Induktion: Lösungen

1 Vollständige Induktion

1.2 Gesamtzahl Roboter nach h Generationen

Wie viele Roboter wurden nach h Generationen gebaut, d.h., wie viele Roboter wären in Existenz, wenn die Roboter nicht nach einem Jahr kaputtgingen?

a)	Voraussetzung	$N_h = 2^{h+1} - 1$
b)	Verankerung	$N_0 = 2^1 - 1 = 1$
c)	Schritt	$N_{h+1} = 2^{h+1+1} - 1 = 2^{h+1} \cdot 2 - 1 = 2^{h+1} + 2^{h+1} - 1 = 2^{h+1} + N_h$

Da in der $(h+1)$ten Generation 2^{h+1} Roboter dazukommen, gilt der Induktionsschritt.

1.4 Anzahl Generationen bis zu einer Milliarde Roboter

```
def log(number, base):
    count = 0

    while number >= base:
        number /= base
        count += 1

    return count

print(log(1000000000000,2))
```

Es wird solange durch die Basis geteilt, bis die Zahl darunterfällt.

1.5 Anwendung des Logarithmus

Bei jeder Faltung eines Blattes Papiers wird dieses in der Dicke verdoppelt. Wie oft musst du ein Blatt Papier mindestens falten, damit der Stapel bis zum Mond reicht? Ein Blatt Papier hat die Dicke 0.1 mm und die Distanz Erde-Mond beträgt ungefähr 384'400 km.

Es gilt, den Faktor

$$\frac{384'400'000}{0.000'1} = 3.84 \cdot 10^{12}$$

durch Verdoppelung zu erreichen. Die obige Logarithmusfunktion liefert dafür als Resultat 41.

```
print(log(3.84E12, 2))      # Ausgabe: 41
```

Also wird der Mond zwischen der 41. und der 42. Faltung erreicht.

Quelle: BD

2 Roboter sortieren

2.1 Min-Sort Algorithmus

2.1.1 Algorithmus

1. Setze den Zeiger *index=0* auf den Beginn der Liste.
2. Finde das Minimum aus der Liste.
3. Vertausche die Position des Minimums mit dem Element an der Position *index*.
4. Erhöhe *index* um eins (for-Schleife).
5. Wiederhole 2. bis 4. solange, bis die Liste sortiert ist.

2.1.2 Anzahl Vergleiche

$$N_n = \frac{n \cdot (n+1)}{2}$$

a)	Behauptung	$N_n = \frac{n \cdot (n+1)}{2}$
b)	Verankerung	$N_1 = \frac{1 \cdot 2}{2} = 1$
c)	Schritt	$N_{n+1} = \frac{(n+1) \cdot (n+1+1)}{2} = \frac{(n+1) \cdot (n+2)}{2} = \frac{(n+1) \cdot n}{2} + \frac{2 \cdot (n+1)}{2} = N_n + n + 1$

Wie viele Schritte werden für eine Liste der Länge 1000 benötigt, wie viele für die Länge 1'000'000?

$N_{32} = 528$

$N_{1'000'000} = 500'000'500'000$

Das Sortieren dauert also bei etwa einer Milliarde Operationen pro Sekunde viel zu lange.

2.2 Merge-Sort: Ein induktiver Ansatz

2.2.4 Anzahl Vergleiche

Inwiefern ist der Merge-Sort Algorithmus schneller als der Min-Sort Algorithmus? Die Geschwindigkeit des Algorithmus hängt wesentlich von der **Anzahl Vergleiche** ab: Wie oft wird eine Zahl mit einer anderen verglichen.

Überlege dir eine intuitive Begründung, weshalb beim Merge-Sort weniger Vergleiche notwendig sind als beim Min-Sort Algorithmus.

Beim Min-Sort Algorithmus wird jeweils jedes Element mit jedem verglichen. Beim Merge-Sort wird jedes Element nur mit dem jeweiligen Nachbarn verglichen.

Überlege dir für den Merge-Sort Algorithmus, wie viele Vergleiche insgesamt notwendig sind. Nimm als Beispiel an, dass eine Liste der Länge 32 sortiert werden soll. Tipp: Um die Frage zu beantworten, benötigst du Wissen aus der Roboterfortpflanzung von *Logarithmus* von oben.

Der Algorithmus wird (wie in der Abbildung) in Stufen unterteilt. Bei der ersten Stufe werden Listen der Länge 1 zu Listen der Länge 2 gemerged. Bei der zweiten Stufe ... usw. usf.

Pro Stufe wird jedes Element genau einmal betrachtet und mit dem Nachbarn verglichen. Pro Stufe finden sich also *n* Vergleiche. Insgesamt finden wir *log(n,2)* Stufen vor, für eine Liste der Länge 32 also

$$32 \cdot \log(32, 2) = 32 \cdot 5 = 160$$

Vergleiche. Also wesentlich weniger als beim Min-Sort Algorithmus.

2.3 Vergleich Min-Sort und Merge-Sort

siehe Lösungsfile *sort_loesung.py*