Grundlagen: Algorithmen und Datenstrukturen

Prof. Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Sommersemester 2010

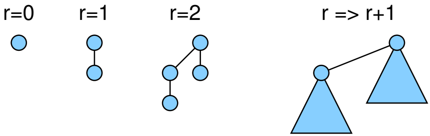
Übersicht

1. Priority Queues
   - Binomial Heaps

Binomial Heaps

basieren auf Binomial-Bäumen

- Form-Invariante:



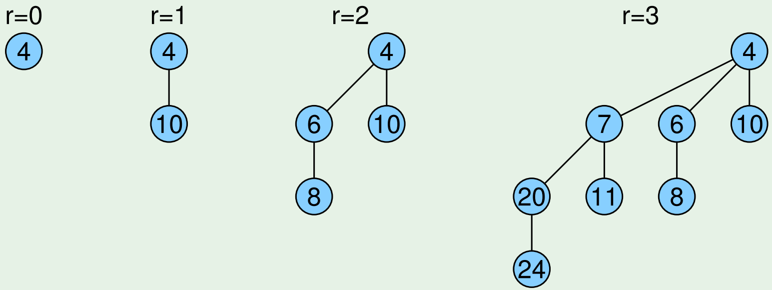
- Heap-Invariante:

$$\text{prio(Vater)} \leq \text{prio(Kind)}$$

Binomial-Bäume

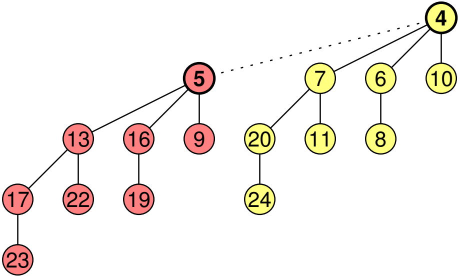
Beispiel

Korrekte Binomial-Bäume:

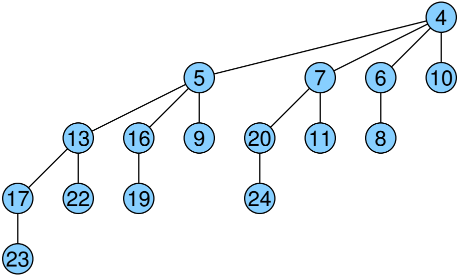


Binomial-Baum: Merge

Wurzel mit größerem Wert wird neues Kind der Wurzel mit kleinerem Wert! (Heap-Bedingung)

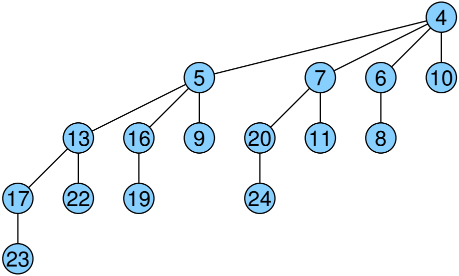


aus zwei B_{r-1} wird ein B_r

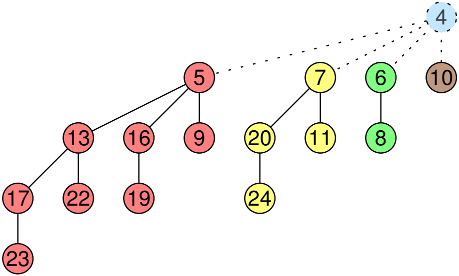


Binomial-Baum: Löschen der Wurzel (deleteMin)

aus einem B_r



werden $B_{r-1}, \ldots, B_0$



Binomial-Baum: Knotenanzahl

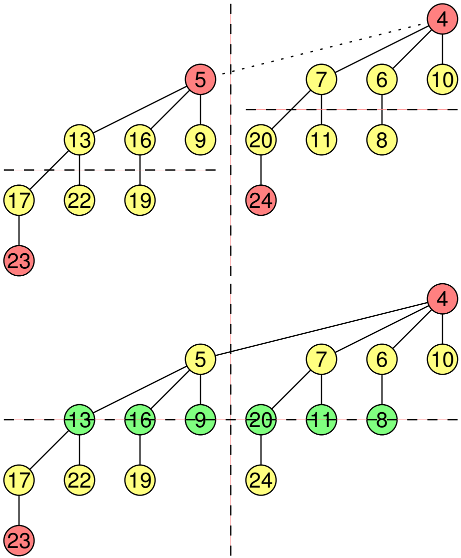
B_r hat auf Level $k \in \{0, \dots, r\}$ genau $\binom{r}{k}$ Knoten

Warum?

Bei Bau des B_r aus 2 B_{r-1} gilt:

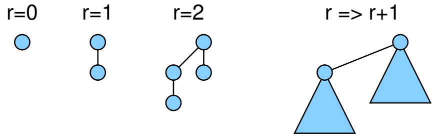
$$\binom{r}{k} = \binom{r-1}{k-1} + \binom{r-1}{k}$$

Insgesamt: B_r hat 2^r Knoten



Binomial-Bäume

Eigenschaften von Binomial-Bäumen:



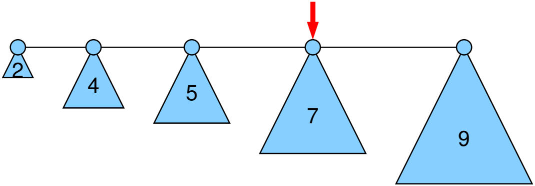
Binomial-Baum vom Rang r

- hat 2^r Knoten
- hat maximalen Grad r (Wurzel)
- zerfällt bei Entfernen der Wurzel in r Binomial-Bäume von Rang 0 bis $r-1$
- hat auf Level $\ell \in \{0, \ldots, r\}$ genau $\binom{r}{\ell}$ Knoten

Binomial Heap

Binomial Heap:

- verkettete Liste von Binomial-Bäumen
- pro Rang maximal 1 Binomial-Baum
- Zeiger auf Wurzel mit minimalem Prioritätswert

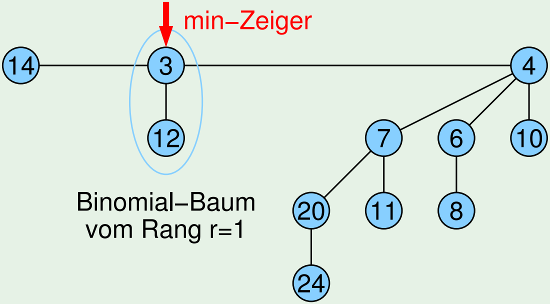


(Zahlen: Ränge)

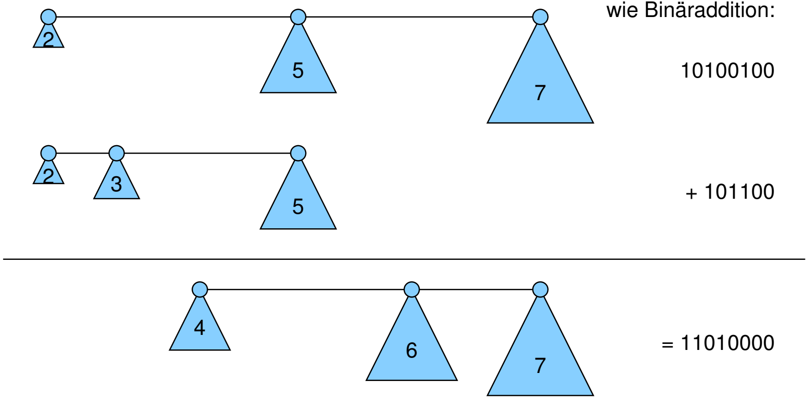
Binomial Heap

Beispiel

Korrekter Binomial Heap:



Merge von zwei Binomial Heaps



wie Binäraddition:

10100100

+ 101100

= 11010000

Aufwand für Merge: $\mathcal{O}(\log n)$

Binomial Heaps

B_i: Binomial-Baum mit Rang i

Operationen:

- merge: $\mathcal{O}(\log n)$
- insert(e): Merge mit B_0, Zeit $\mathcal{O}(\log n)$
- min(): spezieller Zeiger, Zeit $\mathcal{O}(1)$
- deleteMin():
  sei das Minimum in B_i,

  durch Löschen der Wurzel zerfällt der Binomialbaum in $B_0, \ldots, B_{i-1}$

  Merge mit dem restlichen Binomial Heap kostet $\mathcal{O}(\log n)$

Binomial Heaps

Weitere Operationen:

- decreaseKey(e, k): siftUp-Operation in Binomial-Baum von e
  Zeit: $\mathcal{O}(\log n)$

- remove(e): setze prio(e) $= -\infty$ und wende siftUp-Operation auf e an bis e in der Wurzel, dann weiter wie bei deleteMin
  Zeit: $\mathcal{O}(\log n)$

Verbesserungen

- Fibonacci-Heap:

  Verbesserung von Binomial Heaps, so dass amortisierte Kosten von decreaseKey $\mathcal{O}(1)$ sind

- ganzzahlige Werte:

  Priority Queues bekannt, die für decreaseKey und insert Zeit $\mathcal{O}(1)$ und für deleteMin Zeit $\mathcal{O}(\log\log n)$ benötigen