Technische Universität Braunschweig

Fibonacci-Heaps

Sándor Fekete

Fibonacci-Heaps

- Idee: Dynamische Datenstruktur bei sich änderenden Schlüsselwerten
- Fredman und Tarjan 1987
- Kontext: Dijkstra
- Zeiten:

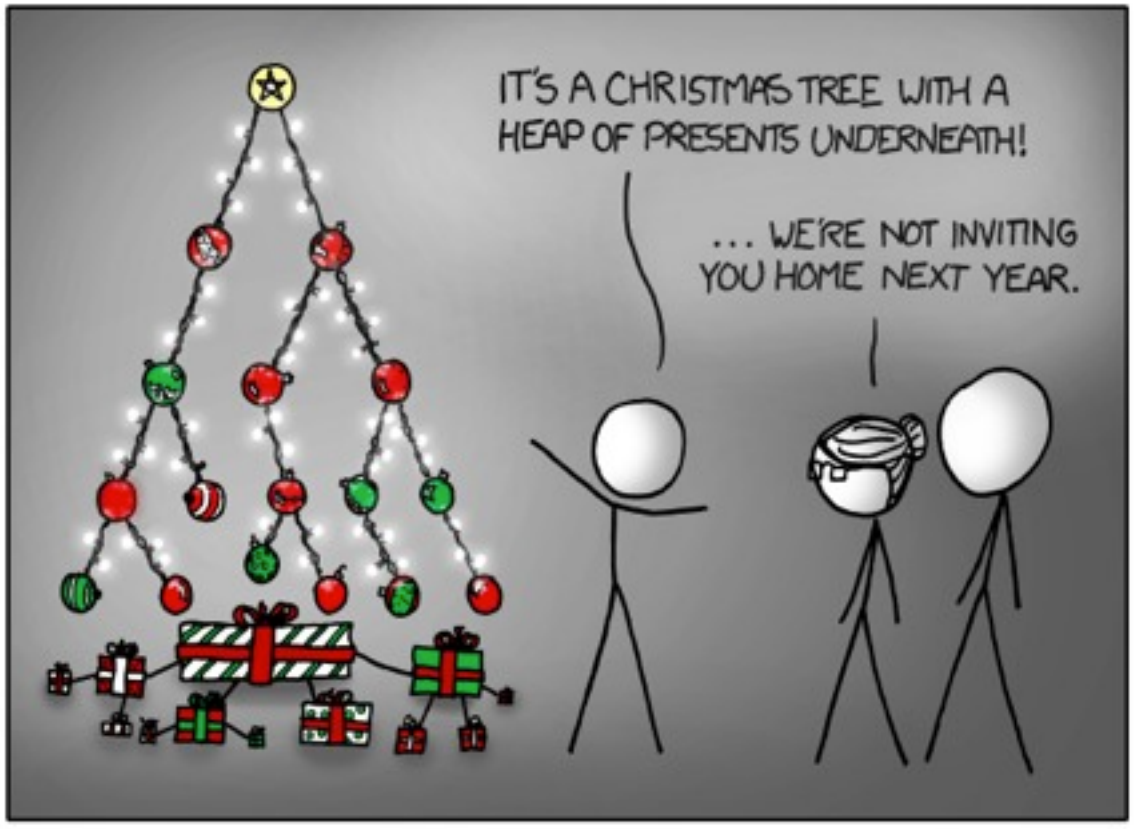
		Heaps			
Operation	Linked List	Binary	Binomial	Fibonacci †	Relaxed
make-heap	1	1	1	1	1
insert	1	log N	log N	1	1
find-min	N	1	log N	1	1
delete-min	N	log N	log N	log N	log N
union	1	N	log N	1	1
decrease-key	1	log N	log N	1	1
delete	N	log N	log N	log N	log N
is-empty	1	1	1	1	1

† amortized

this time

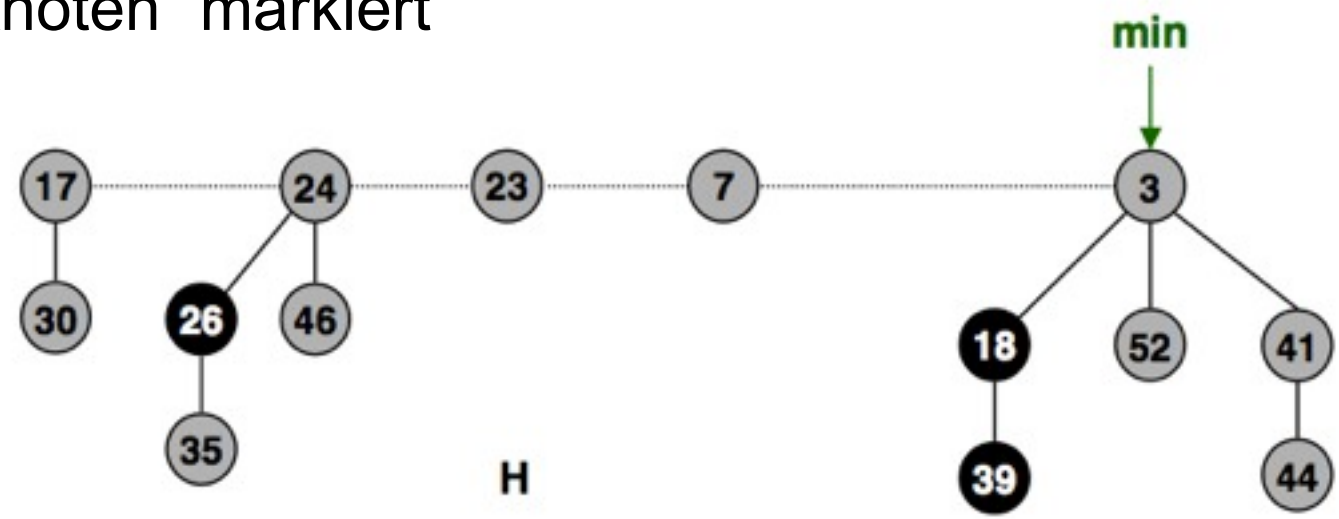
Heaps

- Struktur: Schlüsselwerte nach Größe angeordnet



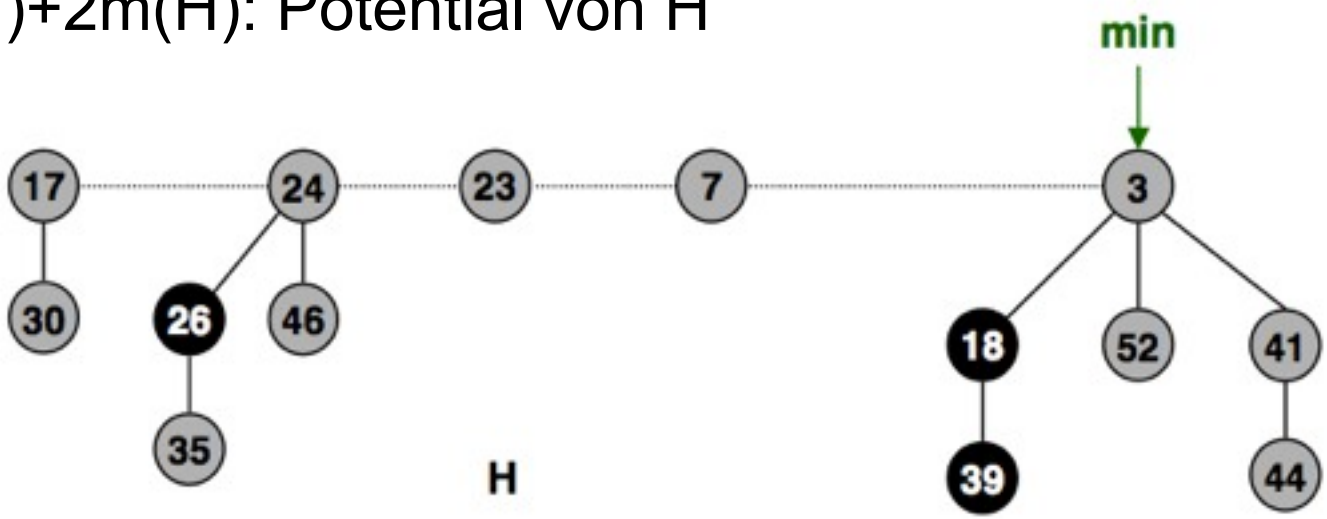
Fibonacci-Heaps: Struktur

- Vereinigung von aufwärts sortierten Bäumen
- Wurzeln per zyklischer doppelt verketteter Liste zusammengefasst (“Schlüsselbund”)
- Pointer auf globales Minimum
- Einige Knoten “markiert”



Fibonacci-Heaps: Größen

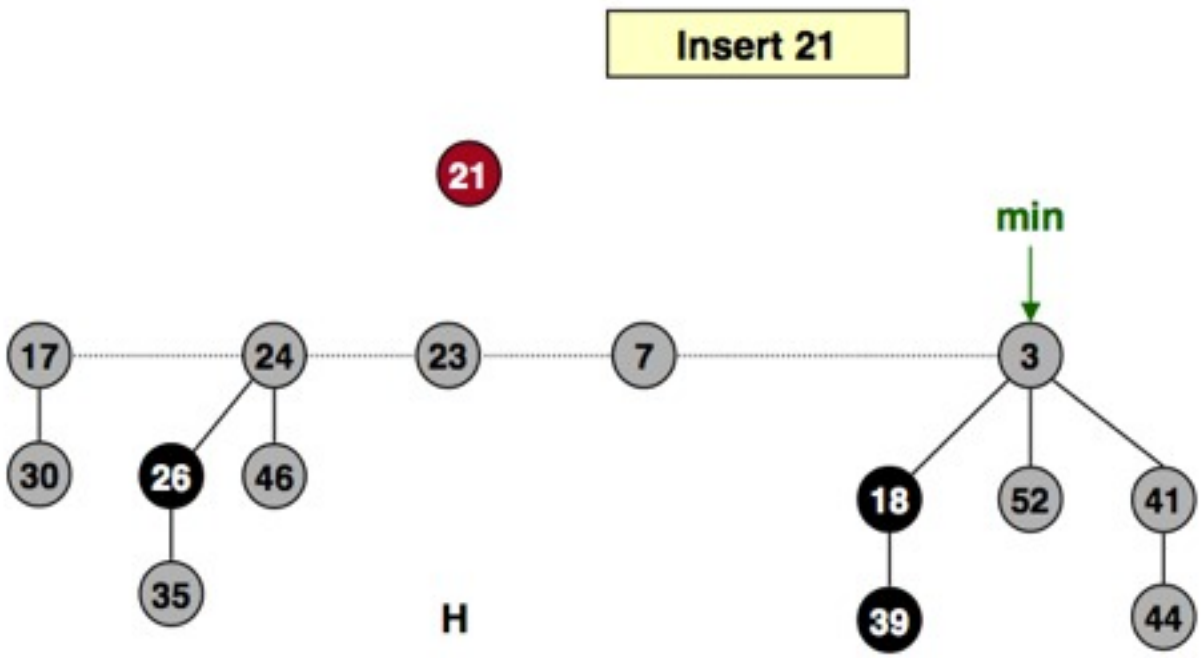
- D[x]: Grad von Knoten x
- mark[x]: Markierung von Knoten x
- t(H): Zahl der Bäume
- m(H): Zahl der markierten Knoten
- P(H)=t(H)+2m(H): Potential von H



t(H)=5, m(H)=3, P(H)=11

Fibonacci-Heaps: Einfügen

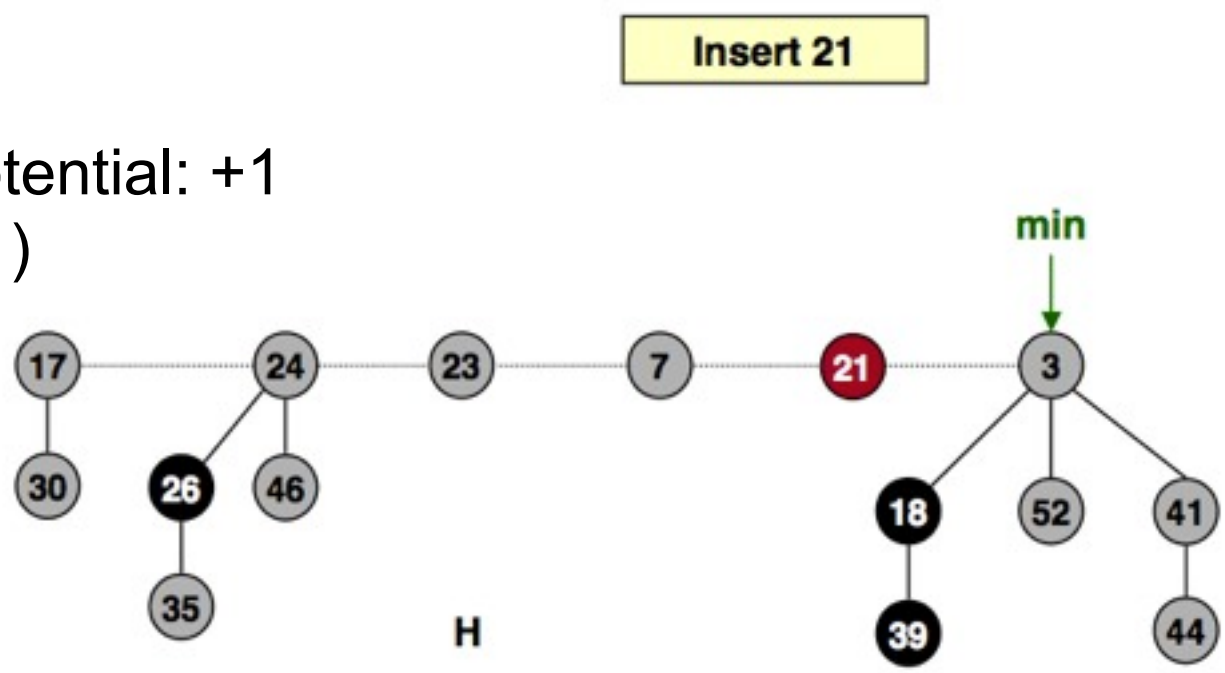
- Neues Element als Baum mit einem Knoten
- Einfügen links vom Baum mit globalem Minimum
- Ggf. globales Minimum aktualisieren



Fibonacci-Heaps: Einfügen

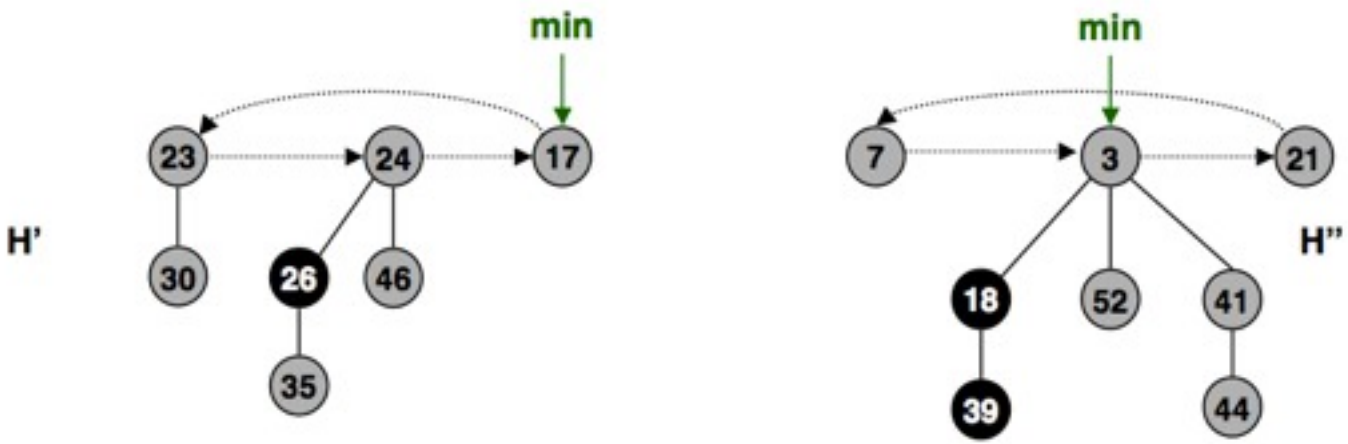
- Neues Element als Baum mit einem Knoten
- Einfügen links vom Baum mit globalem Minimum
- Ggf. globales Minimum aktualisieren

- Kosten: O(1)
- Änderung in Potential: +1
- Amortisiert: O(1)



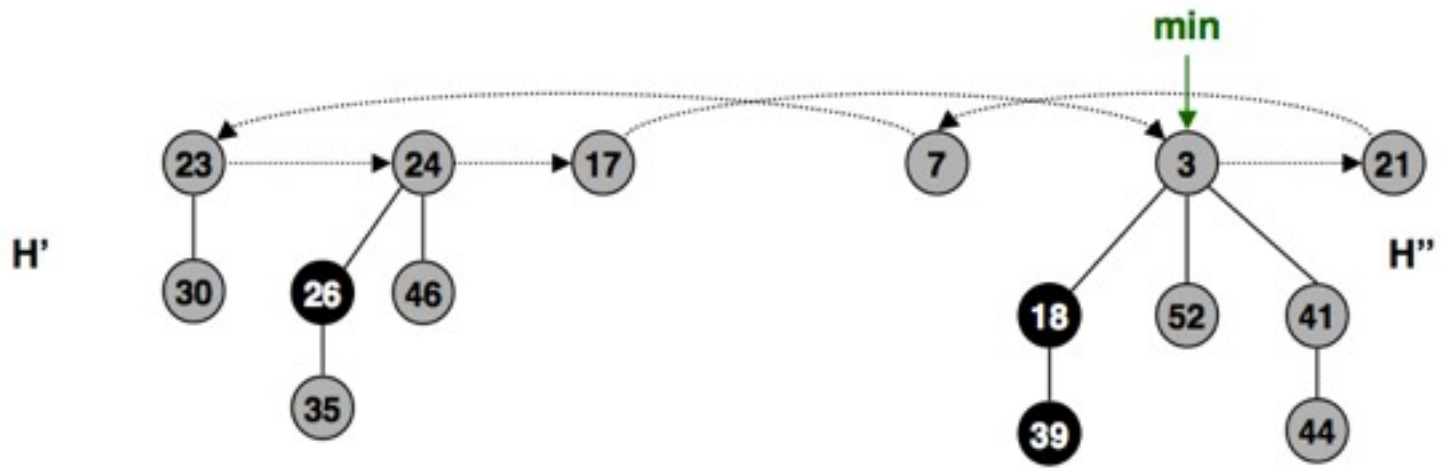
Fibonacci-Heaps: Vereinigung

- Füge Bäume zusammen
- Verschmelze Wurzellisten



Fibonacci-Heaps: Vereinigung

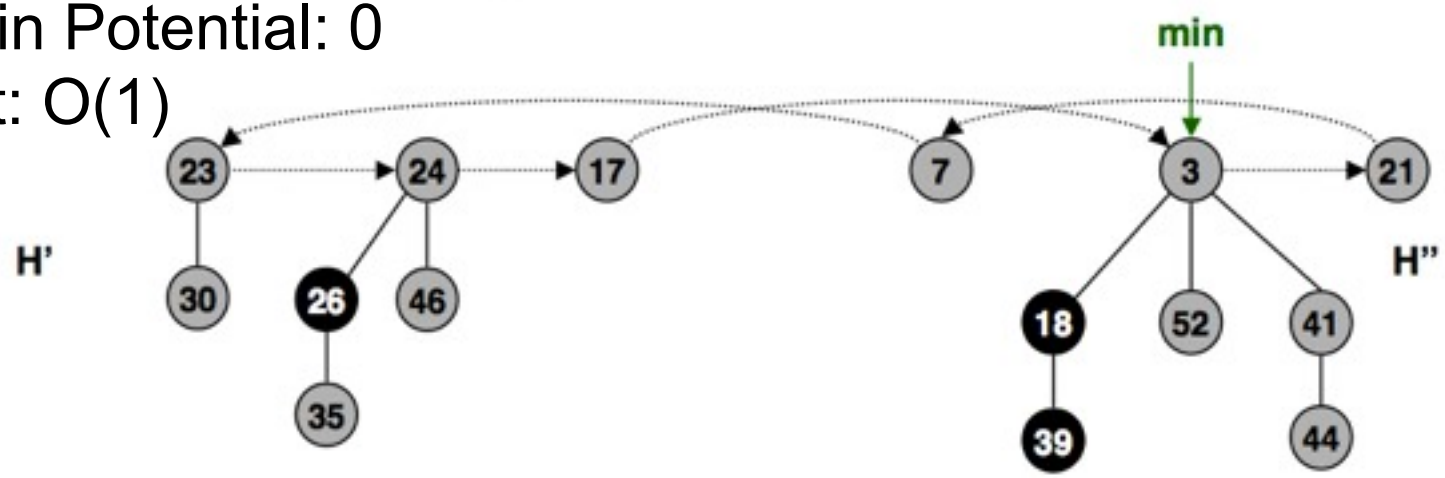
- Füge Bäume zusammen
- Verschmelze Wurzellisten



Fibonacci-Heaps: Vereinigung

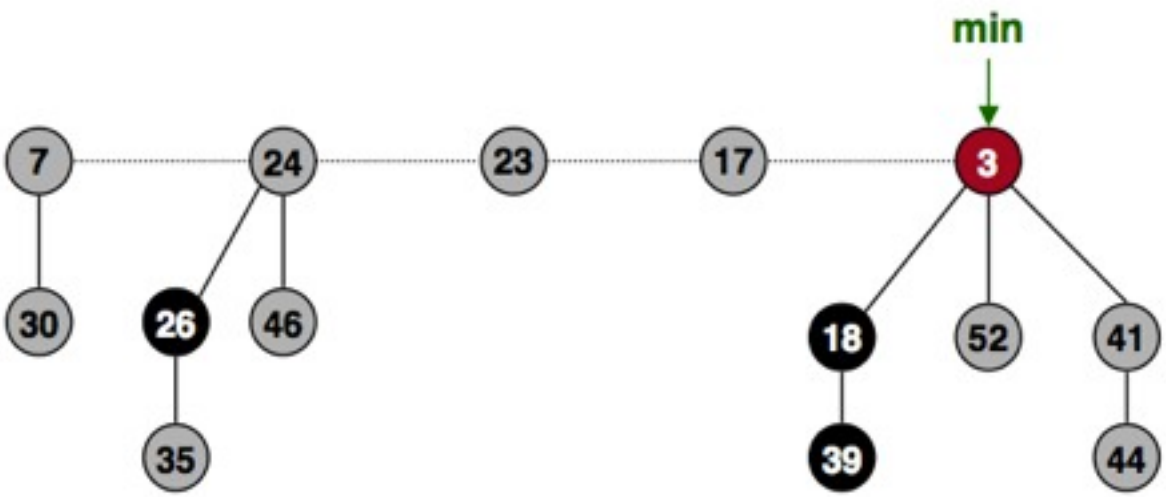
- Füge Bäume zusammen
- Verschmelze Wurzellisten

- Kosten: O(1)
- Änderung in Potential: 0
- Amortisiert: O(1)



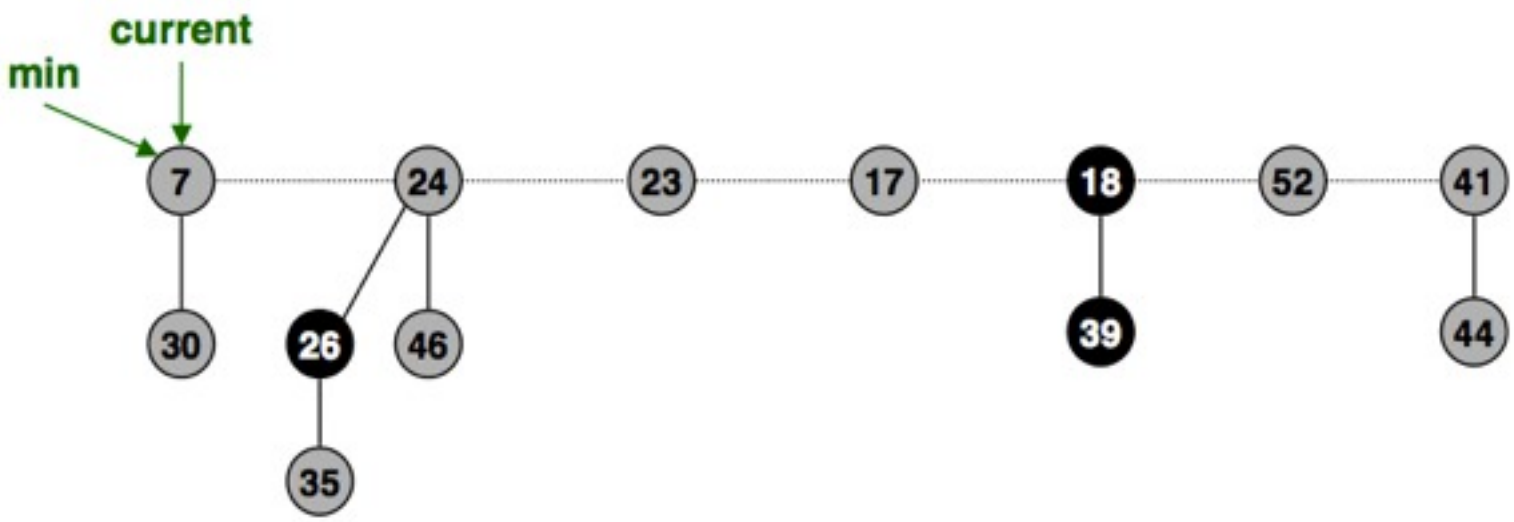
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.



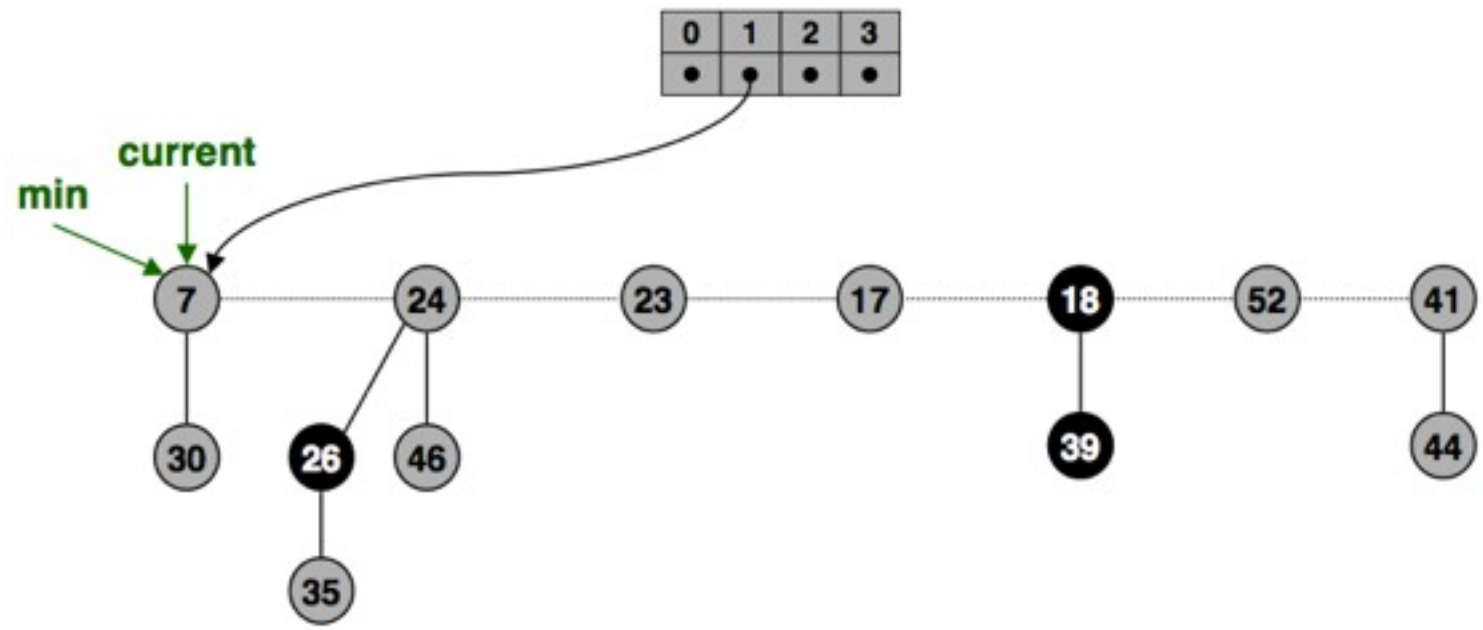
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



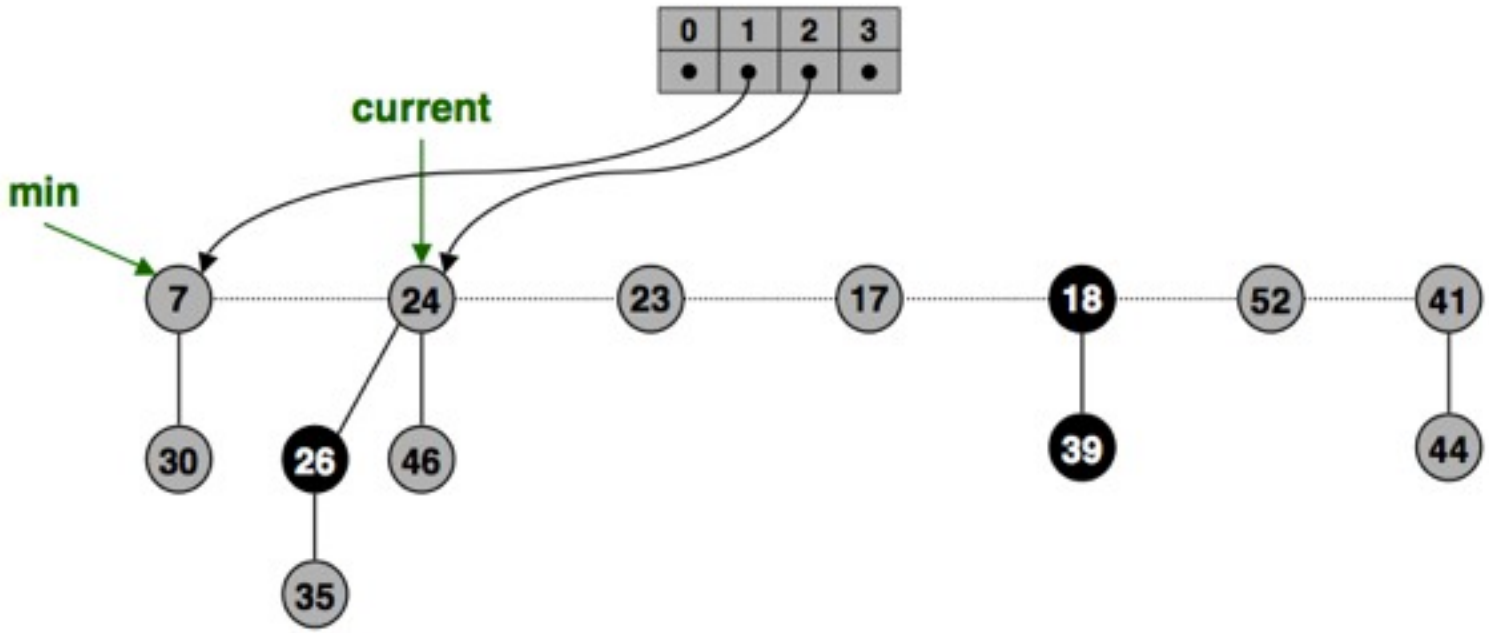
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



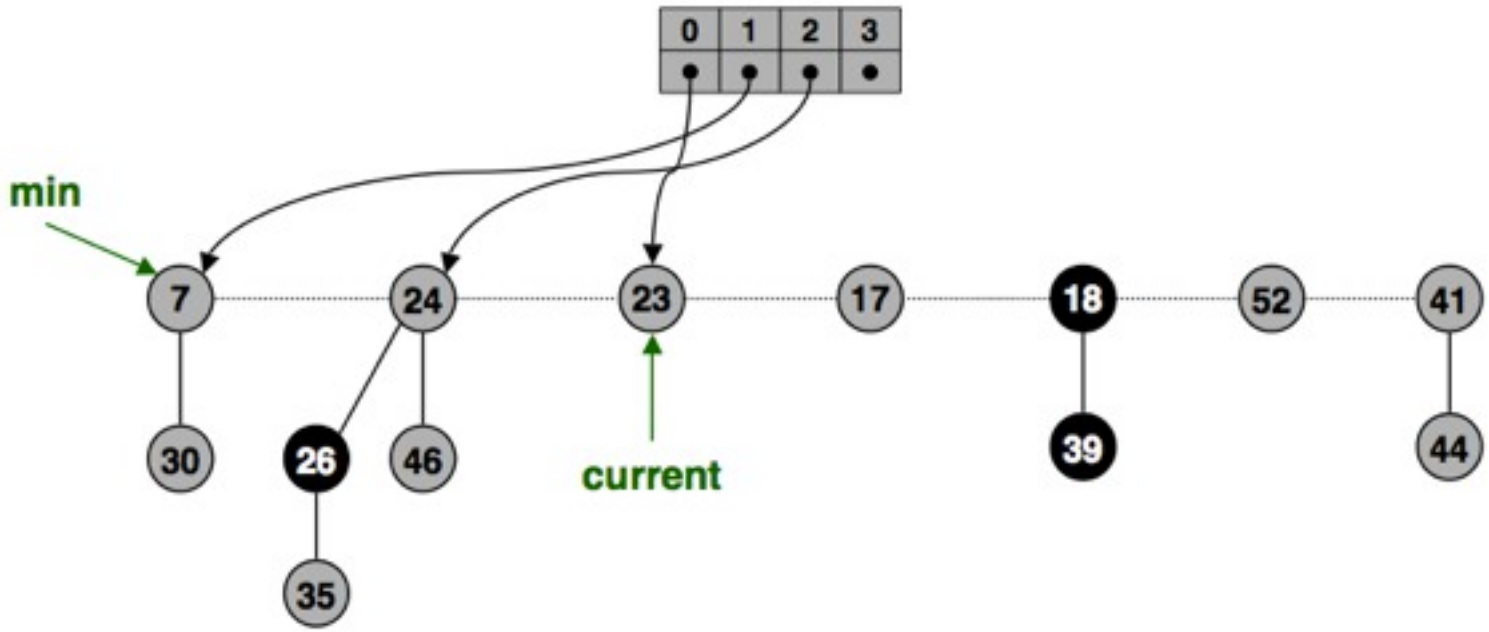
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



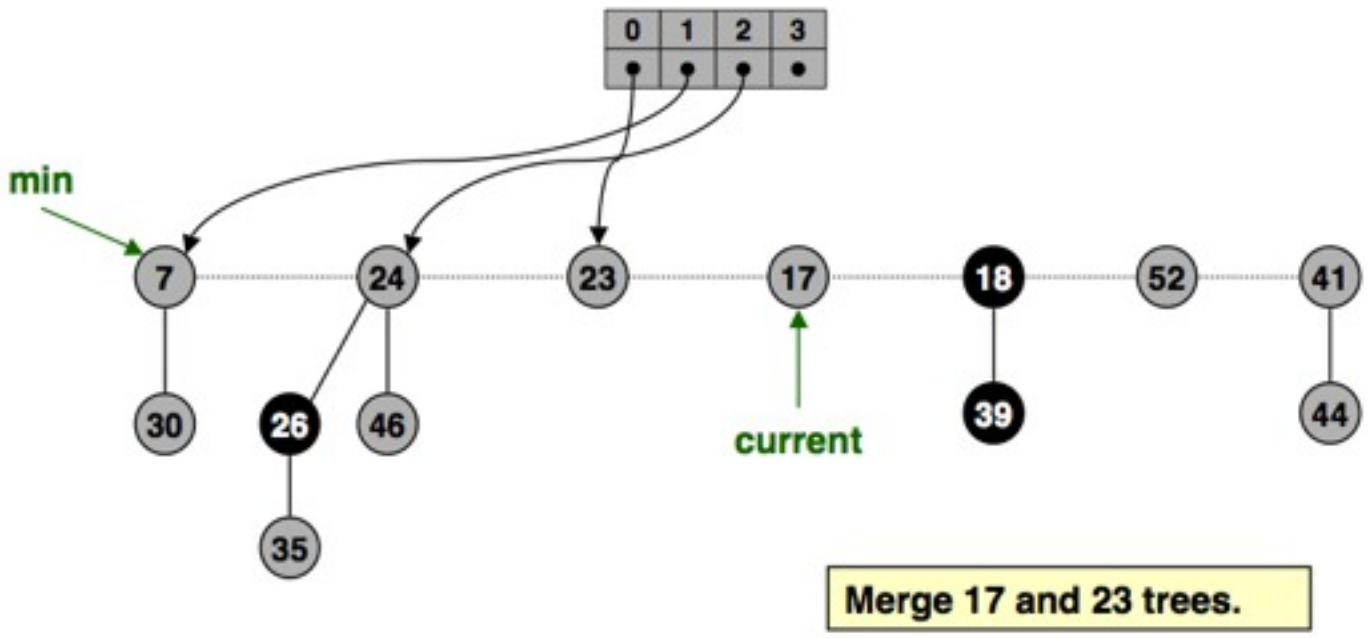
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



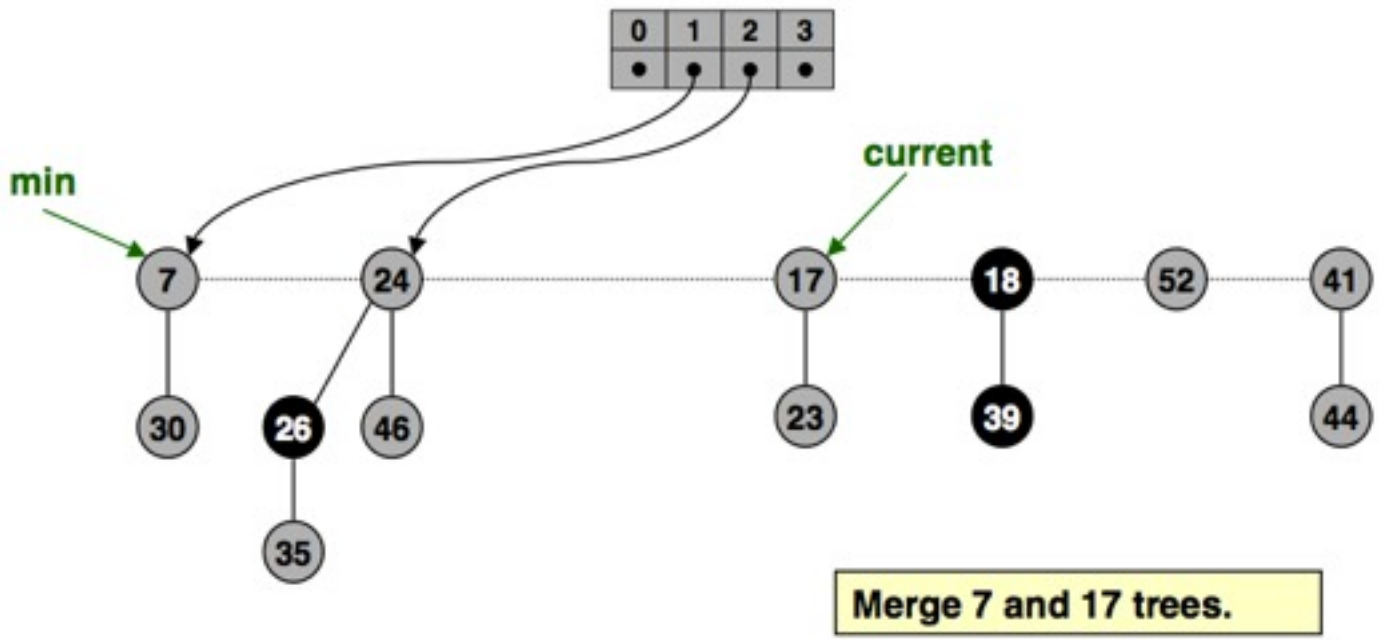
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



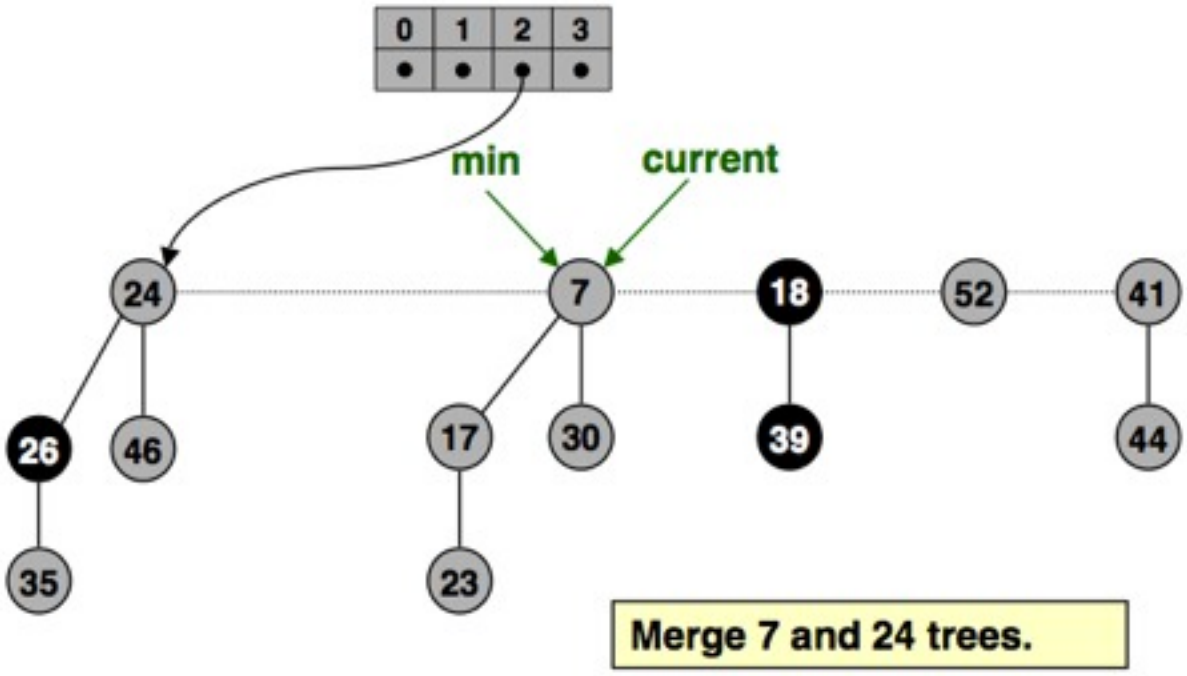
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



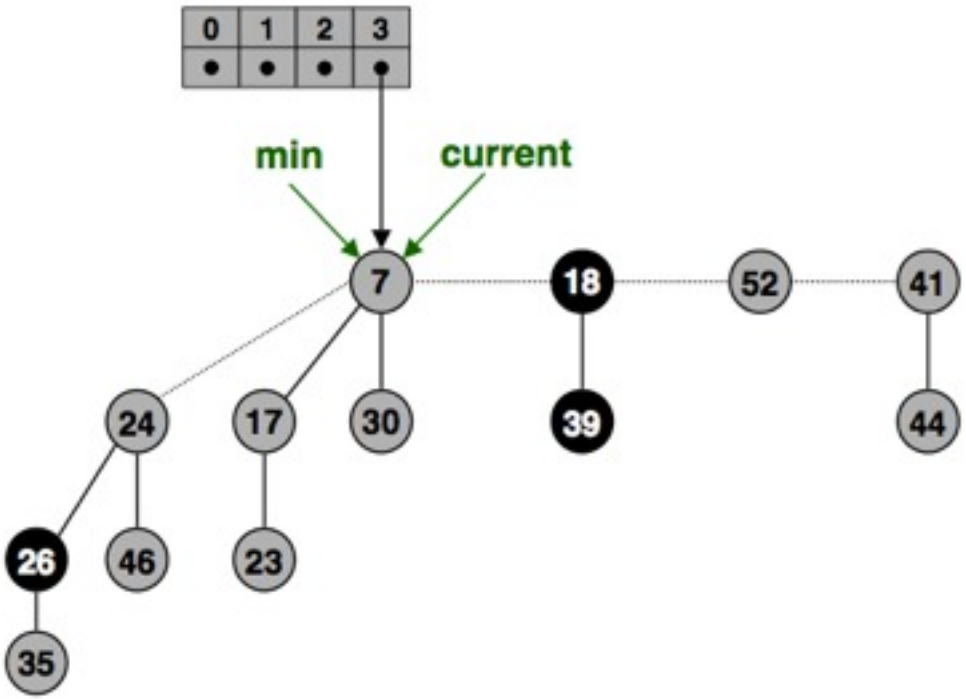
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



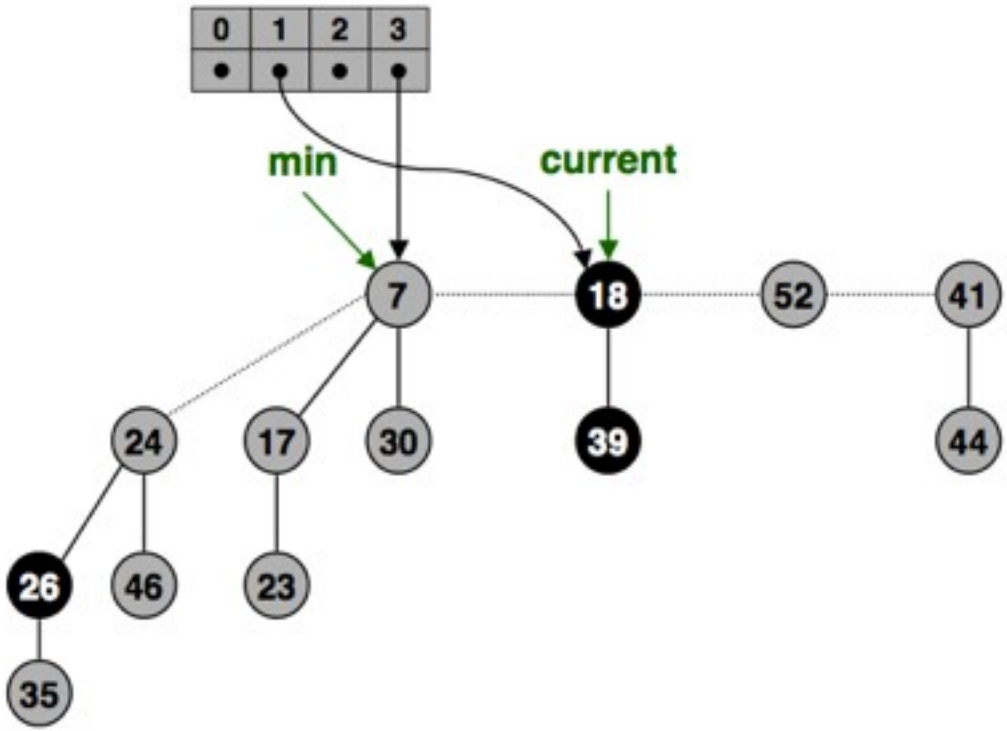
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



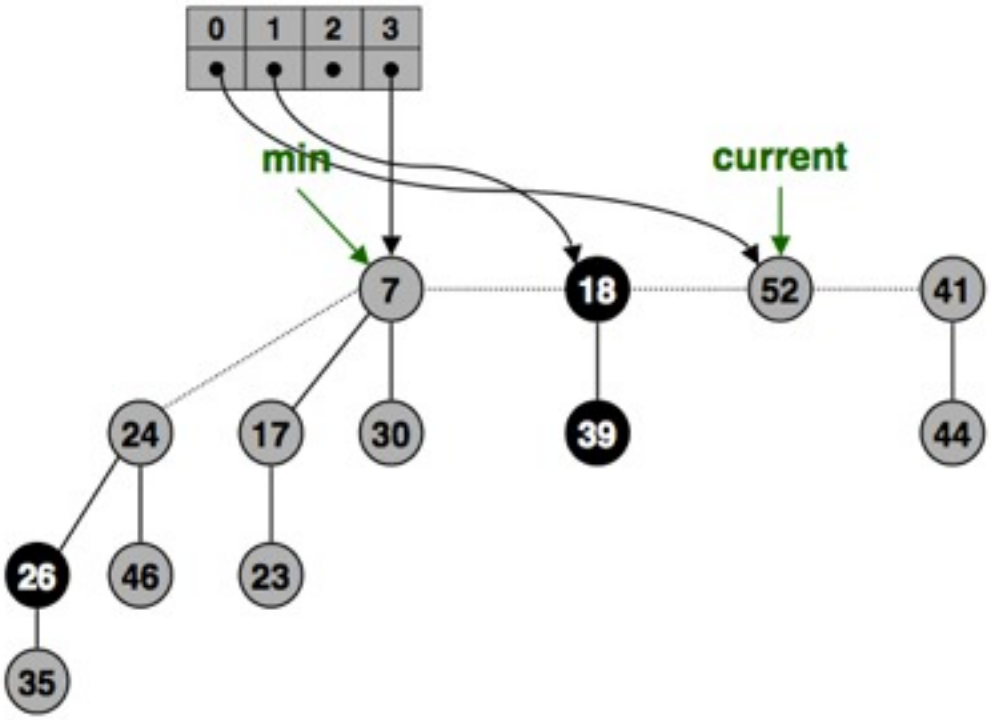
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



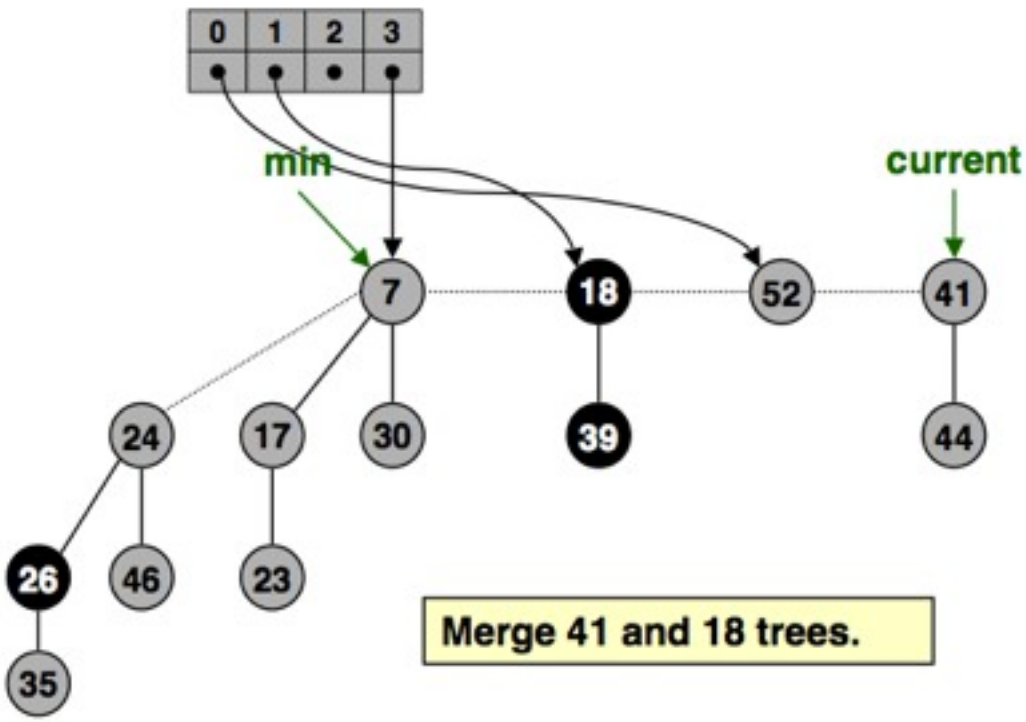
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



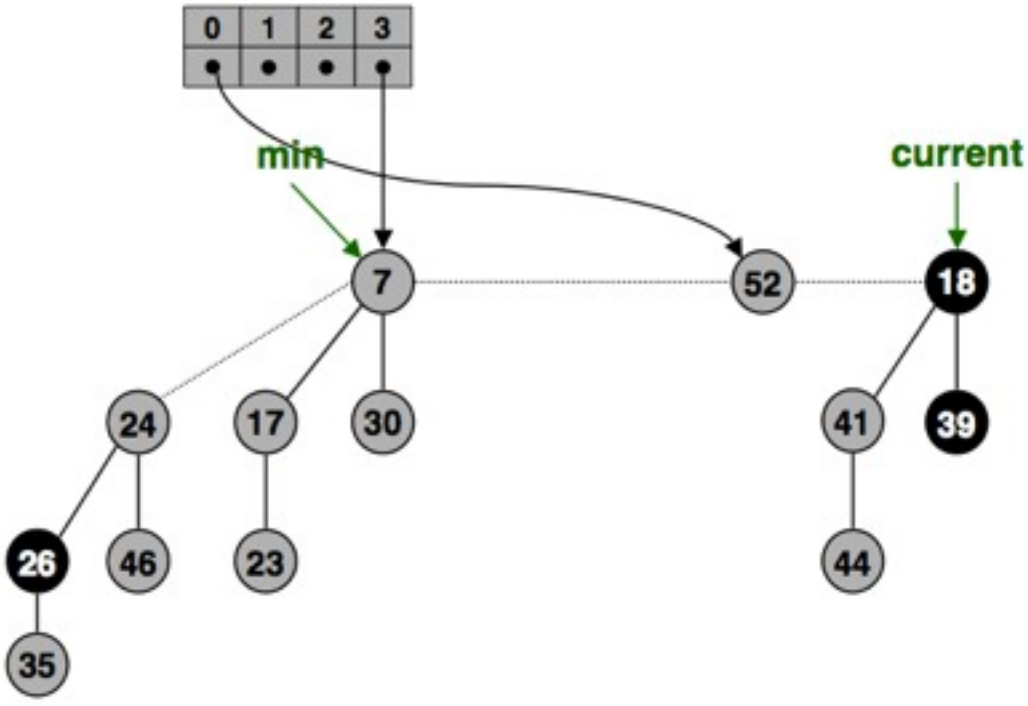
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



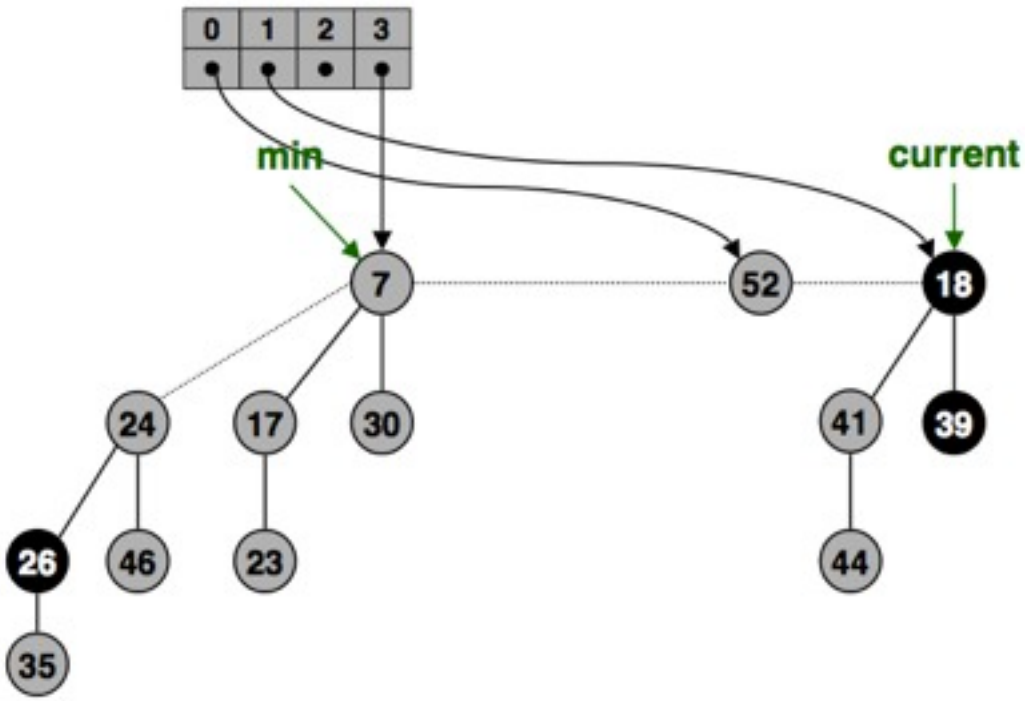
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



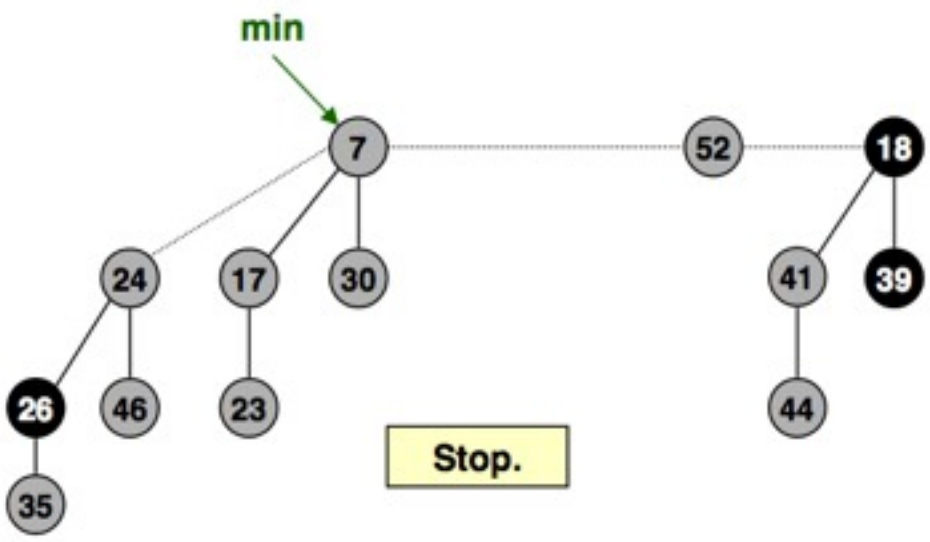
Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



Fibonacci-Heaps: Delete Min

- Lösche Minimum und füge Kinder in Wurzelliste ein.
- Bereinige Bäume, so dass keine zwei denselben Grad haben.



Fibonacci-Heaps: Delete Min

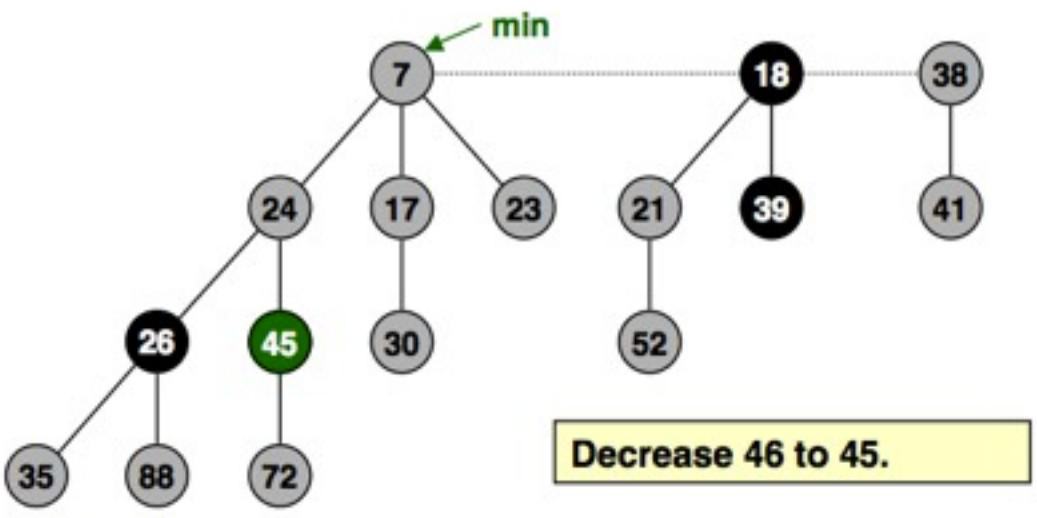
- D(n): Maximalgrad eines Knotens in Fibonacci-Heap mit n Knoten
- t(H): Zahl der Bäume
- P(H)=t(H)+2m(H)

- Kosten: O(D(n)+t(H))
- Amortisierte Kosten: O(D(n))

- Man kann zeigen: D(n)=O(log n)

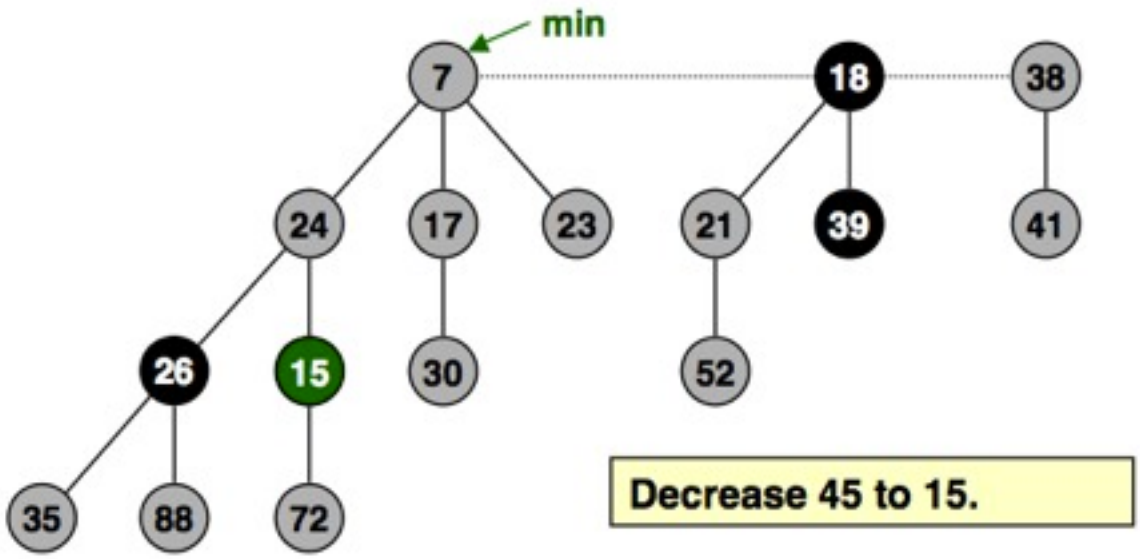
Fibonacci-Heaps: Decrease Key

- Fall 0: Heap-Eigenschaft bleibt erhalten
  - Verringere Wert
  - Aktualisiere ggf. globales Minimum



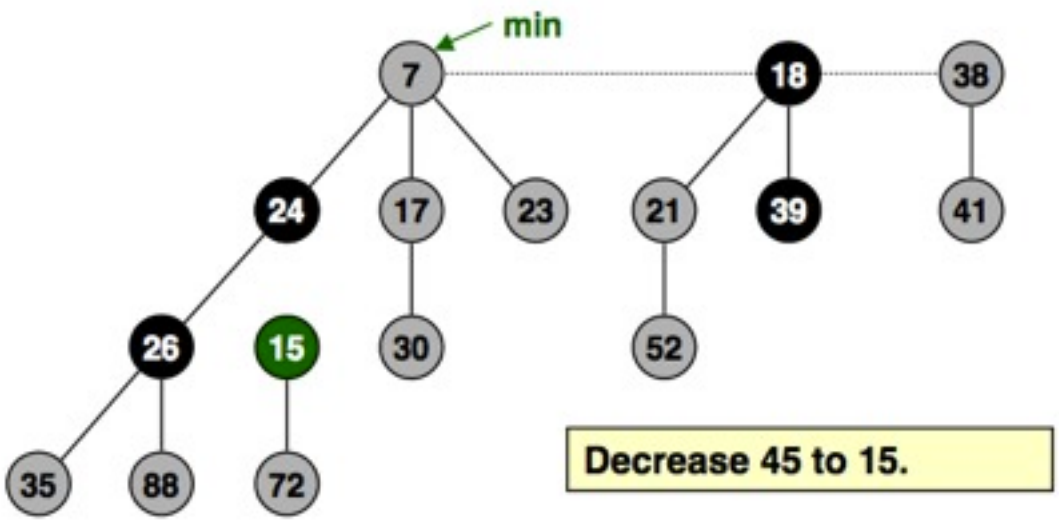
Fibonacci-Heaps: Decrease Key

- Fall 1: Vaterknoten ist unmarkiert.
  - Verringere Wert
  - Lösche Verbindung zum Vaterknoten.
  - Markiere Vaterknoten
  - Füge abgeschnittenen Baum an Wurzel ein.



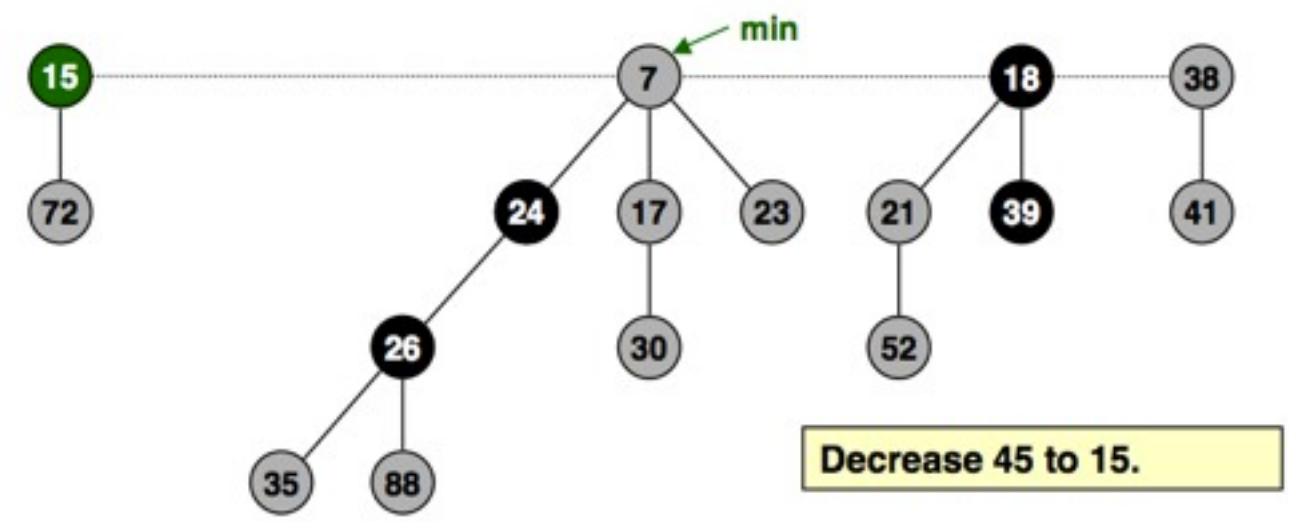
Fibonacci-Heaps: Decrease Key

- Fall 1: Vaterknoten ist unmarkiert.
  - Verringere Wert
  - Lösche Verbindung zum Vaterknoten.
  - Markiere Vaterknoten
  - Füge abgeschnittenen Baum an Wurzel ein.



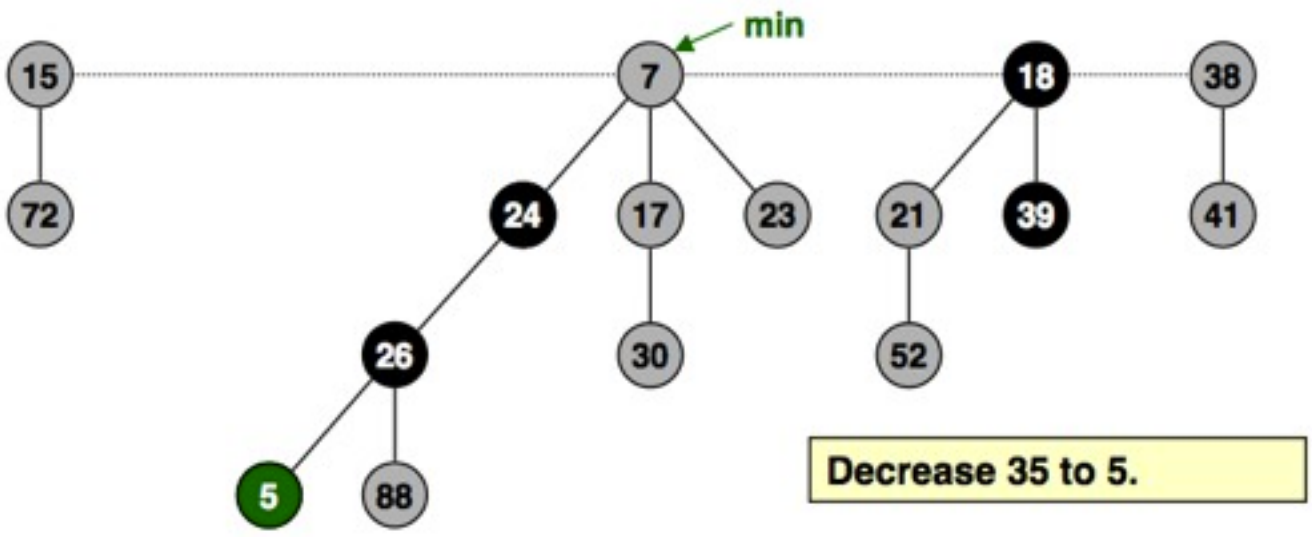
Fibonacci-Heaps: Decrease Key

- Fall 1: Vaterknoten ist unmarkiert.
  - Verringere Wert
  - Lösche Verbindung zum Vaterknoten.
  - Markiere Vaterknoten
  - Füge abgeschnittenen Baum an Wurzel ein.



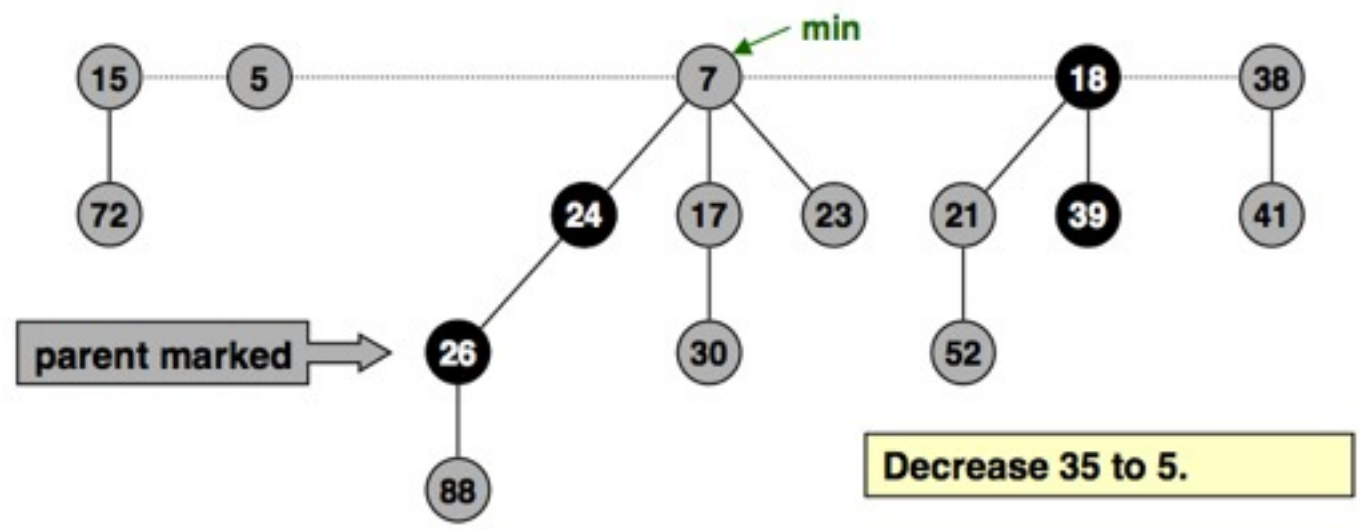
Fibonacci-Heaps: Decrease Key

- Fall 2: Vaterknoten p[x] von x ist markiert.
  - Verringere Wert von x
  - Lösche Verbindung zwischen x und p[x].
  - Füge abgeschnittenen Baum von x an Wurzel ein.
    - Markiere p[p[x]] falls unmarkiert.
    - Schneide p[p[x]] ab falls markiert, entferne Markierung, wiederhole



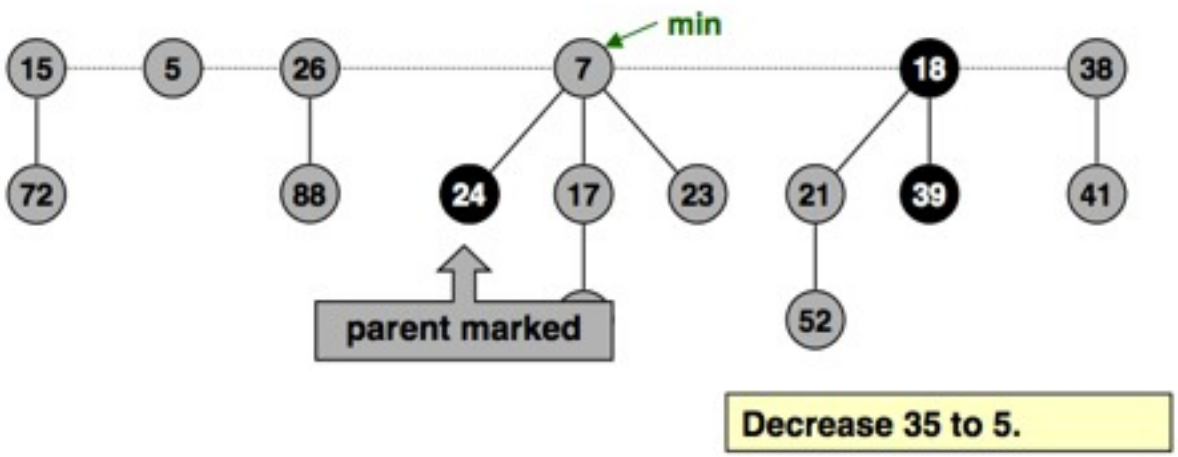
Fibonacci-Heaps: Decrease Key

- Fall 2: Vaterknoten p[x] von x ist markiert.
  - Verringere Wert von x
  - Lösche Verbindung zwischen x und p[x].
  - Füge abgeschnittenen Baum von x an Wurzel ein.
    - Markiere p[p[x]] falls unmarkiert.
    - Schneide p[p[x]] ab falls markiert, entferne Markierung, wiederhole



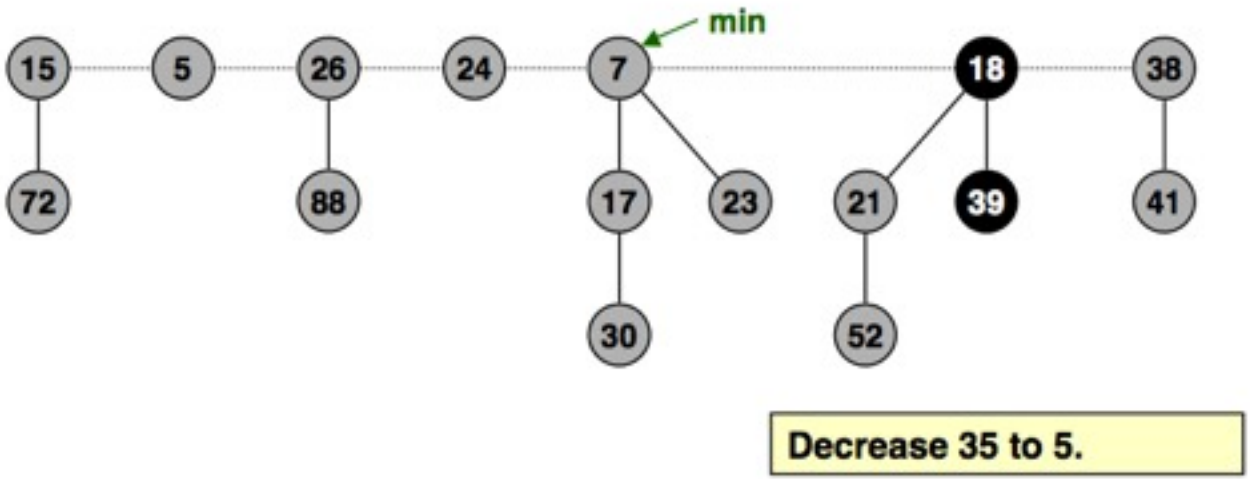
Fibonacci-Heaps: Decrease Key

- Fall 2: Vaterknoten p[x] von x ist markiert.
  - Verringere Wert von x
  - Lösche Verbindung zwischen x und p[x].
  - Füge abgeschnittenen Baum von x an Wurzel ein.
    - Markiere p[p[x]] falls unmarkiert.
    - Schneide p[p[x]] ab falls markiert, entferne Markierung, wiederhole



Fibonacci-Heaps: Decrease Key

- Fall 2: Vaterknoten p[x] von x ist markiert.
  - Verringere Wert von x
  - Lösche Verbindung zwischen x und p[x].
  - Füge abgeschnittenen Baum von x an Wurzel ein.
    - Markiere p[p[x]] falls unmarkiert.
    - Schneide p[p[x]] ab falls markiert, entferne Markierung, wiederhole



Fibonacci-Heaps: Decrease Key

- t(H): Zahl der Bäume
- m(H): Zahl der markierten Knoten
- P(H)=t(H)+2m(H)

Kosten: O(c)

- O(1) für Decrease Key
- O(1) für jede der c Schnittoperationen.

Amortisierte Kosten: O(1)

- t(H')=t(H)+c
- m(H') ≤ m(H)-c+2
- ∂P(H)≤c+2(-c+2)=4-c

Fibonacci-Heaps: Gradbeschränkung

- Man kann zeigen: D(n)=O(log n)

Fibonacci-Heaps: Schlusswort

On Complicated Algorithms

"Once you succeed in writing the programs for [these] complicated algorithms, they usually run extremely fast. The computer doesn't need to understand the algorithm, its task is only to run the programs."

R. E. Tarjan

Mehr in den Literaturverweisen!