Kap. 4.7 Skiplisten

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

Fakultät für Informatik, TU Dortmund

14./15. VO DAP2 SS 2008 9./16. Juni 2009

2. Übungstest

- **Termin:** Di **16. Juni** 2009 im AudiMax, Beginn: 12:15 Uhr (bitte um 12:00 Uhr anwesend sein)
- **Dauer:** 25 Minuten
- **Stoff:** aus VO-Folien, Skript und Übungen bis inkl. B-Bäume (schwerpunktmäßig ab Heap-Sort inkl.)

- Dann ab ca. 12:50 Uhr: Vorlesung bis 13:45 Uhr

Proseminare WS 2009/10

- **Anmeldefrist:** Montag **1.06.** bis Donnerstag **11.06.**
- Die Verteilung ist unabhängig von der Reihenfolge der Anmeldungen

- Infos und Anmeldung:
- http://www.cs.uni-dortmund.de/proseminar

- Dort steht alles weitere.

real-IT-y Kontaktmesse

- Reality-Kontaktmesse für Informatik
- am 19.06. ab 10:00 Uhr
- OH 14, Informatikgebäude
- siehe http://real-IT-y.de

Motivation

„Warum soll ich jetzt hier bleiben?"

Schöne Analyse der Skiplisten

„Was ist daran Besonderes?"

einfacher und schneller als bisherige

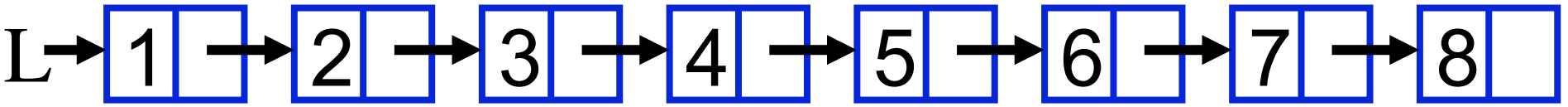
Überblick

- Einführung: Perfekte Skiplisten
- Randomisierte Skiplisten
- Suchen/Einfügen/Entfernen in randomisierte Skiplisten
- Analyse randomisierter Skiplisten

Einführung von Skiplisten

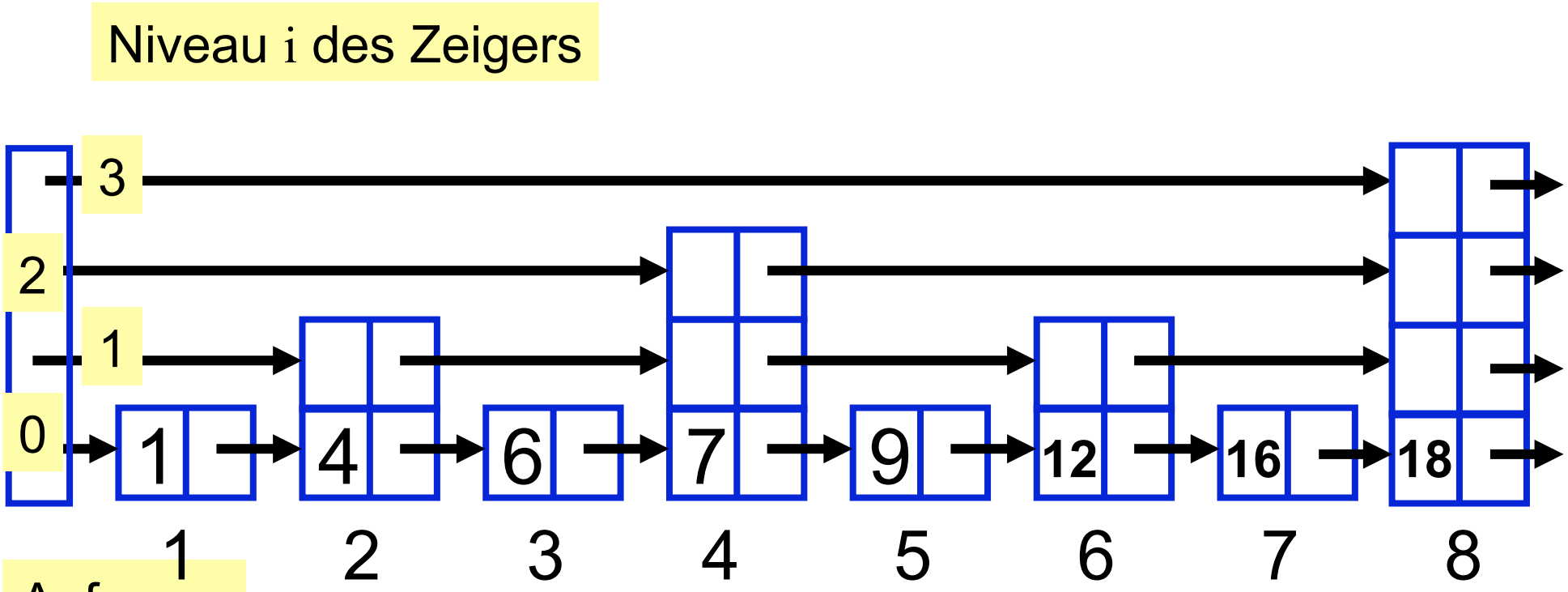
- **Idee:** Simulation von Binary Search auf einfach verketteten Listen
- dies geschieht durch Hinzufügen mehrerer Zeiger auf Nachfolger
- Verallgemeinerung von einfach verketteten Listen

Einfach verkettete Liste:



Perfekte Skiplisten

- **Idee:** Simulation von Binary Search auf einfach verketteten Listen



Perfekte Skiplisten

- Elemente c der Liste besitzen Höhe $h(c)$
- jedes Element c enthält einen Schlüssel und $h(c)+1$ Zeiger auf nachfolgende Listenelemente (LE)
- Anfang und Ende der Liste: je ein Pseudo-Element ohne Daten, gleiche Höhe wie maximales Element in der Liste; Schlüssel des Endelements ist größer als alle anderen
- Zeiger i für $0 \leq i \leq h(c)$ zeigt auf Element, das 2^i Positionen hinter LE c steht oder auf Endelement

Suchen in Skiplisten

Sei s der zu suchende Schlüssel

(1) Folge dem höchsten Zeiger des Anfangselements: der Schlüssel sei s'

(2) Solange $s' < s$: folge dem Zeiger auf gleichem Niveau zu Nachfolgeelement → neues s'

(3) Solange $s' > s$: Reduziere das Niveau um 1: folge diesem Zeiger zu Nachfolger → neues s'

(4) Gehe zu (2)

(5) Falls $s == s'$: STOP: s gefunden

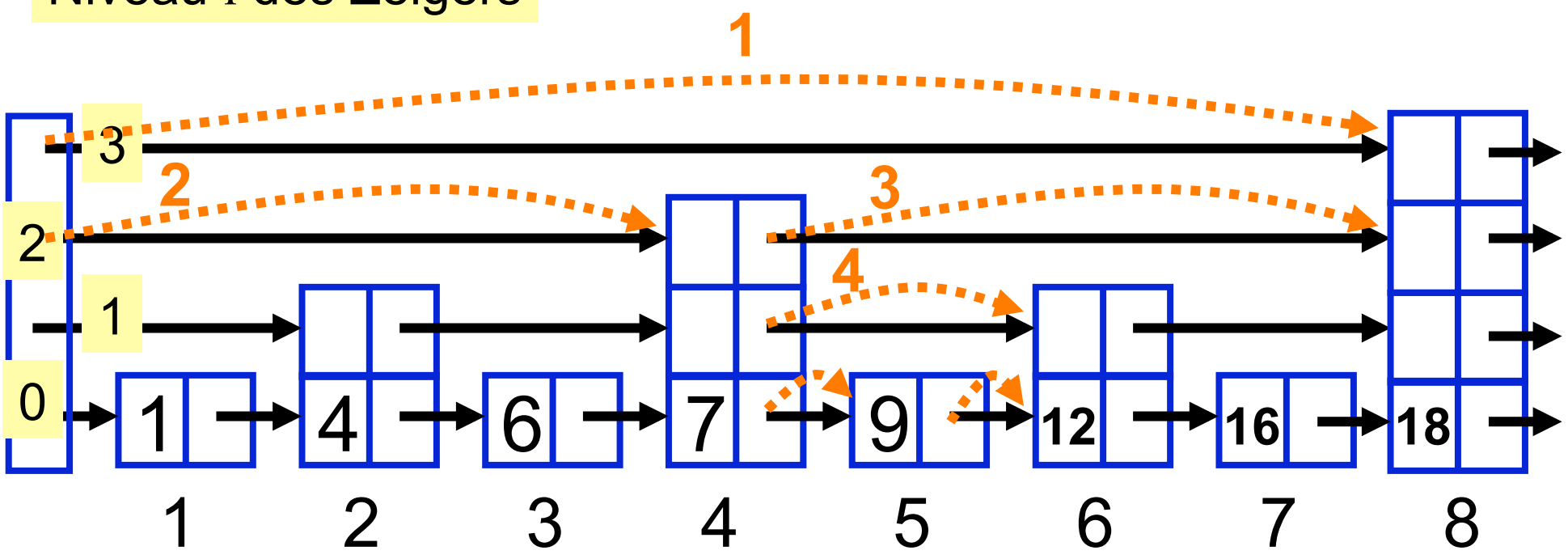
(6) Sonst: wir sind auf Niveau 0 angekommen und haben s nicht gefunden

Suche in Perfekten Skiplisten

Suche nach 12:

Vergleiche

Niveau i des Zeigers



Anfangselement

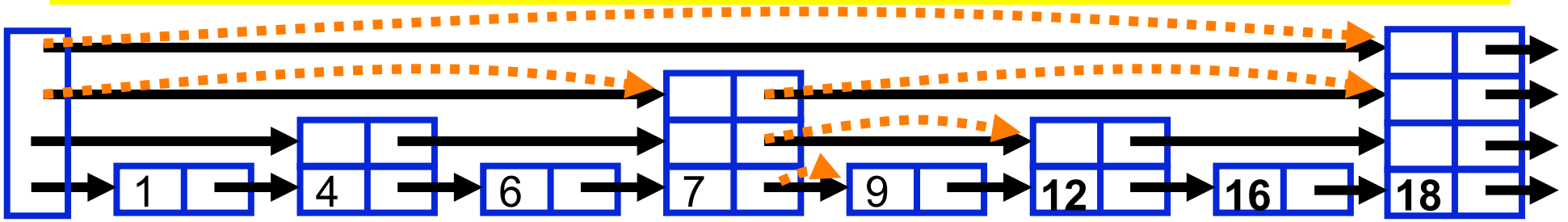
Endelement nicht mehr angezeigt

Datenstruktur einer Skipliste

- c.height: Höhe von Listenelement c
- c.next[i]: Nachfolger auf Niveau i
- L.head: Anfangselement der Liste
- L.height: Höhe des Anfangselements

SEARCH(L,s)

```
(1)  p=L.head
(2)  for i:=L.height to 0 do {
(3)    while p.next[i].key < s do
(4)      p:=p.next[i]
(5)  }
(6)  p:=p.next[0]
(7)  if p.key == s then
(8)    return p
(9)  else return nil
```



Größe der Perfekten Skipliste

Sei $n=2^l$. Die Höhe ist dann also l.

Lemma: Die Anzahl der Zeiger in einer perfekten Skipliste ist $2n + \log n + 1 = O(n)$.

Denn: Die Anzahl aller Zeiger, die nicht von Anfangselement ausgehen:

$$n + \frac{n}{2} + \frac{n}{4} + \cdots + 1 = \sum_{i=0}^{l} \frac{n}{2^i} \leq 2n$$

Die Anzahl aller Zeiger, die vom Anfangselement ausgehen: $l+1=\log n+1$

Analyse der Suchzeit

Beobachtung: Auf jedem Niveau gibt es höchstens zwei Vergleiche.

Denn: Entweder der Schlüssel ist zu groß, dann wird in ein kleineres Niveau abgestiegen. (Es wird nie wieder aufgestiegen.)

Oder der Schlüssel ist zu klein, dann wird ein Vergleich ausgeführt. Aber wir wissen: nach zwei Vergleichen auf dem gleichen Niveau sind wir wieder dort, wo wir schon auf höherem Niveau verglichen haben.

Lemma: Die Suchzeit in einer perfekten Skipliste ist durch $O(\log n)$ beschränkt.

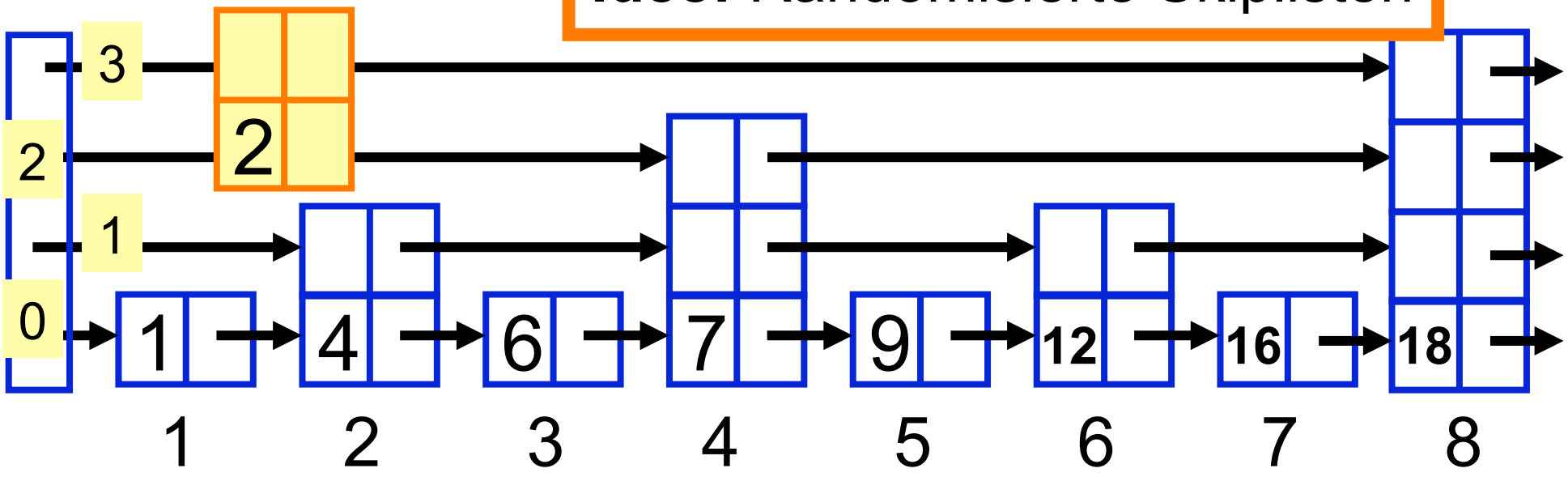
Einfügen und Entfernen in Perfekten Skiplisten

Einfügen von 2:

Hierzu wäre eine komplette Reorganisation mit Laufzeit $\Omega(n)$ notwendig!!!

Niveau i des Zeigers

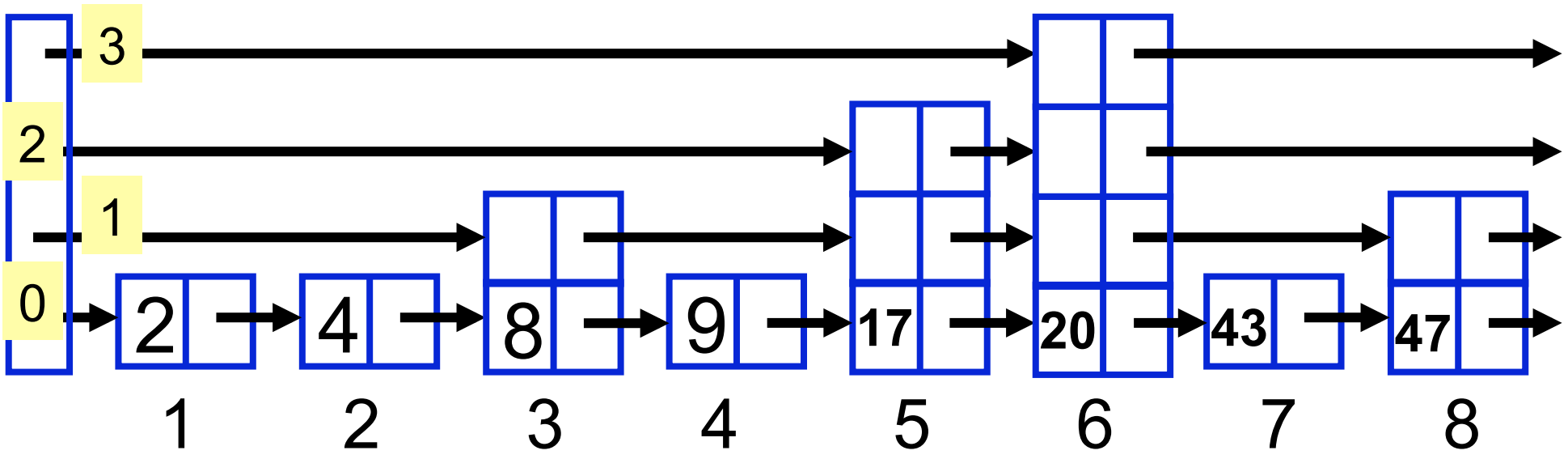
Idee: Randomisierte Skiplisten



Randomisierte Skiplisten

- **Idee:** Die Verteilung der Höhen in der Skipliste entspricht ungefähr der Verteilung in einer perfekten Skipliste.

Wie erreicht man diese Verteilung?



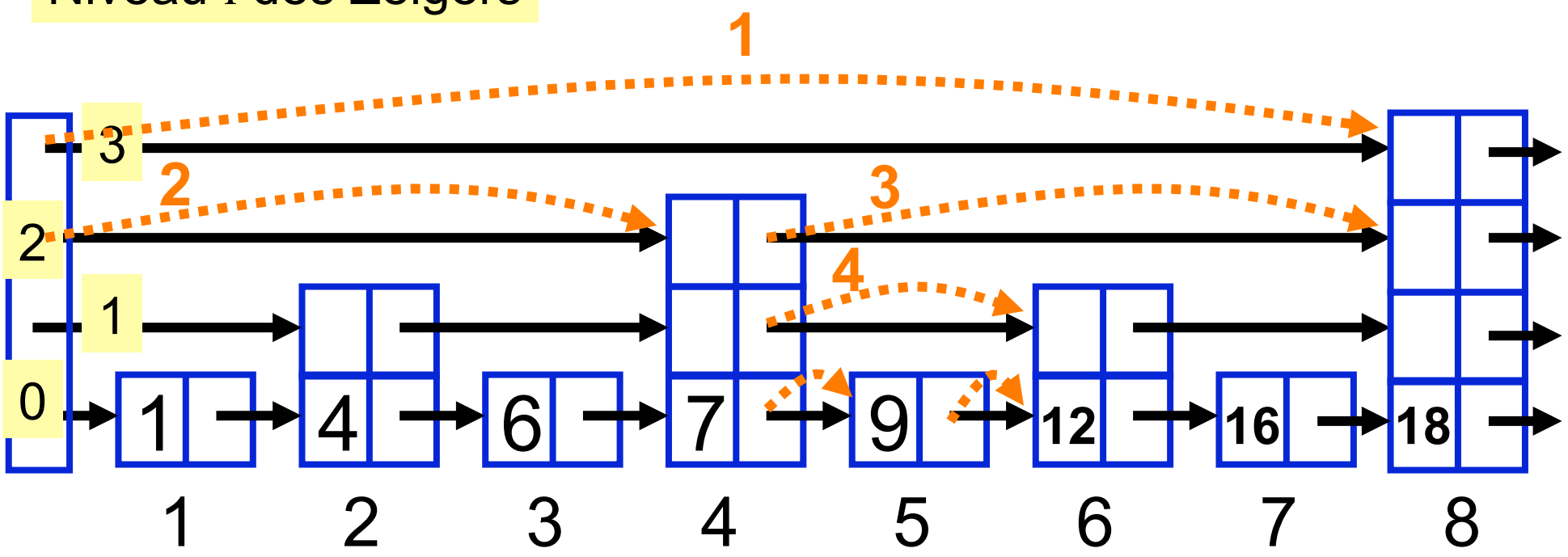
Suche in Randomisierten Skiplisten

Suche nach 12:

identisch mit der Suche in Perfekten Skiplisten

Vergleiche

Niveau i des Zeigers



Anfangselement

Endelement nicht mehr angezeigt

Randomisierte Skiplisten

- Elemente c der Liste besitzen Höhe $h(c)$
- Diese Höhe wird zufällig bestimmt, so dass die Verteilung derjenigen einer perfekten Skipliste entspricht.

- Jedes Element c enthält einen Schlüssel und $h(c)+1$ Zeiger auf nachfolgende Listenelemente
- Anfang und Ende der Liste: je ein Pseudo-Element ohne Daten, gleiche Höhe wie maximales Element in der Liste; Schlüssel des Endelements ist größer als alle anderen

Einfügen in randomisierte Skipliste

(1) Suche die Position, an der eingefügt werden soll

(2) Füge dort einen neuen Container ein, und bestimme dessen Höhe durch folgendes Zufallsexperiment:

(3) Setze die $\text{Höhe} = 0$.

(4) Wiederhole:

(5) Münzwurf: Falls „Zahl“, dann:

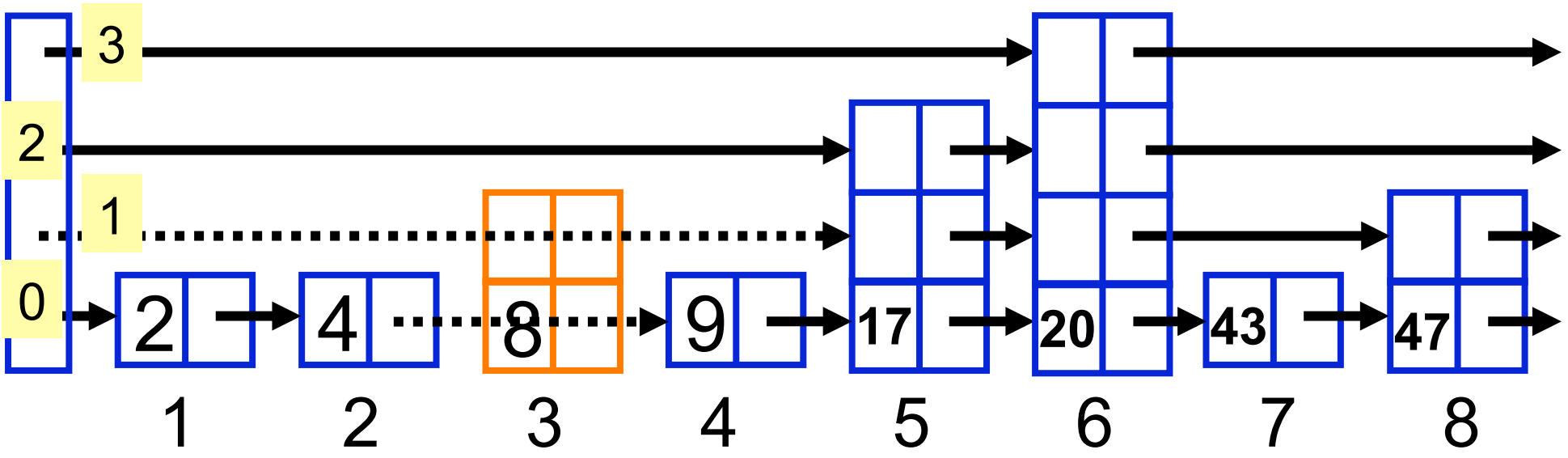
(6) $\text{Höhe}=\text{Höhe}+1$

(7) Solange bis „Kopf“ erscheint.

(8) Aufspalten aller Zeiger die „durch“ neuen Container laufen

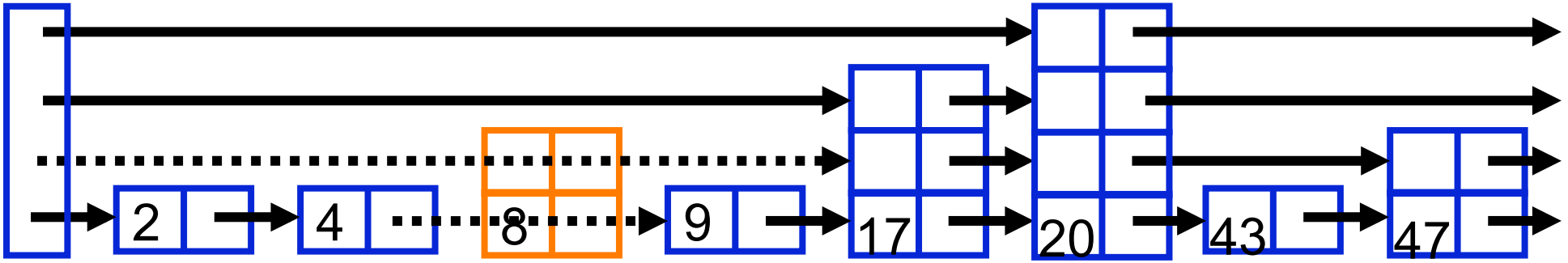
Einfügen in randomisierte Skipliste ff.

- Für das Aufspalten der Zeiger, die durch c laufen, führen wir zusätzliches Array update[i] ein:
- Für jedes i enthält update[i] einen Zeiger auf den am weitesten rechts liegenden Container mit Höhe mindestens i links von c.



INSERT(L,s)

```
(1)  p=L.head
(2)  for i:=L.height to 0 do {
(3)    while p.next[i].key < s do
(4)      p:=p.next[i]
(5)   update[i]=p
(6)  }
(7)  p:=p.next[0]
(8)  if p.key == s then  // Schlüssel s schon da
(9)  else
```



INSERT(L,s) ff.

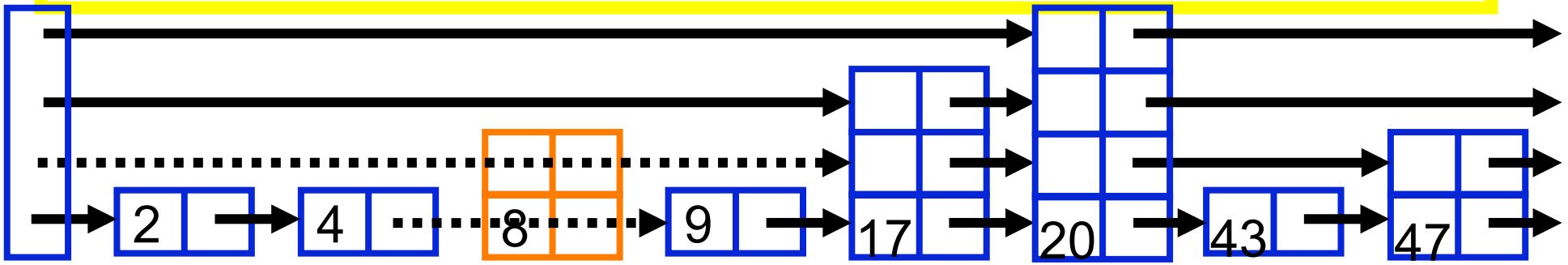
```
(9) else {
(10)  newheight:=randomheight()
(11)  if newheight > L.height then {
(12)      for i:=L.height+1 to newheight do {
(13)         update[i]:=L.head }
(14)      L.height:=newheight
(15) }
(16) p:=newContainer(s,newheight)
(17) for i:=0 to newheight do {
(18)    p.next[i]:=update[i].next[i]
(19)    update[i].next[i]:=p
(20)} }
```

Entfernen aus randomisierter Skipliste

- analog wie Einfügen

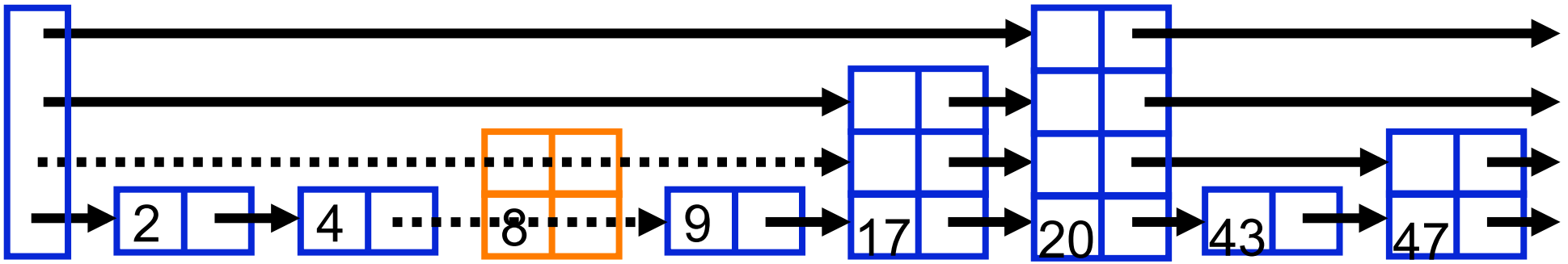
DELETE(L,s)

```
(1)  p=L.head
(2)  for i:=L.height to 0 do {
(3)    while p.next[i].key < s do
(4)      p:=p.next[i]
(5)   update[i]=p
(6)  }
(7)  p:=p.next[0]
(8)  if p.key == s then
```



DELETE(L,s) ff.

```
(8)  if p.key == s then {
(9)     for i:=0 to p.height do
(10)       update[i].next[i]:=p.next[i]
(11)  while L.height ≥ 1 and
                  L.head.next[L.height].key == ∞ do
(8)          L.height := L.height-1
(9)  }
```



Eigenschaften randomisierter Skiplisten

Beobachtung: Skiplisten besitzen kein Gedächtnis, d.h. Elemente, die eingefügt und wieder entfernt wurden haben keinen Einfluss auf das Aussehen der aktuellen Skipliste.

Dies ist bei den anderen Datenstrukturen, wie z.B. den binären Suchbäumen anders.

z.B. insert: 5-7-6-1, del 5 vs. insert: 7-6-5-1 del 5

Analyse randomisierter Skiplisten

Beobachtung: Wir können keine Worst Case Schranke garantieren, denn theoretisch ist es möglich, dass ein Listenelement eine beliebig hohe Höhe erhält.

Dies ist jedoch „unwahrscheinlich“.

Lösung: Wir analysieren die Erwartungswerte und zeigen, dass große Abweichungen von diesen Werten extrem unwahrscheinlich sind.

Analyse randomisierter Skiplisten

Lemma: Die Wahrscheinlichkeit, dass ein Element Höhe
- gleich h erhält, ist $(½)^{h+1}$.
- mindestens h erhält, ist $(½)^h$.

Beweis:

- Die Höhe 0 eines Elements ist nach dem 1. Münzwurf festgelegt. Die Wahrscheinlichkeit hierfür ist ½.
- Nach dem h-ten Münzwurf ist Höhe h-1 festgelegt mit Wahrscheinlichkeit $(½)^h$.
- Für Mindesthöhe h muss h Mal „Zahl“ fallen.

Analyse randomisierter Skiplisten

Lemma: Die erwartete Höhe eines Elements ist 1.

Beweis: Der Erwartungswert für die Höhe ist

$$\sum_{0 \le h < \infty} h \, (½)^{h+1} = 1.$$

Warum summiert sich diese Formel zu 1?

Unendliche geom. Reihe für $|x|<1$: $\sum_{0 \le h < \infty} x^h = 1/(1-x)$.

Differenzieren von L.S. und R.S.: $\sum_{0 \le h < \infty} h \, x^{(h-1)} = 1/(1-x)^2$.

Beide Seiten „mal x^2“: $\sum_{0 \le h < \infty} h \, x^{(h+1)} = x^2/(1-x)^2$.

Unser Fall: $x = (1-x) = ½$

Analyse randomisierter Skiplisten

Lemma: Die Wahrscheinlichkeit, dass eine randomisierte Skipliste eine gewisse Höhe h überschreitet ist $\text{Prob}(H(n) \geq h) \leq \min\{1, n\,(½)^h\}$

Beweis: Die Wahrscheinlichkeit, dass ein Element mindestens Höhe h erhält ist $(½)^h$.

- Sei A_i das Ereignis, dass das i-te Element eine Höhe von mind. h hat. Die Wahrscheinlichkeit, dass eines von n Elementen die Höhe mindestens h hat, ist:

$$\text{Prob}(\bigcup_{1\leq i\leq n} A_i) \leq \sum_{1\leq i\leq n} \text{Prob}(A_i) = n\,(½)^h.$$

Analyse randomisierter Skiplisten

Lemma: Die erwartete Höhe einer randomisierten Skipliste mit n Elementen ist $E(H(n)) \leq \lfloor \log n \rfloor + 2$.

Beweis: Die Erwartungswert für die Höhe ist

$$E(H(n)) = \sum_{0 \leq h < \infty} h \, \mathrm{Prob}(H(n)=h) =$$

$$\mathrm{Prob}(H(n)=1)+\mathrm{Prob}(H(n)=2)+\mathrm{Prob}(H(n)=3)+\ldots$$
$$+\mathrm{Prob}(H(n)=2)+\mathrm{Prob}(H(n)=3)+\ldots$$
$$+\mathrm{Prob}(H(n)=3)+\ldots$$

$$= \sum_{1 \leq h < \infty} \mathrm{Prob}(H(n) \geq h) = \sum_{1 \leq h \leq \lfloor \log n \rfloor + 1} \mathrm{Prob}(H(n) \geq h) + \sum_{\lfloor \log n \rfloor + 2 \leq h < \infty} \mathrm{Prob}(H(n) \geq h)$$

$$\leq \sum_{1 \leq h < \lfloor \log n \rfloor + 1} 1 + \sum_{1 \leq i < \infty} \underbrace{n \, (1/2)^{\lfloor \log n \rfloor + 1 + i}}_{\leq 1} \leq (\lfloor \log n \rfloor + 1) + \sum_{1 \leq i < \infty} (1/2^i)$$

Analyse randomisierter Skiplisten

Lemma: Die erwartete Anzahl der Zeiger in einer randomisierten Skipliste mit n Elementen ist $E(Z(n)) \leq 2n + \lfloor \log n \rfloor + 3$.

Beweis:

- Das i-te Element hat eine durchschnittliche Höhe von 1, besitzt also 2 Zeiger.
- Hinzu kommen die Zeiger vom Anfangselement, das sind $E(H(n)) + 1 \leq \lfloor \log n \rfloor + 3$ viele.

Analyse randomisierter Skiplisten

Lemma: Der erwartete Platzbedarf für eine randomisierte Skipliste mit n Elementen ist $O(n)$.

Beweis: Dies folgt direkt aus dem letzten Lemma.

Analyse randomisierter Skiplisten

Theorem: Die erwartete Rechenzeit für jede der Operationen SEARCH, INSERT und DELETE beträgt $O(\log n)$.

Beweis-Idee: Man zeigt, dass die erwartete Suchpfadlänge logarithmisch beschränkt ist.

- Die Idee ist, den Suchpfad rückwärts zu laufen.
- Jeder Schritt geht mit Wahrscheinlichkeit ½ eine Ebene weiter nach oben oder waagrecht zurück.
- Man zeigt, dass es im Durchschnitt gleich viele Schritte nach oben wie nach links gibt.
- Wir wissen, dass die erwartete Höhe $\lfloor \log n \rfloor + 2$ ist.