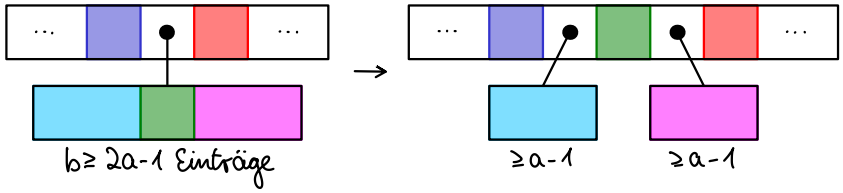
Letztes Mal

(a,b)-Bäume: $a, b \in \mathbb{N}$, $b \geq 2a-1$

- innere Knoten besitzen $\geq a$ (Wurzel ≥ 2 für $n \geq 3$) und $\leq b$ Kinder
- verallgemeinerte BSB-Eigenschaft
- Alle Blätter haben gleiche Tiefe
- Höhe zwischen $O(\log_b n)$ und $O(\log_a n)$
- Suche: $O(b \log_a n)$ Schritte

Einfügen Suche bis zum Blatt, füge ins Blatt ein, wenn der Knoten überläuft (b Einträge), spalte ihn und füge in Elternknoten ein. Wiederhole, bis keine Überläufe mehr.



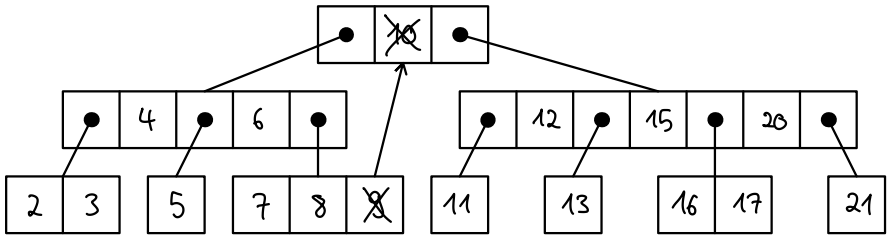
Löschen

- wenn Schlüssel in innerem Knoten:

  Suche Vorgänger/Nachfolger und ersetze den Eintrag dadurch, lösche den Vorgänger/Nachfolger

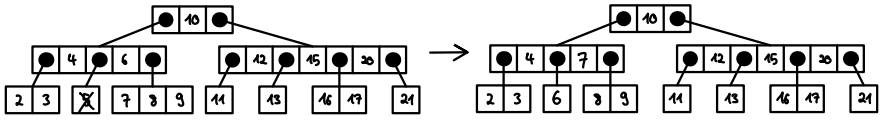
  → Dürfen annehmen, dass wir aus Blatt löschen

  Lösche 10 → Lösche 9 aus Blatt



- **Löschen aus Blatt**
  - $\geq a$ Einträge: kein Problem
  - $= a-1$ Einträge: Versuche, von Geschwister zu stehlen

Lösche 5



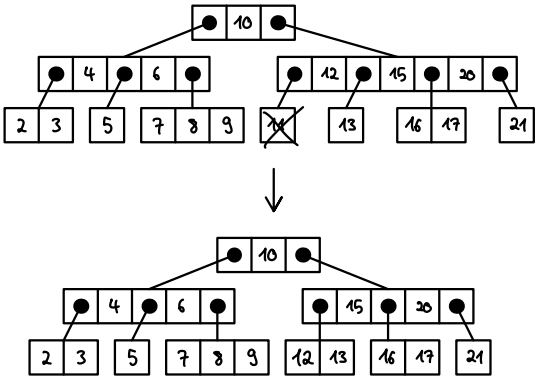
Beide Geschwister haben a-1 Einträge:

verschmelze mit einem Geschwister, hole ein Schlüssel von Elternknoten.

Iteriere, falls nötig.

Wenn beim Iterieren die Wurzel erreicht wird und letzten Eintrag verliert, lösche Wurzel

Lösche 11



Laufzeit $O(b \log_a n)$

Anmerkungen
- (2,3), (2,4)-Bäume statt RS- oder AVL-Bäume
- Externe Datenstruktur: Ein riesiger Baum, gespeichert auf externem Speicher
  Teure Operation: Lesen - Schreiben in die Platte
  → Höhe soll möglichst gering sein
  Speichere so viele Einträge pro Knoten, wie in ein Festplattenblatt passen
  (Verarbeitung im Hauptspeicher ist relativ billig)
  $(a, b \approx 2000)$
  Unerlässlich in DBMS (MySQL, Oracle, ...) und bei Dateisystem (BTRFS)
  BTREE [(a,b)-Baum mit $b=2a$]

Zeichenketten

Alphabet Σ

Zeichenkette ist endliche Folge von Symbolen aus Σ

z.B. $\Sigma = \{a, \dots, z\}$

$\Sigma = \{C, G, T, A\}$

$\Sigma = \{\alpha, \beta, \gamma, \dots\}$

Probleme:
- Wie ähnlich sind zwei Zeichenketten? (z.B. DNA-Vergleich)
- Enthält eine Zeichenkette eine andere? (Textsuche)
- Effiziente Speicherung (Kompression)
- Effizientes Wörterbuch
- ⋮

Effiziente Speicherung

Einfach Weise jedem Zeichen $\lceil \log_2 |\Sigma| \rceil$ viele Bits zu, speichere Konkatenation der Kodierungen

$w = abaaacdaaab \quad \Sigma = \{a,b,c,d\}$

$\downarrow$

00010000001011000001

$a \to 00$
$b \to 01$
$c \to 10$
$d \to 11$

$O(|w| \log |\Sigma|)$ Bits.

Aber Kann ineffizient sein (Häufige Zeichen sollten kurze Codes erhalten)

Code: Funktion $C: \Sigma \to \{0,1\}^*$

Kodierung des Wortes $w = w_1 w_2 \dots w_e$

$C(w) = C(w_1) C(w_2) \dots C(w_e)$

Wollen Code, der eindeutig zu dekodieren ist und der ein Wort in ein möglichst kurzen Bitstring Kodiert

Eindeutige Dekodierbarkeit

$\Sigma = \{a,b,c\}$ $a \to 01$, $b \to 010$, $c \to 10$

was ist 01010?

$01|010$ $a\,b$ — $010|10$ $b\,c$

$01|0|10$ $a\,c\,c$

Problem: $C(a)$ ist ein Präfix von $C(b)$!

Wenn kein Codewort Präfix eines anderen Codeworts ist, heißt der Code **präfixfrei**.

Ein präfixfreier Code ist eindeutig dekodierbar (hier von links nach rechts).

(Aber: nicht jeder eindeutige Code ist Präfixcode)

Ein Präfixcode entspricht einem binären Baum:

- Blätter: Symbole
- Wurzel - Blatt Pfade: Codewörter