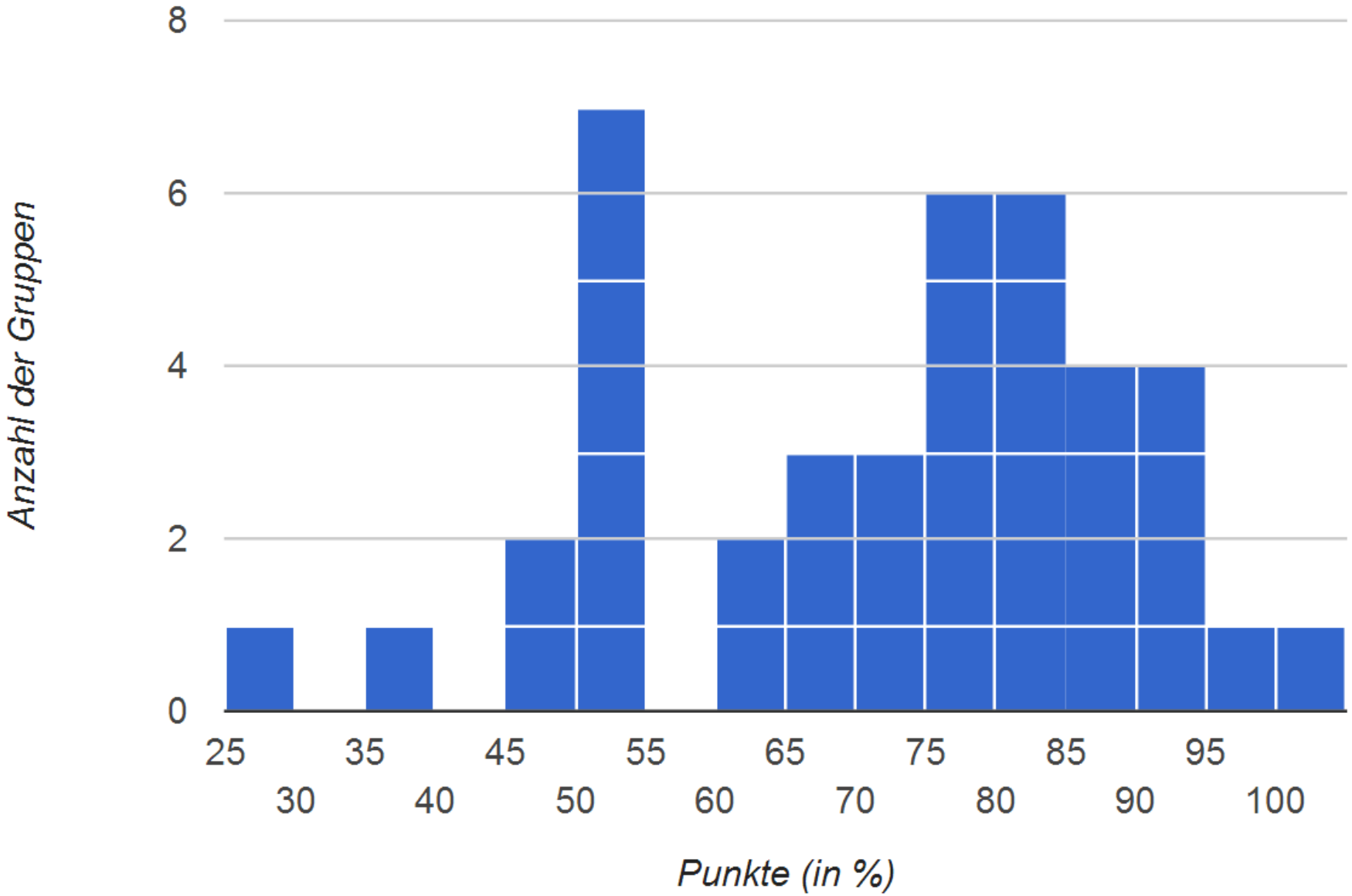
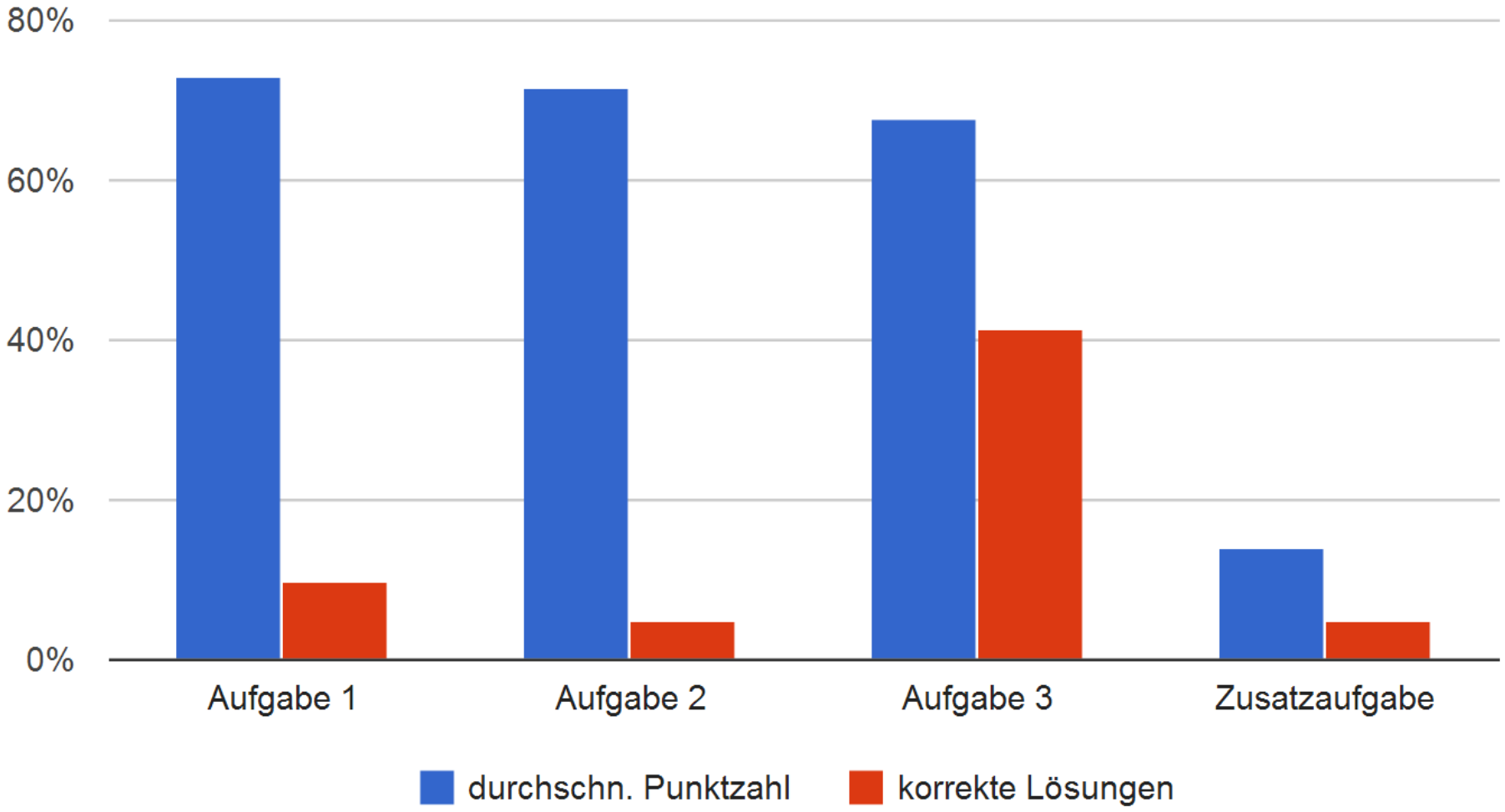
Übungsblatt 6 - Auswertung

Hashing und Graphen

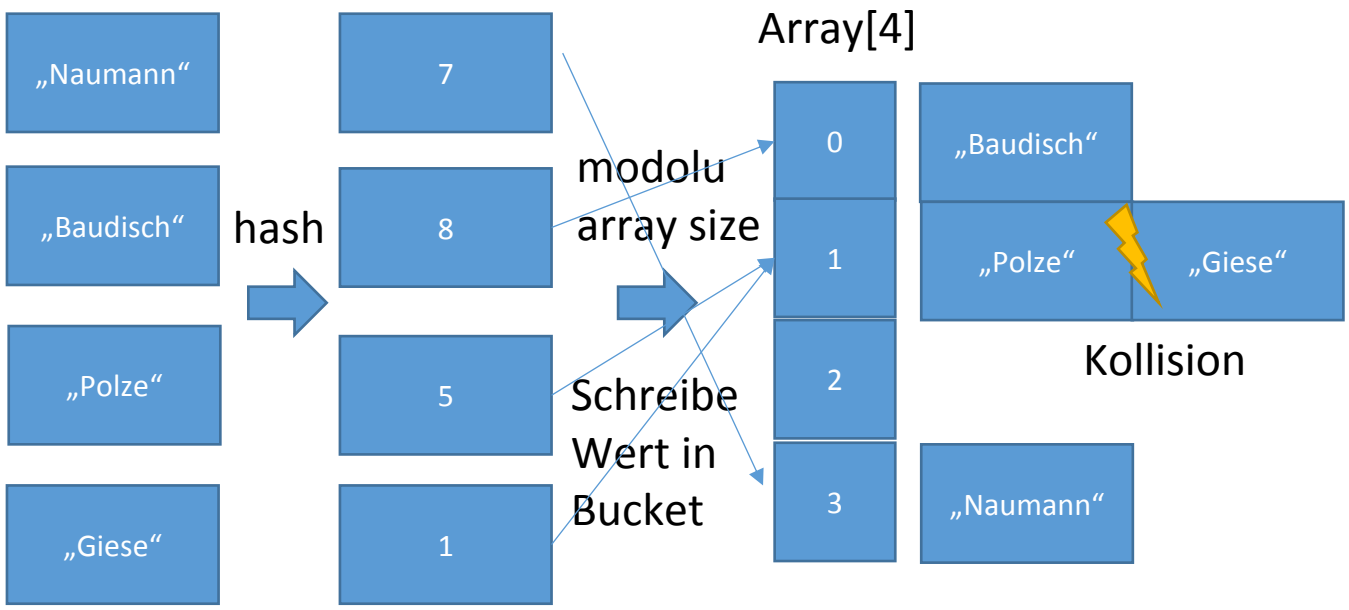
Punktverteilung (Gesamt)



Ergebnisse pro Aufgabe



Hashtabelle



Hashtabelle - Kollisionen

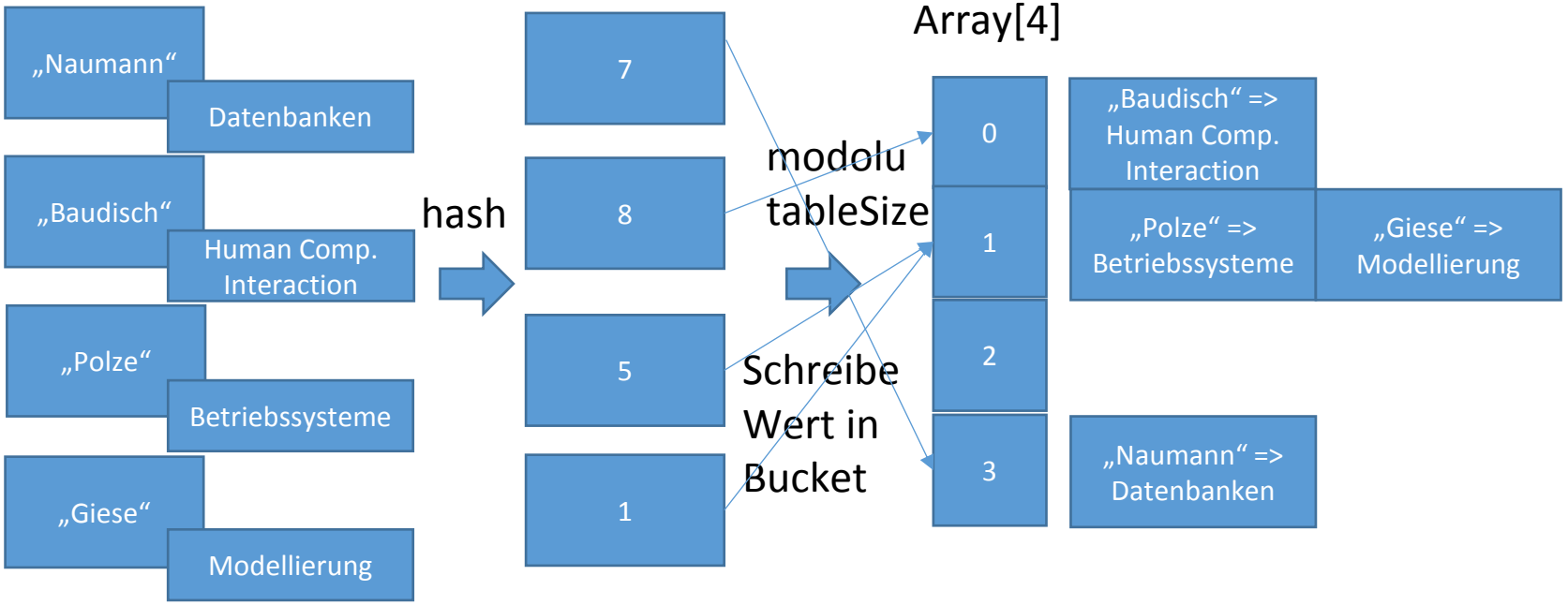
- Kollision → Bucket wird mehrfach belegt

- Lösung:
  - Statt einer Entität wird eine Liste von Entitäten gespeichert
  - Lineares Sondieren
  - Quadratisches Sondieren
- Problem:
  - Suche in O(k) (k = #Kollisionen bzw. Bucketgröße)
  - Lösung: Tabelle vergrößern

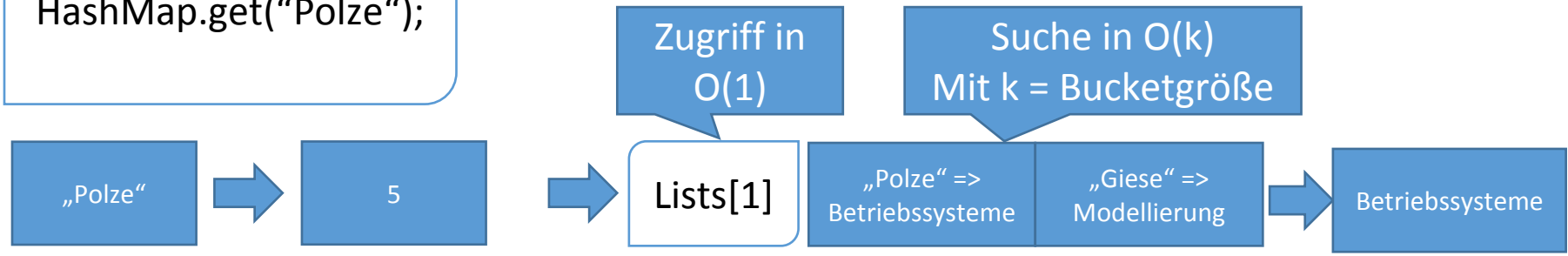
HashMap (assoziativ

```
class public HashMap{
    class private Pair{…}
    …
}
```

X[„Naumann“] = Datenbanken

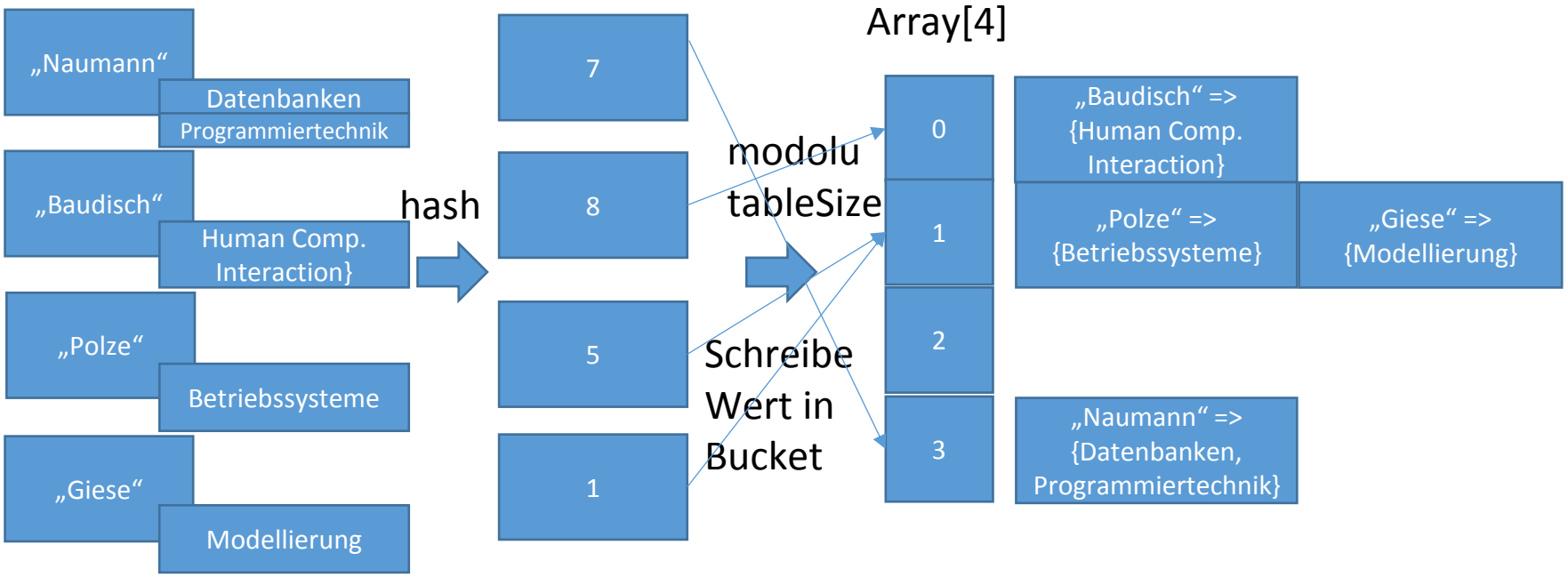


HashMap.get(“Polze“);

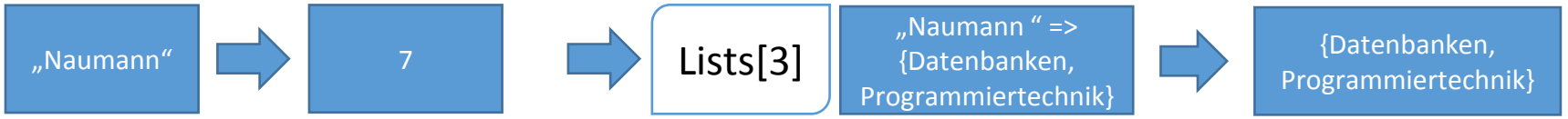


HashMultiMap

X[„Naumann“] = {Datenbanken, Programmiertechnik}



HashMap.get(“Naumann“);



Graphen

Adjazenzmatrix

- Speicherverbrauch $O(|V|^2)$
  - Besonders ein Nachteil bei dünnbesetzen Graphen
  - Bei dichter Besetzung ist die Darstellung kompakter
- Finden von Kanten in O(1)
- Einfügen/Löschen von Knoten aufwendig

Adjazenzliste

- Speicherverbrauch $O(|V|+|E|)$
- Finden von Kanten idR in $O(|V|)$ (geht aber auch geschickter)
- Nachbarn eines Knotens liegen direkt vor
- Einfügen/Löschen von Knoten idR weniger aufwendig

Graphen

- Connected Components
  - Relation symmetrisch *machen*
  - Starten bei beliebigem Knoten k -> Tiefensuche/Breitensuche um alle anderen Knoten in der entsprechenden Gruppe (Component) zu finden
  - Jeden bereits betrachteten Knoten aus der Liste aller Knoten löschen und Verfahren mit den übrigen Knoten fortsetzen
- Dijkstra
  - Siehe Vorlesungsfolien

Zusatzaufgabe

- Qatar Airways Crew möchte Fußballszenen aus der Mediathek herunterladen
  - *N* Intervalle in denen die Videos online sind (*s[i]* bis *e[i]*)
  - *Q* Crews mit jeweils *K* Aufenthalten in Deutschland
- Gesucht
  - Anzahl der Intervalle (distinct) in denen Crew *q* mindestens zu einen Aufenthalt in Deutschland hat ($s[i] \leq a_q[j] \leq e[i]$)
- Mögliche Lösung:
  - IntervallTrees (ähnlich zu BST)

    https://www.youtube.com/watch?v=dQF0zyaym8A