Algorithmen und Datenstrukturen

Prioritätswarteschlangen mit Binomial-Heaps

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

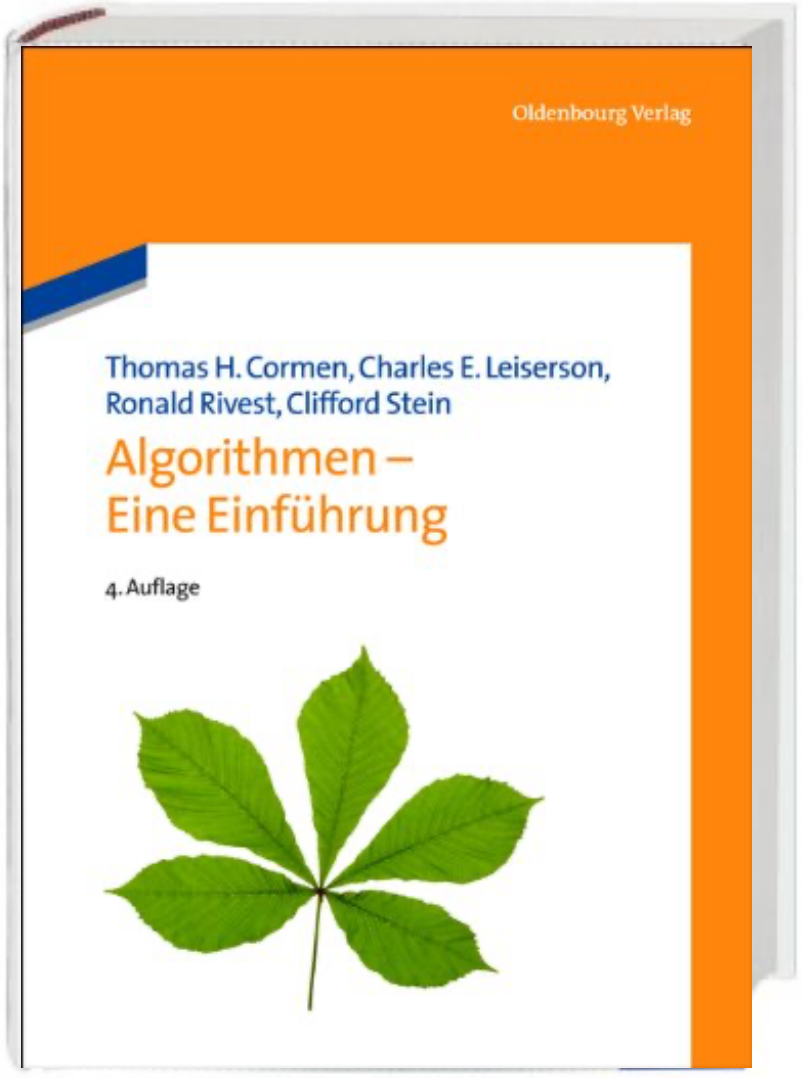
Felix Kuhr (Übungen)

sowie viele Tutoren

Prioritätswarteschlangen: Binärer Heap

Laufzeiten binärer Heap:

- build: O(n)
- insert: O(log n)
- min: O(1)
- deleteMin: O(log n)



Prioritätswarteschlange mit binärem Heap

Operator	Laufzeit
insert	O(log n)
min	O(1)
deleteMin	O(log n)
delete	O(log n)*
decreaseKey	O(log n)*
merge	O(n)

* Wenn Position von e bekannt

R. Mendelson, R. Tarjan, M. Thorup, and U. Zwick.
Melding Priority Queues. Proceedings of 9th SWAT, **2004**

Anwendungen für log-n-merge

1. Lastumverteilung
   - Delegierung der Aufträge für einen Prozessor an einen anderen (evtl. neu hinzugeschalteten) Prozessor
2. Reduce-Operation

   (siehe MapReduce Programmiermodell)
   - Mischung von parallel ermittelten Ergebnissen, jeweils mit Bewertung bzw. Sortierung

Binomial-Heap zum schnellen Verschmelzen

Binomial-Heap basiert auf sog. Binomial-Bäumen

Binomial-Baum muss erfüllen:

- Form-Invariante (r: Rang):

r=0 r=1 r → r+1

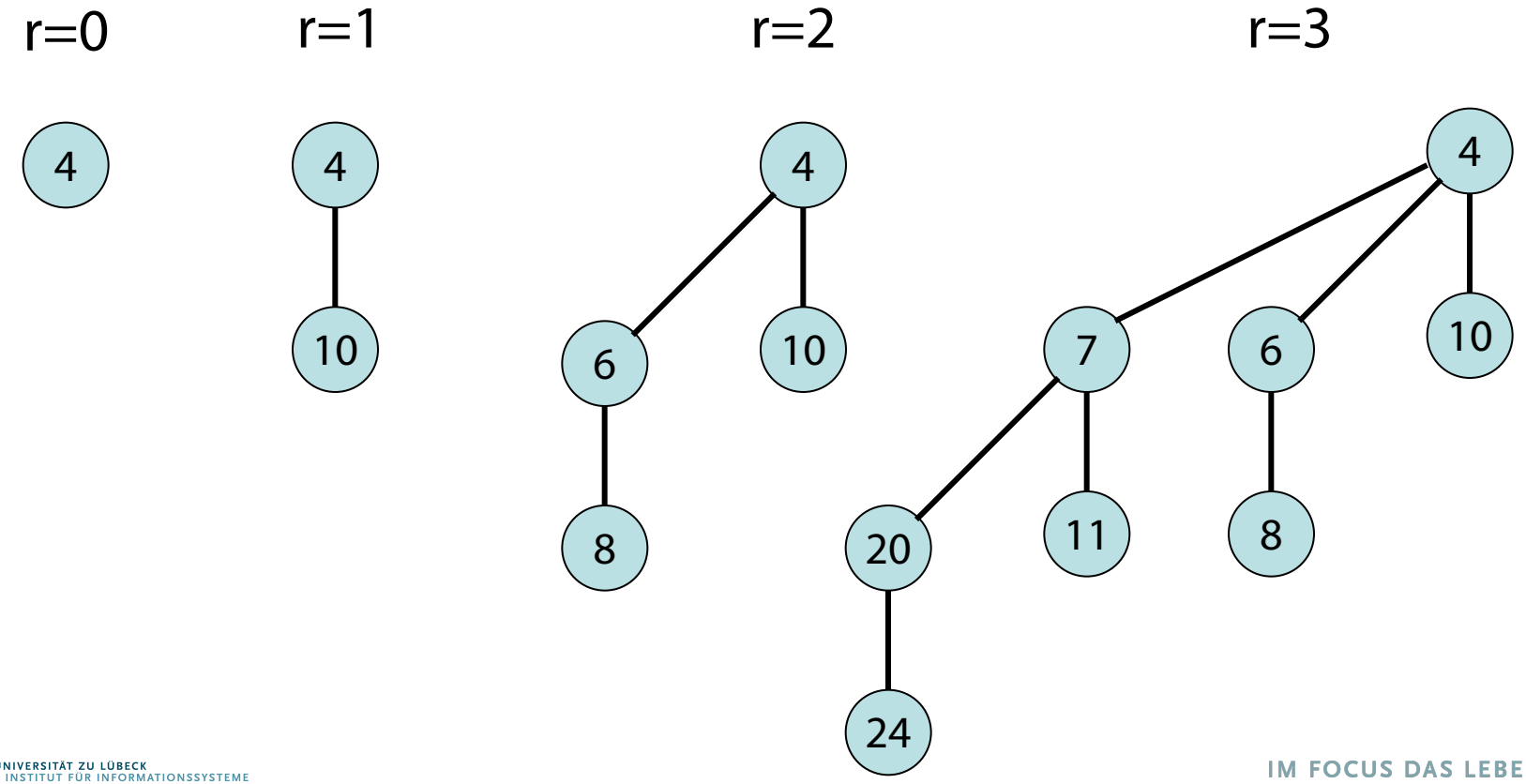
- MinHeap-Invariante: key(Vater) ≤ key(Kinder)

Jean Vuillemin: A data structure for manipulating priority queues. Communications of the ACM 21, S. 309–314, **1978**

Binomial-Heap

Beispiel für korrekte Binomial-Bäume:

Hier mit MinHeap-Eigenschaft



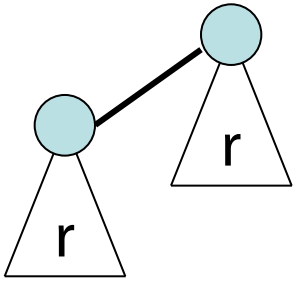
Binomial-Heap

Eigenschaften von Binomial-Bäumen:

r=0

r=1

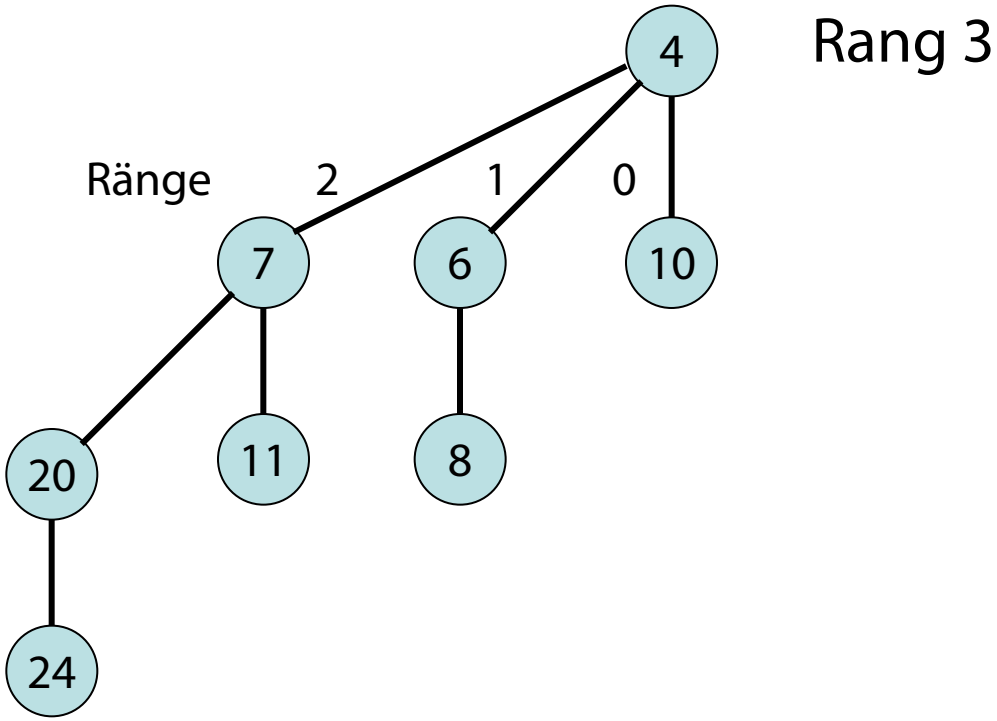
r → r+1



- 2^r Knoten
- maximaler Grad r (bei Wurzel)
- Wurzel weg: Zerfall in Binomial-Bäume mit Rang 0 bis r-1

Binomial-Heap

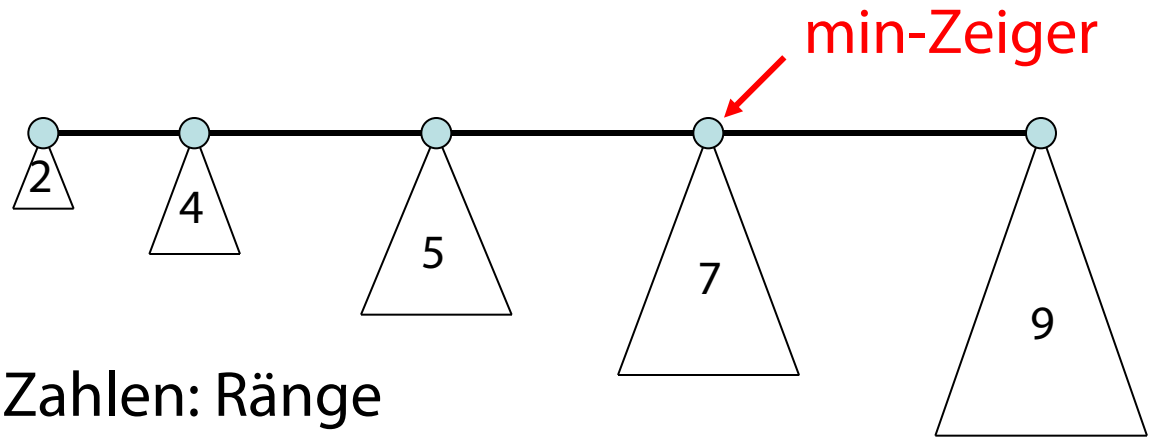
Beispiel für Zerfall in Binomial-Bäume mit Rang 0 bis r-1



Binomial-Heap

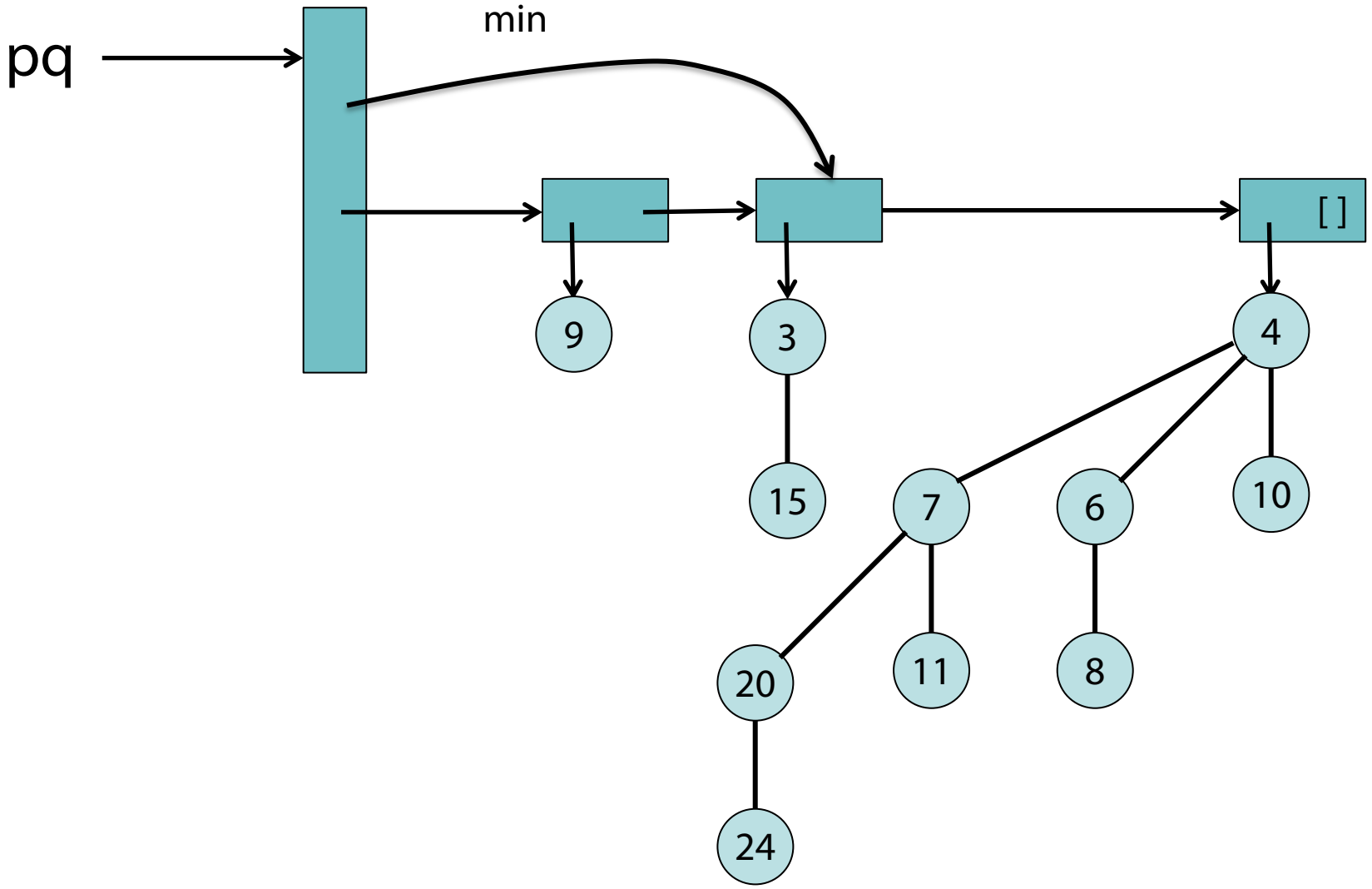
Binomial-Heap:

- verkettete Liste von Binomial-Bäumen
- Pro Rang maximal 1 Binomial-Baum
- Zeiger auf Wurzel mit minimalem key



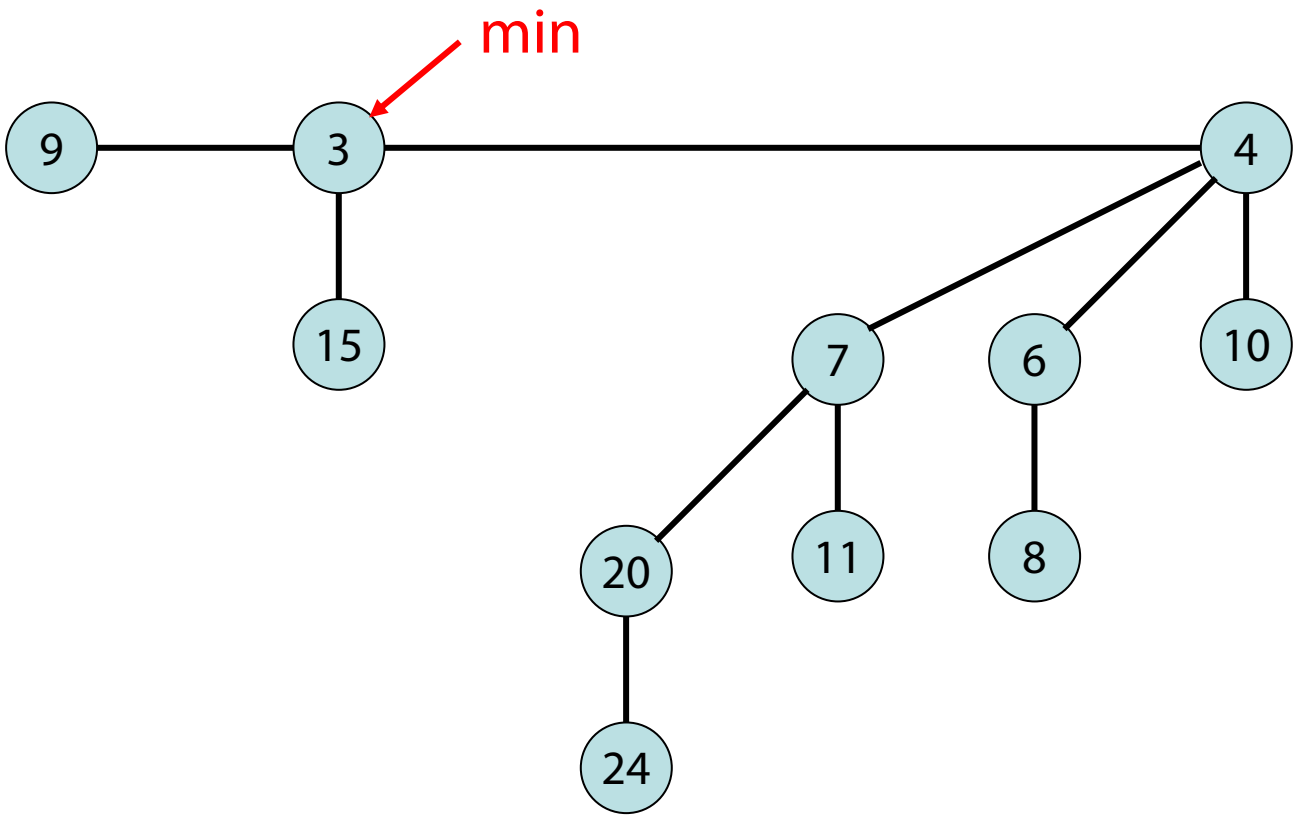
Zahlen: Ränge

Prioritätswarteschlangen als Binomial-Heaps



Binomial-Heap

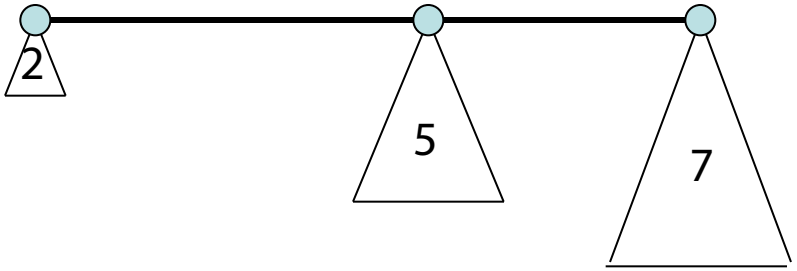
Abstrakte Darstellung:



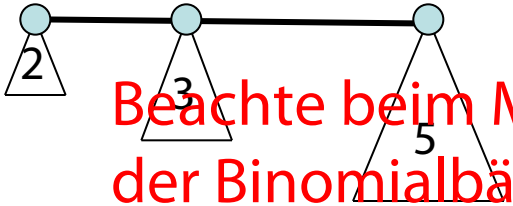
Anzahl der Bäume auf der Kette

- Binomial-Heap-Invariante:
  Pro Rang maximal 1 Binomial-Baum
- Was heißt das?
- Für n Knoten können höchstens log n viele Binomialbäume in der Kette vorkommen
  (dann müssen alle Knoten untergebracht sein)

Beispiel einer Merge-Operation



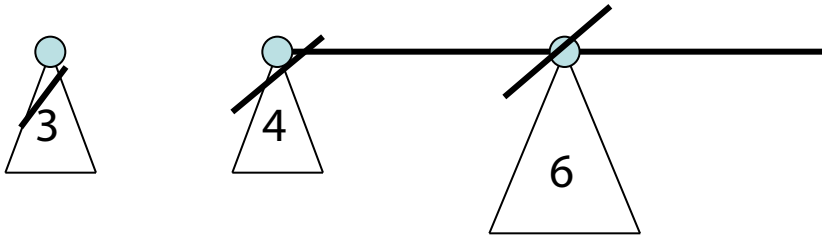
H_1



H_2

Beachte beim Mergen der Binomialbäume die Heap-Eigenschaft!

Zahlen geben die Ränge an



Ergebnis-Heap

Operationen auf Binomial-Heaps

Sei B_i: Binomial-Baum mit Rang i

- merge(pq, pq'): Aufwand für Merge-Operation: O(log n)
- insert(e, pq): Merge mit B_0, Zeit O(log n)
- min: spezieller Zeiger, Zeit O(1)
- deleteMin: sei Minimum in Wurzel von B_i,
  Löschen von Minimum: $B_i \rightarrow B_0,\ldots,B_{i-1}$. Diese zurückmergen in Binomial-Heap. Zeit dafür O(log n).

Binomial-Heap

- decreaseKey(e, pq, Δ): siftUp-Operation in Binomial-Baum von e und aktualisierte min-Zeiger. Zeit O(log n)

- delete(e, pq): (min-Zeiger zeigt nicht auf e) setze key(e):= -∞ und wende siftUp-Operation auf e an, bis e in der Wurzel; dann weiter wie bei deleteMin.
  Zeit O(log n)

Zusammenfassung

Laufzeit	Binärer-Heap	Binomial-Heap
insert	O(log n)	O(log n)
min	O(1)	O(1)
deleteMin	O(log n)	O(log n)
delete	O(log n)	O(log n)
decreaseKey	O(log n)	O(log n)
merge	O(n)	O(log n)