- offen für Ihre Anregungen und Ihre Kritik

Präsenzaufgabe — 5 Minuten Pause

Aufgabe

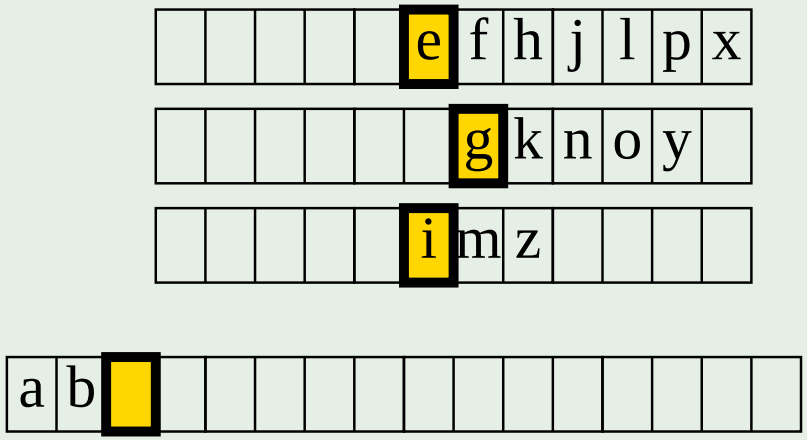
Entwerfen Sie ein Programm, das die Zeit für einen I/O bzw. cache-miss bestimmen kann.

k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

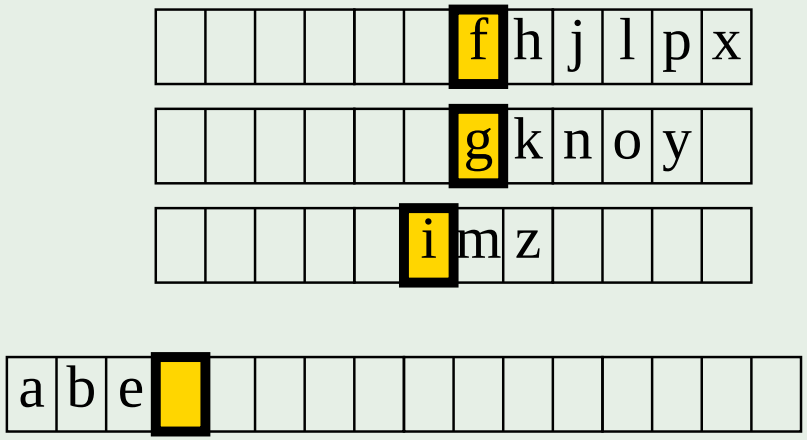


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

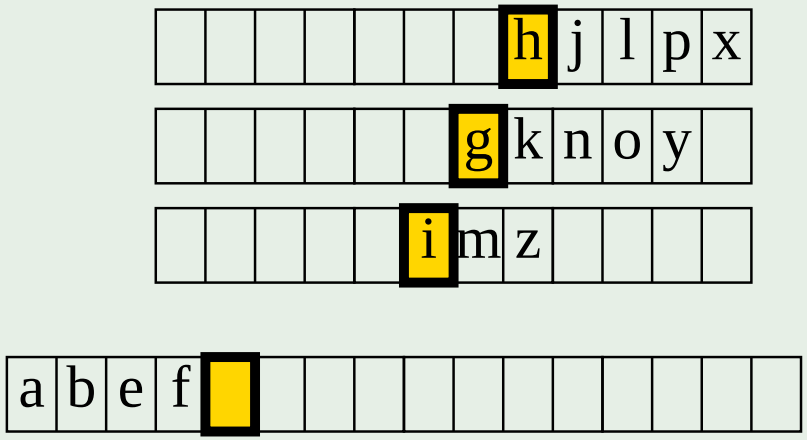


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

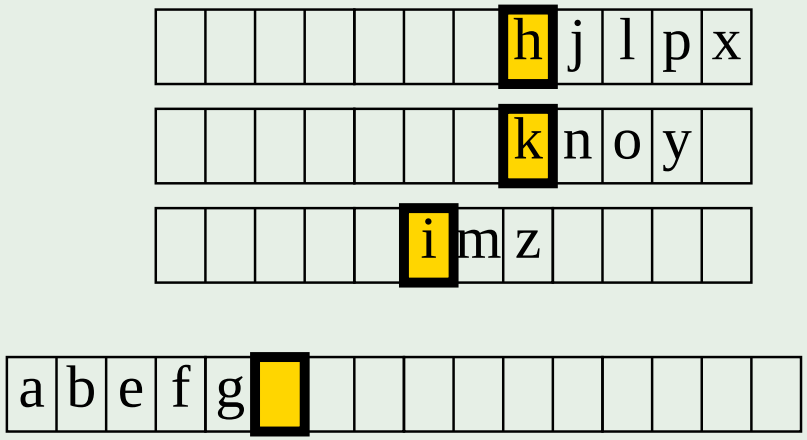


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

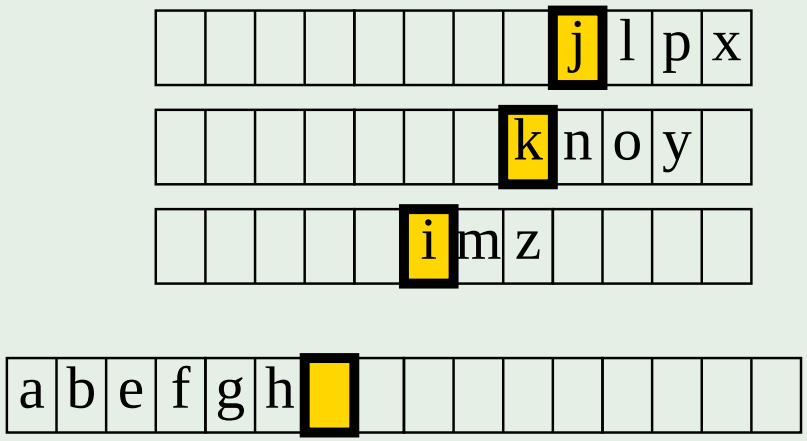


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

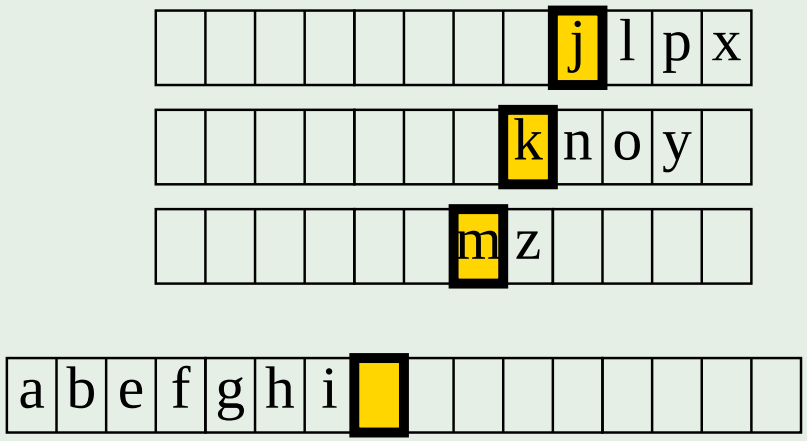


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

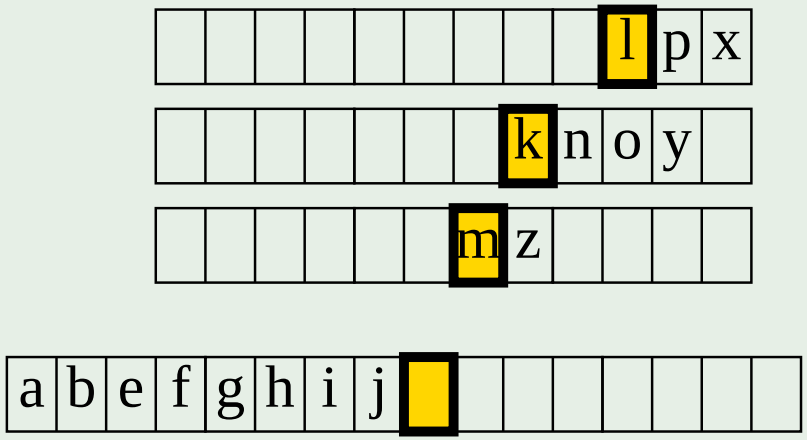


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

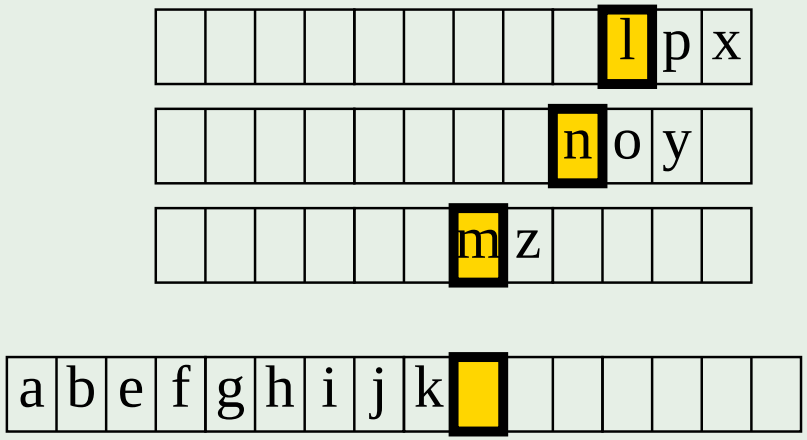


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

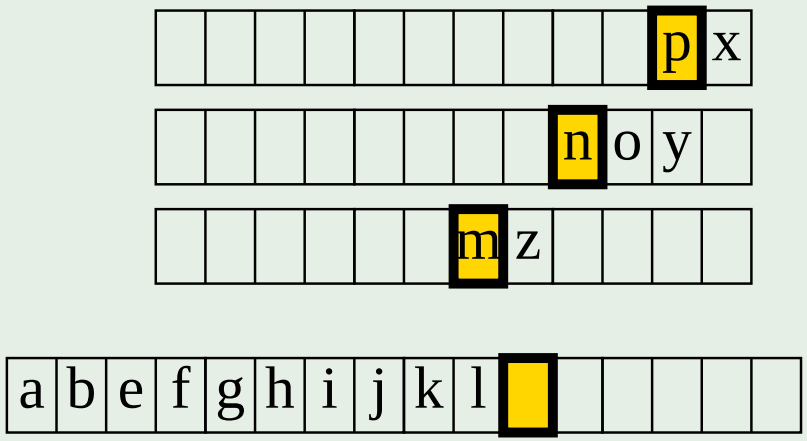


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

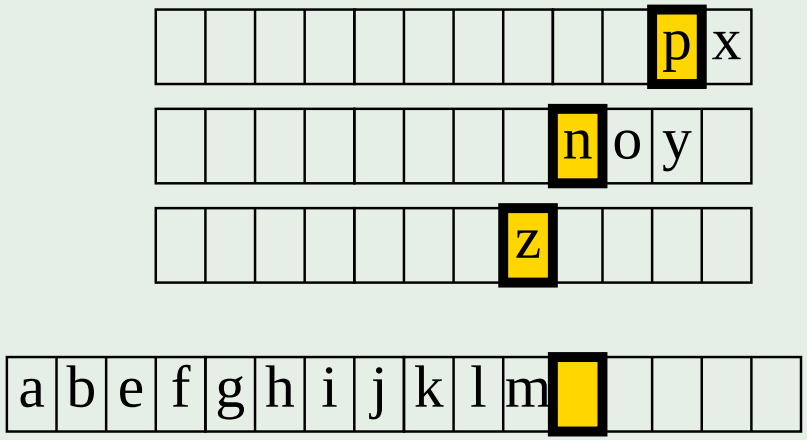


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

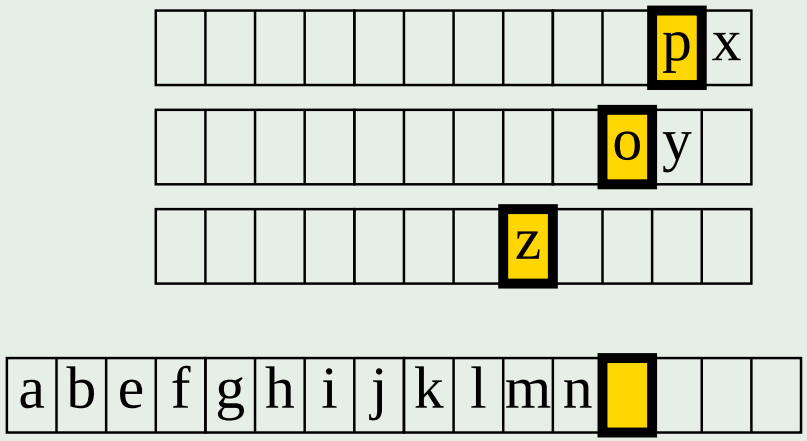


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

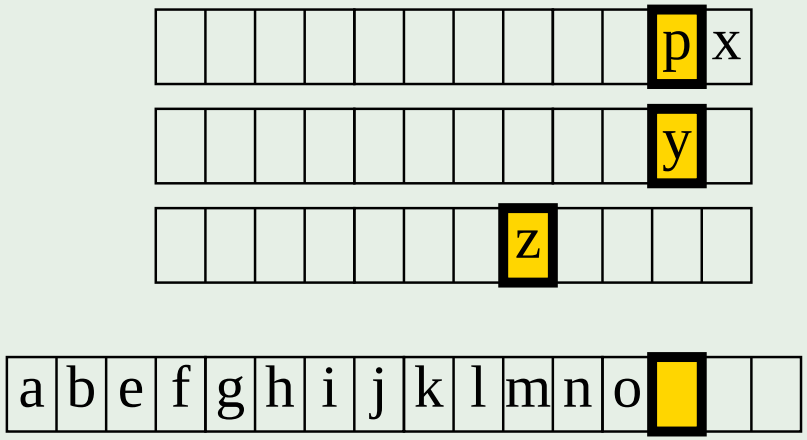


k-Wege Merge

Nur Verschmelzen

Beispiel

$k = 3$

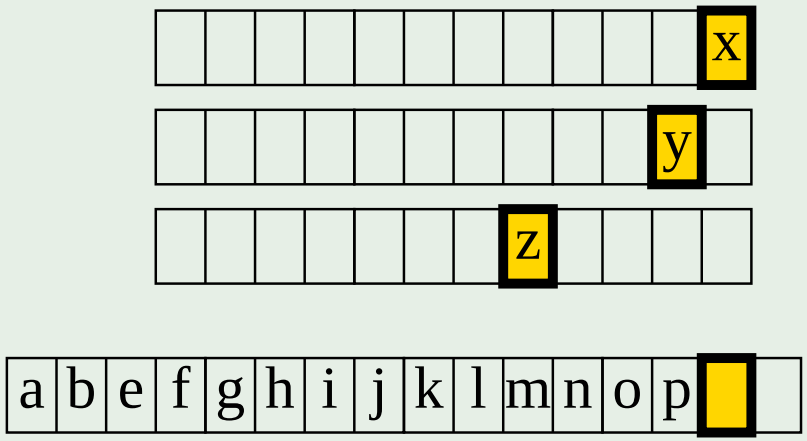


k-Wege Merge

Nur Verschmelzen

Beispiel

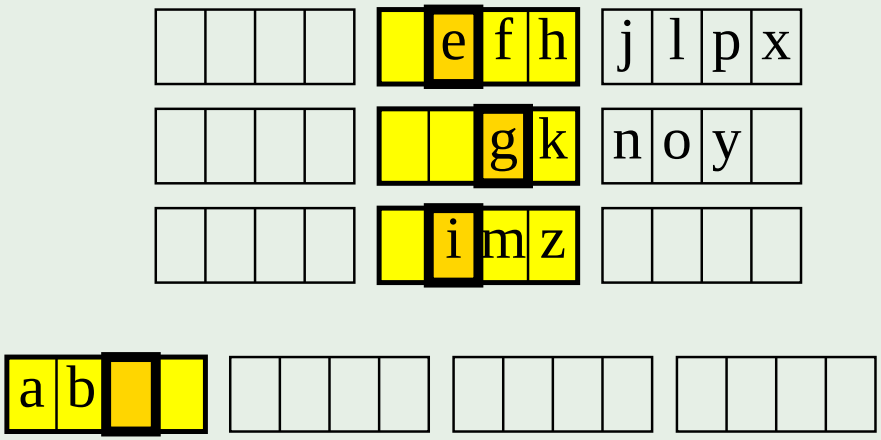
$k = 3$



k-Wege Merge
Verschmelzen und I/O

Beispiel

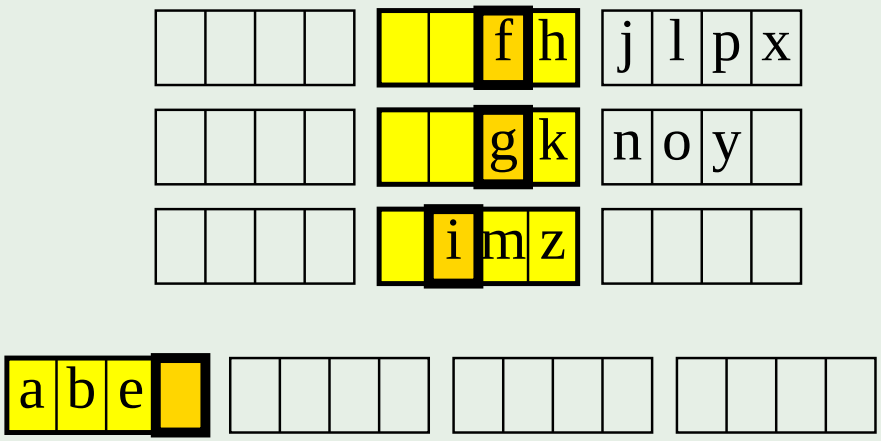
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

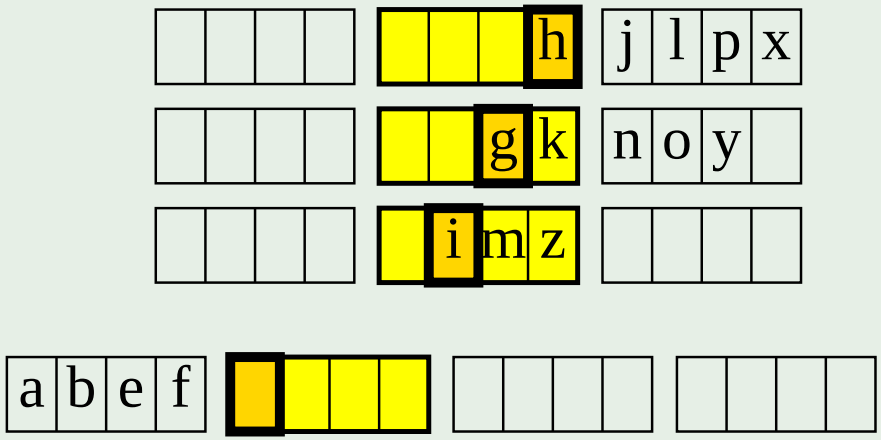
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

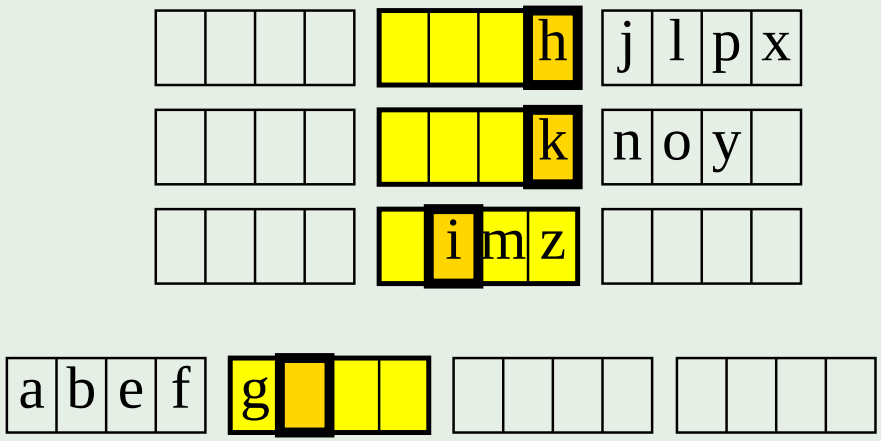
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

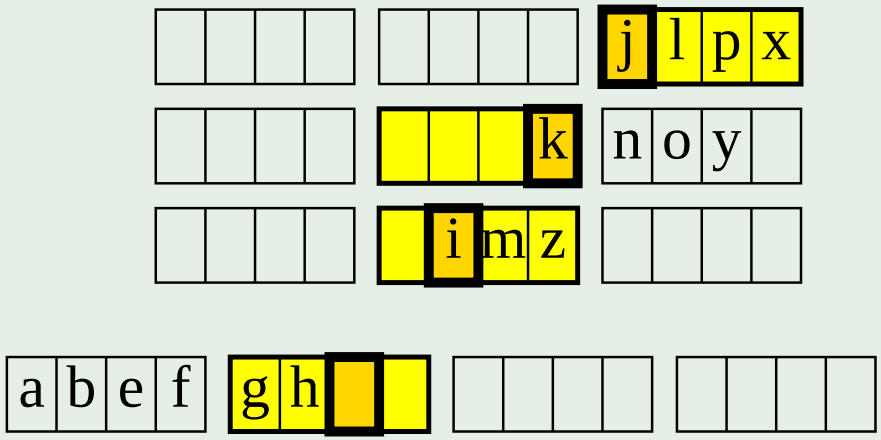
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

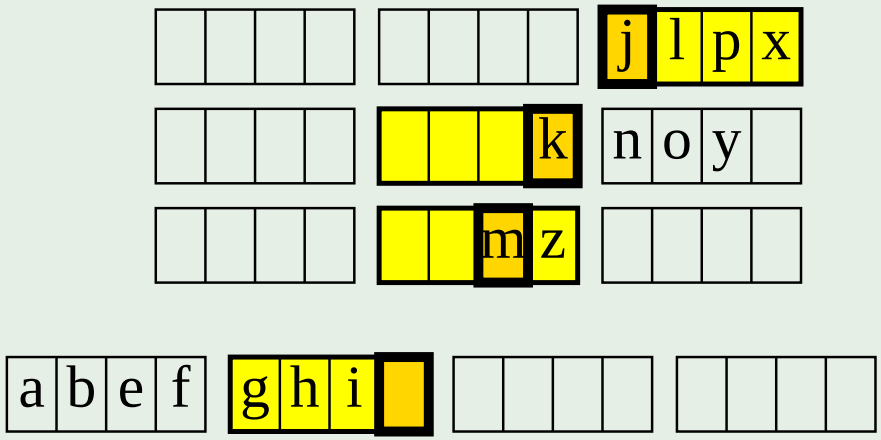
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

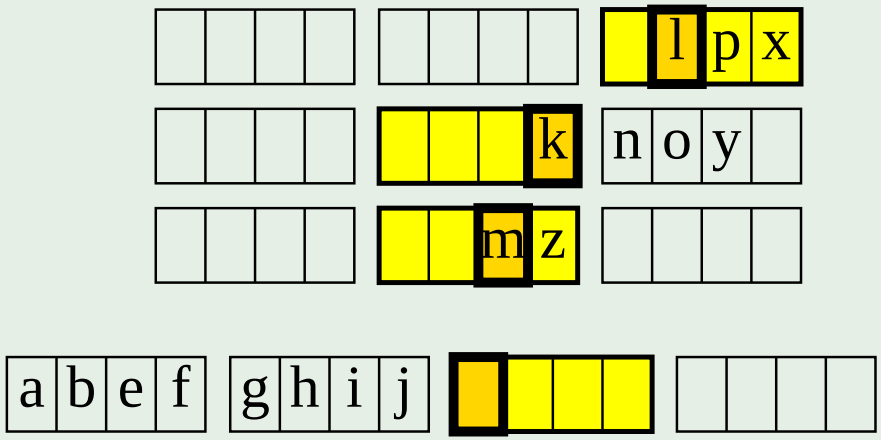
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

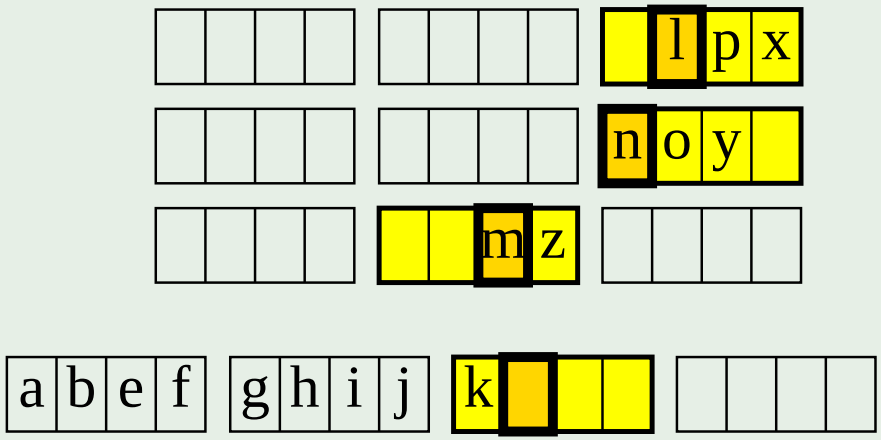
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

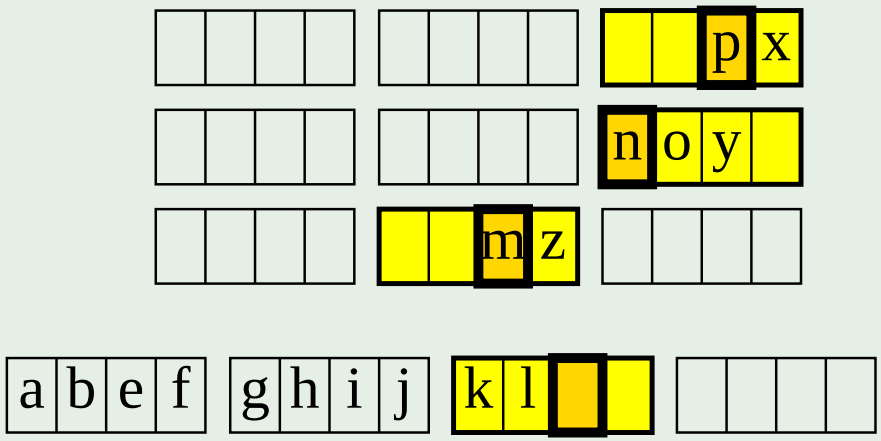
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

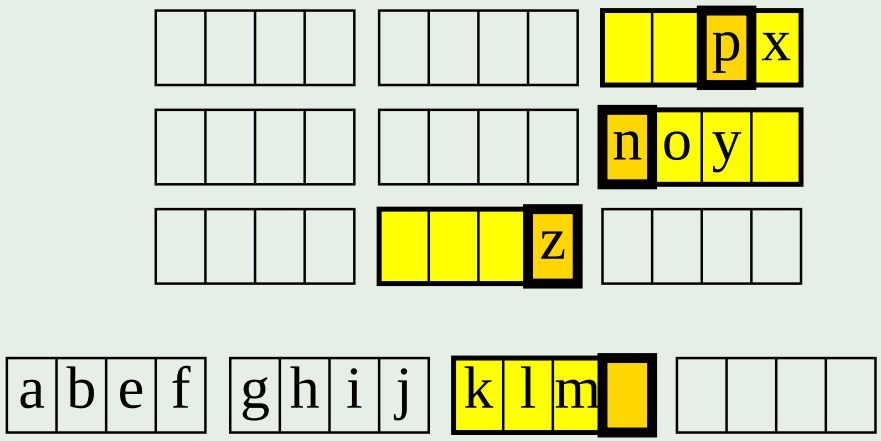
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

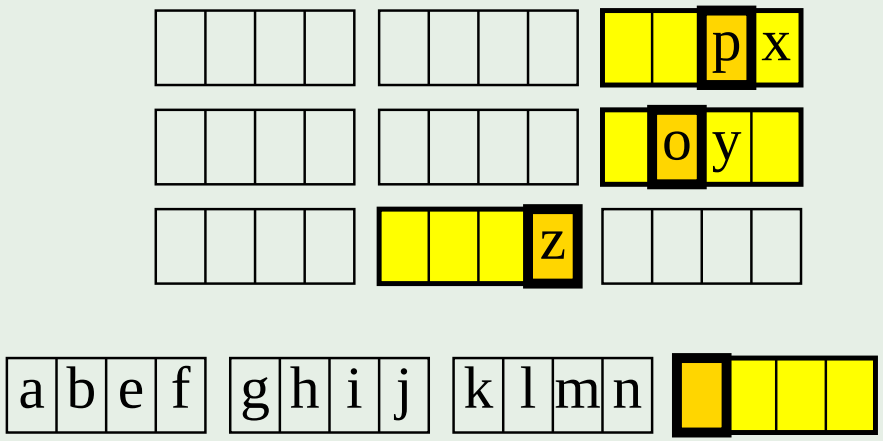
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

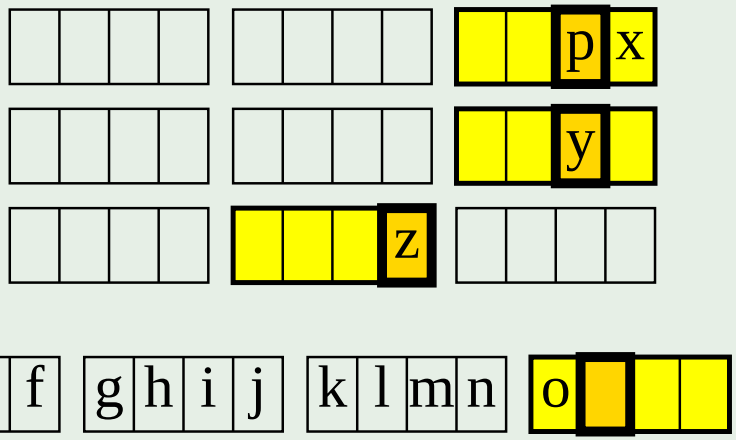
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

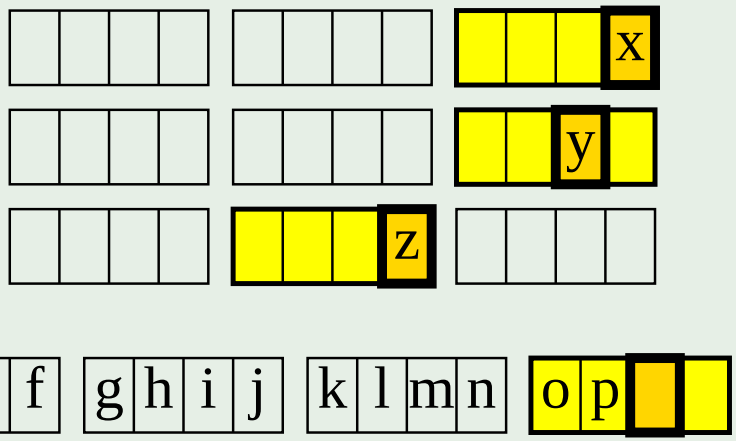
$k = 3, B = 4, M = 4B$



k-Wege Merge
Verschmelzen und I/O

Beispiel

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



Analyse k-Wege MergeSort

klassisches Divide-and-Conquer

Theorem

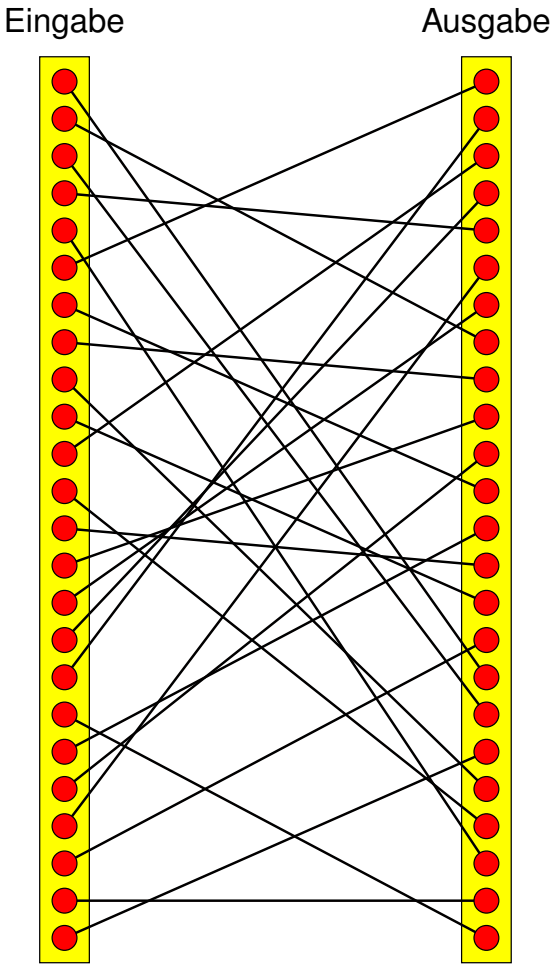
MergeSort nutzt $\mathcal{O}\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{M}\right)$ *I/Os*

	I/Os	# Runs
Basis: Je M Elemente Laden; Sortieren; Speichern;	$\mathcal{O}(N/B)$	N/M
Verschmelzen: je $k = M/B - 1$ sortierte Listen	$\mathcal{O}(N/B)$	$\times 1/k$

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

Technische Universität München

Permutieren
Eine zentrale, elementare Aufgabe



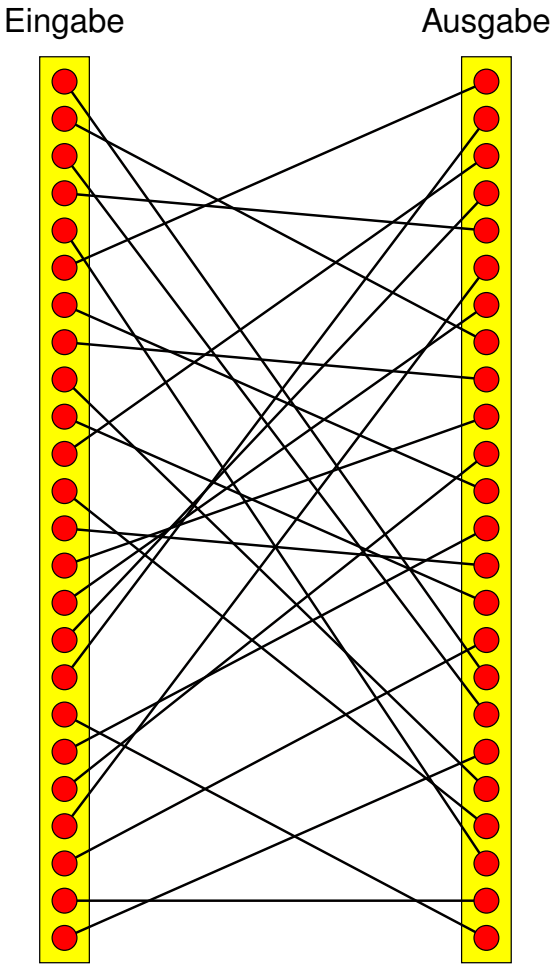
- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen
-

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

durch MergeSort

Permutieren

Eine zentrale, elementare Aufgabe



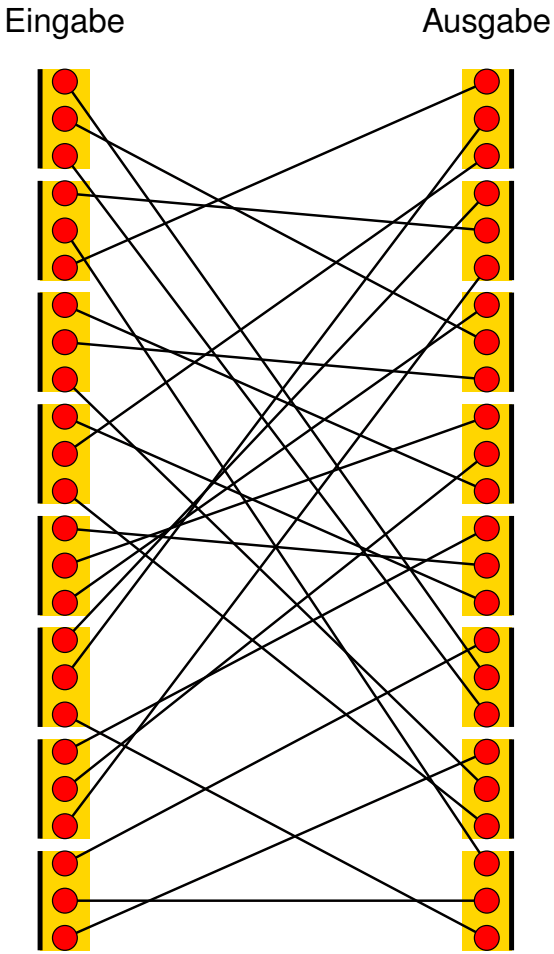
- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen
- $$\mathcal{O}\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Technische Universität München

Permutieren

Eine zentrale, elementare Aufgabe

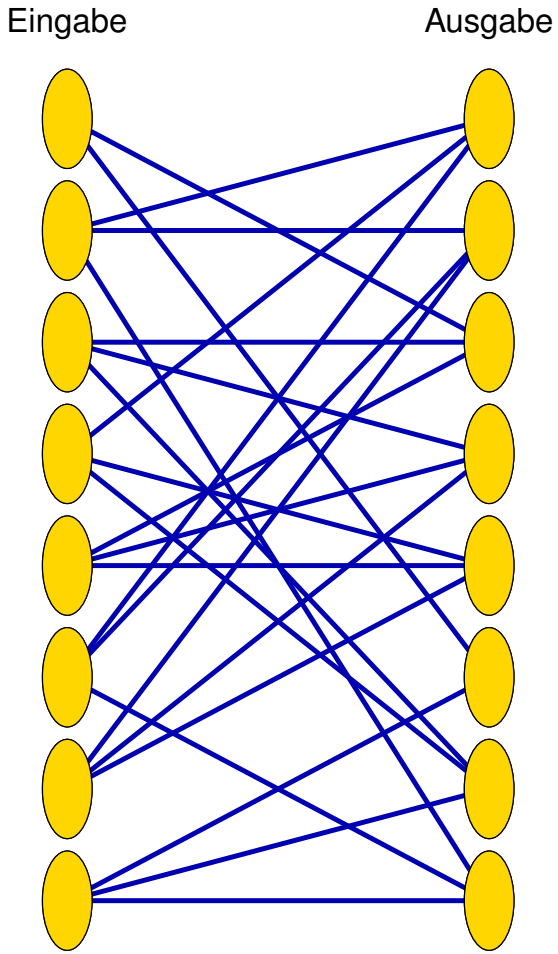


- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen
- $$\mathcal{O}\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{M}\right) \text{ I/O-Ops.}$$

durch MergeSort

Permutieren

Eine zentrale, elementare Aufgabe

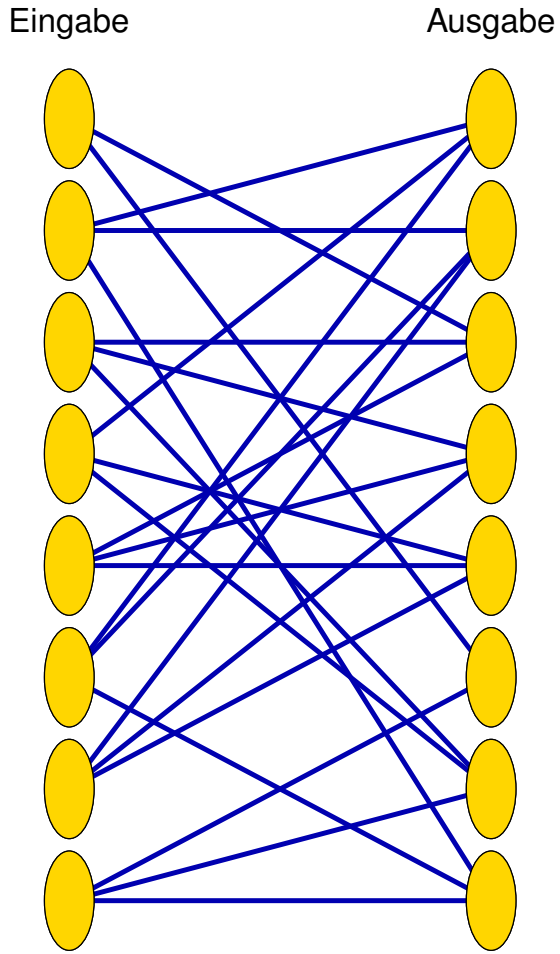


- $\mathcal{O}(N)$ CPU-Operationen
- $\mathcal{O}(N)$ I/O-Operationen
- $\mathcal{O}\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{M}\right)$ I/O-Ops.

durch MergeSort

Permutieren

Eine zentrale, elementare Aufgabe



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

  durch MergeSort