

6.4 Splay-Trees als Suchbäume

In diesem Abschnitt untersuchen wir **Splay-Trees**, eine Datenstruktur, die den MFR-Ansatz auf Bäume überträgt:

Wird auf ein Element durch eine Operation zugegriffen, so wird dieses im Splay-Tree in geeigneter Weise zur Wurzel befördert, um, sollten weitere Zugriffe auf dieses Element folgen, diese zu beschleunigen.

Wir untersuchen hier Splay-Trees als **Suchbäume**, d.h. die Schlüssel stammen aus einem total geordneten Universum, und innerhalb des Splay-Trees soll die Invariante

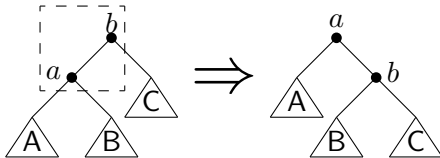
$$\text{“Knoten im lUb} \leq k(x) \leq \text{Knoten im rUb“}$$

gelten.

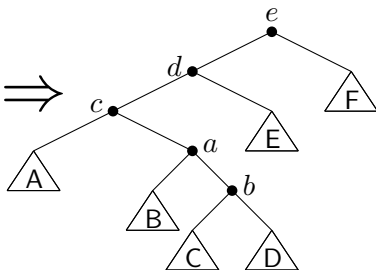
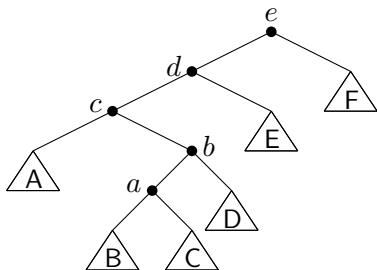
Ansonsten ist ein Splay-Tree ein **interner binärer Suchbaum**.

Wir benutzen **Rotationen**, um unter Beibehaltung der Invariante einen Schlüssel näher zur Wurzel zu bewegen, und zwar Einfach- und Doppelrotationen.

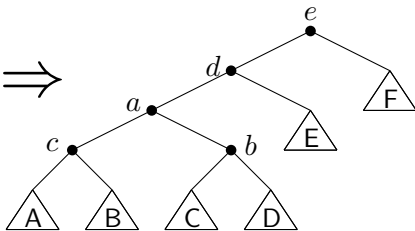
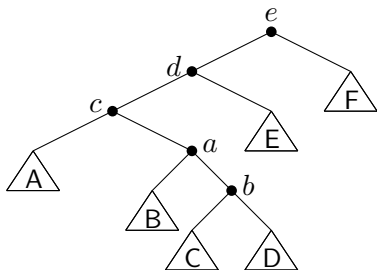
Beispiel 55



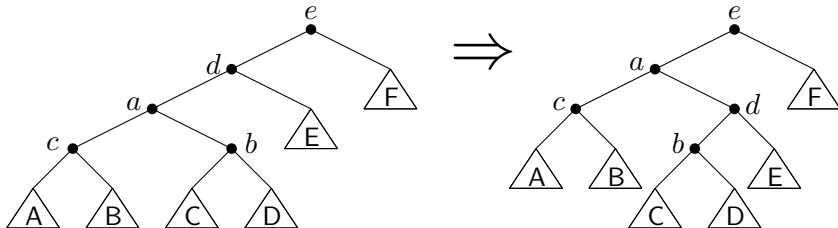
Beispiel 55



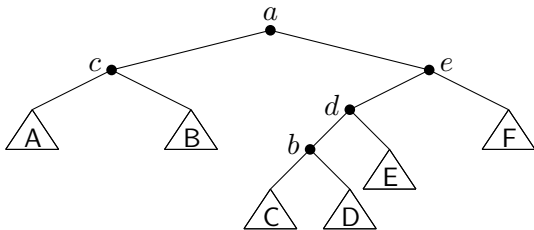
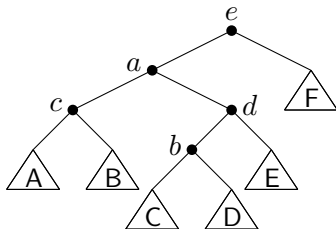
Beispiel 55



Beispiel 55



Beispiel 55

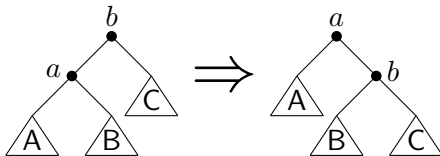


6.4.1 Die Splaying-Operation

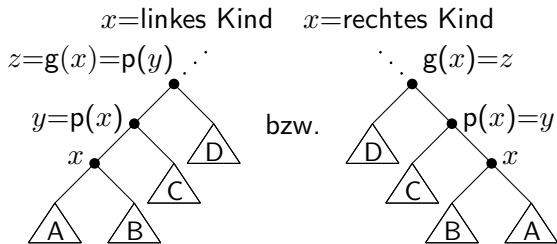
Ein Knoten, auf den zugegriffen wird ($\text{Splay}(x, T)$), wird durch eine Folge von einfachen und doppelten Rotationen an die Wurzel bewegt.

Wir unterscheiden die folgenden Fälle:

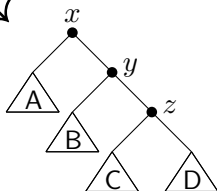
- ① (zig): x ist Kind der Wurzel von T : einfache Rotation



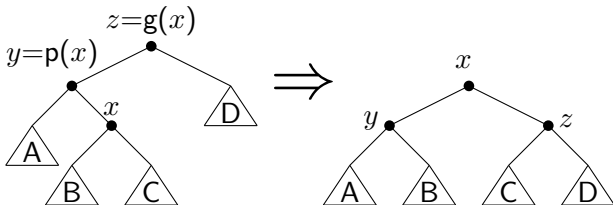
- ② (**zig-zig**): x hat Großvater $g(x)$ und Vater $p(x)$; x und $p(x)$ sind jeweils linke (bzw. rechte) Kinder ihres Vaters.



Doppelrotation



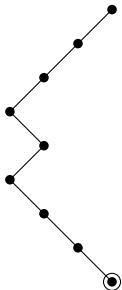
- ③ (**zig-zag**): x hat Großvater $g(x)$ und Vater $p(x)$, x ist linkes (rechtes) Kind von $p(x)$, $p(x)$ ist rechtes (linkes) Kind von $g(x)$.



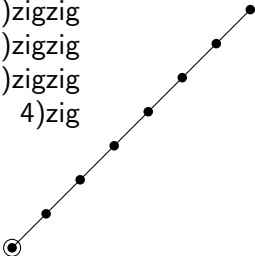
Beispiel 56

Führe die Splaying-Operation jeweils mit dem eingekreisten Element durch:

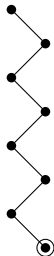
- 1) zigzig
- 2) zigzag
- 3) zigzag
- 4) zigzig



- 1) zigzig
- 2) zigzig
- 3) zigzig
- 4) zig



- 1) zigzag
- 2) zigzag
- 3) zigzag
- 4) zig



6.4.2 Amortisierte Kostenanalyse der Splay-Operation

Jeder Knoten habe ein Gewicht $w(x) > 0$. Das Gewicht $tw(x)$ des Unterbaums mit Wurzel x ist die Summe der Gewichte aller Knoten im Unterbaum. Setze

$$\text{Rang } r(x) = \log(tw(x))$$

$$\text{Potenzial eines Baumes } T = \sum_{x \in T} r(x)$$

Lemma 57

Sei T ein Splay-Tree mit Wurzel u , x ein Knoten in T . Die amortisierten Kosten für $\text{Splay}(x, T)$ sind

$$\leq 1 + 3(r(u) - r(x)) = \mathcal{O} \left(\log \frac{tw(u)}{tw(x)} \right).$$

Beweis:

Induktion über die Folge von (Doppel)Rotationen:

Berechne r und r' , tw und tw' , die Rang- bzw. Gewichtsfunktion vor und nach einem Rotationsschritt. Wir zeigen, dass die amortisierten Kosten im

$$\text{Fall 1 (zig)} \leq 1 + 3(r'(x) - r(x))$$

und in den

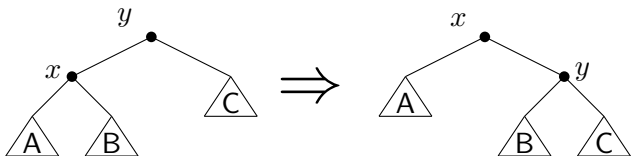
$$\text{Fällen 2 und 3 (zig-zig bzw. zig-zag)} \leq 3(r'(x) - r(x))$$

sind.

y sei der Vater von x , z der Großvater (falls er existiert).

Beweis (Forts.):

① Fall:



Amortisierte Kosten:

$$\leq 1 + r'(x) + r'(y) - r(x) - r(y)$$

$$\leq 1 + r'(x) - r(x),$$

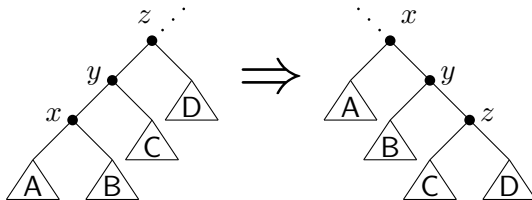
$$\leq 1 + 3(r'(x) - r(x)),$$

$$\text{da } r'(y) \leq r(y)$$

$$\text{da } r'(x) \geq r(x)$$

Beweis (Forts.):

② Fall:



Amortisierte Kosten:

$$\begin{aligned} &\leq 2 + r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z) \\ &= 2 + r'(y) + r'(z) - r(x) - r(y), \quad \text{da } r'(x) = r(z) \\ &\leq 2 + r'(x) + r'(z) - 2r(x), \quad \text{da } r'(x) \geq r'(y) \text{ und } r(y) \geq r(x) \end{aligned}$$

Beweis (Forts.):

Es gilt, dass

$$2 + r'(x) + r'(z) - 2r(x) \leq 3(r'(x) - r(x)),$$

d.h.

$$2r'(x) - r(x) - r'(z) \geq 2.$$

Betrachte dazu die Funktion

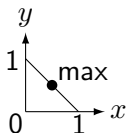
$$f(x, y) = \log x + \log y$$

in dem Bereich

$$x, y > 0, \quad x + y \leq 1.$$

Beweis (Forts.):

Behauptung: $f(x, y)$ nimmt sein eindeutiges Maximum im Punkt $(\frac{1}{2}, \frac{1}{2})$ an.



Beweis der Behauptung: Da die \log -Funktion streng monoton wachsend ist, kann sich das Maximum der Funktion $f(x, y) = \log x + \log y$ nur auf dem Geradensegment $x + y = 1$, $x, y > 0$ befinden. Dadurch erhalten wir ein neues Maximierungsproblem für die Funktion $g(x) = \log(x) + \log(1 - x)$ auf diesem Geradensegment. Da $g(x)$ an den Rändern gegen $-\infty$ strebt, muss es sich um ein lokales Maximum handeln.

Beweis (Forts.):

Die einzige Nullstelle der Ableitung

$$g'(x) = \frac{1}{\ln a} \left(\frac{1}{x} - \frac{1}{1-x} \right),$$

wenn $\log = \log_a$, ist $x = 1/2$ (unabhängig von a).

Weiter ist

$$g''(x) = -\frac{1}{\ln a} \left(\frac{1}{x^2} + \frac{1}{(1-x)^2} \right).$$

Da $g''(0.5) < 0$ ist, nimmt $g(x)$ sein globales Maximum in $x = 0.5$ an. Insgesamt folgt, dass die Funktion $f(x, y) = \log x + \log y$ ihr globales Maximum im Bereich $x, y > 0$, $x + y \leq 1$ an der Stelle $(0.5, 0.5)$ annimmt.

Damit ist die obige Behauptung gezeigt. Wir fahren mit dem Beweis der Abschätzung im Lemma fort.

Beweis (Forts.):

Damit gilt im 2. Fall:

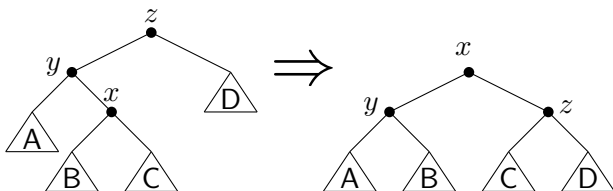
$$r(x) + r'(z) - 2r'(x) = \log\left(\frac{tw(x)}{tw'(x)}\right) + \log\left(\frac{tw'(z)}{tw'(x)}\right) \leq -2,$$

da

$$tw(x) + tw'(z) \leq tw'(x).$$

Beweis (Forts.):

③ Fall:



Amortisierte Kosten:

$$\begin{aligned} &\leq 2 + r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z) \\ &\leq 2 + r'(y) + r'(z) - 2r(x), \quad \text{da } r'(x) = r(z) \text{ und } r(x) \leq r(y) \\ &\leq 2(r'(x) - r(x)), \quad \text{da } 2r'(x) - r'(y) - r'(z) \geq 2. \end{aligned}$$

(Letzteres folgt aus der Behauptung über $f(x, y)$ wie im 2. Fall.)

Beweis (Forts.):

Die Gesamtbehauptung des Lemmas folgt dann durch Aufaddieren der amortisierten Kosten für die einzelnen Schritte (Teleskop-Summe). □

Sei T ein Splay-Tree mit n Knoten x_1, \dots, x_n . Falls sich die Gewichte der Knoten nicht ändern, ist die Verringerung des Potenzials durch eine (beliebige) Folge von Splay-Operationen beschränkt durch

$$\sum_{i=1}^n (\log W - \log w_i) = \sum_{i=1}^n \log \frac{W}{w_i},$$

wobei

$$W := \sum_{i=1}^n w_i,$$

$w_i =$ Gewicht von x_i ,

da das Gewicht des Unterbaums mit Wurzel x_i immer mindestens w_i und höchstens W ist.

Satz 58

Die gesamten Kosten für die m Zugriffe im Splay-Tree sind

$$O((m + n) \log n + m).$$

Beweis:

Wähle $w_i = \frac{1}{n}$ für alle Knoten. Dann sind die amortisierten Kosten für einen Zugriff $\leq 1 + 3 \log n$, da $W = \sum_{i=1}^n w_i = 1$.

Die Verringerung des Potenzials ist

$$\leq \sum_{i=1}^n \log \frac{W}{w_i} = \sum_{i=1}^n \log n = n \log n.$$

Damit sind die reellen Kosten $\leq m(1 + 3 \log n) + n \log n$. □

Satz 59

Sei $q(i)$ die Anzahl der Zugriffe auf das Element x_i (in einer Folge von m Zugriffen). Falls auf jedes Element zugegriffen wird (also $q(i) \geq 1$ für alle i), dann sind die (reellen) Gesamtkosten für die Zugriffe

$$\mathcal{O} \left(m + \sum_{i=1}^n q(i) \cdot \log \left(\frac{m}{q(i)} \right) \right).$$

Beweis:

Setze das Gewicht des i -ten Knotens gleich $\frac{q(i)}{m}$.

$$\Rightarrow W = \sum_{i=1}^n \frac{q(i)}{m} = 1.$$

Der Rest folgt wie zuvor. □

Satz 60

Betrachte eine Folge von Zugriffsoperationen auf eine n -elementige Menge. Sei t die dafür nötige Anzahl von Vergleichen in einem optimalen *statischen* binären Suchbaum. Dann sind die Kosten in einem (anfangs beliebigen) Splay-Tree für die Operationenfolge $\mathcal{O}(t + n^2)$.

Beweis:

Sei U die Menge der Schlüssel, d die Tiefe eines (fest gewählten) optimalen statischen binären Suchbaumes. Für $x \in U$ sei weiter $d(x)$ die Tiefe von x in diesem Suchbaum. Setze

$$tw(x) := 3^{d-d(x)}.$$

Sei T ein beliebiger Splay-Tree für U , $|U| =: n$.

$$\begin{aligned} bal(T) &\leq \sum_{x \in U} r(x) = \sum_{x \in U} \log(3^{d-d(x)}) = \sum_{x \in U} (\log 3)(d - d(x)) = \\ &= (\log 3) \sum_{x \in U} (d - d(x)) = \mathcal{O}(n^2); \end{aligned}$$

$$\sum_{x \in U} tw(x) = \sum_{x \in U} 3^{d-d(x)} \leq \sum_{i=0}^d 2^i 3^{d-i} \leq 3^d \frac{1}{1 - \frac{2}{3}} = 3^{d+1}$$

$$\Rightarrow \log \frac{tw(T)}{tw(x)} \leq \log \frac{3^{d+1}}{3^{d-d(x)}} = \log 3^{d(x)+1}.$$

Beweis (Forts.):

Damit ergibt sich für die amortisierten Kosten von $\text{Splay}(x, T)$

$$\mathcal{O}\left(\log \frac{tw(T)}{tw(x)}\right) = \mathcal{O}(d(x) + 1).$$

Die amortisierten Kosten sind damit

$\leq c \cdot$ Zugriffskosten ($\#$ Vergleiche) im optimalen Suchbaum

(wo sie $d(x) + 1$ sind).

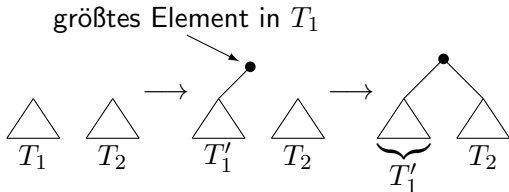
Die gesamten amortisierten Kosten für die Zugriffsfolge sind daher $\leq c \cdot t$.

Die reellen Kosten ergeben sich zu \leq amort. Kosten + Verringerung des Potenzials, also $\mathcal{O}(t + n^2)$. □

6.4.3 Wörterbuchoperationen in Splay-Trees

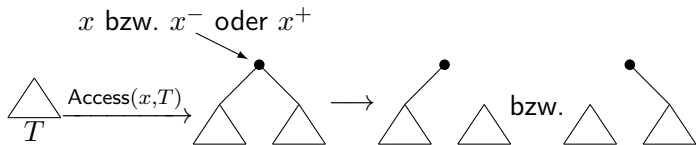
Alle folgenden Operationen werden mit Hilfe von Splay implementiert.

- $Access(x, T)$: \surd (siehe oben)
- $Join(T_1, T_2)$:

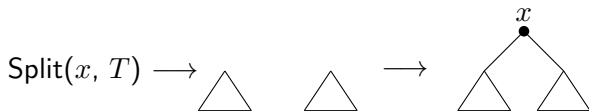


Beachte: Falls $x \in T_1$, $y \in T_2$, dann $x < y$.

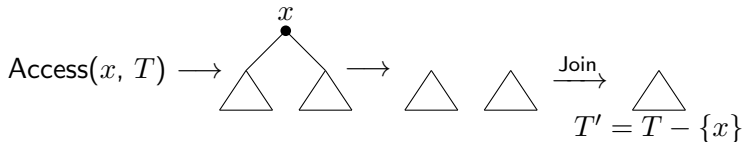
- $Split(x, T)$:



- $Insert(x, T)$:



- $Delete(x, T)$:



Sei $x \in U$, T ein Splay-Tree. Dann bezeichnen wir mit x^- (bzw. x^+) den Vorgänger (bzw. den Nachfolger) von x in U (falls diese existieren). U ist ja total geordnet. Falls x^- bzw. x^+ undefiniert sind, so setzen wir $w(x^-) = \infty$ bzw. $w(x^+) = \infty$.

Weiterhin sei W das Gesamtgewicht aller an einer Wörterbuch-Operation beteiligten Knoten.

Satz 61

Für die amortisierten Kosten der Wörterbuch-Operationen in Splay-Trees gelten die folgenden oberen Schranken ($T, T_1, T_2 \neq \emptyset$):

$$\text{Access}(x, T) : \begin{cases} 3 \log \left(\frac{W}{w(x)} \right) + \mathcal{O}(1), & \text{falls } x \in T \\ 3 \log \left(\frac{W}{\min\{w(x^-), w(x^+)\}} \right) + \mathcal{O}(1), & \text{falls } x \notin T \end{cases}$$
$$\text{Split}(x, T) : \begin{cases} 3 \log \left(\frac{W}{w(x)} \right) + \mathcal{O}(1), & \text{falls } x \in T \\ 3 \log \left(\frac{W}{\min\{w(x^-), w(x^+)\}} \right) + \mathcal{O}(1), & \text{falls } x \notin T \end{cases}$$

Satz 61

Für die amortisierten Kosten der Wörterbuch-Operationen in Splay-Trees gelten die folgenden oberen Schranken ($T, T_1, T_2 \neq \emptyset$):

$$\text{Join}(T_1, T_2) : 3 \log \left(\frac{W}{w(i)} \right) + \mathcal{O}(1), \quad x \text{ maximal in } T_1$$

$$\text{Insert}(x, T) : 3 \log \left(\frac{W - w(x)}{\min\{w(x^-), w(x^+)\}} \right) + \log \left(\frac{W}{w(x)} \right) + \mathcal{O}(1)$$

$x \notin T$

$$\text{Delete}(x, T) : 3 \log \left(\frac{W}{w(x)} \right) + 3 \log \left(\frac{W - w(x)}{w(x^-)} \right) + \mathcal{O}(1),$$

falls x nicht minimal in T

Literatur zu Splay-Trees:



Daniel D. Sleator, Robert E. Tarjan:

Self-adjusting binary search trees

Journal of the ACM **32**(3), pp. 652–686 (1985)