

Algorithmen und Datenstrukturen

B1. Heap und Heapsort - Eine informelle Einführung

Marcel Lüthi and Gabriele Röger

Universität Basel

31.03.2021

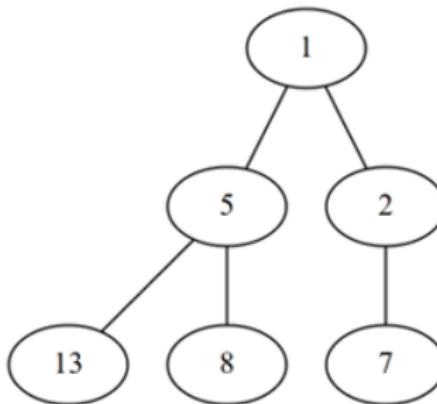
Algorithmen und Datenstrukturen

31.03.2021 — B1. Heap und Heapsort - Eine informelle Einführung

Datenstruktur Heap

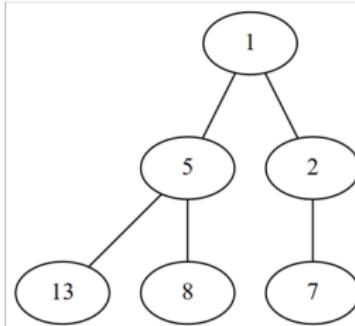
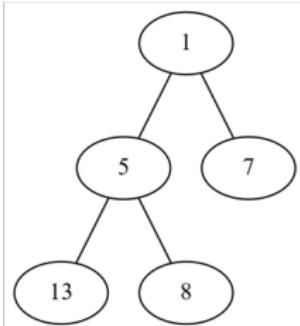
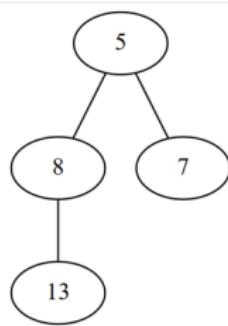
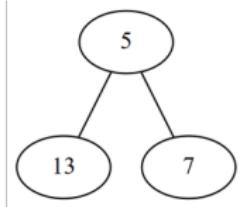
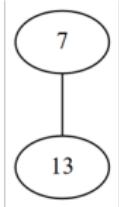
Heap

Ein (binärer) min-Heap ist ein vollständiger binärer Baum, bei dem gilt, dass der Wert in jedem Knoten kleiner gleich dem Wert seiner beiden Kindern (sofern vorhanden) ist.

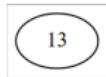
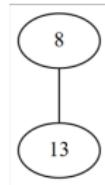
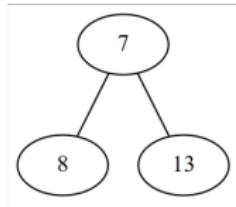
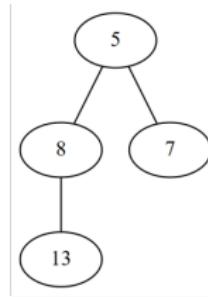
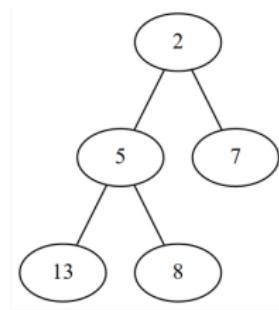
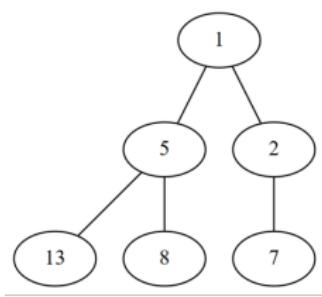


Aufbauen eines Heaps

Elemente: 7, 13, 5, 8, 1, 2



Entfernen des kleinsten Elements vom Heap



Beispiel: Sortieren mit Heaps (Ausblick)

Idee des Algorithmus:

- ▶ Baue Heap aus unsortierter Liste
- ▶ Solange Elemente im Heap sind
 - ▶ Entferne kleinstes Element (Wurzel)
 - ▶ Schreibe Element in (neue) Liste
 - ▶ Stelle Heapbedingung wieder her
- ▶ Neue Liste enthält Elemente in sortierter Reihenfolge

Heapsort: Gleiche Idee, aber inplace.

Sortieren mit Heaps

Offene Fragen:

- ▶ Wie schnell können wir Heap aus n unsortierten Elementen aufbauen?
- ▶ Antwort: Naiv: In $O(n \log_2 n)$ Operationen. Trickreich: In $O(n)$
- ▶ Wie schnell können wir Heapbedingung nach Entfernen wiederherstellen?
- ▶ Antwort: In $O(\log_2 n)$ Operationen
- ▶ Wie gross ist die gesamte Laufzeitkomplexität
- ▶ Antwort: In $O(n \log_2 n)$ Operationen

Komplexität verschoben von Algorithmus nach Datenstruktur

Zusammenfassung

- ▶ Algorithmen und Datenstrukturen arbeiten zusammen
 - ▶ (Teil der) Komplexität kann verschoben werden
- ▶ Datenstrukturen können meist visualisiert/graphisch verstanden werden
- ▶ Oft gilt: Gute Datenstrukturen ⇒ Einfach(ere) Programme

Details von Heapsort folgen ...