1 Invariante

- Vorm ersten Durchlauf erfllt sein (Initialisierung, Induktionsanfang)
- Muss bei jedem Schleifendurchlauf erhalten bleiben (Erhaltng, Induktionsschritt)
- Invar nach Beendigung der Schleife zeigt Korrektheit (Terminierung)

1.1 Rekursionsgleichungen

- ineinander einsetzen
- Summe erkennen und zusammengefasst aufschreiben

1.2 Mastertheorem

1.2.1 Additives Mastertheorem

- a, b, c *positiv* $n = b^k$
- $T(n) \leq \begin{cases} c & n = 1 \\ a \cdot T(\frac{n}{b}) + c & n > 1 \end{cases}$

$$\begin{array}{lll} T(n) \leq c \cdot \frac{a}{a-1} n^{log_b(a)} - \frac{c}{a-1} & = O(n^{log_b(a)}) & falls\ a > 1 \\ T(n) \leq c \cdot \frac{a}{a-1} n - \frac{c}{a-1} & = O(n) & falls\ a = b > 1 \\ T(n) \leq c \cdot log_b(n) + c & = O(log(n)) & falls\ a = 1 \end{array}$$

1.2.2 allgemeines Mastertheorem

- $a, b, d, q \geq 1$
- $T(n) \leq \begin{cases} d & n \leq q \\ a \cdot T(\frac{n}{b}) + f(n) & n > q \end{cases}$

$$\begin{array}{ll} f(n) = O(n^{log_b(a)-\epsilon})\ mit\ \epsilon > 0 & T(n) = O(n^{log_b(a)}) \\ f(n) = \Theta(n^{log_b(a)}) & T(n) = O(n^{log_b(a)} log(n)) \\ f(n) = \Omega(n^{log_b(a)+\epsilon})\ mit\ \epsilon > 0, & T(n) = \Theta(f(n)) \\ a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n), \delta < 1, n \to \infty & \end{array}$$

2 O-Notation

2.1 Definition

- $f = O(g) \Leftrightarrow \exists c > 0 \exists n_0 \in N\ \forall n \geq n_0 : f(n) \leq cg(n)$
  (f wchst asymptotisch hchstens so schnell wie g)
- $f = \Omega(g) \Leftrightarrow g = O(f)$
  (f wachst asymptotisch mindestens so schnell wie g)

- f = $\Theta(g) \Leftrightarrow f = O(g)\ und\ g = O(f)$
  (f und g wachsen asymptotisch gleich schnell)
- $f = o(g) \Leftrightarrow \forall c > 0\ \exists n_0 \in N \forall n \geq n_0 : f(n) < cg(n)$
  $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
  (f wchst asymptotisch langsamer als g)
- $f = \omega(g) \Leftrightarrow g = o(f)$
  (f wchst asymptotisch langsamer als g)

2.2 Eigenschaften

- $f = o(g) \Rightarrow f = O(g)$
- $f = o(g)\ und\ h = o(g) \Rightarrow f + h = o(g)(auch O, \Omega, \omega, \Theta)$
- $f = o(g)\ und\ h = o(g) \Rightarrow f \cdot h = o(g^2)(auch O, \Omega, \omega, \Theta)$

2.3 Reihenfolge

$c < log(n) < n^{\frac{1}{k}} < n < nlog(n) < n^2 < n^k < 2^n$

3 Sortieralgorithmen

3.1 Bubblesort

- Jeden Durchlauf wird das grte Element auf die n-ite Stelle getauscht. Jeder Vergleich ggf ein Swap.
- inkrementelle, inplace, stabil
- $BC : \Theta(n)\ AV : \Theta(n^2)\ WC : \Theta(n^2)$

3.2 Insertionsort

- Key wird in sortiertes Array eingeordnet. Key wird gemerkt, falls kleiner wird das gßere Element auf Pos von Key kopiert aber Key wird erst kopiert, wenn die richtige Stelle gefunden worden ist.
- inkrementelle, inplace, stabil
- $BC : \Theta(n)\ AV : \Theta(n^2)\ WC : \Theta(n^2)$

3.3 Mergesort

- Teile Array bis auf ein Element Array und sortiere beim rekursiven zusammenfgen. Geteilt wird p bis q, q+1 bis r.
  Merge vergleicht erste Pos von den Arrays und fgt immer das kleinere ins Zielarray ein.
- D&C,
- $BC : \Theta(nlog(n))\ AV : \Theta(nlog(n))\ WC : \Theta(nlog(n))$

3.4 Quicksort(A,p,r)

- Teile Array nach Pivotelement und fge Pivot dann an Grenze ein. Dann partition bis q-1 und
- D&C,
- $BC : \Theta(nlog(n))\ AV : \Theta(nlog(n))\ WC : \Theta(n^2))$

4 Heap

4.1 Definition

$Heap := A[i] \geq A[2i] und A[i] \geq A[2i+1]$
jeder Knoten hat mindestens so groen Wert wie seine Kinder
Wird als binrer Baum dargestellt.

4.2 Eigenschafen

- Baumtiefe $\lfloor log(n) \rfloor$
- $A[\lfloor \frac{n}{2} \rfloor + 1]\ bis\ A[n]$ sind Kinder

5 Rechentricks

- Arithmetische Reihe $\sum_{i=1}^{n} k = \frac{n(n+1)}{2}$
- logarithmus!!! bsp laufzeit aus probe 1