Kürzeste Wege

Leitprogrammartige Unterrichtsunterlagen (LPU)

Oliver Probst

23. September 2015

Diese Unterlagen entstanden im Rahmen der Vorlesung

Fachdidaktik Informatik II

im Frühjahrssemester 2015 an der ETH Zürich

unter Betreuung von Jens Maue.

Inhaltsverzeichnis

Teil I.

Konzeption der Unterrichtseinheit

Wir beginnen diesen Teil mit der Analyse der neuen Begriffe und deren Einordnung in das vorhandene Begriffsnetzwerk zur Unterrichtseinheit mit dem Thema *Kürzester Weg*. Im darauf folgenden Kapitel 2 präsentieren wir das Drehbuch und gehen in Kapitel 3 auf das Vorwissen der Schülerinnen und Schüler (SuS) ein. Wir schliessen den Teil mit der Auflistung der Lernziele nach dem Zielebenenmodell in Kapitel 4 ab.

Kapitel **1**

Analyse des Unterrichtsinhaltes

Wir erläutern in diesem Abschnitt zunächst die Fachbegriffe und die Art und Weise, wie sie eingeführt werden, und geben anschliessend eine inhaltliche Abgrenzung an.

1.1. Fachbegriffe

Allgemein führen wir den neuen Stoff dann ein, wenn er benötigt wird. Dadurch verringern wir den Extraneous Load Faktor (da wir „Zusammengehöriges zusammen präsentieren“) was gemäss der Cognitive-Load-Theory [SAK11] sinnvoll ist, damit die Kapazität des Arbeitsgedächtnisses möglichst nicht überschritten wird.
Die Leitprogrammartigen Unterrichtsunterlagen (LPU) knüpfen an den für die SuS bereits bekannten Begriff des (ungerichteten) Graphen (siehe Abschnitt 3.2), bestehend aus Knoten und Kanten, an und führen im ersten Kapitel das Konzept eines **bewerteten Graphen** [OW02] ein. Hierbei wird ein bewerteter Graph als ein Graph mit einer **Bewertungsfunktion**, die jeder Kante **Kosten** zuweist, eingeführt. Anschliessend führen wir den Begriff **Distanzgraph** ein, womit wir uns auf bewertete Graphen mit positiven Kosten beschränken. Alle nachfolgenden Erläuterungen beziehen sich stets auf zusammenhängende Distanzgraphen. Diese Begriffe legen das Fundament für die Betrachtung verschiedener Optimierungsprobleme und das Konzept eines bewerteten Graphen ist somit der erste zentrale Begriff in diesen LPU. Dadurch können wir nun das **Single-Pair-Shortest-Path-Problem** (SPSP-Problem) als einen Repräsentanten eines **Optimierungsproblems** betrachten. In diesem Problem wird der kürzeste Weg zwischen einem gegebenen Paar von Knoten gesucht. Dabei führen wir das Konzept eines **Weges** als ein Tupel von Knoten ein und zeigen auf, wie man die **Länge** eines Weges berechnet. Wir definieren nun den **kürzesten Weg** als einen weiteren zentralen Begriff zwischen zwei Knoten mit Hilfe der **Distanz**. Die Distanz zwischen zwei Knoten wird als das Minimum über die Länge aller möglichen Wege zwischen diesen Knoten definiert. Dabei führen wir die Begriffe **Anfangsknoten** (englisch source) und **Endknoten** für das gegebene Paar von Knoten für das SPSP-Problem ein. Wir formalisieren ein Optimierungsproblem abschliessend durch ein 4-Tupel bestehend aus einer Eingabe, der Menge der zulässigen Lösungen, der Bewertungsfunktion sowie dem Optimierungsziel [Hro04].
Nun sind wir bereit, den **Algorithmus von Dijkstra** als einen Vertreter eines Graphenalgorithmus, zur Lösung des SPSP-Problems zu besprechen[Dij59]. Der Algorithmus kann ausser-

dem dazu verwendet werden, um die kürzesten Wege von einem gegebenen Anfangsknoten zu allen anderen Knoten zu ermitteln. Somit können wir auch das Single-Source-Shortest-Path-Problem (SSSP-Problem) mit diesem Algorithmus lösen. Weiter gehen wir darauf ein, dass das SPSP-Problem ein Unterproblem für das SSSP-Problem ist. Wir erläutern die grundlegende Idee des Algorithmus mit Hilfe der Berechnung von **optimalen Teillösungen** (Optimalitätsprinzip gemäss [OW02]). Abschliessend analysieren wir die Korrektheit sowie die Laufzeit des Algorithmus und erläutern das gierige (**greedy**) Prinzip des Algorithmus.

1.2. Concept-Map

Eine Concept-Map ist in Abbildung 1 dargestellt. Diese Concept-Map stellt die Überlegungen bezüglich der Beziehung der Fachbegriffe untereinander dar. Die Fokusfrage für diese Concept-Map lautet: „Wie berechne ich einen kürzesten Weg?“. Bereits bekannte Begriffe sind grau hinterlegt, neue Begriffe haben einen blauen Hintergrund. Die zentralen Fachbegriffe sind fettgedruckt und stellen jeweils einen Meilenstein in der Bearbeitung der Unterlagen dar.

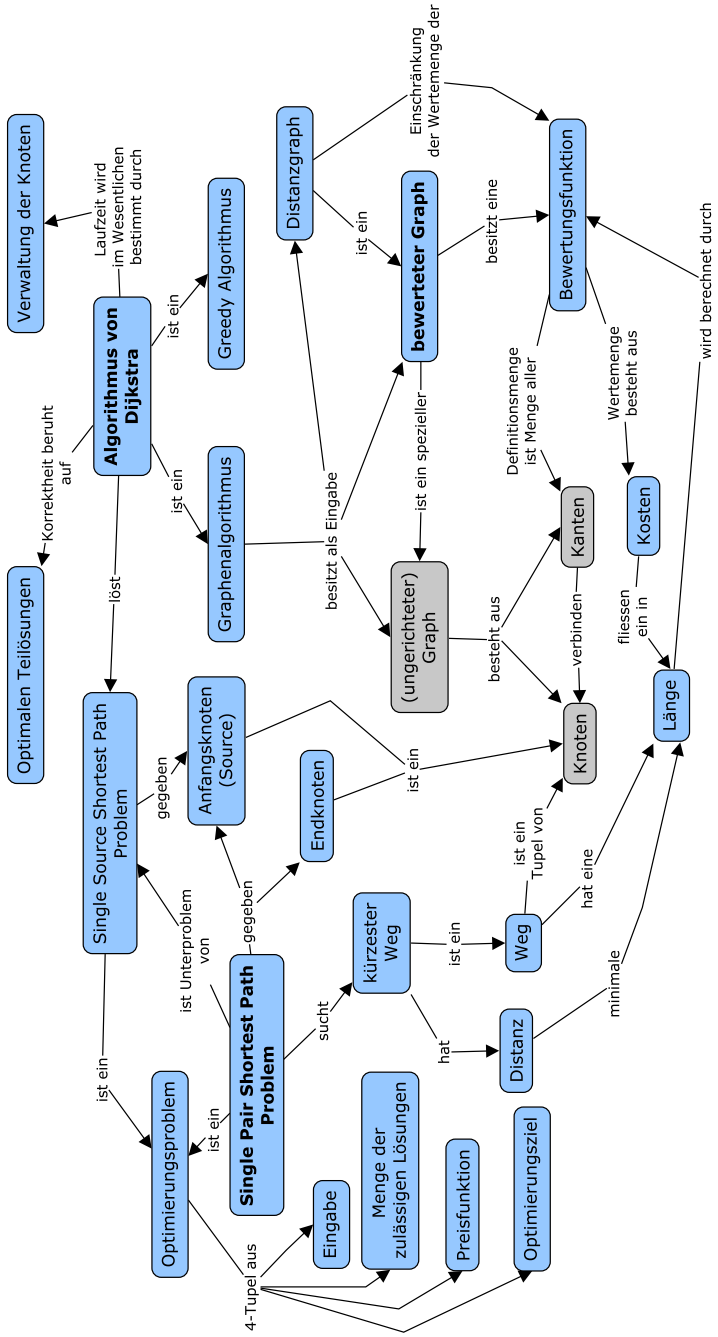

Abbildung 1.: Concept Map mit der Fokusfrage „Wie berechne ich einen kürzesten Weg?“.

1.3. Inhaltliche Abgrenzung

Das Thema *Kürzester Weg* ermöglicht die Betrachtung zahlreicher Aspekte der Informatik. Wir möchten in diesem Abschnitt eine inhaltliche Abgrenzung der behandelten Themen der LPU angeben, indem wir kurz schildern, welche Inhalte im Bezug auf das Thema nicht Bestandteil der LPU sind. Wir geben jeweils ein Begründung an, warum der jeweilige Inhalt *nicht* in die LPU aufgenommen wurde.

- **Gerichtete Graphen**: Der Algorithmus von Dijkstra kann auch auf gerichtete Graphen angewendet werden. Man muss hierbei jedoch zum Beispiel noch den Fall behandeln, falls der Graph nicht stark zusammenhängend ist. Für ungerichtete Graphen ist es ausreichend zu fordern, dass der gegebene Graph zusammenhängend ist, um den Fall zu vermeiden, dass von einem gegebenen Knoten v kein Weg existiert zu einem Knoten u. Wir denken, dass die Behandlung von ungerichteten Graphen ausreichend ist um den Algorithmus von Dijkstra zu studieren. Die Behandlung von Spezialfällen ist ebenfalls ein wichtiger Punkt beim Entwurf eines Algorithmus, steht jedoch aus Gründen des beschränkten Zeitumfanges dieser LPU nicht im Fokus.

- **Zyklen und Pfade**: Kürzeste Wege sind zyklenfrei, falls alle Kosten strikt positiv sind und somit kürzeste Pfade. Für Graphen mit negativen Zyklen gibt es stets einen Anfangsknoten und einen Endknoten, so dass kein kürzester Pfad gefunden werden kann. In diesen LPU werden aus Gründen der Kompaktheit nur Distanzgraphen betrachtet. Wir finden die Unterscheidung zwischen kürzesten Wegen und Pfaden als ein nicht zentrales Thema für diese LPU, da wir die Vermittlung der Zyklenfreiheit in diesem Kontext für ungeeignet erachten. Eine Einführung in das Konzept der Zyklen und Pfade finden wir im Zusammenhang mit Bäumen besser platziert.

- **Algorithmenvergleiche**: Es gibt verschiedene Varianten zur Berechnung eines kürzesten Weges. Unter anderen der Algorithmus von Bellman und Ford [Bel58], [FJ56], die Bidirektionale Suche [Poh71] oder der A*-Algorithmus [HNR68]. Der erste Algorithmus löst die Problematik der negativen Kosten, hat jedoch eine schlechtere Worstcase Laufzeit als der Algorithmus von Dijkstra. Die bidirektionale Suche ermittelt den kürzesten Weg mittels zwei parallelen Suchen. Der letzte Algorithmus ist eine Verallgemeinerung des Algorithmus von Dijkstra. In diesen LPU steht das SPSP-Problem im Zentrum. Wir möchten dessen effiziente Lösbarkeit mit Hilfe eines ausgewählten Algorithmus zeigen und verzichten deshalb auf den Vergleich verschiedener Algorithmen.

- **Implementation von Graphen**: Für Graphenalgorithmen ist eine geschickte Speicherung der Graphen in Form einer Adjazenzliste bzw. einer doppelt verketteten Pfeilliste massgebend für die Laufzeit sowie den Speicherbedarf. Dieses Thema kann zum Beispiel im Rahmen einer allgemeinen Einführung in das Thema *Graphen* behandelt werden. Wir setzen dies nicht voraus, jedoch sind wir der Meinung, dass eine Behandlung dieses Themas auch bei der Bearbeitung dieser LPU von Vorteil ist. Die SuS haben dadurch ein

besseres Verständnis dafür, dass die Ausführung eines Algorithmus auf einem Graphen einfacher ist und somit auch die Ausführung auf einem bewerteten Graphen im Vergleich zu einer Strassenkarte als Bilddatei.

- **Algorithmusimplementation**: Dieses LPU ist konzipiert für zwei bis drei Lektionen. Wir legen den Fokus in diesen Lektionen nicht auf die Implementation, um genügend Zeit für die Diskussion des Algorithmus von Dijkstra zu haben sowie die damit verbundenen Konzepte der Graphentheorie. Es wäre jedoch durchaus denkbar, in anschliessenden Lektionen die Umsetzung des Algorithmus in einer höheren Programmiersprache zu behandeln. Eine Implementation des Algorithmus setzt die Implementation von Graphen voraus, die ebenfalls nicht Teil dieser LPU sind (siehe *Implementation von Graphen*).
- **Laufzeitanalyse und Korrektheit**: Eine Betrachtung der Laufzeit sowie die Korrektheit des Algorithmus gehören zu den wesentlichen Bestandteilen, wenn man einen Algorithmus analysiert. Die Laufzeit für den Algorithmus von Dijkstra variiert je nach der verwendeten Datenstruktur (zum Beispiel binärer Heap versus Fibonacci-Heap). Wir verzichten auf eine detaillierte Laufzeitanalyse unter der Verwendung der Gross-Oh-Notation, da wir der Meinung sind, dass dies eine ordentliche Einführung in die Thematik der Laufzeitanalyse benötigt, sowie Erfahrung im Umgang mit einfacheren Beispielen (zum Beispiel typische Analysen von Laufzeiten bei geschachtelten Schleifen). Wir versuchen jedoch die Laufzeit auf einem für SuS im Gymnasium angepassten Niveau zu diskutieren. Die entsprechende Abgrenzung gilt auch für die Korrektheit des Algorithmus. Wir werden keinen formalen Beweis präsentieren, sondern auf intuitiver Ebene die Korrektheit des Algorithmus besprechen (unter anderem unter der Verwendung des Optimalitätsprinzip gemäss [OW02]).

Kapitel 2

Drehbuch

In diesem Abschnitt werden wir grob den Aufbau der LPU und den damit verbundenen Unterrichtsaufbau erläutern. Die Gestaltung der Lektionen ist dadurch nicht festgelegt, lediglich deren Inhalte und deren Reihenfolge, die sich an [Gal12], [Bar13] und [SS08] orientieren.
Das erste Kapitel der LPU widmet sich der Einführung der bewerteten Graphen und welche Schritte man bei der Modellierung von konkreten Probleme unternehmen muss. Dabei wird sowohl auf eine formale Einführung der bewerteten Graphen Wert gelegt, aber auch auf die korrekte Modellierung von praktischen Problemstellungen mit Hilfe von bewerteten Graphen (Abstraktion).
Das zweite Kapitel widmet sich ganz dem SPSP-Problem und dem Algorithmus von Dijkstra. Die neuen graphentheoretischen Konzepte werden eingeführt und der Algorithmus Schritt für Schritt besprochen. Nachdem der Algorithmus eingeführt wurde, wird erläutert, dass wir damit eine Lösung für eine Instanz des SSSP-Problems erhalten. Dabei wird auf die Beziehung zwischen einer Instanz des SPSP-Problems und einer Instanz des SSSP-Problems für denselben Anfangsknoten eingegangen. Dazu werden verschiedene Problemstellungen, welche alle mit dem Algorithmus von Dijkstra lösbar sind, jedoch einen unterschiedliche Kontext besitzen (zum Beispiel die kürzeste Route gemessen in Kilometern oder die schnellste Route gemessen in Minuten), vorgestellt.
Das letzte Kapitel widmet sich der Analyse des Algorithmus von Dijkstra. Es werden Korrektheit und Laufzeit diskutiert sowie das gierige Prinzip des Algorithmus.

Kapitel **3**

Vorwissen der Schülerinnen und Schüler

Der folgende Abschnitt beschreibt unsere Annahmen bezüglich des Vorwissens der SuS. Wir nennen zunächst die institutionellen Rahmenbedingungen und gehen anschliessend auf die bereits behandelten Unterrichtsthemen ein.

3.1. Institutionelle Rahmenbedingungen

Die LPU richtet sich an Schülerinnen und Schüler, welche die Ausbildung an einem Gymnasium mit mathematisch/naturwissenschaftlichem Schwerpunkt absolvieren. Es wird davon ausgegangen, dass diese LPU in einem zweijährigen Ergänzungsfach Informatik im zweiten Quartal (des ersten Jahres) eingesetzt werden.

3.2. Vorausgesetztes Wissen

Wir nennen zunächst diejenigen Themen, von denen wir ausgehen, dass diese bereits im Unterricht behandelt wurden. Falls das Wissen aus einem anderen Unterrichtsfach stammt, geben wir dies an. Für das vorhandene Wissen geben wir auch jeweils die Tiefe an, die wir voraussetzen. Anschliessend weisen wir auf das zu aktivierende Vorwissen hin.

- Der Begriff *Algorithmus* wurde eingeführt. Die SuS können mit eigenen Worten erklären was ein Algorithmus ist. Sie können dabei an einem Beispiel für einen Algorithmus die Eingabe und Ausgabe nennen. Ausserdem gehen Sie auf die Begriffe Korrektheit und Laufzeit ein. Dabei wurde auf die Begriffe *Problem* (insbesondere Entscheidungsproblem) und *Probleminstanz* eingegangen. Die SuS können ein Problem nennen und eine dazu passende Probleminstanz.
- Die SuS haben das Konzept eines *ungerichteten Graphen* bereits kennengelernt. Sie kennen die Begriffe *Knoten* und *Kante*. Sie sind in der Lage, einen Graphen aus seiner Mengendarstellung in die graphische Darstellung zu überführen und umgekehrt.
- Der Begriff einer *Funktion* wurde den SuS bereits im Mathematikunterricht erläutert. Sie können mit den Begriffen Definitionsmenge und Wertemenge umgehen. Die SuS ha-

ben den Begriff des Tupels ebenfalls bereits behandelt, auch das Summenzeichen wurde eingeführt und von den SuS bereits in einigen Aufgaben verwendet.

Wir setzen keine Programmierkenntnisse voraus, da diese LPU keine Themen behandelt welche eine Implementation verlangen (siehe Abschnitt 1.3). Folgendes Wissen sollte vor der Einführung neuer Konzepte nochmals aktiviert werden:

- Da wir nicht davon ausgehen können das die SuS unmittelbar vor der Bearbeitung dieser LPU sich mit Graphen beschäftigt haben, sollte man die Begriffe Graph, Knoten und Kante und deren Beziehung nochmals kurz wiederholen. Sowohl die formale Notation als auch die graphische Darstellung sollte erneut kurz angesprochen werden, damit die SuS wieder Sicherheit erlangen im Zeichnen und Lesen von Graphen. Dies kann je nach Klasse zum Beispiel auch in einem Informierenden Unterrichtseinstieg oder in einem Advance Organizer eingebaut werden.

Bei allen anderen Punkten bezüglich dem vorhanden Vorwissen gehen wir davon aus, dass die SuS bereits genügend Routine im Umgang mit den damit verbunden Konzepten und Begriffen gesammelt haben, dass eine erneute Aktivierung nicht nötig ist. Für das vorhandene Wissen aus dem Ergänzungsfach Informatik stützt sich diese Begründung auf die unmittelbar zuvor gehaltenen Lektionen (Begriff Algorithmus, prozedurale Programmierung aus dem ersten Quartal), wodurch das erlernte Wissen zeitlich noch nicht all zu lange zurückliegt und ein Umgang damit geübt wurde.

Kapitel **4**

Lernziele der Unterrichtseinheit nach dem Zielebenenmodell

In diesem Abschnitt präsentieren wir unsere Leitidee und die daraus abgeleiteten Lernziele (Dispositionsziele und operationalisierte Lernziele) für die Unterrichtseinheit zum Thema „Kürzeste Wege“ bestehend aus drei Lektionen. Die operationalisierten Lernziele sind auf verschiedenen Ebenen der Lernzieltaxonomie von Bloom angeordnet [BE76].

4.1. Leitidee

Routenplaner sind im heutigen Alltag omnipräsent, wenn es zum Beispiel darum geht, die kürzeste Route zwischen zwei Orten zu ermitteln. Ein zentraler Aspekt bei dieser Aufgabe ist die Modellierung der Strassenkarte als einen bewerteten Graphen. Diese Modellierung erlaubt es, die Planung der Route mit Hilfe des Algorithmus von Dijkstra zu automatisieren. Zeitaufwändige Routenplanungen von Hand können damit durch den Computer innerhalb von Sekunden gelöst werden.
Graphen und kürzeste Wege tauchen in vielen Situationen des Alltags auf und bilden eine mögliche Methode, Problemstellungen abstrakt darzustellen. Durch eine abstrahierte Sichtweise können wiederkehrende Probleme automatisiert gelöst werden. Aus diesem Grund behandeln wir graphentheoretische Konzepte und den Algorithmus von Dijkstra, welche die Basis bilden für eine effiziente und korrekte Ermittlung kürzester Wege.

4.2. Dispositionsziele

Wir geben unsere Dispositionsziele konkret genug an um daraus Material und Operationalisierungen ableiten zu können.

- Die SuS sind sich bewusst, dass bei der mathematischen Modellierung eines konkreten Problems, wie zum Beispiel der Routenplanung, einige Faktoren bei der Modellierung wichtig und andere unwichtig sind und somit nicht in die Modellierung mit aufgenommen werden sollten. Sie erlernen die Methode der Abstraktion, die auch im alltäglichen Leben eingesetzt wird (zum Beispiel Übersichtsplan des Tramnetzes).

- Die SuS haben eine Vorstellung davon, welche grundlegenden Algorithmen sich hinter alltäglichen Verfahren (wie zum Beispiel der Routenplanung) verstecken. Sie sind daran interessiert, weitere dieser Basisalgorithmen kennenzulernen.

4.3. Operationalisierte Lernziele

Im Folgenden werden wir die operationalisierten Lernziele definieren, die die SuS nach erfolgreichem Durcharbeiten unserer LPU erreichen sollen. Anschliessend an die Definition der einzelnen Lernziele werden wir ihren konzeptuellen Aufbau grafisch veranschaulichen (Abb. 2). Die Lernziele sollen erreicht werden ohne Zuhilfenahme zusätzlicher Hilfsmittel. Wenn ein Lernziel eine mündliche Erklärung verlangt, ist stets eine Erklärung in eigenen Worten gemeint. Falls nicht erwähnt wird, an wen sich die Erklärung richtet, sollen die Erklärungen in Worten formuliert werden, die für Mitschüler mit denselben Vorkenntnissen verständlich sind. Dies sind die einzigen Bedingungen und sie gelten für alle in Tabelle 4.1 enthaltenen Lernziele.

Operationalisierte Lernziele

Abk.	Lernziel	Taxonomie
L1	Die SuS können das Konzept eines bewerteten Graphen an einem Beispiel (zum Beispiel die Routenplanung) erklären. Sie verwenden dabei die Begriffe *Bewertungsfunktion* und *Kosten*. Ausserdem erwähnen Sie den Begriff *Distanzgraph* als einen speziellen bewerteten Graphen.	Verständnis
L2	Die SuS sind in der Lage, eine geeignete Situation aus dem Alltag (zum Beispiel ein Ausschnitt einer Velokarte) als Distanzgraphen zu modellieren. Sie abstrahieren dabei von der zugrunde liegenden Situation so weit wie möglich. Nicht relevante Faktoren werden nicht in die Modellierungen aufgenommen (zum Beispiel der Verlauf der Strasse).	Anwendung, Analyse, Synthese
L3	Die SuS erläutern anhand des SPSP-Problems was ein Optimierungsproblem ist. Sie gehen dabei auf die Begriffe der Eingabe, der Menge der zulässigen Lösungen sowie auf die Preisfunktion und das betrachtete Optimierungsziel ein.	Verständnis

L4	Die SuS können das SPSP-Problem als eine Suche nach einem kürzesten Weg zwischen einem Anfangs- und Endknoten beschreiben. Weiter gehen sie darauf ein, dass das SPSP-Problem ein Unterproblem des SSSP-Problems ist und eine Lösung für das SSSP-Problem auch eine Lösung für das SPSP-Problem beinhaltet. Dabei können Sie sich auf das Beispiel der Routenplanung zur Auffindung des kürzesten Weges zwischen zwei Städten stützen.	Verständnis
L5	Die SuS kennen die Definition eines Weges als ein Tupel von Knoten.	Wissen
L6	Die SuS können erläutern, was man unter der Länge eines Weges versteht. Sie gehen dabei auf die Summe der Kosten der Kanten im Weg ein.	Verständnis
L7	Die SuS erklären den Begriff des *kürzesten Weges* zwischen zwei Knoten mit Hilfe der Distanz. Dabei verwenden Sie die Definition der Distanz.	Verständnis
L8	Die SuS nennen den Algorithmus von Dijkstra als einen möglichen Algorithmus zur Lösung des SSSP-Problems.	Wissen
L9	Die SuS können für eine konkrete Instanz des SSSP-Problems eine Lösung mit Hilfe des Algorithmus von Dijkstra von Hand ermitteln. Sie notieren dabei die einzelnen Schritte des Algorithmus.	Anwendung
L10	Die SuS können auf intuitiver Ebene die Laufzeit und Korrektheit des Algorithmus von Dijkstra erläutern. Sie gehen dabei auf das zugrunde liegende Optimalitätsprinzip ein (optimale Teillösungen).	Analyse, Bewertung
L11	Die SuS definieren einen Greedy Algorithmus als ein Verfahren, dass einmal gefällte Entscheidungen, die den Algorithmus näher zur Lösung bringen, nur auf Grund der bisher gesammelten Information durchgeführt werden und dann nicht mehr geändert werden.	Verständnis
L12	Die SuS können das gierige Verfahren anhand dem Algorithmus von Dijkstra erläutern. Sie gehen dabei drauf ein, dass der Algorithmus in jeder Iteration stets denjenigen Knoten mit minimaler Distanz zum Anfangsknoten auswählt und diese Wahl nicht mehr revidiert.	Verständnis, Analyse

Tabelle 4.1.: Operationalisierte Lernziele unter Angabe der jeweiligen Stufe auf der Lernzieltaxonomie von Bloom.

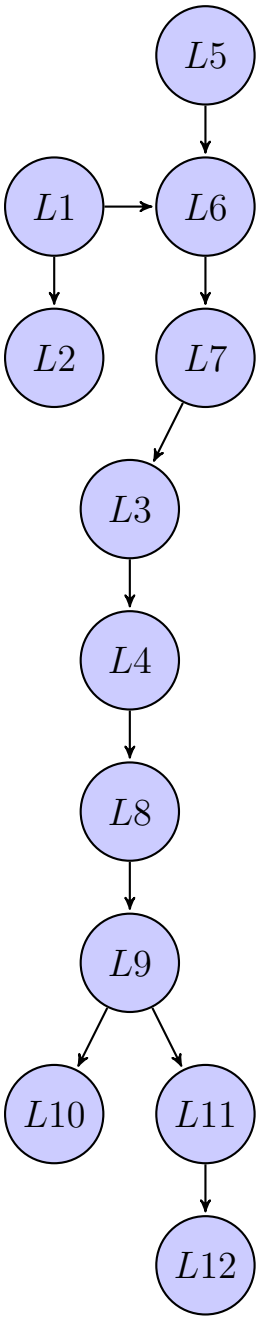

Abbildung 2.: Der Lernzielgraph für die operationalisierten Lernziele aus Abschnitt 4.3

Akürzungsverzeichnis

LPU

Leitprogrammartige Unterrichtsunterlagen 9, 12, 13, 15, 17, 18, 20

SPSP

Single-Pair-Shortest-Path 9, 10, 12, 15, 20, 21, 53–55, 65, 66

SSSP

Single-Source-Shortest-Path 10, 15, 21, 65, 66

SuS

Schülerinnen und Schüler 7, 9, 12, 13, 17–21

Literaturverzeichnis

[Bar13] Barth, Armin P.: *Algorithmik für Einsteiger*. 2. Auflage. Springer Spektrum, 2013

[BE76] Bloom, Samuel B. ; Engelhart, Max D.: *Taxonomie von Lernzielen im kognitiven Bereich*. 5. Auflage. Beltz, 1976 (Beltz Studienbuch)

[Bel58] Bellman, Richard: On a Routing Problem. In: *Quarterly of Applied Mathematics* 16 (1958), S. 87–90

[Dij59] Dijkstra, Edsger W.: A note on two problems in connexion with graphs. In: *Numerische Mathematik* 1 (1959), 12, Nr. 1, S. 269–271

[FJ56] Ford Jr, Lester R.: Network Flow Theory. In: *RAND Corp* Paper P-923 (1956), S. 87–90

[Gal12] Gallenbacher, Jens: *Abenteuer Informatik*. 3. Auflage. Springer Spektrum, 2012

[HNR68] Hart, P.E. ; Nilsson, N.J. ; Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In: *IEEE Transactions on Systems Science and Cybernetics* 4 (1968), July, Nr. 2, S. 100–107

[Hro04] Hromkovič, Juraj: *Algorithmics for Hard Problems, Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*. 2. Auflage. Springer, 2004 (Texts in Theoretical Computer Science. An EATCS Series)

[OW02] Ottmann, Thomas ; Widmayer, Peter: *Algorithmen und Datenstrukturen*. 4. Auflage. Springer Spektrum, 2002

[Poh71] Pohl, Ira: Bi-directional search. In: *Machine Intelligence* 6 (1971), S. 127–140

[SAK11] Sweller, John ; Ayres, Paul ; Kalyuga, Slava: *Cognitive Load Theory*. Springer, 2011 (Explorations in the Learning Sciences, Instructional Systems and Performance Technologies)

[SS08] Sanders, Peter ; Singler, Johannes: Kürzeste Wege. In: Vöcking, Berthold (Hrsg.) ; Alt, Helmut (Hrsg.) ; Dietzfelbinger, Martin (Hrsg.) ; Reischuk, Rüdiger (Hrsg.) ; Scheideler, Christian (Hrsg.) ; Vollmer, Heribert (Hrsg.) ; Wagner, Dorothea (Hrsg.): *Taschenbuch der Algorithmen*. Springer, 2008 (eXamen.press), S. 353–360

Teil II.

Leitprogrammartige Unterrichtsunterlagen

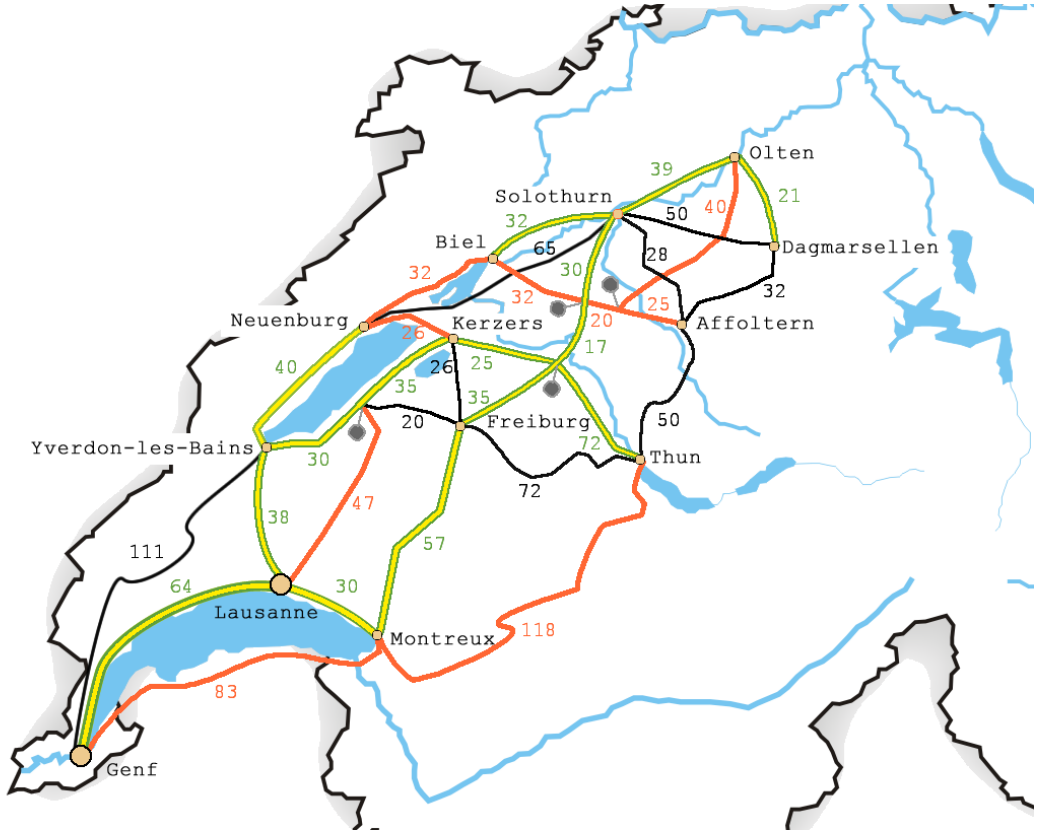

Abbildung 1.: Landkartenausschnitt aus einem Atlas mit ausgewählten Strassen und Städten der Schweiz. Längenangaben in Kilometer.

Wenn wir mit dem Auto in die Ferien fahren ermitteln wir heutzutage häufig die kürzeste Route von unserem Ausgangsort zum Ziel mit einem Routenplaner. Welche Methode steckt dahinter, dass ein Routenplaner in der Lage ist zwischen zwei beliebigen Orten in sehr kurzer Zeit die gewünschte (kürzeste) Route zu ermitteln? In diesen Unterlagen möchten wir uns mit diesem Problem beschäftigen. Betrachten wir den Strassenkartenausschnitt in Abbildung 1, welcher einen Teil des Schweizer Strassennetzes zeigt.

Autobahnen sind gelb-grün eingezeichnet, Hauptstrassen rot und Landstrassen schwarz. In den Städten ist es problemlos möglich von einer Strasse auf die andere zu wechseln. Es gibt auch Strassenkreuzungen die es erlauben, die Strassen zu wechseln, wie zum Beispiel zwischen Biel und Affoltern, jeweils markiert durch eine „Nadel“. Die Länge der Strasse ist in der jeweils passenden Farbe in Kilometern angegeben.
Nehmen wir an, wir wohnen in Olten und planen einen Ausflug mit dem Auto nach Genf um den Völkerbundpalast zu besichtigen. Wie würden Sie fahren um eine möglichst kurze Route,

gemessen in Kilometern, zurückzulegen? Eine Antwort auf diese Frage werden wir Schritt für Schritt in diesen Unterlagen erarbeiten und dabei Verfahren kennenlernen, welche nicht nur für diese Strassenkarte eingesetzt werden können sondern auch für die Beantwortung von weiteren Fragen wie „Was ist die kürzeste Veloroute von Oerlikon zur Roten Fabrik?“ oder „Welches ist die schnellste Zugverbindung von Zürich nach London?“.

Kapitel **1**

Von der Landkarte zum bewerteten Graphen

In diesem Kapitel beschäftigen wir uns damit, wie man die Strassenkarte aus Abbildung 1 in eine Form bringt, sodass nur die wesentlichen Informationen erhalten bleiben. Dazu verwenden wir das mathematische Modell des Graphen und erläutern die wesentlichen Schritte um ein Problem aus der Realität in dieses mathematische Modell zu übertragen. Dies ist eine Vorbereitung auf die Entwicklung eines Algorithmus zur automatisierten Lösung des Problems zur Auffindung der kürzesten Route zwischen zwei Städten.
Am Ende dieses Kapitels sind Sie in der Lage, eine Probleminstanz, wie zum Beispiel eine Strassenkarte, in einen Graphen zu überführen. Sie verwenden dabei sowohl die formale als auch graphische Notation. Zudem erlernen Sie wie man für unser Problem und der damit verbundenen Probleminstanz von wichtigen und unwichtigen Details unterscheidet.

1.1. Modellierung und Abstraktion

Betrachten Sie nochmals die Strassenkarte aus Abbildung 1. Welcher Teil der Karte ist für uns interessant, wenn es darum geht, die kürzeste Route zwischen zwei Städten zu finden? Die Farbe der Strasse spielt dabei zum Beispiel keine Rolle. Wenn es darum geht, einen Teil der realen Welt vereinfacht darzustellen, sprechen wir von ***Modellierung***. Dabei können wir nicht beliebig vereinfachen. Es wäre zum Beispiel nicht sinnvoll, die Stadt Olten in unserem Modell nicht zu berücksichtigen, da wir dann innerhalb von unserem Modell nicht mehr die kürzeste Route von Olten nach Genf ermitteln könnten. Wir müssen also beim Erstellen unseres Modells zwischen wichtigen und unwichtigen Details (Informationen) unterscheiden. Woher wissen wir jedoch was wichtig und was unwichtig ist?
Gehen wir nochmals zur Strassenkarte zurück und betrachten die Information „Farbe der Strasse“. Wenn wir diese Information in unser Modell nicht hinzufügen, können wir die kürzeste Route zwischen zwei Städten immer noch ermitteln, da diese nicht von der Farbe abhängt. Die Farbe der Strasse ist somit eine unwichtige Information wenn wir davon ausgehen, dass wir alle Strassen benutzen dürfen. Würden wir keine Autobahn erlauben (weil wir zum Beispiel keine Vignette kaufen möchten), wäre die Farbe der Strasse eine wichtige Information, da wir dadurch entscheiden ob wir eine Strasse in unser Modell aufnehmen oder nicht (falls es eine Autobahn ist). Dadurch würden wir ein anderes Modell von der Strassenkarte erhalten und

dies beeinflusst die Ermittlung der kürzesten Route ebenfalls. Somit müssen wir die Einteilung in wichtige und unwichtige Informationen stets bezüglich der Problemstellung treffen und die damit verbunden Annahmen. Für unsere Problemstellung können wir die Auswahl einer Information mit der folgenden Frage abdecken:

> *Falls wir die betrachtete Information (z.B. die Strassenfarbe)* ***nicht*** *in unser Modell aufnehmen, können wir dann die kürzeste Route zwischen zwei Städten in unserem Modell immer noch ermitteln?*

Lautet die Antwort „ja“ ist die Information nicht wichtig.

▼ Aufgabe 1.1.

Entscheiden Sie für die folgenden Informationen ob Sie wichtig sind oder nicht, wenn es darum geht die Strassenkarte aus Abbildung 1 zu modellieren. Geben Sie jeweils eine Begründung für Ihre Entscheidung an.

1. *Städtenamen*
2. *Position der Stadt auf der Strassenkarte*
3. *Grösse der Stadt*
4. *Strassenverlauf*
5. *Strassenlänge und Einheit*
6. *Flüsse und Seen*
7. *Strassentyp (zum Beispiel Autobahn oder Landstrasse)*
8. *Strassenverbindungen der Städte untereinander*
9. *Strassenkreuzungen*
10. *Landesgrenze*

Wenn wir bei der Modellierung nur das Wesentliche (Wichtige) berücksichtigen, sprechen wir von ***Abstraktion***. Wir haben in Abbildung 2 die Strassenkarte so dargestellt, dass genau diejenigen Informationen noch vorhanden sind, um das Eingangsproblem noch lösen zu können. Überzeugen Sie sich davon durch einen Vergleich mit der ursprünglichen Karte aus Abbildung 1.

Die Methode der Abstraktion beinhaltet auch noch den Aspekt des Generalisierens. Schauen Sie sich nochmals die zweite Version der Karte an (siehe Abbildung 2). Wir unterscheiden

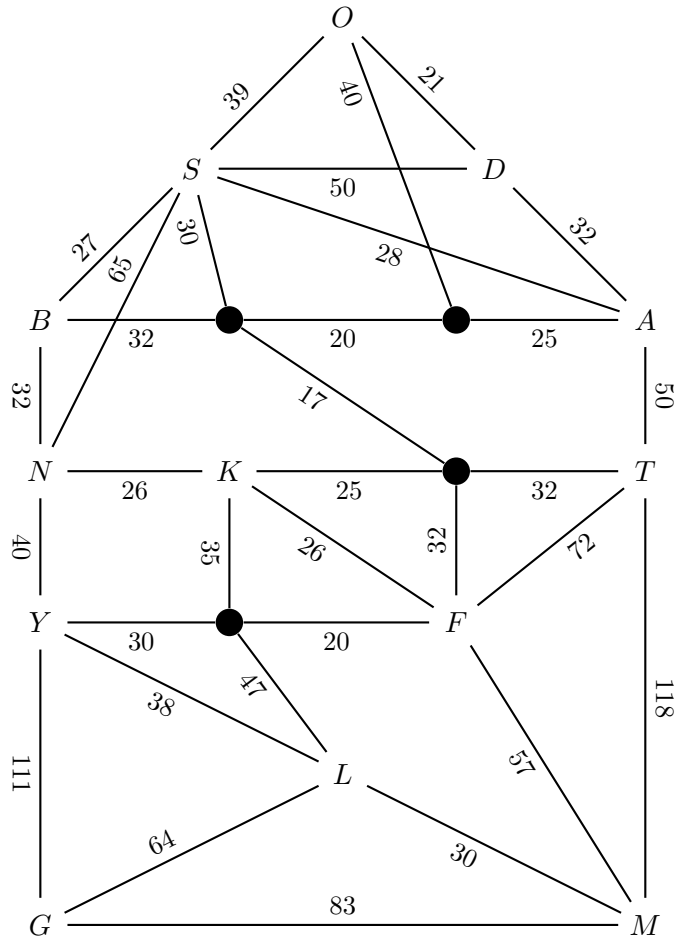

Abbildung 2.: Das erste Modell der Strassenkarte aus Abbildung 1. Die ausgemalten schwarzen Kreise stellen die Strassenkreuzungen dar. Die Buchstaben repräsentieren die jeweiligen Städte, abgekürzt mit dem Anfangsbuchstaben.

hier zwischen Städten und Strassenkreuzungen. Können wir diese beiden Konzepte zusammenführen? Falls wir dies schaffen, können wir unser Modell weiter vereinfachen. Dies wäre zum Beispiel hinsichtlich der Umsetzung in ein Programm von Vorteil, da wir dann keine Fallunterscheidung zwischen Städten und Strassenkreuzungen durchführen müssten.

▼ Aufgabe 1.2.

In dieser Aufgabe werden Sie sich weiter vertraut machen mit der Methode der Abstraktion. Bearbeiten Sie die folgenden Fragen nacheinander. Benutzen Sie dazu Abbildung 2 und betrachten Sie den Ort Kerzers (K) sowie die Strassenkreuzung zwischen Kerzers und Lausanne (L) im Detail.

1. *Stellen Sie sich vor, Sie fahren mit einem Auto von der beschriebenen Kreuzung nach Kerzers. Welche Möglichkeiten haben Sie, wenn Sie in Kerzers mit dem Auto ankommen? Stellen Sie sich nun vor, Sie fahren wieder mit dem Auto. Dieses Mal jedoch von Kerzers in Richtung der Kreuzung. Welche Möglichkeiten haben Sie nun, wenn Sie bei der Kreuzung ankommen? Erkennen Sie eine Gemeinsamkeit?*
2. *Gibt es auch einen Unterschied in dem aus Punkt 1 beschriebenen Szenario?*
3. *Können Sie aus die beiden Konzepte der Stadt und Strassenkreuzung zusammenführen? Betrachten Sie hierzu Ihre Unterschiede und Gemeinsamkeiten. Können Sie diese durch eine kleine Modifikation der Strassenkreuzung auflösen?*

In Aufgabe 1.2 haben Sie sich mit der Generalisierung (Verallgemeinerung) der beiden Konzepte „Stadt“ und „Strassenkreuzung“ beschäftigt. Wir können nun im Rahmen der Abstraktion diese beiden Konzepte zusammenführen und sowohl eine Stadt als auch eine Strassenkreuzung als Verkehrsverknüpfungspunkt betrachten, an dem man von einer Strasse auf eine andere wechseln kann. Wir haben das Modell nun weiter abstrahiert und aus jeder Kreuzung ein Verkehrsverknüpfungspunkt erstellt. Das Ergebnis sehen Sie in Abbildung 3.

Für die aus den Städten hervorgegangenen Verkehrsverknüpfungspunkte haben wir die Buchstaben aus den Stadtbezeichnungen übernommen. Falls unsere Route über eine Strassenkreuzung geht, benötigen wir dafür eine Bezeichnung, damit wir die Route exakt beschreiben können. Auch wenn wir die Problemstellung in ein Programm übertragen wollen, benötigen wir eine Beschreibung aller Verkehrsverknüpfungspunkte. Aufgabe 1.3 beschäftigt sich mit der Wahl der Bezeichnungen.

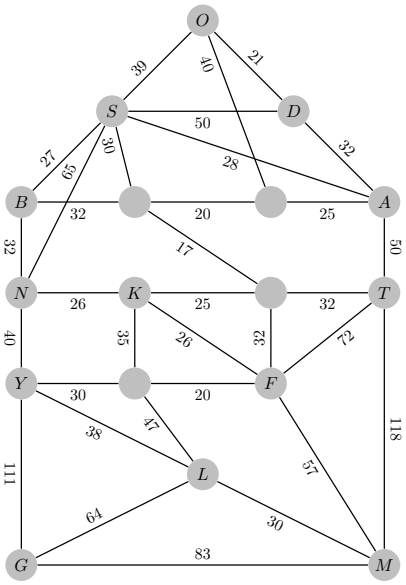

Abbildung 3.: Das zweite Modell der Strassenkarte aus Abbildung 1. Wir haben nun sowohl Städte als auch Strassenkreuzungen zusammengefasst zu Verkehrsverknüpfungspunkten.

▼ Aufgabe 1.3.

Während wir modellieren und abstrahieren, müssen wir stets darauf achten, dass wir innerhalb unseres Modells das gestellte Problem noch lösen können. Für welche der folgenden Bezeichnungen für die noch nicht bezeichneten Verkehrsverknüpfungspunkte aus Abbildung 3 würden Sie sich entscheiden? Begründen Sie Ihre Wahl.

- *Wir wissen, dass alle noch nicht bezeichneten Punkte Strassenkreuzungen sind. Wir bezeichnen alle mit X und merken uns dies.*
- *Jeder noch nicht bezeichnete Verkehrsverknüpfungspunkt war einmal eine Strassenkreuzung. Wir geben jedem dieser Punkte die Abkürzung S für Strassenkreuzung. Dann vergessen wir dies auch nicht in der späteren Lösung, dass es sich um eine Strassenkreuzung handelt.*
- *Wir wählen die Bezeichnungen X_1, X_2, X_3 und X_4. Anschliessend weisen wir jedem noch nicht bezeichneten Verkehrsverknüpfungspunkt eine dieser Bezeichnungen zu.*

1.2. Bewertete Graphen

Wir sind nun bereit, aus unserem (vereinfachten) Modell der Strassenkarte das mathematische Modell abzuleiten. Betrachten Sie Abbildung 4. Wir haben für die restlichen Verkehrsverknüpfungspunkte die Bezeichnungen C, E, P und X gewählt (überzeugen Sie sich davon, dass dies eine gültige Bezeichnung ist).

Vielleicht ist Ihnen während der Bearbeitung dieses Kapitels schon der Begriff des ***ungerichteten Graphen*** in den Sinn gekommen. Auf diesem wollen wir nun aufbauen. Auch in unserem Beispiel gibt es ***Knoten*** (Verkehrsverknüpfungspunkte) die durch ***Kanten*** (Strassen) verbunden sind. Jedoch haben wir noch die Länge der Strassen eingezeichnet. Wir können auch hier auf die Graphentheorie bauen und das Konzept eines Graphen erweitern.

Definition 1 (Bewerteter Graph).

Ein ungerichteter Graph $G = (V, E)$ mit einer reellwertigen Bewertungsfunktion $c\colon E \to \mathbb{R}$ (englisch: cost) heisst ***bewerteter Graph***. Für eine Kante $e \in E$ bezeichnet $c(e)$ die ***Kosten*** der Kante e.

Wir haben also aus der konkreten Problemstellung aus Abbildung 1 durch Modellierung und Abstraktion einen bewerteten Graphen erzeugt (siehe Abbildung 4). Die Bewertungsfunktion weist dabei für unser konkretes Problem jeder Kante die Länge der Strasse zu. Allgemein weist sie jedoch jeder Kante Kosten zu. Diese können für eine andere Problemstellung eine andere Beudeutung haben (zum Beispiel Zeit oder Geld). In einem bewerteten Graphen haben wir jedoch davon abstrahiert und die Bewertungsfunktion besitzt als ***Definitionsmenge*** die Menge aller Kanten (das heisst E von G) und als ***Wertemenge*** die Menge $\mathbb{R}$.
Wir können die graphische Notation aus Abbildung 4 mittels der Definition 1 wie folgt formalisieren:

- Knotenmenge $V = \{O, S, D, B, A, N, K, T, Y, F, L, G, M, T, C, E, P, X\}$
- Kantenmenge

$$E = \left\{ \begin{array}{l} \{S,O\}, \{S,D\}, \{S,A\}, \{S,B\}, \{S,C\}, \{S,N\}, \{O,D\}, \{O,E\}, \{D,A\}, \\ \{E,A\}, \{C,E\}, \{C,B\}, \{B,N\}, \{P,C\}, \{P,T\}, \{P,K\}, \{P,F\}, \{P,K\}, \\ \{A,T\}, \{F,T\}, \{K,F\}, \{N,K\}, \{N,Y\}, \{Y,X\}, \{X,F\}, \{F,M\}, \{T,M\}, \\ \{Y,L\}, \{Y,G\}, \{X,L\}, \{L,G\}, \{G,M\}, \{L,M\} \end{array} \right\}$$

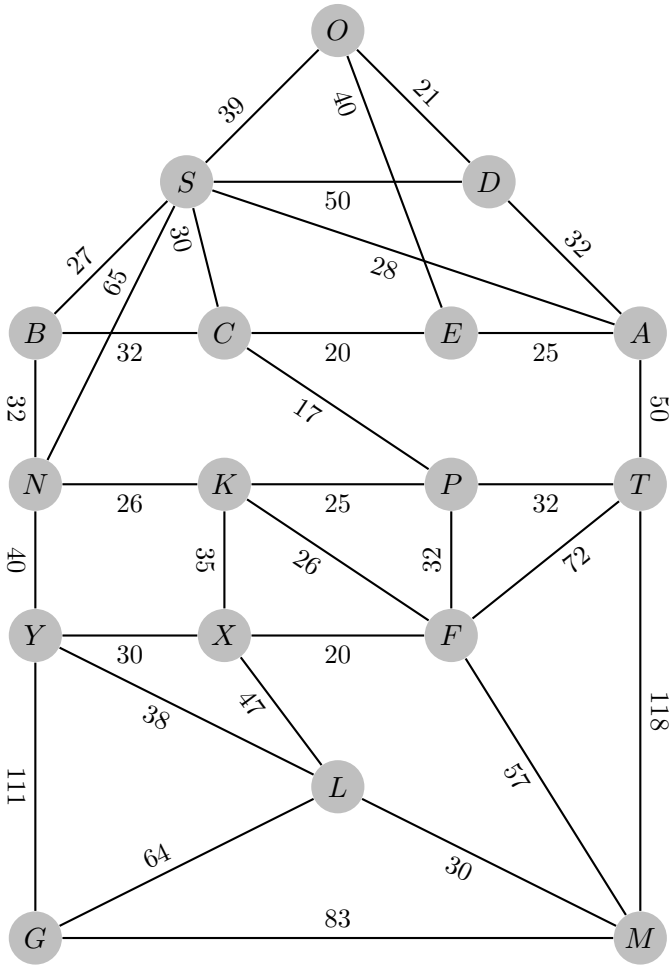

Abbildung 4.: Das dritte Modell für unsere Strassenkarte, welches zugleich das finale Modell darstellt. Dies können wir leicht in ein mathematisches Modell übertragen.

- Kosten $c\colon E \to \mathbb{R}$ mit

$$c(\{S,O\}) = 39, c(\{S,D\}) = 50, c(\{S,A\}) = 28, c(\{S,B\}) = 27, c(\{S,C\}) = 30,$$
$$c(\{S,N\}) = 65, c(\{O,D\}) = 21, c(\{O,E\}) = 40, c(\{D,A\}) = 32, c(\{E,A\}) = 25,$$
$$c(\{C,E\}) = 20, c(\{C,B\}) = 32, c(\{B,N\}) = 32, c(\{P,C\}) = 32, c(\{P,T\}) = 32,$$
$$c(\{P,K\}) = 25, c(\{P,F\}) = 35, c(\{A,T\}) = 50, c(\{F,T\}) = 72, c(\{K,F\}) = 26,$$
$$c(\{N,K\}) = 26, c(\{N,Y\}) = 40, c(\{Y,X\}) = 30, c(\{X,F\}) = 20, c(\{F,M\}) = 57,$$
$$c(\{T,M\}) = 118, c(\{Y,L\}) = 38, c(\{Y,G\}) = 111, c(\{X,L\}) = 47, c(\{L,G\}) = 64,$$
$$c(\{G,M\}) = 83, c(\{L,M\}) = 30$$

Falls ein bewerteter Graph G nur nicht negative Kosten besitzt, das heisst die Bewertungsfunktion ist definiert als $c : E \to \mathbb{R}_{\geq 0}$ mit $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} : x \geq 0\}$, sagen wir G ist ein **Distanzgraph**. Ein Distanzgraph ist somit ein spezieller bewerteter Graph.

▼ Aufgabe 1.4.

Notieren Sie für den Graphen $G_1 = (V_1, E_1)$ (siehe Abbildung 5) formal die Knotenmenge V_1, die Kantenmenge E_1 sowie die Kosten für jede Kante $u \in E$.

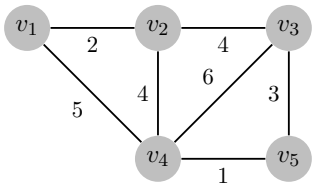

Abbildung 5.: Der bewertete Graph G_1.

▼ Aufgabe 1.5.

Betrachten Sie die Karte in Abbildung 6. Sie zeigt einen Aussschnitt aus dem Veloroutennetz der Stadt Zürich. Die grünen Routen zeigen die empfohlenen Routen, die grün-gestrichelten Routen zeigen die empfohlenen Routen auf Naturbelag und die orangen Routen zeigen die schnellen, ergänzenden Verbindungen für Routinierte gemäss der Stadt Zürich. Das Gemeinschaftszentrum Schwamendingen möchte gerne die kürzesten Velorouten zwischen zwei Orten in Schwamendingen ermitteln können. Um sich einen Überblick zu verschaffen versuchen Sie die Velokarte als bewerteten Graphen zu modellieren. Wie sieht der bewertete Graph G_K aus? Beachten Sie die folgenden Punkte bei der Modellierung:

a) Die Velokarte in Abbildung 6 besitzt zahlreiche Informationen. Welche sind von zentraler Bedeutung? Welche sind nicht wichtig?

b) Benutzen Sie die Methode der Abstraktion um das Wesentliche aus der Karte zu ermitteln.

c) In der Karte gibt es unterschiedliche Velorouten (zum Beispiel grün und orange Routen). Können Sie dies Routen verallgemeinern?

d) Können wir die Karte als Distanzgraphen modellieren? Begründen Sie Ihre Antwort.

e) Welche Bedeutung haben die Knoten im bewerteten Graphen? Wie ist ihre Bezeichnung zu wählen?

f) Welche Bedeutung haben die Kanten im bewerteten Graphen?

g) Wie sieht die Bewertungsfunktion aus? Was ist die Definitionsmenge? Was ist die Wertemenge?

Recherchieren Sie fehlende Informationen mit Hilfe des Internets.

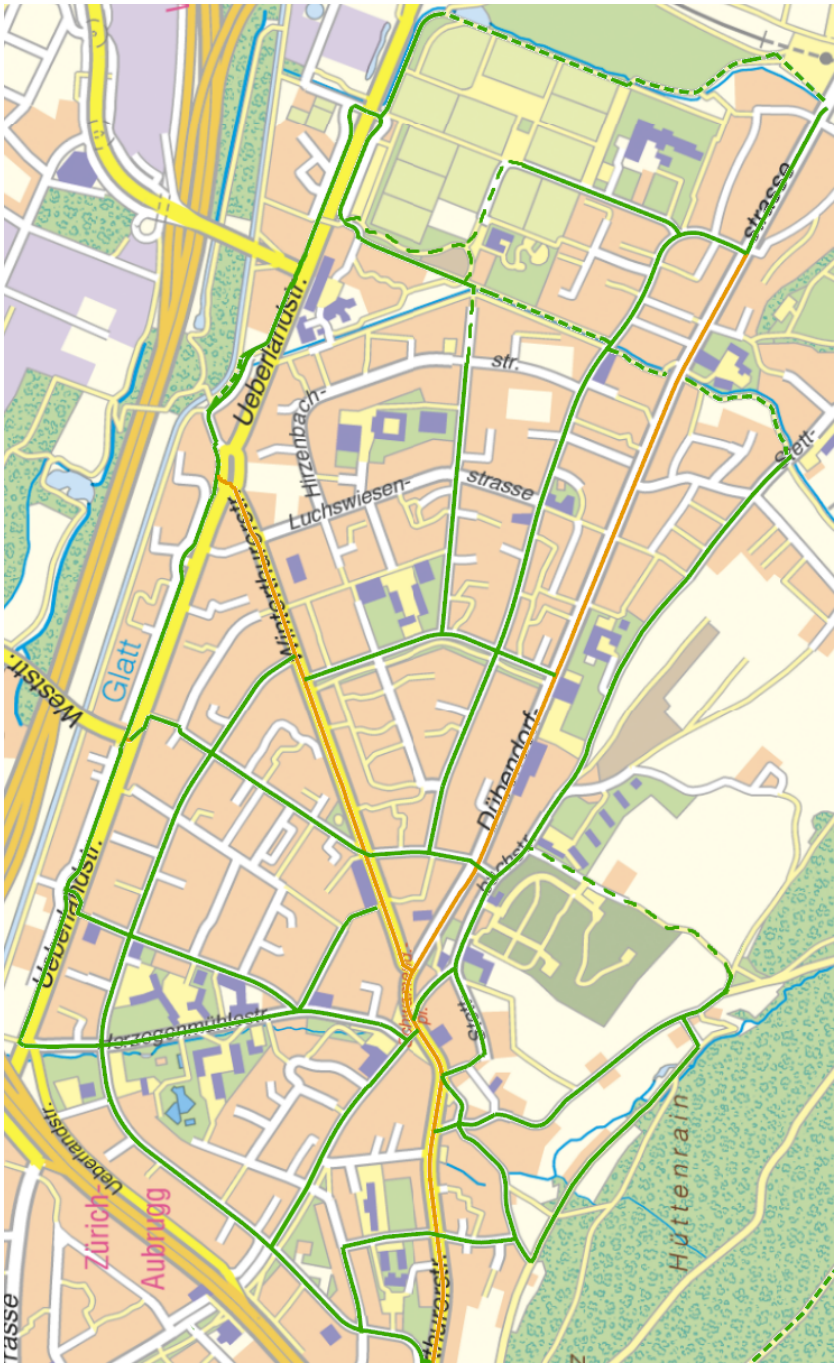

Abbildung 6.: Velorouten in Zürich-Schwamendingen.[1]

Welche Vorteile haben wir durch dieses mathematische Modell zur Problemlösung erhalten? Vergleichen Sie das Resultat der Modellierung der Strassenkarte (Abbildung 4 und der Velokarte aus Aufgabe 1.5. Wir haben es geschafft, unterschiedliche Karten mit unterschiedlicher Darstellung in eine konsistente Darstellung zu bringen. Die Gemeinsamkeit der Darstellungen ist dabei der bewertete Graph. In diesem Fall haben wir es sogar geschafft, einen Distanzgraphen für beide Karten zu modellieren. Dies haben wir durch die Methode der Abstraktion geschafft. Wir können uns nun darauf konzentrieren, einen Algorithmus zu entwickeln, der unsere Fragestellungen allein durch die Eingabe eines bewerteten Graphen löst. Dies bedeutet, wir entwickeln nicht einen Algorithmus der die kürzeste Route für Strassenkarten findet und einen anderen der dies für Velokarten tut, sondern nur einen einzigen, der dies für bewertete Graphen tut. Wir entwickeln also einen Algorithmus, der unser Problem unter der Eingabe eines bewerteten Graphen löst. Darum kümmern wir uns im nächsten Kapitel.

1.3. Zusammenfassung

Wir modellieren eine konkrete Probleminstanz (wie zum Beispiel die Strassenkarte aus Abbildung 1) für ein Problem, um die Realität vereinfacht darstellen zu können. Dabei verwenden wir die Methode der Abstraktion, um die wesentlichen Informationen aus der Probleminstanz darzustellen. Dabei sollte man darauf achten, ob oberflächlich unterschiedliche Konzepte (wie zum Beispiel Strassenkreuzungen und Städte) wirklich unterschiedlich modelliert werden müssen, oder ob diese zu einem gemeinsamen Konzept verallgemeinert werden können (Generalisierung). Eine Strassenkarte als eine Probleminstanz für das Problem zur Auffindung einer kürzesten Route zwischen zwei Städten kann somit durch einen bewerteten Graphen modelliert werden. Ein bewerteter Graph ist ein (ungerichteter) Graph mit einer Bewertungsfunktion. Die Bewertungsfunktion weist jeder Kante im Graphen eine reelle Zahl zu, die wir Kosten nennen. Sind die Kosten stets nicht negativ, so sprechen wir von einem Distanzgraphen.

[1] Angepasster Ausschnitt aus dem Internetstadtplan der Stadt Zürich (`https://www.stadt-zuerich.ch/ted/de/index/taz/mobilitaet/fuss-_und_veloverkehr/veloverkehr/velorouten.html`). Das Copyright liegt bei Orell Füssli.

1.4. Lernkontrolle

1. Was versteht man unter dem Begriff der Modellierung? Gehen Sie dabei auf die Methode der Abstraktion ein.

2. Erläutern Sie das Konzept eines bewerteten Graphen.

3. Stellen Sie den bewerteten Graphen $G_2 = (V_2, E_2)$ mit der Knotenmenge $V_2 = \{v_1, v_2, v_3, v_4, v_5\}$, der Kantenmenge $E_2 = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}\}$ und den Kosten $c(\{v_1, v_2\}) = c(\{v_1, v_4\}) = -1, c(\{v_1, v_3\}) = c(\{v_1, v_5\}) = 1$ graphisch dar. Handelt es sich um einen Distanzgraphen? Begründen Sie Ihre Antwort.

1.5. Musterlösungen

Lösung zur Aufgabe 1.1

1. Diese Information ist wichtig. Ohne die Städtenamen oder wenigstens einer Abkürzung davon können wir nicht die kürzeste Route zwischen zwei Städten bestimmen. Wenn wir die Städtenamen weglassen, könnte eine andere Person (oder wir selbst nach einiger Zeit) nicht mehr herausfinden, wo zum Beispiel Olten in unserem Modell zu finden ist. Wenn wir dies nicht bestimmen können, dann können wir auch nicht die kürzeste Route in unserem Modell bestimmen, und somit ist die Information wichtig.

2. Die Position der Städte in der Strassenkarte ist für uns nicht wichtig. Wir interessieren uns nur für die Verbindungen *zwischen* den Städten und die dafür benötigte Länge. Die kürzeste Route zwischen Olten und Genf ändert sich nicht, wenn wir die Strassenkarte zum Beispiel um 90 Grad drehen.

3. Grössere Städte sind in der Strassenkarte mit einem grösseren Kreis eingezeichnet. Dies hat jedoch keinen Einfluss auf die Berechnung der kürzesten Route zwischen zwei Städten. Wir können diese Information somit als unwichtig einstufen.

4. Wenn wir einmal eine Strasse gewählt haben, müssen wir dieser gemäss ihrem Verlauf auch folgen. Wir können den Strassenverlauf nicht anpassen. Der Verlauf hat jedoch keinen Einfluss auf unsere Berechnungen bezüglich der kürzesten Route. Wir interessieren uns für die Länge der Strasse gemessen in Kilometern. Ob die Strasse nun durch Kurven 30 km lang ist oder eine gerade Strecke bildet, ist uns hier egal. Der Strassenverlauf ist somit nicht wichtig.

5. Diese Information ist sehr wichtig. Wir sind daran interessiert die kürzeste Route zwischen Olten und Genf zu ermitteln und dabei die Anzahl der zu absolvierenden Kilometer möglichst klein zu halten. Deshalb müssen wir wissen, wie lang eine Strecke zwischen zwei Städten ist. Ausserdem müssen die Streckenlängen vergleichbar sein. Darum sollten wir alle Längen in die gleiche Einheit (zum Beispiel Kilometer) transformieren. Dann können wir die Einheiten in unserem Modell weglassen, da diese einheitlich sind. Würden wir die Längen nicht in dieselbe Einheit umwandeln und in unserem Modell die Einheiten auch nicht darstellen, dann würden wir bei der Ermittlung der kürzesten Route eventuell einen Fehler machen (zum Beispiel eine Strecke mit der Länge 100 als länger ansehen als eine Strecke mit der Länge 5, obwohl die erste Länge in Metern angegeben ist und die zweite in Kilometern).

6. Wir sind mit dem Auto unterwegs und es gibt keine Fähren. Die kürzeste Route wird somit nicht durch Seen oder Flüsse beeinflusst. Alle möglichen Transportwege sind durch Strassen eingezeichnet. Die Information ist also unwichtig.

7. Diese Information ist nicht wichtig. Die Begründung ist ähnlich zum Strassenverlauf. Wir interessieren uns nicht für die Fahrtzeit, die für gewöhnlich auf der Autobahn durch die mögliche höhere Geschwindigkeit kürzer ist, sondern für die Streckenlänge in Kilometern. Dabei kommt es nicht auf den Strassentyp an.

8. Wir sind darauf angewiesen, zu wissen welche Verbindungen unter den Städten möglich sind. Wäre dies nicht wichtig, so könnten wir zum Beispiel die Verbindung zwischen Lausanne und Genf, Yverdon-les-Bains und Genf sowie Montreux und Genf in unserem Modell einfach weglassen. Dann könnten wir jedoch nicht mehr die kürzeste Route von Olten nach Genf in unserem Modell ermitteln, da es laut unserem Modell keine Verbindung nach Genf gibt (obwohl diese in der ursprünglichen Strassenkarte vorhanden ist). Wir sehen, die Strassenverbindungen sind wichtig.

9. Strassenkreuzungen ermöglichen dem Autofahrer, von einer Strasse auf eine andere zu wechseln. Bei der Ermittlung der kürzesten Route ist dies wichtig. Würden wir in unserem Modell keine Strassenkreuzungen erlauben, erhalten wir in unserem Modell eventuell eine längere Route als eigentlich nötig, da man in der ursprünglichen Problemstellung Strassenkreuzungen berücksichtigt hat. Somit ist diese Information wichtig.

10. In welchem Land die Strasse verläuft ist für die Berechnung der kürzesten Route nicht wichtig, wenn wir davon ausgehen, dass der Autofahrer sich in jedem Land frei bewegen darf (was wir tun wollen).

Lösung zur Aufgabe 1.2

1. Sowohl in der Stadt als auch bei einer Strassenkreuzung ist es möglich, von einer Strasse auf eine andere zu wechseln. Wenn wir zum Beispiel mit dem Auto von Lausanne nach Kerzers fahren, können wir uns bei der Kreuzung für die Strasse nach Yverdon (Y), Kerzer (K) oder Freiburg (F) entscheiden. Auch in Kerzers (K) können wir die Strasse nach Neuenburg (N) nehmen, die Strasse nach Freiburg (F) oder in Richtung Thun (T) fahren. Dies geht unabhängig davon in welche Richtung wir fahren oder woher wir kommen.

2. Betrachten wir die Möglichkeiten die wir mit einem Auto in einer Stadt haben und bei einer Kreuzung nur unter dem Blickwinkel der Routenplanung, so können wir in beiden Fällen die Strasse wechseln (nicht mehr und nicht weniger). Der einzige Unterschied besteht darin, dass in der Strassenkarte die Städte einen Namen besitzen und die Strassenkreuzungen nicht.

3. Wir geben den Strassenkreuzungen ebenfalls einen Namen und führen die gleiche Darstellung wie für Städte in unserem Modell ein. Wenn gewünscht können wir uns separat noch notieren, welche Namen für Strassenkreuzungen stehen.

Lösung zur Aufgabe 1.3

Wir können weder alle Strassenkreuzungen mit X bezeichnen noch können wir allen Strassenkreuzungen den Namen S geben. Haben wir dies einmal getan, können wir diese nicht mehr unterscheiden. Wenn wir unser Modell einer weiteren Person zeigen würden und dieser den Auftrag geben würden, die kürzeste Route von der Strassenkreuzung X oder S nach Genf zu ermitteln, wüsste die Person nicht welche Strassenkreuzung gemeint ist. Zudem wäre die Wahl des Buchstaben S noch zu verwechseln mit der Bezeichnung für die Stadt Solothurn. Wir müssen somit jeder Strassenkreuuzung einen eindeutigen Namen geben. Dies ist durch die Bezeichnungen X_1, X_2, X_3 und X_4 gegeben. Keine der Bezeichnung wird bereits verwendet und jede Strassenkreuzung bekommt ihre eindeutige Bezeichnung.

Lösung zur Aufgabe 1.4

Die Knotenmenge besteht aus den grauen Kreisen. Wir übernehmen die Bezeichnungen aus der graphischen Darstellung und erhalten somit:

$$V_1 = \{v_1, v_2, v_3, v_4, v_5\}$$

Die Kantenmenge besteht aus den Verbindungen zwischen den Knoten, das heisst aus allen schwarzen Linien in der graphischen Darstellung. Dabei übertragen wir die graphische Darstellung einer Kante als eine Menge von zwei Knoten. Zum Beispiel erhalten wir für die Kante zwischen v_1 und v_2 die Menge $\{v_1, v_2\}$. Wir erhalten die Kantenmenge:

$$E_1 = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_2, v_4\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}\}$$

Die Kosten sind an den Kanten notiert und können durch die Bewertungsfunktion wie folgt beschrieben werden:

$$\begin{aligned} c(\{v_1, v_2\}) &= 2 \\ c(\{v_1, v_5\}) &= 5 \\ c(\{v_2, v_4\}) &= 4 \\ c(\{v_2, v_3\}) &= 4 \\ c(\{v_3, v_4\}) &= 6 \\ c(\{v_3, v_5\}) &= 3 \\ c(\{v_4, v_5\}) &= 1 \end{aligned}$$

(Das Funktionsargument ist hier jeweils eine Kante. Die Wertemenge ist $\{2, 5, 4, 6, 3, 1\}$.)

Lösungen zur Lernkontrolle 1.4

1. „Modellierung (lateinisch: modulus = "Gebäude in verkleinertem Maßstab") bezeichnet die vereinfachte Beschreibung eines wirklichen Systems. Modelle werden zur Abbildung

und zum Verständnis der natürlichen Realität verwendet und sind in vielerlei Hinsicht von Nutzen.“[2]
Wir haben in diesem Kapitel die Strassenkarte („das wirkliche System“) vereinfacht dargestellt. Dabei haben wir die Methode der Abstraktion verwendet, das bedeutet wir haben zwischen wichtigen und unwichtigen Informationen unterschieden und nur die wichtigen Informationen in das Modell aufgenommen. Ausserdem haben wir verschiedene Konzepte zusammengefasst mit Hilfe der Generalisierung (nämlich Strassenkreuzungen und Städte).

2. Ein bewerteter Graph ist eine Erweiterung eines (ungerichteten) Graphen. Ein bewerteter Graph besitzt somit weiterhin Knoten und Kanten. Zusätzlich werden allen Kanten Kosten zugewiesen. Dies geschieht über eine Bewertungsfunktion, die jeder Kante eine reelle Zahl zuweist. Bewertete Graphen können eingesetzt werden, um etwa Strassenverbindungen zwischen Städten und die dazugehörigen Entfernungen zu modellieren.

3. Abbildung 7 zeigt den Graphen G_2.

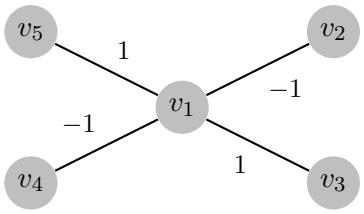

Abbildung 7.: Der bewertete Graph G_2.

Der Graph G_2 ist *kein* Distanzgraph, da es Kanten gibt, welche negative Kosten besitzen.

[2] Quelle: (`http://www.bio5.rwth-aachen.de/index.php?option=com_content&view=article&id=67&Itemid=132&lang=de`)

Kapitel **2**

Der Algorithmus von Dijkstra

Dieses Kapitel zeigt Ihnen ein Verfahren, mit dem Sie die Problemstellung aus Kapitel 1 lösen können. Dazu führen wir weitere graphentheoretische Konzepte ein und machen deutlich, warum wir das Problem durch einen Algorithmus automatisiert lösen sollten. Nachdem Sie dieses Kapitel bearbeitet haben, können Sie das Konzept der kürzesten Route auf bewertete Graphen übertragen und dabei die wichtigsten Definitionen wiedergeben. Zusätzlich werden Sie in der Lage sein zu erläutern, welche Komponenten zu einem Problem gehören bei dem es darum geht unter allen Lösung eine beste auszuwählen. Abschliessend können Sie für eine gegebene Strassenkarte die kürzeste Route mit Hilfe des eingeführten Algorithmus bestimmen. Dabei werden Sie sehen, dass oberflächlich unterschiedliche Probleme durch unsere Modellierung mit dem gleichen Algorithmus gelöst werden können.

2.1. Kürzester Weg

Wir sind auf der Suche nach der kürzesten Route zwischen den Städten Olten und Genf für die Strassenkarte aus Abbildung 1. Diese Aufgabenstellung erfordert es, unter allen möglichen Routen von Olten nach Genf nicht eine beliebige auszuwählen, sondern diejenige, welche die kürzeste Strecke besitzt. Wir möchten also die Entfernung von Olten nach Genf minimieren. Wenn wir unter allen möglichen Lösungen für ein Problem die „beste“ Lösung ermitteln wollen, sprechen wir von einem ***Optimierungsproblem***. Was wir unter einer besten Lösung verstehen, sagt uns die folgende Definition.

Definition 2 (Optimierungsproblem).

Ein ***Optimierungsproblem*** besteht aus einer ***Eingabe***, der ***Menge der zulässigen Lösungen*** für diese Eingabe, einer ***Preisfunktion*** und einem ***Optimierungsziel***. Die Preisfunktion ordnet jeder Lösung eine reelle Zahl zu. Für das Optimierungsziel wird entweder Minimum oder Maximum gewählt.

Ähnlich zur Bewertungsfunktion bei bewerteten Graphen (siehe Definition 1) haben wir hier eine Preisfunktion die jeder zulässigen Lösung Kosten zuweist. Wählen wir für das Op-

timierungsziel „Minimum“ so sprechen wir von einem ***Minimierungsproblem***, wählen wir „Maximum“ nennen wir es ein ***Maximierungsproblem***.
Betrachten wir noch einmal das Problem zu entscheiden ob eine gegebene Zahl eine Primzahl ist oder nicht (Primzahltest). Als Eingabe für dieses Problem bekommen wir eine natürliche Zahl und müssen darauf als Ausgabe (Lösung) entweder „JA“ liefern, falls die gegebene Zahl eine Primzahl ist und ansonsten „NEIN“. Probleme, welche als Lösung nur „JA“ oder „NEIN“ besitzen haben wir als Entscheidungsprobleme bezeichnet. Optimierungsprobleme (wie zum Beispiel die Berechnung der kürzesten Route zwischen zwei Städten) besitzen ebenfalls eine Eingabe (in unserem Beispiel ist es die Strassenkarte, der Startort und der Zielort). Die möglichen Ausgaben (Lösungen) von Optimierungsproblemen können wir nicht auf „JA“ oder „NEIN“ beschränken, da sie durch den Kontext des Problems bestimmt sind. Es wäre nicht sinnvoll bei der Frage nach der kürzesten Route zwischen Olten und Genf mit „JA“ oder „NEIN“ zu antworten. Wir erwarten als Lösung eine konkrete Route, mit der zusätzlichen Bedingung, dass die Entfernung zwischen Olten und Genf minimal ist.
Abbildung 8 zeigt die Beziehungen für unser Beispielproblem zur Definition. Die Ausgangslage für unser Problem umfasst eine Strassenkarte sowie einen Startort und einen Zielort. Dies entspricht der Eingabe. Als Menge der zulässigen Lösungen erlauben wir alle Routen die vom gegebenen Startort zum gegebenen Zielort führen. Die Preisfunktion im unserem Beispiel weist jeder möglichen Route, das heisst jeder zulässigen Lösung, ihre Länge in Kilometern zu. Schlussendlich wollen wir die kürzeste Route finden, dass heisst wir betrachten gemäss Definition 2 ein Minimierungsproblem.

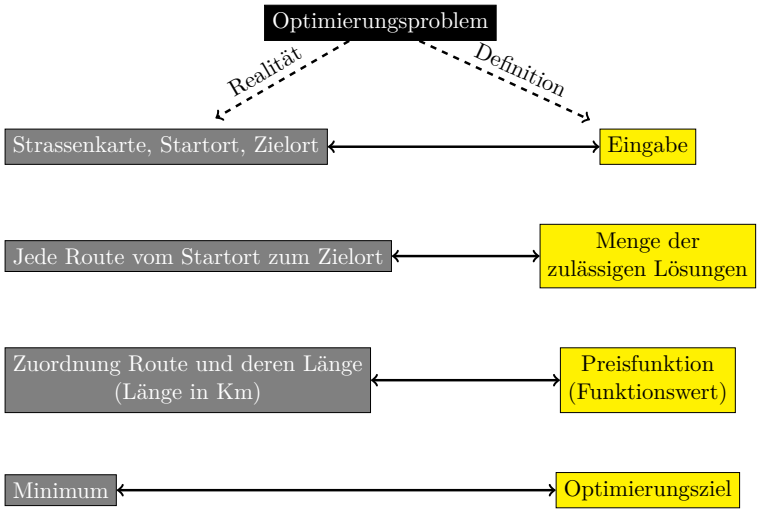

Abbildung 8.: Diese Grafik veranschaulicht die Beziehung zwischen der Definition eines Optimierungsproblems und unserem Problem zur Auffindung der kürzesten Route zwischen zwei Städten.

▼ Aufgabe 2.1.

Lesen Sie die Aufgabenstellung aus Aufgabe 1.5. Handelt es sich dabei um ein Optimierungsproblem? Falls ja, formulieren Sie dies gemäss Definition 2 indem Sie Eingabe, die Menge der zulässigen Lösungen, die Preisfunktion sowie das Optimierungsziel benennen. Falls nicht, geben Sie eine Begründung an warum es sich nicht um ein Optimierungsproblem gemäss Definition 2 handelt.

Wir haben Optimierungsprobleme bisher aus der Sicht von realen Problemstellungen betrachtet. Nun möchten wir unsere Überlegungen in unsere Modellwelt übertragen. Dies bedeutet wir müssen die vier Elemente eines Optimierungsproblems identifizieren. In Kapitel 1 haben wir einen Teil der Eingabe bereits modelliert. Wir haben festgestellt, dass wir eine Strassenkarte mit Hilfe eines bewerteten Graphen modellieren können. Startort und Zielort entsprechen in unserem Modell zwei ausgewählten Knoten. Wir bezeichnen den Knoten der den Startort darstellt als ***Anfangsknoten*** und denjenigen Knoten der dem Zielort entspricht als ***Endknoten***. Abbildung 9 stellt diese Überlegungen grafisch dar und setzt diese in Beziehung zu den bisherigen Überlegungen. Wir wählen in unserem Modell dasselbe Optimierungsziel, da wir die Lösung in unserem Modell später einfach auf die reale Problemstellung übertragen wollen. Damit wir die Menge der zulässigen Lösungen und die Preisfunktion in unserem Modell beschreiben können benötigen wir zunächst einige weitere graphentheoretische Konzepte.

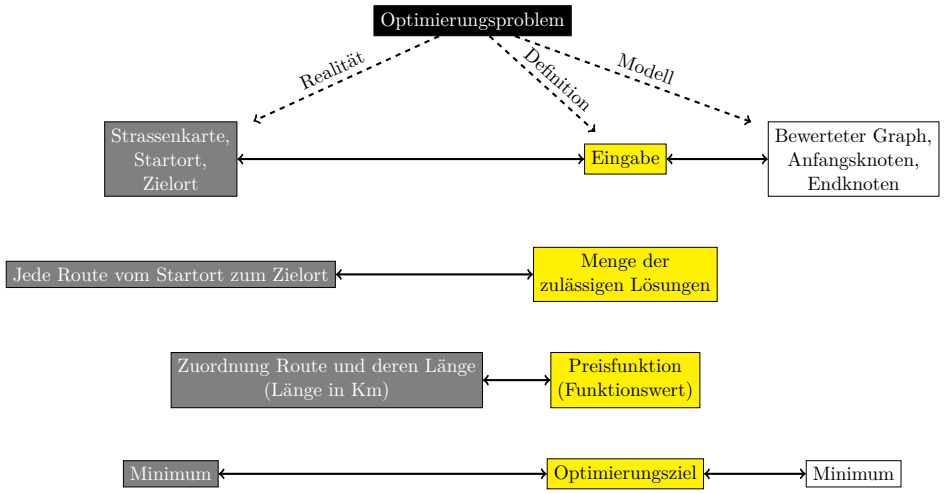

Abbildung 9.: Wir haben die Grafik aus Abbildung 8 erweitert und unsere bisherigen Erkenntnisse über die Beziehung des Modells zur Definition eines Optimierungsproblems hinzugefügt.

Wir können eine Route auf der Strassenkarte aus Abbildung 1 dadurch beschreiben, in dem wir die Städte aufzählen die wir nacheinander anfahren. Eine Route von Olten nach Thun kann

man beschreiben durch den Startort Olten, Dagmarsellen, Affoltern und den Zielort Thun. Da eine Strasse stets exakt zwei Orte miteinander verbindet ist diese Angabe der Route eindeutig (überzeugen Sie sich davon mit einem Blick auf die Strassenkarte). Diese Idee können wir auf unser Modell übertragen. Anstatt die Städte aufzuzählen, nennen wir diejenigen Knoten die wir nacheinander besuchen. Wir sprechen dann nicht mehr von einer Route, sondern von einem ***Weg***. Folgende Definition fasst unsere Überlegungen zusammen.

Definition 3 (Weg).

Ein ***Weg*** $p = (v_1, v_2, ..., v_k)$ in einem ungerichteten Graphen $G = (V, E)$ mit $v_i \in V$, $i = 1 \dots k$, ist ein k-Tupel von Knoten, wobei kein Knoten in p mehrfach vorkommt und zwischen allen aufeinander folgenden Knoten v_j und $v_{j+1} j = 1 \dots k-1$ jeweils eine Kante $u = \{v_j, v_{j+1}\} \in E$ existiert. Wir nennen v_1 den ***Anfangsknoten*** und v_k den ***Endknoten***.

Betrachten wir beispielsweise den Graphen G_1 aus Abbildung 5, dann ist das Tupel $p_1 = (v_1, v_2, v_3, v_4)$ ein Weg in G_1. Das Tupel $p_2 = (v_1, v_2, v_4, v_1)$ ist kein Weg, da der Knoten v_1 mehrfach vorkommt. Das Tupel $p_3 = (v_1, v_2, v_5)$ ist auch kein Weg in G_1, da es keine Kante in E_1 gibt zwischen den Knoten v_2 und v_5. Auch für ungerichtete, bewertete Graphen ist die Definition 3 gültig. Man verzichtet jedoch bei der Notation eines Weges auf die Angabe der Kosten.

▼ Aufgabe 2.2.

Bearbeiten Sie die folgenden Teilaufgaben.

a) Heben Sie den Weg $p = (v_2, v_3, v_4, v_5)$ im Graphen G_3 farblich hervor (Abbildung 10).

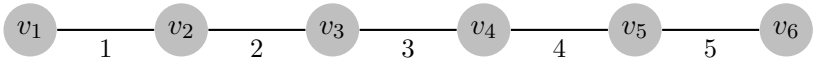

Abbildung 10.: Der bewertete Graph G_3.

b) Ist das Tupel $p' = (v_3, v_4, v_5, v_1, v_2)$ ein Weg in G_3? Begründen Sie Ihre Entscheidung. Ist die Antwort positiv, heben Sie den Weg farblich hervor. Erläutern Sie im anderen Fall, wie man den Graphen G_3 erweitern kann, sodass das Tupel p' ein Weg in G_3 ist (falls möglich).

c) Beschreiben Sie einen Weg im Graphen aus Abbildung 4 mit dem Anfangsknoten O und dem Endknoten G durch ein geeignetes Tupel.

Durch das Konzept des Weges können wir nun in unserem Modell, dem bewerteten Graphen, neu die Menge der zulässigen Lösungen modellieren. Die Menge der zulässigen Lösungen besteht aus allen Wegen zwischen dem Anfangsknoten und dem Endknoten, des gegebenen bewerteten Graphen. Es bleibt die Preisfunktion übrig. In unserem Beispiel können wir jeder Route eine Länge in Kilometern zuweisen. Diese Zuweisung bildet die Preisfunktion, der Funktionswert ist die Länge der Route. Den Funktionswert, das heisst die Länge, berechnen wir in dem wir jeweils diejenigen Länge auf der Route summieren. Wir haben bereits die Route in unser Modell übertragen, nun können wir dies für die Länge ebenfalls tun um damit die Preisfunktion abzubilden.

Definition 4 (Länge eines Weges).

Die *Länge* $l(p)$ eines Weges $p = (v_1, v_2, ..., v_k)$ in einem bewerteten Graphen $G = (V, E)$ mit $v_i \in V$, $i = 0 \dots k$, und Bewertungsfunktion $c : E \to \mathbb{R}$ ist die Summe der Kosten aller Kanten zwischen den Knoten aus p. Kurz:

$$l(p) = \sum_{i=0}^{k-1} c(\{v_i, v_{i+1}\})$$

Somit hat der Weg $p_1 = (v_1, v_2, v_3, v_4)$ aus dem Graphen G_1 in Abbildung 5 die Länge $l(p_1) = \sum_{i=1}^{3} c(\{v_i, v_{i+1}\}) = c(\{v_1, v_2\}) + c(\{v_2, v_3\}) + c(\{v_3, v_4\}) = 2 + 4 + 6 = 12$.

▼ Aufgabe 2.3.

Bestimmen Sie die Länge $l(p)$ für den Weg $p = (v_2, v_3, v_4, v_5)$ im Graphen G_3 aus (Abbildung 10).

Die Beziehung zwischen unserem Beispiel, der Definition und dem Modell ist in Abbildung 11 dargestellt. Wir haben es geschafft, die konkrete Problemstellung aus der Realität vollständig zu modellieren. Wir haben uns dabei vom Kontext der Strassenkarte gelöst und benutzen zur Modellierung ausschliesslich graphentheoretische Konzepte.

Das auf der rechten Seite in Abbildung 11 formulierte Optimierungsproblem ist ein bekanntes Problem in der Informatik und unter dem Namen ***Single-Pair-Shortest-Path-Problem (SPSP-Problem)*** bekannt. Die Bezeichnung *single pair* kommt daher, dass die Eingabe ein Paar von Knoten (dem Anfangs- und Endknoten) enthält. Der Begriff *shortest path* ist im deutschsprachigen Raum als ***kürzester Weg*** bekannt und wie folgt definiert.

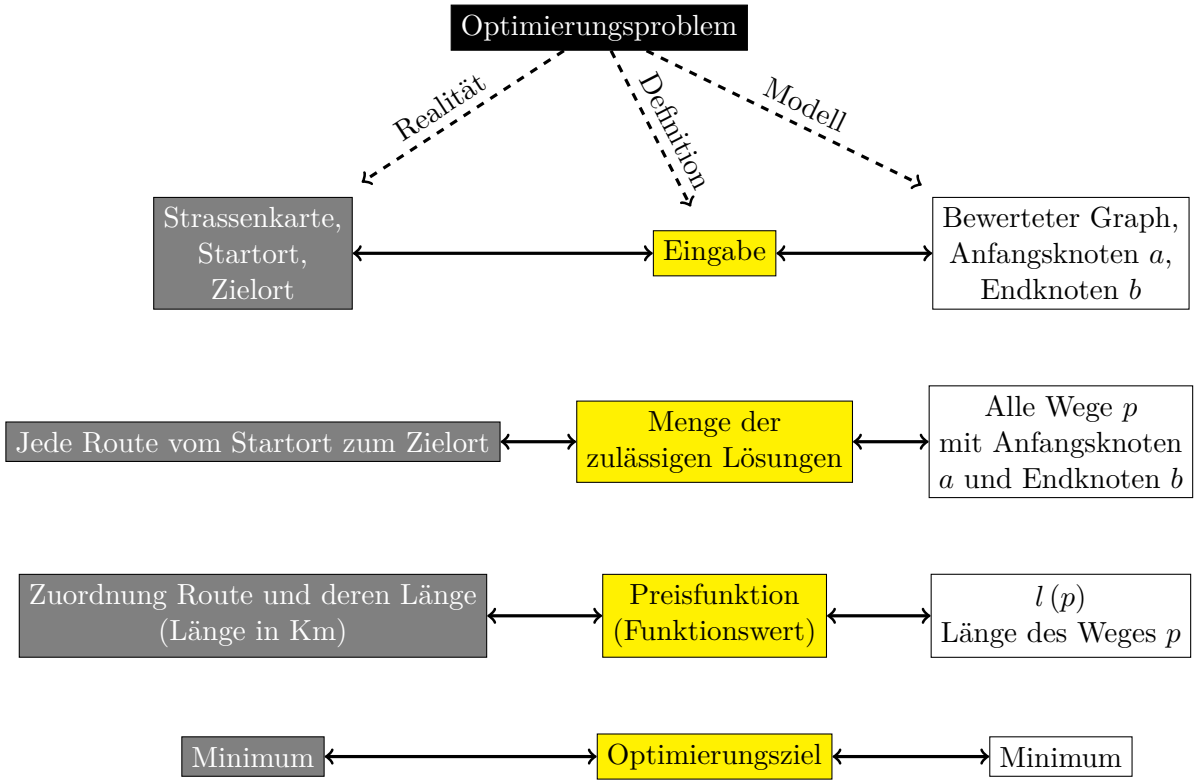

Abbildung 11.: Die Beziehungen zwischen der Problemstellung aus der Realität, der Definition eines Optimierungsproblems und dem Modell für die Problemstellung aus der Realität.

Definition 5 (Kürzester Weg).

Die ***Distanz*** $d(v,u)$ von einem Knoten v zu einem Knoten u ist definiert als das Minimum über die Länge aller Wege von v nach u. Kurz: $d(v,u) = min\{l(p) \mid p \text{ ist ein Weg von } v \text{ nach } u\}$.
Ein Weg p mit Anfangsknoten v und Endknoten u heisst ***kürzester Weg*** zwischen v und u, falls $l(p) = d(v,u)$.

Die Definition des kürzesten Weges zwischen zwei Knoten entspricht somit der kürzesten Route zwischen zwei Orten. Betrachten wir nochmals den Graphen G_3 aus Abbildung 5. Zusammen mit dem Anfangsknoten v_1 und dem Endknoten v_4 bildet er eine Eingabe für das SPSP-Problem. Eine konkrete Eingabe bezeichnen wir sowohl für Entscheidungsprobleme, als

auch für Optimierungsprobleme, als ***Probleminstanz*** oder kurz ***Instanz***. Vergleichen wir dies wieder mit dem Primzahltest. Die Zahl 371 wäre ein Probleminstanz für dieses Entscheidungsproblem. Für das SPSP-Problem wäre eine Instanz gegeben durch einen bewerteten Graphen, einen Startknoten sowie einen Endknoten, das heisst zum Beispiel G_3, v_1 und v_4 bilden eine Instanz für das SPSP-Problem.
Wie finden wir den kürzesten Weg für diese Instanz (G_3, v_1 und v_4)? Eine Möglichkeit besteht darin, alle möglichen Wege zwischen diesen Knoten zu betrachten, jeweils die Länge pro Weg zu berechnen und denjenigen mit der kürzesten Länge auszuwählen. Wir erhalten vier Wege

- $p_1 = (v_1, v_2, v_4)$
- $p_2 = (v_1, v_2, v_3, v_4)$
- $p_3 = (v_1, v_2, v_3, v_5, v_4)$
- $p_4 = (v_1, v_4)$

mit den Längen $l(p_1) = 6, l(p_2) = 12, l(p_3) = 10$ und $l(p_4) = 5$. Somit ergibt sich die Distanz $d(v_1, v_4) = min\{l(p_1), l(p_2), l(p_3), l(p_4)\} = min\{6, 12, 10, 5\} = 5$. Da $l(p_4) = d(v_1, v_4) = 5$ gilt, ist p_4 ein kürzester Weg von v_1 nach v_4.

▼ Aufgabe 2.4.

Finden Sie für den Graphen G_W (aus Abbildung 12) einen kürzesten Weg von v_1 nach v_{16}.

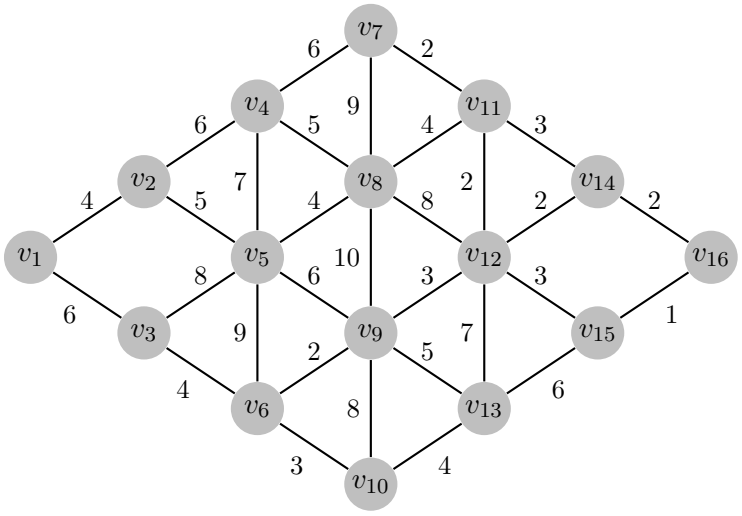

Abbildung 12.: Der bewertete Graph G_W.

Die Lösungsmethode für unsere Problemstellung bei der wir alle möglichen Wege auflisten und anschliessend denjenigen mit der minimalen Länge als kürzesten Weg auswählen, nennen wir die ***Brute-Force-Methode*** oder auch ***erschöpfende Suche*** genannt, da wir alle Möglichkeiten durchprobieren. Dies kann sehr zeitaufwendig sein (wie lange brauchen Sie für die Aufgabe 2.4?). Im nächsten Abschnitt beschäftigen wir uns damit, wie wir es besser machen können.

2.2. Algorithmus von Dijkstra

Wir präsentieren nun den ***Algorithmus von Dijkstra***, welcher uns schrittweise die kürzesten Wege von einem gegebenen Anfangsknoten a zu *allen anderen* Knoten in einem bewerteten Graphen ermittelt. Der Algorithmus gehört zu den Graphenalgorithmen und vermeidet die Brute-Force-Methode, da er nicht alle Kombinationen vom Anfangsknoten a zu allen anderen ausprobiert, sondern zunächst eine Teillösung für die kürzesten Wege berechnet und diese dann schrittweise erweitert. Wir können diese Idee am Graphen aus Abbildung 12 wie folgt illustrieren. Nehmen wir an wir möchten vom Knoten v_1 die kürzesten Wege zu allen anderen Knoten bestimmen, dass heisst auch den kürzesten Weg von v_1 nach v_7. Der Algorithmus von Dijkstra betrachtet bei der Berechnung des kürzesten Weges nun nicht alle möglichen Wege von v_1 zu v_7, sondern berechnet zunächst den kürzesten Weg zum Knoten v_2 und baut weitere Berechnungen darauf auf.
Um die kürzesten Wege Schritt für Schritt berechnen zu können, berechnet der Algorithmus eine Einteilung der Knoten in drei Gruppen. Jeder Knoten gehört entweder zur Gruppe der ***gewählten Knoten***, zur Gruppe der ***Randknoten*** oder zur Gruppe der ***unerreichten Knoten***. Diese Einteilung ist nicht fix, sondern jeder Knoten gehört zunächst zu den unerreichten Knoten, dann zu den Randknoten und zuletzt zu den gewählten Knoten. Ist ein Knoten einmal in diese Gruppe eingeteilt, bleibt er dort bis zur Terminierung des Algorithmus. Abbildung 13 zeigt einen Graphen mit diesen drei Gruppen, wobei v_1 der Anfangsknoten ist.
Die grundlegende Strategie des Algorithmus basiert nun darauf, dass wir in jedem Schritt denjenigen Knoten aus den Randknoten auswählen der die minimale Distanz zum Anfangsknoten besitzt. Dadurch können wir schrittweise unsere kürzesten Wege verlängern und diese bleiben dann auch kürzeste Wege. Die Korrektheit dieses Verfahrens erläutern wir im nächsten Kapitel. Bevor wir den Algorithmus im Detail beschreiben, geben wir einige Hinweise zur Notation und Bedeutung der Gruppen.

2.2.1. Beschreibung der Gruppen

Gewählte Knoten

Für einen gewählten Knoten v steht der kürzeste Weg vom Anfangsknoten a bereits fest und ändert sich nicht mehr. In unserem Beispiel aus Abbildung 13 ist dies für v_1 der Fall. Wir markieren diesen Knoten rot und schreiben die Distanz zum Anfangsknoten in rot neben den

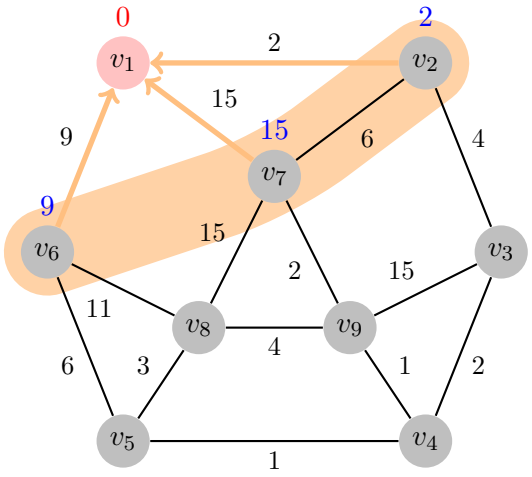

Abbildung 13.: Ein bewerteter Graph, dessen Knoten in drei Gruppen aufgeteilt sind: gewählte Knoten (rot), Randknoten (oranger Hintergrund) und unerreichte Knoten (grau).

Knoten (da v_1 der Anfangsknoten ist, notieren wir die Distanz 0).

Randknoten

Für alle Randknoten ist bereits ein Weg vom Anfangsknoten a bekannt. Ein Randknoten muss nicht direkt eine Kante mit dem Anfangsknoten teilen, jedoch stets mit exakt einem Knoten aus der Gruppe der gewählten Knoten. Über diese Kante gelangen wir zum Anfangsknoten und markieren diese mit orange. Die Randknoten heben wir durch einen orangen Hintergrund hervor. Dies ist für die Knoten v_2, v_6 und v_7 in Abbildung 13 dargestellt. Ausserdem notieren wir die die bisherige Distanz von einem Randknoten zum Anfangsknoten in blauer Farbe neben dem Randknoten. Wir verwenden keine rote Farbe, da sich die Distanz zum Anfangsknoten noch ändern kann und somit auch der kürzeste Weg. Dies ist zum Beispiel für den Knoten v_7 in Abbildung 13 der Fall. In der dargestellten Situation ist v_7 ein Randknoten mit Distanz 15 zum Anfangsknoten v_1. Wir sehen jedoch, dass es einen noch kürzeren Weg (v_1, v_2, v_7) mit Distanz 8 gibt.

Unerreichte Knoten

Für alle unerreichten Knoten kennen wir noch keinen Weg vom Anfangsknoten a. Unerreichte Knoten müssen zuerst zu den Randknoten aufgenommen werden. Diese Knoten sind nicht farblich hervorgehoben. In unserem Beispielgraph aus Abbildung 13 gehören die Knoten v_3, v_4, v_5, v_8 und v_9 zu dieser Gruppe.

2.2.2. Beschreibung der Phasen

Wir erläutern nun die einzelnen Schritte des Algorithmus. Wir gehen davon aus, dass die Eingabe aus einem bewerteten Graphen $G = (V, E)$ mit Bewertungsfunktion $c : E \to \mathbb{R}_{\geq 0}$ und Anfangsknoten a besteht, wobei $a \in V$ gilt. Wir geben jeweils zunächst eine intuitive Erläuterung der Phase, gefolgt von der Beschreibung der Phase mittels Pseudocode. Jede Phase wird mit einem Beispiel abgeschlossen.

Phase 1: Initialisierung

Beschreibung Bevor die eigentliche Ermittlung der kürzesten Wege beginnt, führen wir einige Vorbereitungsschritte durch. Zu Beginn sind alle (auch der Anfangsknoten a) unerreicht und werden deshalb grau markiert. Da wir die Distanzen zum Anfangsknoten noch nicht kennen, setzen wir diese auf den Wert „unendlich“ (blaue Zahlen). Es gibt noch keine Randknoten. Nun fahren wir fort, indem wir den Anfangsknoten zur Gruppe der gewählten Knoten hinzufügen (roter Knoten). Dies können wir tun, da wir den kürzesten Weg vom Knoten a zum Knoten a bereits kennen. Dieser Weg hat die Distanz null (rote Zahl).
Nun fügen wir alle Knoten die eine Kante mit dem Anfangsknoten a teilen zu den Randknoten hinzu (oranger Hintergrund) und aktualisieren die Distanz zum Anfangsknoten. Damit ist die Initialisierung abgeschlossen.

Pseudocode Wir haben eine Prozedur *AktualisiereRandknoten* erstellt, damit wir diese in der zweiten Phase ebenfalls benutzen können.

Phase 1 Initialisierung.

1: Füge alle Knoten $v \in V$ ohne den Anfangsknoten a zur Gruppe der unerreichten Knoten hinzu. Setze für diese Knoten die Distanz auf ∞ (blau). Initialisiere die Menge der Randknoten mit der leeren Menge ($\emptyset$).
2: Füge den Anfangsknoten a zur Gruppe der gewählten Knoten hinzu. Markiere ihn rot und notiere die Distanz $d(a, a) = 0$ (rot) neben dem Knoten.
3: AktualisiereRandknoten(R, a).

Beispiel Wir führen nun die Initialisierung (Phase 1) des Algorithmus von Dijkstra an einem Beispiel vor. Als Eingabe benutzen wir hierzu den Graphen G_4 aus Abbildung 14. Wir wählen v_1 als Anfangsknoten.

- *Zeile 1 und 2:* Wir initialisieren $R = \emptyset$ und notieren neben allen Knoten die Distanz ∞. Es gibt keine Randknoten. Nun markieren wir den Knoten v_1 rot. Die Distanz von v_1 zum Anfangsknoten beträgt 0, da dieser der Anfangsknoten selbst ist. Wir aktualisieren die Distanz neben dem Knoten und färben die Zahl rot. Der Knoten v_1 gehört nun zur Gruppe der gewählten Knoten. Der Fortschritt ist in Abbildung 15 dargestellt.

```
1: procedure AktualisiereRandknoten(Randknoten R, gewählter Knoten v)
2:     for all Kante e = {v, v'} ∈ E do
3:         if v' ist rot markiert then
4:             do nothing
5:         else if d(a, v) + c({v, v'}) < d(a, v') then
6:             Färbe (falls vorhanden) die orange Kante die mit v' verbunden ist schwarz.
7:             Markiere die Kante {v, v'} mit der Richtung (v', v) orange.
8:             d(a, v') ← d(a, v) + c({v, v'}).
9:             if v' ∉ R then Füge v' zu R hinzu. end if
10:        end if
11:    end for
12: end procedure
```

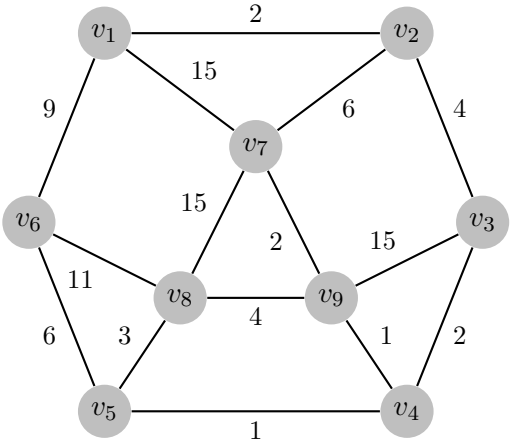

Abbildung 14.: Der Graph G_4.

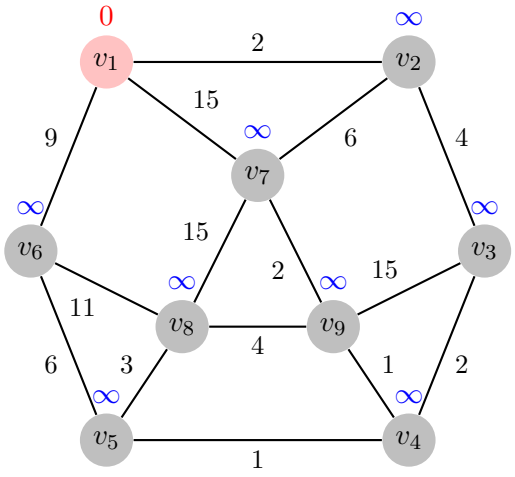

Abbildung 15.:
Nach Zeile 1 und 2 (Initialisierung).

- *Zeile 3:* Wir führen die Prozedur *AktualisiereRandknoten* mit den Parametern v_1 und $R = \emptyset$ durch. Wir müssen die Kanten $\{v_1, v_2\}, \{v_1, v_6\}$ und $\{v_1, v_7\}$ untersuchen:
  - $\{v_1, v_2\}$: v_2 ist nicht gewählt und es gilt
    $d(a, v_1) + c(\{v_1, v_2\}) = 0 + 2 < \infty = d(a, v_2)$. Wir markieren die Kante $\{v_1, v_2\}$ orange und geben dieser die Richtung (v_2, v_1).
    Wir aktualisieren die Distanz: $d(a, v_2) = 0 + 2 = 2$. Wir haben diesen Stand in Abbildung 16 dargestellt. Füge v_2 zu R hinzu.
  - $\{v_1, v_7\}$: v_7 ist nicht gewählt und es gilt
    $d(a, v_1) + c(\{v_1, v_7\}) = 0 + 15 < \infty = d(a, v_7)$. Wir markieren die Kante $\{v_1, v_7\}$

orange und geben dieser die Richtung (v_7, v_1).
Wir aktualisieren die Distanz: $d(a, v_7) = 0 + 15 = 15$. Wir haben diesen Stand in Abbildung 17 dargestellt. Füge v_7 zu R hinzu.

– $\{v_1, v_6\}$: v_6 ist nicht gewählt und es gilt $d(a, v_1) + c(\{v_1, v_6\}) = 0 + 9 < \infty = d(a, v_6)$. Wir markieren die Kante $\{v_1, v_6\}$ orange und geben dieser die Richtung (v_6, v_1). Wir aktualisieren die Distanz: $d(a, v_6) = 0 + 9 = 9$. Wir haben diesen Stand in Abbildung 18 dargestellt. Füge v_6 zu R hinzu.

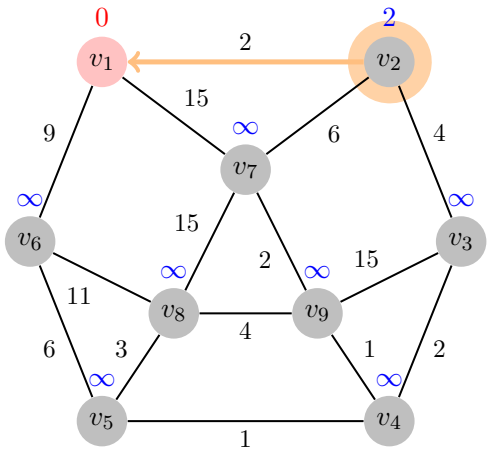

Abbildung 16.:
Nach Zeile 3 (Initialisierung) für $\{v_1, v_2\}$.

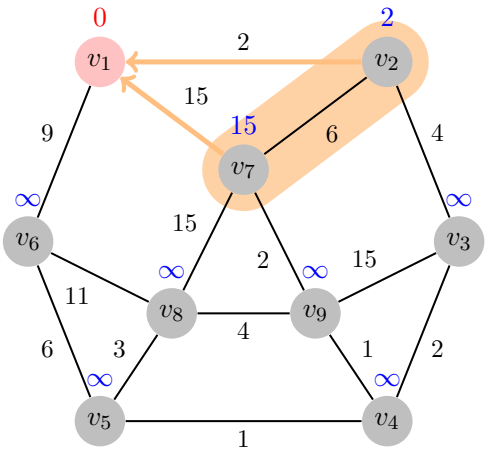

Abbildung 17.:
Nach Zeile 3 (Initialisierung) für $\{v_1, v_7\}$.

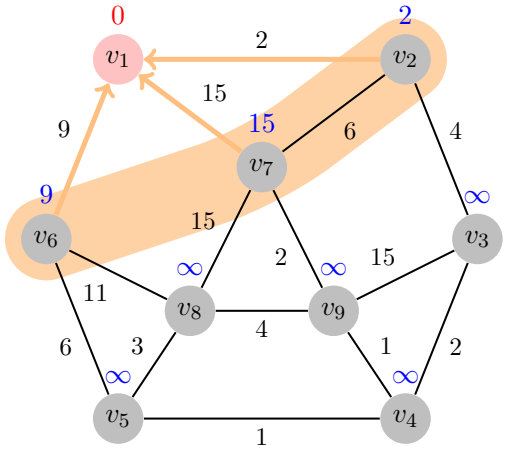

Abbildung 18.:
Nach Zeile 3 (Initialisierung) für $\{v_1, v_6\}$.

Phase 2: Ermittle kürzeste Wege

Beschreibung Nun ermitteln wir alle kürzesten Wege vom Anfangsknoten a. Dazu wählen wir den nächstgelegenen Randknoten aus und fügen diesen zur Menge der gewählten Knoten hinzu. Mit diesem Schritt verlängern wir die bereits kürzesten Wege zu allen bereits gewählten Knoten um einen weiteren Knoten. Dabei wählen wir denjenigen aus, der die minimale Distanz zum Anfangsknoten hat. Der gewählte Knoten besitzt exakt eine orange Kante zu einem bereits gewählten Knoten. Diese färben wir nun rot und behalten dabei die Richtung der Kante bei. Die Richtung hilft uns später bei der Rekonstruktion des kürzesten Weges. Die Distanz zum Anfangsknoten ist nun fix und wird mit rot notiert. Abschliessend müssen wir die Randknoten aktualisieren, da es ein neuer gewählter Knoten gibt. Falls wir nun einen grauen Knoten durch eine Kante mit dem neuen gewählten Knoten erreichen können, fügen wir den grauen Knoten zu den Randknoten hinzu und notieren uns die Distanz zum Anfangsknoten. Bereits vorhandene Randknoten die eine Kante mit dem neuen gewählten Knoten teilen müssen aktualisiert werden. Es muss geprüft werden ob es nun möglich ist über den neuen gewählten Knoten eine kleinere Distanz zum Anfangsknoten zu erreichen.
Wir führen diese Schritte solange durch bis es keine Randknoten mehr gibt.

Pseudocode Wir benutzen für die zweite Phase wieder die Prozedur *AktualisiereRandknoten*.

Phase 2 Ermittle kürzeste Wege ab a.

1: **while** die Menge der Randknoten nicht leer ist, das heisst $R \neq \emptyset$ **do**
2: Wähle einen Knoten $v \in R$ aus, mit minimaler Distanz zum Anfangsknoten a (falls es mehrere gibt, kann einer dieser beliebig gewählt werden).
3: Entferne v aus R und füge v zu der Menge der gewählten Knoten hinzu (rot markieren).
4: Färbe nun die orange Kante zwischen den zwei roten Knoten ebenfalls rot, behalte die Richtung bei (Pfeilspitze).
5: Färbe die Distanz zum Anfangsknoten ebenfalls rot.
6: AktualisiereRandknoten(R, v).
7: **end while**

Beispiel Wir setzen das Beispiel aus Abschnitt 2.2.2 (Phase 1) fort.

- *Schleifendurchlauf 1:* $R = \{v_2, v_6, v_7\}$ ist nicht leer, führe Schleife aus Zeile 2 durch.
  - *Zeile 2:* v_2 hat unter allen Knoten in R die minimale Distanz 2. Wir wählen diesen Knoten aus.

– *Zeile 3:* Wir aktualisieren R in dem wir v_2 entfernen. $R = \{v_6, v_7\}$. Wir markieren v_2 rot und der Knoten gehört nun zur Gruppe der gewählten Knoten. Siehe Abbildung 19.

– *Zeile 4 und 5:* Wir färben die orange Kante $\{v_1, v_2\}$ rot und merken uns die Richtung (v_2, v_1) (Pfeilspitze). Die blaue Distanz am Knoten v_2 färben wir ebenfalls rot. Siehe Abbildung 20.

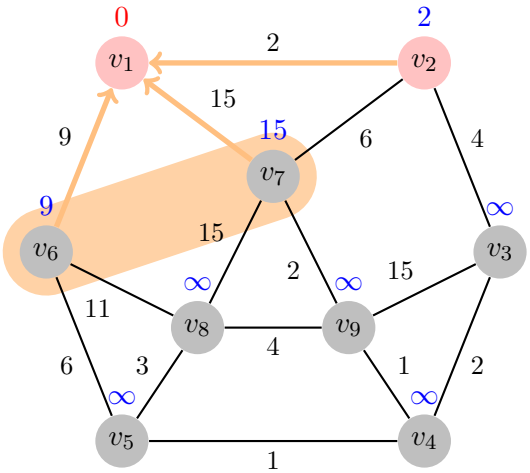

Abbildung 19.:
Nach Zeile 3 (Phase 2) für den Knoten v_2.

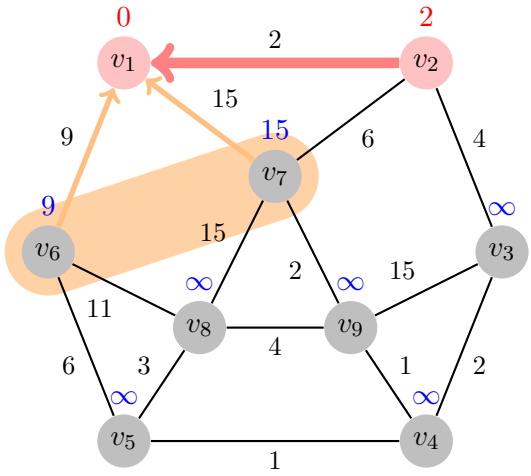

Abbildung 20.:
Nach Zeile 4 und 5 (Phase 2) für v_2.

– *Zeile 6:* Wir führen die Prozedur *AktualisiereRandknoten* mit den Parametern v_2 und $R = \{v_6, v_7\}$ durch. Wir müssen die Kanten $\{v_1, v_2\}, \{v_2, v_7\}$ und $\{v_2, v_3\}$ untersuchen.

  * $\{v_1, v_2\}$: v_1 ist gewählt, wir müssen nichts unternehmen.
  * $\{v_2, v_7\}$: v_7 ist nicht gewählt (nur Randknoten) und es gilt $d(a, v_2) + c(\{v_2, v_7\}) = 2 + 6 < 15 = d(a, v_7)$. Wir färben die Kante $\{v_1, v_7\}$ schwarz und markieren die Kante $\{v_2, v_7\}$ orange und geben dieser die Richtung (v_7, v_2). Wir aktualisieren die Distanz: $d(a, v_7) = 2 + 6 = 8$. Der Knoten ist bereits ein Randknoten. Siehe Abbildung 21.
  * $\{v_2, v_3\}$: v_3 ist nicht gewählt und es gilt $d(a, v_2) + c(\{v_2, v_3\}) = 2 + 4 < \infty = d(a, v_3)$. Wir markieren die Kante $\{v_2, v_3\}$ orange und geben dieser die Richtung (v_3, v_2). Wir aktualisieren die Distanz: $d(a, v_3) = 2 + 4 = 6$. Wir haben diesen Stand in Abbildung 22 dargestellt. Füge v_3 zu R hinzu.

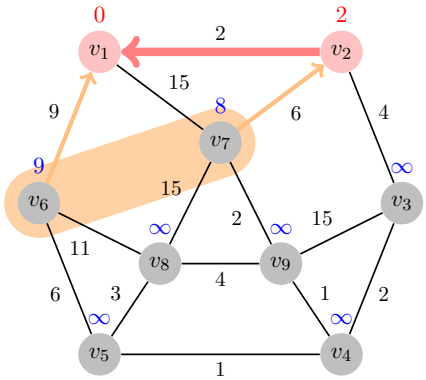

Abbildung 21.:
Nach Zeile 6 (Phase 2) für $\{v_2, v_7\}$.

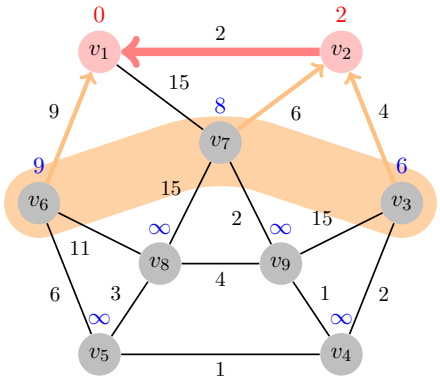

Abbildung 22.:
Nach Zeile 4 und 5 (Phase 2) für v_2.

- *Schleifendurchlauf 2:* $R = \{v_3, v_6, v_7\}$ ist nicht leer, führe Schleife aus Zeile 2 durch.
  - *Zeile 2:* v_3 hat unter allen Knoten in R die minimale Distanz 6. Wir wählen diesen Knoten aus.
  - *Zeile 3:* Wir aktualisieren R indem wir v_3 entfernen. $R = \{v_6, v_7\}$. Wir markieren v_3 rot, und der Knoten gehört nun zur Gruppe der gewählten Knoten. Siehe Abbildung 23.
  - *Zeile 4 und 5:* Wir färben die orange Kante $\{v_2, v_3\}$ rot und merken uns die Richtung (v_3, v_2) (Pfeilspitze). Die blaue Distanz am Knoten v_3 färben wir ebenfalls rot. Siehe Abbildung 24.

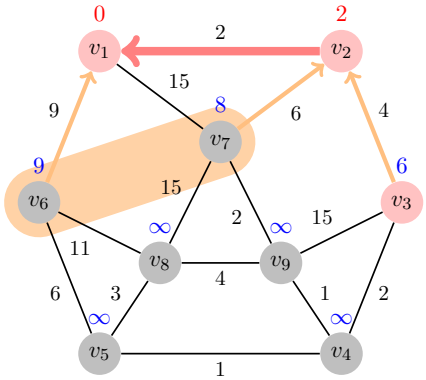

Abbildung 23.:
Nach Zeile 3 (Phase 2) für den Knoten v_3.

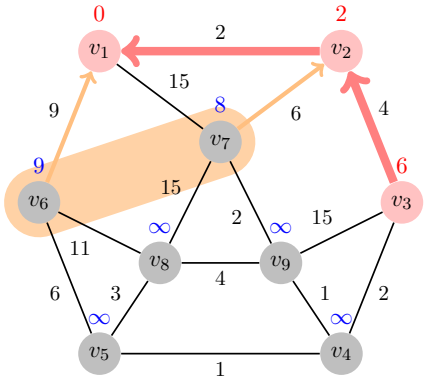

Abbildung 24.:
Nach Zeile 4 und 5 (Phase 2) für v_3.

– *Zeile 6:* Wir führen die Prozedur *AktualisiereRandknoten* mit den Parametern v_3 und $R = \{v_6, v_7\}$ durch. Wir müssen die Kanten $\{v_2, v_3\}, \{v_3, v_4\}$ und $\{v_3, v_9\}$ untersuchen. Wir erhalten den Stand aus Abbildung 25.

Nach weiteren sechs Schleifendurchläufen erhalten wir das Resultat aus Abbildung 26.

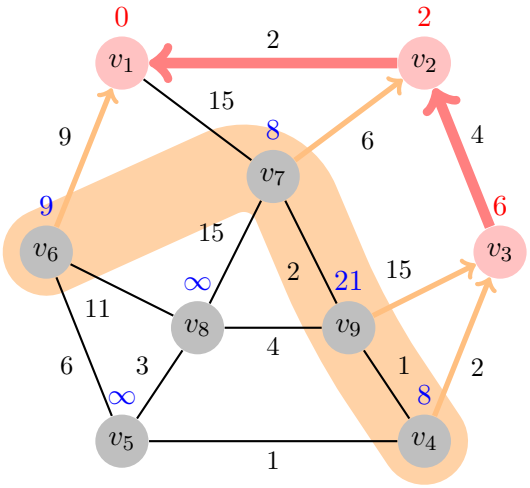

Abbildung 25.:
Nach Zeile 6 (Phase 2) für $\{v_2, v_3\}, \{v_3, v_4\}$ sowie $\{v_3, v_9\}$.

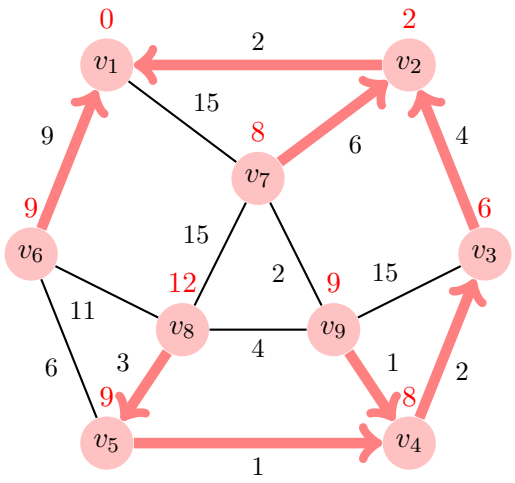

Abbildung 26.:
Die Ausgabe am Ende der Ausführung des Algorithmus.

2.2.3. Lösung ermitteln

Wir haben nun gesehen, wie wir von einem gegeben Anfangsknoten alle kürzesten Wege berechnen können. Wie können wir nun konkret einen kürzesten Weg ermitteln? Betrachten wir dazu nochmals die Lösung aus Abbildung 26. Wir sind nun zum Beispiel am kürzesten Weg von v_1 nach v_9 interessiert. Dazu starten wir beim Knoten v_9 und können dort direkt die Distanz zum Anfangsknoten ablesen (in unserem Beispiel hat der Knoten v_9 die Distanz neun). Anschliessend folgen wir den roten Pfeilen bis zum Anfangsknoten, dabei hilft uns die im Algorithmus vermerkte Richtung der Kanten (Pfeilspitzen). Dieses Verfahren können wir für alle Knoten durchführen.

▼ Aufgabe 2.5.

Finden Sie für den Graphen G_W (aus Abbildung 27) alle kürzesten Weg vom Anfangsknoten v_1. Wenden Sie dazu den Algorithmus von Dijkstra an. Erläutern Sie ebenfalls wie Sie den kürzesten Weg vom Knoten v_1 zum Knoten v_{16} ermitteln.

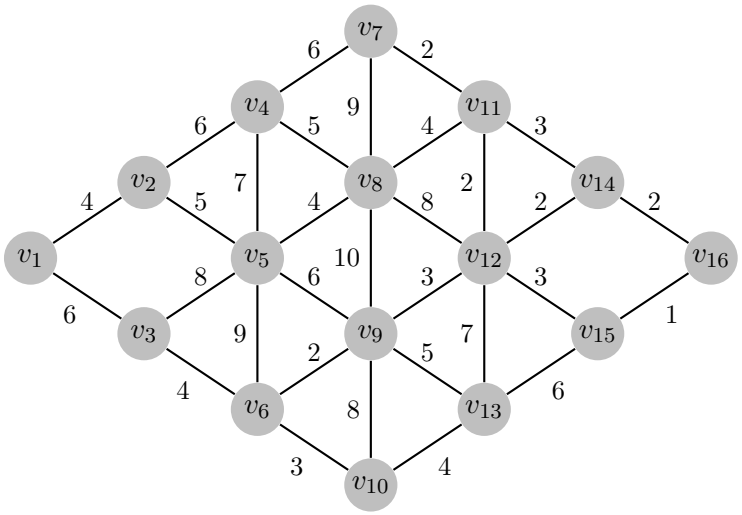

Abbildung 27.: Der bewertete Graph G_W.

2.2.4. SSSP-Problem

Wenn wir die Lösung in Abbildung 26 betrachten, stellen wir fest, dass wir (wie bereits erwähnt) nicht nur den kürzesten Weg zwischen einem Anfangs- und einem Endknoten ermittelt haben, sondern alle kürzesten Wege von einem Anfangsknoten zu allen Endknoten. Können wir mit dieser Lösung auch das SPSP-Problem lösen? Die Antwort ist positiv. Falls wir den Algorithmus von Dijkstra für den Anfangsknoten aus der Eingabe für ein SPSP-Problem wählen, erhalten wir durch die Ausführung des Algorithmus von diesem Anfangsknoten die kürzesten Wege zu allen anderen Knoten. Also auch sicher zum Endknoten der in der Eingabe des SPSP-Problems spezifiziert ist. Dies bedeutet, dass wir für eine Probleminstanz des SPSP-Problem einfach den bewerteten Graphen und den spezifizierten Anfangsknoten dem Algorithmus von Dijkstra übergeben. Dieser berechnet uns dann die kürzesten Wege von diesem Anfangsknoten zu allen anderen Knoten. Anschliessend wählen wir einfach den Weg zwischen Anfangs- und Endknoten aus dieser Lösung aus. Wir nennen das Problem, welches von einem Anfangsknoten die kürzesten Wege zu allen anderen Knoten sucht, das ***Single-Source-Shortest-Path-Problem (SSSP-Problem)***. Der Begriff *source* wird deshalb verwendet, weil im Englischen der Anfangsknoten als *source* (deutsch: Quelle) bezeichnet wird.

▼ Aufgabe 2.6.

Wenden Sie für den bewerteten Graphen aus Abbildung 4 den Algorithmus von Dijkstra an, um vom Anfangsknoten O (Olten) alle kürzesten Wege zu berechnen. Ermitteln Sie daraus den kürzesten Weg von Olten nach Genf. Wie lang ist die Route? Wie muss gefahren werden?

2.3. Zusammenfassung

Der Algorithmus von Dijkstra berechnet uns für einen gegebenen Anfangsknoten die kürzesten Wege zu allen Knoten. Dieses Optimierungsproblem nennen wir das Single-Source-Shortest-Path-Problem (SSSP-Problem). Aus einer Lösung für dieses Problem können wir einfach eine Lösung für das SPSP-Problem ermitteln, so fern die Probleminstanzen den gleichen bewerteten Graphen teilen und denselben Anfangsknoten besitzen. Dann beinhaltet die Lösung zum Single-Source-Shortest-Path-Problem auch den kürzesten Weg, der im SPSP-Problem gesucht wird. Ein kürzester Weg zwischen zwei Knoten v und u ist ein Weg, der die Länge zwischen v und u minimiert. Die Länge wird dabei durch die Summe der Kosten für die Kanten zwischen den Knoten v und u bestimmt.

2.4. Lernkontrolle

Sie wollen von der Haltestelle *Sternen Oerlikon* zur Haltestelle *Bahnhofquai/HB* gelangen (siehe Abbildung 28) und dabei nur Bus oder Tram benutzen. Umsteigen ist erlaubt, so oft Sie wollen. Da Sie Ihren Zug am Hauptbahnhof Zürich nicht verpassen wollen, möchten Sie möglichst schnell zum Ziel gelangen, das bedeutet die Fahrtzeit reduzieren. Wie würden Sie fahren? Versuchen Sie den Ausschnitt des Liniennetzplans als bewerteten Graphen zu modellieren. Recherchieren Sie die Fahrtzeiten gegebenenfalls im Internet. Wenden Sie anschliessend den Algorithmus von Dijkstra an um die Lösung für Ihr Problem zu bestimmen.

[1] Ausschnitt aus dem Liniennetzplan der VBZ Züri-Linie (`http://www.stadt-zuerich.ch/vbz/de/index/fahrplan/liniennetzplaene.html`)

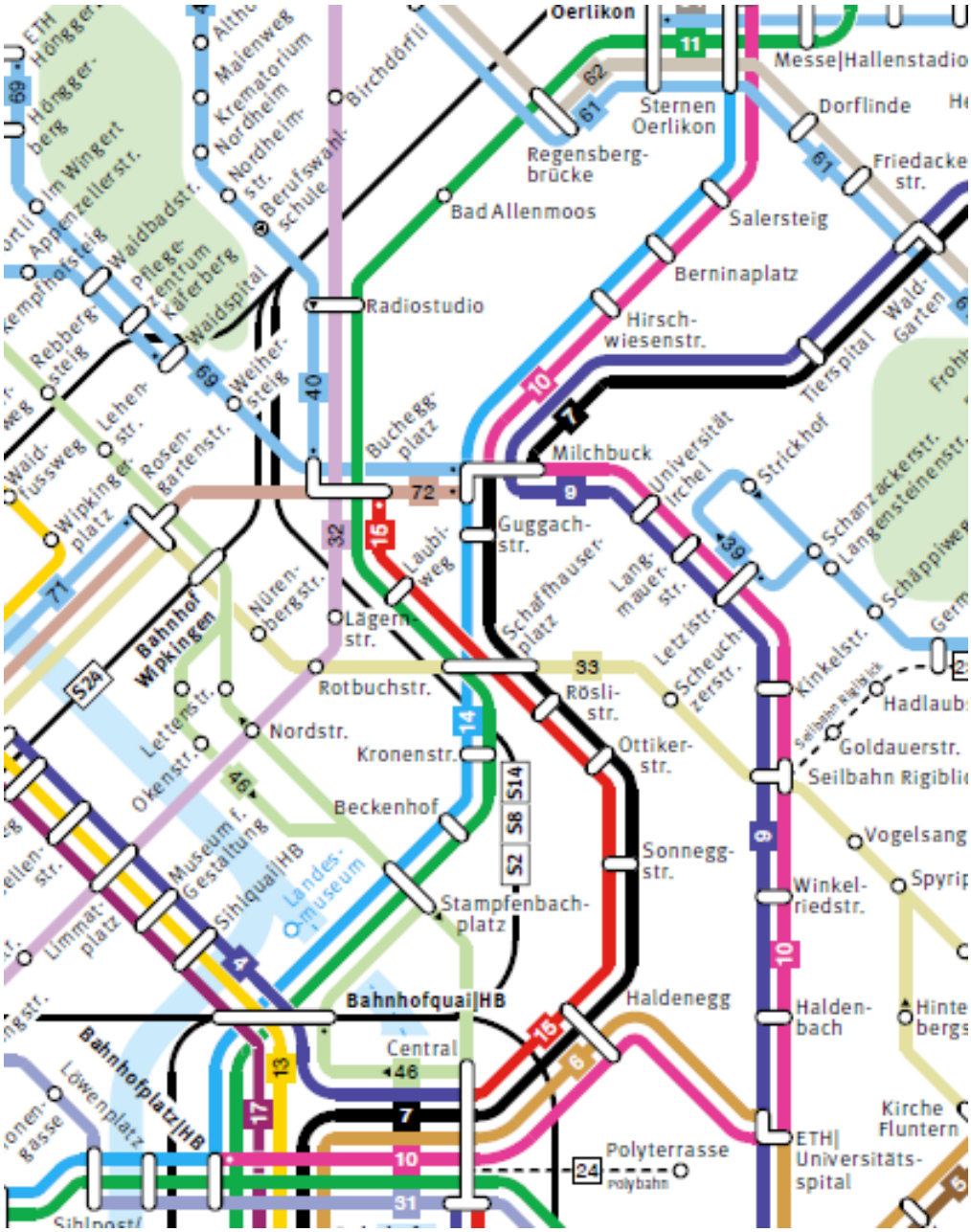

Abbildung 28.: Tramlinien (weisse Zahlen), Buslinien (schwarze Zahlen) und S-Bahnlinien (weisse Rechtecke, schwarze Zahlen) in der Stadt Zürich.[1]

2.5. Musterlösungen

Lösung zur Aufgabe 2.1

Es handelt sich um ein Optimierungsproblem. Die Eingabe besteht aus einer Velokarte mit den eingezeichneten Velorouten, sowie einem Startort und einem Zielort. Die Menge der zulässigen Lösungen besteht aus jeder Veloroute zwischen dem Startort und dem Zielort. Dabei weisen wir jeder Lösung, also jeder Veloroute zwischen dem Start- und Zielort, die Länge der Route zu. Dies bildet die Preisfunktion. Da wir die Strecke zwischen zwei Orten minimieren möchten („möglichst kurz“ laut Aufgabenstellung), handelt sich um ein Minimierungsproblem.

Lösung zur Aufgabe 2.2

a) Der Weg $p = (v_2, v_3, v_4, v_5)$ ist im Graphen G_3 (siehe Abbildung 29) rot markiert.

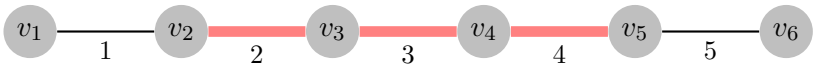

Abbildung 29.: Der bewertete Graph G_3.

b) Das Tupel ist kein Weg, da es im Graphen G_3 keine Kante zwischen v_5 und v_1 gibt. Wir können jedoch den Graphen G_3 erweitern zum Graphen $G_3' = (V, E')$, wobei wir die Kantenmenge E um die Kante $\{v_5, v_1\}$ erweitern, das heisst $E' = E \cup \{v_5, v_1\}$. Ausserdem müssen wir der Kante Kosten zuweisen, dies können wir beliebig tun. Wir haben uns für $c(\{v_1, v_5\}) = 5$ entschieden. Der Graph G_3' ist in Abbildung 30 dargestellt.

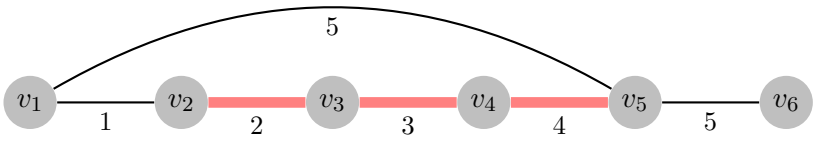

Abbildung 30.: Der bewertete Graph G_3.

c) Wir beschreiben einen Weg von O nach G durch das Tupel $p_c = (O, S, B, N, Y, G)$

Lösung zur Aufgabe 2.3

Der Weg $p = (v_2, v_3, v_4, v_5)$ ist im Graphen G_3 aus Abbildung 29 markiert. Um seine Länge zu bestimmen können wir zunächst die im Weg enthaltenen Kanten bestimmen. Diese sind $\{v_2, v_3\}$, $\{v_3, v_4\}$ und $\{v_4, v_5\}$. Nun bestimmen wir die Kosten dieser Kanten. Wir erhalten $c(\{v_2, v_3\}) = 2$, $c(\{v_3, v_4\}) = 3$ und $c(\{v_4, v_5\}) = 4$. Wir können nun $l(p)$ bestimmen durch einfaches Summieren der Kosten und erhalten $l(p) = 2 + 3 + 4 = 9$.

Lösung zur Aufgabe 2.4

Es gibt mehrere Lösungen. Eine ist in Abbildung 31 dargestellt. Die Distanz vom Knoten v_{16} zum Knoten v_1 beträgt 19.

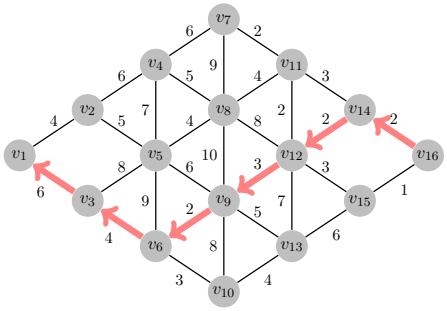

Abbildung 31.: Ein kürzester Weg ist rot markiert.

Lösung zur Aufgabe 2.5

Es gibt mehrere Lösungen, da es Situationen gibt bei denen innerhalb der Gruppe der Randknoten mehrere Knoten diegleiche Distanz aufweisen. Eine mögliche Lösung ist in Abbildung 32 dargestellt.

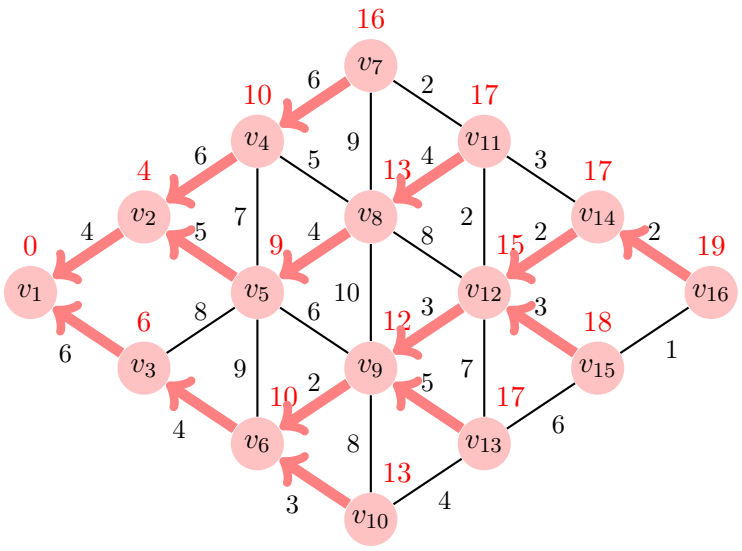

Abbildung 32.: Die kürzesten Wege vom Anfangsknoten v_1 aus.

Lösung zur Aufgabe 2.6

Im Graphen G_{route} aus Abbildung 33 sind die kürzesten Wege und die entsprechenden Distanzen rot eingezeichnet. Wir können nun einfach am Knoten G die Länge der kürzesten Route von O nach G ablesen, dies sind 240 Kilometer. Somit ist die kürzeste Strecke von Olten nach Genf 240 Kilometer lang. Man muss folgende Route wählen: Olten, Solothurn, Biel, Neuenburg, Yverdon-les-Bains, Lausanne und dann Genf. Dies konnten wir ebenfalls im Graphen G_{route} aus Abbildung 33 ablesen, indem wir von Genf (das heisst G) den roten Kanten gefolgt sind.

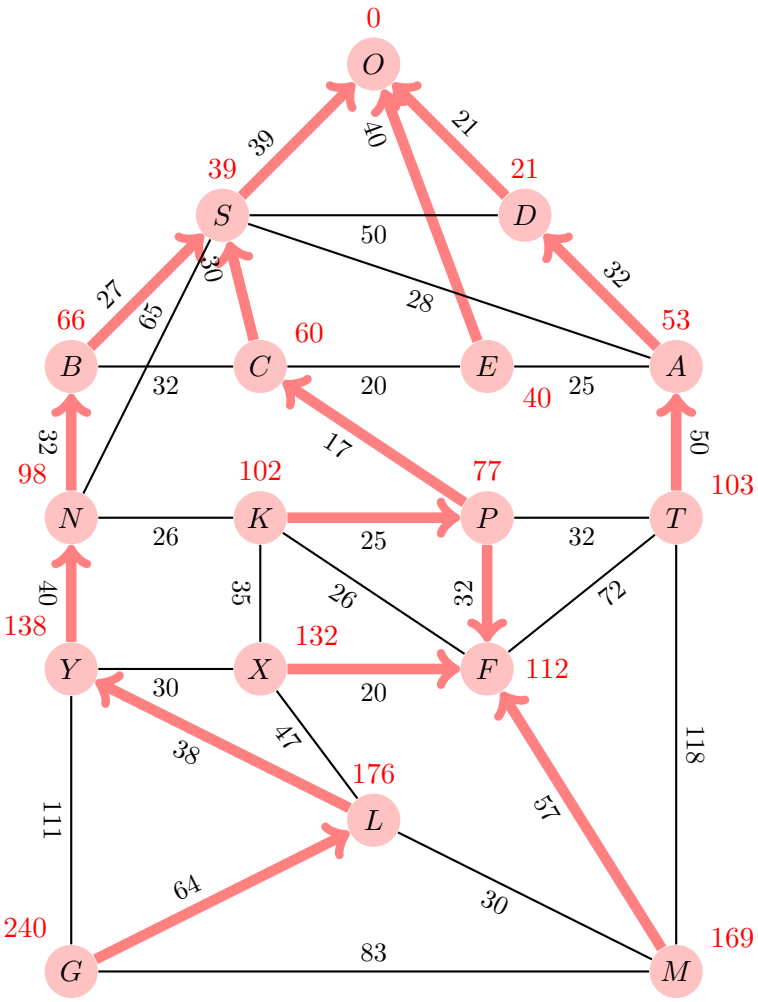

Abbildung 33.: Die kürzesten Wege von Olten zu allen Orten.

Kapitel 3

Analyse des Algorithmus von Dijkstra

In diesem Kapitel beschäftigen wir uns mit dem zu Grunde liegenden Prinzip, der Laufzeit sowie der Korrektheit des Algorithmus von Dijkstra. Sie setzen sich damit auseinander, ob der Algorithmus für jeden bewerteten Graphen das richtige Resultat liefert und stets terminiert. Ausserdem schauen wir uns an, ob der Algorithmus wirklich schneller ist als die Brute-Force-Methode. Nachdem Sie dieses Kapitel bearbeitet haben, können Sie die Korrektheit und Laufzeit des Algorithmus von Dijkstra nachvollziehen. Sie sind sich bewusst, warum der Algorithmus die korrekten kürzesten Wege ermittelt, obwohl nicht alle möglichen Wege betrachtet werden. Sie könne ausserdem das Prinzip des Algorithmus in eigenen Worten erläutern.

3.1. Prinzip des Algorithmus von Dijkstra

Welche grundlegende Idee steckt hinter dem Algorithmus von Dijkstra? Überlegen wir uns noch einmal wie der Algorithmus arbeitet. In jedem Schritt wird aus der Gruppe der Randknoten ein Knoten mit minimaler Distanz ausgewählt. Dieser Knoten wird aus der Gruppe der Randknoten entfernt und zur Gruppe der gewählten Knoten hinzugefügt. Ist es möglich, dass ein Knoten der einmal in der Gruppe der Randknoten war und ausgewählt wurde, wieder zur Gruppe der Randknoten hinzugefügt wird? Die Antwort ist negativ, da in keiner der Phasen (siehe Pseudocode in Abschnitt 2.2.2 und Abschnitt 2.2.2) eine Anweisung definiert ist, die einen Knoten aus der Gruppe der ausgewählten Knoten entfernt. Auch in der Prozedur *AktualisiereRandknoten* verhindern wir explizit die Bearbeitung einer Kante, welche bereits einen gewählten Knoten beinhaltet (siehe Pseudocode der Prozedur in Abschnitt 2.2.2).
Dies bedeutet, falls der Algorithmus von Dijkstra sich einmal entschieden hat, einen Knoten zur Gruppe der gewählten Knoten hinzuzufügen (wir markieren den Knoten und die Kante rot), dann bleibt dieser bis zum Ende in dieser Gruppe. Weiter ist auch die Distanz vom Anfangsknoten zu bereits gewählten Knoten fix und wird nicht mehr angepasst.
Wir können also das grundlegende Prinzip des Algorithmus von Dijkstra wie folgt zusammen fassen:

Die Entscheidung einen Knoten als gewählt zu markieren und dann seine Distanz zum Anfangsknoten festzulegen wird nicht mehr rückgängig gemacht.

Entscheidungen zu Treffen und diese nicht mehr zu revidieren ist beim Entwurf von Algorithmen eine verbreitete Technik. Man sagt der Algorithmus verwendet ein *gieriges* Verfahren beziehungsweise der Algorithmus ist ein ***Greedy Algorithmus*** (englisch: gieriger Algorithmus). Dieses lässt sich wie folgt charakterisieren.

Definition 6 (Greedy Algorithmus).

Ein *Greedy Algorithmus* zeichnet sich dadurch aus, dass er Entscheidungen, die während dem Ablauf des Algorithmus getroffen werden um der gesuchten Lösung näher zu kommen, allein auf Grund der bisher bekannten Informationen getroffen werden. Ausserdem werden die einmal getroffenen Entscheidungen nicht mehr rückgängig gemacht.

Übersetzen wir dies auf die Ausführung des Dijkstra Algorithmus. Die zu treffenden Entscheidungen haben wir bereits angesprochen. Der Algorithmus von Dijkstra ist auf der Suche nach den kürzesten Wegen von einem Anfangsknoten zu allen anderen Knoten. Um der Lösung näher zu kommen muss er schrittweise entscheiden, für welchen Knoten er als nächstes den kürzesten Weg zum Anfangsknoten bestimmt. Dies Entscheidung wird allein durch die bisher bekannten Informationen gefällt. Wenn der Algorithmus die Entscheidung treffen muss, entscheidet er sich für einen Knoten der gerade zur Gruppe der Randknoten gehört. Dabei wird nicht darauf geachtet, ob ein noch nicht erreichbarer Knoten vielleicht die besser Wahl wäre. Die Entscheidung basiert nur auf den bisher berechneten Distanzen. Wurde ein Knoten zur Gruppe der gewählten Knoten hinzugefügt, das heisst es wurde eine Entscheidung getroffen, wird diese nicht mehr rückgängig gemacht. Es wird also kein Knoten aus der Gruppe der gewählten Knoten mehr entfernt.
Kontrastieren wir dieses gierige Verfahren abschliessend noch mit einem nicht gierigen Verfahren. Wir wollen einen Weg durch das Labyrinth aus Abbildung 34 finden.

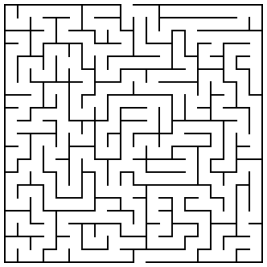

Abbildung 34.: Ein Labyrinth mit einem Eingang am unteren Rand und einem Ausgang am oberen Rand.[1]

[1]Die Bilder wurden mit Hilfe des *Maze Generator* erzeugt und angepasst (`http://www.mazegenerator.net/`).

Ein erster Versuch ist in Abbildung 35 in blauer Farbe eingezeichnet. Wir sehen dieser Versuch bringt uns nicht zum Ausgang. Daher müssen wir wieder zurückgehen und uns für eine andere Möglichkeit entscheiden. Eine weitere Möglichkeit ist in Abbildung 36 dargestellt. Auch hier kommen wir nicht zum Ausgang.

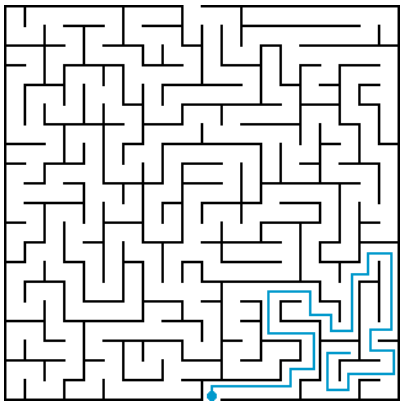

Abbildung 35.:
Ein erster möglicher Versuch durch das Labyrinth zu gehen.

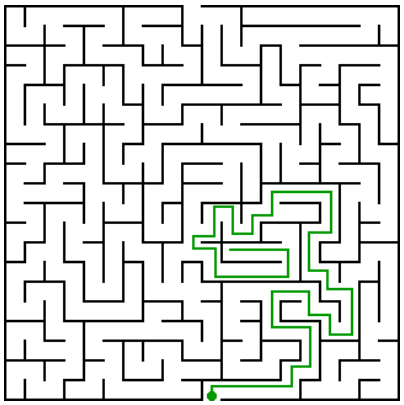

Abbildung 36.:
Ein weiterer Versuch durch das Labyrinth zu gehen.

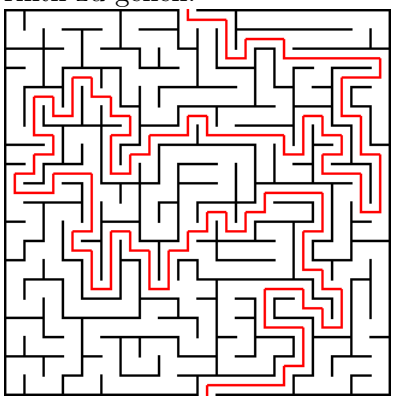

Abbildung 37.: Die Lösung für das Labyrinth.

Wir sehen, dass eine einmal gefällte Entscheidung auf der Suche nach dem Ausgang des Labyrinths unter Umständen wieder rückgängig gemacht werden muss. Wir probieren verschiedene Möglichkeiten aus um näher an die Lösung zu kommen. Jedoch müssen wir manche Möglichkeiten auch wieder verwerfen. Wir können also nicht bereits gefällte Entscheidungen festnageln, wie es ein gieriges Verfahren machen würde, da wir eventuell in einer Sackgasse landen und eine andere Möglichkeit ausprobieren müssen. Die Lösung ist in Abbildung 37 dargestellt. Das hier beschriebene Verfahren ist unter dem Namen ***Versuch-und-Irrtum*** (englisch: trial and

error) bekannt.
Nun könnte man sich fragen: Ist dieses gierige Prinzip im Algorithmus von Dijkstra überhaupt korrekt? Berechnet der Algorithmus immer die korrekte (optimale) Lösung oder übersieht er unter Umständen einen kürzesten Weg? Damit beschäftigen wir uns im nächsten Abschnitt.

3.2. Korrektheit des Algorithmus von Dijkstra

In diesem Abschnitt wollen wir uns davon überzeugen, dass der Algorithmus von Dijkstra korrekt arbeitet. Um die Korrektheit eines Algorithmus zu zeigen, müssen wir zwei Eigenschaften untersuchen:

- Der Algorithmus muss für jede Eingabe terminieren.
- Der Algorithmus muss für eine gültige Eingabe das korrekte Resultat berechnen.

In unserem Fall müssen wir also für den Algorithmus von Dijkstra untersuchen ob dieser für jede Eingabe immer beendet wird (terminiert) und ob er für jede gültige Eingabe, bestehend aus einem bewerteten Graphen und einem Anfangsknoten, die optimale Lösung berechnet wird. Wir untersuchen beide Aspekte separat.

3.2.1. Terminierung

Wenn wir nochmal den Pseudocode in Abschnitt 2.2.2 und Abschnitt 2.2.2 betrachten, sehen wir zwei Schleifen. Diese können unter Umständen dazu führen, dass unser Algorithmus nicht stoppt. Die Schleife in der Prozedur *AktualisiereRandknoten* iteriert über alle Kanten des Graphen aus der Eingabe. Da ein Graph per Definition aus einer endlichen Knotenmenge besteht, gibt es auch nur eine endliche Menge von Kanten. Somit ist eine Endlosschleife in der Prozedur *AktualisiereRandknoten* ausgeschlossen.
Die zweite Schleife iteriert über die Randknoten. Da wir jedoch in jedem Schritt mindestens einen Randknoten aus R entfernen und keinen entfernen Knoten wieder einfügen, wird auch diese Schleife sicherlich stoppen. Da wir keine Rekursion verwenden, haben wir uns davon überzeugt, dass der Algorithmus immer terminieren wird.
Wir müssen auch noch untersuchen ob der Algorithmus für *alle* Eingaben auch wirklich terminiert. Wir haben dies oben nur für alle gültigen Eingaben getan. Wir können eine Eingabe einfach darauf prüfen ob sie gültig ist. Falls nein, führen wir den Algorithmus nicht aus und terminieren somit sicherlich für alle möglichen Eingaben.

3.2.2. Korrektheit

Um uns davon zu überzeugen, dass der Algorithmus stets die optimale Lösung berechnet, müssen wir das Hinzufügen einer Kante zu einem bereits bestehenden kürzesten Weg genauer anschauen. Betrachten wir einen kürzesten Weg $p = (v_1, v_2, \ldots, v_k)$ von v_1 nach v_k, wie zum

Beispiel im Graphen G_5 in Abbildung 38 rot markiert für $k = 5$. Wir bezeichnen jeden Weg $p' = (v_i, \ldots, v_j)$ mit $1 \leq i < j \leq k$ als einen Teilweg von p. In unserem Beispiel ist der Weg $p'_1 = (v_2, v_3, v_4)$ (siehe blaue Umrandung in Abbildung 38) ein Teilweg von $p_1 = (v_1, v_2, v_3, v_4, v_5)$.

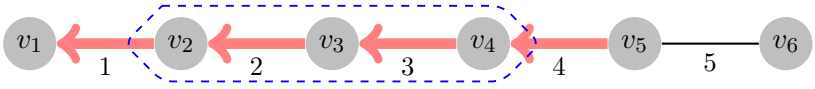

Abbildung 38.: Der bewertete Graph G_6 mit dem kürzesten Weg $p_1 = (v_1, v_2, v_3, v_4, v_5)$

Mit einem Widerspruchsbeweis können wir zeigen, dass jeder Teilweg von v_i nach v_j (dass heist p') ebenfalls ein kürzester Weg von v_i nach v_j ist. Die Ausgangssituation ist in Abbildung 39 dargestellt. Gestrichelte Kanten deuten an, dass weitere Knoten dazwischen liegen können. Die roten Kanten markieren den kürzesten Weg.

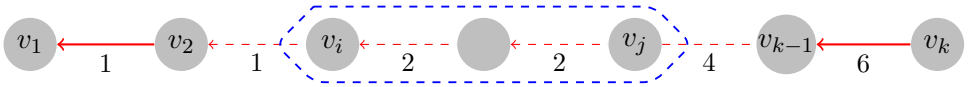

Abbildung 39.: Die Ausgangssituation für unseren Beweis: Ein kürzester Weg $p = (v_1, v_2, \ldots, v_k)$ und ein Teilweg $p' = (v_i, \ldots, v_j)$ (angedeutet durch gestrichelte Linien).

Für den Beweis nehmen wir das Gegenteil an, das bedeutet, der Teilweg p' ist *nicht* der kürzeste Weg von v_i nach v_j. Daher muss es einen anderen Weg geben von v_i nach v_j der kürzer ist. Diesen bezeichnen wir mit p''. Wir haben diese Situation in Abbildung 40 dargestellt. Die grün gepunkteten Kanten deuten den Weg p'' an. Dies kann auch weitere Knoten enthalten, muss aber nicht. Dies spielt für die Argumentation keine Rolle.

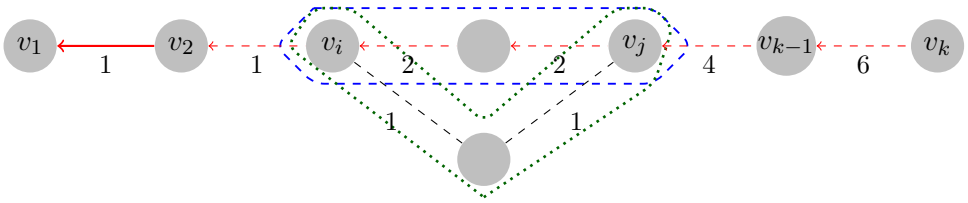

Abbildung 40.: Der kürzeste Weg $p = (v_1, v_2, \ldots, v_k)$, der Teilweg $p' = (v_i, \ldots, v_j)$ und ein weiterer Teilweg p''.

Da p' ein Teilweg von p ist können wir diesen Teilweg in p auch durch den Teilweg p'' ersetzen, da p'' ebenfalls von v_i nach v_j verläuft. Nun haben wir den Weg p kürzer gemacht, da wir den Teilweg p' durch den kürzeren Teilweg p'' ersetzt haben. Wir haben die Situation in Abbildung 41 dargestellt.

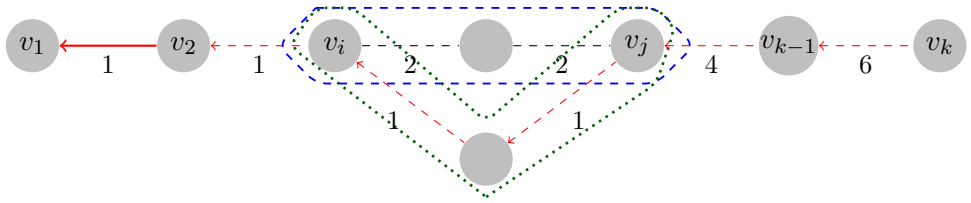

Abbildung 41.: Der Teilweg p' durch p'' ersetzt ergibt einen kürzeren Weg von v_1 nach v_k.

Dies ist jedoch ein Widerspruch zu unserer Ausgangssituation: p ist bereits der kürzeste Weg von v_1 nach v_k, das heisst wir können die Distanz nicht weiter verkleinern. Somit ist das Gegenteil von unserer Annahme wahr, das bedeutet, jeder Teilweg von v_i nach v_j (das heisst p') ist ebenfalls ein kürzester Weg von v_i nach v_j. Die Eigenschaft das jeder Teilweg eines kürzesten Weges wieder der kürzeste Weg ist nennen wir Optimalitätsprinzip. Auf dieser Eigenschaft beruht das gierige Verfahren des Algorithmus von Dijkstra. Wir können mit dem Anfangsknoten starten, da zu diesem der kürzeste Weg bereits bekannt ist. Anschliessend wählen wir aus der Menge der Randknoten denjenigen mit minimaler Distanz zum Anfangsknoten und verlängern somit schrittweise einen bisher gefunden kürzesten Weg. Durch das Optimalitätsprinzip bleiben bereits gefundene kürzeste Weg immer noch gültig. Wir bilden aus optimalen Teillösungen somit die optimale Gesamtlösung.

3.3. Laufzeit des Algorithmus von Dijkstra

Wir können eine Intuition dafür bekommen, welche Laufzeit der Algorithmus von Dijkstra besitzt, wenn wir uns überlegen, wie oft ein Knoten während der Ausführung betrachtet wird. In der ersten Phase (siehe Abschnitt 2.2.2) betrachten wir jeden Knoten aus V in $G = (V, E)$ mindestens einmal, dass bedeutet wir betrachten mindestens $|V|$ Knoten, einige vielleicht mehrmals, wegen der Prozedur *AktualisiereRandknoten*. Die Prozedur *AktualisiereRandknoten* iteriert über alle Kanten des übergebenen Knotens. Ein Knoten kann im schlechtesten Fall mit allen anderen Knoten verbunden sein, das heisst die Prozedur betrachtet maximal $|V| - 1$ Knoten. Kommen wir nun zur zweiten Phase. Die Schleife iteriert über die Randknoten. Da es vorkommen kann, dass jeder Knoten ausser der Anfangsknoten mindestens einmal zu den Randknoten gehört, betrachten wir also für den kompletten Schleifendurchlauf $|V| - 1$ Knoten. Pro Schleifendurchlauf wird aus den Randknoten ein Knoten mit minimaler Distanz ausgewählt. Diese Operation erfordert einen Vergleich aller Randknoten. Wie viele Randknoten sind pro Schleifendurchlauf vorhanden? Im schlechtesten Fall (das heisst im ersten Durchlauf) können bis zu $|V|-1$ Knoten vorhanden sein, das heisst wir betrachten auch hier $|V|$ Knoten. Zusätzlich rufen wir die Prozedur *AktualisiereRandknoten* auf, die nochmals $|V| - 1$ Knoten betrachtet (im schlechtesten Fall). Wir haben nun die Laufzeit für den schlechtesten Fall analysiert, indem wir die Anzahl der betrachteten Knoten untersucht haben. Wir können dies Zusammenfassen

mit $|V| + (|V| - 1)$ Betrachtungen in Phase 1 und $(|V| - 1) \times ((|V| - 1) + (|V| - 1))$ Betrachtungen im schlechtesten Fall für Phase 2.
Wie viele Knoten betrachtet die Brute-Force-Methode? Im schlechtesten Fall ist jeder Knoten mit jedem anderen Knoten verbunden. Wir sprechen dann von einem vollständigen Graphen (siehe für ein Beispiel Abbildung 42). Für jeden Anfangsknoten a existiert somit ein Weg zu einem beliebigen anderen Knoten, der $n-1$ Knoten besitzt, wobei n die Anzahl der Knoten im vollständigen Graph G_K beschreibt. Die Brute-Force-Methode muss also unter anderen Wege betrachten mit n Knoten. Da jeder Knoten mit jedem anderen verbunden ist, gibt es von einem Anfangsknoten aus $(n-1)!$ viele Wege mit n Knoten. Wir betrachten also hier deutlich mehr Knoten als im Algorithmus von Dijkstra. Diese Betrachtung zeigt bereits, dass der Algorithmus von Dijkstra gemessen an der Anzahl der betrachteten Knoten besser abschneidet. Die Anzahl der betrachteten Knoten beeinflusst auch die Laufzeit. Je mehr Knoten betrachtete werden müssen, desto Länger die Laufzeit. Somit ist der Algorithmus von Dijkstra tatsächlich schneller als die Brute-Force-Methode.

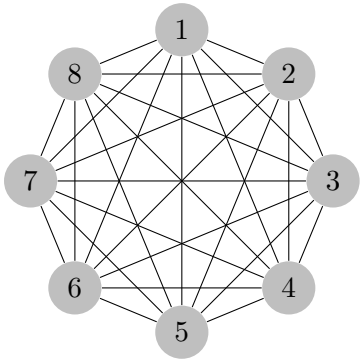

Abbildung 42.: Ein vollständiger Graph mit 8 Knoten.

3.4. Zusammenfassung

In diesem Kapitel haben wir das gierige Verfahren des Algorithmus von Dijkstra kennengelernt. Der Algorithmus verhält sich gierig, da einmal gewählte Knoten nicht mehr rückgängig gemacht werden und der kürzeste Weg zu diesen Knoten nicht mehr verändert wird. Wir haben die Korrektheit dieses Prinzips mit dem Optimalitätsprinzip von kürzesten Wegen bei der Hinzunahme einer weiteren Kante begründet und die Laufzeit mit Hilfe der Anzahl der betrachteten Knoten während der Ausführung analysiert und dabei festgestellt, dass diese Anzahl deutlich geringer ist als die Anzahl der betrachteten Knoten in der Brute-Force-Methode.

3.5. Lernkontrolle

1. Definieren Sie was man unter einem gierigen Algorithmus versteht. Erläutern Sie dann dieses Prinzip am Algorithmus von Dijkstra.

2. Gierige Verfahren tauchen auch in alltäglichen Situationen auf. An der Kasse versucht man das Rückgeld meist mit möglichst wenigen Münzen beziehungsweise Scheinen dem Kunden zu geben. Können Sie das gierige Prinzip dahinter erläutern? Welche Münze beziehungsweise Schein würden Sie bei einem Rückgeld von CHF 5.75 zuerst wählen?