Karlsruher Institut für Technologie

Karlsruher Institut für Technologie

Algorithmen I
Tutorium 32

Eine Lehrveranstaltung im SS 2017 (mit Folien von Christopher Hommel)

Daniel Jungkind (ufesa@kit.edu) | 23. Juni 2017

INSTITUT FÜR THEORETISCHE INFORMATIK

Schwarzes Brett

- **Probeklausur** kriegt ihr nächstes Mal (30. Juni)
- Zu **Blatt 6**: Wer bei Aufgabe 3 für fehlendes *„läuft in O(1)“* **–1 P** Abzug bekommen hat: Björn war nochmal gnädig. ☺
  $\Rightarrow$ Blatt bitte **nächstes Mal mitbringen**!
- **Raumverlegung**: Nächstes Tut (**30. Juni**) in Raum **–108** (direkt nebenan).
- **VL-Verlegung**: VL/Übung am **Mi, 28. Juni**, im Fasanengarten-HS!

Zum letzten Tut bzw. Blatt (#7)

GROßES SORRY! ☹

„*InsertionBuildHeap*“ war **nicht wörtlich** gemeint!

$\Rightarrow$ Da wird **nichts eingefügt**, nur das chaotische Array genommen, **wie es ist** und darauf dann mit *siftDown* getauscht

Ich bitte vielmals um Entschuldigung. Musste dafür dementsprechend leider Abzug geben.

Sortierte Folgen

Die eierlegende Wollmilchdatenstruktur

Sortierte Folgen

Heap- und stichfest?

- **Ziel**: eine **dynamische** und **stets sortierte** Datenstruktur
- **Operationen**:
  Einfügen,
  Entfernen,
  Finden des nächstkleineren/größeren Elements
  $\Rightarrow$ *so schnell wie möglich*
- **Idee**: Binärer Heap – sieht sortiert aus, ist es aber **nicht**!

Sortierte Folgen

Einfach sortierter Binärbaum

- **Vorschlag**: Binärbaum mit strengerer Ordnung:
  $\forall v \in V : \text{LeftChild}(v) \leqslant v < \text{RightChild}(v)$
- Intuitiv: Laufzeiten in $O(\log n)$ mittels **binärer Suche**
- **Worst-Case**: Füge aufsteigende Folge ein
  $\Rightarrow$ Lange Kette entsteht („Baum **unbalanciert**“),
  Laufzeiten in $O(n)$ ☹
  $\Rightarrow$ I. A. eher ungeeignet

$\Rightarrow$ Wollen **balancierten** Baum (alle Blätter haben gleiche Tiefe)

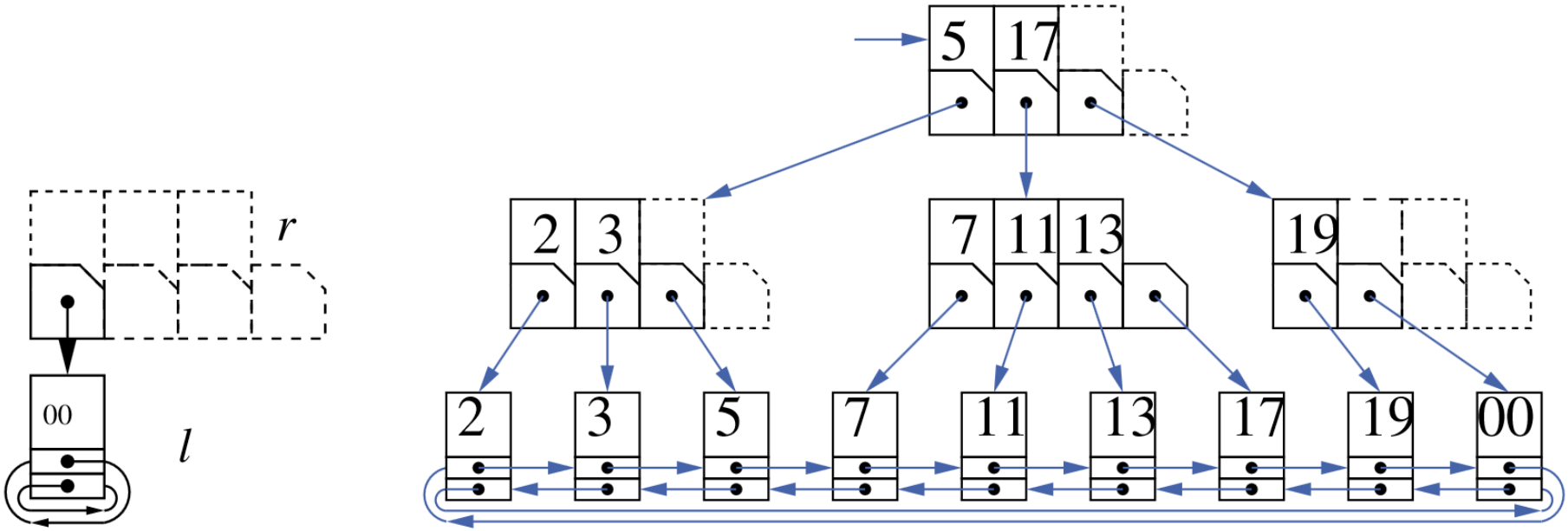
Sortierte Folgen

(a, b)-Bäume

- **Besser**: Baum mit **flexiblem** Knotengrad
  $\Rightarrow$ Anzahl **Kinder** zwischen $a...b$
  Ausnahme: Wurzel kann weniger haben
- Dafür sinnvoll: $a \geqslant 2$ und $b \geqslant 2a - 1$
- Jeder **Knoten** hat ein **Navigations-Array**:
  Einträge mit $(k$: Key, T_k : Subtree$)$:
  T_k führt nur zu Elementen $e \leqslant k$
  Letzter Eintrag: kein Key k, führt zu Elementen $e >$ letztes k
- **Blätter**: Eigentliche Elemente/Daten als **verkettete Liste**
- Zur Vermeidung von Sonderfällen:
  „Dummy-Wert“ ∞ ganz am Ende

Sortierte Folgen – (a, b)-Bäume

Beispiel: (2, 4)-Baum („00“ steht in VL für ∞)



Sortierte Folgen – (a, b)-Bäume

Finden von (nächstgrößeren (-kleineren)) Elementen

- **Geg.**: Wert e
  Ges.: (Nächstgrößeres) Element $z \geqslant e$

$\Rightarrow$ Starte bei **Wurzel**
Suche kleinstes Element im Navigationsarray
$j := \min \{ j \mid e \leqslant j \}$
Blatt-Ebene erreicht? $\Rightarrow$ **return** j
Sonst **Wiederhole** auf Subtree von j oder ganz rechtem Link falls $\nexists j$

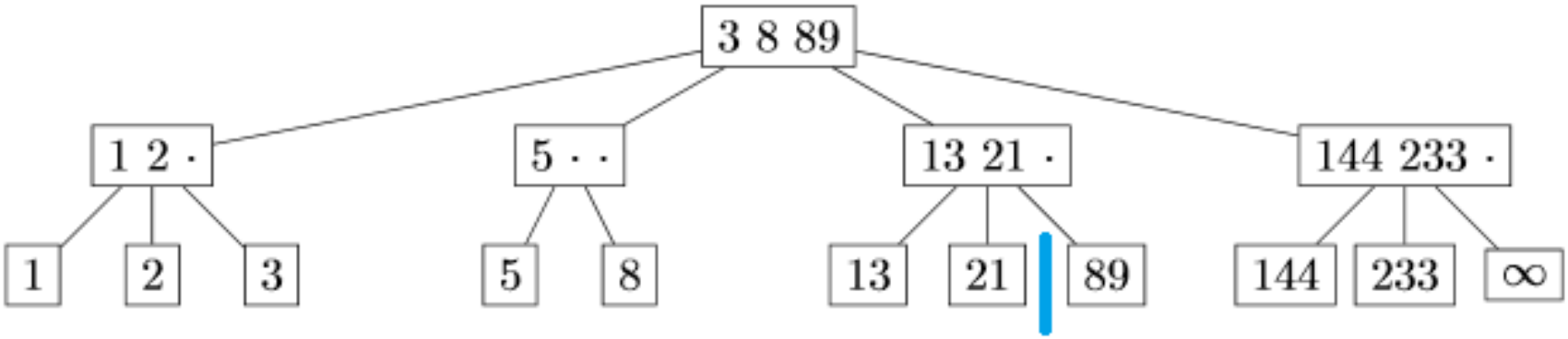
- **Laufzeit** in $O(b \cdot \text{Höhe}) = O(b \cdot \log_a n)$

- Finden von *nächstkleinerem* Element:
  Finde nächstgrößeres;
  Falls $j \neq e$: Nehme Vorgänger von j in verketteter Liste

Sortierte Folgen – (a, b)-Bäume

Einfügen von Elementen

1. Finde Einfügestelle (wie beim Suchen)

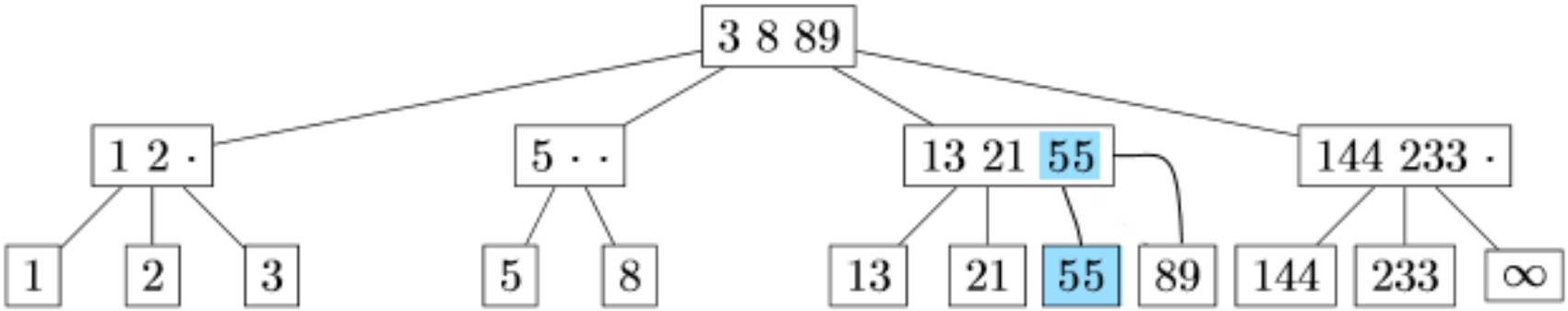


Sortierte Folgen – (a, b)-Bäume

Einfügen von Elementen

1. Finde Einfügestelle (wie beim Suchen)

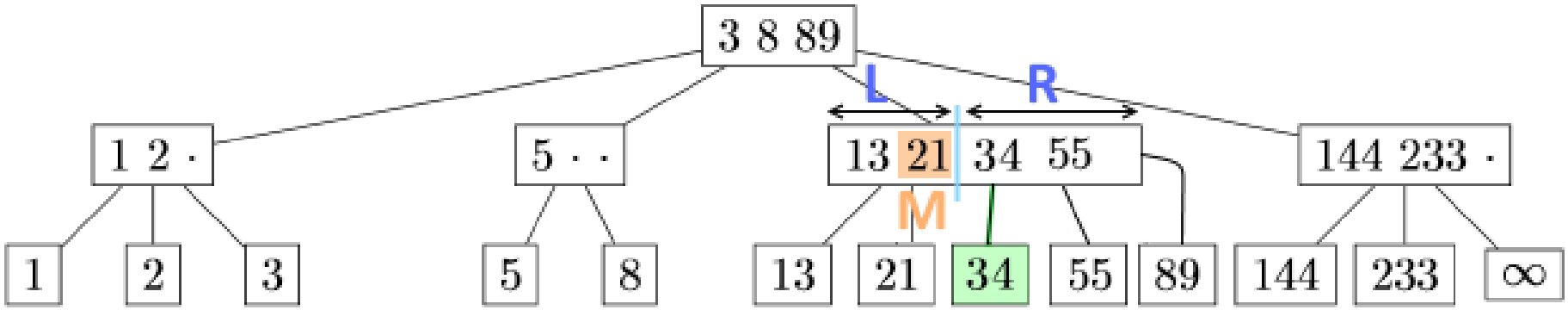
2a. **Fall 1**: Platz im Navigationsarray frei?
⇒ Einfügen, im Nav-Array verlinken, fertig! ☺
(falls neues Maximum: **Verlinkung anpassen**!)



Sortierte Folgen – (a, b)-Bäume

Einfügen von Elementen (Forts.)

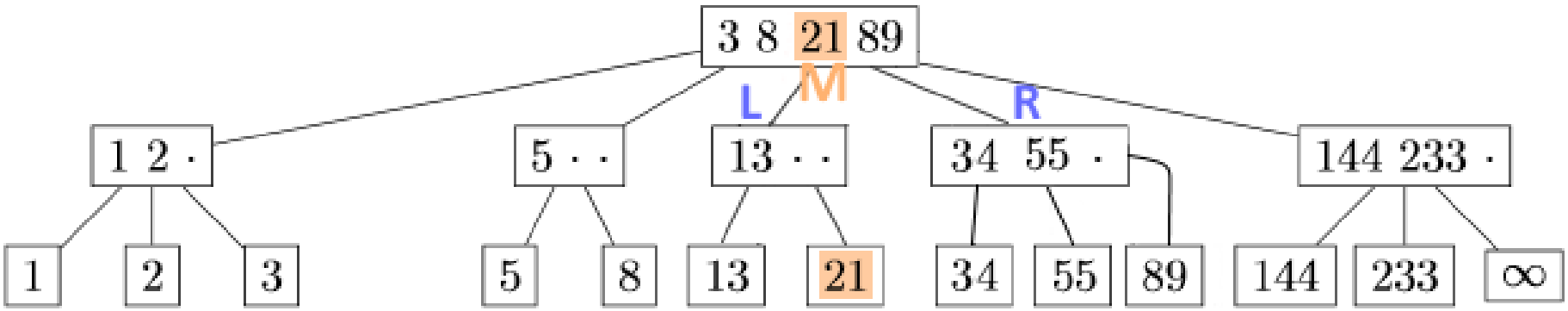
2b. **Fall 2**: Kein Platz im Nav-Array frei? $\Rightarrow$ *„split“*
D.h. Element einfügen, Knoten **halbieren**:
Linker Teil L (enthält **Mittelelement** M), **Rechter** Teil R



Sortierte Folgen – (a, b)-Bäume

Einfügen von Elementen (Forts.)

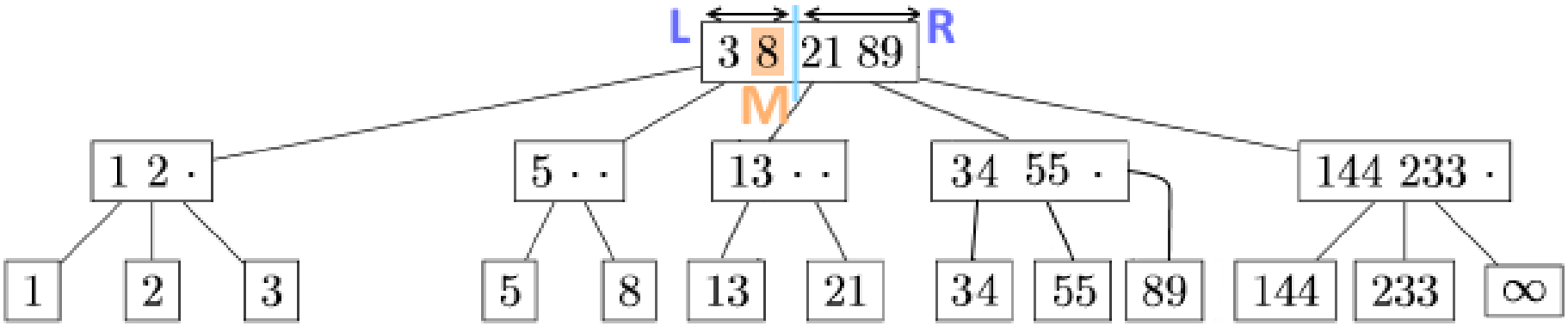
3. Füge *M* in Vorgänger ein, hänge *L* als Subtree daran; *R* hängt schon im Vorgänger



Sortierte Folgen – (a, b)-Bäume

Einfügen von Elementen (Forts.)

3. Füge *M* in Vorgänger ein, hänge *L* als Subtree daran; *R* hängt schon im Vorgänger

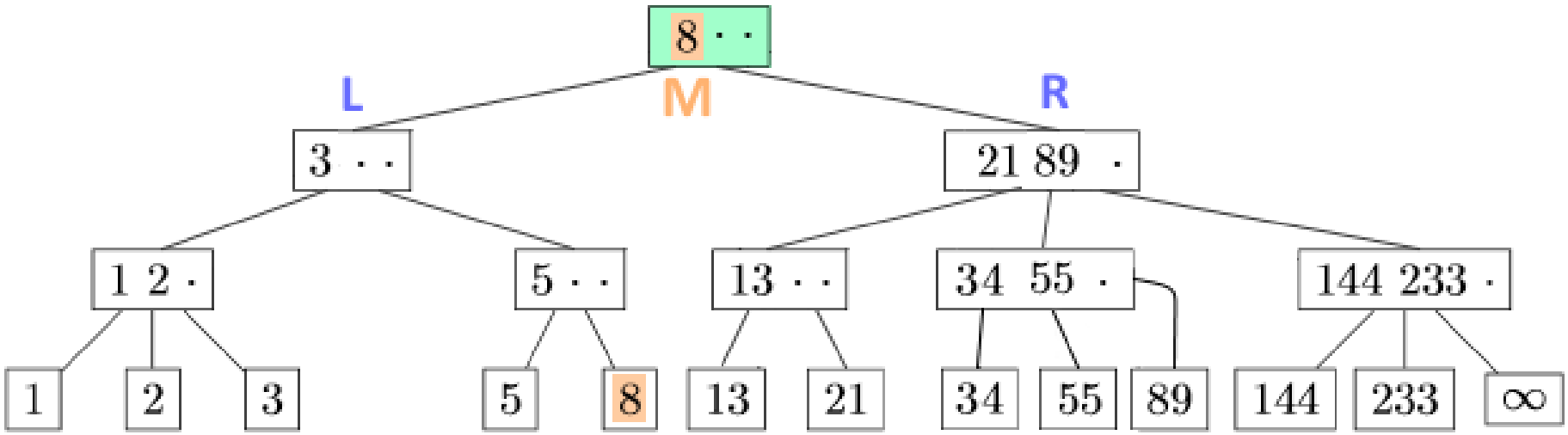


4. Vorgänger **voll**? $\Rightarrow$ **Recurse** from step 2b.

Sortierte Folgen – (a, b)-Bäume

Einfügen von Elementen (Forts.)

3. Füge *M* in Vorgänger ein, hänge *L* als Subtree daran; *R* hängt schon im Vorgänger

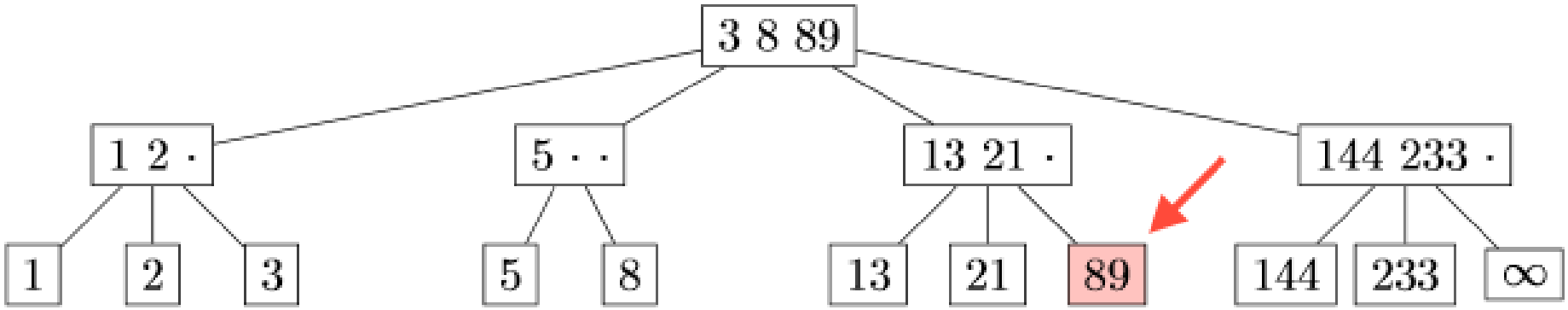


4. Vorgänger **voll**? $\Rightarrow$ **Recurse** from step 2b.
   Endet ggf. mit Anlegen einer *neuen Wurzel*

Sortierte Folgen – (a, b)-Bäume

Entfernen von Elementen

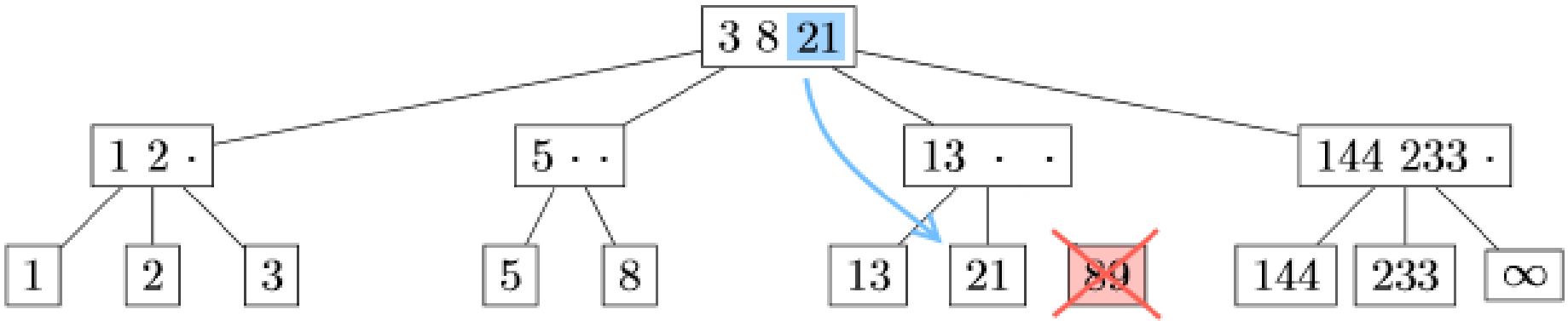
1. Einfach: Finden



Sortierte Folgen – (a, b)-Bäume

Entfernen von Elementen

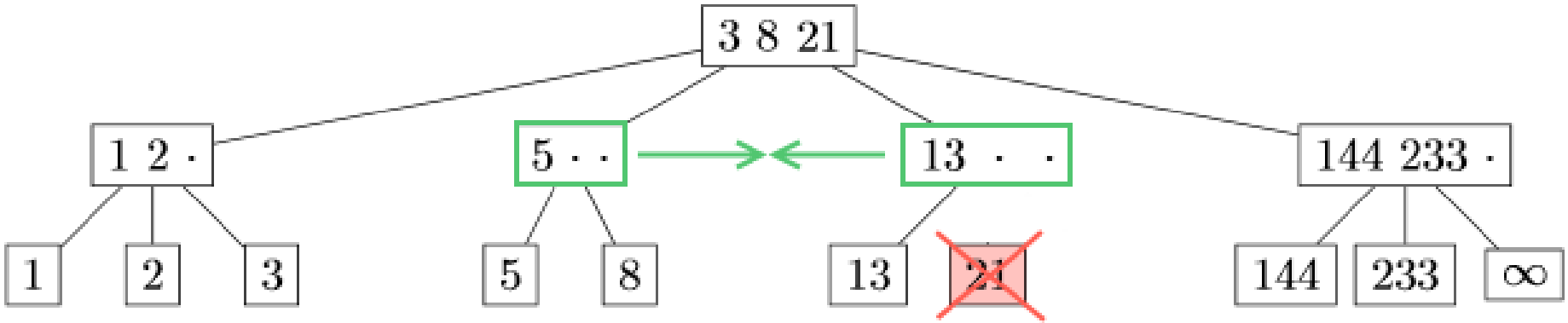
1. Einfach: Finden und Entfernen.
   Knotenmaximum wurde entfernt?
   $\Rightarrow$ **Aktualisiere Verlinkung auf neues Maximum!**



Sortierte Folgen – (a, b)-Bäume

Entfernen von Elementen

2. Knoten jetzt **zu klein?**
   2a. **Fall 1**: ...und ∃ Nachbar, der leer genug?
   ⇒ *„fuse“*: Knoten zusammenfügen



Vorgänger jetzt zu klein? ⇒ Recurse from step 2.

Sortierte Folgen – (a, b)-Bäume

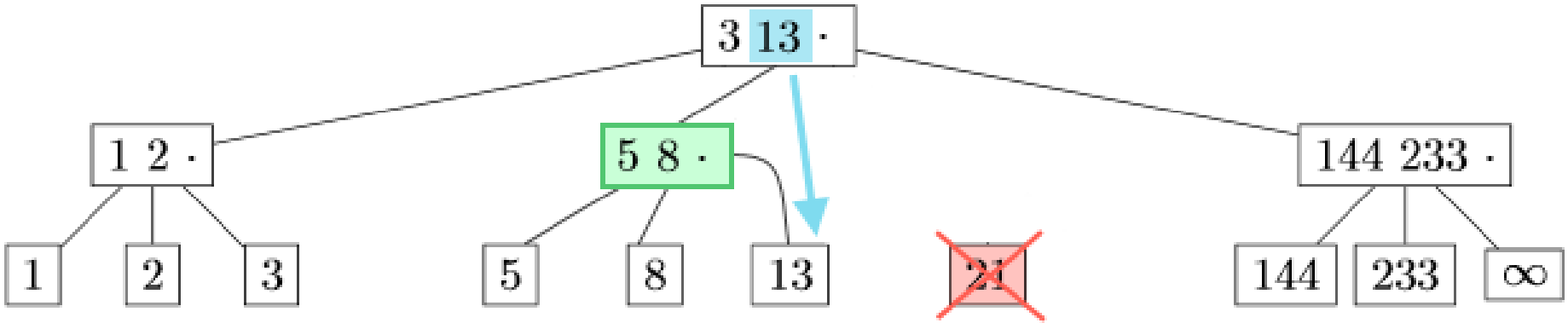
Entfernen von Elementen

2. Knoten jetzt **zu klein?**

   2a. **Fall 1**: ...und ∃ Nachbar, der leer genug?

   ⇒ *„fuse“*: Knoten zusammenfügen

   ...und **Verlinkung anpassen!**

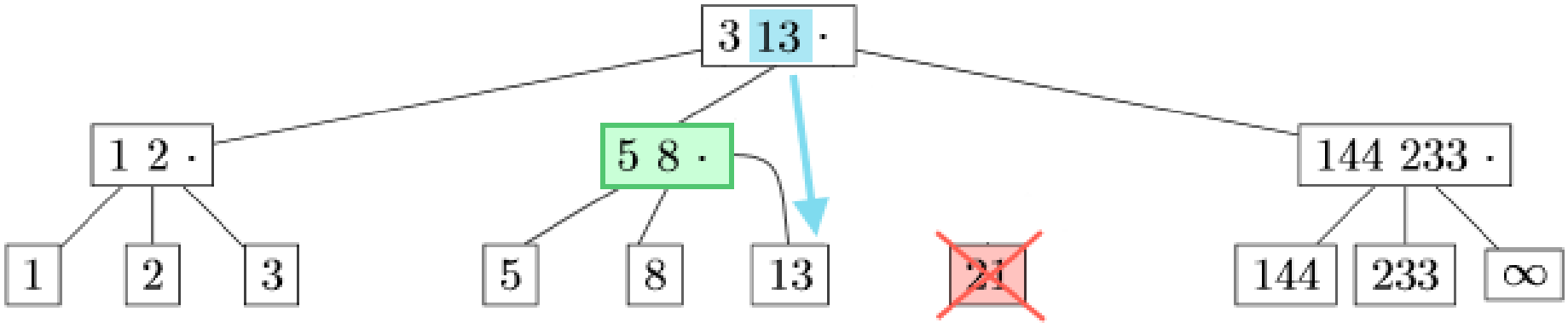


Vorgänger jetzt zu klein? ⇒ Recurse from step 2.

Sortierte Folgen – (a, b)-Bäume

Entfernen von Elementen

2. Knoten jetzt **zu klein?**
   2a. **Fall 1**: ...und ∃ Nachbar, der leer genug?
   ⇒ *„fuse“*: Knoten zusammenfügen
   ...und **Verlinkung anpassen!**



Vorgänger jetzt **zu klein**? ⇒ **Recurse** from step 2.

Sortierte Folgen – (a, b)-Bäume

Entfernen von Elementen

2. Knoten jetzt **zu klein?**

   2b. **Fall 2**: ...und $\exists$ Nachbar, der voll genug?

   $\Rightarrow$ *„balance“*: Klaue Elemente vom fetten Nachbarn

   (von links: maximale, von rechts: minimale Elemente)

   ...und **Verlinkung anpassen!**

Sortierte Folgen – (a, b)-Bäume

Laufzeiten:

$$\left.\begin{array}{l} \textit{locate} \\ \textit{insert} \\ \textit{remove} \end{array}\right\} \text{in } O(b \cdot \text{Höhe}) = O(b \cdot \log_a n) \quad \text{(für konst. } a, b\text{: } O(\log n)).$$