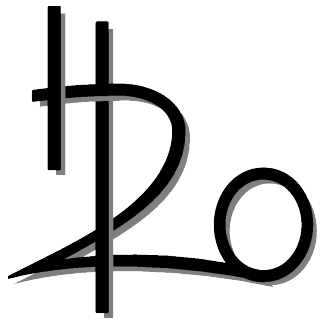
Heapsort



Ralf Dorn

Heinrich-Hertz-Gymnasium

18. Oktober 2018

Heapsort

Prinzip:

Der Heapsort ist ein höheres Sortierverfahren, welches auf oder in dem selben Array (in-place) per Vergleiche sortiert. Er ist schneller als die einfachen Sortierverfahren, und verwendet einen Heap. Man sagt er ist asymptotisch optimal für Sortierverfahren basierend auf Vergleiche.

Warum ist er schneller?

Heapsort

Heap:

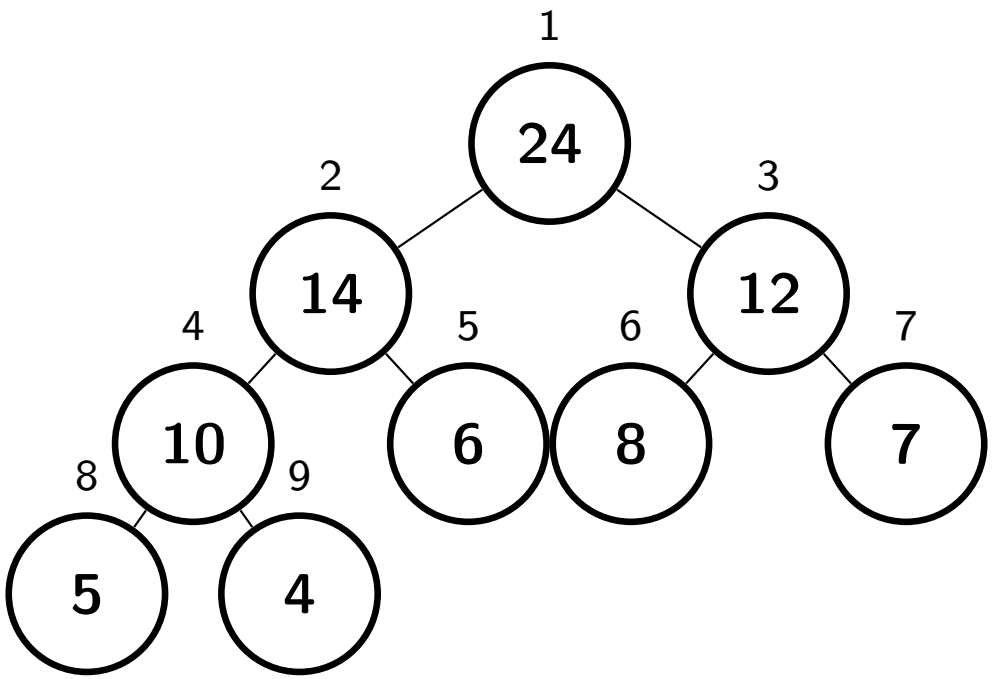
Damit der Heapsort schnell operiert, ist eine baumartige Datenstruktur vorteilhaft.
Eigenschaften:

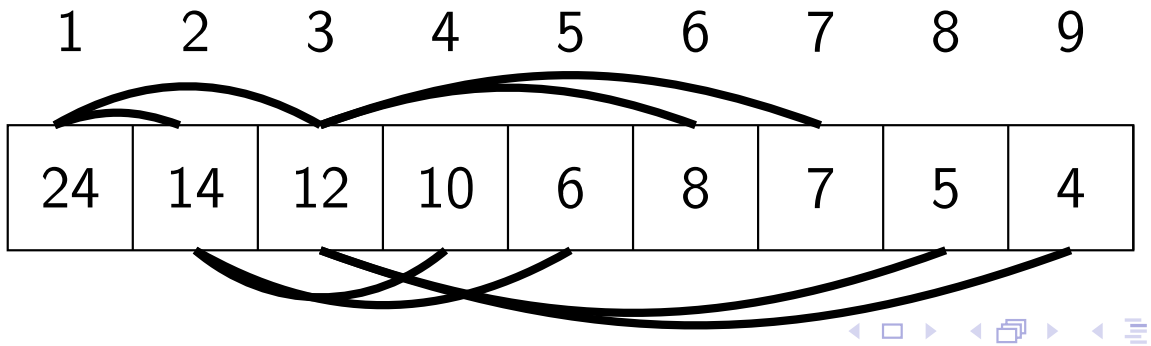
- balancierter Binärbaum (partiell geordnet)
- sortiert (ordinale Schlüsselwerte zu zugehörigen Datenwerten, totale Ordnung)
- vollständiger Binärbaum (alle Ebenen sind komplett gefüllt außer der letzten)
- abstrakt

Heap-Bedingung: (maxHeap) Die Schlüsselwerte der Kinder sind stets kleiner als die der Eltern $\Rightarrow$ es steht in der Wurzel stets das größte Element

Heapsort

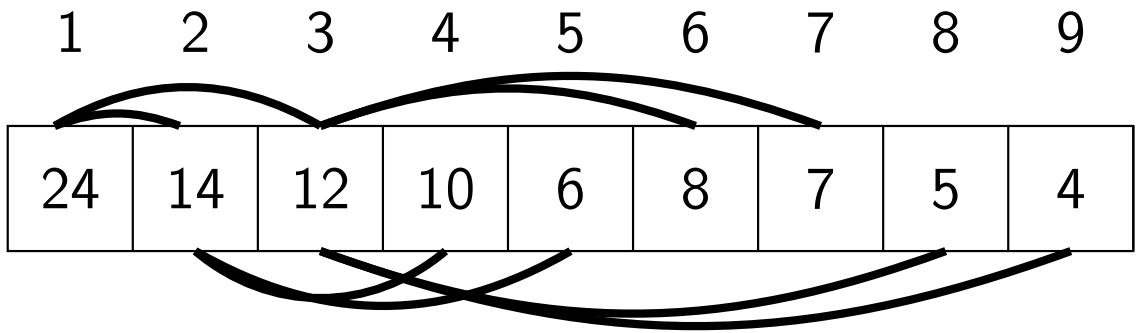
Beispiel für einen Heap (hier: Maxheap)





Heap

Wie werden die Knotenwerte im Array abgelegt?



Parent(i) = return $\left\lfloor \frac{i}{2} \right\rfloor$
Left(i) = return 2·i
Right(i) = return 2i+1

Heapssort

Wie arbeitet man nun mit einem Heap?

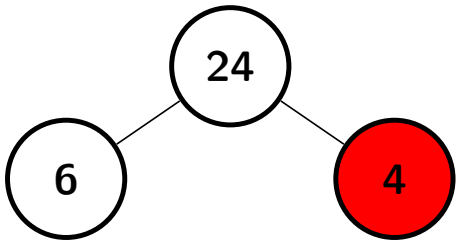
Zuerst muss man überlegen, wie man einen Heap am besten aufbaut.

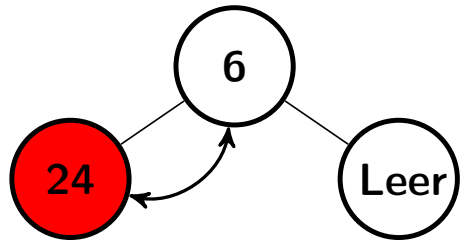
Wenn man also einen Heap vorliegen hat, hat man das größte Element in der Wurzel stehen. Entnimmt man dieses, bleibt die Frage zu klären, wie man die beiden Teilheaps (linker und rechter Teilheap) wieder zu einem gültigen Heap verbindet.

Hat man diese Fragen geklärt, dann beschränkt sich der Heapsort lediglich auf die Entnahme der Wurzeln und das Anwenden der Korrekturen am Heap.

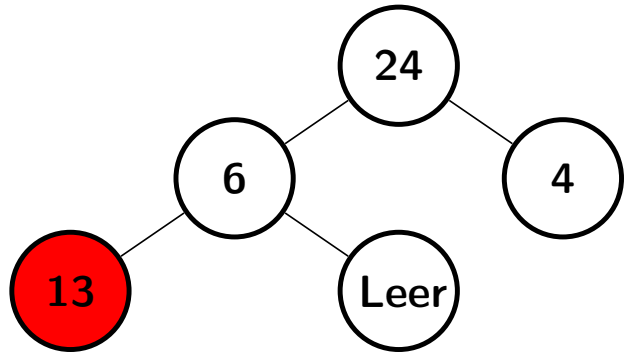
Heapssort

Einfügen in einen Heap: Ist der Heap leer, dann wird das erste Element als Blatt eingefügt. Z.B. ist die Folge 6, 24, 4, 13 einzufügen.



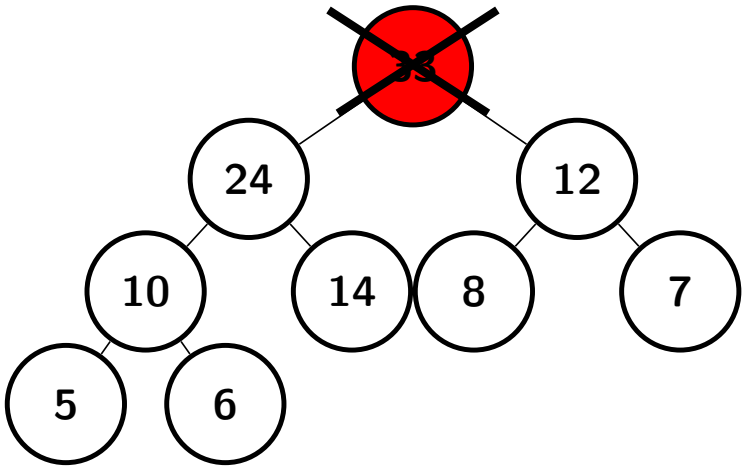


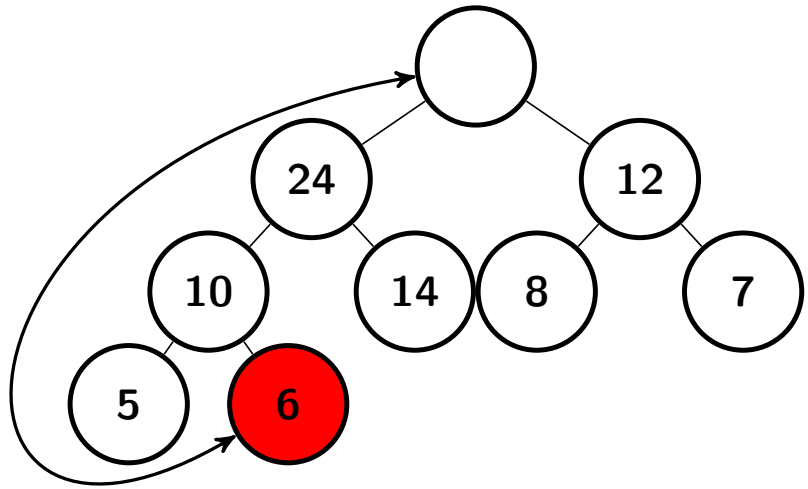
Das neue Element wird an Ende des Baumes eingefügt. Dann wird rekursiv mit dem Parentelement verglichen und ggf. getauscht.



Heapssort

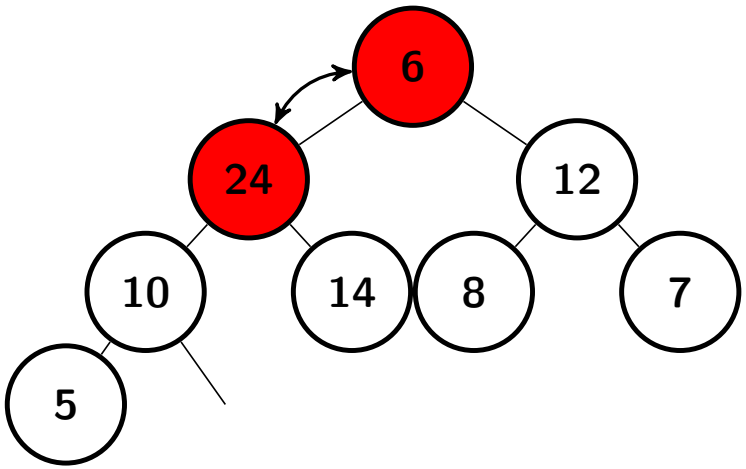
Entnehmen der Wurzel und verbinden beider Teilheaps: Bei Entnahme des maximalen Elementes fügt man das letzte Element vom Heap in die Wurzel ein und lässt es dann rekursiv nach unten sinken.

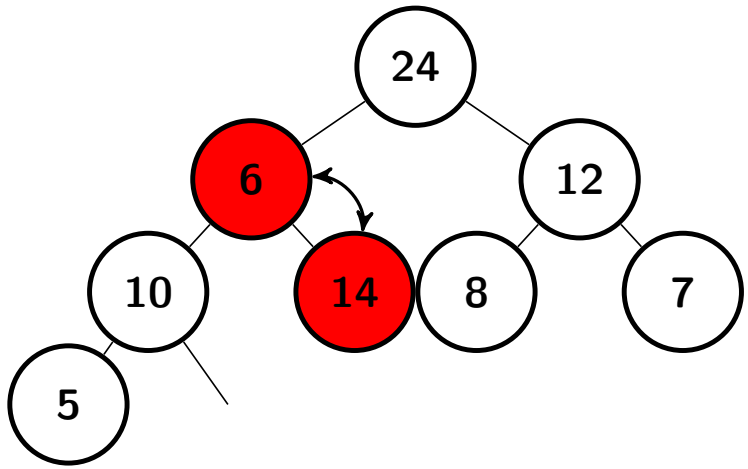




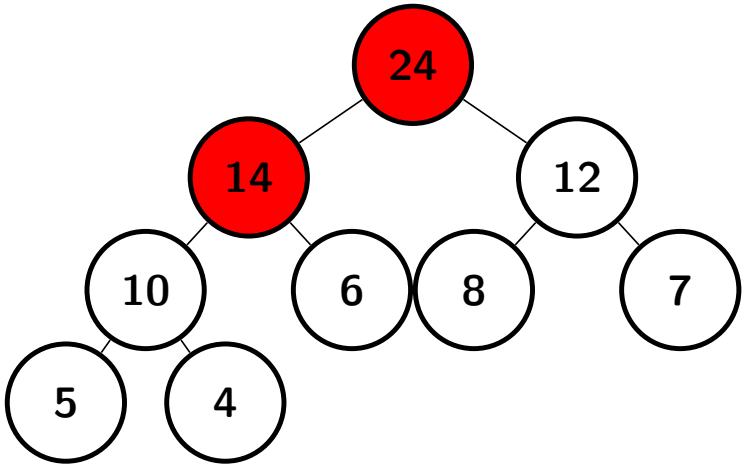
Heapssort

Entnehmen der Wurzel und verbinden beider Teilheaps: Bei Entnahme des maximalen Elementes fügt man das letzte Element vom Heap in die Wurzel ein und lässt es dann rekursiv nach unten sinken.





Heapssort



Man kann nach diesem Prinzip auch Heaps reparieren, wenn sie an irgendeiner Stelle die Heapeigenschaft (Wie lautet diese Invariante?) verletzen.

Aufgabe: Formuliere eine Schnittstelle für die Klasse `Heapsort`!

Heapsort

Schnittstellenbeschreibung:

Wir brauchen

- `insert()` //fügt ein Element am Ende des Heaps ein und stellt die Heapeigenschaft wieder her.
- `deletemax()` //liefert das Wurzelelement und setzt das letzte Element des Heaps in die Wruzel und stellt die Heapeigenschaft wieder her
- `top()` liefert das maximale Element, ohne es zu entfernen
- `heapify()` repariert ggf. den Heap
- `heapsort()` sortiert mithilfe eines Heaps eine Menge von ordinalen Daten
- `build()` baut aus einer Menge von Daten einen Heap auf

Heapsort

Daraus resultieren folgende Signaturen:

- `insert()`
- `deletemax()`
- `top()`
- `heapify()`
- `heapsort()`
- `build()`

$\Rightarrow$ Wird im Unterricht verglichen! Siehe Implementierung !

Graphen

Was ist das Besondere am `Heapsort`?
Oder anders: Wie lautet die Komplexitätsklasse?

`build()` braucht maximal $\mathcal{O}(\log n)$ Schritte

`heapsort()` muss n Elemente aus dem Heap entnehmen, danach muss jedesmal `build()` aufgerufen werden.

Also: $\mathcal{O}(n \log n)$

Besser geht's nicht!