Interdisziplinäres Projekt

Weiterführende Graphalgorithmen

Mark J. Becker, Aleksejs Voroncovs, Ruslan Zabrodin

27. April 2015

Weiterführende Graphalgorithmen

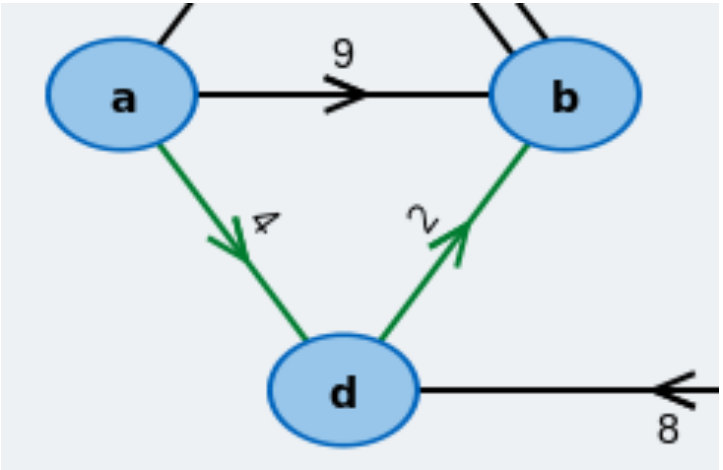
- All-Pairs Shortest Path Problem
- Matchingprobleme in bipartiten Graphen
- Eulertour Problem
- Chinese Postman Problem

Floyd-Warshall

- Sucht nach kürzesten Pfade zwischen allen Paaren von Knoten
- Voraussetzung: keine negative Kreise
- Gerichtet & Ungerichtet
- $O(n^3)$
- **Prinzip:** dynamische Programmierung

Floyd-Warshall

- Kürzester Weg:
  - Bei jedem Schritt versucht der Algorithmus den Weg zu verbessern

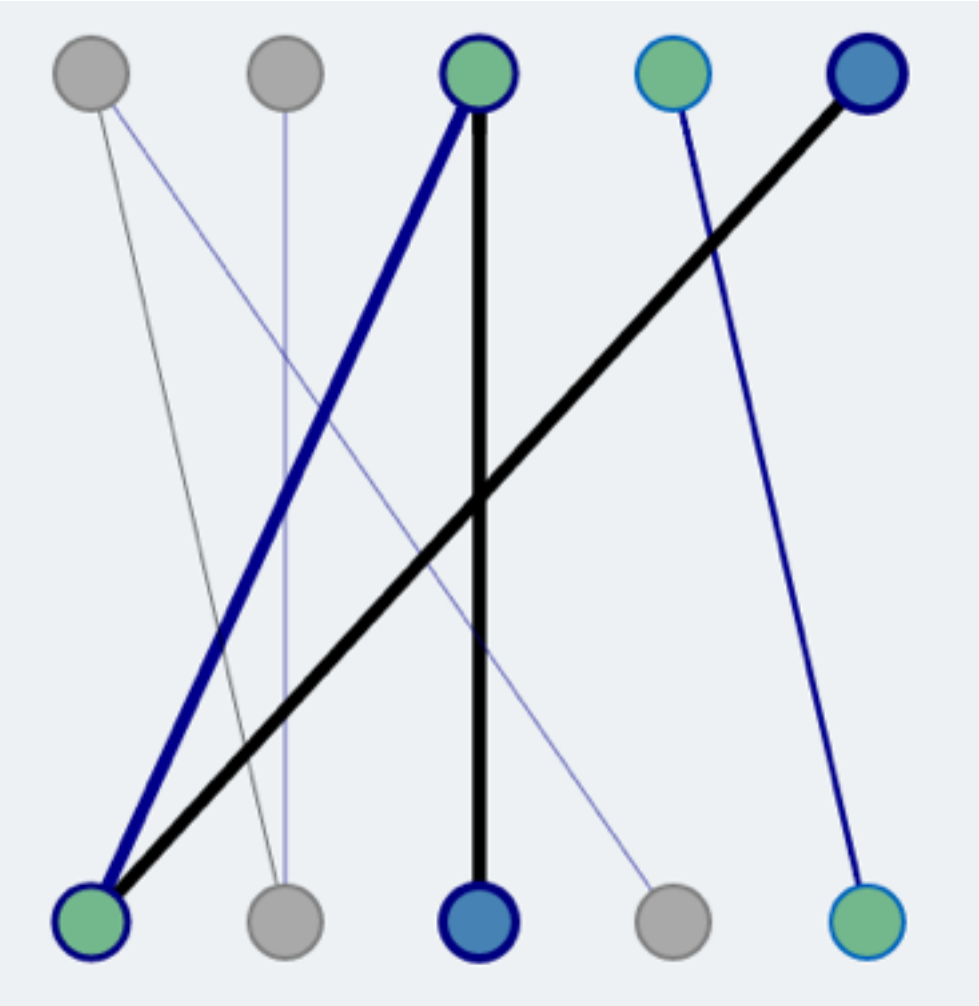


Hopcroft-Karp

- Suche nach **maximalen Matchings** in bipartiten Graphen
  - verbesserte Laufzeit
- **kürzeste Augmentationswege**
  - knotendiskunkte Augmentationswege
  - inklusions-maximale Menge

Hopcroft-Karp

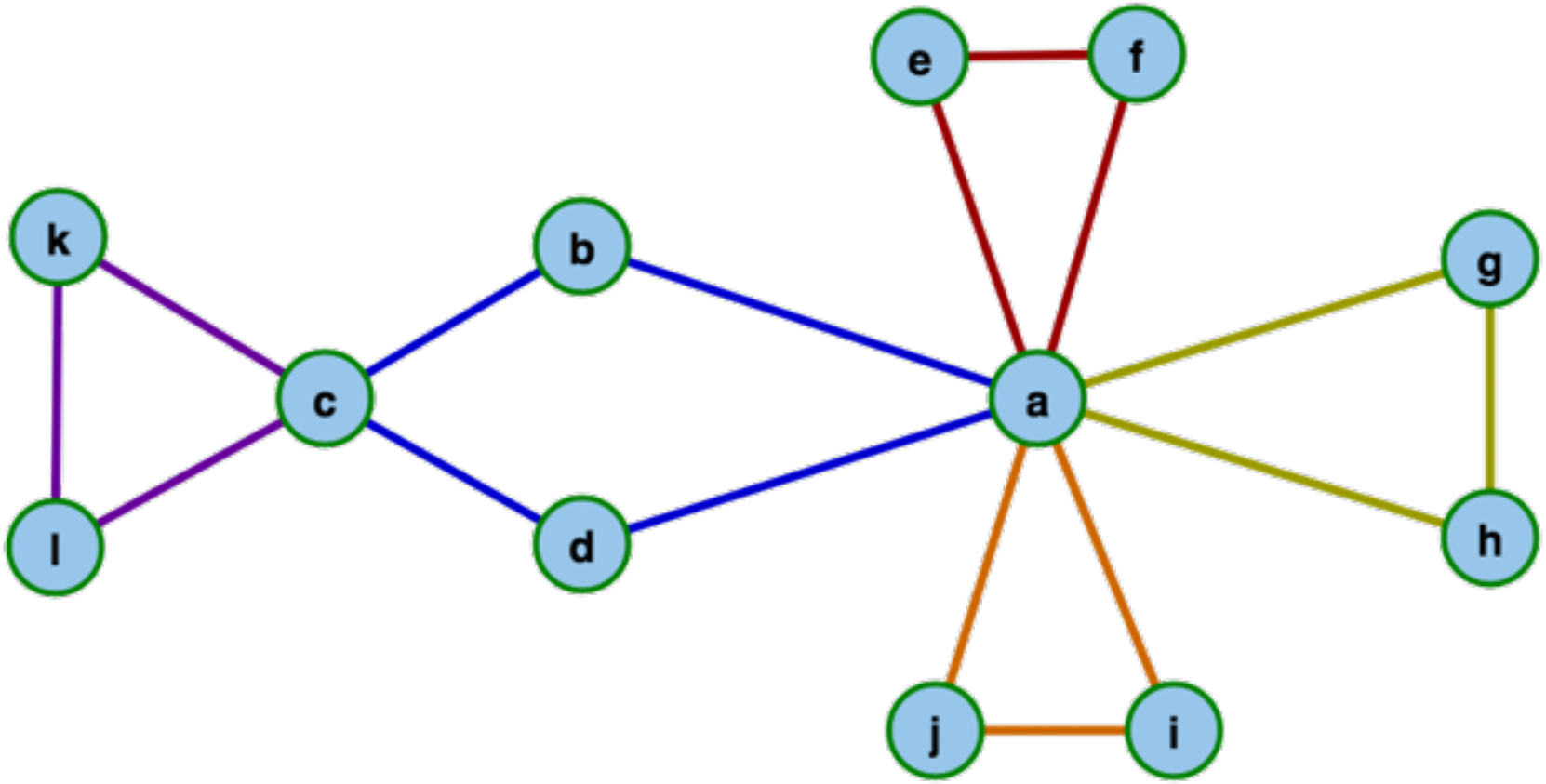
- Augmentationsweg:
  - kürzester
  - knotendisjunkt



Hierholzer Algorithmus

- Eulertour Problem für geeignete Graphen
- **Fokus:**
  - Voraussetzungen des Graphen
  - Gerichtet & Ungerichtet
- **Prinzip:** Disjunkte Kreise

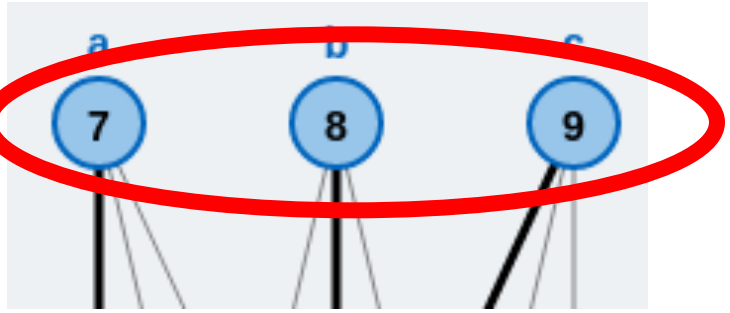
Hierholzer Algorithmus



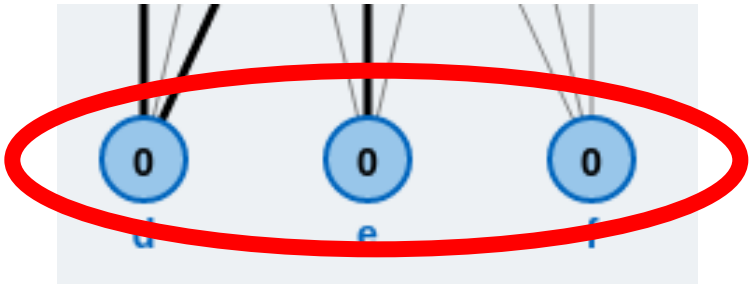
Ungarische Methode

- **Ziel:** Suche des maximalen Matchings
- **Hauptschritte:**
  - Ursprüngliche Markierungen zuweisen

Ungarische Methode



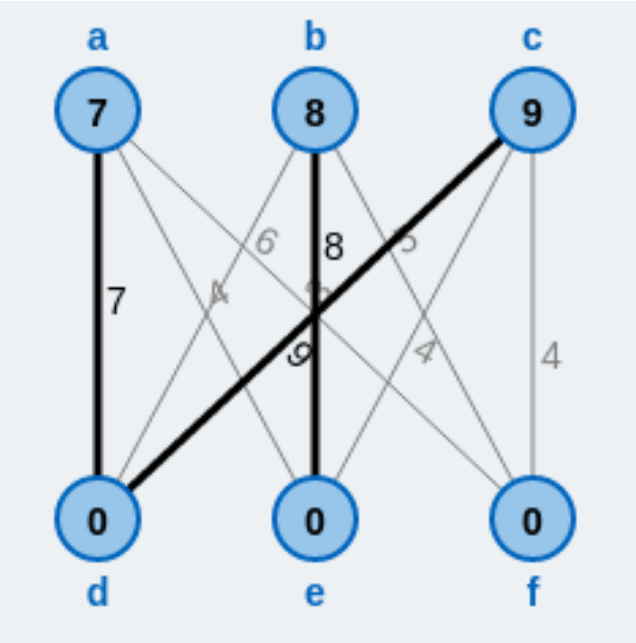
Markierungen



Ungarische Methode

- **Ziel:** Suche des maximalen Matchings
- **Hauptschritte:**
  - Ursprüngliche Markierungen zuweisen
  - Gleichheitsgraph bestimmen

Ungarische Methode

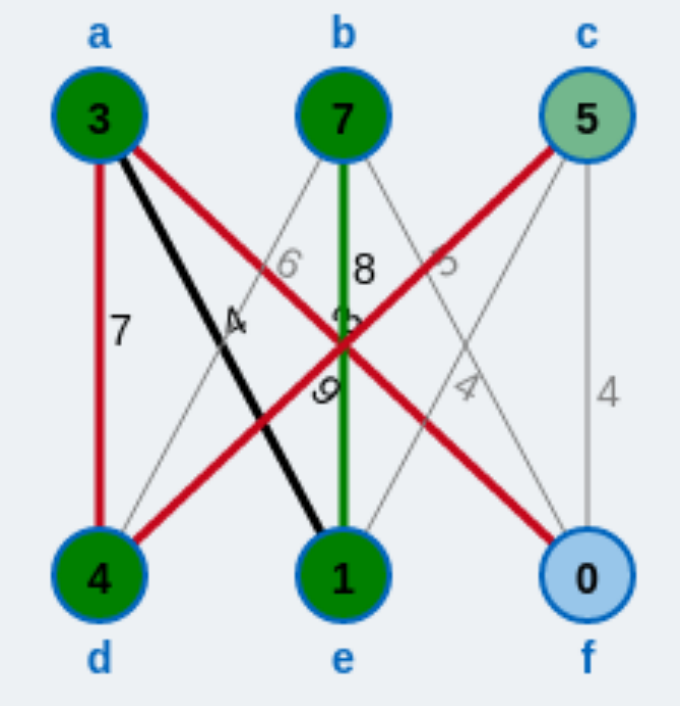


Gleichheitsgraph

Ungarische Methode

- **Ziel:** Suche des maximalen Matchings
- **Hauptschritte:**
  - Ursprüngliche Markierungen zuweisen
  - Gleichheitsgraph bestimmen
  - Augmentationsweg finden und aktuelles Matching erweitern

Ungarische Methode

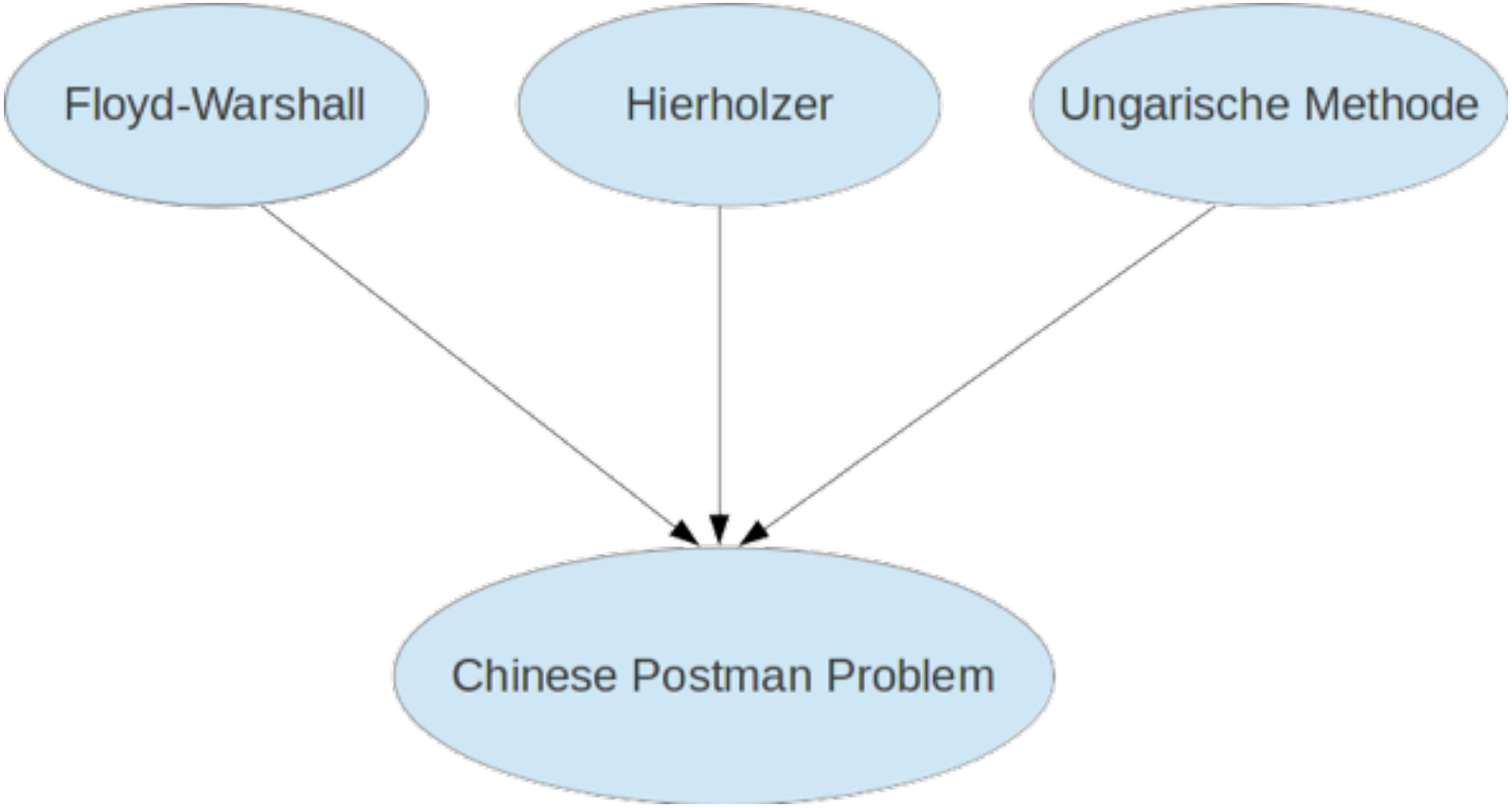


Augmentationsweg

Ungarische Methode

- **Ziel:** Suche des maximalen Matchings
- **Hauptschritte:**
  - Ursprüngliche Markierungen zuweisen
  - Gleichheitsgraph bestimmen
  - Augmentationsweg finden und aktuelles Matching erweitern
  - Wenn nötig, Markierungen anpassen

Chinese-Postman

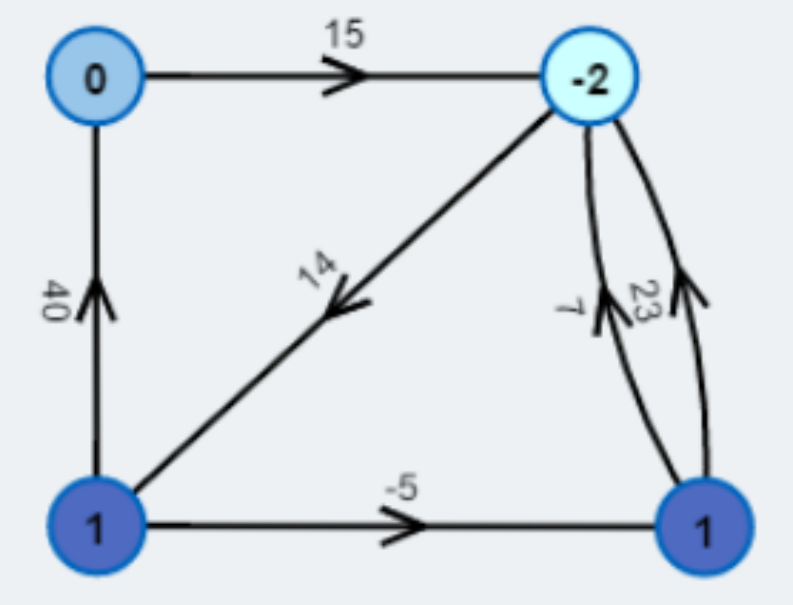


Chinese-Postman

- **gerichtete** Version
- Ziel: Einfügen neuer Pfade, sodass Graph **eulersch** wird

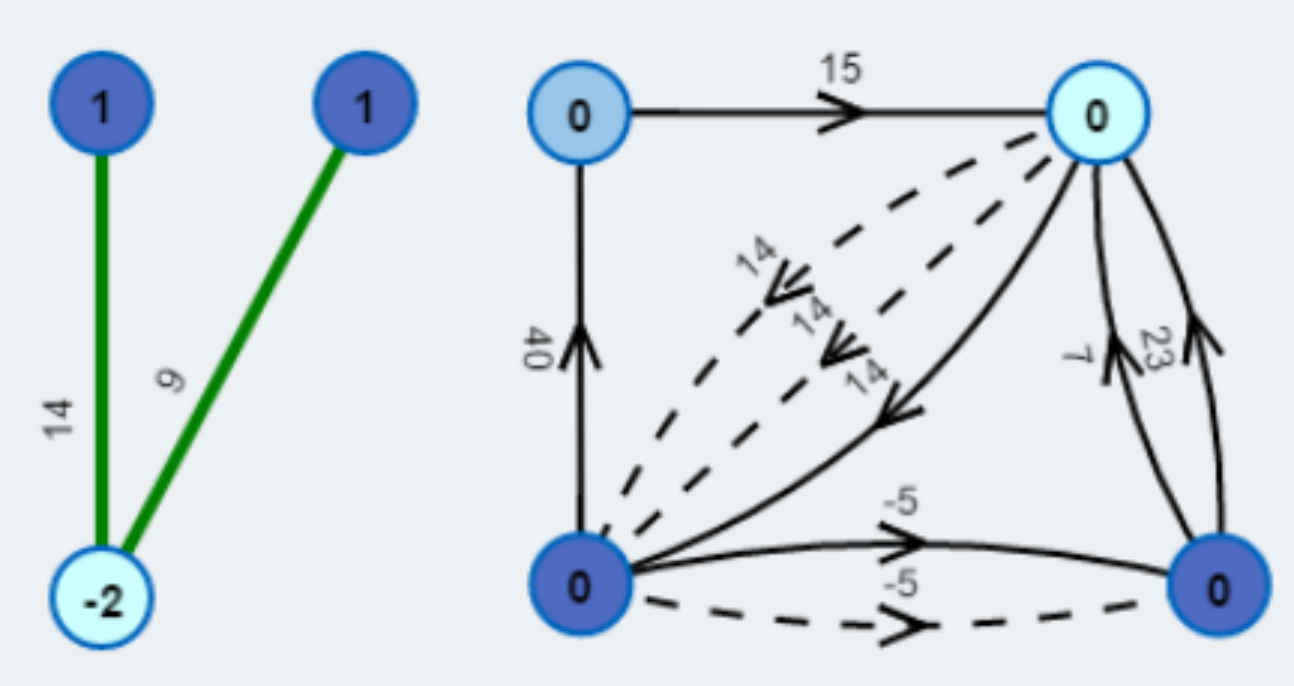
Chinese-Postman

- Differenz der Ausgangs- und Eingangsgrade
  - bestimmt die Anzahl der zusätzlichen Pfaden



Chinese-Postman

- **bipartiter Matchinggraph** erstellt und mit Ungarischer Methode gelöst
  - die optimalen Pfade werden eingefügt



Neuheiten in der Implementierung

LaTeX Formeln

- Weiterführende Algorithmen
- Open-Source MathJAX Bibliothek
- Nutzung der gewohnten Syntax

LaTeX Formeln

Ein alternierender Pfad ist eine Abfolge von Kanten, bei dem die Kanten abwechselnd im Matching M enthalten und nicht enthalten $E \setminus M$ sind (vgl. Bild 5).
Ein alternierender Pfad ist zusätzlich augmentierend, wenn Anfangs- und Endknoten beide frei sind (vgl. Bild 6).

Um das bestehende noch leere Matching zu verbessern, sucht sich der Algorithmus einen Knoten aus X, der noch keinen Matchingpartner besitzt. Ausgehend von diesem, versucht der Algorithmus einen augmentierenden Pfad zu finden. Dazu baut der Algorithmus schrittweise einen alternierenden Pfad, solange bis dieser augmentierend wird oder es keine weitere Kante mehr gibt. Die dabei besuchten Knoten teilen wir in die Mengen S für besuchte Knoten aus X und die Menge T für besuchte Knoten aus Y ein.

Es lässt sich nicht immer ein augmentierender Pfad finden. Dann muss der Algorithmus die Markierungen verbessern, indem er zunächst ein Δ bestimmt. Der Algorithmus betrachtet alle Paare aus besuchten Knoten $s \in S$ und unbesuchten Knoten $y \in Y \setminus T$ und bildet das Minimum aus der Summe ihrer Markierungen abzüglich des Gewichts der Kante zwischen beiden Knoten.

$$\Delta = \min_{s \in S \wedge y \in Y \setminus T} \{l(s) + l(y) - w(s,y)\}$$

Mittels Δ kann man nun die Markierungen aller Knoten anpassen. Danach bestimmt der Algorithmus einen neuen Gleichheitsgraph und sucht erneut nach einem augmentierenden Pfad.

$$l'(v) = \begin{cases} l(v) - \Delta & v \in S \\ l(v) + \Delta & v \in T \\ l(v) & sonst \end{cases}$$

Wenn der Algorithmus einen augmentierenden Pfad findet, verbessert er das Matching, indem er die Kanten des Pfades, die nicht in dem Matching enthalten sind, dem Matching hinzufügt und die Kanten des Pfades, die im Matching enthalten sind, entfernt. Dieses Vorgehen wird wiederholt, bis jeder Knoten einen Matchingpartner hat und das Matching so perfekt ist. Nach dem o.g. Theorem ist das bestimmte Matching dann bezüglich der Kantengewichte ein maximales Matching.

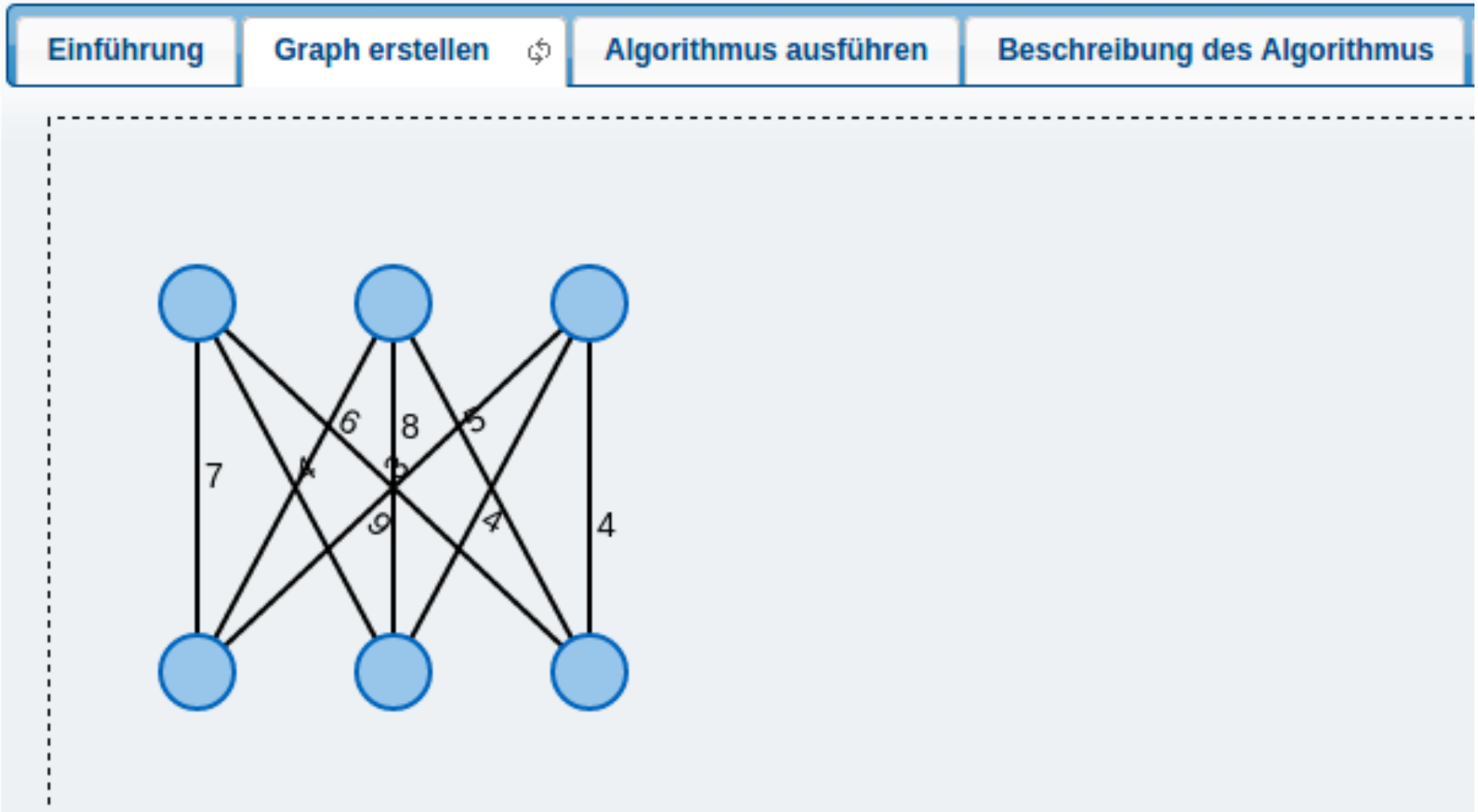
Bipartite Graphen

- **Vorher:** beliebige Graphen
- **Ziele:**
  - Algorithmen auf bipartiten Graphen ausführen
  - Die Darstellung anschaulich machen
  - Weitere Entwicklung vereinfachen

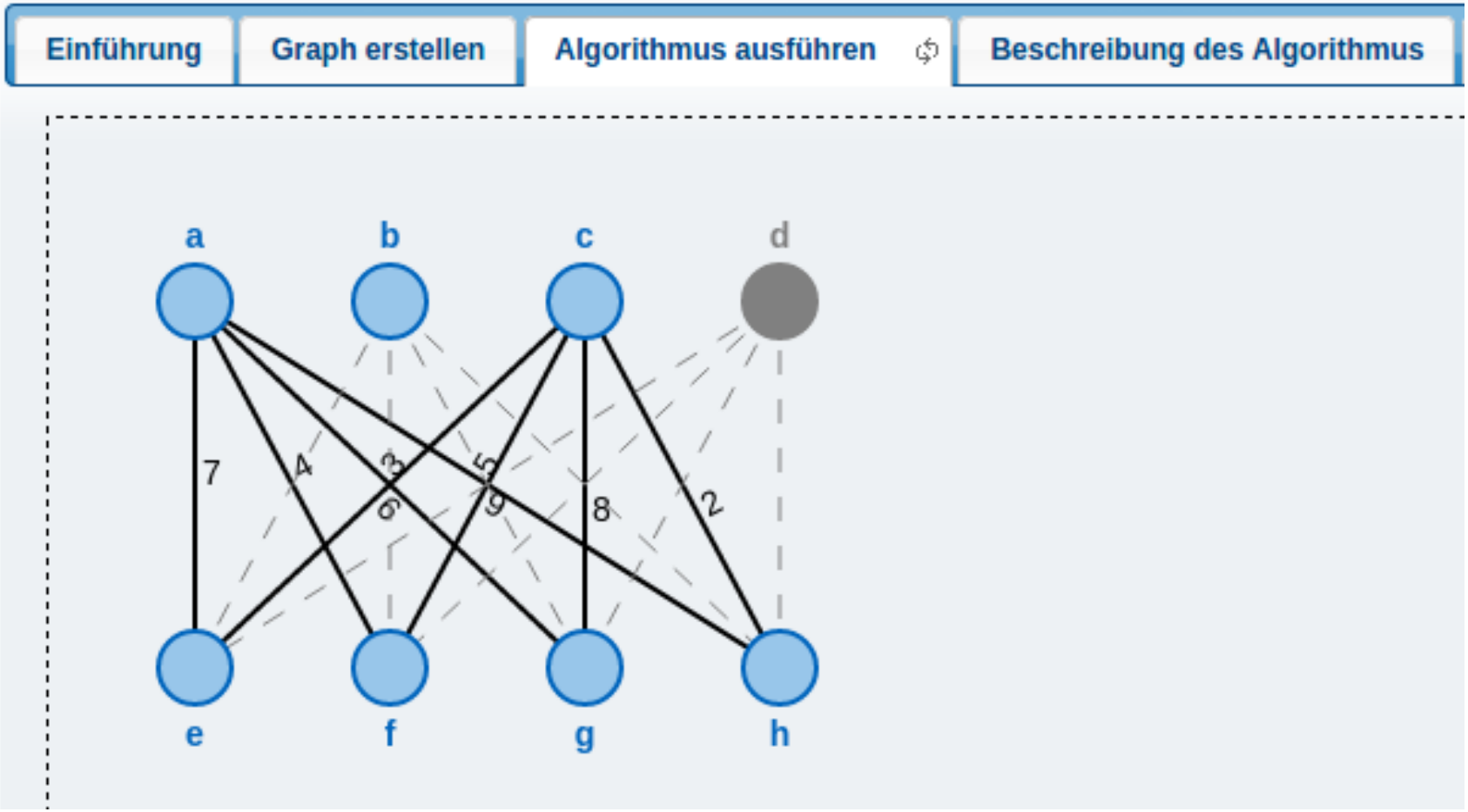
Bipartite Graphen

- Zwei Knotenmengen
- Kanten nur zwischen den Mengen
- Der Graph wird erweitert, damit er vollständig ist und die gerade Anzahl von Knoten enthält (Ungarische Methode)

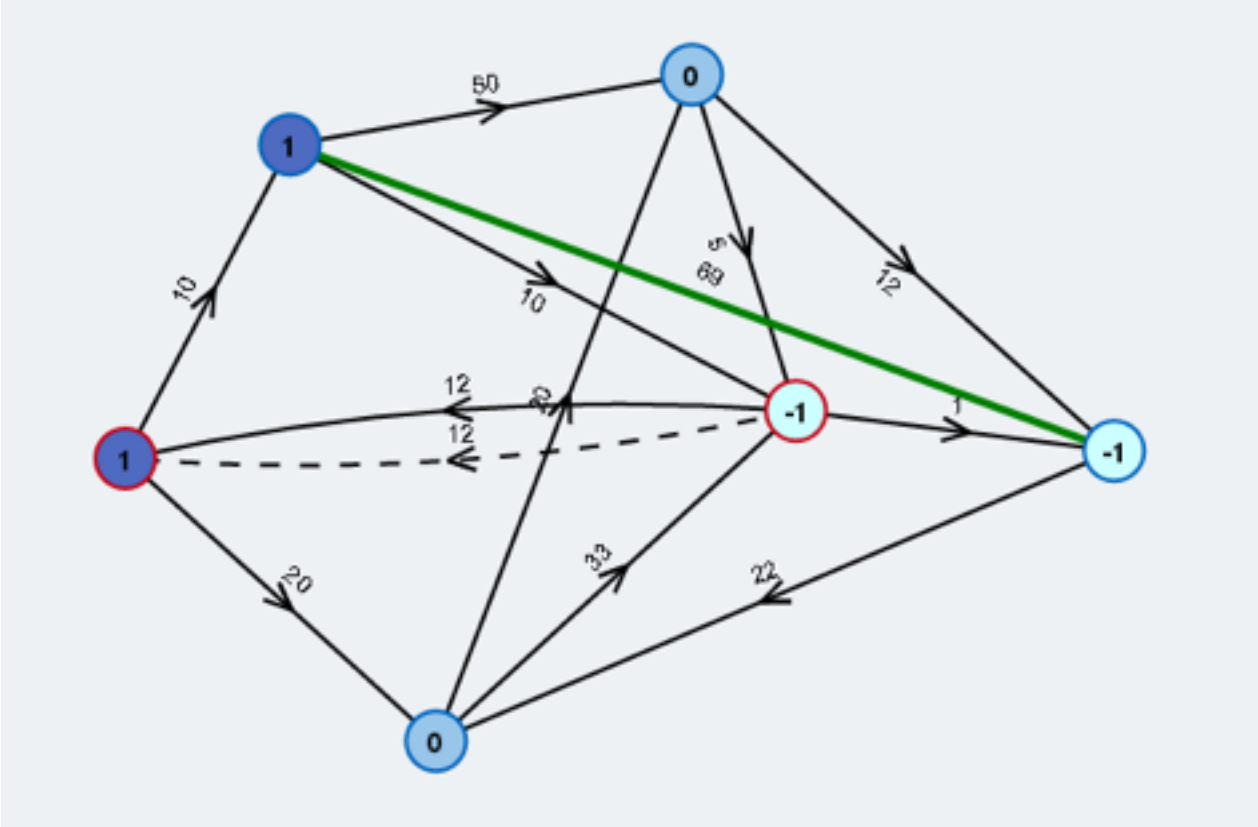
Bipartite Graphen



Bipartite Graphen



Multigraphen & Animationen



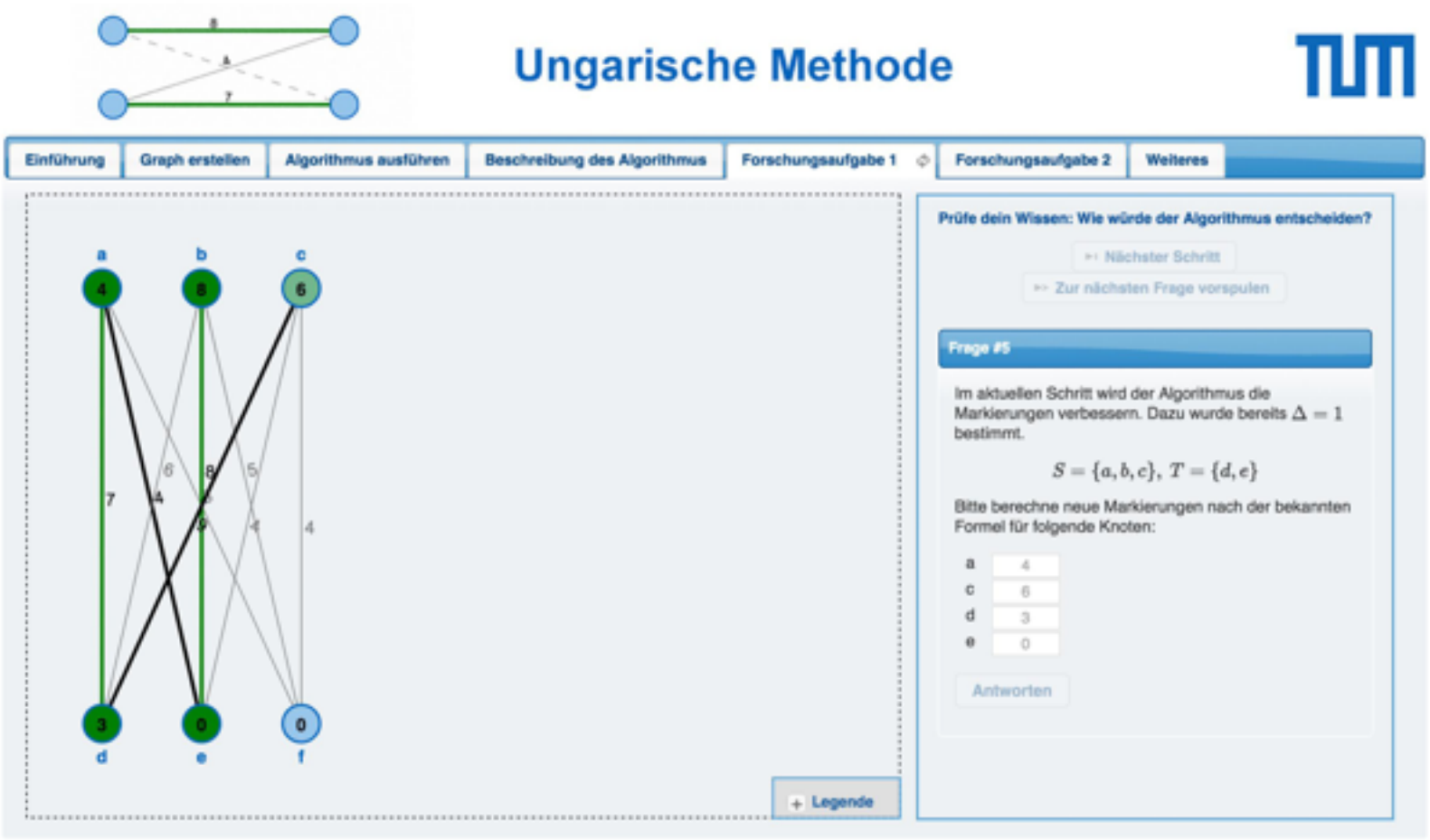
Zufällig generierte Fragen

- **Vorher:** Überwiegend statische Fragen
- **Ziele:**
  - Mehr Freiheiten für den Nutzer
  - Vereinfachungen für den Entwickler

Zufällig generierte Fragen

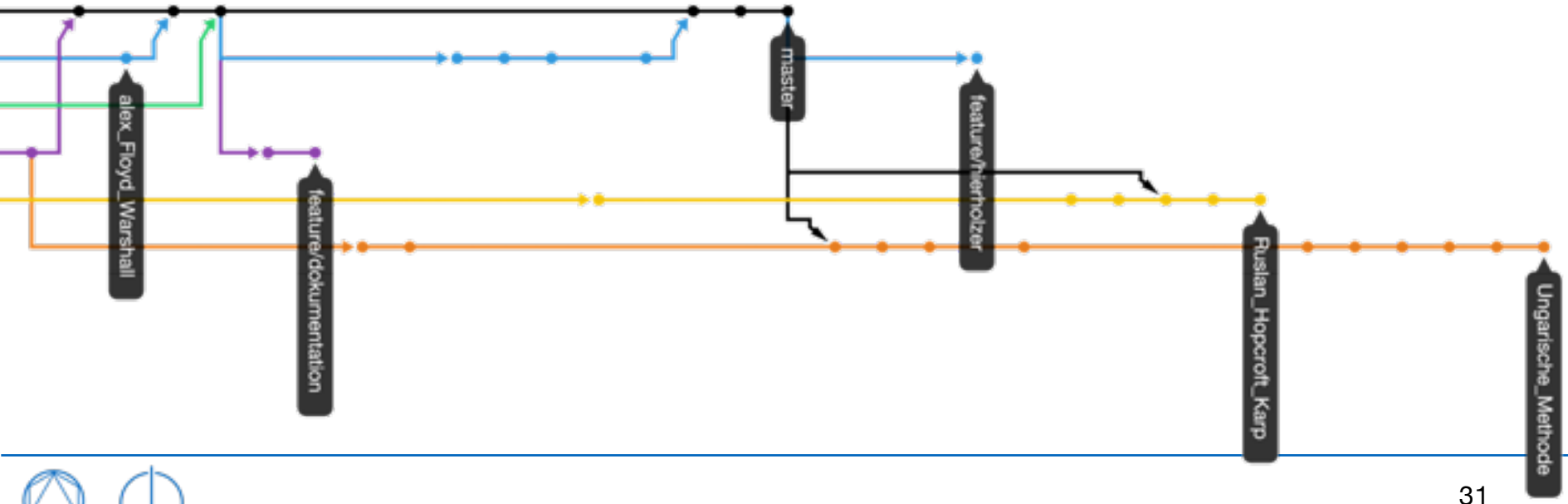
- Selbst erstellter Graph
- Wahrscheinlichkeiten
- Meist zufällige Elemente in der Frage
- **Implementierung:** Code des Algorithmus bleibt unangetastet

Zufällig generierte Fragen



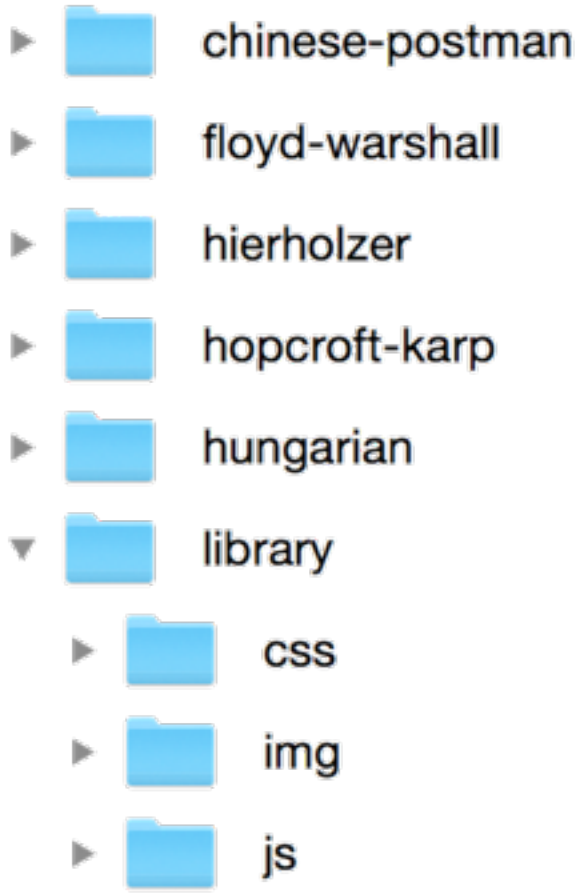
Teamarbeit

- Herausforderung
- Gemeinsames Github Repository
- Orientiert am Gitflow Branching Model



Konzept: Gemeinsame Dateien

- fünf separate Projektordner
- **Ziel:** gemeinsam genutzte Dateien auslagern
  - Layout / Design (Stylesheets + Bilder)
  - Javascript Bibliotheken



Konzept: Gemeinsame Dateien

- Python: difflib

```
------------------------------------
chinese-postman/img/TUMLogo.png
-> 100.00% identical to floyd-warshall/img/TUMLogo.png
-> 100.00% identical to hierholzer/img/TUMLogo.png
-> 100.00% identical to hopcroft-karp/img/TUMLogo.png
-> 100.00% identical to hungarian/img/TUMLogo.png
------------------------------------

------------------------------------
chinese-postman/js/siteAnimation.js
-> 67.78% identical to hierholzer/js/siteAnimation.js
-> 91.49% identical to hopcroft-karp/js/siteAnimation.js
-> 72.42% identical to hungarian/js/siteAnimation.js
------------------------------------
```

Questions / Fragen?