Aufgabe 1

(a) $3n^3 + 8n^2 + n \in O(n^3)$

Def. $O(g(n)) := \{f(n) \mid \exists\, c, n_0 > 0, \forall n \geq n_0 : f(n) \leq c \cdot g(n)\}$

Bew: $f(n) = 3n^3 + 8n^2 + n$

$g(n) = n^3$

$3n^3 + 8n^2 + n \leq 3n^3 + 8n^3 + n^3$

$3n^3 + 8n^2 + n \leq 14n^3$

$\Rightarrow \exists\, c = 14, n \geq 1$

Es gilt für die Def.

□

(b) $2^n \in o(n!)$

Bew: $f(n) = 2^n$, $g(n) = n!$

Nach Def $o(g(n)) := \{f(n) \mid \forall c > 0\ \exists\, n_0 > 0, \forall n \geq n_0 : f(n) < c \cdot g(n)\}$

für $\forall\ c > 0$, falls $n_0 = 4$, gilt für

$2^n \leq c \cdot n!$

Es gilt für die Def.

□

(c) $2\log_n \in \Omega((\log n)^2)$

Bew: $f(n) = 2\log n$ $\quad g(n) = (\log n)^2$

$2\log n \geq c \cdot (\log n)^2$

$2\log n \geq c \cdot \log n \cdot \log n \quad | : \log n$

$2 \geq c \cdot \log n$

für $c = 1$, $n_0 = 4$, $\exists\, n = 8$, wobei $2 < \log 8$, $2 < 3$, wiederspruch!

□

(d) Bew: $\max\{f(n), g(n)\} \in \Theta(f(n)+g(n))$ für nicht negative Funktionen f und g.

Weil $f(n) \geq 0$, $g(n) \geq 0$, $\exists n_f$, für jede $n > n_f$, dass $f(n) \geq 0$
$\exists n_g$, für jede $n > n_g$, um $g(n) \geq 0$ zu gelten.

$\Rightarrow f(n)+g(n) \geq f(n)$, für jede $n > n_f$.
$f(n)+g(n) \geq g(n)$, für jede $n > n_g$.

setz $n_{max} = \max(n_f, n_g)$. dann für $\forall n > n_{max}$
$f(n)+g(n) \geq 0$ ist immer klar.

Also $f(n)+g(n) \geq \max(f(n), g(n))$ ist klar. mit $\forall n > n_{max}$.

setz $F_{max}(n) = \max(f(n), g(n))$

$f(n) \leq F_{max}(n)$, $g(n) \leq F_{max}(n)$

$\Rightarrow f(n)+g(n) \leq 2 \cdot F_{max}(n)$ ist klar

$\Rightarrow \frac{f(n)+g(n)}{2} \leq \max(f(n), g(n)) \leq f(n)+g(n)$

$\exists c_1 = \frac{1}{2}$, $c_2 = 1$, für $\forall n \geq \max(n_f, n_g)$ gilt für die Def $\square$

Aufgabe 2

$2^{n^2} \ > \ n^n \ > \ (n+1)! \ = \ n! \ > \ (2^n)^2 \ > \ 3^n \ > \ 2^n \ > \ n^{100} \ > \ 10^{100}n \ >$
$n \log n \ > \ \sqrt{n} \ > \ (\log n)^2 \ > \ \log n^3 \ = \ \log n \ = \ \log \sqrt{n} \ > \ \sqrt{\log n}$

Aufgabe 3

a)
In der Vorlesung wurden diese vier Algorithmen gezeigt:
Selectionsort, Insertionsort, Mergesort und Quicksort.
Davon sind jedoch nur Insertion sort und Mergesort stabil.

Beispiel für Selectionsort:
Die Zahlen in Klammern sind die Startindices

unsorted array $= [2, 1, 4(2), 6, 4(4), 3]$

Wenn man jetzt mit dem Selectionsort aus der Vorlesung sortiert, dann bekommt man folgendes Array:
sorted array $= [1, 2, 3, 4(4), 4(2), 6]$

Die einzelnen Schritte sind:
$2 \rightarrow 1, \quad 2 \rightarrow 2, \quad 4(2) \rightarrow 3, \quad 6 \rightarrow 4(4), \quad 6 \rightarrow 4(2), \quad 6 \rightarrow 6$

Wie man sieht sind die beiden 4 in der falschen Reihenfolge, das bedeutet, dass Selectionsort instabil ist.

b)
Selectionsort:
Wir würden beim Aufruf von Selectionsort am Anfang ein weiteres Array mit dem Startindex der Einzelnen Komponente befüllen. Also zum Beispiel:
$[0, 1, 2, 3]$ für eine liste mit 4 Elementen.

Bei jedem Tausch von Elementen im Hauptarray, sollte auch im Indexarray die Indices getauscht werden, also zum Beispiel:
Hauptarray: $[5, 3, 1]$
Indexarray: $[0, 1, 2]$
Tausch von 5 und 1 im Hauptarray:
$[5, 3, 1] \rightarrow [1, 3, 5]$

Also wird auch im Indexarray getauscht:
$[0, 1, 2] \rightarrow [2, 1, 0]$

Ein letzten Schritt den wir machen müssen, ist eine weitere If-bedingung einbauen, die bei gleichen Elementen die Indices im Indexarray überprüft.

Damit wissen wir also welches Element, wenn diese gleich sind(zum Beispiel 4 und 4), im Startarray den kleineren Index hatte, und können somit Stabil sortieren.