Technische Universität München

Fakultät für Mathematik
Lehrstuhl für Angewandte Geometrie und Diskrete Mathematik

Adaption eines didaktischen Konzepts zur Darstellung weiterführender Graphalgorithmen in einer Web-Applikation

Algorithmus von Floyd-Warshall
Algorithmus von Hierholzer
Algorithmus von Hopcroft und Karp
Ungarische Methode
Chinese Postman Problem

Abschlussbericht für ein interdisziplinäres Projekt von
Mark-Johannes Becker, Aleksejs Voroncovs, Ruslan Zabrodin

Themensteller:	Prof. Dr. Peter Gritzmann
Betreuer:	M.Sc. Wolfgang Ferdinand Riedl
Abgabedatum:	15. Mai 2015

Hiermit erkläre ich, dass ich diese Arbeit selbstständig angefertigt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Mai 2015

Mark-Johannes Becker, Aleksejs Voroncovs, Ruslan Zabrodin

Abstract

Here we give a short summary of the project or thesis of length at most a quarter of a page.

Zusammenfassung

Hier schreibt man eine kurze Zusammenfassung der Arbeit im Umfang von maximal einer Viertelseite.

Inhaltsverzeichnis

1 Einleitung **1**

2 Aufbau der Webanwendungen **3**
2.1 Algorithmus von Floyd-Warshall . . . 3
2.2 Algorithmus von Hierholzer (Mark) . . . 3
2.3 Algorithmus von Hopcroft und Karp . . . 3
2.4 Ungarische Methode . . . 3
2.5 Chinese Postman Problem . . . 4

3 Implementierung **5**
3.1 Installation (Mark) . . . 5
3.2 MathJAX (Mark) . . . 5
3.3 Bipartite Graphen . . . 7
3.4 Multigraphen . . . 7
3.5 Zufällig generierte Fragen (Mark) . . . 7
3.6 Gemeinsam genutzte Dateien (Mark) . . . 9

4 Zusammenfassung **11**

Abbildungsverzeichnis **13**

Index **13**

Literatur **15**

Kapitel 1

Einleitung

Einleitung Motivation Referenz auf führere IDPs (Quellen!) didaktisches Konzept, kurz Aufbau der Dokumentation, inkl. Benennung wer hat was gemacht (Benotung)

Kapitel 2

Aufbau der Webanwendungen

funktionale Beschreibung der Apps jedes Tab

2.1 Algorithmus von Floyd-Warshall

Besonderheiten Visualisierung Forschungsfragen

2.2 Algorithmus von Hierholzer (Mark)

Beim Algorithmus von Hierholzer war es uns wichtig, die resultierende Eulertour verständlich zu visualisieren. Dazu werden zunächst alle Subtouren in unterschiedlichen Farben dargestellt. Einzelne Subtouren können auf der Ergebnisseite einzeln hervorgehoben werden. Der Benutzer erkennt mittels der Farben außerdem, welcher Teil der Tour aus welcher Subtour stammt. Die Tour kann durch eine Animation (vgl. Abbildung 2.1) in der richtigen Reihenfolge abgelaufen werden, sodass der Nutzer verifizieren kann, dass es sich wirklich um eine Eulertour handelt.

Die erste Forschungsaufgabe soll sicherstellen, dass der Nutzer die wichtigsten Abläufe des Algorithmus versteht. Sie behandelt daher im Wesentlichen den Ablauf des Algorithmus, das Konzept des Knotengrads und das Verbinden von Touren.

In einer weiteren Forschungsaufgabe werden die Besonderheiten des Algorithmus bei Anwendung auf einen gerichteten Graphen betrachtet. Der Nutzer betrachtet die geänderten Voraussetzungen, die der Graph erfüllen muss und die veränderte Auswahl der Kanten zur Konstruktion von Subtouren.

2.3 Algorithmus von Hopcroft und Karp

Besonderheiten Visualisierung Forschungsfragen

2.4 Ungarische Methode

Besonderheiten Visualisierung Forschungsfragen

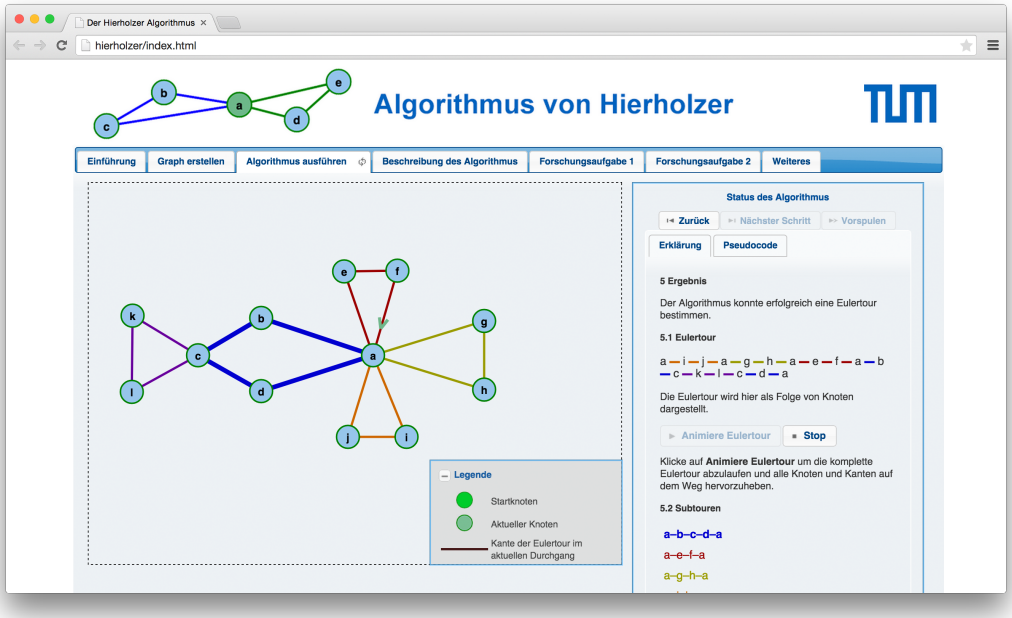

Abbildung 2.1: Visualisierung der Eulertour mittels Animation

2.5 Chinese Postman Problem

Besonderheiten Visualisierung Forschungsfragen

Kapitel 3

Implementierung

3.1 Installation (Mark)

Zur Installation sind keine speziellen Anforderungen zu erfüllen. Die Webapplikationen wurden vollständig mittels HTML5, JavaScript und CSS implementiert, sodass keine zusätzliche serverseitige Software benötigt wird. Zum Aufrufen der Anwendungen wird ein moderner Webbrowser benötigt.

Die Bereitstellung erfolgt über den Online Dienst GitHub, der das verteile Versionskontrollsystem Git benutzt. Alle Anwendungen liegen in einem gemeinsamen Repository, welches unter `https://github.com/herzog31/adv-graph-algorithms` erreichbar ist. Zur Installation kann man entweder auf der Repository Seite die letzte freigegebene Version (Release) als Zip oder Tar Archiv herunterladen oder die aktuellste Version des Repositories mittels folgendem Befehl in das aktuelle Verzeichnis kopieren.

```
1 git clone -b master https://github.com/herzog31/adv-graph-↷
    ↪ algorithms.git
```

Abbildung 3.1: Befehl zum Kopieren des GitHub Repositories

Um die Anwendungen als Webanwendungen online bereitzustellen wird ein Webserver benötigt. Hier emfiehlt sich die Installation des Apache oder nginx HTTP Servers.

Die lokale Ausführung ist ohne einen Webserver möglich. Verschiedene Webbrowser besitzen allerdings eine Sicherheitsrichtlinie, die das Öffnen von lokalen Dateien über JavaScript verbieten. Diese Sicherheitsrichtline kann jedoch durch spezielle Einstellungen umgangen werden. Für Google Chrome ist dies über den Start Parameter `-allow-file-access-from-files` möglich.

3.2 MathJAX (Mark)

Unter den Tabs „Beschreibung des Algorithmus“ werden die komplizierten Algorithmen, wie bspw. die Ungarische Methode, in möglichst einfachen Worten als Fließtext erklärt. Wir gehen davon aus, dass der Nutzer während der Bearbeitung der Forschungsaufgaben,

verschiedene Sachverhalte in der Algorithmenbeschreibung nachschlagen muss. Das wiederholte Lesen von vollständigen Absätzen wollten wir allerdings vermeiden.

Dazu haben wir zusätzlich zur textuellen Beschreibung, wichtige Punkte der Algorithmen auch als mathematische Formeln dargestellt. Der Nutzer erlangt so durch das erste Lesen der Beschreibung ein Grundverständnis der Algorithmen und kann während der Bearbeitung aufkommende Fragen durch die dargestellten Formeln schneller erschließen.

Zur Darstellung der Formeln in den Webapplikationen verwenden wir JavaScript Bibliothek MathJAX[1]. Diese ist frei unter der Apache-Lizenz erhältlich und wird u.a. auch von Wikipedia zur Darstellung jeglicher mathematischer Ausdrücke genutzt. Mit der Bibliothek ist es möglich mit LaTeX beschriebene Ausdrücke als SVG oder PNG im Webbrowser zu rendern.

Damit die Bibliothek in den Webanwendungen genutzt werden kann, muss sie wie folgt eingebunden werden (vgl. Abbildung 3.2.

```
1 <script type="text/javascript" src="../library/js/mathjax/↷
    ↪ MathJax.js?config=TeX-AMS-MML_SVG.js&locale=de"></script>
2 <script type="text/x-mathjax-config">
3   MathJax.Hub.Config({
4     showMathMenu: false,
5     showMathMenuMSIE: false
6   });
7 </script>
```

Abbildung 3.2: Einbinden der MathJAX Bibliothek

Nach der Einbindung werden sämtliche LaTeX Ausdrücke welche sich zwischen den Begrenzungszeichen `\(` und `\)` befinden, automatisch übersetzt. In einem Beispiel wird hier der HTML Code in Abbildung 3.3 von MathJAX im Browser gerendert (vgl. Abbildung 3.4.

```
1 <p style="text-align:␣center;">
2   \(\Delta = \min\limits_{s \in S\ \wedge\ y \in Y \setminus T↷
      ↪ }\{l(s) + l(y) - w(s,y)\}\)
3 </p>
```

Abbildung 3.3: MathJAX Beispiel: HTML Code

Werden Ausdrücke zur Laufzeit in das DOM mittels JavaScript eingefügt, so müssen diese separat übersetzt werden (vgl. Abbildung 3.5).

[1] https://www.mathjax.org/

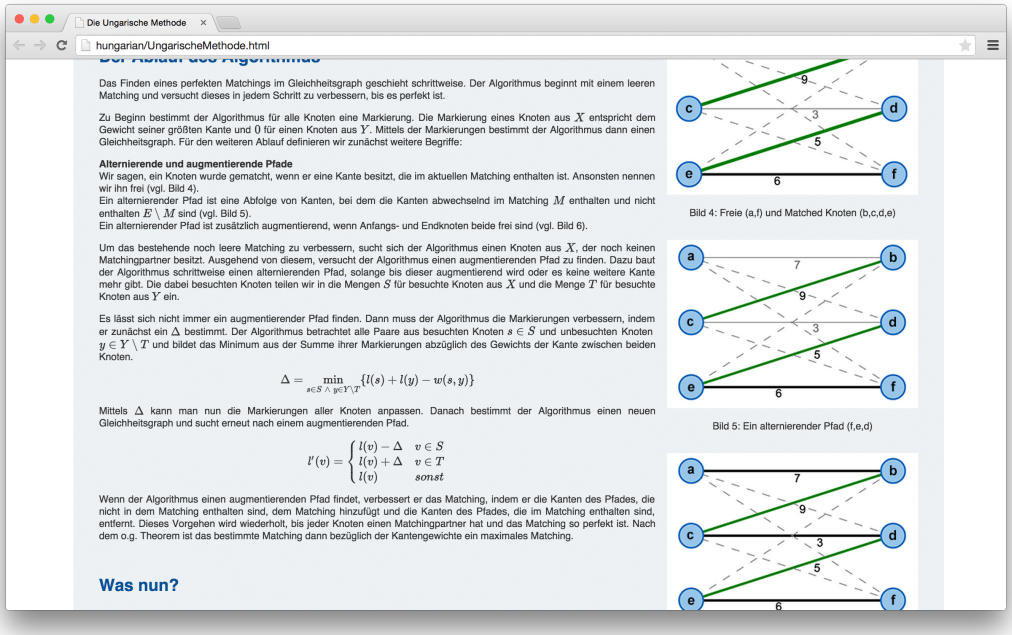

Abbildung 3.4: MathJAX Beispiel: Darstellung im Browser

```
1 MathJax.Hub.Queue(["Typeset", MathJax.Hub]);
```

Abbildung 3.5: Befehl zum erneuten Rendern von Ausdrücken im HTML Dokument

3.3 Bipartite Graphen

Motivation Beispiel

3.4 Multigraphen

Motivation Beispiel

3.5 Zufällig generierte Fragen (Mark)

In bisherigen auf diesem Framework basierenden Projekten war die Varianz der Fragen in den Forschungsaufgaben problematisch. Die Forschungsaufgaben behandelten meistens statische Fragen an vorher festgelegten Stellen zu vorgegebenen Graphen. Daraus

resultierend bietet sich dem Nutzer ein wenig abwechslungsreiches Erlebnis und die Motivation, eine Forschungsaufgabe mehrfach zu bearbeiten, ist nicht gegeben.

In unserer Implementierung verfolgen wir daher einen anderen Ansatz. Die Forschungsaufgaben sind grundsätzlich so aufgebaut, dass sie die gleiche Struktur und den gleichen Quellcode wie die reguläre Ausführung des Algorithmus benutzen. Das schließt auch den vom Nutzer selbst gewählten Graph aus dem „Graph erstellen“ Tab mit ein. Statt statischen Fragen definiert der Entwickler eine Menge von Fragetypen. Vor jedem Schritt des Algorithmus wird dann anhand einer Wahrscheinlichkeitsmatrix bestimmt, ob zu dem folgenden Schritt eine Frage gestellt wird und wenn ja, welcher Fragetyp gewählt werden soll. Außerdem sollen wenn möglich die Inhalte der generierten Fragen eines Fragetyps variieren. So werden beispielsweise die Knoten, zu denen ein Wert bestimmt werden soll, zufällig gewählt (vgl. Abbildung 3.6).

Mit dieser Implementierung gelingt es, dass in jeder Bearbeitung einer Forschungsaufgabe unterschiedliche Kombinationen von Fragen gestellt werden. Zusätzlich kann der Nutzer durch selbst erstelle Graphen auch Spezialfälle testen.

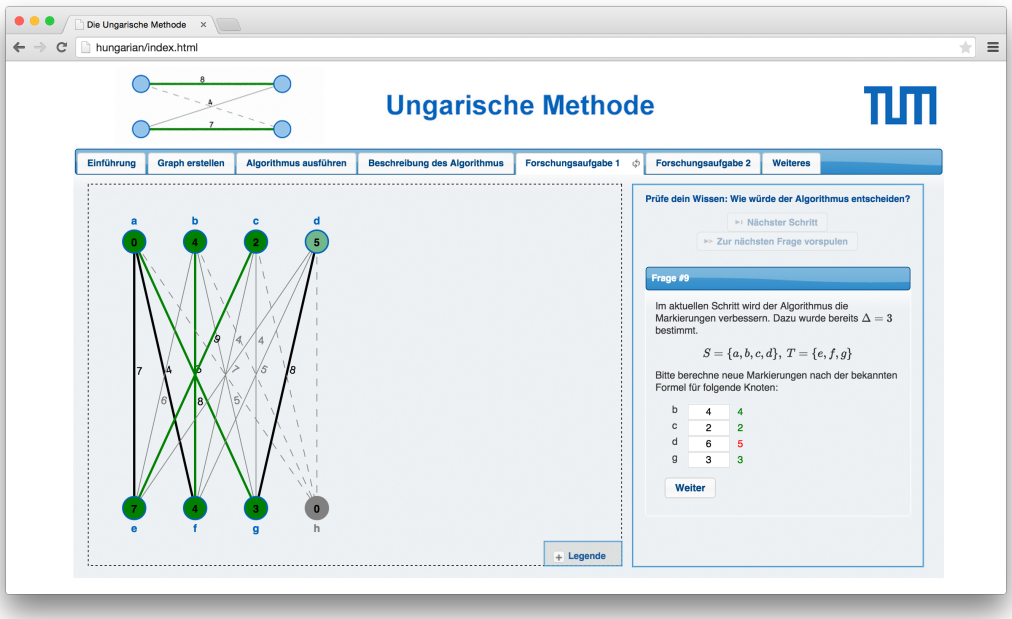

Abbildung 3.6: Beispiel für eine zufällig generierte Frage

Implementierung

In der Funktion `this.nextStepChoice` wird vor der Auswahl des nächsten Schritts des Algorithmus mittels der Funktion `this.askQuestion` bestimmt, ob und welche Frage gestellt werden soll. Für jeden Fragetyp exisiert eine separate Funktion (bspw. `this.generateNextStepQuestion`), die auf den aktuellen Graph zugreift und eine Frage mit zufälligen Elementen generiert und in das DOM Element `#tf1_div_questionModal` schreibt. Die gegebene Antwort wird durch Aufruf der Funktion `this.saveAnswer` gespeichert und mit der richtigen Lösung verglichen. Am Ende der Forschungsaufgabe kann mittels `this.showQuestionResults` eine Übersicht angezeigt werden, die zeigt, wieviele Fragen richtig beantwortet wurden. Weitere Informationen sind der Inline-Dokumentation zu entnehmen.

3.6 Gemeinsam genutzte Dateien (Mark)

Im Rahmen dieses interdisziplinären Projekts wurden Webapplikationen für insgesamt fünf Algorithmen entwickelt. Jede Applikation liegt in einem von den anderen Algorithmen unabhängigen Projektordner, der jeweils auch das Graph Framework aus früheren Projekten enthält. Daher lagen im gesamten Repository viele Dateien mehrfach vor. Dies war insofern problematisch, da Änderungen, die alle Anwendungen betreffen in mehreren Dateien vorgenommen werden mussten.

Unser Ziel war es daher, gemeinsam genutzte Dateien auszulagern. Wir beschränken uns hier auf Dateien, welche das Layout und Design der Anwendungen definieren sowie universelle Programmbibliotheken (bspw. MathJAX). Diese Beschränkung ist notwendig, um die Flexibilität im Entwicklungsprozess zu gewährleisten. Es existieren weitere Dateien, die sich zum aktuellen Zeitpunkt anwendungsübergreifend nicht signifikant unterscheiden. Diese sind allerdings algorithmenspezifisch und könnten in zukünftigen Entwicklungsphasen weiter verändert werden.

Zur Auslagerung legten wir zunächst einen `library` Ordner an. Das Finden von Dateiduplikaten übernimmt ein Python Skript (vgl. `deduplicate.py`), welches mittels der `difflib` Bibliothek alle Dateien in den Projektordnern paarweise miteinander vergleicht und einen Ähnlichkeitsquotienten bestimmt. Das Skript zeigt dann alle Dateien, die einen festgelegten Änhlichkeitsgrenzwert überschreiten an (vgl. Abbildungen 3.7 und 3.8). Anhand des Quotienten lässt sich ablesen, ob Dateien im Gesamten oder in Teilen ausgelagert werden können. Das Verschieben in den `library` Ordner und die Anpassung der Pfade in den Anwendungen erfolgt dann manuell.

Für zukünftige Projekte können die Parameter des Skripts angepasst werden. Weitere Informationen hierzu sind der Inline-Dokumentation zu entnehmen.

```
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
```

Abbildung 3.7: Vollständige Übereinstimmung bei der Datei `TUMLogo.png`

```
chinese-postman/js/siteAnimation.js
-> 67.01% identical to hierholzer/js/siteAnimation.js
-> 91.49% identical to hopcroft-karp/js/siteAnimation.js
-> 71.13% identical to hungarian/js/siteAnimation.js
```

Abbildung 3.8: Mäßige Übereinstimmungen bei der Datei `siteAnimation.js`

Kapitel 4

Zusammenfassung

Zusammenfassung der wesentlichen Punkte

Abbildungsverzeichnis

2.1 Eulertour Animation 4

3.1 Repository Kopieren 5
3.2 MathJAX Einbindung 6
3.3 MathJAX Beispiel Code 6
3.4 MathJAX Beispiel Browser 7
3.5 MathJAX Render Befehl 7
3.6 Zufällig generierte Frage 8
3.7 Gemeinsame Dateien, Beispiel 1 10
3.8 Gemeinsame Dateien, Beispiel 2 10

Literatur

[Tan08] T. Tantau. *The TikZ and PGF Packages. Manual for version 2.00.* Lübeck, Feb. 2008.