DHBW Mannhheim

2. Semester Cyber Security

Algorithmen und Komplexität

N.W. & J.T

13. Juni 2020

Aufgabe Nr. 1

Eigenschaften der Groß-O-Notation

1. Geben Sie die Definition der $\mathcal{O}$-Notation an.
2. Es sei $f \in \mathcal{O}(h_1)$ und $g \in \mathcal{O}(h_2)$. Zeigen Sie, dass $f \cdot g \in \mathcal{O}(h_1 \cdot h_2)$ gilt.
3. Beweisen Sie, dass für alle $f : \mathbb{N} \to \mathbb{R}_+$ gilt, dass $f \in \mathcal{O}(f)$.
4. Beweisen Sie, dass für $f,\ g : \mathbb{N} \to \mathbb{R}_+$ und $d \in \mathbb{R}_+$ gilt, dass
   $g \in \mathcal{O}(f) \to d \cdot g \in \mathcal{O}(f)$.
5. Beweisen Sie, dass für $f,\ g,\ h : \mathbb{N} \to \mathbb{R}_+$ gilt, dass
   $f \in \mathcal{O}(h) \ \wedge g \in \mathcal{O}(h) \to f + g \in \mathcal{O}(h)$.
6. Beweisen Sie, dass für $f,\ g,\ h : \mathbb{N} \to \mathbb{R}_+$ gilt, dass
   $f \in \mathcal{O}(g) \ \wedge g \in \mathcal{O}(h) \to f \in \mathcal{O}(h)$.
7. Angenommen $f,\ g,\ h : \mathbb{N} \to \mathbb{R}_+$. Außerdem wird angenommen, dass der Grenzwert von
   $$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$
   existiert. Beweisen sie, dass dann auch $f \in \mathcal{O}(g)$ gilt.
8. Es seien $f,\ g \in \mathbb{R}_+$. Geben Sie die Definition $f \sim g$ an.

Aufgabe Nr. 1

1) $O(g) := \{ f \in \mathbb{R}_+^{\mathbb{N}} \mid \exists k \in \mathbb{N} : \exists c \in \mathbb{R}_+ : \forall n \in \mathbb{N} : (n \geq k \to f(n) \leq c \cdot g(n) \}$

2) $f \in O(h_1) \wedge g \in O(h_2) \qquad f \cdot g \in O(h_1 \cdot h_2)$

$k_1, k_2 \in \mathbb{N} \qquad c_1, c_2 \in \mathbb{R}_+ \qquad n \in \mathbb{N}$

$\forall n \in \mathbb{N} : (n \geq k_1 \to f(n) \leq c_1 \cdot h_1(n))$
$\forall n \in \mathbb{N} : (n \geq k_2 \to g(n) \leq c_2 \cdot h_2(n))$

$k := \max(k_1, k_2) \qquad c := c_1 \cdot c_2$

$f(n) \cdot g(n) \leq c_1 \cdot h_1(n) \cdot c_2 \cdot h_2(n)$

$\Leftrightarrow f(n) \cdot g(n) \leq c_1 \cdot c_2 \cdot h_1(n) \cdot h_2(n)$

$\Leftrightarrow f(n) \cdot g(n) \leq c \cdot h_1(n) \cdot h_2(n)$

$\Rightarrow f \cdot g \in O(h_1 \cdot h_2)$

$\square$

3) $f \in O(f)$

Definiere $k := 0 \qquad c := 1$

$\forall n \in \mathbb{N} : (f(n) \leq c \cdot f(n))$

$\square$

4) $g \in O(f) \to \alpha \cdot g \in O(f)$

$k' \in \mathbb{N} \qquad c' \in \mathbb{R}$

$\forall n \in \mathbb{N} : (n \geq k' \to g(n) \leq c' \cdot f(n))$ | Ungleichung $\cdot \alpha$

$\forall n \in \mathbb{N} : (n \geq k' \to \alpha \cdot g(n) \leq c' \cdot \alpha \cdot f(n))$

$k := k' \qquad c := \alpha \cdot c'$

$\forall n \in \mathbb{N} : (n \geq k \to \alpha \cdot g(n) \leq c \cdot f(n))$

$\Rightarrow \alpha \cdot g \in O(f)$

$\square$

5) $f \in O(h) \wedge g \in O(h) \to f + g \in O(h)$

$k_1, k_2 \in \mathbb{N} \qquad c_1, c_2 \in \mathbb{R}_+$

$\forall n \in \mathbb{N} : (n \geq k_1 \to f(n) \leq c_1 \cdot h(n))$
$\forall n \in \mathbb{N} : (n \geq k_2 \to g(n) \leq c_2 \cdot h(n))$

$k := \max(k_1, k_2) \qquad c := c_1 + c_2$

$f(n) + g(n) \leq c_1 \cdot h(n) + c_2 \cdot h(n)$

$f(n) + g(n) \leq (c_1 + c_2) \cdot h(n)$

$f(n) + g(n) \leq c \cdot h(n)$

$\Rightarrow f + g \in O(h)$

6) $f \in O(g) \wedge g \in O(h) \to f \in O(h)$

$k_1, k_2 \in \mathbb{N} \qquad c_1, c_2 \in \mathbb{R}_+$

$\forall n \in \mathbb{N} : (n \geq k_1 \to f(n) \leq c_1 \cdot g(n))$
$\forall n \in \mathbb{N} : (n \geq k_2 \to g(n) \leq c_2 \cdot h(n))$

$f(n) \leq c_1 \cdot g(n) \qquad g(n) \leq c_2 \cdot h(n) \quad | \cdot c_1$

$\Leftrightarrow c_1 \cdot g(n) \leq c_2 \cdot c_1 \cdot h(n)$

$k := \max(k_1, k_2) \qquad c := c_1 \cdot c_2,$

$f(n) \leq c_1 \cdot g(n) \qquad c_1 \cdot g(n) \leq c \cdot h(n)$

$f(n) \leq c \cdot h(n)$

$\Leftrightarrow \quad f \in O(h)$ ▫

7) Siehe Skript S. 15

8) $f \sim g$ gilt gdw.

$\frac{f(x)}{g(x} = 1$ ist ▫

Aufgabe Nr. 2

Groß-O-Notation

1. Zeigen Sie, dass $n^2 \in \mathcal{O}(2^n)$ ist.
2. Zeigen Sie auch, dass $n^3\mathcal{O}(2^n)$ gilt.
3. Zeigen Sie: $\log_2(n) \in \mathcal{O}(\ln(n+1))$
4. Zeigen Sie, dass $\ln^2(n) \in \mathcal{O}(\sqrt{n})$ gilt.
5. Versuchen Sie zu zeigen, dass $n^\alpha \in \mathcal{O}(2^n)$, wenn angenommen werden kann, dass $\alpha \in \mathbb{N}$ vorausgesetzt ist.

Aufgabe Nr. 2

1) $n^2 \in O(2^n)$

$$\frac{n^2}{2^n} \overset{\text{L'hôpital}}{=} \frac{2n}{\ln(2)\cdot 2^n} \overset{LH}{=} \frac{2}{\ln(2)^2\cdot 2^n}$$

$$\lim_{n\to\infty} \frac{2}{\ln(2)^2\cdot 2^n} = 0 \quad \square$$

Siehe S. 15 im Skript für den genauen Beweis.

2) $n^3 \in O(2^n)$

$$\frac{n^3}{2^n} \overset{LH}{=} \frac{3n^2}{\ln(2)\cdot 2^n} \overset{LH}{=} \frac{6n}{\ln(2)^2\cdot 2^n} \overset{LH}{=} \frac{6}{\ln(2)^3\cdot 2^n}$$

$$\lim_{n\to\infty} \frac{6}{\ln(2)^3\cdot 2^n} = 0 \quad \square$$

3) $\log_2(n) \in O(\ln(n+1))$

Konstanter Faktor!

$$\ln(n) = \frac{\log_2(n)}{\log_2(e)} \quad \rightarrow \quad \log_2(e)\cdot \ln(n) = \log_2(n)$$

$$\frac{\ln(n)}{\ln(n+1)} \quad \Rightarrow \text{z.z.} \quad \ln(n) \in O(\ln(n+1))$$

$$\lim_{n\to\infty} \frac{\ln(n)}{\ln(n+1)} = 1 \quad \square$$

Somit existiert der Grenzwert und die Aussage ist bewiesen

4) $\ln^2(n) \in O(\sqrt{n})$

$$\frac{\ln^2(n)}{\sqrt{n}} \overset{LH}{=} \frac{\frac{2\cdot\ln(n)}{n}}{\frac{1}{2\sqrt{n}}} = \frac{2\cdot\ln(n)}{n}\cdot\frac{2\sqrt{n}}{1} = \frac{2\cdot\ln(n)\cdot 2\sqrt{n}}{n}$$

$$= 2\cdot\ln(n)\cdot 2\cdot\sqrt{n}\cdot\frac{1}{n}$$

$$= 4\cdot\ln(n)\cdot\frac{\sqrt{n}}{n}$$

$$= 4\cdot\ln(n)\cdot\frac{1}{\sqrt{n}}$$

$$= \frac{4\cdot\ln(n)}{\sqrt{n}}$$

$$\overset{LH}{=} \frac{\frac{4}{n}}{\frac{1}{2\sqrt{n}}}$$

$$= \frac{4}{n}\cdot\frac{2\sqrt{n}}{1}$$

$$= \frac{8\cdot\sqrt{n}}{n}$$

$$= 8\cdot\frac{1}{\sqrt{n}}$$

$$\frac{8}{\sqrt{n}}$$

$$\lim_{n\to\infty} \frac{8}{\sqrt{n}} = 0 \quad \square$$

5) $a \in \mathbb{N}$

$$n^a \in O(2^n)$$

$$f(n) = n^a$$

$$f^{(1)}(n) = a \cdot n^{a-1}$$

$$f^{(2)}(n) = a \cdot (a-1) \cdot n^{a-2}$$

$$\vdots$$

$$f^{(a)}(n) \quad \prod_{i=0}^{a-1} a-i$$

$$f^{(a+1)}(n) = 0$$

$\square$

Aufgabe Nr. 3

Rekurrenzgleichungen

Lösen Sie folgende Rekurrenzgleichungen:

1. $x_{n+2} = x_{n+1} + x_n$ für welche gilt: $x_0 = 0$ und $x_1 = 1$
2. $x_{n+2} = 4 \cdot x_{n+1} - 4 \cdot x_n + 1$ für welche gilt: $x_0 = 1$ und $x_1 = 3$
3. $a_{n+2} = \frac{1}{6} \cdot a_{n+1} + \frac{1}{6} \cdot a_n$ für welche gilt: $a_0 = 0$ und $a_1 = \frac{5}{6}$
4. $a_{n+2} = -\frac{1}{2} \cdot a_{n+1} + \frac{1}{2} \cdot a_n$ für welche gilt: $a_0 = 2$ und $a_1 = 1$
5. $a_{n+2} = a_{n+1} + 2 \cdot a_n + 1$ für welche gilt: $a_0 = 0$ und $a_1 = -\frac{1}{2}$
6. $a_{n+2} = a_n + 2$ für welche gilt: $a_0 = 2$ und $a_1 = 1$
7. $a_{n+2} = 2 \cdot a_n - a_{n+1}$ für welche gilt: $a_0 = 0$ und $a_0 = 3$
8. $a_{n+2} = 7 \cdot a_{n+1} - 10 \cdot a_n$ für welche gilt: $a_0 = 0$ und $a_0 = 3$
9. $a_{n+1} = 2^n \cdot a_n$ für welche gilt: $a_1 = 1$
10. Stellen Sie mit dem Ansatz $a_k := f(2^k)$ eine Rekurrenzgleichung auf und lösen Sie diese.
    $f(n) = 2 \cdot f(n \backslash 2) + \log_2(n)$
    Es gelten folgende Anfangsbedingungen: $x_0 = 0$ und $x_1 = 1$

Aufgabe Nr. 3

1) $x_{n+2} = x_{n+1} + x_n \qquad x_0 = 0 \quad x_1 = 1$

Homogener Teil

$x_n = \lambda^n$

$$\lambda^{n+2} = \lambda^{n+1} + \lambda^n \quad | :\lambda^n$$

$$\lambda^2 = \lambda + 1 \quad | -\lambda^2 | \cdot (-1)$$

$$0 = \lambda^2 - \lambda - 1 \quad | PQ$$

$$\lambda_{1/2} = \frac{1}{2} \pm \sqrt{\frac{1}{2}^2 + 1}$$

$$= \frac{1}{2} \pm \sqrt{\frac{5}{4}}$$

$$= \frac{1}{2} \pm \frac{\sqrt{5}}{2}$$

$$\lambda_1 = \frac{1+\sqrt{5}}{2} \qquad \lambda_2 = \frac{1-\sqrt{5}}{2}$$

$$x_n = \alpha \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n + \beta \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n$$

Inhomogener Teil liegt nicht vor -> wird ausgelassen

LGS mit den Anfangsbedingungen

$$x_0 = 0 = \alpha + \beta$$

$$x_1 = 1 = \alpha \cdot \frac{1+\sqrt{5}}{2} + \beta \cdot \frac{1-\sqrt{5}}{2}$$

$$\alpha = -\beta$$

$$1 = -\beta \cdot \frac{1+\sqrt{5}}{2} + \beta \cdot \frac{1-\sqrt{5}}{2}$$

$$1 = -\beta \cdot \left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right) + \beta \cdot \left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)$$

$$1 = \cancel{-\frac{1}{2}\beta} - \frac{\sqrt{5}}{2}\beta + \cancel{\frac{1}{2}\beta} - \frac{\sqrt{5}}{2}\beta$$

$$1 = 2 \cdot \left(-\frac{\sqrt{5}}{2}\right) \cdot \beta \quad | :2$$

$$\frac{1}{2} = -\frac{\sqrt{5}}{2} \cdot \beta \quad | : \left(-\frac{\sqrt{5}}{2}\right)$$

$$\frac{1/2}{-\frac{\sqrt{5}}{2}} = -\frac{\sqrt{5}}{5} = \frac{1}{\sqrt{5}} = \beta$$

Da $-\alpha = \beta$

$$\frac{1}{\sqrt{5}} = \alpha$$

$$x_n = \frac{1}{\sqrt{5}} \cdot \frac{1+\sqrt{5}}{2} - \frac{1}{\sqrt{5}} \cdot \frac{1-\sqrt{5}}{2}$$

2) $X_{n+2} = 4 \cdot X_{n+1} - 4 \cdot x_n + 1$

Homogener Teil

$X_{n+2} = 4 \cdot x_{n+1} - 4 \cdot x_n$

$X_n = \lambda^n$

$\lambda^{n+2} = 4 \cdot \lambda^{n+1} - 4 \cdot \lambda^n \quad | : \lambda^n$

$\lambda^2 = 4 \cdot \lambda - 4 \quad | -\lambda^2 \; | \cdot (-1)$

$0 = \lambda^2 - 4\lambda + 4 \quad | PQ$

$\lambda_{1/2} = 2 \pm \sqrt{4 - 4}$

$= 2 \pm 0$

$\lambda_1 = 2 \quad \lambda_2 = 2 \quad ⚡ \; \lambda_1 = \lambda_2$

$X_n = \alpha \cdot 2^n + \beta \cdot 2^n \cdot n$

Inhomogenen Teil lösen

$X_{n+2} = 4 \cdot x_{n+1} - 4 \cdot x_n + 1$

$\gamma = x_n$

$\gamma = \cancel{4\gamma} - \cancel{4 \cdot \gamma} + 1$

$\gamma = 1$

$x_n = \alpha \cdot 2^n + \beta \cdot 2^n n + 1$

LGS mit den Anfangsbedingungen

$x_0 : \quad 1 = \alpha + 1 \quad | -1$

$X_1 = \quad 3 = \alpha \cdot 2 + \beta 2 + 1$

$\alpha = 0$

$3 = 2\beta + 1 \quad | -1 \; | : 2$

$1 = \beta$

$X_n = 2^n \cdot n + 1$

3) $a_{n+2} = \frac{1}{6} \cdot a_{n+1} + \frac{1}{6} \cdot a_n$

$\lambda^{n+2} = \frac{1}{6} \cdot \lambda^{n+1} + \frac{1}{6} \cdot \lambda^n \quad | : \lambda^n \quad | : \lambda^2 \quad | \cdot (-1)$

$= \lambda^2 - \frac{1}{6}\lambda - \frac{1}{6} \quad | PQ$

$\lambda_{1/2} = \frac{1}{12} \pm \sqrt{\frac{1}{144} + \frac{24}{144}}$

$= \frac{1}{12} \pm \sqrt{\frac{25}{144}}$

$= \frac{1}{12} \pm \frac{5}{12}$

$\lambda_1 = \frac{6}{12} \qquad \lambda_2 = \frac{-4}{12}$

$= \frac{1}{2} \qquad = -\frac{1}{3}$

$a_n = \alpha \cdot \frac{1}{2}^n + \beta \cdot -\frac{1}{3}^n$

$a_0 = 0 = \alpha + \beta$

$a_1 = \frac{5}{6} = \alpha \cdot \frac{1}{2} + \beta \cdot -\frac{1}{3}$

$\alpha = -\beta$

$\frac{5}{6} = -\beta \cdot \frac{1}{2} - \frac{1}{3}\beta$

$= -\frac{5}{6}\beta \quad | : -\frac{5}{6}$

$-1 = \beta$

$1 = \alpha$

$a_n = \left(\frac{1}{2}\right)^n - 1 \cdot \left(-\frac{1}{3}\right)^n$

Aufgabe Nr. 4

Master Theorem

1. Geben Sie die Definition des Master-Theorems an.
2. Schätzen Sie mit Hilfe des Master-Theorems die Komplexität von f ab.
   $f(n) = 2 \cdot f(n \backslash 2) + n$
3. Schätzen Sie $g(n) = 4 \cdot g(n \backslash 3) + (\frac{2}{3})^2 \cdot n$ mit Hilfe des Master-Theorems ab.
4. Schätzen Sie $g(n) = 4 \cdot g(n \backslash 5) + (\frac{3}{2})^3 \cdot n^2$ mit Hilfe des Master-Theorems ab.
5. Schätzen Sie $g(n) = 4 \cdot g(n \backslash 3) + 2 \cdot n^{\log_3(4)} + n$ mit Hilfe des Master-Theorems ab.

Aufgabe Nr. 4

1) Master - Theorem

$\alpha, \beta \in \mathbb{N}$, $\delta \in \mathbb{R}_+$

$\beta \geq 2$, $\delta \geq 0$

$f: \mathbb{N} \to \mathbb{R}_+$ und f hat die Form:

$$f(n) = \alpha \cdot f(n /\!/ \beta) \cdot O(n^\delta)$$

3 Fälle

1. $\alpha < \beta^\delta \rightarrow f \in O(n^\delta)$
2. $\alpha = \beta^\delta \rightarrow f \in O(\log_\beta(n) \cdot n^\delta)$
3. $\alpha > \beta^\delta \rightarrow f \in O(n^{\log_\beta(\alpha)})$

2) $g(n) = 4 \cdot g(n /\!/ 3) + \left(\frac{2}{3}\right)^2 \cdot n$

$\alpha = 4$ $\beta = 3$ $\delta = 1$

$4 > 3^1$ (3. Fall)

$f \in O(n^{\log_3(4)})$

3) $g(n) = 4 \cdot g(n /\!/ 5) + \left(\frac{3}{2}\right)^3 \cdot n^2$

$\alpha = 4$ $\beta = 5$ $\delta = 2$

$4 < 5^2$ (1. Fall)

$f \in O(n^2)$

4) $g(n) = 4 \cdot g(n /\!/ 3) + 2 \cdot n^{\log_3(4)} + n$

$\alpha = 4$ $\beta = 3$ $\delta = \log_3(4)$

$4 = 3^{\log_3(4)}$ (2. Fall)

$f \in O(\log_3(n) \cdot n^{\log_3(4)})$

Aufgabe Nr. 5

1. Geben Sie die Gleichung an, mit der wir Merge Sort definiert haben. Alternativ ist auch der Pythoncode von Merge Sort akzeptabel.

2. Wir haben die Funktion, die für zwei sortierte Listen L_1, L_2 berechnet, wie viele Vergleichsoperationen beim Aufruf von $merge(L_1, L2)$ geschehen, mit $cmpCount$ bezeichnet. geben Sie die Ungleichung an, die wir für das Verhältnis zwischen $\#L_1, \#L_2$ und $cmpCount(L_1, L_2)$ aufgestellt haben.

3. Schätzen Sie mit Hilfe des Master-Theorems die Komplexität von $MergeSort$ ab.

Aufgabe Nr. 5

a) Formale Definition:

$n := len(L)$

$n < 2 \rightarrow sort(L) = L$

$n > 2 \rightarrow merge(sort(L[:n//2]), sort(L[n//2:]))$

$merge(L_1, []) = L_1$

$merge([], L_2) = L_2$

$L_1 \neq [] \wedge L_2 \neq []:$

1. $x_1 \leq x_2 \rightarrow merge([x_1 | R_1], [x_2 | R_2]) = [x_1] + merge(R_1, [x_2|R_2])$
2. $x_1 > x_2 \rightarrow merge([x_1 | R_1], [x_2 | R_2]) = [x_2] + merge([x_1|R_1], R_2)$

Code:

```
def sort(L)
    n = len(L)
    if n < 2:
        return L
    else:
        return merge(sort(L[:n//2]), sort(L[n//2:]))

def merge(L1, L2):

    if L1 = []:
        return L2
    if L2 = []
        return L1
    x1, R1 = L1[0], L1[1:]
    x2, R2 = L2[0], L2[1:]
    if x1 <= x2:
        return [x1] + merge(R1, L2)
    else:
        return [x2] + merge(L1, R2)
```

b) $cmpCount(L_1, L_2) \leq \#L_1 + \#L_2$

c) $2 \cdot f(n/2) \cdot O(n) = f(n)$

$\alpha = 2, \quad \beta = 2 \quad \delta = 1$

$2 = 2 \quad \rightarrow 2.$ Fall

$f(n) \in O(\log_2(n) \cdot n)$

Aufgabe Nr. 6

Sortierproblem

1. Definieren Sie:
   (a) Partielle Ordnung
   (b) Lineare Ordnung
   (c) Quasiordnung
   (d) Totale-Quasiordnung
2. Geben Sie die Definition des Sortierproblems an.

Sortierproblem

Partielle Ordnung

1. $\forall x \in S: x \leq x$ (Reflexiv)
2. $\forall x,y \in S: (x \leq y \land y \leq x \rightarrow x = y)$ (anti-Stimmetrie)
3. $\forall x,y,z \in S: (x \leq y \land y \leq z \rightarrow x \leq z)$ (transitiv)

Quasi Ordnung

1. $\forall x \in S: x \leq x$ (Reflexiv)
2. $\forall x,y,z \in S: (x \leq y \land y \leq z \rightarrow x \leq z)$ (transitiv)

Lineare Ordnung

1. Partiell geordnet
2. $\forall x,y \in S: (x \leq y \lor y \leq x)$ (Linear)

Totale Quasiordnung

1. Quasiordnung
2. $\forall x,y \in S: (x \leq y \lor y \leq x)$ (Linear)

Sortierproblem

gegeben sei $\langle M, \leq \rangle$ in totaler Quasiordnung.
L ist Liste aus M
gesucht ist S mit folgenden Bedingungen

1. $\forall i \in \{0, \dots, len(S)-2\}: S[i] \leq S[i+1]$
2. $\forall x \in M: count(x, L) = count(x, S)$

Aufgabe Nr. 7

Sortieralgorithmen

1. Insertion Sort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
2. Selection Sort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
3. Merge Sort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
   (d) Komplexität im Master-Theorem
4. Quicksort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
5. Counting Sort
   (a) Formale Defintion mit allen Phasen
   (b) Implementierung
   (c) Komplexität
6. Radix Sort
   (a) Vorgehensweise
   (b) Implementierung
7. Heapsort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität

Aufgabe Nr. 7

1)

a) Formale Definition

```
Sort([]) = []
Sort([x]+R) = insert(x, sort(R))
insert(x, []) = [x]
x≤y
-> insert([x], [y]+R) = [x]+[y]+R
x>y
-> insert([x], [y]+R) = [y] + insert(x,R)
```

best $O(n)$

worst $\frac{1}{2} \cdot n^2 + O(n)$

average $\frac{1}{4} \cdot n^2 + O(n)$

```
def sort(L):
    if L == []:
        return []
    x, R = L[0], L[1:]
    return insert(x, sort(R))

def insert(x, L):
    if L == []:
        return [x]
    y, R = L[0], L[1:]
    if x <= y:
        return [x]+[y]+R
    else:
        return [y] + insert(x, R)
```

2)

Formale Definition

```
sort([]) = []
x := min(L)
-> sort(L) = [x] + sort(delete(x, L))
delete(x, []) = []
delete(x, [x]+R) = R
x≠y
-> delete(x, [y]+R) = delete(x, R)
```

immer $\frac{1}{2} \cdot n^2 + O(n)$

Code

```
def sort(L):
    if L == []:
        return []
    x = min(L)
    return [x] + delete(x, L)

def delete(x, L):
    if L == []:
        return []
    if x == L[0]:
        return L[1:]
    else:
        return [L[0]] + delete(x, L[1:])
```

3)

Formale Definition:

```
n: len(L)
n<2 -> sort(L) = L
n≥2 -> sort(L) = merge(sort(L[:n//2], L[n//2:])
merge(L1, []) = L1
merge([], L2) = L2
L1≠[] ∧ L2≠[] ∧ x1 ≤ x2
-> merge([x1|R1], [x2|R2]) = [x1] + merge(R1, [x2, R2])
L1≠[] ∧ L2≠[] ∧ x1 > x2
-> merge([x1|R1], [x2|R2]) = [x2] + merge([x1|R1] | R2)
```

$O(\log_2(n) \cdot n)$

Code:

```
def sort(L):
    n = len(L):
    if n < 2:
        return L
    return merge( sort(L[:n//2]), sort([n//2:])

def merge(L1, L2):
    if L1 == []:
        return L2
    if L2 == []:
        return L1
    x1, R1 = L[0], L[1:]
    x2, R2 = L[0], L[1:]
    if x1 <= x2:
        return [x] + merge(R1, L2)
    if x1 > x2:
        return [x2] + merge(L1, R2)
```

4)

Formale Definition

```
Sort([]) = []
Sort([x]+R) = sort([y ∈ R | y<x]) + [x] + sort([y ∈ R | y>x])
```

worst case : $O(n^2)$

average : $O(n \cdot \log_2(n))$

Code

```
def sort(L)
    if L == []:
        return L
    x,R = L[0], L[1:]
    S = [y for y in R if y < x]
    B = [y for y in R if y > x]
    return sort(S) + [x] + sort(B)
```

7)

Formale Definition

```
Nil. toList() = []
h≠Nil ∧ h.top() = <p, _>
-> h.toList[] = [p] + h.remove().toList()
```

Aufgabe Nr. 8

ADTs + Set und Maps

1. Geben Sie die formale Definition von ADTs an.
2. Geben Sie die formale Definition des ADT Stack an.
3. Beschreiben Sie was ein Generator in einem
4. Beschreiben Sie den Shunting-Yard-Algorithmus.
5. Geben Sie die formale Definition des ADT Map an.
6. Was haben geordnete Binärbäume, AVL Bäume, Tries und co. miteinander zu tun, was verbindet sie?
7. Geben Sie die formale Definition und die Komplexität von geordneten Binärbäumen an.
8. Geben Sie die formale Definition und die Komplexität von AVL Bäumen an.
9. Geben Sie die formale Definition und die Komplexität von Tries an.
10. Beschreiben Sie, was eine Prioritätswarteschlange ist.
11. Geben Sie die formale Definition und die Komplexität von Prioritätswarteschlangen an.
12. Geben Sie die formale Definition und die Komplexität des Heaps an.