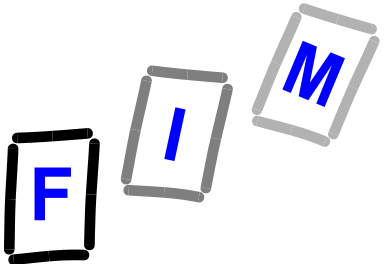
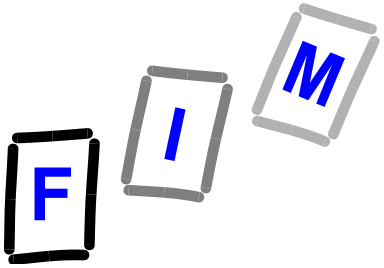


Betriebssysteme

Kap J, Teil C:
Paging, Pagereplacement



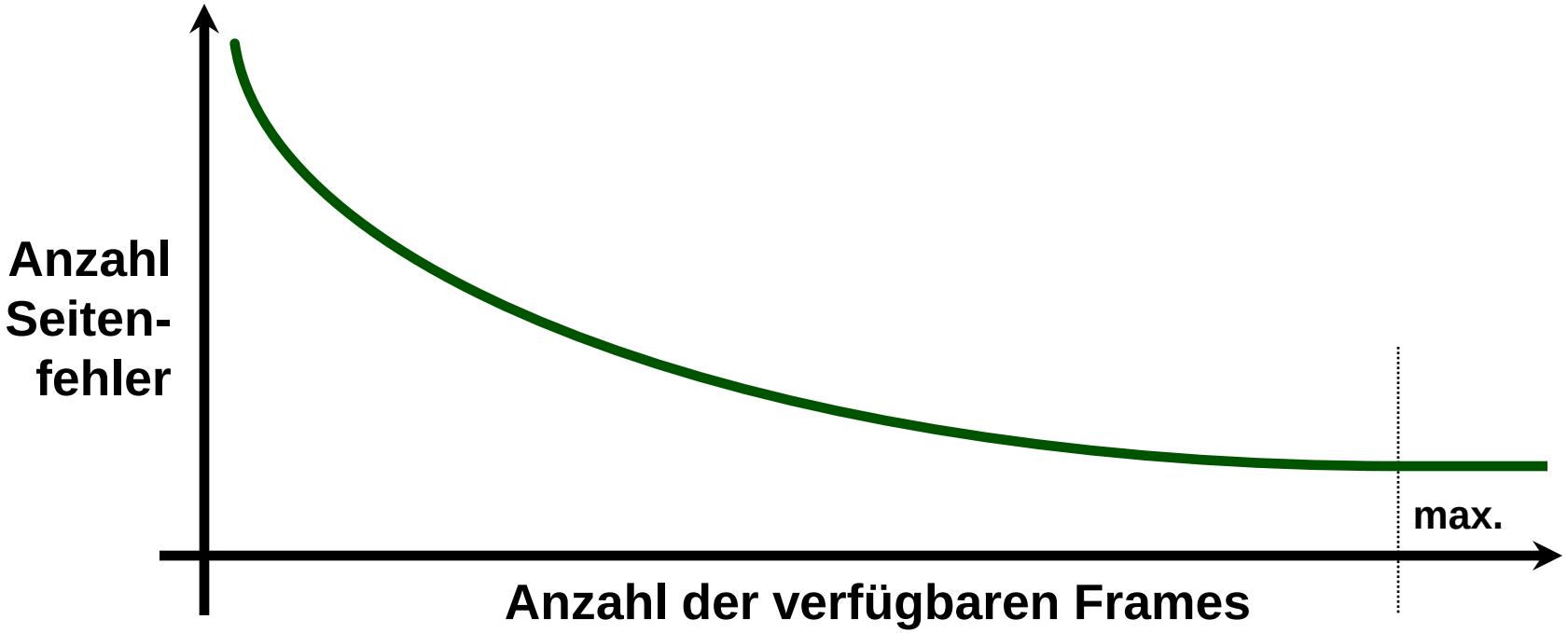
Welche Seite soll ausgelagert werden?

- **Ein- / Auslagern benötigt Zeit**
  - ➡ **Kontextwechsel erforderlich**
    - » **Wechsel zu einem BS-Prozess, welcher für das Management der Seitenfehler zuständig ist**
    - » **Wechsel zurück zum unterbrochenen Prozess**
  - ➡ **Daten müssen vom Hauptspeicher in den Sekundärspeicher übertragen werden und umgekehrt**
- **Ziel des Seitenaustausch-Algorithmus:**

  Anzahl der Seitenfehler möglichst klein halten

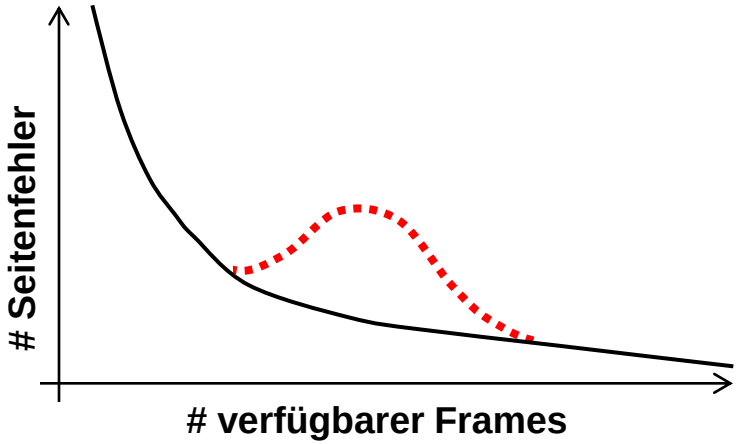
Belady's Anomalie (1)

Mehr realer Hauptspeicher
→ weniger Seitenfehler
(Idealfall / Standardfall)



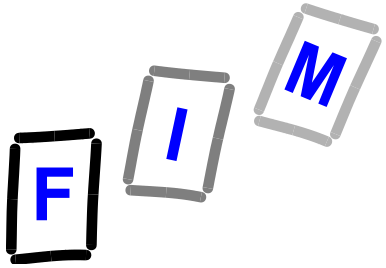
(vergleiche Bild 9.7 in Silberschatz/Galvin)

Belady's Anomalie (2)



Standardfall: mehr verfügbare Frames → weniger Seitenfehler

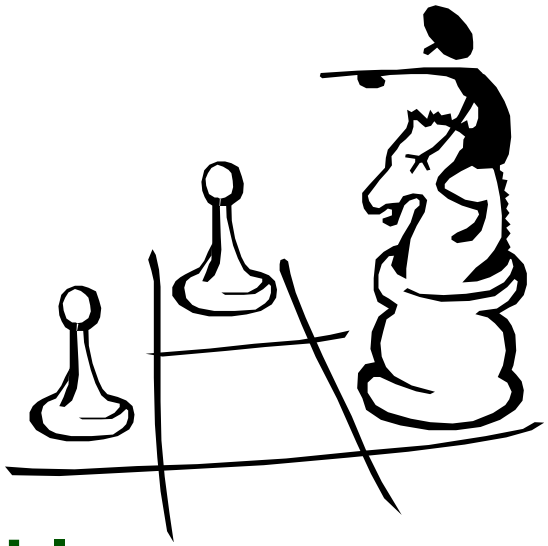
Anomalie, wie von Belady beschrieben



Einfache Erklärung der Anomalie

- **Wenn man Waren in einem Geschäft verkauft, braucht er/sie Geld zum Herausgeben**
- **Wenn man die Anzahl der Münzen erhöht, dann ist es auch wahrscheinlicher, dass man immer herausgeben kann.**
- **Aber: Wenn man die gesamte Summe erhöht und gleichzeitig einige, aber die „falschen“ Münzen wegnimmt, z.B. 1 € Münzen, würde es nichts helfen und die Situation sogar schlimmer machen**

Optimaler Seitenaustausch



Vorgegeben:

- Anzahl der Seitenrahmen
- Page-Reference-String

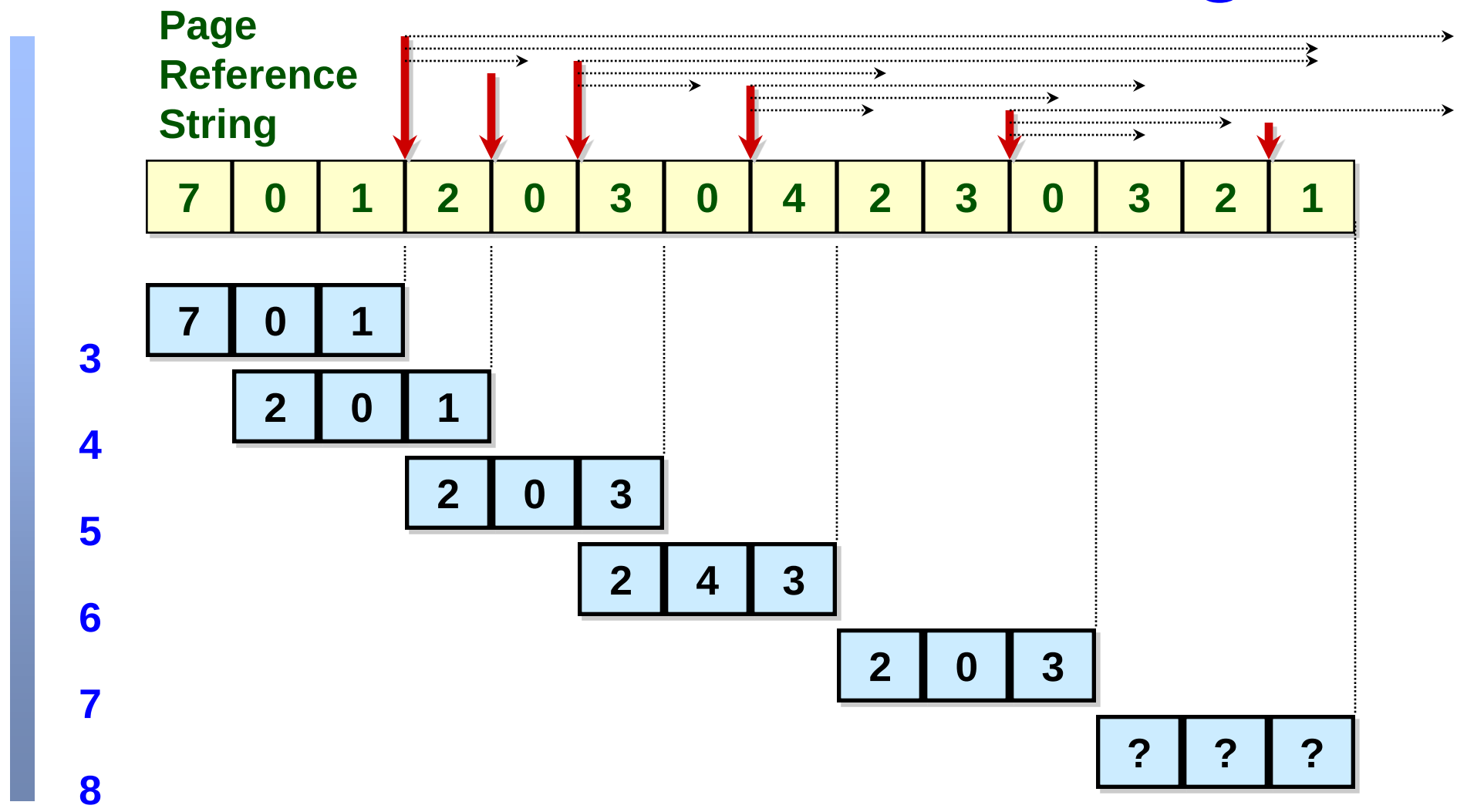
Gewünschtes Ergebnis:

- Minimale Anzahl von Seitenfehlern

Strategie:

Wähle zum Austausch diejenige Seite, die in Zukunft am längsten nicht referenziert wird

Beispiel: Optimale Seitenersetzung

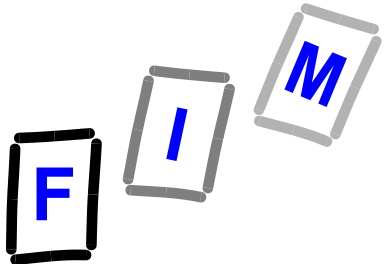


Optimale Seitenersetzung

Orakel

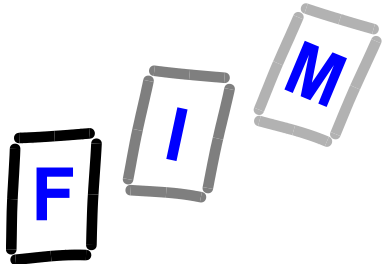
Seitenauswechslungs-Algorithmus

- Blick in die Zukunft notwendig
- Page-Reference-String ist von vielen Umständen abhängig, z.B.
  - ➡ Benutzerent-scheidungen
  - ➡ Fehlerbedingungen
  - ➡ OS-Scheduling
  - ➡ Parallel laufende Prozesse
  - ➡ Anzahl verfügbarer Seitenrahmen
  - ➡ . . . . . .



Optimaler Seiten-austausch: Nachteil

- Wir haben schon gesehen, dass der Algorithmus den „Blick in die Zukunft“ braucht, um arbeiten zu können
- Dieser ist aber dem BS nicht verfügbar
- Man benötigt realisierbare Algorithmen
- Anschliessend überlegen wir die notwendige Hardware-Unterstützung



LRU = Least Recently Used

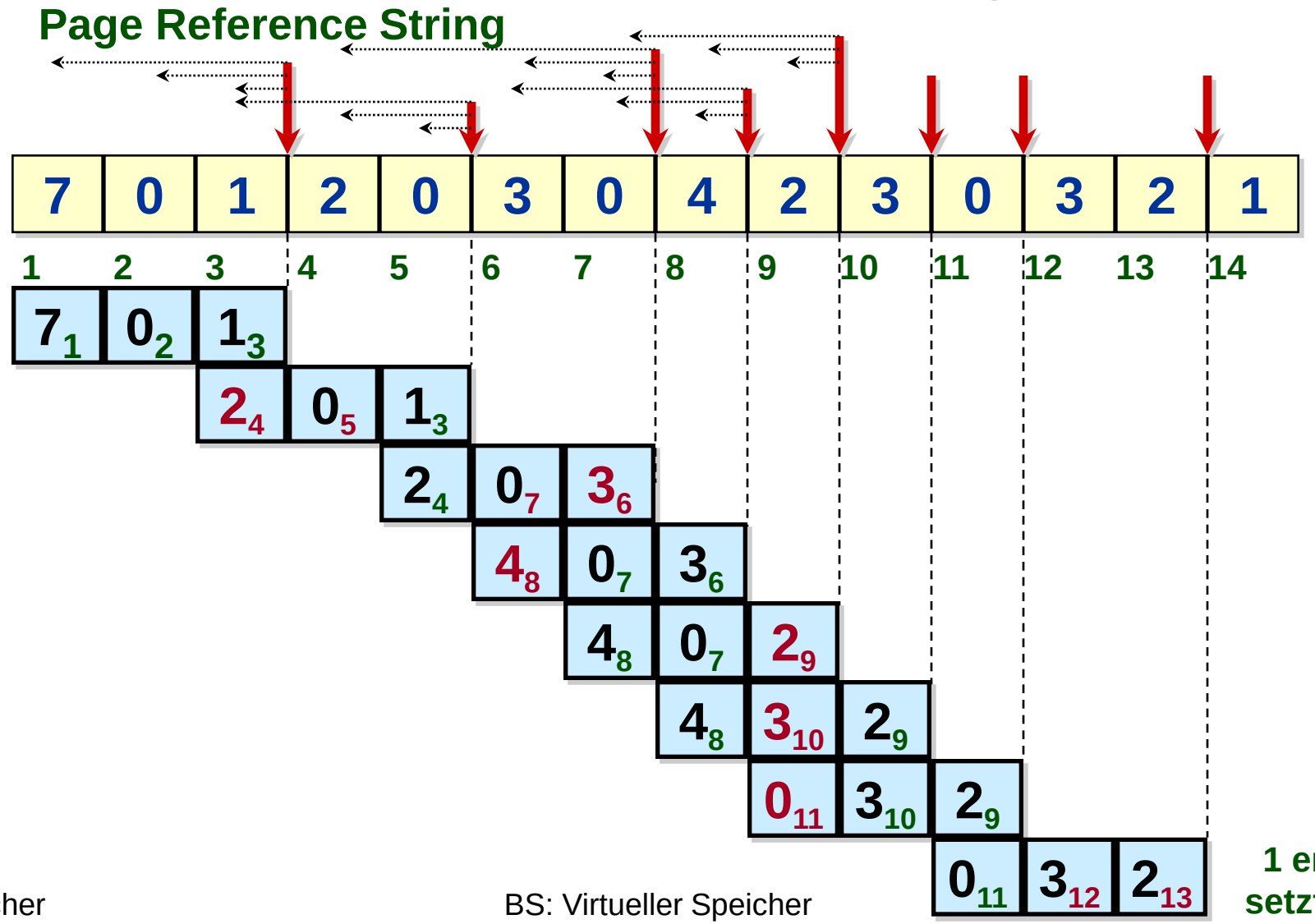
- **Basisannahme:**
  Vergangenheit dient als Annahme für die Zukunft

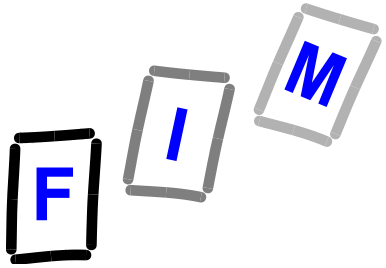
- **Auszulagernde Seite:**

Jene Seite, die am längsten nicht referenziert wurde

(d.h. deren letzte Verwendung am längsten zurückliegt)

Beispiel: Least Recently Used





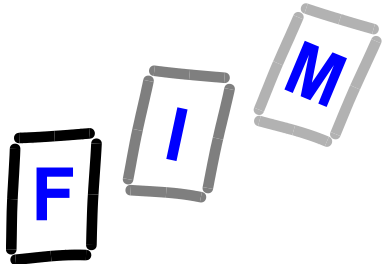
LRU: Notwendiger Hardware-Support

Relativ effizient, häufig eingesetzt

- ➡ **Leidet nicht unter Belady's Anomalie, aber hoher Hardwareaufwand zusätzlich, Z.B. Seitenverzeichnis-Eintrag um einen Zeitstempel erweitern.**
  - » **Gib logische Uhr dazu, wird bei jedem Speicherzugriff hinaufgezählt**
  - » **Bei einer Seitenreferenz: Kopiere Uhrzähler in das Zeitstempelfeld der Seite/Frame**

Diskussion:

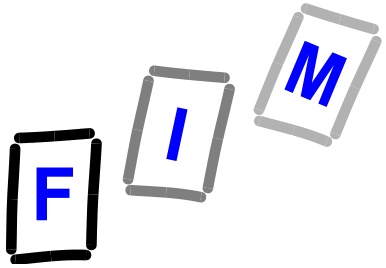
Da die Vergangenheit ohnehin nur eine Annährung der Zukunft darstellt, wäre nicht eine Annäherung von LRU mit weniger Hardwareaufwand sinnvoll?



LRU Approximation: z.B. „Zweite Chance“

- Auch **Not Used Recently (NUR)** oder ***Uhr-Algorithmus*** genannt

- Wenig Hardware-Ansprüche:
  - ➡ Ein Referenzbit pro Seitenrahmen
  - ➡ Wird auf 1 gesetzt, wenn ein Zugriff auf den Seitenrahmen erfolgt
  - ➡ Die Seite wird ausgewechselt:
    - » Wenn das Bit auf 1 gesetzt ist, wird es gelöscht (0) und die Seite erhält eine „2. Chance“ auf Verbleib. Ein anderes „Opfer“ wird ausgewählt
    - » Wenn das Bit gelöscht ist (0), dann ist die Seite das „Opfer“

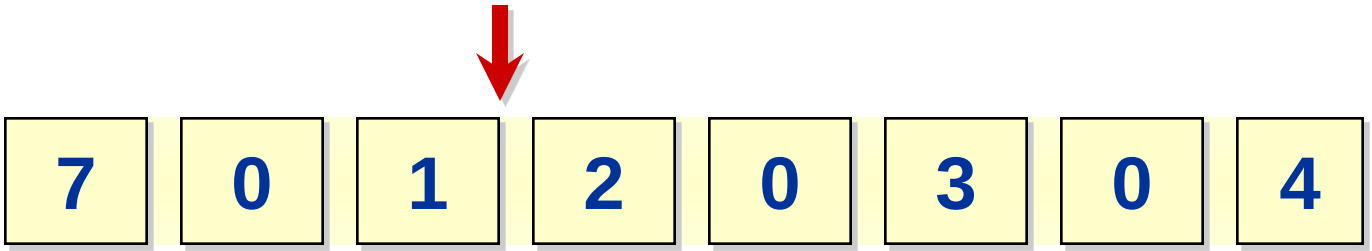


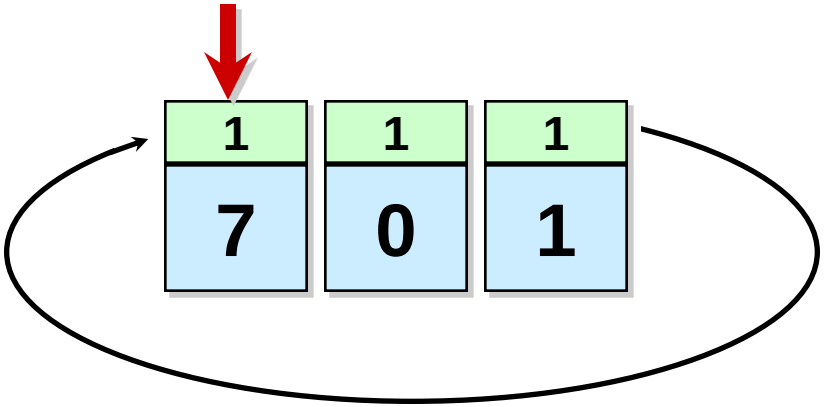
Implementierung des NUR-Algorithmus

- Verwende eine zirkuläre Liste
- Die Seite, die als nächste ersetzt werden soll, wird durch einen Zeiger markiert
  - ➡ Grundsätzlich ein FIFO Algorithmus, aber
- Wenn ein Seitenrahmen zum Ersetzen gebraucht wird, rückt der Zeiger zyklisch vor, bis er einen Seitenrahmen findet, dessen Referenzbit gelöscht wurde.
  - ➡ Diese Seiten wird ersetzt
- Der Zeiger löscht alle Referenzbits, die er beim zyklischen Vorrücken antrifft

NUR (not used recently)

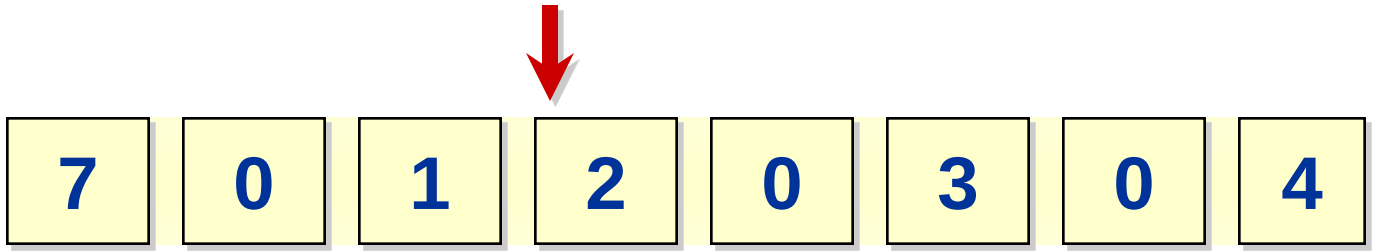
Page Reference String

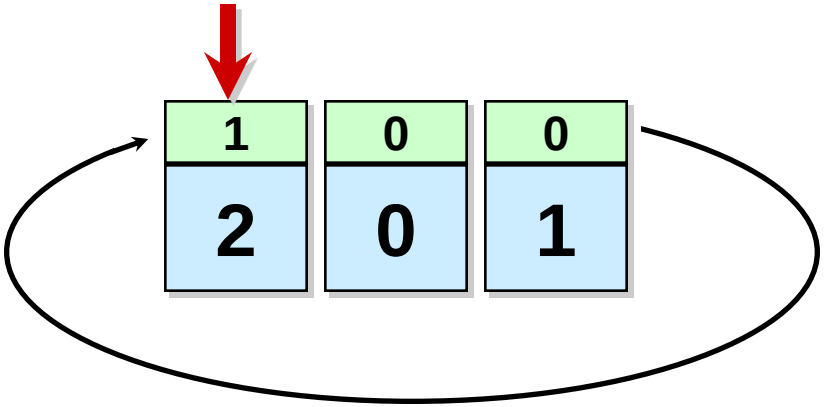




NUR (not used recently)

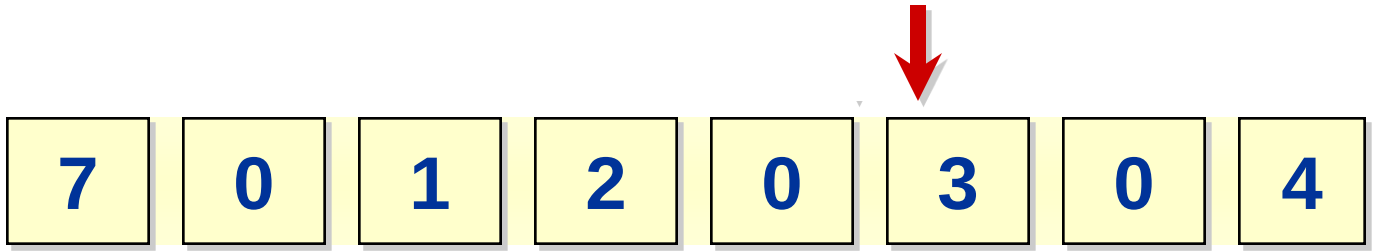
Page Reference String

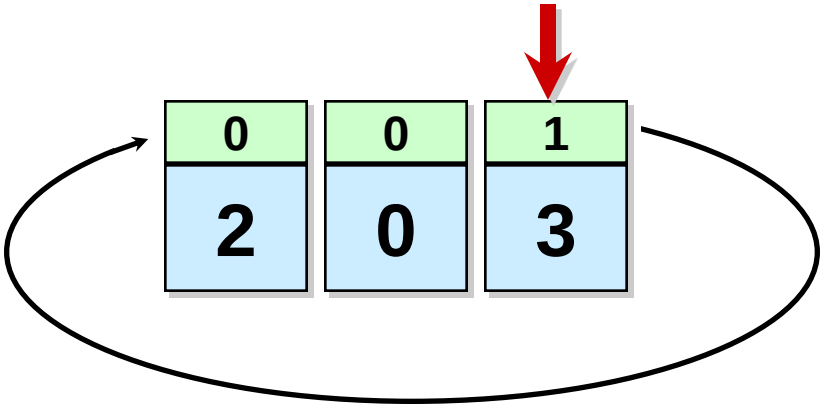




NUR (not used recently)

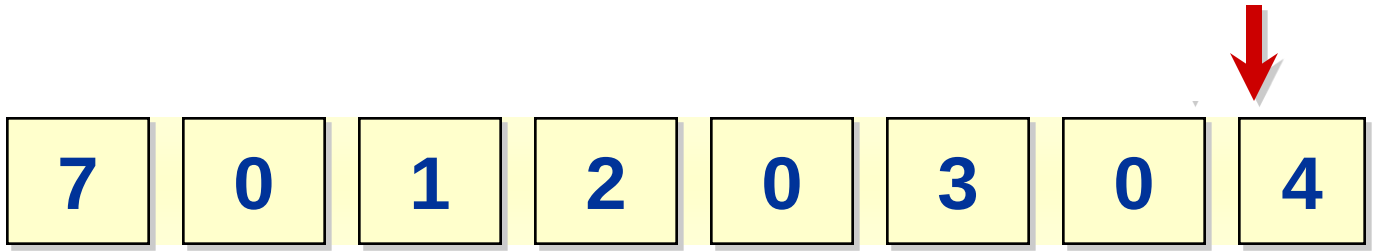
Page Reference String

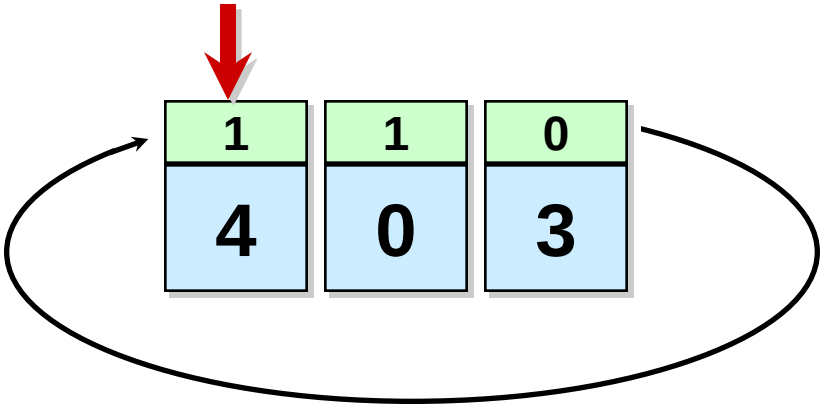


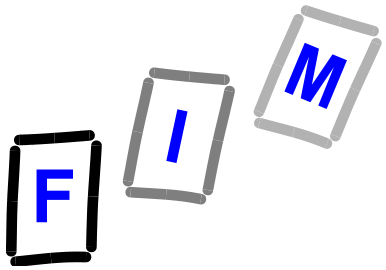


NUR (not used recently)

Page Reference String

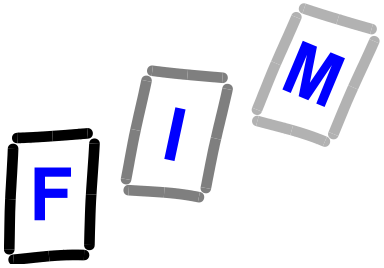






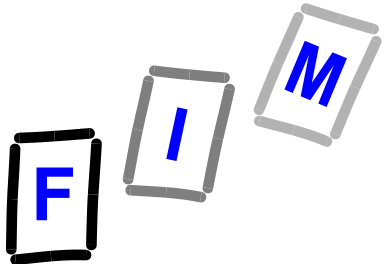
„Zweite Chance“ + Dirty Bit

acc	dirty	Bewertung
0	0	In letzter Zeit nicht verwendet und nicht modifiziert → Optimale Seite zum Ersetzen.
0	1	In letzter Zeit nicht verwendet aber modifiziert → Nicht so gut, da Rückschreiben des Seitenrahmens auf die Disk erforderlich.
1	0	In letzter Zeit verwendet aber “sauber“ wird wahrscheinlich bald wieder benötigt.
1	1	In letzter Zeit verwendet und zusätzlich modifiziert - wird wahrscheinlich bald wieder benötigt. Zusätzlich Rückschreiben der Seite auf die Disk (Page File) erforderlich.



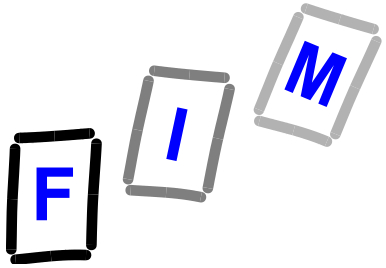
Zuweisungsalgorithmen
(engl.: allocation algorithms)

- **Gleiche Zuweisung**
  - ➡ **Jeder Prozess erhält die gleiche Anzahl an Seitenrahmen**
- **Proportionale Zuweisung**
  - ➡ **Weist den verfügbaren Speicher (Seitenrahmen) je nach Größe des Prozesses zu**
  - ➡ **„Priorität“ wird auch beachtetet**
- **Anmerkung: Wenn der Multiprogramming-Grad erhöht oder gesenkt wird, ist die Anzahl der zugewiesenen Seitenrahmen entsprechend anzupassen**



Zuweisungsalgorithmen Global ↔ Lokal

- **Wenn ein Prozess einen Seitenrahmen braucht, kann er genommen werden von:**
  - ➡ **Der Menge von Seitenrahmen, die dem Prozess zugewiesen sind**
    Lokaler Seitenaustausch
  - ➡ **Von einem anderen Prozess**
    Globaler Seitenaustausch
    - » **Z.B. von Prozessen mit niedriger Priorität**



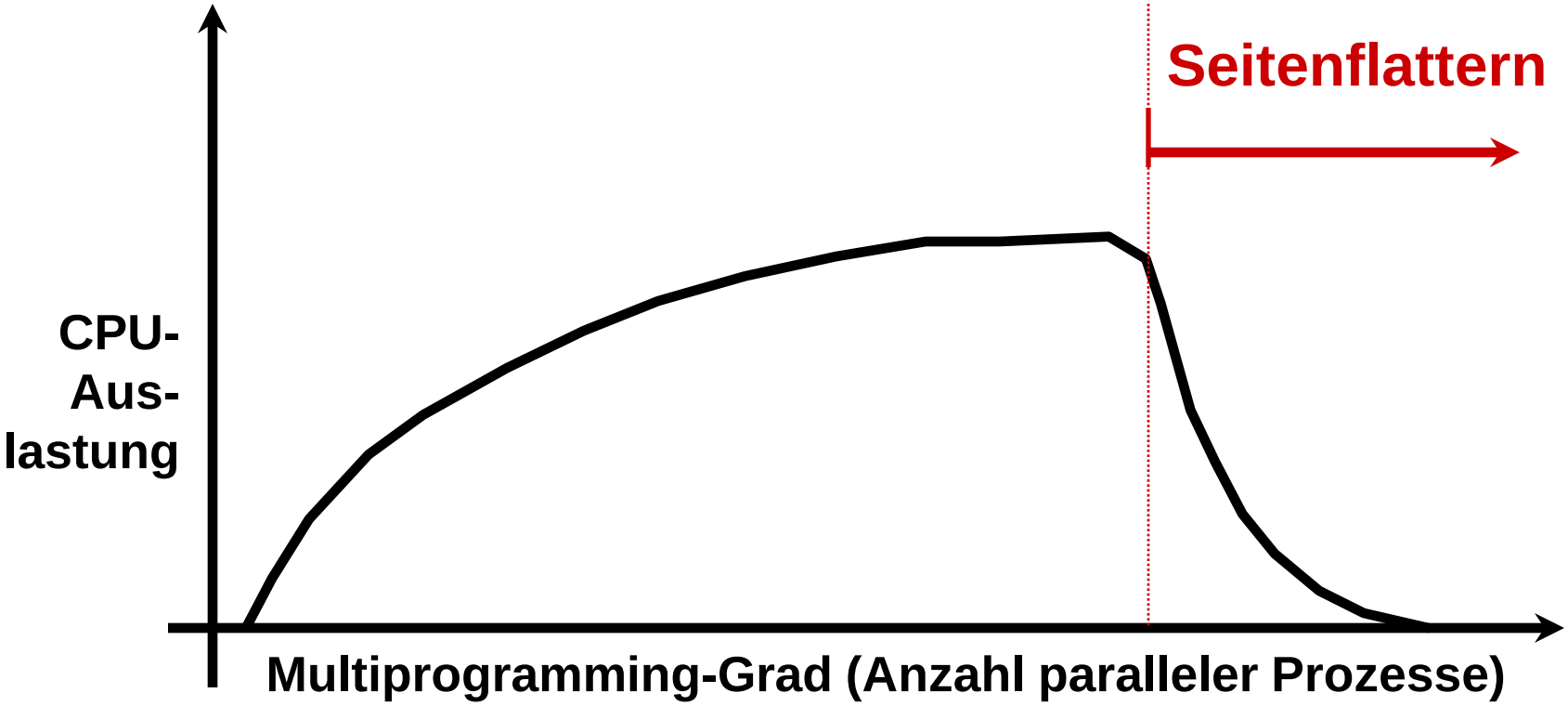
Seitenflattern
(engl.: thrashing)

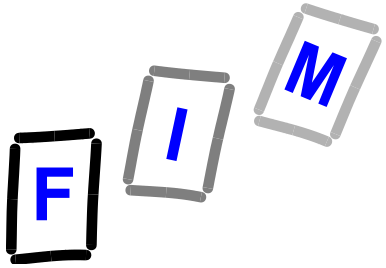
- **Wenn (zu) viele Prozesse parallel laufen**
  - ➡ **Jedem Prozess können weniger Seitenrahmen zugeordnet werden**
  - ➡ **Ein Prozess, der weniger Seitenrahmen zugewiesen hat, löst mehr Seitenfehler aus**
  - ➡ **Das System ist damit beschäftigt, auf diese Seitenfehler zu reagieren**
- **Phänomen „Seitenflattern“ (thrashing)**

  → Weniger CPU-Auslastung

  → Weniger Durchsatz

Seitenflattern

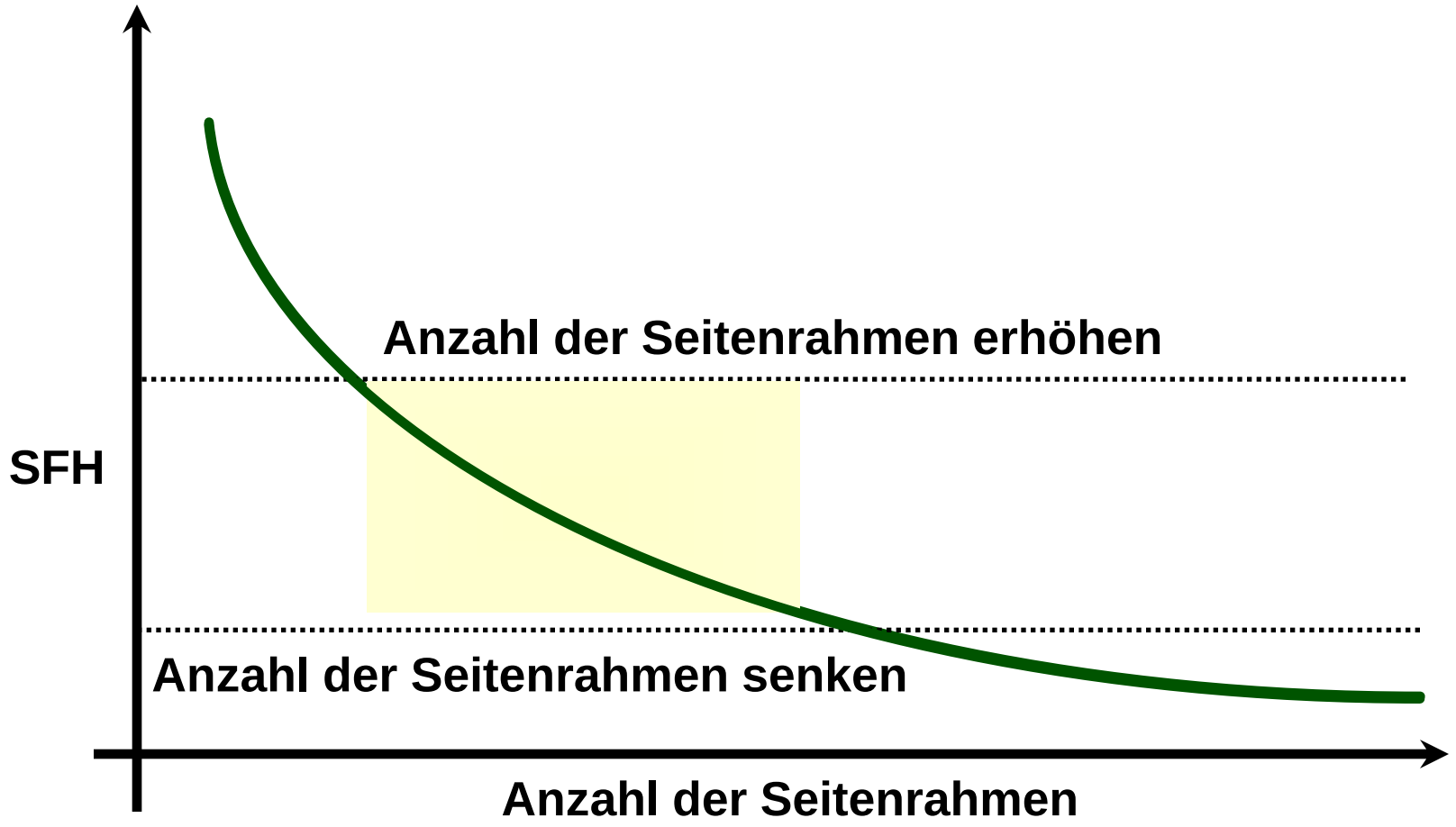




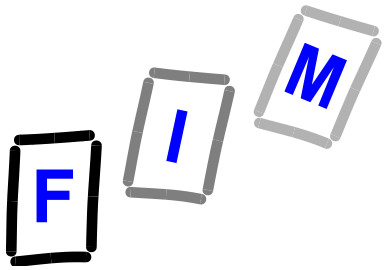
Reaktion des BS auf Seitenflattern

- **Dem laufenden Prozess mehr Seitenrahmen zuordnen, indem man anderen Prozesse Seitenrahmen entzieht**
  - ➡ **Eine globale Seitenrahmen-Zuweisungs-Strategie ist dafür notwendig**
- **Multitasking-Grad senken, indem man einen oder mehrere Prozesse zeitweise anhält**
- **Anmerkung: Wenn eine lokale Austauschstrategie verwendet wird, dann beeinflusst das Seitenflattern eines Prozesses die anderen Prozesse nicht direkt, aber indirekt durch die I/O Interrupts**

SFH
Seitenfehler-Häufigkeit

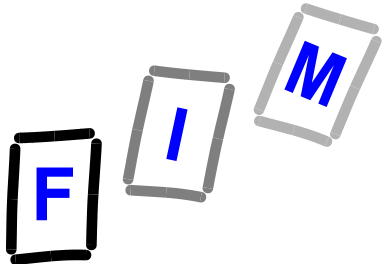


(vergleiche Bild 9.17 in Silberschatz/Galvin)



Arbeitsmenge Working Set

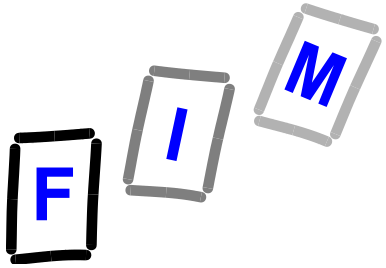
- ***WS ist die Anzahl der Seiten, die ein Prozess im physischen Speicher haben muss, damit er effizient ausgeführt werden kann***
- **Ein Prozess mit weniger als seinem Working Set ist anfällig für Seitenflattern**
- **W2K Bezeichnung ist leicht verschieden**
  - ➡ **Working Set zum Zeitpunkt t ist die Menge der Seiten, die einem Prozess bei t zugeordnet ist**
    - » **Unterstellt dabei, dass das BS die Zuweisung ohnedies optimal durchführt**



Lokalität

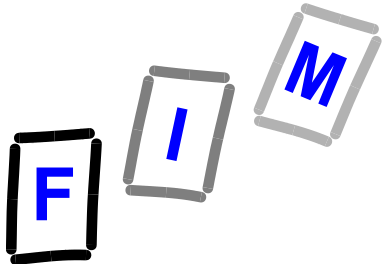
Programme tendieren dazu, in kleinen Regionen/Bereichen ihres Adressraums innerhalb eines Beobachtungsintervalles zu laufen

- **Schleifen wie**
  - **do { … } while (<condition>);**
  - **while (<condition>) { … } ;**
  - **for (i = 1; i <= largenumber; i++) { … } ;**
- **DLL,**
  welche gerade dynamisch in den Speicher geladen wurde



Lokalität

- **Diese Region und die dazugehörigen Seiten ändern sich sehr langsam in Bezug auf Anzahl und Größe**
  - ➡ **Normalerweise bestimmen die letzten n referenzierten Seiten innerhalb eines Zeitfensters Δ das Working Set**
    - » **Anmerkung: n ist nicht die Größe des Working Set**
- **Prozesse tendieren dazu, eine „Lokalität der Referenzen“ zu haben**

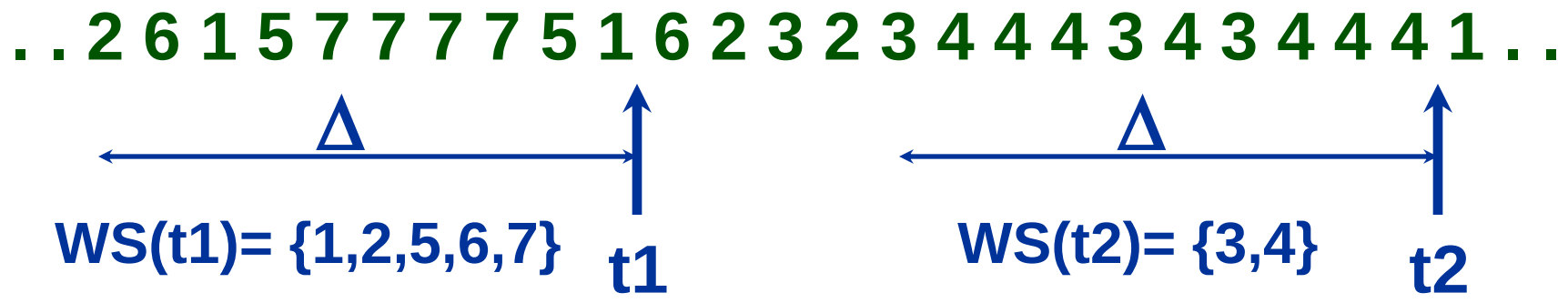


Working Set

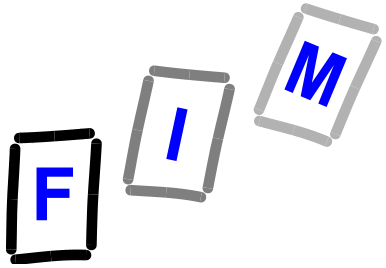
- **Offensichtlich:**
  Mindestens diese Menge von Seiten, die im Zeitfenster Δ referenziert wurde, sollte zum Zeitpunkt t für einen Prozess im Speicher gehalten werden
- **Working Set zur Zeit t**
- **Working Set ist eine Annäherung der Lokalität des Programmes**

Lokalitätsmodell

- Basiert auf der Annahme der Lokalität
- Working Set Window Δ
- Working Set: Die bei den letzten Δ Speicherzugriffen referenzierten Seiten
- Δ = 10 hier im Beispiel



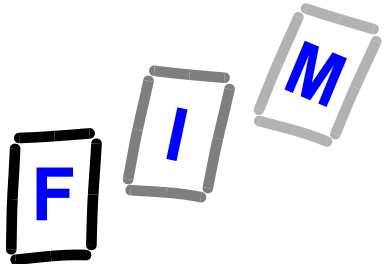
(vergleiche Bild 9.16 in Silberschatz/Galvin)



Größe des Working Set Fensters?

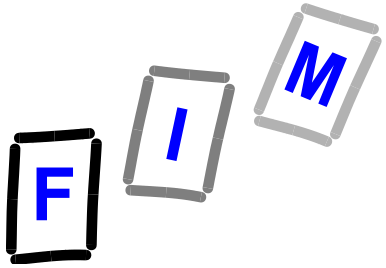
- **Wahl des Δ**

  - ➡ **Δ zu klein**
    → Daraus resultierende Anzahl der Seitenrahmen für Working Set zu klein: Seitenflattern
  - ➡ **Δ zu groß**
    → Daraus resultierende Anzahl der Seitenrahmen für Working Set zu groß: zu viele Seitenrahmen sind dem Prozess zugeordnet (ineffizient)



LRU
Lokalität

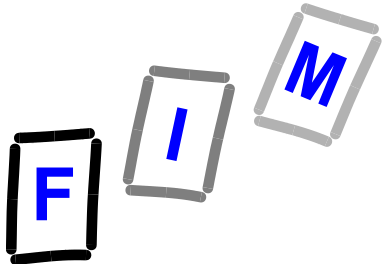
- **Man betrachte die grundlegende Idee, die hinter LRU steht:**
  - ➡ **Benutzt die Vergangenheit des Page-Reference-Strings, um die Zukunft „vorherzusehen“**
  - ➡ **Basiert auf dem Prinzip der Lokalität**
  - ➡ **Die zuletzt benutzten Seiten kommen dem Working Set nahe**
    - » **Beachte: Ausgetauscht wird die Seite, die am längsten nicht referiert wurde**



Lokalität unter “Programmkontrolle”

```
CONST N= grosse Zahl;
  VAR   A: ARRAY N OF
          ARRAY N OF INTEGER;

FOR v0:=0 TO N-1 DO
  FOR v1:=0 TO N-1 DO
    A[v0,v1]:=0;
  END; (*for*)
  END; (*for*)
```



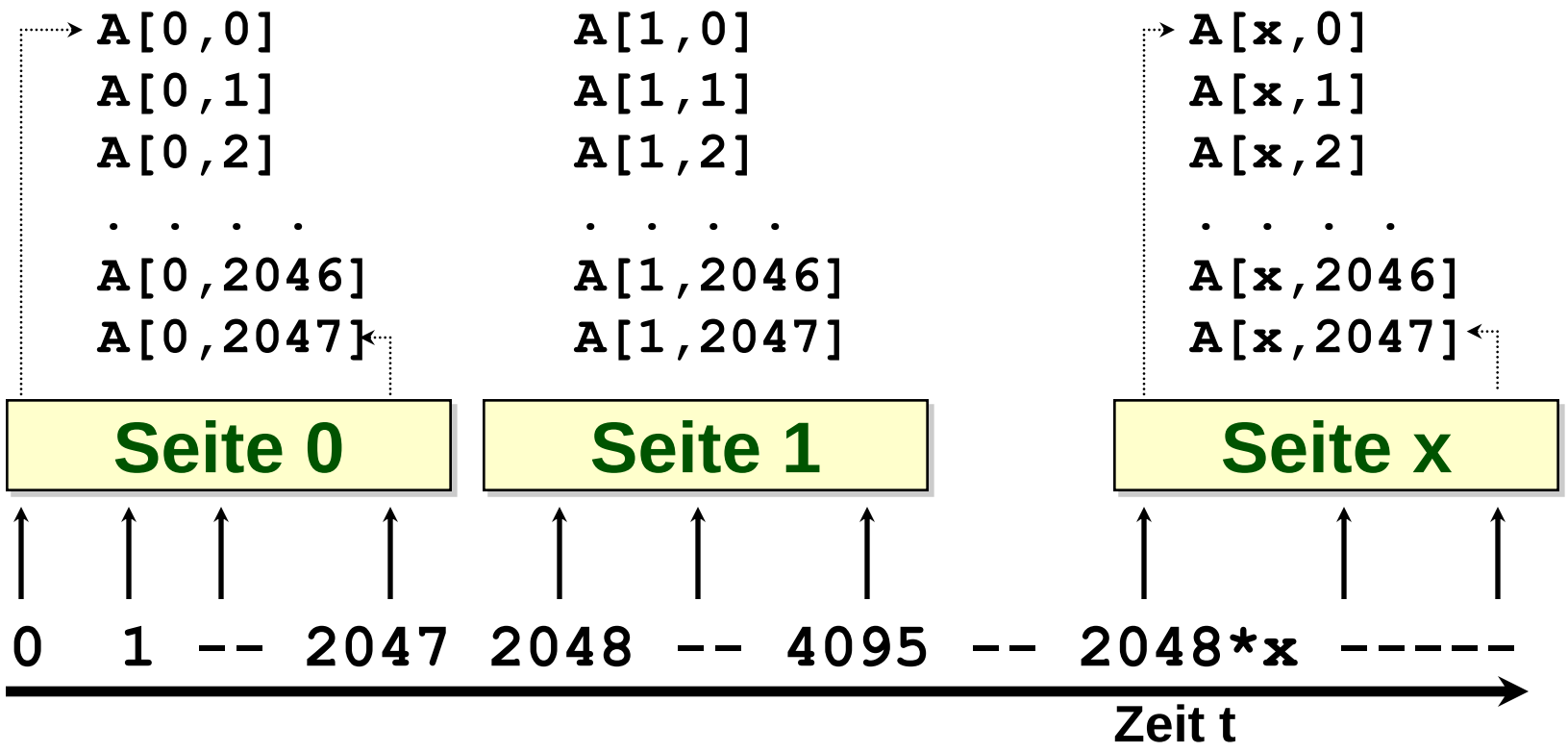
Überlegungen zum Programmbeispiel

Einige Annahmen:

- **INTEGER sind 2 Byte**
- **N=2048**
- **damit Matrixgröße 8 MB**
- **Seitengröße 4 KB (4096 Bytes)**
- **Verwendetes Aufrufverfahren:**
  Zweite Chance (NUR)

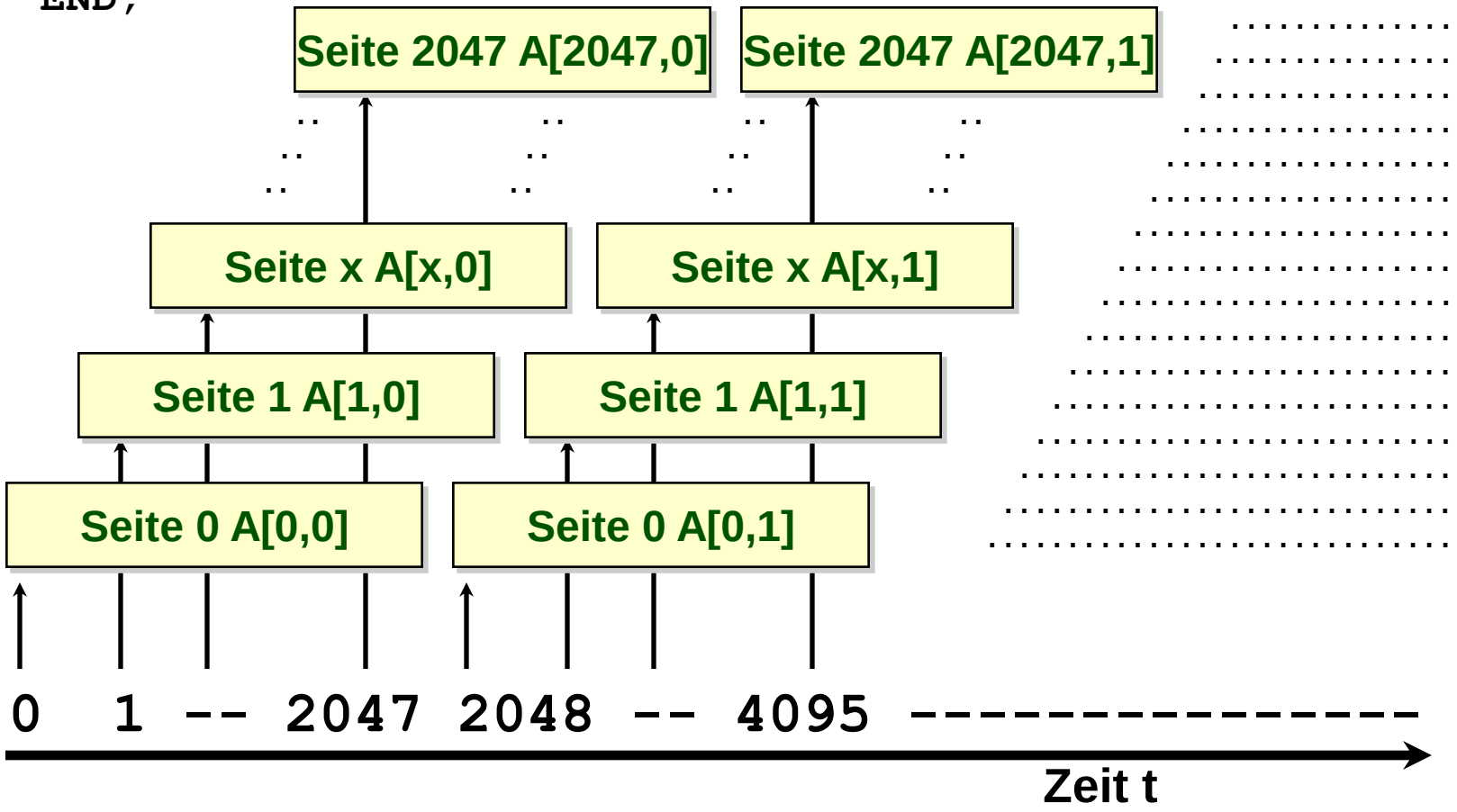
Richtiger Zugriff auf die Matrix (Ver01)

```
FOR v0:=0 TO N-1 DO
  FOR v1:=0 TO N-1 DO A[v0,v1]:=0; END;
END;
```



Falscher Zugriff auf die Matrix (Ver10)

```
FOR v0:=0 TO N-1 DO
  FOR v1:=0 TO N-1 DO A[v1,v0]:=0; END;
END;
```



Überlegungen zum Programmbeispiel

- **Ausgangsbasis: Bei Start und Ende steht keine Seite der Matrix A im realen Hauptspeicher (d.h. alle Seiten sind zu Beginn ein- und bei Abschluss auszulagern).**

Platz für Array KB	Frames	Ver01 Load	Ver01 Save	Ver10 Load	Ver10 Save
4	1	2.048	2.048	4.194.304	4.194.304
*8.188	*2.047	2.048	2.048	4.194.304	4.194.304
8.192	2.048	2.048	2.048	2.048	2.048

Zeitvergleich:

2.048 Pagefaults à 10 ms = ~ 20 Sek. (~½ Min.)

4.194.304 Pagefaults à 10 ms = ~ 42000 Sek. (~1/2 Tag)

(* ohne Überlegungen Diskcaching)