

2.5 Bäume

- 2.5.1 Binäre Suchbäume
- 2.5.2 Optimale Suchbäume
- 2.5.3 Balancierte Bäume
- 2.5.4 Skip-Listen
- 2.5.5 Union-Find-Strukturen



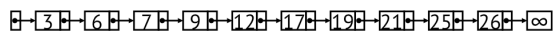
Idee

- Worst-Case Aufwand für Search(), Insert(), Delete() bei Rot-Schwarz-Bäumen ist $O(\log n)$
- Schwer zu implementieren




Idee

- Idee: Halte eine einfach verkettete sortierte Liste



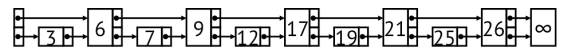
Suche $\leq n$


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 



Lineare Suchzeit! => Füge Sprungmarken/Abkürzungen ein!

Idee

- Idee: Halte eine einfach verkettete sortierte Liste
- Füge Abkürzungen zwischen den Elementen ein



Suche $\leq \lceil n/2 \rceil + 1$


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 

Hierarchische Abkürzungen!

Idee

- Idee: Halte eine einfach verkettete sortierte Liste
- Füge Abkürzungen zwischen den Elementen ein
- Füge Abkürzungen zwischen den Abkürzungen ein

Suche $\leq \lceil n/4 \rceil + 2$

5

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

9 hat 3 Level => 3 Nachfolger!
 17 hat Level 2 => 2 Nachfolger!
 ...
 Sentinel am Ende (NIL)

Definition

- Skip-Listen sind sortierte, vorwärts-verkettete Listen
- Jeder Knoten k enthält $\text{level}(k) \geq 1$ Vorwärtszeiger
- Der i -te Vorwärtszeiger eines Knotens k ($1 \leq i \leq \text{level}(k)$) zeigt auf den nächsten Knoten k' mit $\text{level}(k') \geq i$ oder auf NIL
- Der Level eines Knotens wird bei der Erzeugung des Knotens zufällig festgelegt und danach nicht mehr geändert!



6

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

NIL = Sentinel!
 Hohe Level sollten seltener sein, da dann große Sprünge auf hohen Leveln gemacht werden können! Also keine Gleichverteilung! =>

Definition



- Erzeugung eines zufälligen Levels
- RandomLevel()
 $v \leftarrow 1$
 while random() < p and v < MaxLevel do
 $v \leftarrow v + 1$
 return v
- Level-Verteilung für $p = 1 / 2$
 - $\approx 50\%$ Level 1
 - $\approx 25\%$ Level 2
 - ...

 Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer


Geometrische Verteilung!


Search

- Search(list, search_key)
 $x \leftarrow \text{list.head}$
 for $i \leftarrow \text{list.level}$ downto 1 do
 while $x.\text{next}(i).\text{key} < \text{search_key}$ do
 $x \leftarrow x.\text{next}(i)$
 $x \leftarrow x.\text{next}(i)$
 if $x.\text{key} = \text{search_key}$ then
 return x
 else
 return not found;

 Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer



Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Search

Search(19)

The diagram shows a B-tree with a root node and nine leaf nodes. The root node contains keys 3, 6, 9, 12, 17, 19, 21, 25, and 26. The leaf nodes contain values 3, 7, 9, 12, 17, 19, 21, 25, and 26. An orange arrow points from the root node to the leaf node containing the value 19.

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 FRIEDRICH-ALEXANDER
 UNIVERSITÄT
 ERLANGEN-NÜRNBERG

Search


Search(19)

The diagram shows a B-tree with a root node and nine leaf nodes. The root node contains keys 3, 6, 9, 12, 17, 19, 21, 25, and 26. The leaf nodes contain values 3, 7, 9, 12, 17, 19, 21, 25, and 26. An orange arrow points from the root node to the leaf node containing the value 19.

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 FRIEDRICH-ALEXANDER
 UNIVERSITÄT
 ERLANGEN-NÜRNBERG


Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer


Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer


Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Search

Search(19)

The diagram illustrates a B-tree search for the value 19. The root node contains keys 6, 9, 17, 25, and ∞ . The leaf nodes contain values 3, 7, 9, 12, 17, 19, 21, 25, and 26. Arrows show the search path starting from the root, moving to the leaf containing 19, and then to the leaf containing 17.

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 FRIEDRICH-ALEXANDER
 UNIVERSITÄT
 ERLANGEN-NÜRNBERG

Search


Search(19)

The diagram illustrates a B-tree search for the value 19. The root node contains keys 6, 9, 17, 25, and ∞ . The leaf nodes contain values 3, 7, 9, 12, 17, 19, 21, 25, and 26. Arrows show the search path starting from the root, moving to the leaf containing 19, and then to the leaf containing 19.

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 FRIEDRICH-ALEXANDER
 UNIVERSITÄT
 ERLANGEN-NÜRNBERG


Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Search

Search(19)


Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Search

Search(19)

gefunden!

21

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-SCHILLER
UNIVERSITÄT

Insert

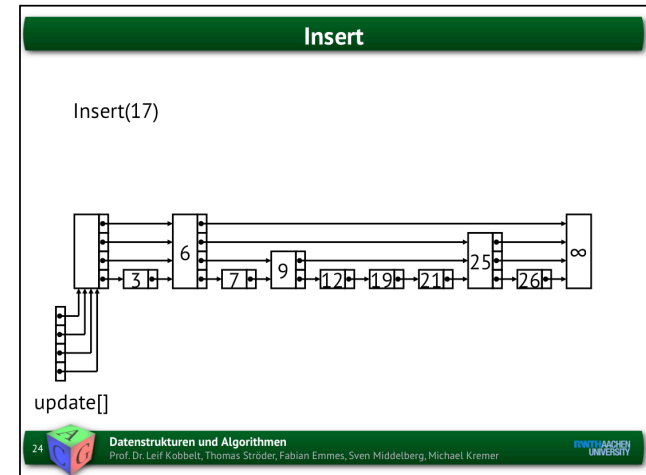
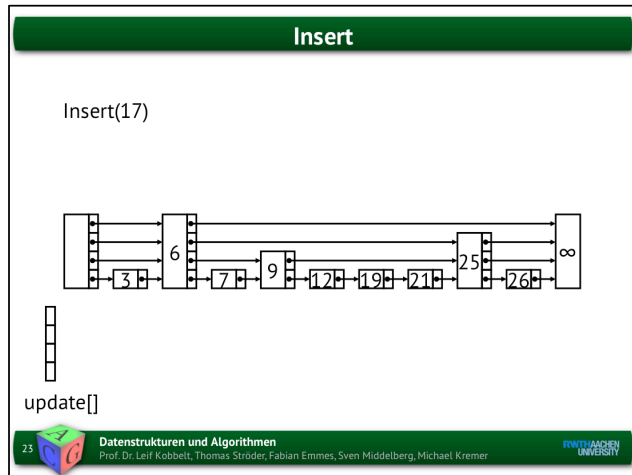
- Insert(list, search_key)
 - update[1..list.level] ← list.head
 - x ← list.head
 - for i ← list.level downto 1 do
 - while x.next(i).key < search_key do
 - x ← x.next(i)
 - update[i] ← x
 - x ← x.next(i)
 - if x.key ≠ search_key then
 - new_level ← RandomLevel()
 - if new_level > list.level then
 - list.level ← new_level
 - x = MakeNode(new_level, search_key)
 - for i ← 1 to new_level do
 - x.next(i) ← update[i].next(i)
 - update[i].next(i) ← x

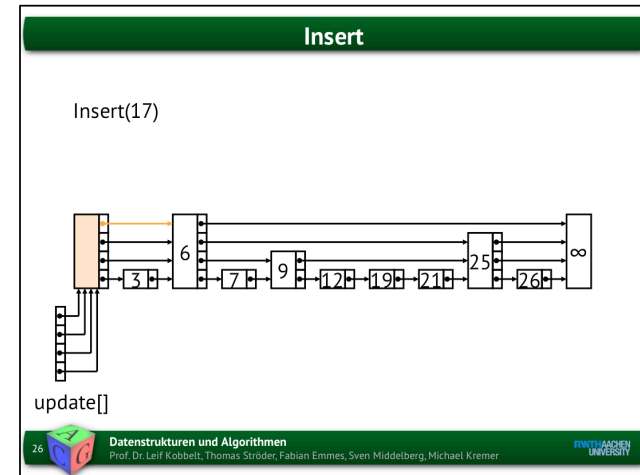
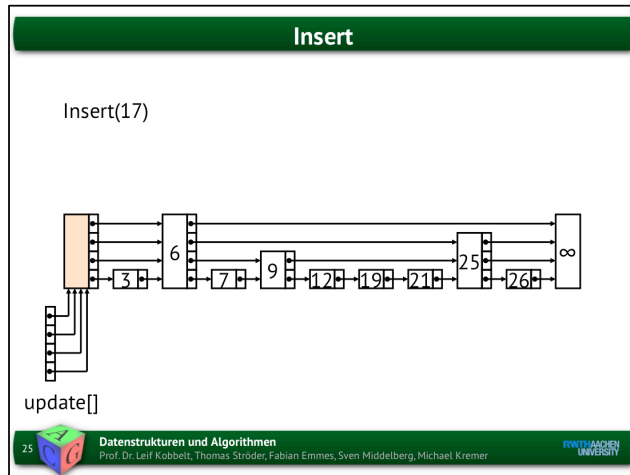
22

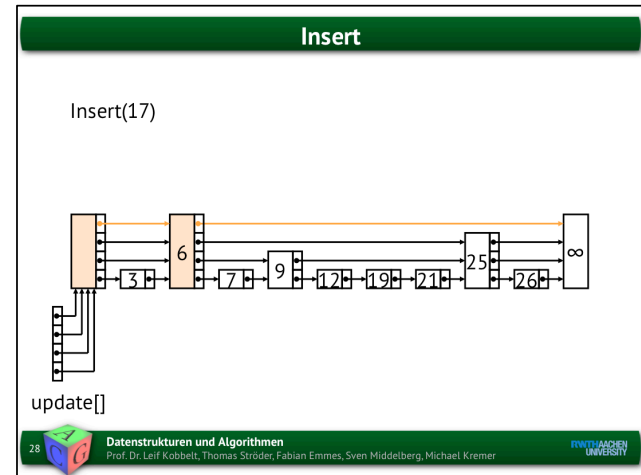
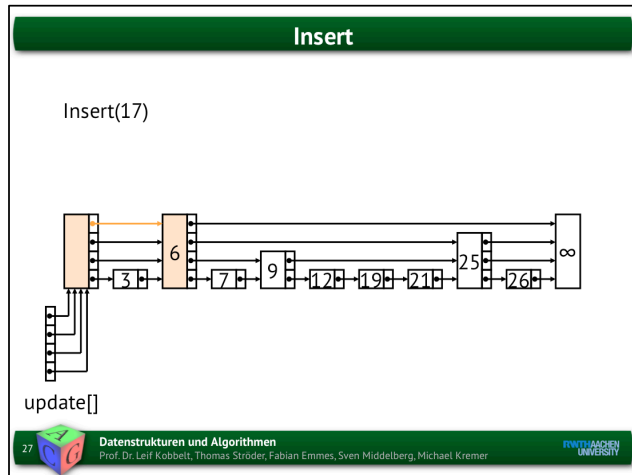
Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

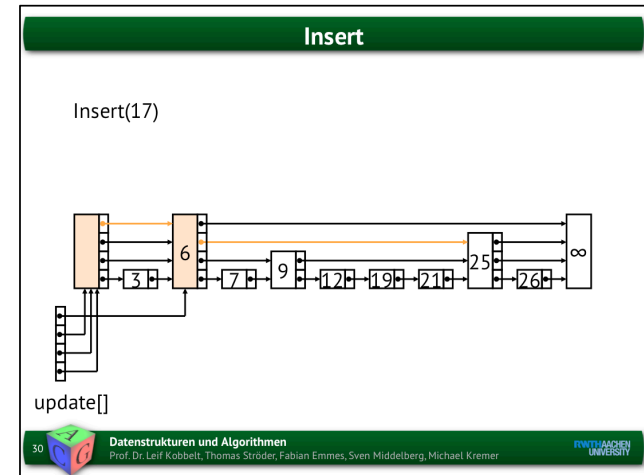
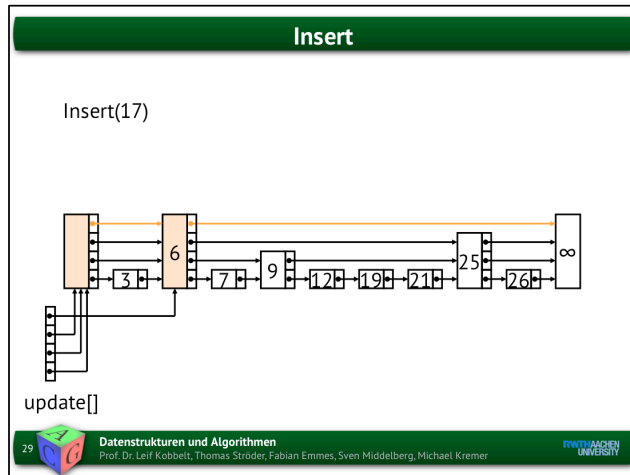
FRIEDRICH-SCHILLER
UNIVERSITÄT

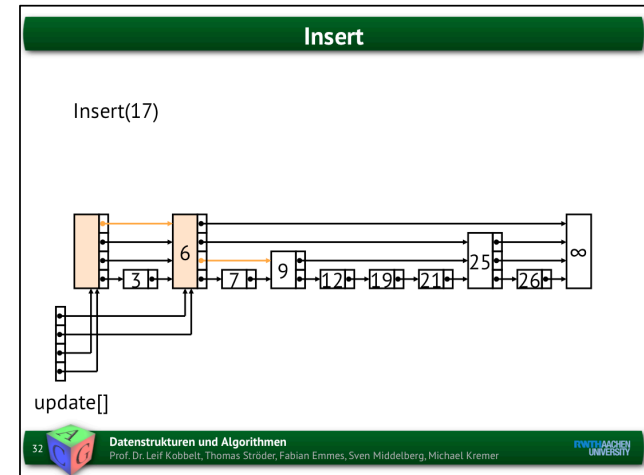
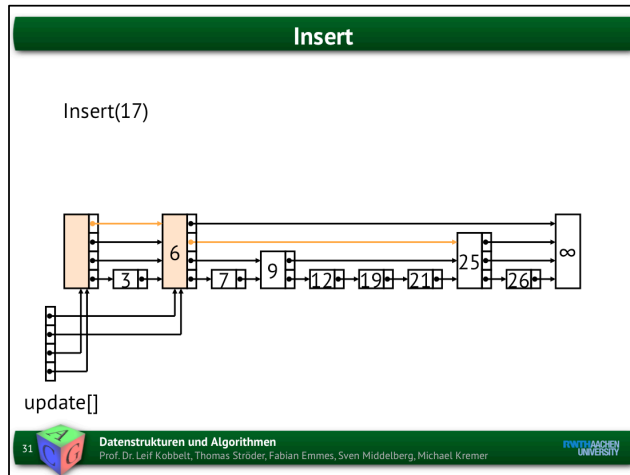
Update Array speichert Vorgänger von neu einzufügendem Knoten

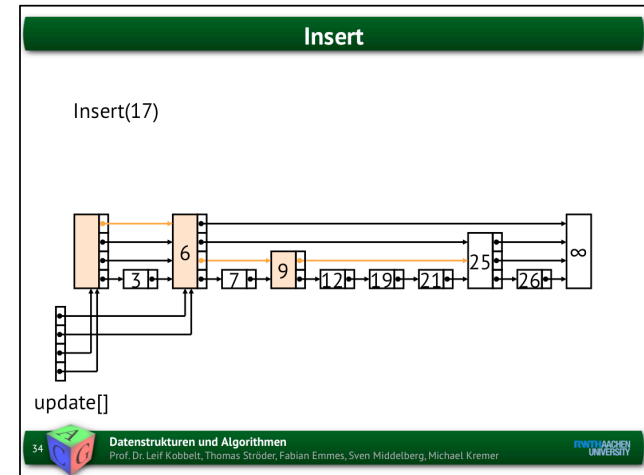
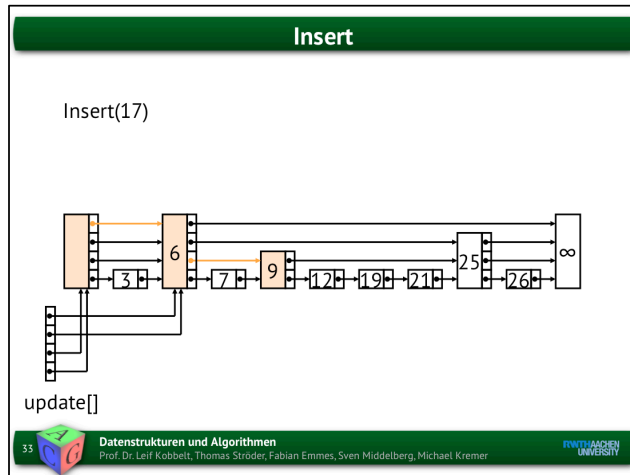


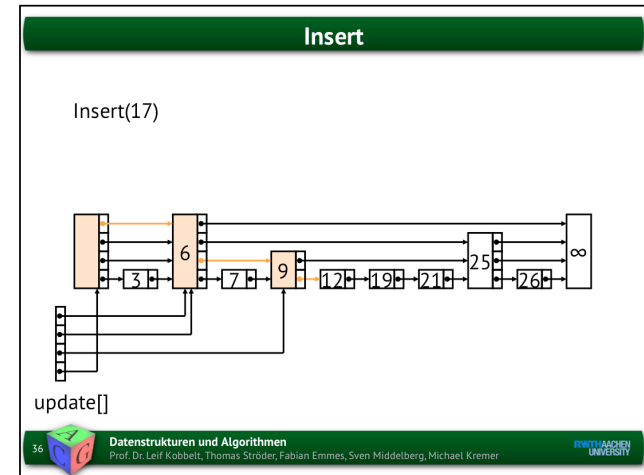
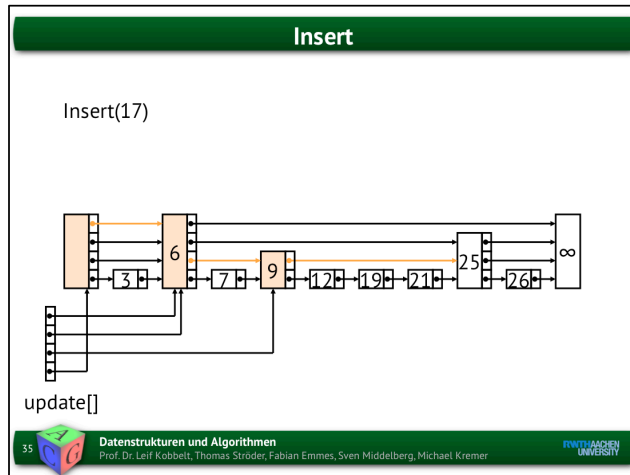


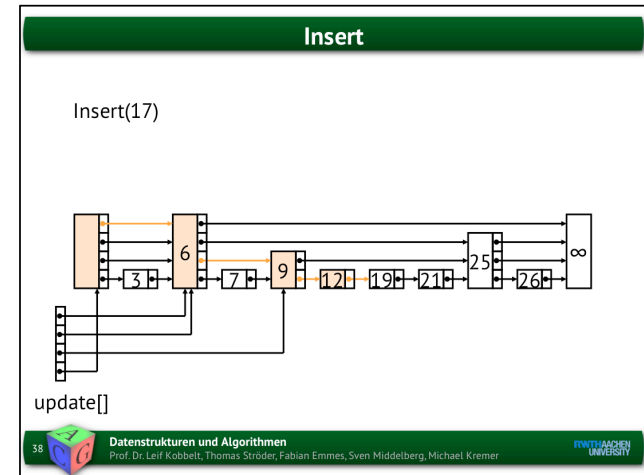
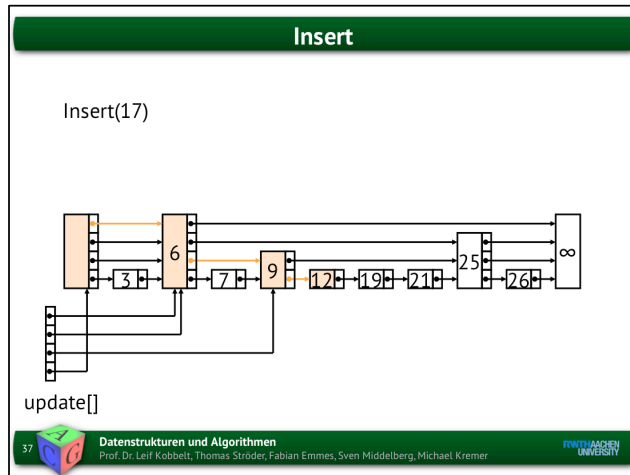


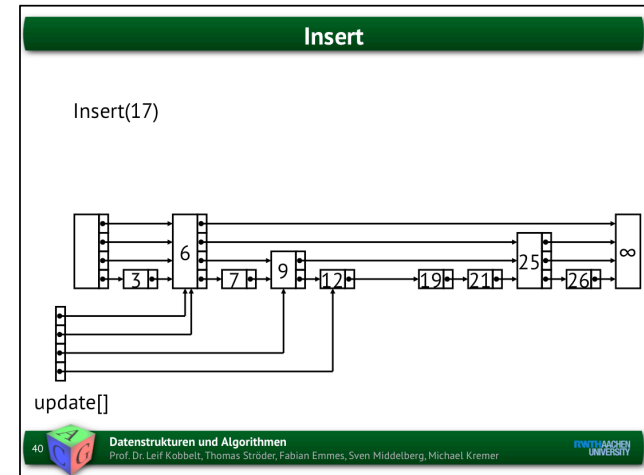
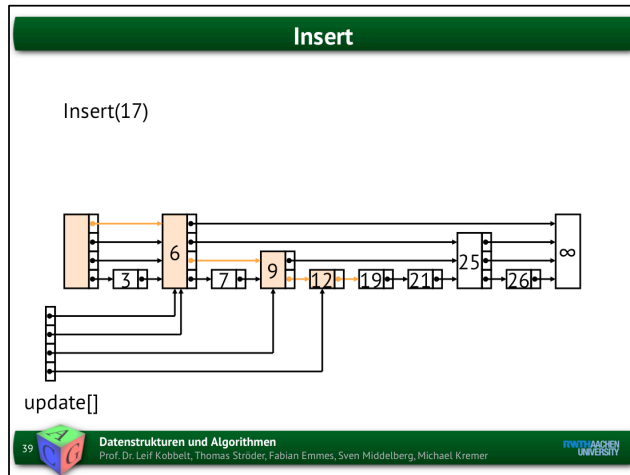


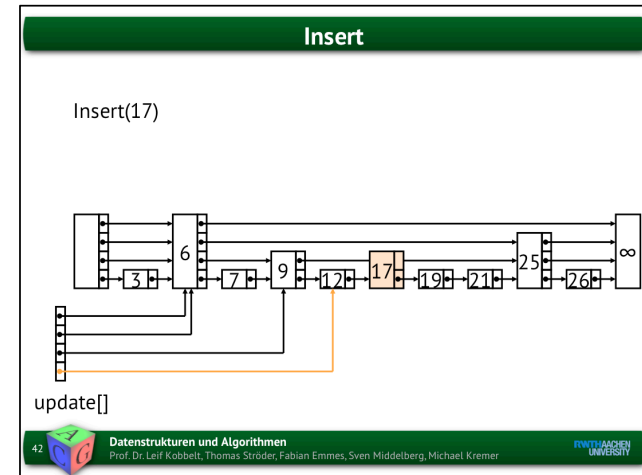
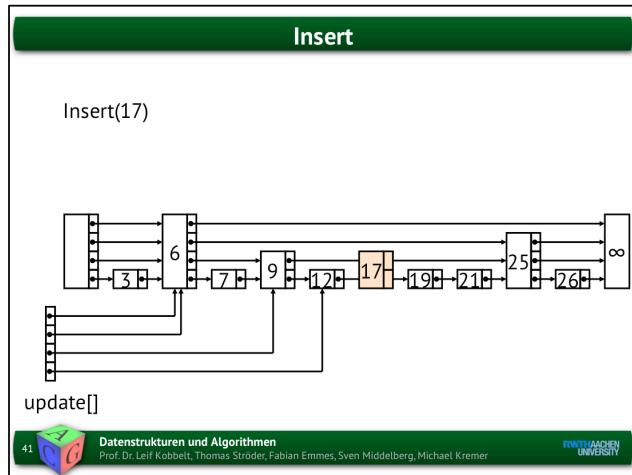


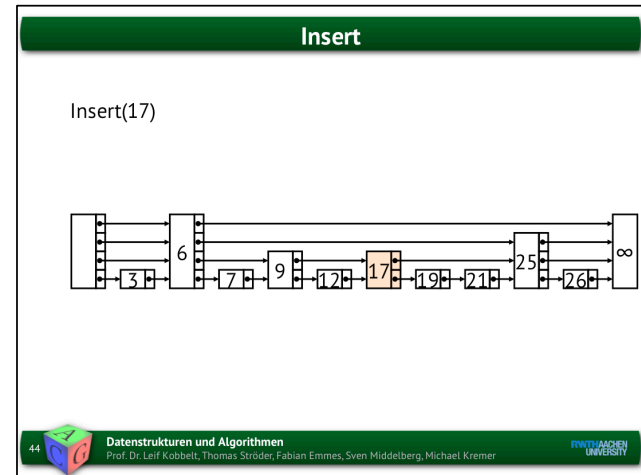
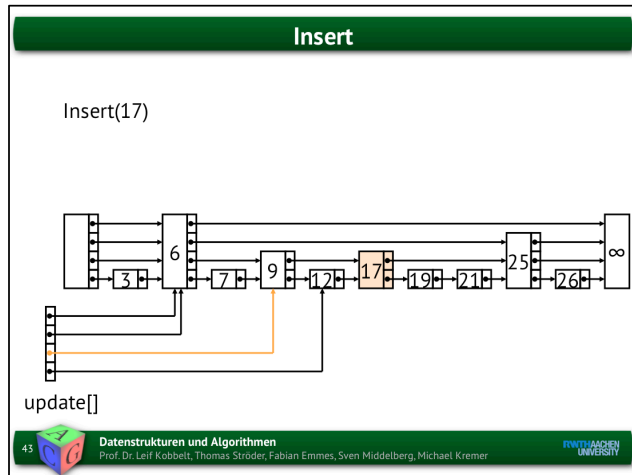












Delete

- Delete(list, search_key)
 - update[1..list.level] ← list.header
 - x ← list.header
 - for i = list.level **downto** 1 do
 - while x.next(i).key < search_key do
 - x ← x.next(i)
 - update[i] ← x
 - x ← x.next(i)
 - if x.key = search_key then
 - for i ← 1 to x.level do
 - update[i].next[i] ← x.next[i]
 - free(x)
 - while list.level > 1 and list.header.next(list.level) = NIL do
 - list.level ← list.level - 1

Letzten beide Zeilen behandeln den Fall, dass der einzige Knoten aus dem höchsten Level gelöscht wird. Dann dekrementiere list.level so lange, bis auf dem höchsten Level wieder Knoten existieren.

Analyse

- Was ist die erwartete Länge des Suchpfads?
- Was ist die erwartete Höhe der Knoten?
- Es sei $L(n) = \log_{1/p} n$, insbesondere $p^{L(n)}=1/n$
- Analysiere die Zahl der horizontalen und vertikalen Schritte getrennt!

$L(n)$ = Level auf dem die erwartete Anzahl von Knoten onstant ist (also unabhängig von n ist)
 = bspw. Level, auf welchem wir genau $1/p$ Knoten erwarten

Analyse

- Was ist die erwartete Länge des Suchpfads?
- Idee: Analysiere den Suchpfad rückwärts.

Search(19)

47 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Analyse

- Was ist die erwartete Länge des Suchpfads?
- Idee: Analysiere den Suchpfad rückwärts.

Search(19)

48 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

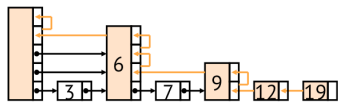
Drei Abschätzungen!

- 1: Anzahl Schritte um auf Level $L(n)$ zu kommen ($O(\log n)$)
- 2: Anzahl Schritte um von Knoten mit Level $L(n)$ nach links zu kommen ($O(1)$)
- 3: Anzahl Schritte um von Knoten mit Level $L(n)$ nach ganz oben zu kommen ($O(\log n)$)

Analyse

- Was ist die erwartete Länge des Suchpfads?
- Idee: Analysiere den Suchpfad rückwärts.

Search(19)



Datenstrukturen und Algorithmen

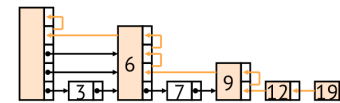
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

UNIVERSITÄT WÜRZBURG

Analyse

- Es sei $C(k)$ die erwartete Anzahl der Schritte, um k Level nach oben zu steigen.
- $C(0) = 0$

Search(19)



Datenstrukturen und Algorithmen

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

UNIVERSITÄT WÜRZBURG

Wenn man 0 Level erklimmen möchte, muss man keinen Schritt machen => $C(0) = 0!$

Analyse

$C_k = (1-p) \times (1+C_k) + p \times (1+C_{k-1})$
 $C_k = 1 + C_k - p - p \times C_k + p + p \times C_{k-1}$
 $C_k = 1 / p + C_{k-1}$
 $C_k = k / p$

51 Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Wahrscheinlichkeit p: es gibt ein weiteres Level, gehe einen Schritt nach oben und man muss ein Level weniger erklimmen=> $(1+C_{k-1})$
 Wahrscheinlichkeit 1-p: es gibt ein weiteres Level, gehe einen Schritt nach links und man muss weiterhin k Level erklimmen=> $(1+C_k)$

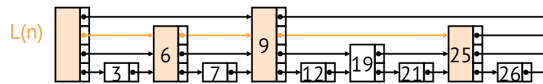
Analyse

1. die erwartete Anzahl der Schritte, um $L(n)-1$ Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p$
2. ...

52 Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Analyse

- Die erwartete Zahl der verbleibenden horizontalen Schritte bis zum Anfang der Liste ist durch die Zahl der Knoten mit $\text{Level} \geq L(n)$ beschränkt.
- Es gibt $np^{L(n)-1} = 1/p$ solcher Knoten



Analyse

- Es sei w_i die Wahrscheinlichkeit, dass ein Knoten q mit $\text{level}(q) \geq i$ erzeugt wird

$$\begin{aligned}
 w_i &= \sum_{j=i}^{\infty} p^{j-1}(1-p) \\
 &= \frac{1-p}{p} \sum_{j=i}^{\infty} p^j \\
 &= \frac{1-p}{p} \left(\sum_{j=0}^{\infty} p^j - \sum_{j=0}^{i-1} p^j \right) \\
 &= \frac{1-p}{p} \left(\frac{1}{1-p} - \frac{1-p^i}{1-p} \right) \\
 &= p^{i-1}
 \end{aligned}$$



Analyse

- Die Wahrscheinlichkeit, dass ein Knoten q mit $\text{level}(q) \geq L(n)$ erzeugt wird, ist also

$$w_{L(n)} = p^{L(n)-1} = \frac{1}{p} p^{\log_{1/p} n} = \frac{1}{pn}$$

und die erwartete Zahl von Knoten mit Level $\geq L(n)$ ist somit $1 / p$



Analyse

1. die erwartete Anzahl der Schritte, um $L(n)-1$ Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p$
2. Die erwartete verbleibende horizontale Länge bis zum Startknoten ist $\leq 1 / p$
3. ...



Analyse

- Im Startknoten müssen wir von Level $L(n)$ zum maximalen Level kommen
- Die Wahrscheinlichkeit, dass der Level eines Elements $> k$ ist, ist p^k
- Die Wahrscheinlichkeit, dass kein Knoten mit Level $> k$ existiert, ist $(1-p^k)^n$
- Die Wahrscheinlichkeit, dass der maximale Level der n Elemente $> k$ ist, ist $1-(1-p^k)^n \leq np^k$
- Ist M der maximale Level, so ist $P(M > k) \leq np^k$



Analyse

$$\begin{aligned} E(M) &= \sum_{k=0}^{L(n)} kP(M = k) + \sum_{k=L(n)+1}^{\infty} kP(M = k) \\ &\leq L(n) + \sum_{k=L(n)+1}^{\infty} kP(M > k-1) \\ &\leq L(n) + \sum_{k=L(n)+1}^{\infty} knp^{k-1} \\ &= L(n) + \sum_{k=0}^{\infty} (k + L(n) + 1)np^{k+L(n)} \end{aligned}$$



Analyse

$$\begin{aligned} E(M) &= \dots \\ &= L(n) + \sum_{k=0}^{\infty} (k + L(n) + 1) n p^{k+L(n)} \\ &= L(n) + \sum_{k=0}^{\infty} (k + L(n) + 1) p^k \\ &= L(n) + \sum_{k=0}^{\infty} k p^k + \sum_{k=0}^{\infty} (L(n) + 1) p^k \\ &= L(n) + \frac{p}{(1-p)^2} + (L(n) + 1) \frac{1}{1-p} \\ &= O(L(n)) \end{aligned}$$



Analyse

1. die erwartete Anzahl der Schritte, um $L(n)-1$ Level nach oben zu steigen, ist $C_{L(n)-1} = (L(n)-1) / p$
2. Die erwartete verbleibende horizontale Länge bis zum Startknoten ist $\leq 1 / p$
3. Der erwartete maximale Level ist $O(L(n))$



Analyse

1. die erwartete Anzahl der Schritte, um $L(n)-1$ Level nach oben zu steigen, ist
 $C_{L(n)-1} = (L(n)-1) / p = O(L(n))$
2. Die erwartete verbleibende horizontale Länge bis zum Startknoten ist $\leq 1 / p = O(1)$
3. Der erwartete maximale Level ist $O(L(n))$



Analyse

- Die erwartete Länge des Suchpfades bei einer Skip-Liste mit n Elementen ist
 $O(L(n)) = O(\log_{1/p} n) = O(\log n)$

