Algorithmen und Datenstrukturen 1

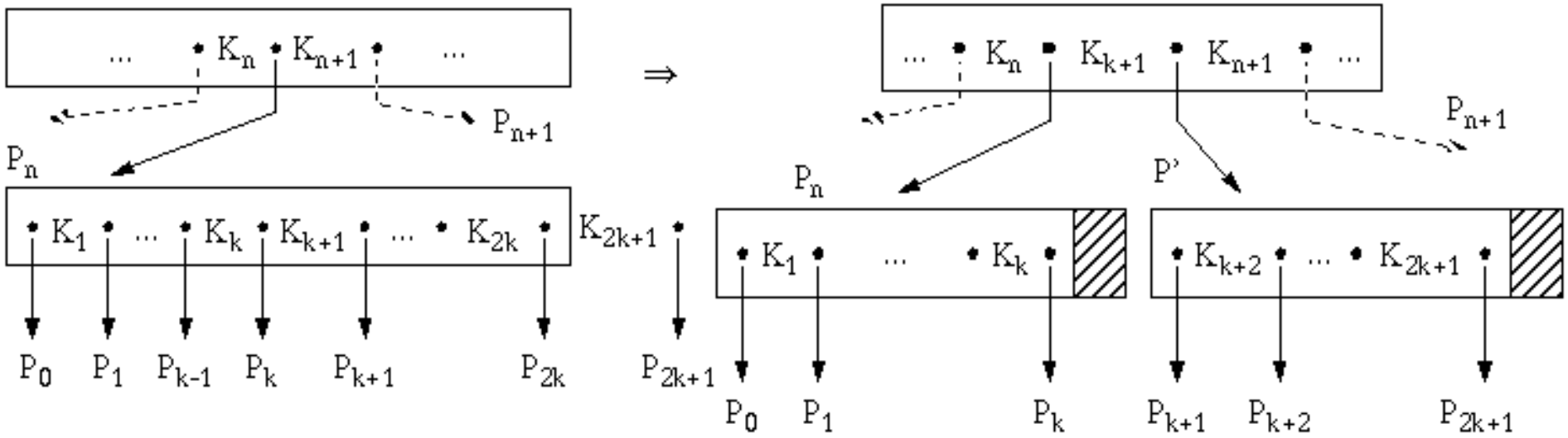
11. Vorlesung

Peter F. Stadler

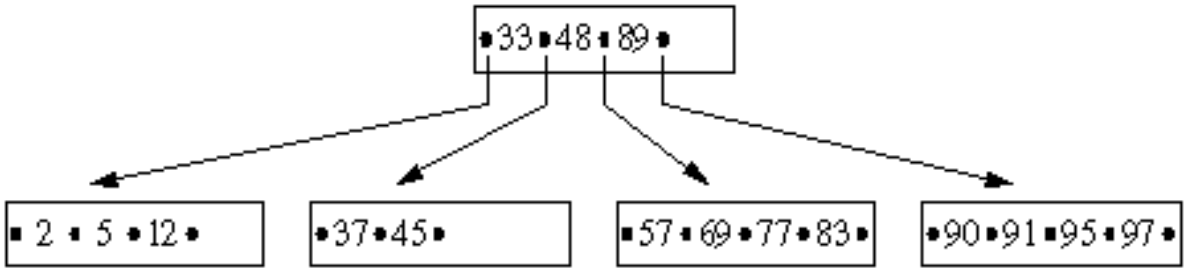
Universität Leipzig
Institut für Informatik
studla@bioinf.uni-leipzig.de

Wdhlg.: Einfügen in B-Bäumen

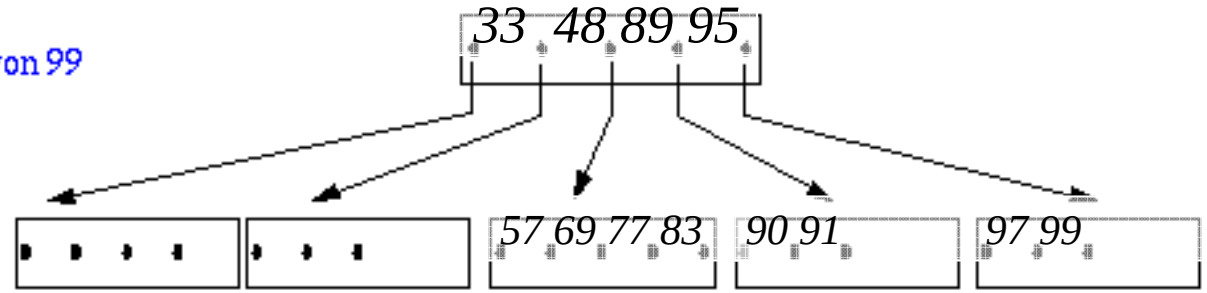
- **Bei B-Bäumen ist Wachstum von den Blättern zur Wurzel hin gerichtet**
- **Einfügealgorithmus (ggf. rekursiv)**
  - Suche Einfügeposition: Wenn Platz vorhanden ist, speichere Element, sonst schaffe Platz durch Split-Vorgang und füge ein
- **Split-Vorgang als allgemeines Wartungsprinzip**



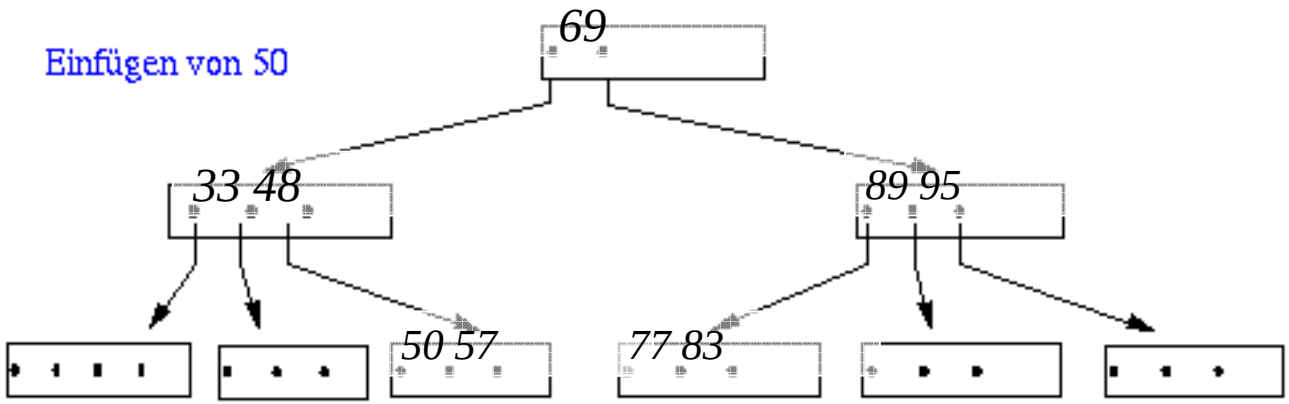
Beispiel: Aufbau eines B-Baumes der Klasse τ(2,h) II



Einfügen von 99



Einfügen von 50



Kostenanalyse für Suche und Einfügen

Kostenmaße

- Anzahl der zu holenden Seiten: f (fetch)
- Anzahl der zu schreibenden Seiten (#geänderter Seiten): w (write)

Direkte Suche

- $f_{min} = 1$: der Schlüssel befindet sich in der Wurzel
- $f_{max} = h$: der Schlüssel ist in einem Blatt
- **mittlere Zugriffskosten** $h - \frac{1}{k} \leq f_{avg} \leq h - \frac{1}{2k}$ (für h > 1)

Beim B-Baum sind die maximalen Zugriffskosten h eine gute Abschätzung der mittleren Zugriffskosten.

- Bei h = 3 und einem k = 100 ergibt sich $2.99 \leq f_{avg} \leq 2.995$

Sequentielle Suche

- Durchlauf in symmetrischer Ordnung: $f_{seq} = N$
- Pufferung der Zwischenknoten im Hauptspeicher wichtig!

Einfügen

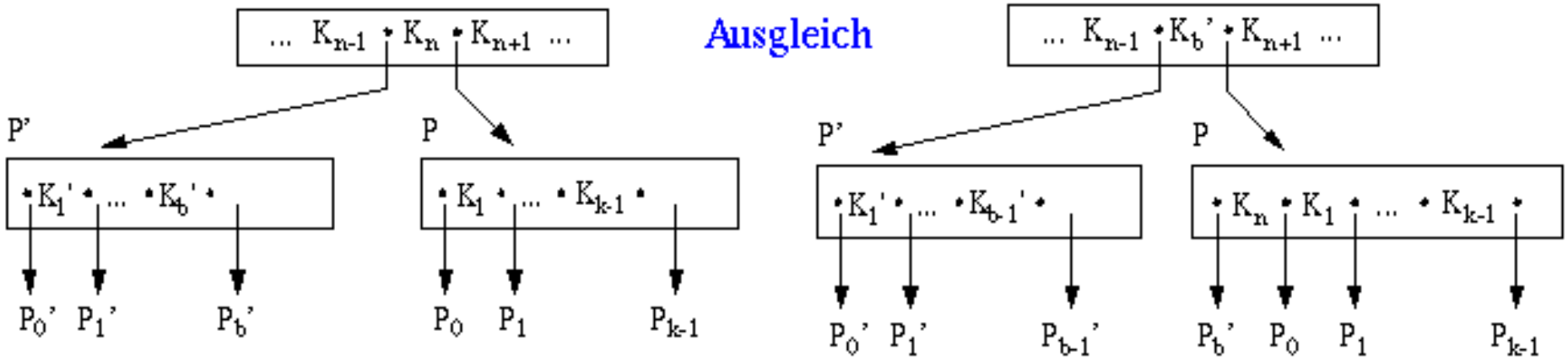
- günstigster Fall - kein Split-Vorgang: $f_{min} = h$; $w_{min} = 1$
- durchschnittlicher Fall: $f_{avg} = h;\ w_{avg} < 1 + \frac{2}{k}$

Löschen in B-Bäumen I

Die B-Baum-Eigenschaft muss wiederhergestellt werden, wenn die Anzahl der Elemente in einem Knoten kleiner als k wird.

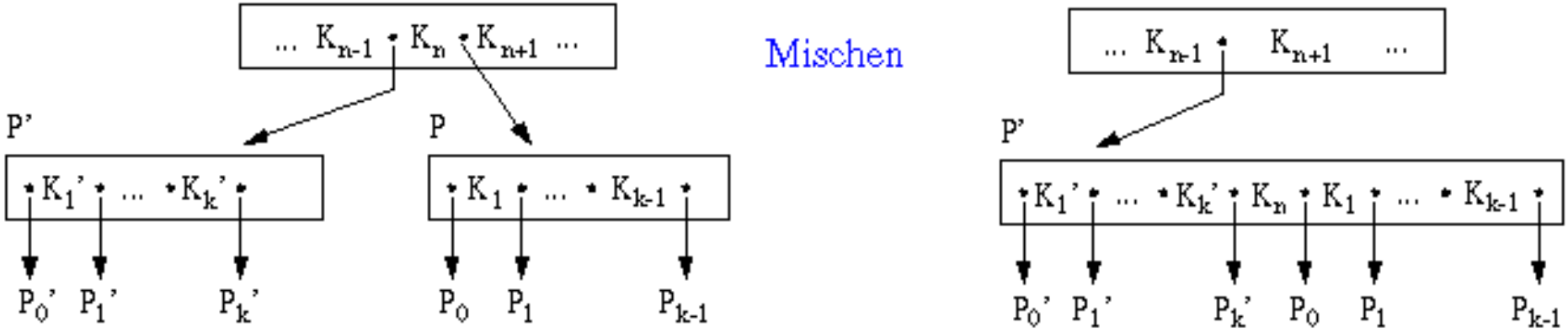
Durch Ausgleich mit Elementen aus einer Nachbarseite oder durch Mischen Konkatenation mit einer Nachbarseite wird dieses Problem gelöst.

- Maßnahme 1: Ausgleich durch Verschieben von Schlüsseln (Voraussetzung: Nachbarseite P' hat mehr als k Elemente; Seite P hat k-1 Elemente (oder mehr))



Löschen in B-Bäumen I

Maßnahme 2: Mischen von Seiten



Löschalgorithmus

(1) Löschen in Blattseite

- Suche x in Seite P
- Entferne x in P und wenn

  a) #E ≥ k in P: tue nichts

  b) #E = k-1 in P und #E > k in P': gleiche Unterlauf über P' aus

  c) #E = k-1 in P und #E = k in P': mische P und P'.

(2) Löschen in innerer Seite

- Suche x
- Ersetze x = K_i durch kleinsten Schlüssel y in $B(P_i)$ oder größten Schlüssel y in $B(P_{i-1})$ (d.h. nächstgrößerer oder nächstkleinerer Schlüssel im Baum)
- Entferne y im Blatt P
- Behandle P wie unter 1

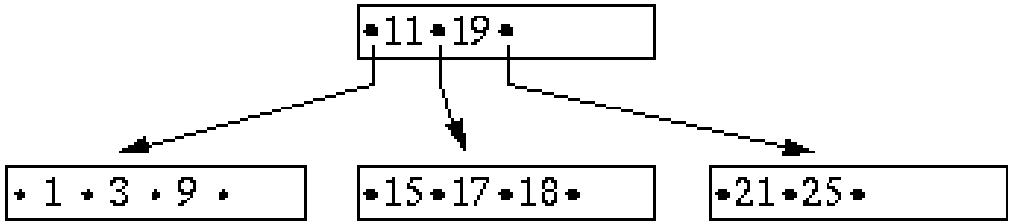
Kostenanalyse für das Löschen

- günstigster Fall: $f_{min} = h$; $w_{min} = 1$;
- obere Schranke für durchschnittliche Löschkosten
  (drei Anteile: 1. Löschen, 2. Ausgleich, 3. anteilige Mischkosten):

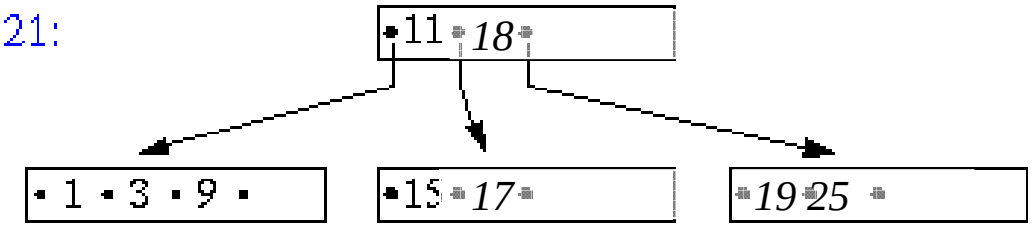
$$f_{avg} \leq f_1 + f_2 + f_3 < h + 1 + \frac{1}{k}$$

$$w_{avg} \leq w_1 + w_2 + w_3 < 2 + 2 + \frac{1}{k} = 4 + \frac{1}{k}$$

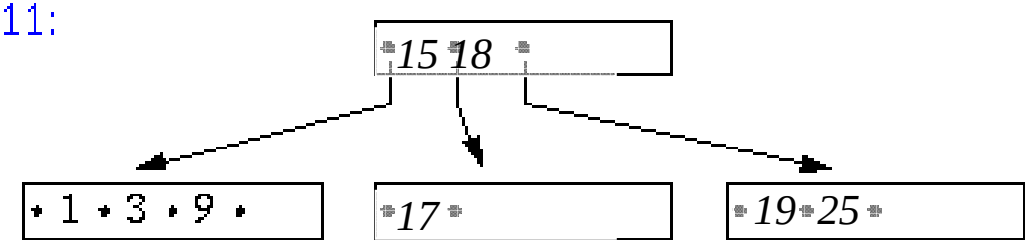
Löschen in B-Bäumen: Beispiel

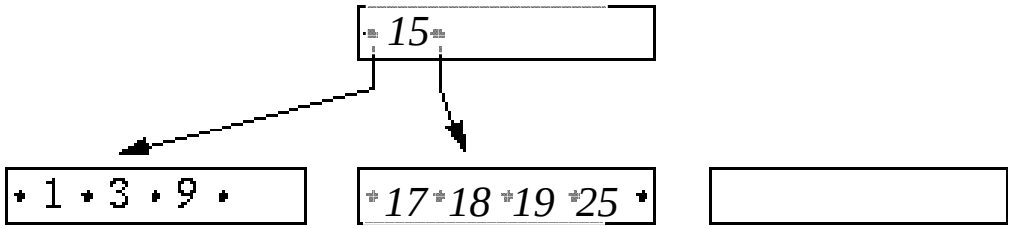


Lösche 21:



Lösche 11:





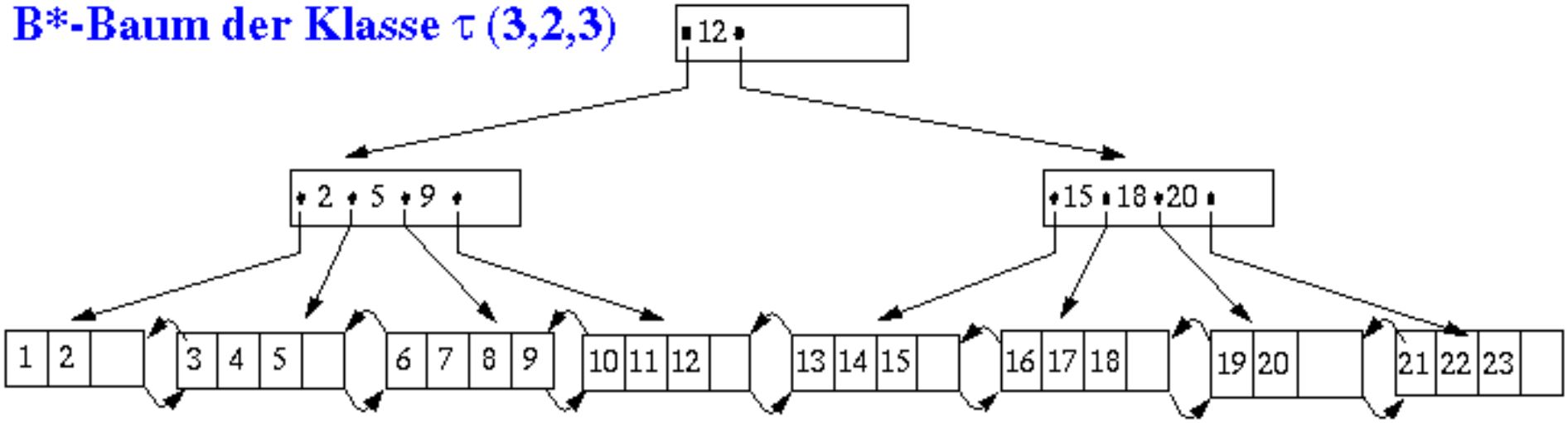
B*-Bäume I

Hauptunterschied zu B-Baum: in inneren Knoten wird nur die Wegweiser-Funktion ausgenutzt

- innere Knoten führen nur (K_i, P_i) als Einträge
- Information (K_i, D_i) wird in den Blattknoten abgelegt. Dabei werden alle Schlüssel mit ihren zugehörigen Daten in Sortierreihenfolge in den Blättern abgelegt werden.
- Für einige K_i ergibt sich eine redundante Speicherung. Die inneren Knoten bilden also einen Index, der einen schnellen direkten Zugriff zu den Schlüsseln gestattet.
- Der Verzweigungsgrad erhöht sich beträchtlich, was wiederum die Höhe des Baumes reduziert
- Durch Verkettung aller Blattknoten lässt sich eine effiziente sequentielle Verarbeitung erreichen, die beim B-Baum einen umständlichen Durchlauf in symmetrischer Ordnung erforderte

B*-Baum ist die für den praktischen Einsatz wichtigste Variante des B-Baums

B*-Bäume II

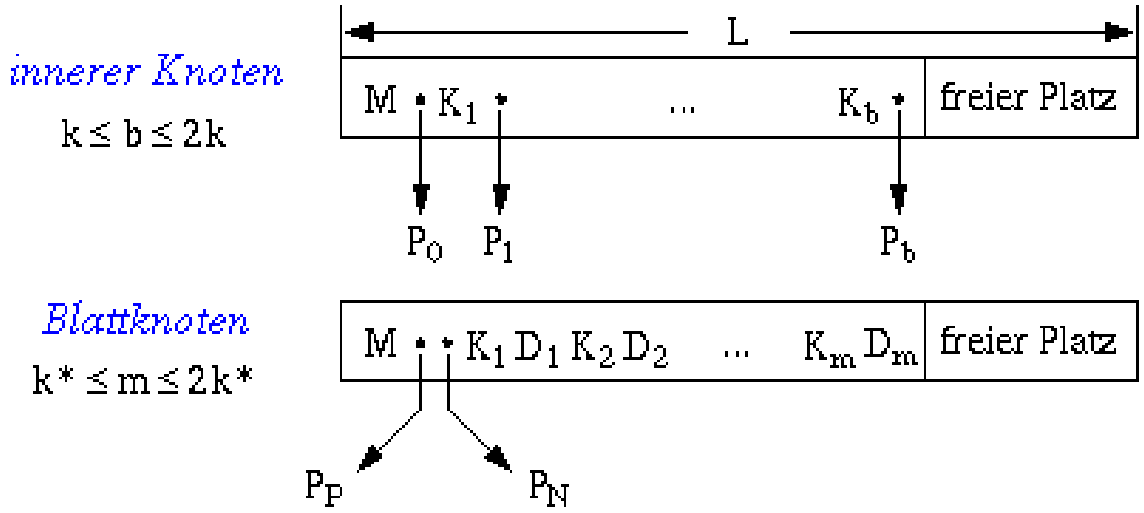

B*-Baum der Klasse τ (3,2,3)
12
2 5 9
15 18 20
1 2
3 4 5
6 7 8 9
10 11 12
13 14 15
16 17 18
19 20
21 22 23

B*-Bäume III

Def.: Seien k, k* und h* ganze Zahlen, h*>0, k, k* > 0.
Ein Baum B der Klasse τ(k, k*, h*) ist entweder ein leerer Baum oder ein geordneter Baum, für den gilt:

1. Jeder Pfad von der Wurzel zu einem Blatt besitzt die gleiche Länge h*-1.
2. Jeder Knoten außer der Wurzel und den Blättern hat mindestens k+1 Söhne, die Wurzel mindestens 2 Söhne, außer wenn sie ein Blatt ist.
3. Jeder innere Knoten hat höchstens 2k+1 Söhne.
4. Jeder Blattknoten mit Ausnahme der Wurzel als Blatt hat mindestens k* und höchstens 2k* Einträge.

Unterscheidung von zwei Knotenformaten



Feld M enthalte Kennung des Seitentyps sowie Zahl der aktuellen Einträge

Parameter eines B*-Baumes

Da die Seiten eine feste Länge L besitzen, lässt sich aufgrund der obigen Formate k und k* bestimmen:

$$L = l_M + l_P + 2 \cdot k(l_K + l_P) \; ; \qquad k = \left\lfloor \frac{L - l_M - l_P}{2 \cdot (l_K + l_P)} \right\rfloor$$

$$L = l_M + 2 \cdot l_P + 2 \cdot k^*(l_K + l_D) \; ; \; k^* = \left\lfloor \frac{L - l_M - 2l_P}{2 \cdot (l_K + l_D)} \right\rfloor$$

Höhe des B*-Baumes

$$1 + \log_{2k+1}\left(\frac{n}{2k^*}\right) \leq h^* \leq 2 + \log_{k+1}\left(\frac{n}{2k^*}\right) \quad \text{für} \quad h^* \geq 2 \; .$$

B- und B*-Bäume: Quantitativer Vergleich

Seitengröße sei L= 2048 B. Zeiger Pi, Hilfsinformation und Schlüssel Ki seien 4 B lang.

Fallunterscheidung:

- eingebettete Speicherung: lD = 76 Bytes
- separate Speicherung: lD = 4 Bytes, d.h., es wird nur ein Zeiger gespeichert.

Allgemeine Zusammenhänge:

	B-Baum	B*-Baum
n_{min}	$2 \cdot (k+1)^{h-1} - 1$	$2k^* \cdot (k+1)^{h^*-2}$
n_{max}	$(2k+1)^{h} - 1$	$2k^* \cdot (2k+1)^{h^*-1}$

Vergleich für Beispielwerte:

B-Baum

B-Baum: 2k(lK+lD+lp)=L-lM-lp

h	Datensätze separat (k=85) n_{min}	Datensätze separat (k=85) n_{max}	Datensätze eingebettet (k=12) n_{min}	Datensätze eingebettet (k=12) n_{max}
1	1	170	1	24
2	171	29.240	25	624
3	14.791	5.000.210	337	15.624
4	1.272.112	855.036.083	4.393	390.624

B*-Baum

h	Datensätze separat (k=127, k* = 127) n_{min}	Datensätze separat (k=127, k* = 127) n_{max}	Datensätze eingebettet (k=12, k* = 127) n_{min}	Datensätze eingebettet (k=12, k* = 127) n_{max}
1	1	254	1	24
2	254	64.770	24	6.120
3	32.512	16.516.350	3.072	1.560.600
4	4.161.536	4.211.669.268	393.216	397.953.001

Historie und Terminologie

Originalpublikation B-Baum:

- R. Bayer, E. M. McCreight. Organization and Maintenance of Large Ordered Indexes. Acta Informatica, 1:4. 1972. 290-306.

Überblick:

- D. Comer: The Ubiquitous B-Tree. ACM Computing Surveys, 11:2, Juni 1979, pp. 121-137.

B*-Baum Originalpublikation:

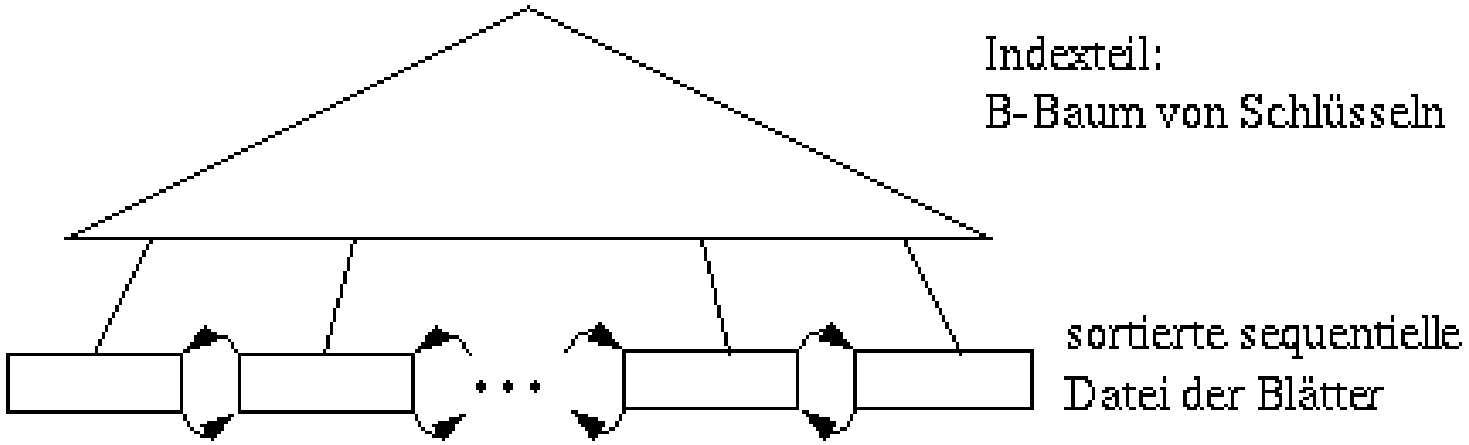
- D. E. Knuth: The Art of Programming, Vol. 3, Addison-Wesley, 1973.

Terminologie:

- Bei Knuth: B*-Baum ist ein B-Baum mit garantierter 2 / 3-Auslastung der Knoten
- B+-Baum ist ein Baum wie hier dargestellt
- Heutige Literatur: B*-Baum = B+-Baum.

B*-Bäume: Operationen

B*-Baum entspricht einer geketteten sequentiellen Datei von Blättern, die einen Indexteil besitzt, der selbst ein B-Baum ist. Im Indexteil werden insbesondere beim Split-Vorgang die Operationen des B-Baums eingesetzt.



Grundoperationen beim B*-Baum

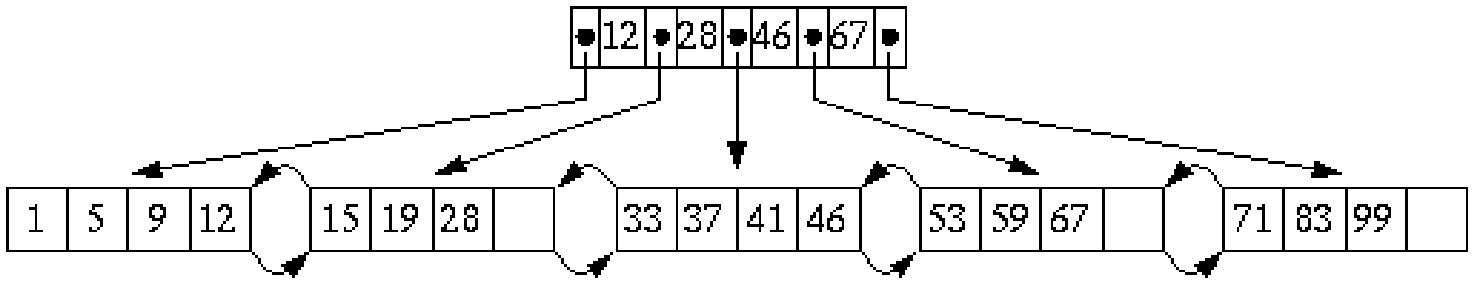
(1) Direkte Suche: Da alle Schlüssel in den Blättern, kostet jede direkte Suche h* Zugriffe. h* ist jedoch im Mittel kleiner als h in B-Bäumen, da k größer ist wegen des Fehlens der Daten (günstigeres f_{avg} als beim B-Baum)

(2) Sequentielle Suche: Sie erfolgt nach Aufsuchen des Linksaußen der Struktur unter Ausnutzung der Verkettung der Blattseiten. Es sind zwar ggf. mehr Blätter als beim B-Baum zu verarbeiten, doch da nur h*-1 innere Knoten aufzusuchen sind, wird die sequentielle Suche ebenfalls effizienter ablaufen.

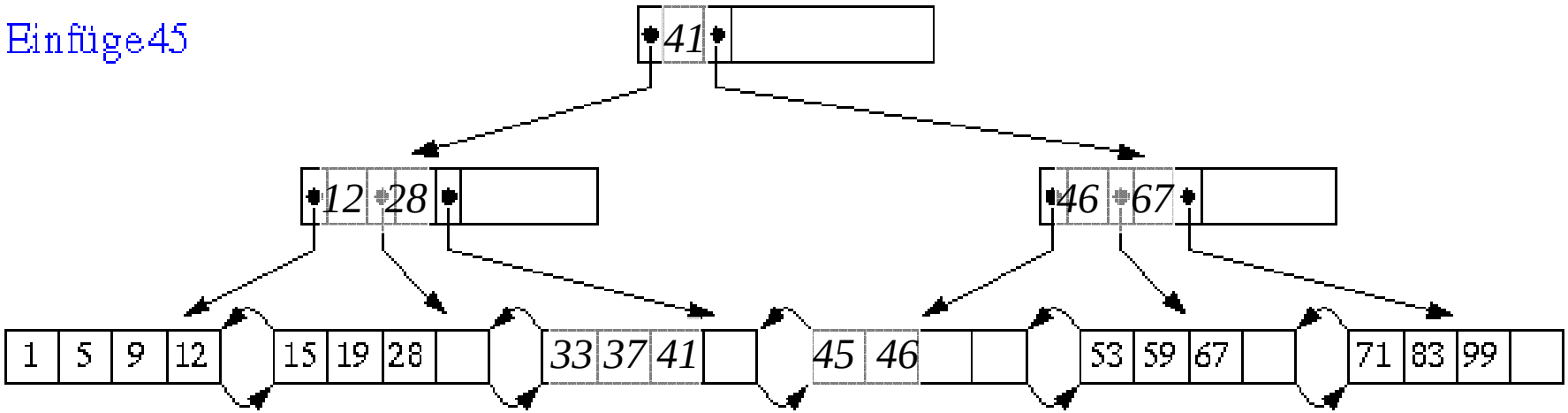
(3) Einfügen: Von Durchführung und Leistungsverhalten dem Einfügen in einen B-Baum sehr ähnlich. Bei inneren Knoten wird die Spaltung analog zum B-Baum durchgeführt. Beim Split-Vorgang einer Blattseite muss gewährleistet sein, dass jeweils die höchsten Schlüssel einer Seite als Wegweiser in den Vaterknoten kopiert werden.

(4) Löschen: Datenelemente werden immer von einem Blatt entfernt (keine komplexe Fallunterscheidung wie beim B-Baum). Weiterhin muss beim Löschen eines Schlüssels aus einem Blatt dieser Schlüssel nicht aus dem Indexteil entfernt werden; er behält seine Funktion als Wegweiser.

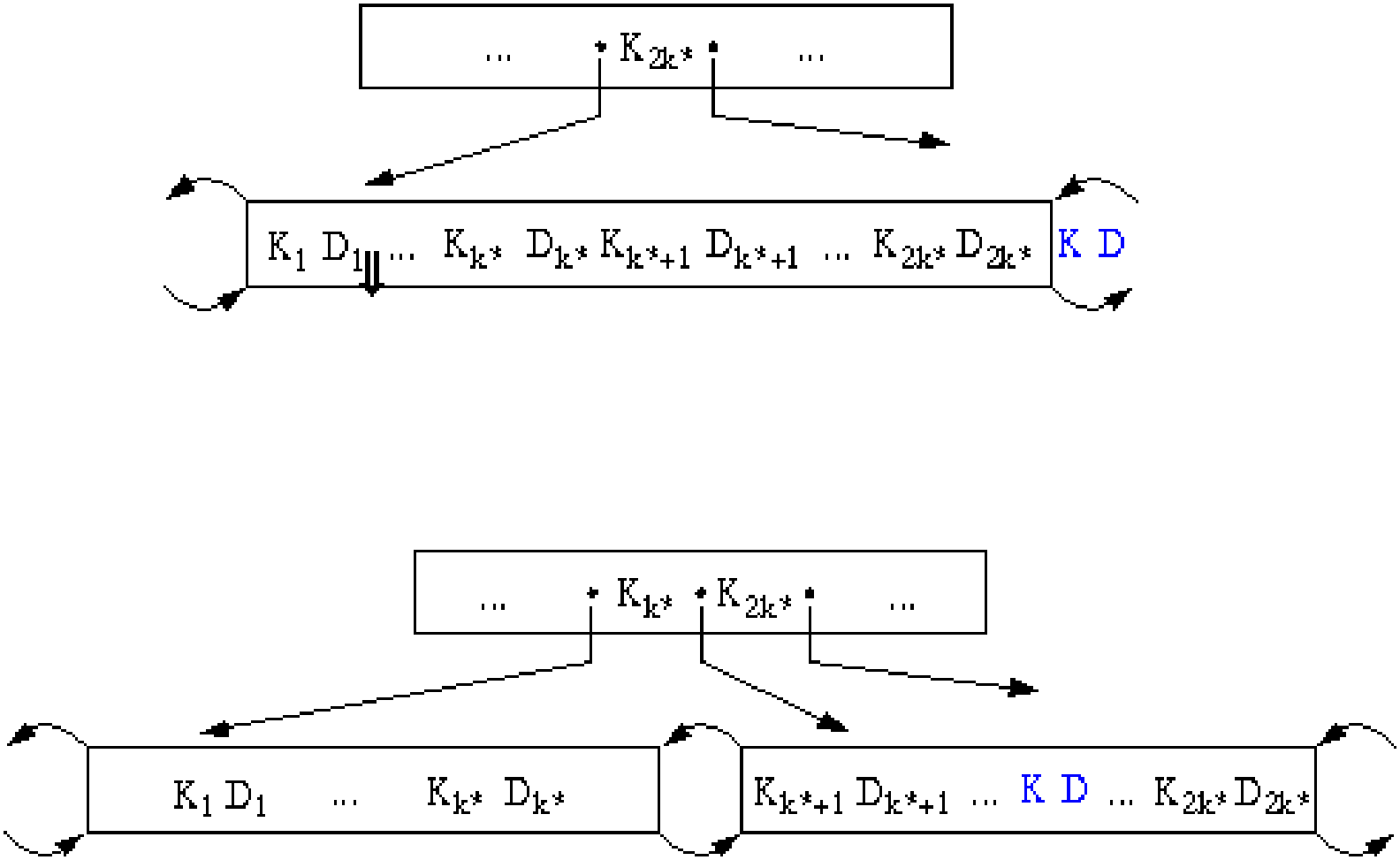
Einfügen im B*-Baum



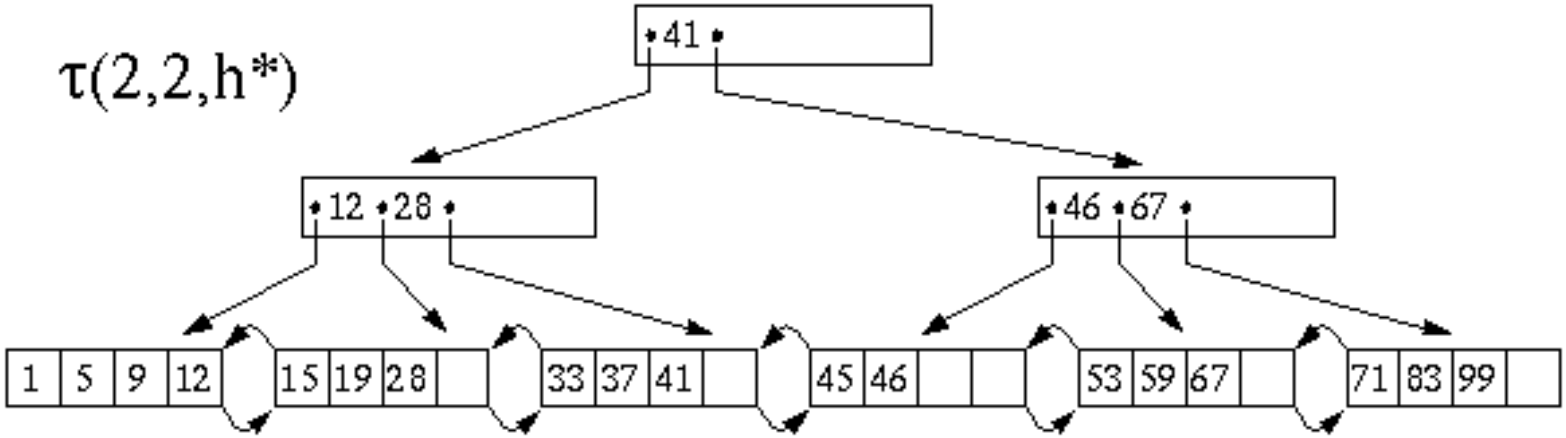
Einfüge45



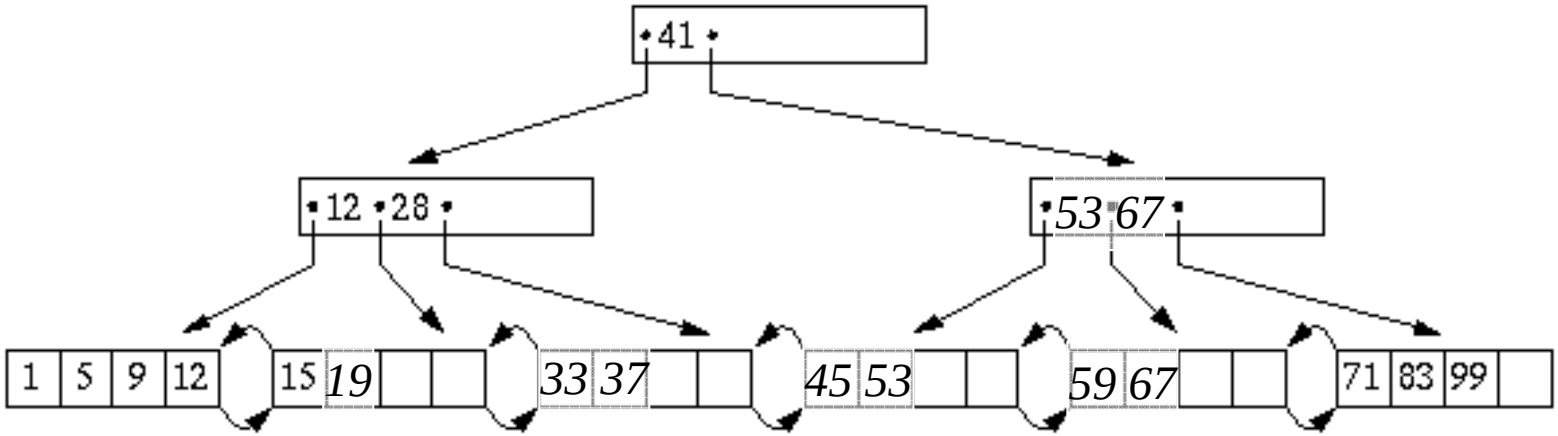
B*-Bäume: Schema für Split-Vorgang



Löschen im B*-Baum: Beispiel

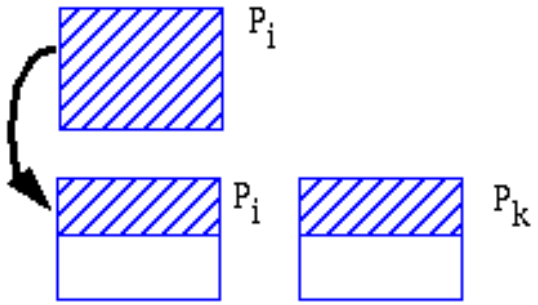


Lösche 28, 41, 46



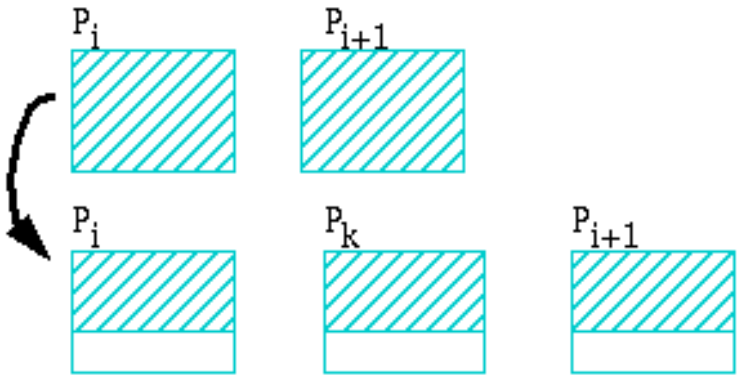
Verallgemeinerte Überlaufbehandlung I

Standard (m=1): Überlauf führt zu zwei halb vollen Seiten

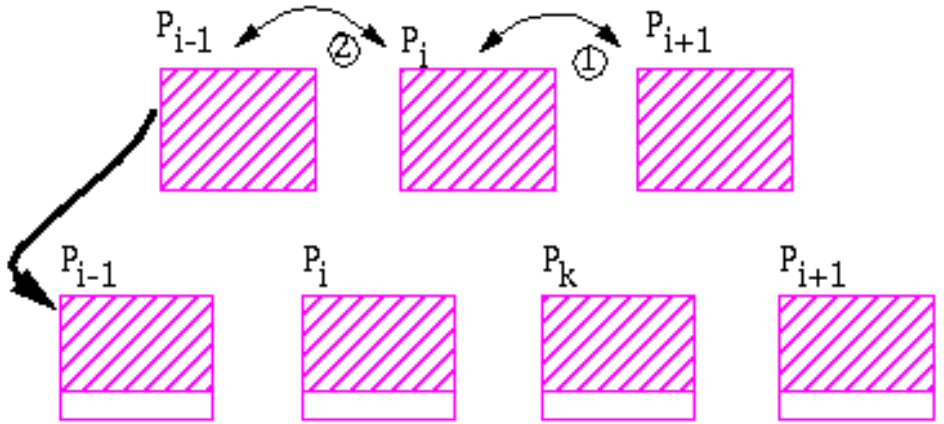


m > 1: Verbesserung der Belegung

m = 2



m = 3



Verallgemeinerte Überlaufbehandlung II

Speicherplatzbelegung als Funktion des Split-Faktors

Split-Faktor	Belegung β_{min}	Belegung β_{avg}	Belegung β_{max}
1	$1/2 = 50\%$	$\ln 2 \approx 69\%$	1
2	$2/3 = 66\%$	$2 \cdot \ln(3/2) \approx 81\%$	1
3	$3/4 = 75\%$	$3 \cdot \ln(4/3) \approx 86\%$	1
m	$\frac{m}{m+1}$	$m \cdot \ln\left(\frac{m+1}{m}\right)$	1

- **Vorteile der höheren Speicherbelegung**
  - geringere Anzahl von Seiten reduziert Speicherbedarf
  - geringere Baumhöhe
  - geringerer Aufwand für direkte Suche
  - geringerer Aufwand für sequentielle Suche
- **erhöhter Split-Aufwand (m > 3 i.a. zu teuer)**