

Beispiel 8

$$F(z) := \sum_{n \geq 0} 2^n z^n = \frac{1}{1 - 2z}$$

$$G(z) := \sum_{n \geq 0} n z^n = \frac{z}{(1 - z)^2}$$

$$\Rightarrow F(z)G(z) = \frac{z}{(1 - z)^2(1 - 2z)} = \sum_{n \geq 0} \sum_{i=0}^n (n - i) 2^i z^n$$

Partialbruchzerlegung:

$$\begin{aligned}\frac{z}{(1-z)^2(1-2z)} &= \overbrace{\frac{-2}{(1-z)}}^{(1)} + \frac{-z}{(1-z)^2} + \overbrace{\frac{2}{1-2z}}^{(2)} \\ &= \underbrace{\sum_{n \geq 0} 2^{n+1} z^n}_{(2)} - \sum_{n \geq 0} n z^n - 2 \underbrace{\sum_{n \geq 0} z^n}_{(1)} ;\end{aligned}$$

Also:

$$\begin{aligned}\sum_{i=0}^n 2^i (n-i) &= [z^n](FG)(z) \\ &= [z^n] \left(\sum_{n \geq 0} (2^{n+1} - n - 2) z^n \right) \\ &= 2^{n+1} - n - 2 .\end{aligned}$$

7.4 Transformation des Definitions- bzw. Wertebereichs

Beispiel 9

$$f_0 = 1$$

$$f_1 = 2$$

$$f_n = f_{n-1} \cdot f_{n-2} \text{ für } n \geq 2 .$$

Setze

$$g_n := \log f_n .$$

Dann gilt

$$g_n = g_{n-1} + g_{n-2} \text{ für } n \geq 2$$

$$g_1 = \log 2 = 1, \quad g_0 = 0 \text{ (für } \log = \log_2 \text{)}$$

$$g_n = F_n \text{ (} n\text{-te Fibonacci-Zahl)}$$

$$f_n = 2^{F_n}$$

Beispiel 10

$$f_1 = 1$$

$$f_n = 3f_{\frac{n}{2}} + n; \text{ für } n = 2^k ;$$

Setze

$$g_k := f_{2^k} .$$

Beispiel 10

Dann gilt:

$$g_0 = 1$$

$$g_k = 3g_{k-1} + 2^k, \quad k \geq 1$$

Damit ergibt sich:

$$g_k = 3^{k+1} - 2^{k+1}, \quad \text{also}$$

$$\begin{aligned} f_n &= 3 \cdot 3^k - 2 \cdot 2^k \\ &= 3(2^{\log 3})^k - 2 \cdot 2^k \\ &= 3(2^k)^{\log 3} - 2 \cdot 2^k \\ &= 3n^{\log 3} - 2n. \end{aligned}$$

Kapitel II Höhere Datenstrukturen

1. Grundlegende Operationen

Es sei U das Universum von Schlüsseln mit einer (totalen) Ordnung \leq . $S \subseteq U$ sei eine Teilmenge der Schlüssel. Gegeben seien eine Menge von Datensätzen x_1, \dots, x_n , wobei jeder Datensatz x durch einen Schlüssel $k(x) \in S$ gekennzeichnet ist. Jeder Datensatz x besteht aus seinem Schlüssel $k(x)$ und seinem eigentlichen Wert $v(x)$.

<i>IsElement</i> (k, S):	ist $k \in S$, wenn ja, return $v(k)$
<i>Insert</i> (k, S):	$S := S \cup \{k\}$
<i>Delete</i> ($k; S$):	$S := S \setminus \{k\}$
<i>FindMin</i> (S):	return $\min S$
<i>FindMax</i> (S):	return $\max S$
<i>DeleteMin</i> (S):	$S := S \setminus \min S$
<i>ExtractMin</i> (S):	return $\min S, S := S \setminus \min S$
<i>DecreaseKey</i> (k, Δ, S):	ersetze k durch $k - \Delta$
<i>Union</i> (S_1, S_2):	$S_1 := S_1 \cup S_2$
<i>Find</i> (k):	falls $k \in S$, so finde x mit $k = k(x)$
<i>Merge</i> (S_1, S_2):	$S_1 := S_1 \cup S_2$, falls $S_1 \cap S_2 = \emptyset$
<i>Split</i> (S_1, k, S_2):	$S_2 := \{k' \in S_1 \mid k' \geq k\}$ $S_1 = \{k' \in S_1 \mid k' < k\}$
<i>Concatenate</i> (S_1, S_2):	$S_1 := S_1 \cup S_2$; Voraus.: $FindMax(S_1) \leq FindMin(S_2)$

Datenstrukturklasse	mindestens angebotene Funktionen	realisiert in
Wörterbuch (Dictionary)	<i>IsElement()</i> , <i>Insert()</i> , <i>Delete()</i>	Hashtable, Suchbäume
Vorrangwarteschlange (Priority Queue)	<i>FindMin()</i> , <i>Insert()</i> , <i>Delete()</i> , <i>[IsElement()]</i>	balancierte, leftist Bäume, Binomial Queues
Mergeable heaps	<i>FindMin()</i> , <i>Insert()</i> , <i>Delete()</i> , <i>Merge()</i>	2-3-Bäume, Binomial Queues, Leftist-Bäume
Concatenable queues	<i>FindMin()</i> , <i>Insert()</i> , <i>Delete()</i> , <i>Concatenate()</i>	2-3-Bäume

Definition 11

- Bei einem **externen Suchbaum** werden die Schlüssel nur an den Blättern gespeichert, die inneren Knoten enthalten Verwaltungsinformationen.
- Bei **internen Suchbäumen** liegen die Schlüssel an den internen Knoten, die Blätter sind leere Knoten. Zeiger auf Blätter sind daher NIL-Pointer und werden gewöhnlich nicht angegeben.

2. (Balancierte) Suchbäume

Wir betrachten zunächst zwei Familien höhenbalancierter externer Suchbäume.

2.1 (a,b)-Bäume

Definition 12

Ein (a, b) -Baum ist ein externer Suchbaum mit folgenden Eigenschaften:

- 1 alle Blätter haben die gleiche Tiefe;
- 2 die Anzahl der Kinder eines jeden internen Knoten ist $\leq b$ und $\geq a$ (für die Wurzel: ≥ 2);
- 3 es gilt $b \geq 2a - 1$.
- 4 (a, b) -Bäume mit $b = 2a - 1$ heißen auch **B-Bäume**.

Bei jedem internen Knoten v mit $d = d(v)$ Kindern werden $d - 1$ Schlüssel k_1, \dots, k_{d-1} gespeichert, so dass (mit $k_0 := -\infty$, $k_d = +\infty$) gilt:

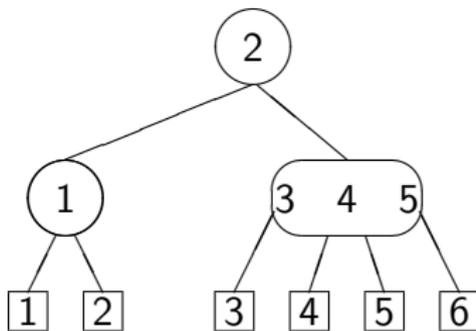
$$k_{i-1} < \text{alle Schlüssel im } i\text{-ten Unterbaum von } v \leq k_i .$$

Also z.B.: $k_i :=$ maximaler Schlüssel im i -ten Unterbaum von v .

Weiter gilt (wie in jedem Suchbaum)

$$\max\{\text{Schlüssel im } i\text{-ten UB}\} \leq \min\{\text{Schlüssel im } i + 1\text{-ten UB}\} .$$

Beispiel 13



(2, 4)-Baum

Lemma 14

Sei T ein (a, b) -Baum mit n Blättern und der Höhe h (Höhe definiert als Anzahl der Kanten auf einem Pfad von der Wurzel zu einem Blatt). Dann gilt:

- 1 $2a^{h-1} \leq n \leq b^h$;
- 2 $\log_b n = \frac{\log n}{\log b} \leq h \leq 1 + \log_a \frac{n}{2}$.

Beweis:

- 1 n ist maximal für vollständigen Baum mit Verzweigungsgrad b und minimal für einen Baum, wo die Wurzel Grad 2 und alle anderen internen Knoten Grad a haben.
- 2 folgt durch Umformung aus (1).



Operationen:

1. *IsElement*(k, S)

$v :=$ Wurzel von T ;

while $v \neq$ Blatt **do**

$i = \min\{s; 1 \leq s \leq (\#Kinder \text{ von } v) + 1 \text{ und } k \leq k_s\}$;

$v := i$ -tes Kind von v ;

if $k = k(v)$ **then return** $v(k)$ **else return false fi** ;

Zeitbedarf: $\theta(h) = \theta(\log n)$

2. $Insert(k, S)$

Führe $IsElement(k, S)$ aus; \rightsquigarrow Blatt w ;

if $k \neq k(w)$ **then**

co falls $k < \max S$, enthält w den kleinsten
 Schlüssel $\in S$, der $> k$ ist **oc**

 füge k **if** $k < \max S$ **then** links **else** rechts **fi** von w
 ein;

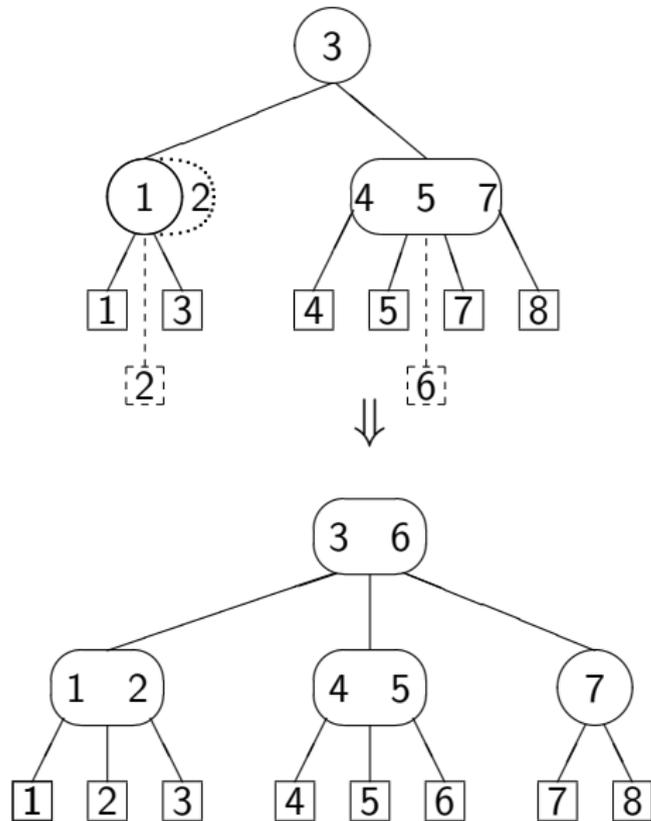
$v :=$ Vater von w ;

if v hat nun mehr als b Kinder **then**

 Rebalancierung(v)

fi

fi



Rebalancierung(v):

- spalte v in zwei Knoten v_1 und v_2 , v_1 übernimmt die linke „Hälfte“ der Kinder von v , v_2 den Rest; da

$$a \leq \lfloor \frac{b+1}{2} \rfloor \leq \lceil \frac{b+1}{2} \rceil \leq b \text{ und } b \geq 2a - 1,$$

erfüllen v_1 und v_2 die Gradbedingung;

- **Vereinfachung:** Falls ein unmittelbarer Nachbar v' von v Grad $< b$ hat, übernimmt v' das äußerste linke/rechte Kind von v .
- Falls nun der Vater u von v mehr als b Kinder hat, führe Rebalancierung(u) aus;
- Falls u die Wurzel des Baums ist, kann dadurch eine neue Wurzel geschaffen werden und die Höhe des (a, b) -Baums wachsen.

Kosten $\mathcal{O}(\log n)$