

# Heapsort

## Algorithmen und Datenstrukturen I

---

Prof. Dr. Oliver Braun

Letzte Änderung: 18.11.2019 07:38

- bisherigen Sortierverfahren benötigen im schlechtesten Fall eine Laufzeit von  $\Theta(N^2)$ 
  - alles sog. *allgemeine* Sortierverfahren, die nur den Schlüssel verwenden
- Heapsort (Sortieren mit einer Halde) kommt auch im schlechtesten Fall mit  $O(N \log N)$  Operationen aus

- Heapsort: Prinzip Sortieren durch Auswahl, aber geschickt organisiert
- Heap ist eine Datenstruktur in der die Auswahl des größten Elements in einem Schritt möglich ist
- $F = k_1, k_2, \dots, k_N$  heißt **Heap**, wenn gilt  $k_i \leq k_{\lfloor \frac{i}{2} \rfloor}$  für  $2 \leq i \leq N$  oder anders ausgedrückt:

$$k_i \geq k_{2i} \text{ und } k_i \geq k_{2i+1} \text{ mit } 2i \leq N \text{ und } 2i + 1 \leq N$$

- Beispiel:  $F = 8, 6, 7, 3, 4, 5, 2, 1$  genügt der Heap-Bedingung
- lässt sich anschaulich als Binärbaum darstellen

- haben Heap aus  $N$  Schlüsseln
- sortieren **absteigend**
- solange Heap nicht leer:
  - wähle ersten Schlüssel (= größter)
  - stelle Heap-Bedingung für die restlichen Schlüssel her
- um **aufsteigend** zu sortieren
  - das aus dem Heap entfernte Maximum nach hinten stellen

- nach dem Entfernen des ersten Wertes (Wurzel) haben wir zwei Teil-Heaps
- wir schreiben den Schlüssel mit dem höchsten Index nach vorne (an die Wurzel)
- und lassen ihn nach hinten (unten) versickern
  - wir tauschen immer wieder mit dem **größeren** seiner beiden Nachfolger
  - bis beide Nachfolger kleiner sind
  - oder der Schlüssel ganz unten angekommen ist

```
1 void siftdown(int *a, int i, int m) {
2     while (2 * i + 1 < m) { // hat linken Nachfolger
3         int j = 2 * i + 1; // linker Nachfolger
4         if (j < m - 1) { // hat rechten Nachfolger
5             if (a[j] < a[j + 1])
6                 j = j + 1; // a[j] ist der größere
7         }
8         if (a[i] < a[j]) { // a[i] muss versickern
9             int t = a[i];
10            a[i] = a[j];
11            a[j] = t;
12            i = j; // weiter versickern
13        } else
14            i = m; // fertig
15    }
16 }
```

- Heapsort benötigt die Folge zu Beginn als Heap
- Methode:

$F = k_1, k_2, \dots, k_N$  wird in einen Heap umgewandelt, indem die Schlüssel  $k_{\lfloor \frac{N}{2} \rfloor}, k_{\lfloor \frac{N}{2} \rfloor - 1}, \dots, k_1$  (in dieser Reihenfolge) in  $F$  versickern.

## Heapsort in C++

```
1 void heapsort(int *a, int n) {
2     // stelle Anfangsheap her
3     for (int i = n / 2; i ≥ 0; i--)
4         siftdown(a, i, n);
5     for (int i = n - 1; i > 0; i--) {
6         // tausche a[0] mit a[i], versickere a[0]
7         int t = a[i];
8         a[i] = a[0];
9         a[0] = t;
10        siftdown(a, 0, i);
11    }
12 }
```

- Analyse: siehe Ottmann & Widmayer: Algorithmen und Datenstrukturen, S. 110, 112