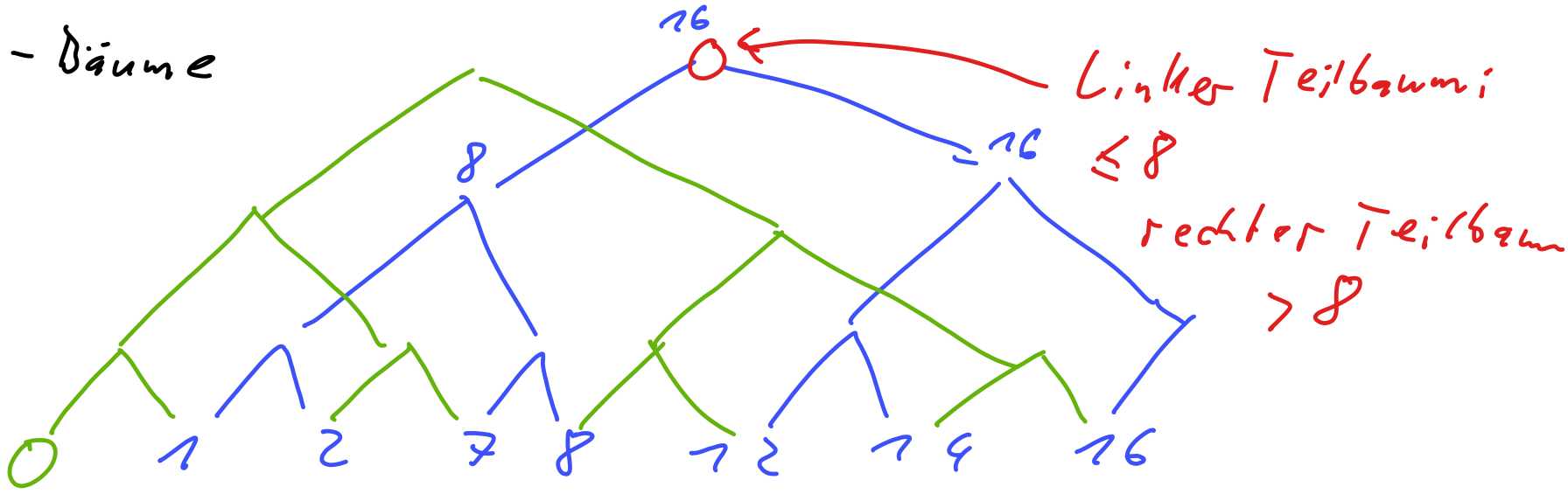
(2,3)-Bäume



Warum balancierte Binärbäume?

- Binärbaum (2 Kinder pro Knoten): einfache Entscheidung für linken oder für rechten Teilbaum
- Balanciert: Nur $\log n$ viele Schritte von Wurzel zu Blatt.

Warum nicht einfach ein balancierter Binärbaum?

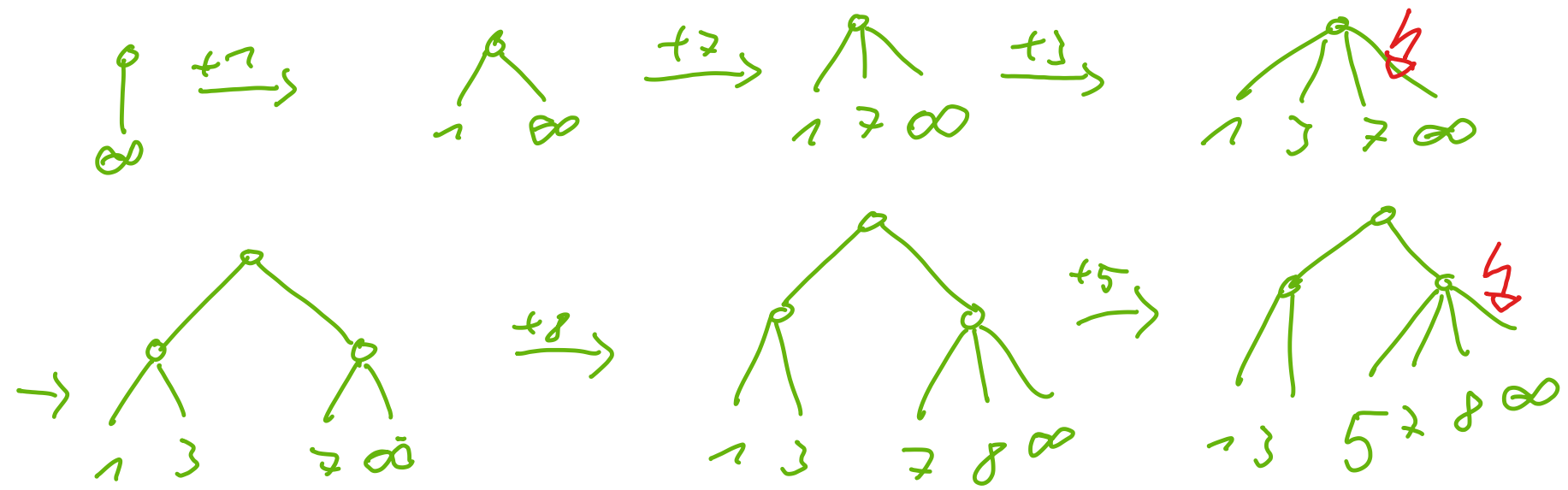
- Wir wollen die gespeicherten Elemente dynamisch ändern
- Forderung balanciert + binär zu strikt.

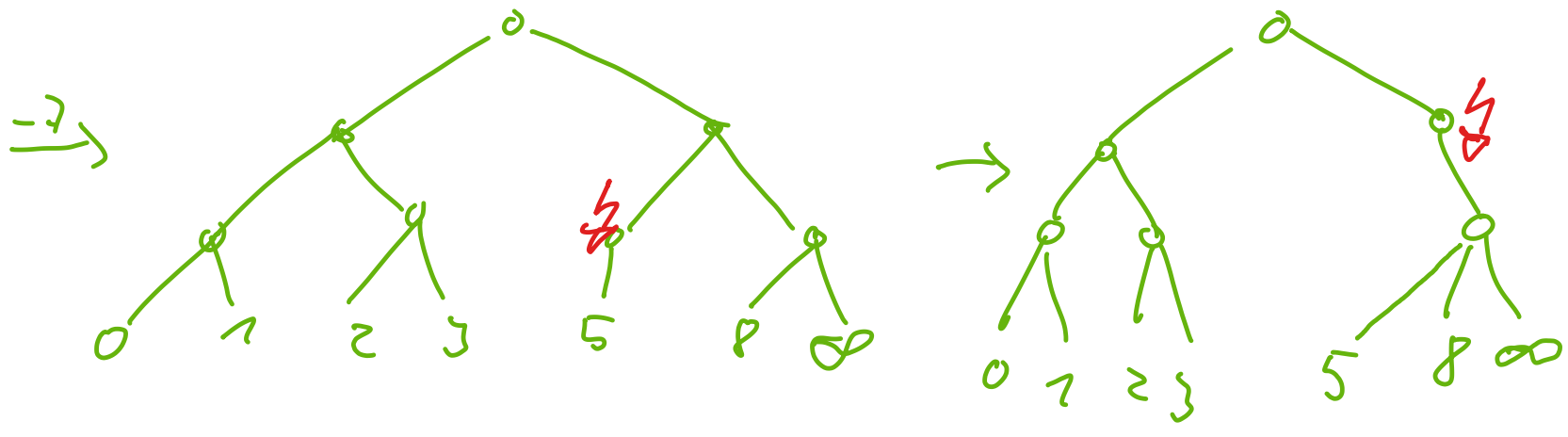
Wie lösen wir das Problem?

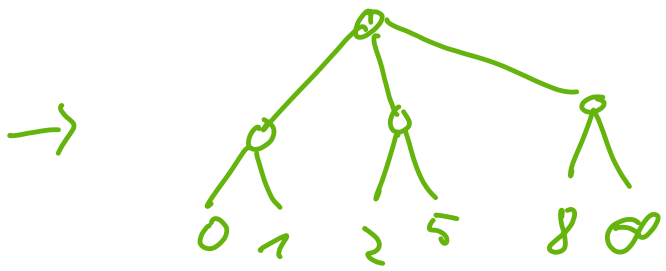
- Erlauben Knoten mit 2 oder 3 Kindern
- aber: maximal balanciert $\rightsquigarrow$ jedes Blatt hat die selbe Tiefe

Wie fügen wir Elemente ein?

- Plan: So faul wie möglich neue Elemente einfügen


+7
+7
+3
+8
+5
+2
+0





Amortisierte Analyse

Was bedeutet „amortisierte Kosten"?

Definition: Betrachte eine DS, sodass jede Folge von K Operationen Gesamtlaufzeit $O(K \cdot f(n))$ hat. Dann sagen wir, dass jede Operation amortisierte Kosten $O(f(n))$ hat.

Warum ist das eine nützliche Definition?

Beobachtung: Bei der Analyse eines Algos so zu tun, als würden pro Operation einer DS nur die amortisierten Kosten anfallen, liefert eine korrekte Schranke für die tatsächlichen Kosten des Algorithmus.

Ist es schlecht, wenn meine DS „nur" amortisiert schnell ist?

Für die Gesamtlaufzeit des Algos komplett irrelevant.

→ Amortisiert $O(1)$ pro Operation ist genauso gut wie tatsächlich $O(1)$ pro Operation.

tatsächliche Kosten:
- insert(k): füge Element hinten in Liste ein $\rightsquigarrow \Theta(1)$
- decimate(): lösche jedes 10te Element $\rightsquigarrow \Theta(n)$
  ($n = \#$Elemente in der Liste)

Beobachtungen:
- Damit decimate teuer ist müssen vorher viele Elemente eingefügt worden sein.
- Je teurer mein decimate, desto mehr Elemente werden gelöscht
- Jedes Element kann nur einmal gelöscht werden.

Plan: Charge Kosten von decimate auf Einfügeoperationen der gelöschten Elemente

Also: betrachte decimate mit n Kostentoken
$\rightsquigarrow$ Es werden $c = \left\lfloor \frac{n}{10} \right\rfloor$ Elemente $e_1, \dots, e_c$ gelöscht.

Charge für jedes $i \in \{1, \dots, c\}$ 10 Kostentoken auf insert(e_i).
$\Rightarrow$ decimate ist $10 \cdot c = 10 \cdot \left\lfloor \frac{n}{10} \right\rfloor \geq \left(\frac{n}{10} - 1\right) \cdot 10 = n - 10$ Token losgeworden. $\Rightarrow \leq 10$ übrig $\rightsquigarrow O(1)$

$\rightsquigarrow$ Aufpassen: jedes Element wird nur einmal gelöscht
$\Rightarrow$ zugehöriges insert bekommt nur einmal 10 Token
$\Rightarrow O(1)$