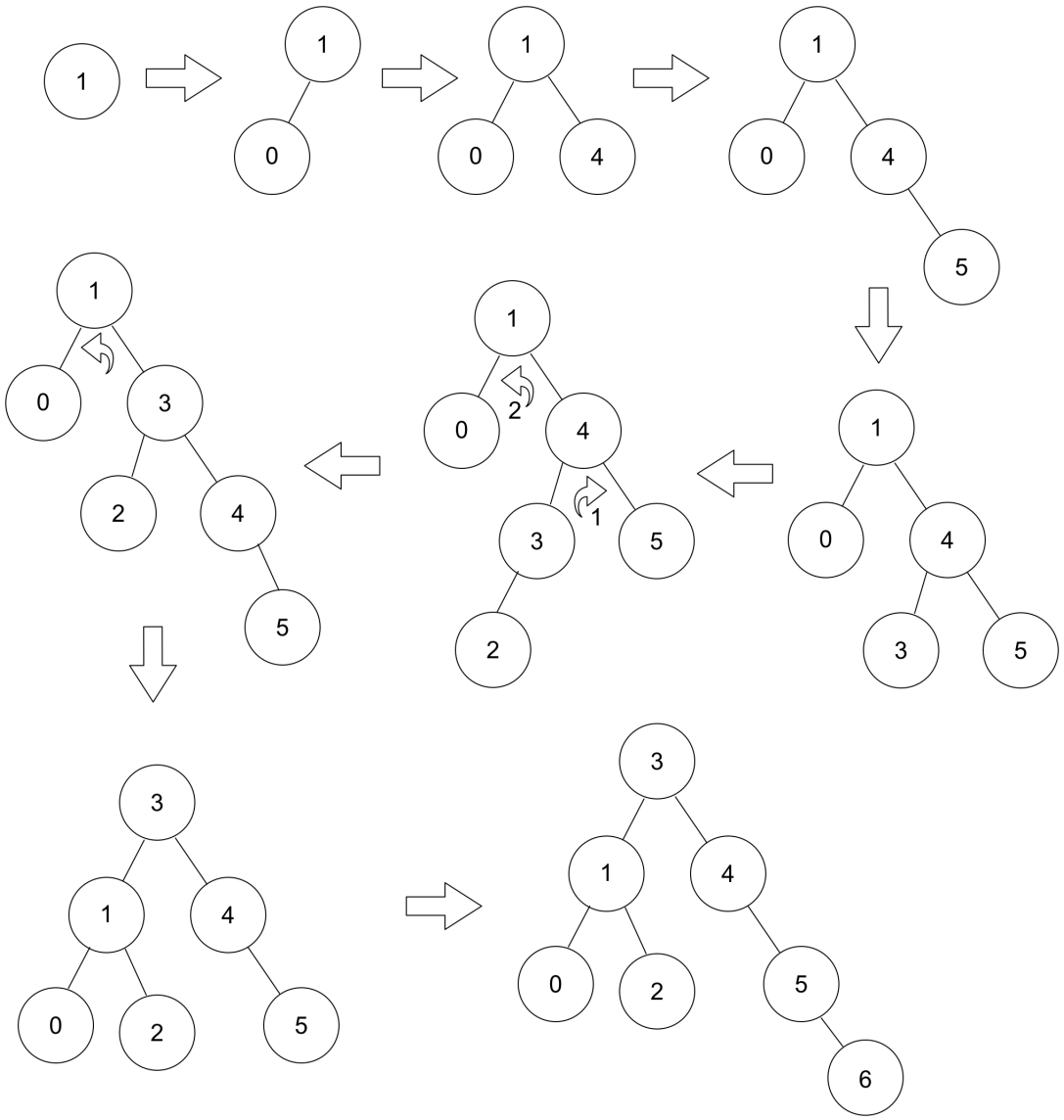
# Datenstrukturen und effiziente Algorithmen
# Blatt 6

Markus Vieth, David Klopp, Christian Stricker

4. Dezember 2015

# Aufgabe 1

Anmerkung:
Eine Rotation besteht aus einer konstanten Anzahl von Verknüpfungsänderungen angewendet auf eine konstante Anzahl von Knoten. Die Laufzeit liegt somit in O(1).




# Aufgabe 2

Füge rekursive den Wurzelknoten von $T_2$ an den größten Knoten in $T_1$. Beim wiederhochspringen der Rekursion wird die Höhe der Teilbäume verglichen (Da jeder Knoten seine Höhe kennt, reicht eine einfache Addition um die richtige Höhe des Knotens im $T_1$ zu bestimmen) und notfalls rotiert. Es muss nur ab dem größten Knoten des $T_1$ Baumes der Gesamtbaum ausgeglichen werden, da die Teilbäume des größten Knoten von $T_1$ schon ausgeglichen sind.
Laut Vorlesung ist der Gesamtbaum ausgeglichen, wenn der Rotier-Algorithmus rekursive einmal durchgelaufen ist.
Die Laufzeit ist O(log(n)), da einmal der größte Knoten in $T_1$ gesucht wird.
Da $h_1 \leq h_2$ gilt: $\log(m) \leq \log(n) \Rightarrow O(\log(n) + \log(m)) = O(\log(n))$

# Aufgabe 3

## a)

Für die Ermittlung des einzusparenden Betrags, müssen folgende Randbedienungen beachtet werden, sei dabei m die aktuelle Größe des Arrays:

- Das Einfügen oder Löschen eines Elementes kostet 1€
- Das Einfügen des (m+1)ten Elementes kostet:
  - 1€ für das Einfügen
  - 2m€ für das Anlegen des neuen Arrays
  - m€ für das kopieren des bisherigen Arrays

Beim vergrößern des Arrays müssen alle vorherigen Einträge kopiert werden, dabei werden die vorderen Einträge insgesamt häufiger kopiert, als die hinteren. Deshalb müssen in einem m langem Array die letzten $\frac{m}{2}$ den Kopiervorgang der ersten $\frac{m}{2}$ Elemente, sowie den eigenen mitbezahlen, also zusätzlich 2€ einsparen. Auch das Anlegen des neuen 2m langen Arrays muss von diesen $\frac{m}{2}$ Elementen bezahlt werden, also $\frac{2m\,e}{\frac{m}{2}} = \frac{4m\,e}{m} = 4\,e$. Wir haben also das Einfügen/Löschen 1€, das Kopieren 2€ und das Anlegen des neuen Arrays 4€, also eine Einzahlung in Höhe von 7€ pro Operation.

| Operation | Einzahlung | Kosten | Endkontostand |
|---|---|---|---|
| m Elemente Einfügen/löschen | 7m€ | m€ | 6m€ |
| (m+1)te Element einfügen | 7€ | 1€+2m€+m€ | 6m€+7€-1€-3m€=3m€+6€ |
| weitere m-1 Elemente einfügen | (7m-7)€ | (m-1)€ | (3m+6+7m-7-m+1)€=9m€ |
| (2m+1)te Element einfügen | 7€ | (1+4m+2m)€ | (9m+7-1-4m-2m)€ =3m€-6€ |

Wie aus der Tabelle ersichtlich, ist der Endkontostand nach dem Vergrößern des Arrays jedes mal gleich, somit ist die Einzahlung ausreichend.
Da das Einbezahlen eines konstanten Betrags pro Operation alle Kosten abdeckt, müssen die amortisierten Kosten auch konstant sein.

## b)

**Gegenbeispiel** Nach dem einfügen des (m+1)ten Elements wird dieses wieder gelöscht:

| Operation | Kosten | Gesamtkosten | Anzahl der Operationen | Durchschnittskosten |
|---|---|---|---|---|
| m Elemente Einfügen/löschen | m€ | m€ | m | 1€ |
| (m+1)te Element einfügen | 1€+2m€+m€ | 1€+4m | m+1 | <4€ |
| (m+1)te Element löschen | (1+m+m)€ | 2€+6m | m+2 | <6€ |

Aus der Tabelle wird ersichtlich, dass das n-malige Einfügen und Löschen des (m+1)ten Elements lineare Kosten.

## c)

Der Aufwand von a) ist aus dem selben Grund nicht konstant, wie er in dem Aufgabenteil b) gezeigt wurde.
Zum Aufwand der b)

- Das Löschen kostet 1€
- Das Verkleinern kostet:
  - Erzeugen des Arrays $\frac{n}{2}$€
  - Kopieren $\frac{n}{4}$€

$\frac{3n}{4}$ Löschoperationen müssen das Erzeugen $\frac{n}{2}$€ und das Kopieren $\frac{n}{4}$€ bezahlen, also insgesamt $\frac{n}{2}+\frac{n}{4}$€= $\frac{3n}{4}$€. Somit müssen pro Löschoperation lediglich 2€ gespart werden. Da wir danach wieder die selbe Ausgangssituation haben, wie in Aufgabenteil a) gezeigt, wird ein Fall wie in Aufgabenteil b) vermieden.