Hasso Plattner Institut

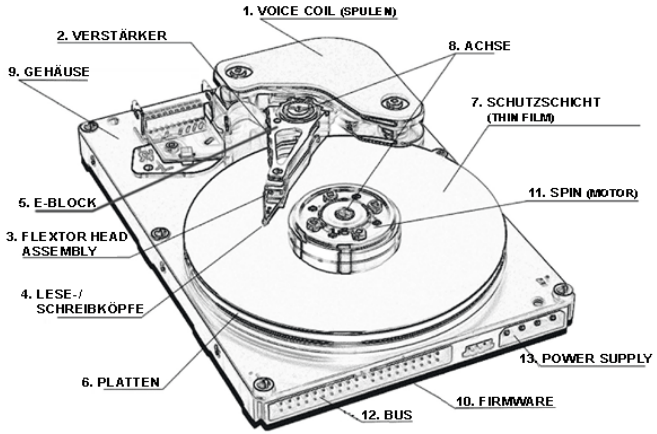
IT Systems Engineering | Universität Potsdam

Übung Datenbanksysteme II

Physische Speicherstrukturen

Maximilian Jenders

Folien basierend auf
Thorsten Papenbrock



Übung Datenbanksysteme II

- Übung
  - Maximilian Jenders (Maximilian.Jenders@hpi.de)
- Tutoren
  - Andreas Burmeister
  - Jaspar Mang
  - Julian Risch

- Mailingliste
  - fragen-dbs2-2014@hpi.uni-potsdam.de
  - Hilfe zur Selbsthilfe: Fragen gegenseitig beantworten
  - Tutoren und Mitarbeiter lesen mit
  - Keine Email erhalten? -> Melden

Übung Datenbanksysteme II

- Neues Abgabesystem
  - Abgabe *immer* in Zweiergruppen
  - Abgabe und Korrektur nur noch digital
  - Abgaben anderer Kurse bitte ignorieren
  - URL: http://www.dcl.hpi.uni-potsdam.de/submit/
- Neuer Übungsmodus
  - Mit Abgabe der 1. Übung verbindliche Wahl des Modus
  - Übungsbearbeitung verbindlich oder optional?
  - Wenn optional: Trotzdem Zweiergruppen finden
  - Wahl per Abgabe 1 Aufgabe 0 (verbindlich für alle)
  - Modus pro Gruppe gleich
- Übungsthemen ≠ Hausaufgabenthemen

Erinnerung:

Dezimal- vs. Binär-Einheiten

Dezimalpräfixe			Binärpräfixe	
Name (Symbol)	Bedeutung[G 1]	Unterschied (gerundet)	IEC-Name (IEC-Symbol)	Bedeutung
Kilobyte (kB)[G 2]	10^3 Byte = 1.000 Byte	2,40 %	Kibibyte (KiB)[G 3]	2^{10} Byte = 1.024 Byte
Megabyte (MB)	10^6 Byte = 1.000.000 Byte	4,86 %	Mebibyte (MiB)	2^{20} Byte = 1.048.576 Byte
Gigabyte (GB)	10^9 Byte = 1.000.000.000 Byte	7,37 %	Gibibyte (GiB)	2^{30} Byte = 1.073.741.824 Byte
Terabyte (TB)	10^{12} Byte = 1.000.000.000.000 Byte	9,95 %	Tebibyte (TiB)	2^{40} Byte = 1.099.511.627.776 Byte
Petabyte (PB)	10^{15} Byte = 1.000.000.000.000.000 Byte	12,6 %	Pebibyte (PiB)	2^{50} Byte = 1.125.899.906.842.624 Byte
Exabyte (EB)	10^{18} Byte = 1.000.000.000.000.000.000 Byte	15,3 %	Exbibyte (EiB)	2^{60} Byte = 1.152.921.504.606.846.976 Byte
Zettabyte (ZB)	10^{21} Byte = 1.000.000.000.000.000.000.000 Byte	18,1 %	Zebibyte (ZiB)	2^{70} Byte = 1.180.591.620.717.411.303.424 Byte
Yottabyte (YB)	10^{24} Byte = 1.000.000.000.000.000.000.000.000 Byte	20,9 %	Yobibyte (YiB)	2^{80} Byte = 1.208.925.819.614.629.174.706.176 Byte

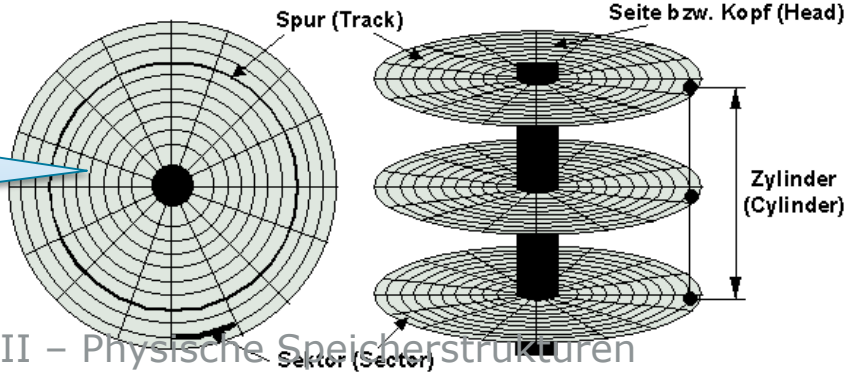
1. ↑ SI-Präfixe sind nur für SI-Einheiten standardisiert; Byte ist keine SI-Einheit
2. ↑ wird gelegentlich mit „KB“ abgekürzt
3. ↑ wird gelegentlich mit „KB“ abgekürzt, um den Unterschied zu „kB“ zu kennzeichnen (nicht standardisiert)

Physische Speicherstrukturen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5
- Rotationsgeschwindigkeit: 5000 U/min
- Lesekopfbewegung über n Spuren: (1 + 0,002 · n) ms
- Blockgröße: 1024 Byte
- Lücken pro Spur: 10 %

0,5 Start
0,002 · n Bewegung
0,5 Stop

Aufteilung der Sektoren
auf den Spuren muss
nicht tortenförmig sein!
→ evtl. außen mehr Sektoren



Aufgabe 1:

Speicherkapazität berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5

a) Gesucht: Kapazität einer Spur

K_{Spur} = Sektorgröße · Sektoren pro Spur = 512 Byte · 64 = 32 KiB

b) Gesucht: Kapazität einer Oberfläche

$K_{Oberfläche} = K_{Spur}$ · Spuren pro Oberfläche = 32 KiB · 2048 = 64 MiB

c) Gesucht: Kapazität der Festplatte

$K_{Platte} = K_{Oberfläche}$ · Anzahl Oberflächen = 64 MiB · 2 · 5 = 640 MiB

Aufgabe 1:

Speicherkapazität berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5

d) Gesucht: Anzahl Zylinder auf dieser Festplatte

2048 (= Anzahl Spuren)

e) Gesucht: Sind die folgenden Blockgrößen zulässig?

- 256 Byte — Nein: Blöcke können nicht kleiner als 1 Sektor sein
- 1.024 Byte — Ja: Block umfasst genau 2 Sektoren
- 51.200 Byte — Nein: Pro Spur gibt es nur 64 · 512 = 32.768 Byte

Aufgabe 2:

Latenzzeiten berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Rotationsgeschwindigkeit: 5000 U/min
- Lesekopfbewegung über n Spuren: (1 + 0,002 · n) ms
- Blockgröße: 1024 Byte
- Lücken pro Spur: 10 %

a) Gesucht: Minimale, maximale und durchschnittliche Latenzzeit zum Lesen eines Blocks

Latenzzeit = Kommunikation + Seek + Rotation + Transfer

vernachlässigbar

für alle drei Fälle gleich

Aufgabe 2:
Latenzzeiten berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Rotationsgeschwindigkeit: 5000 U/min
- Lesekopfbewegung über n Spuren: (1 + 0,002 · n) ms
- Blockgröße: 1024 Byte
- Lücken pro Spur: 10 %

a) Gesucht: Minimale, maximale und durchschnittliche Latenzzeit zum Lesen eines Blocks

Latenzzeit = Kommunikation + Seek + Rotation + Transfer

Rotationszeit = 1 U / Rotationsgeschw. = 1 / 5000 min = 12 ms

Sektorwinkel = 360° · 90 % / 64 = 5,0625°
Lückenwinkel = 360° · 10 % / 64 = 0,5625°

Blockwinkel = 2 Sektorlänge + 1 Lückenlänge (2. Lücke ist ja egal)
= 2 · 5,0625° + 1 · 0,5625° = 10,6875°

$L_{Transfer}$ = 10,6875° / 360° · 12 ms = 0,35625 ms

Aufgabe 2:
Latenzzeiten berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Rotationsgeschwindigkeit: 5000 U/min
- Lesekopfbewegung über n Spuren: $(1 + 0{,}002 \cdot n)$ ms
- Blockgröße: 1024 Byte
- Lücken pro Spur: 10 %

a) Gesucht: Minimale, maximale und durchschnittliche Latenzzeit zum Lesen eines Blocks

Latenzzeit = Kommunikation + Seek + Rotation + Transfer

$L_{Transfer\_min}$ = 0,35625 ms (= $L_{Transfer}$)
L_{Seek_min} = 0 ms (Lesekopf auf passender Spur)
$L_{Rotation\_min}$ = 0 ms (Lesekopf genau vor dem Block)

L_{min} = 0,35625 ms + 0 ms + 0 ms = 0,35625 ms

Aufgabe 2:
Latenzzeiten berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Rotationsgeschwindigkeit: 5000 U/min
- Lesekopfbewegung über n Spuren: $(1 + 0{,}002 \cdot n)$ ms
- Blockgröße: 1024 Byte
- Lücken pro Spur: 10 %

a) Gesucht: Minimale, maximale und durchschnittliche Latenzzeit zum Lesen eines Blocks

Latenzzeit = Kommunikation + Seek + Rotation + Transfer

$L_{Transfer\_max} = 0{,}35625$ ms (= $L_{Transfer}$)
$L_{Seek\_max} = (1 + 0{,}002 \cdot 2048)$ ms $= 5{,}096$ ms (über alle Spuren)
$L_{Rotation\_max} = 12$ ms (volle Rotation)

$L_{max} = 0{,}35625$ ms $+ 5{,}096$ ms $+ 12$ ms $= 17{,}45225$ ms

Aufgabe 2:
Latenzzeiten berechnen

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Rotationsgeschwindigkeit: 5000 U/min
- Lesekopfbewegung über n Spuren: (1 + 0,002 · n) ms
- Blockgröße: 1024 Byte
- Lücken pro Spur: 10 %

a) Gesucht: Minimale, maximale und durchschnittliche Latenzzeit zum [illegible]n eines Blocks

"1/3 der Spuren" ist eine gute An-näherung

$\neq L_{Seek\_max} / 2$

Latenzzeit = Kommunikation + Se[illegible]k + Rotatio[illegible] + Transfer

$L_{Transfer\_avg}$ = 0,35625 ms (= $L_{Transfer}$)

L_{Seek_avg} = (1 + 0,002 · 2048 / 3) ms ≈ 2,355 ms (avg. Distanz)

$L_{Rotation\_avg}$ = 12 ms / 2 = 6 ms (halbe Rotation)

L_{avg} = 0,35625 ms + 2,355 ms + 6 ms = 8,71125 ms

Aufgabe 3:

Daten speichern

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5
- Blockgröße: 1024 Byte

- Datei mit 100.000 Tupeln der Größe 100 Byte
- Es gibt keine Tupel, die auf mehrere Blöcke aufgeteilt sind

a) Gesucht: Anzahl Tupel pro Block

$$\begin{aligned}\text{Tupel pro Block} &= \lfloor(\text{Blockgröße} / \text{Tupelgröße})\rfloor \\ &= \lfloor(1024\ \text{Byte} / 100\ \text{Byte})\rfloor \\ &= 10\end{aligned}$$

Aufgabe 3:

Daten speichern

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5
- Blockgröße: 1024 Byte

- Datei mit 100.000 Tupeln der Größe 100 Byte
- Es gibt keine Tupel, die auf mehrere Sektoren aufgeteilt sind

b) Gesucht: Anzahl Blöcke für vollständige Datei

Blöcke für Datei = ⌈(Anzahl Tupel / Tupel pro Block)⌉
= ⌈(100.000 / 10)⌉
= 10.000

Aufgabe 3:

Daten speichern

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5
- Blockgröße: 1024 Byte

- Datei mit 100.000 Tupeln der Größe 100 Byte
- Es gibt keine Tupel, die auf mehrere Sektoren aufgeteilt sind

c) Gesucht: Maximale Anzahl Tupel auf der gesamten Festplatte

Aufgabe 3:
Daten speichern

- Sektorgröße: 512 Byte
- Sektoren pro Spur: Ø 64
- Spuren pro Oberfläche: 2048
- Oberflächen pro Platte: 2
- Anzahl Platten: 5
- Blockgröße: 1024 Byte

- Datei mit 100.000 Tupeln der Größe 100 Byte
- Es gibt keine Tupel, die auf mehrere Sektoren aufgeteilt sind

c) Gesucht: Maximale Anzahl Tupel auf der gesamten Festplatte

$$\text{Blöcke pro Spur} = \text{Sektoren pro Spur} / \text{Sektoren pro Block}$$
$$= 64 / (1024 / 512) = 32$$

$$\text{Tupel pro Festplatte} = \text{Tupel pro Block} \cdot \text{Blöcke pro Spur} \cdot \text{Spuren pro Oberfläche} \cdot \text{Oberflächen pro Platte} \cdot \text{Anzahl Platten}$$
$$= 10 \cdot 32 \cdot 2048 \cdot 2 \cdot 5 = 6.553.600$$

Aufgabe 4:

Daten lesen

- Anzahl Zylinder: 8192
- Seekzeit für n Zylinder: (1 + 0,002 · n) ms
- Rotationszeit: Ø 6,5 ms
- Transferzeit: Ø 0,5 ms
- Initiale Lesekopfposition: 4000

- Anfragen:

Eintreffen der Anfrage	0 ms	3 ms	11 ms	19 ms
Angefragter Zylinder	6500	2000	8000	3500

a) Gesucht: Bearbeitung der Anfragen mit **First-Come, First-Served**

b) Gesucht: Bearbeitung der Anfragen mit dem **Elevator Algorithmus**

Aufgabe 4:
Daten lesen

Eintreffen der Anfrage	0 ms	3 ms	11 ms	19 ms
Angefragter Zylinder	6500	2000	8000	3500

- Anzahl Zylinder: 8192
- Seekzeit für n Zylinder: (1 + 0,002 · n) ms
- Rotationszeit: Ø 6,5 ms
- Transferzeit: Ø 0,5 ms
- Initiale Lesekopfposition: 4000

Anfangsbeispiel

Start Time	Request Queue	Request in Progress	End Time	Direction
0ms	6500	6500		

Aufgabe 4:
Daten lesen

Eintreffen der Anfrage	0 ms	3 ms	11 ms	19 ms
Angefragter Zylinder	6500	2000	8000	3500

- Anzahl Zylinder: 8192
- Seekzeit für n Zylinder: (1 + 0,002 · n) ms
- Rotationszeit: Ø 6,5 ms
- Transferzeit: Ø 0,5 ms
- Initiale Lesekopfposition: 4000

Anfangsbeispiel

Start Time	Request Queue	Request in Progress	End Time	Direction
0ms	6500	6500	(1+0,002·2500)+6,5+0,5 =13ms	→
13ms				

Aufgabe 4:

Daten lesen

Eintreffen der Anfrage	0 ms	3 ms	11 ms	19 ms
Angefragter Zylinder	6500	2000	8000	3500

- Anzahl Zylinder: 8192
- Seekzeit für n Zylinder: (1 + 0,002 · n) ms
- Rotationszeit: Ø 6,5 ms
- Transferzeit: Ø 0,5 ms
- Initiale Lesekopfposition: 4000

First-Come, First-Served

Start Time	Request Queue	Request in Progress	End Time	Direction
0ms	6500	6500	(1+0,002·2500)+6,5+0,5 =13ms	→
13ms	2000 8000	2000	(1+0,002·4500)+6,5+0,5 =17ms	←
30ms	8000 3500	8000	(1+0,002·6000)+6,5+0,5 =20ms	→
50ms	3500	3500	(1+0,002·4500)+6,5+0,5 =17ms	←
67ms				

Aufgabe 4:
Daten lesen

Eintreffen der Anfrage	0 ms	3 ms	11 ms	19 ms
Angefragter Zylinder	6500	2000	8000	3500

- Anzahl Zylinder: 8192
- Seekzeit für n Zylinder: (1 + 0,002 · n) ms
- Rotationszeit: Ø 6,5 ms
- Transferzeit: Ø 0,5 ms
- Initiale Lesekopfposition: 4000

Elevator Algorithmus

Start Time	Request Queue	Request in Progress	End Time	Direction
0ms	6500	6500	(1+0,002·2500)+6,5+0,5 =13ms	→
13ms	8000 2000	8000	(1+0,002·1500)+6,5+0,5 =11ms	→
24ms	3500 2000	3500	(1+0,002·4500)+6,5+0,5 =17ms	←
41ms	2000	2000	(1+0,002·1500)+6,5+0,5 =11ms	←
52ms				

Aufgabe 5:

Daten auf verschiedenen Spuren

- Rotations- und Lesekopfgeschwindigkeit sind konstant, d.h.
  - Rotationslatenz und Seektime sind unverändert
  - sequentielle Datentransferrate ist auf äußeren Spuren größer, falls dort mehr Sektoren pro Spur angelegt wurden

- Wo sollte man daher die folgenden Dateien für die genannten Zugriffe positionieren (innen, mitte, außen)?
  - seltene, sequentielle Scans …
    - einer großen Datei
    - einer kleinen Datei
  - häufiger, random Zugriff auf …
    - eine kleine Datei
    - eine große Datei per Index

z.B Tabelle

z.B Systemkatalog

Daten auf verschiedenen Spuren

- Wo sollte man daher die folgenden Dateien für die genannten Zugriffe positionieren (innen, mitte, außen)?
  - seltene, sequentielle Scans …
    - ◇ einer großen Datei: **außen**
      - Kosten dominiert durch sequentiellen Datentransfer
      - Sequentieller Datentransfer ist außen am schnellsten
    - ◇ einer kleinen Datei: **innen**
      - Kosten dominiert durch initialen Seek und Rotation (Lesen einer kleinen Datei ist effektiv Random I/O)
      - Innen wird nichts optimiert, aber das Lagern kleiner, selten zugegriffener Dateien tut hier am wenigsten weh

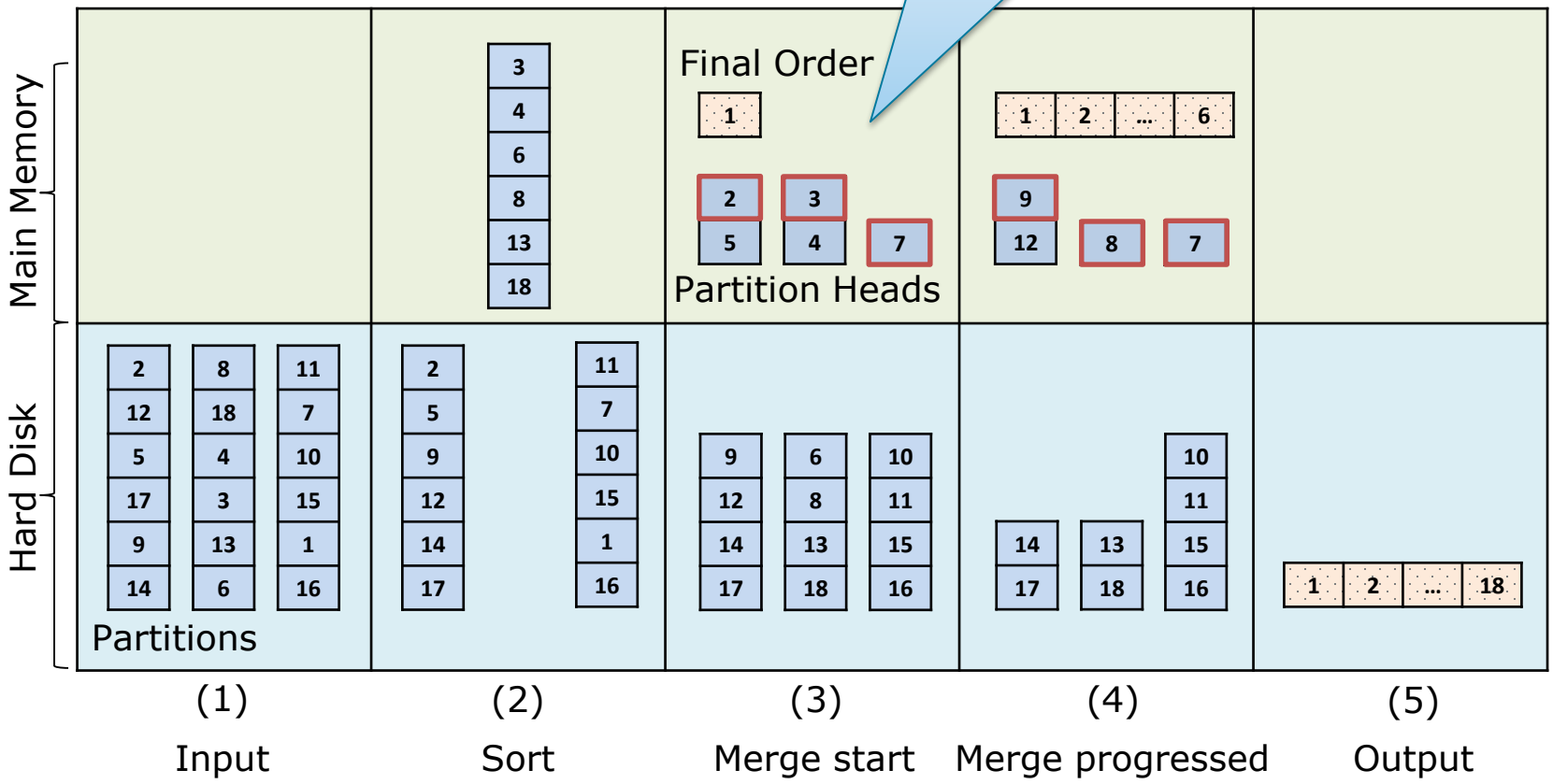
Daten auf verschiedenen Spuren

- Wo sollte man daher die folgenden Dateien für die genannten Zugriffe positionieren (innen, mitte, außen)?
  - häufiger, random Zugriff auf …
    - eine kleine Datei: **mitte**
      - Kosten dominiert durch Seek und Rotation
      - Seek wird in der Mitte minimiert
        (Wegen häufigem Zugriff ist die Optimierung der Seektime hier am wichtigsten)
    - eine große Datei per Index: **innen**
      - Kosten dominiert durch häufigen Seek zwischen Datei und Index
      - Seek wird durch nahes Zusammenlegen minimiert
      - Beim Platzieren innen sparen wir den jeweils wertvollen äußeren und mittleren Platz

Daten sortieren: TPMMS

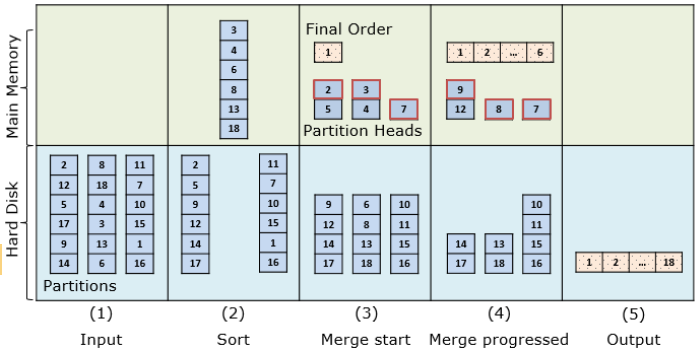
Prefetching! Vorteil?

Algorithmus:



Aufgabe 6:

Daten sortieren: TPMMS



Gedankenspiel: Three-Phase-Multiway-Merge-Sort

1. Warum könnte er notwendig sein?

   Es passen nicht alle Köpfe vorsortierter Teillisten in den RAM.

2. Wie könnte ein Three-Phase-Multiway-Merge-Sort funktionieren?

   P1: Sort; P2: (Pre-)Merge jeweils n Teillisten; P3: Merge alle Listen

   Um wie viel steigen die Lese- und Schreibkosten?

   1 x alle Tupel lesen + 1 x alle Tupel schreiben

3. Die Partitionen müssen nicht unbedingt Hauptspeicher-groß sein.

   Welche Vorteile könnte eine Partitionsgröße $\frac{Hauptspeichergröße}{2}$ haben?

   Sortieren im RAM müsste nicht in-place sein und ist so O(n*log(n))