Plan für heute (und die nächste Woche)

- HashMap und TreeMap
  - Aufgabe: Wörterbuchimplementierung am Wochenende

- Theorie zu Maps
  - Hashtabellen
  - binäre Suchbäume

- Implementierungsaufgaben zu binären Suchbäumen

- optional: AVL-Bäume oder Rot-Schwarz-Bäume
  - Optimierung zu binären Suchbäumen

- ToDo-Liste weitermachen

Wörterbuch

Lampe - lamp

Ball - ball

Auto - car

Ampel - traffic light

Buch - book

Kuh - cow

Abbildung von "Schlüsseln" ("Keys") auf "Werte" ("Values")

Map

Komplexität im Worst-Case:

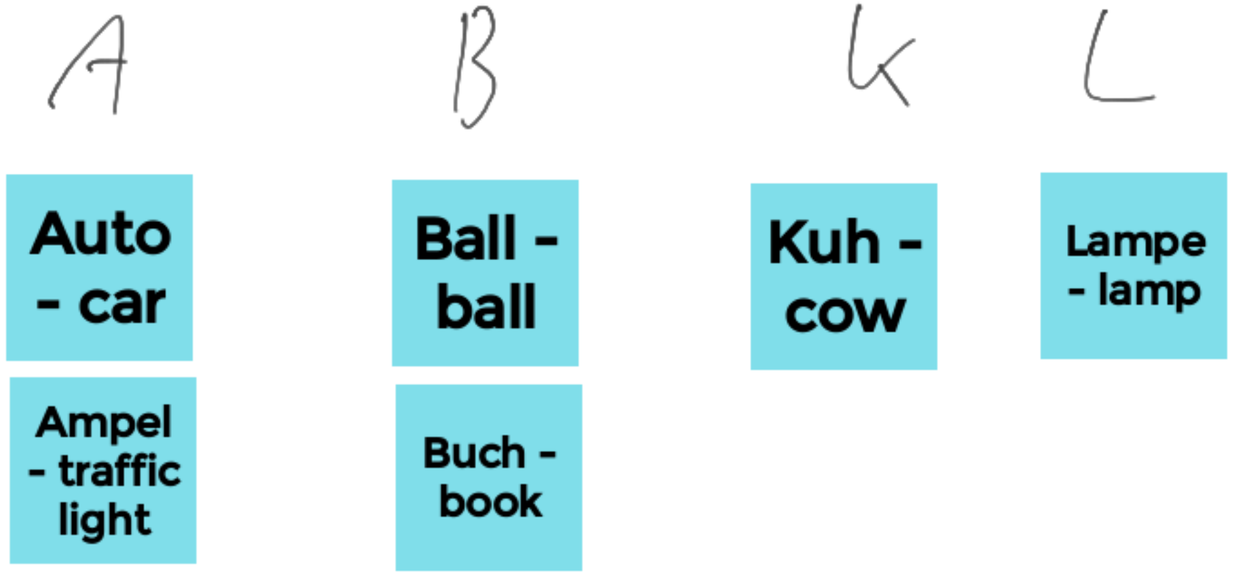
O(n)

(Linear)

Im Schnitt deutlich besser, im Idealfall konstant.

Wörterbuch

HashMap



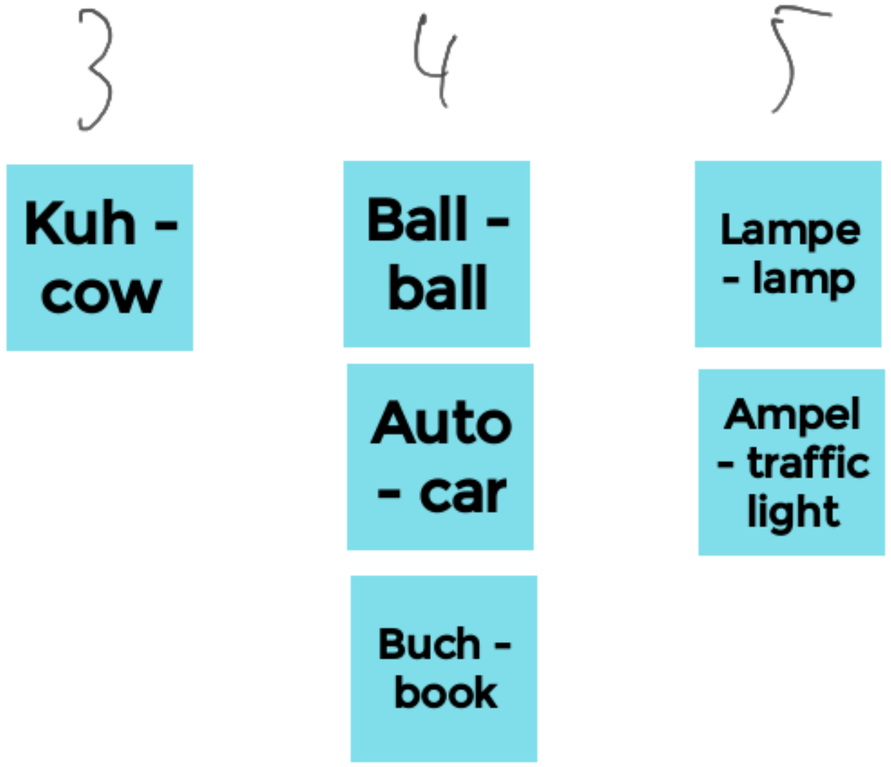
Idee: Elemente werden in "Buckets" einsortiert.

1. Ansatz: Anfangsbuchstaben

- Hashfunktion: Bildet jeden Schlüssel auf einen möglichst eindeutigen "Hash" ab.
- Wird genutzt, um die Stelle (den "Bucket") zu finden, an der das Element steht.
- Im Idealfall enthält jeder Bucket nur ein Element, dann geht der Zugriff sehr schnell.

Wörterbuch

HashMap



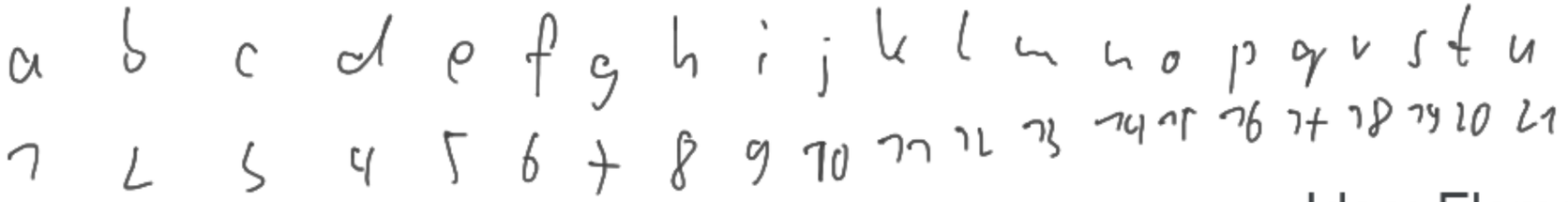
Idee: Elemente werden in "Buckets" einsortiert.

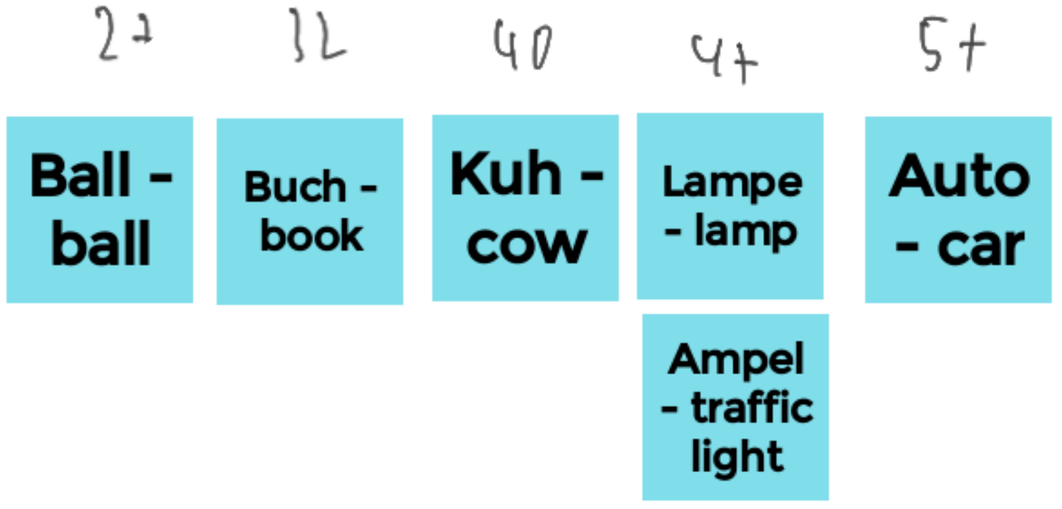
1. Ansatz: Anfangsbuchstaben
2. Ansatz: Länge des Schlüssels

- Hashfunktion: Bildet jeden Schlüssel auf einen möglichst eindeutigen "Hash" ab.
- Wird genutzt, um die Stelle (den "Bucket") zu finden, an der das Element steht.
- Im Idealfall enthält jeder Bucket nur ein Element, dann geht der Zugriff sehr schnell.

Wörterbuch

HashMap





Idee: Elemente werden in "Buckets" einsortiert.

1. Ansatz: Anfangsbuchstaben
2. Ansatz: Länge des Schlüssels
3. Ansatz: Schlüssel-"Quersumme"
   - jedem Buchstaben eine Zahl zuordnen und die Summe berechnen.

- Hashfunktion: Bildet jeden Schlüssel auf einen möglichst eindeutigen "Hash" ab.
- Wird genutzt, um die Stelle (den "Bucket") zu finden, an der das Element steht.
- Im Idealfall enthält jeder Bucket nur ein Element, dann geht der Zugriff sehr schnell.

Wörterbuch

HashMap

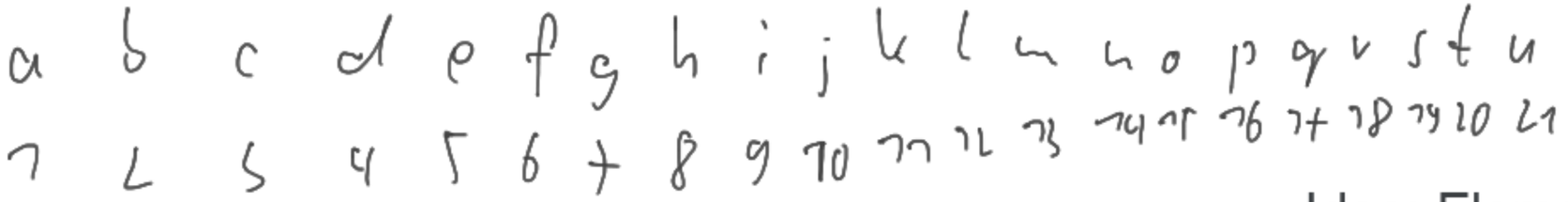
Lampe - lamp

Ball - ball

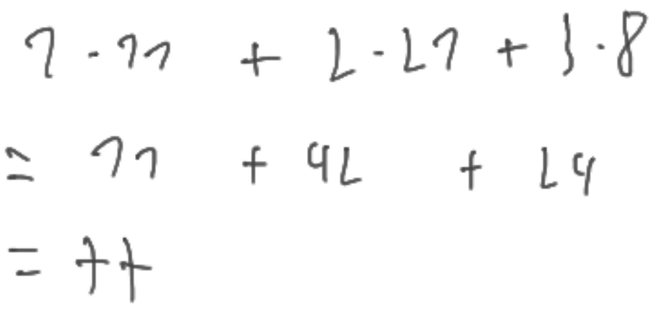
Auto - car

Ampel - traffic light

Buch - book



Kuh - cow



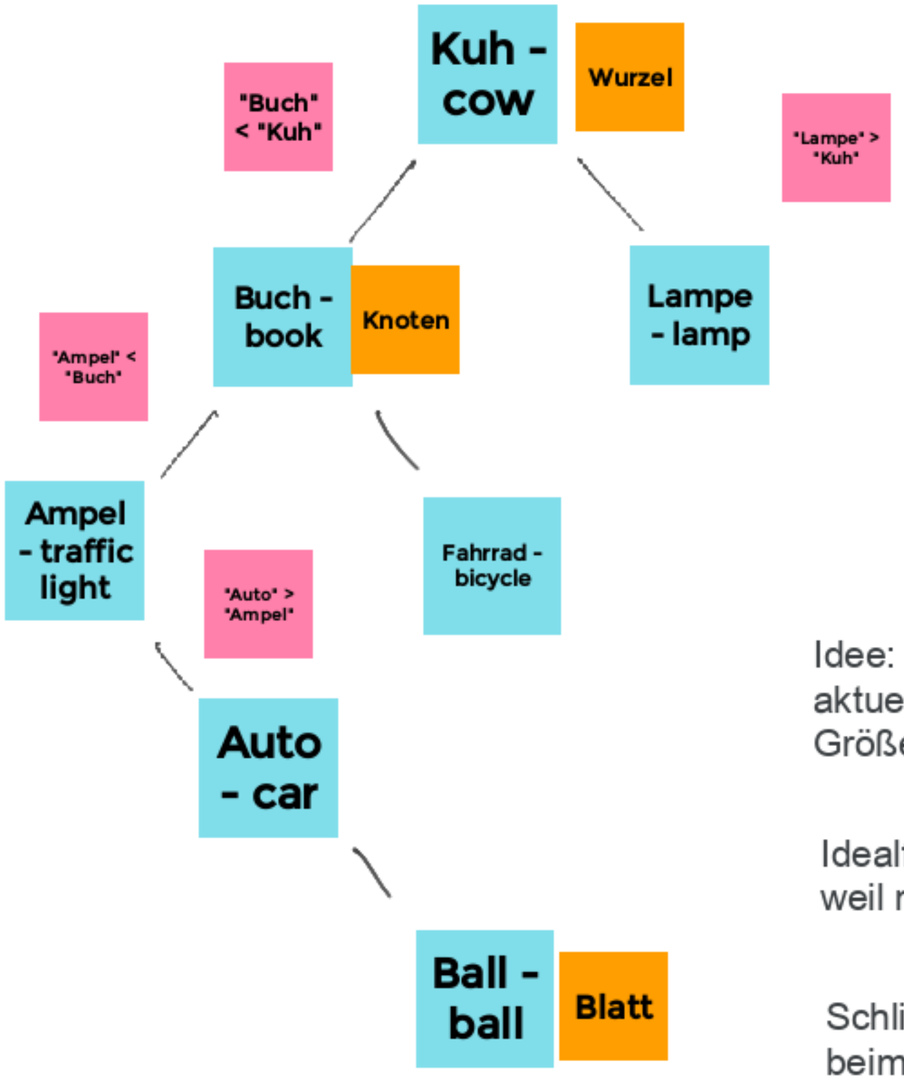
Idee: Elemente werden in "Buckets" einsortiert.

1. Ansatz: Anfangsbuchstaben
2. Ansatz: Länge des Schlüssels
3. Ansatz: Schlüssel-"Quersumme"
   - jedem Buchstaben eine Zahl zuordnen und die Summe berechnen.
4. Ansatz: Gewichte Schlüssel-Quersumme.
   - (1 x 1. Buchstabe) + (2 x 2. Buchstabe) + ..

- Hashfunktion: Bildet jeden Schlüssel auf einen möglichst eindeutigen "Hash" ab.
- Wird genutzt, um die Stelle (den "Bucket") zu finden, an der das Element steht.
- Im Idealfall enthält jeder Bucket nur ein Element, dann geht der Zugriff sehr schnell.

Wörterbuch

Suchbaum



Idee: Elemente werden immer mit dem aktuellen Knoten verglichen und je nach Größe links oder rechts unterhalb einsortiert.

Idealfall: Baum hat eine logarithmische Tiefe, weil man wenige Lücken bekommt.

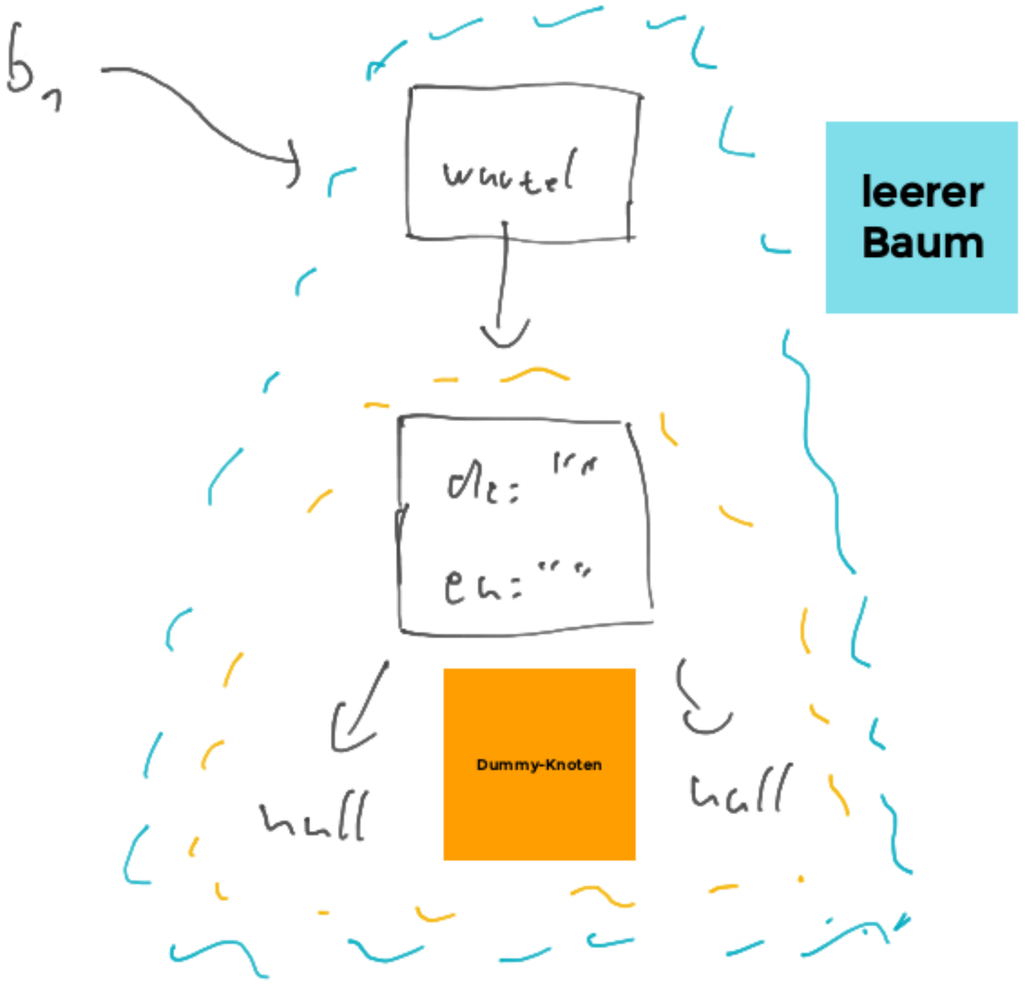
Schlimmster Fall: Die Elemente sind schon beim Einfügen sortiert. Dann wird der Baum de facto eine Liste.

Java-Aufbau eines Baums

```
Baum b1 = new Baum();
```

führt aus...

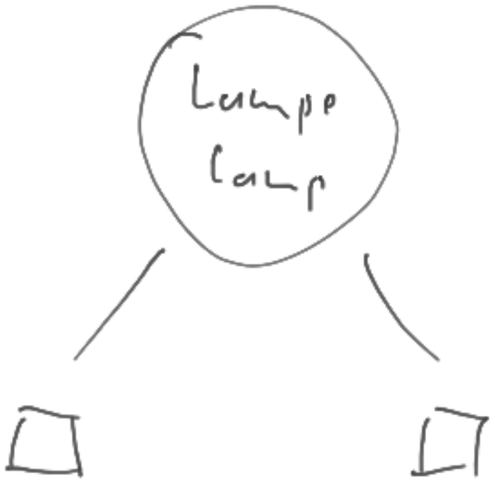
```
public Baum() {
  wurzel = new Knoten();
}
```



leerer Baum:

Wurzel ist Dummy

Baum mit einem Element



Datenknoten mit
zwei Dummies

Baum mit mehreren Elementen