

FANTASTISCHE HEAPS

... und wo sie zu finden sind.
Datastructure Love!

Florian Sihler

11. November 2022
SP, Universität Ulm

FANTASTISCHE HEAPS

... und wo sie zu finden sind.
Datastructure Love!

Florian Sihler



Officially supported by the Pingu-Foundation for Emotional Support. There will be light and there will be cute waddlers!

11. November 2022
SP, Universität Ulm

2

Disclaimer

- Dieser „Heap“ hat nichts mit dem aus der Rekursions-Episode zu tun!
- Wir beschränken uns auf Heap-*Bäume*
- Wir ignorieren Rotationen
- Die Folien sind in ~ 3.5 Stunden entstanden^{also reduziert}

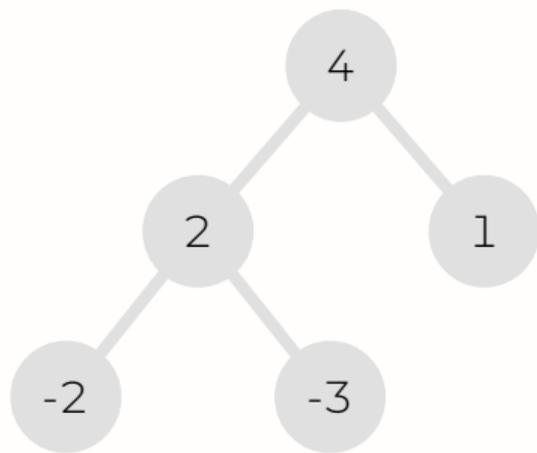
[6]: Episode-Rekursion Sihler, 2021

[5]: EPK-Package, tikzpingus
Sihler, 2021

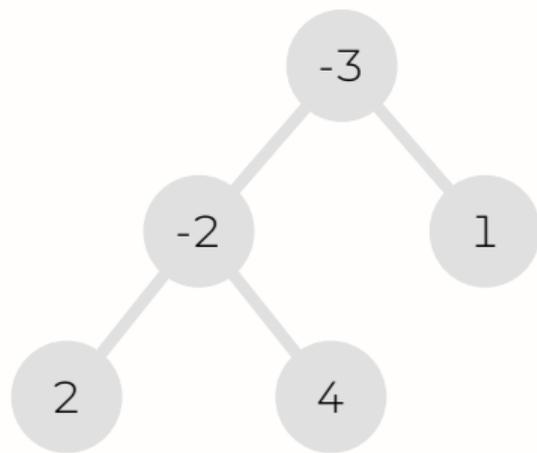


3

Die Muster-Kinder



- **Max-Heap**
Eltern \geq Kinder



- Min-Heap**
Eltern \leq Kinder

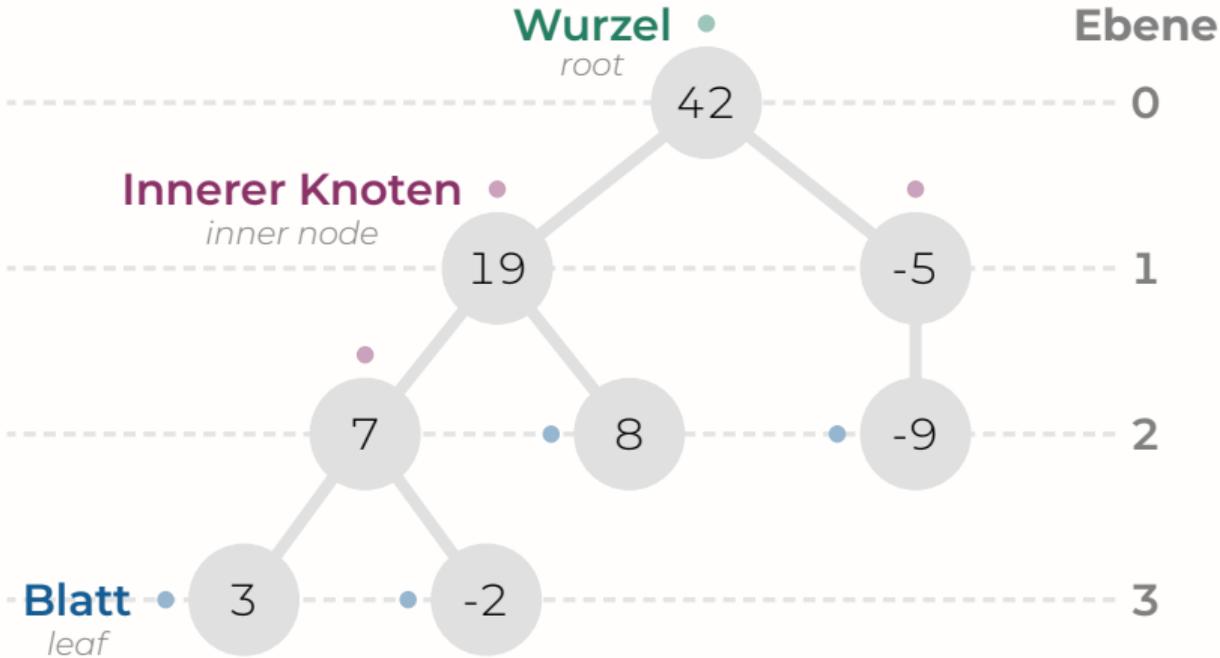


Blätter sind also Knoten ohne weitere Kindknoten.



4

Die Anatomie



Die Linien zwischen den Knoten heißen einfach „Kante“ (edge).





5

Formales

- Für uns ein Baum
 - Graph $G = (V, E)$ mit $|V|$ Knoten und $|E|$ Kanten
 - V und E sind endlich
 - Azyklisch & zusammenhängend („maximal azyklisch“)
- Sogar ein Binärbaum!
 - Jeder Knoten hat maximal zwei Nachfolger
 - Wir konzentrieren uns auf ausgewogene
- Bäume lassen sich als rekursive Struktur auffassen

[6]: *Episode-Rekursion* Sthler, 2021

[2]: *The average height of binary trees and other simple trees*
Flajolet, 1982



6

Heapify



Herstellung der Heap-Eigenschaft durch Vertauschungen!



- *Up-Heapify*: Prüft Blätter → Wurzel
- *Down-Heapify*: Prüft Wurzel → Blätter



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



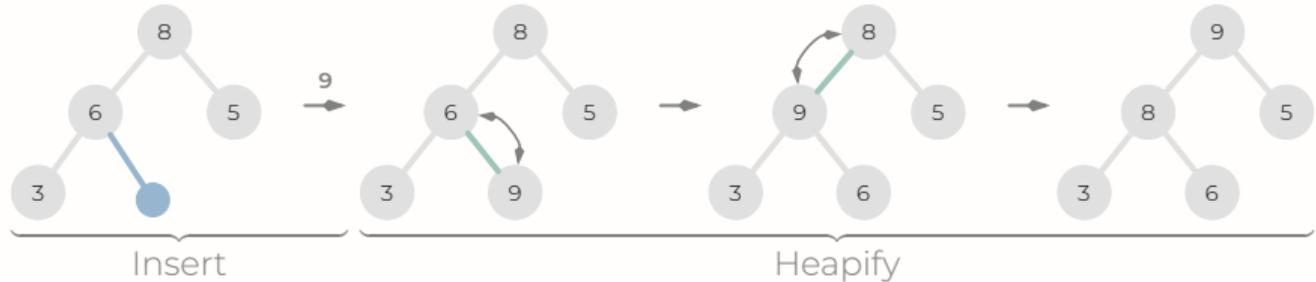
7

Insert/Put

Ist die unterste Ebene voll, so erschaffen wir einfach eine Neue!



- Auf der niedersten Ebene, so weit links wie möglich
- Anschließend: *heapify*



Initiales



Arten von Heaps



Operationen



Arrays



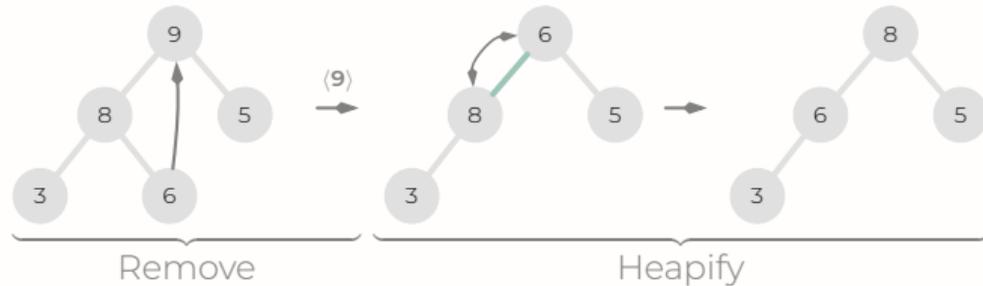
Abschluss



8

Remove/Get

- Ersetze oberstes Element durch „letztes“
- Anschließend: *heapify*



9

Weitere Operationen

- Find/Extract Min/Max
- Replace (one node with another)
- Meld/Join

[3]: *Refined complexity analysis for heap operations* Fredman, 1987

[7]: *Heap - A Data Structure for Efficient Programs* Singh, 2013



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



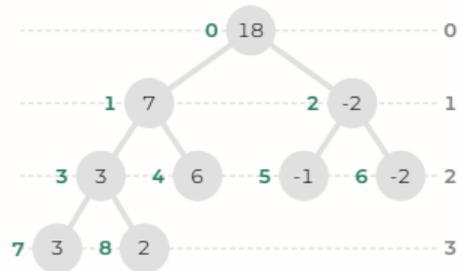
10

Erstaunliche Erkenntnisse

Annahmen:

Heap ist bis auf letzte Ebene komplett gefüllt.
Letzte Ebene füllt sich von links an.

- Jede gefüllte Ebene i enthält 2^i Elemente
- Die Nummerierung nach Breitendurchlauf erlaubt Adressierung!
- Das linke Kind von n ist $2 \cdot n + 1$, das rechte Kind $2 \cdot n + 2$
- Der Elternknoten von n ist $\lfloor \frac{n-1}{2} \rfloor$



Hmmmm... Dat is ne Array-Nummerierung!



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



11

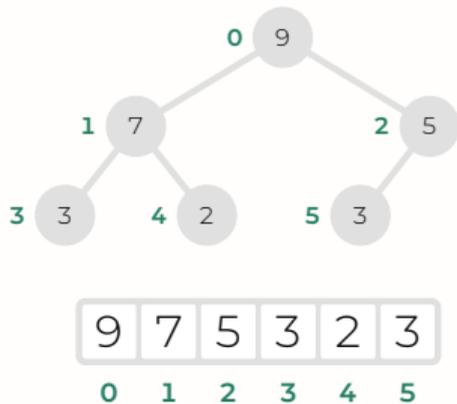
Ein Heap als Array

- Heaps sind meist „nur“ Arrays
- Hilfsroutinen erlauben die angenehme Arbeit:

```
int left(int n) { return 2 * n + 1; }  
int right(int n) { return 2 * n + 2; }  
int parent(int n) { return (n - 1) / 2; }
```

- Checks schaden natürlich nicht!

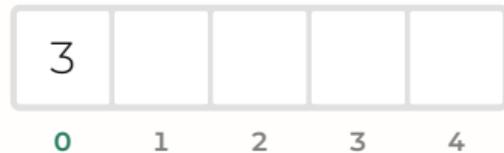
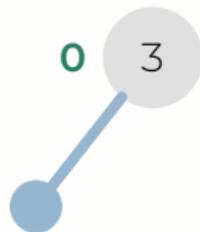
Tipp: Bei Arrays ist meist auch *swap* äußerst hilfreich!



12.1

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



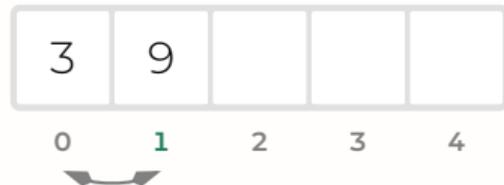
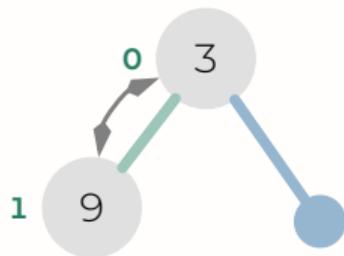
Abschluss



12.2

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



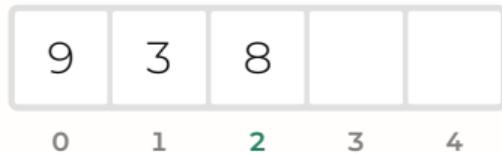
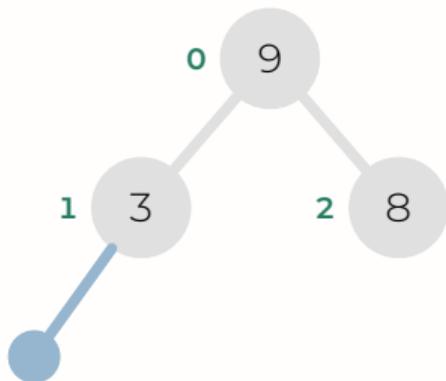
Abschluss



12.3

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



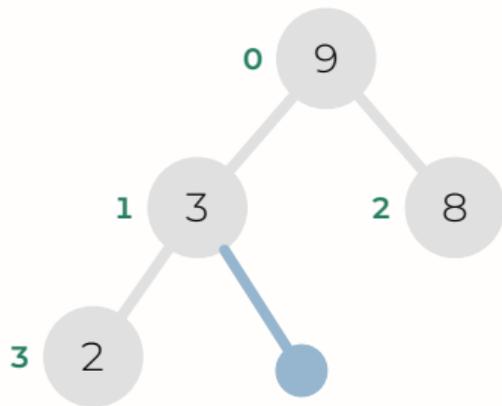
Abschluss



12.4

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



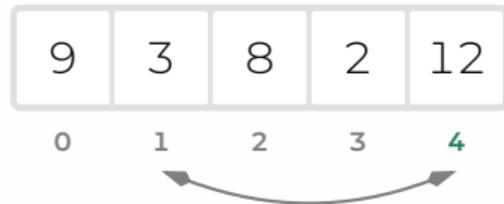
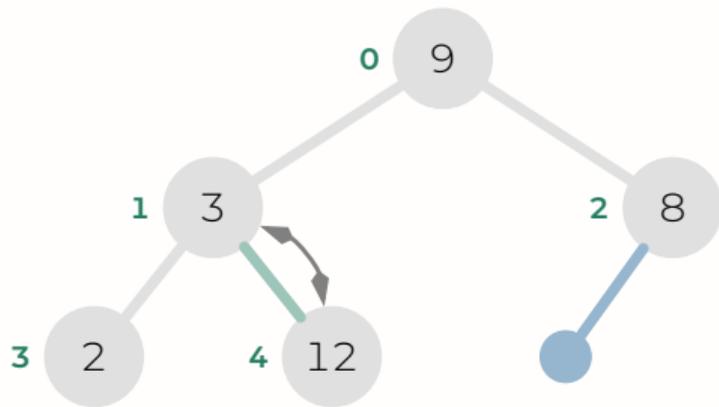
Abschluss



12.5

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



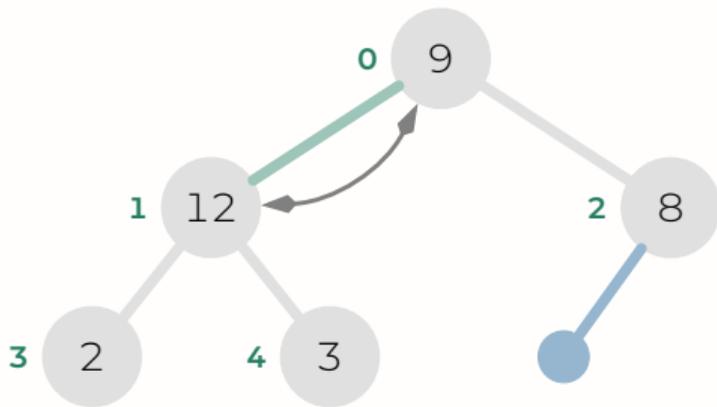
Abschluss



12.6

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



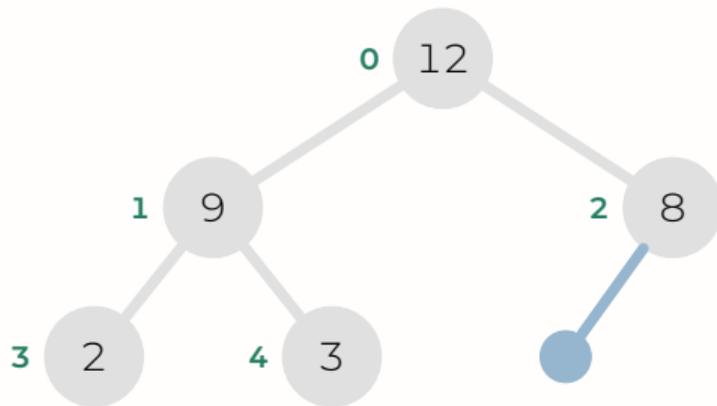
Konstruiert man den Heap nur durch *insert*, verletzt stets max. das neue Element die Heap-Eigenschaft. Weiter ist er schön ausgewogen.



12.7

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



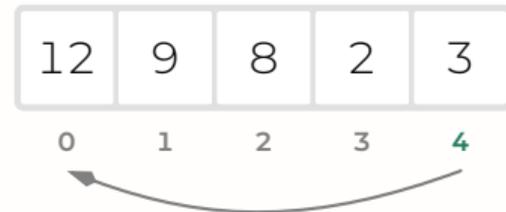
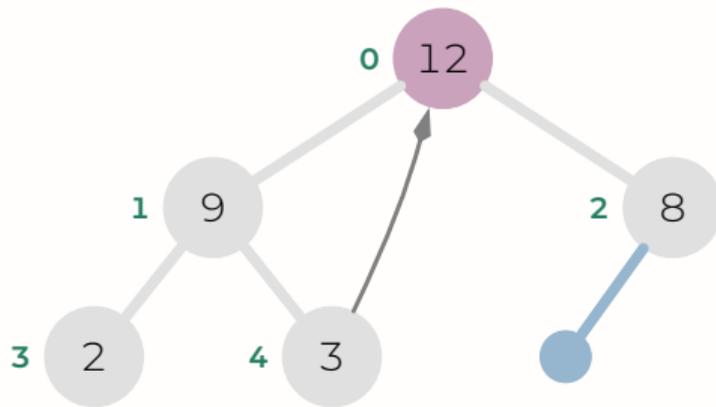


Das Entfernen zur Abwechslung nun mit „down-heapify“.

12.8

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



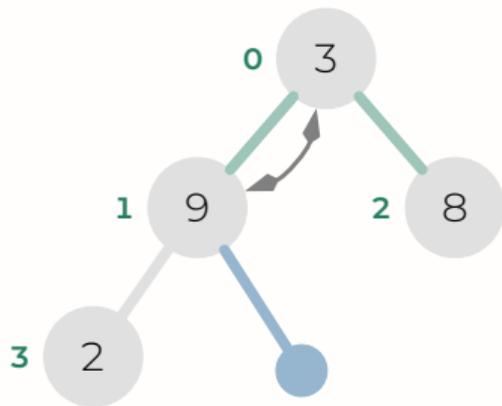
12.9

Ein absteigendes Beispiel

Das Entfernen zur Abwechslung nun mit „down-heapify“.
Bei mehreren Konflikten: tausche mit *extremem* Element.



[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



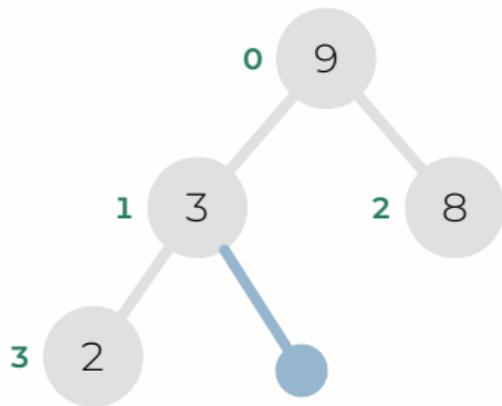


Das Entfernen zur Abwechslung nun mit „down-heapify“.
Bei mehreren Konflikten: tausche mit *extremem* Element.

12.10

Ein absteigendes Beispiel

[3, 9, 8, 2, 12]



Initiales



Arten von Heaps



Operationen



Arrays



Abschluss



13

Zur Übung

- Visualisiere *insert* [3, 5, 8, 3, -5, 6] (in der Reihenfolge) im *Min*-Heap. Am Besten simultan im Baum und im Array.
- Visualisiere nun das Entfernen, bis der Heap leer ist
- Bei diesen und anderen Datenstrukturen hilft die *praktische* Auseinandersetzung!

[4]: Suchbäume
GeeksforGeeks, 2020

[1]: Exercises for Algorithms and
Data Structures Carzaniga, 2016



- [1] Antonio Carzaniga. „Exercises for Algorithms and Data Structures“. 2016
- [2] Philippe Flajolet und Andrew Odlyzko. „The average height of binary trees and other simple trees“. 1982
- [3] Michael L. Fredman und Thomas H. Spencer. „Refined complexity analysis for heap operations“. 1987
- [4] GeeksforGeeks. *Suchbäume*. 2020
- [5] Florian Sihler. *L^AT_EX*-Package, *tikzpingus*. 2021
- [6] Florian Sihler. *Episode-Rekursion*. 2021
- [7] Sanjay Singh. „Heap : A Data Structure for Efficient Programs“. 2013

Florian Sihler

Ulm, 11. November 2022

