Schriftliche Arbeit zur Pädagogischen Prüfung
für das Lehramt an Gymnasien

Routingalgorithmen im Informatikunterricht?
–
Ein Unterrichtsversuch in einem Informatikkurs der Jahrgangsstufe 12

von

Christian Datzko, Studienreferendar

Studienseminar Osnabrück

Osnabrück, 22. Mai 2007

Formatiert mit LaTeX 2_ε.

Einleitung

> „Über die ganze Zeit war die Informatik für viele immer ein Buch mit sieben Siegeln: In der Schule haben viele gar keinen Unterricht in diesem Fach gehabt und wenn, dann war das oft ein Grundkurs im Programmieren oder im Umgang mit Tabellenkalkulation und Textverarbeitung.
>
> Wie funktioniert aber ein Computer und vor allem, was steckt hinter den vielfältigen Programmen, die für ihn verfügbar sind? Wie kann unser PC in wenigen Sekunden die kürzeste Strecke von Berlin nach München bestimmen und dabei auch noch aktuelle Staumeldungen einbeziehen? Wie werden riesige Datenmengen in einem Augenblick sortiert? Woher nehmen die Banken die Gewissheit, dass ihr Portal für Internet-Banking sicher sei?“(Gallenbacher, 2006, S. VII)

Dieser Anspruch, der die Informatik nicht nur für Geeks spannend erscheinen lässt und große Konzepte im Gegensatz zu kleinen Details in den Vordergrund stellt, ist entscheidend für den Informatikunterricht des Gymnasiums Carolinum. In diesem Sinn soll diese Examensarbeit einen Teil dazu beitragen, die informatische Allgemeinbildung von Schülern[1] zu fördern.

Diese Arbeit ist die Hausarbeit gemäß §13 PVO-Lehr II[2]. Der in dieser Arbeit untersuchte Unterrichtsversuch steht unter dem Titel „Routingalgorithmen im Informatikunterricht?“. Das Fragezeichen macht die Zielrichtung deutlich: Ist es möglich – und wenn ja, wie – Routingalgorithmen in einem regulären Informatikkurs der Jahrgangsstufe 12 mit grundlegendem Anforderungsniveau erfolgreich zu vermitteln wären?

Die Planung hierzu ist in Kapitel 1 dargelegt. Die Schüler des Kurses if20 (siehe Kapitel 1.1) sollten den Dijkstra-Algorithmus erarbeiten. Als Anwendung sollten sie einen Graphen zur Modellierung der Wege in ihrer Schule erstellen. Die Durchführung der Examensreihe wird in Kapitel 2 berichtet und in Kapitel 3 kritisch ausgewertet. In Kapitel 3.4 fasse ich die Erkenntnisse zusammen.

Das Literaturverzeichnis, die verwendeten Materialien sowie exemplarische Schülerlösungen sind im Anhang angefügt. Dort sind auch die Versicherung, dass ich diese Arbeit alleine erstellt habe, und eine DVD-ROM zu finden. Diese DVD-ROM enthält neben einer digitalen Version dieser Arbeit und allen Materialien auch alle verwendeten und online verfügbaren Quellen.

[1] Im Weiteren benutze ich das Wort Schüler geschlechtsneutral.

[2] Verordnung über die Ausbildung und die Zweiten Staatsprüfungen für Lehrämter vom 18. Oktober 2001 (Nds.GVBl. Nr.28/2001 S.655; SVBl. 12/2001 S.485)

An dieser Stelle möchte ich mich bei meiner Frau für ihre Unterstützung und ihr Verständnis, bei Herrn Dr. Gieseke für seine geduldige und individuelle Betreuung während des Referendariats und bei den Kollegen des Gymnasiums Carolinum, die mir immer freundlich und hilfreich zur Seite standen, bedanken. Zuletzt gilt natürlich auch der Dank an die Schüler des Kurses if20, die das Thema willig und freundlich erarbeitet haben.

Inhaltsverzeichnis

1 Didaktisch-methodisches Konzept

In diesem Kapitel lege ich die Planungen und Überlegungen dar, die die Grundlage der Examensreihe bilden. Ausgehend von der Beschreibung der Lerngruppe in Kapitel 1.1 und der Sachanalyse in Kapitel 1.2 zeige ich die didaktischen (Kapitel 1.3) und methodischen Überlegungen (Kapitel 1.4) auf. Diese führen zu der geplanten Phasierung in Kapitel 1.5 und den Erwartungen an den Ertrag der Examensreihe in Kapitel 1.6.

1.1 Beschreibung der Lerngruppe

Der Kurs if20 des Gymnasiums Carolinum in Osnabrück ist ein Informatikkurs der Jahrgangsstufe 12 mit grundlegendem Anforderungsniveau. Er wurde zu Beginn des Schuljahres neu eingerichtet und wird von mir seidem eigenverantwortlich unterrichtet. Er wird nur bis zum Ende des Schuljahres bestehen, da keiner der Schüler das Fach Informatik als Prüfungsfach im Abitur gewählt hat. Er besteht aus zwei Schülerinnen und acht Schülern. Hervorzuheben ist, dass neun Schüler das gesellschaftswissenschaftliche Profil[1] und ein Schüler das sprachliche Profil gewählt haben. Es gibt in diesem Kurs kein Schüler, der das naturwissenschaftliche Profil gewählt hat, da es in derselben Jahrgangsstufe einen Informatikkurs mit erhöhtem Anforderungsniveau gibt[2]. Dies ist jedoch kein Zeichen von Leistungsverweigerung. Die Schüler sind durchgehend willig und fähig, angemessene Leistungen zu erbringen. Zudem herrscht in der Lerngruppe eine bemerkenswert freundliche Atmosphäre, die von Interesse am Fach geprägt ist, auch wenn dieses nicht so intensiv ausgeprägt ist, wie es zum Beispiel in einem Kurs mit erhöhtem Anforderungsniveau zu erwarten wäre.

Die Schüler des Kurses if20 haben in der 11. Klasse Informatik bei verschiedenen Fachlehrern gehabt, wurden jedoch in weiten Teilen nach den Vorgaben des Schulcurriculums unterrichtet, so dass auf die Vorkenntnisse des Schulcurriculums aufgebaut werden konnte. Unterschiede in der Schwerpunktsetzung in den Kursen sind jedoch an einigen Stellen bemerkbar, so haben zum Beispiel David und Tobias in der Programmiersprache Java nur begrenzte Vorkenntnisse aus der 11. Klasse. Tobias macht dies jedoch durch andere Vorkenntnisse wett, Katharina hat sich durch ihre ordentliche und gewissenhafte Arbeit besondere Fähigkeiten erarbeitet. Dies wird besonders in Kapitel 2.5 deutlich.

[1] Das Gymnasium Carolinum bietet in Umsetzung der Profiloberstufe in Niedersachsen drei Profile an: Das gesellschaftswissenschaftliche, das sprachliche und das naturwissenschaftliche Profil.

[2] Demzufolge wurde der Kurs if20 zum Beispiel von Florian als „Abdeckerkurs“ bezeichnet.

Nachdem sich der Kurs sich im 1. Halbjahr mit Datenbanksystemen und PHP[3] beschäftigt hat, haben die Schüler im Vorfeld der Examensreihe die Programmiersprache Java genauer kennengelernt. Dort wurden Grundlagen wie Schleifen, Datentypen und Reihungen[4] wiederholt und das Konzept der Rekursion erarbeitet. Über das Königsberger Brückenproblem[5] erfolgte der Einstieg in den Themenbereich der Graphen. Implementierungsmöglichkeiten wie Nachbarschaftsmatrix und Nachbarschaftsliste (vergleiche Kapitel 1.2) wurden erarbeitet. Anhand der Frage, ob ein Graph zusammenhängend ist, wurde das im Anhang in Kapitel B.1 abgedruckte Arbeitsblatt zum Entwickeln nicht-trivialer Algorithmen behandelt und ein vollständiges Programm geschrieben.

Ergänzend dazu ist es wichtig zu erwähnen, dass bestimmte Vorkenntnisse nicht vorhanden waren, die für eine erfolgreiche Umsetzung der Examensreihe nützlich sind: Die Schüler haben nur rudimentäre Erfahrungen in der Komplexitätstheorie und kennen übliche Komplexitätsklassen nicht. Zudem haben sie kaum Kenntnisse über Pseudocode gehabt.

1.2 Sachanalyse

Routing, das Finden kürzester Wege zwischen zwei Knoten in einem kantengewichteten Graphen, ist ein relativ altes Problem der Informatik. Routing bezeichnet das Festlegen von Wegen in Graphen, die bestimmten Bedingungen genügen sollen. Üblicherweise wird nach einem Weg gefragt, der kürzer oder genauso kurz wie alle anderen möglichen Wege ist. Spätestens mit der Realisierung paketvermittelter Netze wie dem ARPANET 1969 oder in Netzen im Fernmeldewesen war die Notwendigkeit gegeben, Algorithmen für eine automatische und effiziente Vermittlung zu nutzen. Angewendet wird Routing neben dem schon erwähnten Internet vor allem in Routenplanern. Es ist auch ein zentrales Moment in der autonomen Robotik (vergleiche Herzberg (2005)).

Zum besseren Verständnis sollte zunächst klargestellt werden, was ein Graph ist. Nach Vornberger (1997) und Diestel (2006) versteht man unter einem Graphen eine Menge von Knoten und Kanten. Knoten können durch einen oder mehrere Kanten verbunden sein, Kanten verbinden genau zwei Knoten miteinander. Werden den Kanten eines Graphen Werte zugewiesen, so spricht man von einem kantengewichteten Graphen. In diesem Fall soll das Kantengewicht größer als 0 sein, um Zyklen zu vermeiden, die kein oder gar ein negatives Gewicht haben. Multigraphen und Hypergraphen sind für diese Examensarbeit nicht relevant. Auch sollen Kanten keine Richtungen enthalten (vergleiche Kapitel 1.3.2).

Im Computer werden Graphen üblicherweise mittels einer Adjazenzmatrix beziehungsweise durch Adjazenzlisten dargestellt. Eine Adjazenzmatrix ist eine $n \times n$-Matrix, wobei n die Anzahl der Knoten des Graphen ist. In den Einträgen (i,j)[6] wird das Gewicht der Kante zwischen den

[3] PHP ist eine Skriptsprache, die üblicherweise auf einem Webserver läuft und automatisch HTML-Seiten generiert, dabei kann PHP auch auf Datenbanken zugreifen.

[4] Der Begriff Reihung wird von Softwareentwicklern häufig durch den englischen Begriff „Array“ ersetzt.

[5] Vergleiche Reed (1994) sowie Bibliographisches Institut in Leipzig (1880-1898).

[6] … und bei ungerichteten Graphen auch in den Einträgen (j,i)…

Knoten i und j gespeichert. Üblicherweise wird ein Eintrag (i,j) der Matrix, bei dem keine Kante zwischen i und j existiert, mit einem besonderen Wert, zum Beispiel 0 oder ∞ markiert. Im Weiteren wird eine Adjazenzmatrix auch Nachbarschaftsmatrix genannt.

In Adjazenzlisten hingegen wird zu jedem Knoten eine Liste aller benachbarten Knoten gespeichert. Ich habe eine weitere Form der Darstellung verwendet[7], die aus einer Liste von Kanten besteht. Diese nenne ich Nachbarschaftsliste[8]. Eine Nachbarschaftsliste hat zwar den Nachteil, dass in speziellen Graphen nicht alle Knoten berücksichtigt werden. Dies ist der Fall, wenn der Knoten mit dem höchsten Index mit dem Rest des Graphen nicht verbunden ist: Die Anzahl der Knoten muss indirekt über den höchsten vorhandenen Index in der Nachbarschaftsliste geschlossen werden. Für die konkret betrachtete Anwendung von Graphen ist dies jedoch irrelevant, da nur verbundene Knoten tatsächlich erreicht werden können. Daher ist die leichte Verständlichkeit ein großer Vorteil.

Mehrere Algorithmen sind als klassische Routingalgorithmen zu benennen: Der Bellman-Ford-Algorithmus (Bellman, 1958), der Dijkstra-Algorithmus (Dijkstra, 1959), der Floyd-Warshall-Algorithmus (Floyd (1962) sowie Warshall (1962)) und der A*-Algorithmus (Hart et al., 1968). Hiervon werden der Bellman-Ford-Algorithmus und der Dijkstra-Algorithmus regelmäßig für Routing im Internet verwendet (Wikipedia, 2007g).

Der *Bellman-Ford-Algorithmus* zeichnet sich laut Wikipedia (2007c) dadurch aus, dass er bei n Knoten und m Kanten $n-1$-Mal die momentanen Entfernungen aller Knoten an allen Kanten überprüft. Zudem kann er negative Zyklen erkennen. Seine Laufzeit ist $O(n \cdot m)$. Der *Dijkstra-Algorithmus* (Wikipedia, 2007d) jedoch flutet den Graphen vom Startknoten aus und verbessert die maximalen Entfernungen. Er ist effizienter mit einer Laufzeit von mindestens $O(n \cdot \log n + m)$, je nach Variante. Der *Floyd-Warshall-Algorithmus* berechnet, wie in Wikipedia (2007b) dargestellt, nicht nur die kürzesten Wege von einem bestimmten Knoten aus, sondern alle kürzesten Wege des Graphen. Folgerichtig ist er mit $O(m^3)$ ineffizienter. In diesem Algorithmus wird das Prinzip der dynamischen Programmierung angewendet. Zuletzt wird im *A*-Algorithmus* (Wikipedia, 2007a) eine Heuristik genutzt, um möglichst schnell ein Ergebnis zu haben. Dieses ist in Abhängigkeit von der Heuristik optimal. Auch dieser Algorithmus kann nicht schneller als $O(n \cdot n + m)$ sein.

Wie in Kapitel 1.3.2 dargelegt, habe ich mich für den Dijkstra-Algorithmus entschieden. Da die Originalpublikation von Dijkstra (1959) deskriptiv aufgebaut ist und wenige technische Details nennt, gibt es verschiedene Implementierungen, die jeweils einen unterschiedlichen Grad an Effizienz haben.

Der der Examensreihe zugrunde liegende Algorithmus ist im Anhang in Kapitel B.10 dargestellt. Er nutzt nicht alle möglichen Optimierungen, wie sie zum Beispiel von Sanders et al. (2006), Gallenbacher (2006, S. 19) oder Zürcher (2006) verwendet werden. Gründe dafür sind ebenfalls

[7] Diese Form wird in der mir bekannten Literatur nicht verwendet.

[8] Bedeutungsüberschneidungen mit einer üblichen Übersetzung von Adjazenzlisten als Nachbarschaftslisten kann zu Missverständnissen führen, weshalb ich sie in Zukunft Kantenliste nennen werde. Für diese Examensarbeit bleibe ich jedoch bei Nachbarschaftsliste. Vergleiche hierzu Kapitel 3.5.2.

in Kapitel 1.3.2 dargestellt. Das Prinzip jedoch ist dasselbe: Aktivieren des Startknotens, von allen aktiven Knoten aus alle direkt erreichbaren Knoten besuchen, den gegangenen Weg, so er denn kürzer ist als eventuell vorher schon gegangene Wege, dokumentieren und einen neu beziehungsweise auf kürzerem Weg erreichten Knoten aktivieren. Gibt es keine aktiven Knoten mehr, so kann der kürzeste Weg vom Zielknoten rückwärts zum Startknoten gefunden werden[9].

1.3 Didaktische Analyse

Die didaktische Analyse gliedert sich in eine Legitimation gemäß Kapitel 1.3.1 und eine didaktische Reduktion sowie die Abwägung von Alternativen entsprechend Kapitel 1.3.2. Zunächst wird begründet, warum das Thema der Examensreihe ein relevanter Unterrichtsgegenstand ist, und danach wird dieser sinnvoll eingegrenzt und analysiert. Die didaktische Phasierung ist in Kapitel 1.5 dargelegt, wo sie gleichzeitig mit den entsprechenden Sozialformen und Materialien und Medien verbunden wird.

1.3.1 Legitimation

Das Thema der Examensreihe berührt, wie in Kapitel 1.2 dargelegt, verschiedene Themenbereiche. Diese werden durch die derzeit noch immer gültigen Rahmenrichtlinien (Niedersächsisches Kultusministerium, 1993) legitimiert. Hierbei sind besonders Modellierung, Problemlösen und exemplarische Anwendungsbereiche von Computern als zentrale Aspekte zu nennen. Auch die Forderungen der EPA (Kultusministerkonferenz, 2004) korrelieren mit den Zielen der Examensreihe. Hier ist neben den schon durch die Rahmenrichtlinien legitimierten Bereichen insbesondere das Kommunizieren und Kooperieren zu nennen. Besonders zu erwähnen ist, dass in den Beispielen für schriftliche Abiturprüfungen ein ähnlich gelagerter Graphenalgorithmus geprüft wird (Kultusministerkonferenz, 2004, S. 36 ff.). Das Schulcurriculum Informatik des Gymnasiums Carolinum (Fachkonferenz Informatik Gymnasium Carolinum Osnabrück, 2007) sieht jedoch Graphen bisher nicht vor. Für das zweite Halbjahr der Jahrgangsstufe 12 in Informatik sind dort objektorientierte Modellierung und abstrakte Datentypen vorgesehen. Ein Abweichen ist jedoch aufgrund der Besonderheiten der Lerngruppe (vergleiche Kapitel 1.1) möglich. Spätestens ab 2008 werden Graphen in das Schulcurriculum aufgenommen, da dies in den Zentralabiturthemen 2010 in Niedersachsen für die schriftliche Prüfung verbindlich vorgeschrieben werden soll (siehe Niedersächsisches Kultusministerium (2007)).

Aber auch über die formale Legitimation hinaus ist die Beschäftigung mit Routing im Informatikunterricht sinnvoll. Hartmann et al. (2006, S. 32) erweitern die von Schwill in Schubert und Schwill (2004) übernommenen Kriterien für fundamentale Ideen, die ursprünglich von Bruner (1960) entwickelt wurden, derart, dass sie fünf Kriterien für fundamentale Ideen fordern: Horizontalkriterium, Vertikalkriterium, Zeitkriterium, Sinnkriterium und Repräsentationskriterium.

[9] Die Darstellung des Algorithmus in Hartmann et al. (2006, S. 120) ist zu verkürzt dargestellt, daher habe ich hier eine eigene Darstellung angegeben.

Diese Kriterien lassen sich nicht nur für fundamentale Ideen anwenden, sondern sind auch ein guter Anhaltspunkt für die Auswahl jeglicher Unterrichtsinhalte.

Dem Horizontalkriterium, also der Frage, ob es vielfältige Anwendungen gibt, genügt das Thema Routing vollkommen: In Datennetzen, Routenplanern, Navigationssystemen und vielem mehr ist die Frage nach dem kürzesten Weg wichtig (vergleiche auch Gallenbacher (2006, S. 37)).

Auch in Bezug auf das Vertikalkriterium ist Routing flexibel: Vom einfachen Finden des kürzesten Weges in einem einfachen Graphen bis hin zu komplexen Beweisen über Effizienz verschiedener Algorithmen kann Routing angewendet werden.

Die Frage nach dem kürzesten Weg stellte schon vor dem Aufkommen von Computern ein Problem dar. Stamm (2007) zum Beispiel verpackt den Algorithmus in eine glaubhafte Geschichte eines chinesischen Kaisers. Die 50er und 60er Jahre, in der die in Kapitel 1.2 genannten Routingalgorithmen entwickelt wurden, kann man in der Informatik bereits als historisches Zeitalter bezeichnen. Dass diese bis heute vielfach angewendet werden, erfüllt das Zeitkriterium.

Ein Alltagsbezug ist heute mehr denn je gegeben: Der Markt der mobilen Navigationssysteme boomt, und das Internet ist im Alltag der Menschen präsent wie nie zuvor. Insbesondere für Schüler der Jahrgangsstufe 12, die ihren Führerschein machen oder täglich im Internet surfen, ist damit das Sinnkriterium erfüllt[10].

Auch dem letzten Kriterium, dem Repräsentationskriterium, ist leicht ganz zu genügen: Von konkreten Wegen, zum Beispiel in der Schule, über Veranschaulichungen durch Karten, Abstraktion durch gezeichnete Graphen bis hin zur Repräsentation durch eine Nachbarschaftsliste oder Nachbarschaftsmatrix lassen sich Netze, Routen und auch Routingalgorithmen verschiedenst darstellen, um Schülern unterschiedlichste Anwendungsbeispiele zu veranschaulichen[11]. Damit kommt der Behandlung von Routing eine besondere Bedeutung zu.

1.3.2 Reduktion und didaktische Alternativen

Der Titel dieser Arbeit lautet „Routingalgorithm*en* im Informatikunterricht?“. Wie in der Einleitung auf S. III erläutert, ist dieser Titel als Frage nach der Möglichkeit zu verstehen. Daher ist es nicht Ziel der Examensreihe, möglichst viele verschiedene Routingalgorithmen zu unterrichten, wie zum Beispiel die in Kapitel 1.2 genannten, sondern vielmehr an einem paradigmatischen Beispiel die Möglichkeiten der Umsetzung zu untersuchen. Die Wahl dieses Algorithmus basiert auf den folgenden Überlegungen.

Wie in Kapitel 1.2 dargestellt, gibt es vier klassische Routingalgorithmen. Diese unterscheiden sich sowohl in ihrer Konzeption als auch in ihrer Effizienz. Also braucht es Kriterien für die Auswahl. Ich wähle dazu folgende Kriterien:

- Verständlichkeit für Schüler der Jahrgangsstufe 12

[10] Gallenbacher (2006, S. 1) stellt dies deutlich dar.
[11] Dies haben Hartmann et al. (2006, S. 118 ff.) selber dargelegt.

- verschiedene Anwendungsmöglichkeiten
- Einfachheit
- Eignung für entdeckendes Lernen (vergleiche Kapitel 1.4)

In der folgenden Tabelle werden die vier Algorithmen in Bezug auf die Kriterien analysiert:

Kriterium	*Bellman-Ford*	*Dijkstra*	*Floyd-Warshall*	*A**
Verständlichkeit	verständlich, jedoch scheint die $n-1$-malige Wiederholung zunächst nicht einsichtig	verständlich, benötigt jedoch ein Konzept „aktiver“ Knoten	als Weiterentwicklung des Bellman-Ford-Algorithmus verständlich	verständlich, der Intuition nahe, Optimalität ist nicht sofort einsichtig
Anwendungen	wird häufig eingesetzt	wird häufig eingesetzt	eher in speziellen Bereichen, bietet mehr als die anderen	wird häufig eingesetzt
Einfachheit	einfach aufgebaut	nicht trivial	einfach aufgebaut	durch die Heuristik komplex
Entdeckbarkeit	bedingt, die $n-1$ -malige Wiederholung ist zunächst nicht einsichtig	gegeben	nicht gegeben, da zunächst viel mehr berechnet wird und nur ein kleiner Teil des Ergebnisses benutzt wird	nicht gegeben, insbesondere wegen der Nutzung von Heuristiken

Analyse klassischer Routingalgorithmen

Nach Abwägen von Vor- und Nachteilen im Blick auf die obige Kriterien habe ich den Dijkstra-Algorithmus als Beispiel für einen Routingalgorithmus gewählt. Dies ist nicht unüblich, Gallenbacher (2006, S. 1 ff.), Hartmann et al. (2006, S. 118 ff.), Lutz-Westphal (2004), Sanders et al. (2006) und Zürcher (2006) verwenden ihn als paradigmatisches Beispiel von Routingalgorithmen. Röthlisberger und Wittmann (1994) verwendet zwei Routingalgorithmen, von denen jedoch einer wieder der Dijkstra-Algorithmus ist. Hingegen ist mir kein Beispiel bekannt, das nicht den Dijkstra-Algorithmus als *den* Routingalgorithmus für typische Anwendungen verwendet.

Die Examensreihe wird auf den Dijkstra-Algorithmus beschränkt, da dieser zum Einen als einziger den vier Kriterien aus meiner Sicht hinreichend genügt, und da zum Anderen eine Konzentration auf *einen* Algorithmus den Lernerfolg erhöht. Vermischungen der Vorgehensweise der unterschiedlichen Algorithmen können so von vornherein ausgeschlossen werden.

Eine weitere zu begründende Reduktion ist die, dass nur ungerichtete Graphen verwendet werden. Dies reduziert die realistischen Anwendungen, da zum Beispiel im Straßennetz Einbahnstraßen üblich sind. Diese Reduktion liegt an der gewählten Repräsentation von Graphen im Computer:

Eine Nachbarschaftsliste in meinem Sinne (vergleiche Kapitel 1.2) kann zunächst keine Richtung speichern. Der Dijkstra-Algorithmus jedoch arbeitet auf einer Nachbarschaftsmatrix, so dass eine Erweiterung auch zu einem späteren Zeitpunkt möglich wäre. Dies ist ein übliches Vorgehen. Auch Gallenbacher (2006) geht zunächst den Weg ungerichteter Graphen und erweitert den Algorithmus später zu gerichteten Graphen.

Wie in Kapitel 1.2 dargestellt, gibt es extrem unterschiedliche Versionen und Darstellungen des Dijkstra-Algorithmus. Hervorzuhebende Varianten sind sicherlich der Dreizeiler in Röthlisberger und Wittmann (1994, S. 4) und die Variante in der Originalpublikation von Dijkstra (1959). Häufig versteckt sich auch in einer „unverdächtig aussehenden" Zeile ein größerer Aufwand, so zum Beispiel in Zeile 4 des Pseudocodes von Sanders et al. (2006). Anders als die von mir verwendete Version (vergleiche Kapitel B.10 im Anhang) ist es zum Beispiel üblich, jeweils nur einen aktiven Knoten zu haben und nicht eine Liste aktiver Knoten. Diese Versionen haben gegenüber der von mir verwendeten den Vorteil, dass sie etwas effizienter sind, weil sie als Heuristik immer den bisher kürzesten Weg nehmen (so zum Beispiel in Gallenbacher (2006, S. 19) der Punkt 2 des Pseudocodes). Ein Weglassen dieser Optimierung, die den Algorithmus nicht prinzipiell verlangsamt, macht ihn jedoch leichter implementierbar, weshalb ich mich für eine Phasierung wie im Anhang in Kapitel B.3 entschieden habe.

Nach den in diesem Kapitel genannten Kriterien gibt es ein Problem am Dijkstra-Algorithmus, das behoben werden muss: Der Dijkstra-Algorithmus ist nicht trivial aufgebaut. Neben der Problemstellung müssen an mehreren Stellen als Hilfestellung Informationen in die Lerngruppe gegeben werden, um ihr zu helfen. Dazu gehört eine Unterscheidung zwischen aktiven und nicht aktiven Knoten und die damit verbundene Phasierung sowie die Vorgabe von Details der Implementierung, zum Beispiel wie der kürzeste Weg ausgegeben wird. Diese beiden Punkte sind nicht kompliziert, es bedarf aber einer genaueren Begründung.

Die Unterscheidung von Knoten in aktive und nicht-aktive Knoten ermöglicht beim Dijkstra-Algorithmus zu speichern, welche Knoten noch besucht werden muss. Dadurch werden die Schüler nach meiner Einschätzung nicht dazu verführt darüber nachdenken, wie sie das Problem einen kürzesten Weg zu finden in einem Schritt lösen könnten. Das Konzept aktiver Knoten knüpft zudem an einen den Schülern bekannten Algorithmus an, der überprüft, ob ein Graph zusammenhängend ist. Eine Alternative hat Stamm (2007) vorgelegt, indem er in seiner Einstiegsgeschichte die Berechnungen parallel laufen lässt und zudem den Blickwinkel von vornherein auf den Weg zwischen zwei Knoten begrenzt. Lutz-Westphal (2004) nennt dies passend „Froschperspektive". Die Umsetzung dessen in einen einzelnen Algorithmus wäre nicht trivial, hierzu würde eine Vorrangwarteschlange benötigt.

Die Details der Implementierung, die vorgegeben werden müssen, sind zum Einen Hilfen zum Umsetzen der strukturierten, aber umgangssprachlichen Formulierung des Dijkstra-Algorithmus und zum Anderen zeitsparende Hinweise zum Konzipieren der verschiedenen Datentypen. Dies ist laut Hartmann et al. (2006, S. 22 ff.) und Humbert (2005, S. 72) legitim, beide sprechen sich dafür aussprechen, der Implementierung und den technischen Details kein zu großes Gewicht beizumessen, um den allgemeinen Konzepten und Ideen mehr Raum zu geben. Dies gilt insbesondere für die Lerngruppe.

Das Entdecken eines Routingalgorithmus reicht für eine vollständige Bearbeitung von Routing nicht aus. Erst eine praktische Anwendung macht ihn für die Schüler lebensnah erfahrbar. Da bietet es sich an, ein konkretes Wegenetz zu untersuchen. Hierfür gibt es viele Möglichkeiten. In der von mir betrachteten Literatur ist von Fluglinien (Röthlisberger und Wittmann, 1994), vollständigen U-Bahnnetzen (Lutz-Westphal, 2004) bis hin zu Straßenkarten (Gallenbacher, 2006) alles zu finden. Es überrascht jedoch nicht, dass das Internet in der Regel nicht als Einstiegsbeispiel verwendet wird, da die Frage nach dem optimalen Weg von Datenpaketen zu weit entfernt ist von der Lebensrealität der meisten Schüler. Die verwendeten Beispiele haben jedoch alle eine Gemeinsamkeit: Sie sind in einem überschaubaren Rahmen gehalten. Die Anzahl der Knoten und Kanten liegt meist unter 100, so dass eine manuelle Plausibilitätsprüfung oder gar ein manueller Beweis des optimalen Weges einfach durchgeführt werden kann. Die Beispiele sind jedoch auch umfangreich genug, dass sich der Einsatz eines Routingalgorithmus lohnt.

In Osnabrück würde sich zum Beispiel das Stadtbusnetz anbieten. Es besitzt einen gewünschten Umfang und ist nicht rein sternförmig aufgebaut, so dass es echte Alternativen gibt, die zu bedenken wären, zum Beispiel für eine Verbindung zwischen dem Hauptbahnhof und der Haltestelle Hasetor. Auch die Straßen der Innenstadt sind verwinkelt genug, so dass die Frage nach einem kürzesten Weg eine echte Frage ist. Ich habe mich jedoch für die Wege in der Schule entschieden: Durch die Umbauten der letzten Jahre gibt es viele Wegealternativen (vergleiche Kapitel B.5 im Anhang). Die Bedingung der Überschaubarkeit ist bei gleichzeitigen echten Fragestellungen erfüllt. Zudem ist die Schule innerhalb einer Doppelstunde arbeitsteilig vermessbar. Da alles auf dem Schulgelände stattfindet, gibt es keine praktischen Fragen nach der Aufsichtspflicht oder Unfallgefahren, die auftreten könnten, wenn Schüler das Schulgelände verlassen müssten, um Wege auszumessen.

Eine weitere wichtige didaktische Entscheidung ist die, den Konstruktivismus[12] als Lernmodell zu verwenden. Im bewussten und unbewussten Nachdenken über Lerngegenstände konstruiert sich jeder Schüler ein Bild des Lerngegenstandes in seinem Kopf. Dieses ist zunächst grob und potentiell fehlerhaft, und muss daher mit der Zeit verbessert werden. Hierzu dienen metakognitive Tätigkeiten, die durch das Bewusstmachen des eigenen Denkens Fehlvorstellungen effektiv korrigieren können (Sjuts, 1999, S. 42-44).

In Abgrenzung zum radikalen Konstruktivismus (Leuders, 2003, S. 47) soll hier davon ausgegangen werden, dass es tatsächlich einen Lerngegenstand gibt, dessen Existenz bestimmte Entscheidungen vorgibt. So wird einer möglichen Orientierungslosigkeit bezüglich des Lerngegenstands vorgebeugt. Der daraus folgende pragmatische Konstruktivismus fordert daher die Integration folgender Punkte: Anknüpfen an schon vorhandenes Können und Wissen, Aktivieren der Schüler, Integration von Phasen der Metakognition sowie das Ansprechen von Fehlvorstellungen. Dies ist ein entscheidendes Moment für die Phasierung in Kapitel 1.5.

[12] Leuders (2003, S. 36) schreibt: „Diese konstruktivistische Vorstellung ist [...] durchaus noch zeitgemäß“, während er Piagets Behaviorismus in Frage stellt.

Mein Hauptanliegen ist es also, den Schülern in der Examensreihe den Dijkstra-Algorithmus zu vermitteln und ihn anhand eines selbst zu erstellenden Graphen, der die zu Fuß zurückzulegenden Wege im Gymnasium Carolinum modelliert, erfahrbar zu machen. Die dabei zu erreichenden Lernziele sind in Kapitel 1.6 formuliert.

1.4 Methodische Analyse

Die methodische Analyse gliedert sich in übergreifende Überlegungen in Kapitel 1.4.1, in dem die grundlegenden methodischen Überzeugungen begründet werden, Überlegungen zu geeigneten Sozialformen in Kapitel 1.4.2 und einigen Gedanken zu den verwendeten Materialien und Medien in Kapitel 1.4.3.

1.4.1 Übergreifende Überlegungen

Meiner Examensreihe liegen mehrere methodische Überlegungen zugrunde. Zunächst soll das Lernen entdeckend stattfinden. Dies erfordert ein gewisses Maß an Offenheit des Unterrichts [13], die Schüler sollen divergent und potentiell kreativ[14] denken. Da jedoch der Lerngegenstand von sich aus nicht divergent angelegt ist, muss diese Offenheit an bestimmten Stellen beschnitten werden. Humbert (2005, S. 42) gibt ein Beispiel für eine mögliche Vorgehensweise, indem er den Schülern zunächst prozessorientierte und später ergebnisorientierte Lernhilfen zur Verfügung stellt.

Dieser scheinbare Widerspruch zwischen Offenheit und Geschlossenheit kann durch das Sandwich-Prinzip gelöst werden (Herold und Landherr, 2003, S. 79 ff.). Eine sinnvolle Abfolge von individuellen und kollektiven Phasen fördert divergentes und konvergentes Denken. So ist sowohl ein gewisses Maß an Offenheit gewährleistet, als auch eine Vereinheitlichung des Lernstandes. Zudem fördert es die Kommunikation über eigene Vorstellungen, was wiederum die Metakognition anregt und somit eine zentrale Forderung des Konstruktivismus erfüllt.

Ein weiteres zentrales Moment der Examensreihe ist die Handlungsorientierung: Sie soll die Schüler aktivieren und erhöht damit nachweislich die Leistungen (vergleiche Gudjons 2003, S. 102 ff., Meyer 1987a, S. 214 ff. sowie Meyer 1987b, S. 395 ff.). Sie ist ein wesentlicher Grund für die Forderung einer praktischen Anwendung in Kapitel 1.3.2.

Nach dem Sandwich-Prinzip steht am Anfang einer Unterrichtsreihe ein Advance Organizer (Herold und Landherr 2003, S. 61 ff. sowie Hartmann et al. 2006, S. 109 ff.). Ein Advance Organizer ist ein Überblick über den Lerngegenstand und ermöglicht den Schülern, im Kopf zu überprüfen, wo sie gerade stehen. Dieser kann auf verschiedene Art vorgestellt werden, zum Beispiel als kur-

[13] „Die Lernenden erlangen neues Wissen, indem sie persönliche Erfahrungen machen und Dinge hinterfragen. Sie entwickeln durch Staunen, sich Wundern und Zweifeln ihre eigenen (vieleicht naiven) Theorien und müssen dabei mitunter alte Vorstellungen und zuvor aufgestellte Hypothesen verwerfen“ (Hartmann et al., 2006, S. 92).

[14] Ich verwende diese Einschränkung des Wortes „Kreativität“ bewusst, da von Hentig (1998) zu Recht eine inflationäre Verwendung dieses Begriffes und einen damit verbundenen Bedeutungsverlust beklagt.

zer Vortrag oder als Visualisierung. Ich wähle hier den Vortrag, da der Lerngegenstand schlecht visualisiert werden kann. Danach sollten die Schüler möglichst eigenständig vorgehen können – eine sinnvolle Phasierung entsteht durch die von Humbert (2005, S. 42) genannten Einschnitte.

Ein wichtiger Bestandteil des Advanced Organizers ist das Aufwerfen des zentralen Problems, das es zu lösen gilt. Humbert (2005, S. 70) nennt das Lösen eines Problems sogar eine „zentrale Kategorie des Informatikunterrichts" und erhebt es zur Methode. Ein einzelnes Problem kann tatsächlich eine ganze Unterrichtsreihe tragen (Humbert, 2005, S. 71).

Damit folgt die Examensreihe wesentlichen Aspekten von Selbstorganisiertem Lernen. Es wird nicht vollständig umgesetzt, was, wie bei Herold und Landherr (2003, S. 2 ff.) angedeutet, auch aus der Sicht von Selbstorganisiertem Lernen akzeptabel ist.

1.4.2 Sozialformen

Aus den grundlegenden methodischen Überlegungen in Kapitel 1.4.1 lässt sich eine natürliche Vielfalt von Sozialformen ableiten. Das Sandwich-Prinzip erfordert Phasen der individuellen Erarbeitung beziehungsweise der Erarbeitung in kleinen Gruppen, aber auch Diskussionen im Plenum. Dies wird umso deutlicher, wenn man bedenkt, dass die Examensreihe nach Hartmann et al. (2006, S. 65) projektartige Züge trägt, was laut Humbert (2005, S. 70) typisch für den Informatikunterricht ist.

Für Phasen, in denen Divergenz zugelassen werden soll, jedoch ein Austausch zwischen Schülern notwendig ist, bieten sich kleine Gruppen an, die alle an derselben Aufgabe arbeiten, diese aber unterschiedlich bearbeiten können. Das Erfinden eines Algorithmus (vergleiche Kapitel 1.5.1) ist ein Beispiel für eine solche Phase. Die Lösungswege können verschieden sein, aber die Schüler brauchen ein gegenseitiges Korrektiv um ihre Kreativität kontrollieren zu können (von Hentig, 1998, S. 11). Das dafür notwendige Reden über Informatik ist unerlässlich, sowohl für schulische Zwecke im Sinne von Kommunikation zum Austausch, Korrektur von Fehlvorstellungen oder Peer Tutoring als auch in der Industrie (vergleiche Müller und Padberg (2007, S. 2)).

Phasen, die zusammenführen, werden sinnvoll in Plenumsdiskussionen realisiert. Hier kann jeder sich mit seinen Ideen und Argumenten einbringen. An anderen Stellen gibt es Phasen, die parallel laufen können (vergleiche Abbildung 1 auf S. 12). Diese können in arbeitsteiligen Gruppen ablaufen, zum Beispiel das parallele Vermessen der Schule und Implementieren des Dijkstra-Algorithmus (vergleiche Kapitel 1.5.3).

Das Gruppenpuzzle bietet die Möglichkeit, beide Elemente, das Paralle und das divergent Arbeitsteilige, effizient miteinander zu vereinen. Insbesondere Herold und Landherr (2003) weisen auf die Vorzüge dieser Sozialform hin. Sie bietet sich sowohl für die Erarbeitung neuer Inhalte als auch wie in diesem Fall für die vertiefende Erarbeitung von Informationen an (vergleiche Kapitel 1.5.4).

1.4.3 Materialien/Medien

Die verwendeten Materialien sind in Anhang in Kapitel B aufgeführt. Die entsprechenden Quellen sind dort genannt. Im Kern beschränken sich die Materialien auf eine Graphik aus Röthlisberger und Wittmann (1994), die als Dijkstra-Beispiel umgearbeitet ist, sowie Informationen zum Aufbau der Schule. Die Beschränkung auf dieses eine graphische Beispiel soll den Wiedererkennungswert fördern und Konfusion durch verschiedene Beispiele vermeiden. Es beinhaltet alle relevanten Momente im Sinne von Kapitel 1.3.2. Die Abfolge der Schritte in der Hausaufgabe im Anhang in Kapitel B.3 stellt eine Repräsentationsform des Dijkstra-Algorithmus dar. Die anderen Repräsentationen als umgangssprachliche Beschreibung, in Pseudocode und in Java werden von den Schülern selber erarbeitet.

Neben den üblichen Medien wie Tafel oder Tageslichtprojektor werden im Informatikunterricht auch sogenannte „Neue Medien" verwendet. Den Schülern steht bei Bedarf je ein internetfähiger Computer zur Verfügung. Dieser bietet zusätzlich den Java-Editor[15] mit installiertem Java Development Kit[16].

1.5 Phasierung

Nach der Klärung der Frage, was die Schüler lernen sollen, stellt sich die Frage, wie der Lerngegenstand sinnvoll schrittweise eingeführt werden kann. Die gewählte Phasierung ist in Abbildung 1 auf Seite 12 gezeigt. Sie korelliert dabei in Bezug auf die Entwicklung des Dijkstra-Algorithmus mit den Vorgaben des den Schülern bekannten Arbeitsblattes im Anhang in Kapitel B.1. Zur Sicherung und Vertiefung schließt sich eine weitere Arbeitsphase mit einem abschließenden Test an. Parallel zur Entwicklung des Algorithmus wird das Anwendungsbeispiel konkret erarbeitet. Wie das Ablaufdiagramm zeigt, ist ein konzeptionelles Wissen notwendig, bevor konkrete Messungen gemacht werden können. Gleichzeitig ist es zur Sicherung sinnvoll, den entwickelten Algorithmus anhand des Anwendungsbeispiels auszuprobieren.

1.5.1 1./2. Stunde (Doppelstunde): Einstieg und erste Begegnung mit Routing

Die Problematisierungsphase beginnt unüblicherweise schon vor dem Advance Organizer. In dem Vortest[17] werden neben einigen Überprüfungen (Aufgaben 1, 2 und 4) schon die ersten Aufgaben zum Thema Routing behandelt. Konkret soll anhand eines Beispiels ein kürzester Weg gefunden werden (Aufgabe 3a). Der Weg der Lösungsfindung soll zugleich reflektiert werden, um die Metakognition anzuregen. Dadurch, dass dieses schon im Vortest geschieht, wird jeder einzelne Schüler dazu angeregt, sich mit dem Thema auseinanderzusetzen. Die anschließende Sicherung soll nur in Form einer einfachen Verständnisfrage sicher stellen, dass der kürzeste Weg A – D – E – C von allen als solcher erkannt wird.

[15]URL: `http://lernen.bildung.hessen.de/informatik/javaeditor/index.htm`
[16]URL: `http://java.sun.com/javase/`
[17]Dieser Vortest ist im Anhang in Kapitel B.2 abgedruckt.

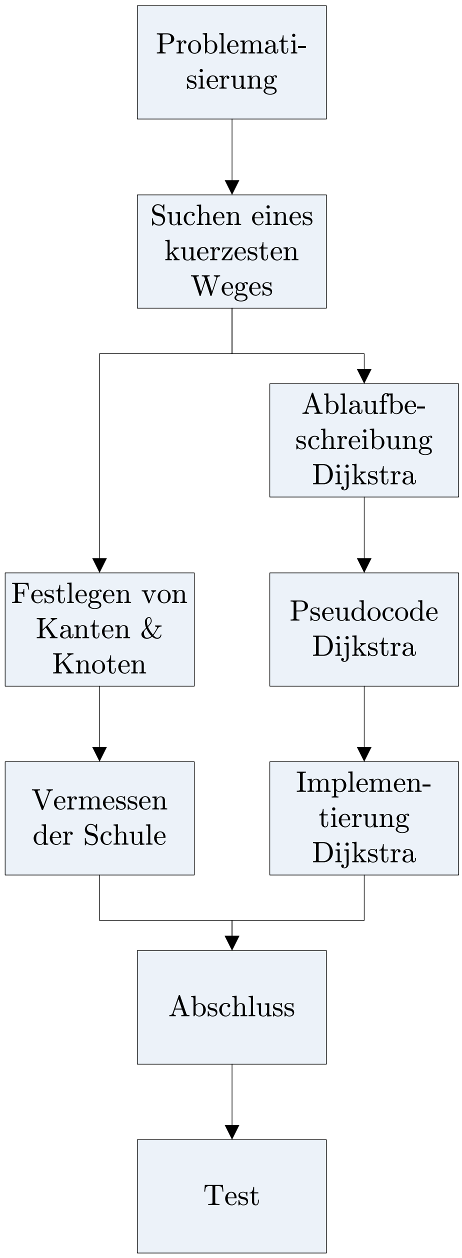


Abbildung 1: Phasierung

Die darauf folgende Arbeitsphase mit den beiden Sicherungen geht zunächst in der Gruppe und dann im Plenum in medias res, indem die Schüler dort ihre Ideen und Ansätze zum Finden eines kürzesten Weges erarbeiten, sich gegenseitig vorstellen und beurteilen können. Die Forderung nach einem gemeinsamen Gruppenergebnis zwingt die Schüler dazu, bei unterschiedlichen Vorschlägen Gründe für und wider die einzelnen Vorschläge zu nennen. Das abschließende Unterrichtsgespräch wiederholt, was schon in den einzelnen Gruppen geschehen ist, bietet jedoch zudem die Gelegenheit, dass auftretende Fehlvorstellungen im Plenum ausführlich besprochen werden können. Das divergente Denken der Schüler im Test und teilweise in den Gruppen wird hier gewürdigt, bevor in der Hausaufgabe (siehe Kapitel B.3 im Anhang) der erste Kontakt mit dem Dijkstra-Algorithmus stattfindet.

Unterrichts-phase	**Stichworte zum Inhalt**	**Sozialform**	**Material/Medium**
Einstieg	Kurztest	Einzelarbeit	Arbeitsblatt (siehe Kapitel B.2 im Anhang)
Sicherung	Vergleich des kürzesten Weges	Unterrichts-gespräch	Graph auf Folie
Advance Organizer	Vorstellen des Vorhabens: Vermessen der Schule, Erarbeiten eines Algorithmus zur Suche des kürzesten Weges	LV	–
Erarbeitung	Umgangssprachliche Formulierung, wie man den kürzesten Weg finden könnte (Punkt 2 des Arbeitsblattes im Anhang in Kapitel B.1)	3er Gruppenarbeit	Graph des Arbeitsblattes
Sicherung I	Gemeinsame Formulierung in den Gruppen	3er Gruppenarbeit	–
Sicherung II	Präsentation und Diskussion der Gruppenergebnisse	Unterrichts-gespräch	–
Hausaufgabe	Begegnung mit dem Dijkstra-Algorithmus	–	Arbeitsblatt mit Dijkstra-Beispiel (siehe Kapitel B.3 im Anhang)

Übersicht über den geplanten Stundenverlauf der 1./2. Stunde

1.5.2 3. Stunde: Der Dijkstra-Algorithmus

Durch die Hausaufgabe werden die Schüler eine Vorstellung des Dijkstra-Algorithmus erlangt haben. Diese ist jedoch potenziell divergent und fehlerhaft, daher folgt analog zu den ersten beiden Stunden eine Gruppenarbeit, in der dies diskutiert werden kann. Zur Sicherung und Homogenisierung erfolgt wieder eine Plenumsphase. Der dort entwickelte Pseudocode wird aufgrund der fehlenden Übung im Umgang mit Pseudocode (vergleiche Kapitel 1.1) teilweise durch den Lehrer gelenkt sein.

Wie in Abbildung 1 auf S. 12 dargestellt, kann die Implementierung unabhängig von den Vermessungen erfolgen. Da das Prinzip des Dijkstra-Algorithmus den Schülern an dieser Stelle klar ist, kann darauf verzichtet werden, dass alle Schüler diesen Algorithmus implementieren. Humbert (2005, S. 72) schreibt dazu: „[. . .] das Ziel des Informatikunterricht besteht nicht in der Qualifikation von Softwareentwicklerinnen, sondern in der Vermittlung allgemeinbildender Elemente der Informatik“. Er relativiert dies, indem er wie Hartmann et al. (2006, S. 23 ff.) darauf hinweist, dass beides Teil des Informatikunterrichts sei. In konkreten Fällen jedoch ist ein arbeitsteiliges Vorgehen möglich. Ähnliches gilt für das Vermessen der Schule: Dass etwas konkret vermessen wird, ist unabdingbar, wie in Kapitel 1.3.2 dargestellt. Jedoch ist es möglich, dass dies nur von einem Teil der Schüler gemacht wird, die ihre Erfahrungen dann mit den anderen Schülern teilen.

Als Vorbereitung für die nächste Stunde wird die weitere Vorgehensweise besprochen. Somit können einzelne Arbeiten schon in der Hausaufgabe vorbereitet werden. Die Schüler müssen eingeteilt werden, das konkrete Vorhaben muss besprochen und die Hausaufgaben müssen verteilt werden. Diese sind konkret das Kennenlernen des Programmrahmens (siehe Kapitel B.4 im Anhang) und das Festlegen der Knoten und Kanten in den einzelnen zu vermessenden Abschnitten der Schule[18].

Unterrichtsphase	**Stichworte zum Inhalt**	**Sozialform**	**Material/Medium**
Einstieg	Vergleich der Hausaufgabe	Unterrichtsgespräch	Folie mit Arbeitsblatt (siehe Kapitel B.3 im Anhang)
Erarbeitung	Ablaufbeschreibung des Dijkstra-Algorithmus (Punkt 3 des Arbeitsblattes im Anhang in Kapitel B.1)	3er Gruppenarbeit	Arbeitsblatt (siehe Kapitel B.3 im Anhang)
Sicherung	Pseudocode des Dijkstra-Algorithmus (Punkt 4 des Arbeitsblattes im Anhang in Kapitel B.1)	Unterrichtsgespräch	Arbeitsblatt (siehe Kapitel B.3 im Anhang), Tafel
Erarbeitung	Planen der nächsten Doppelstunde: Welche Informationen müssen gesammelt werden? Wie wollen wir die Informationen notieren? Wer macht was?	Unterrichtsgespräch	Tafel, Folien mit Grundriss der Schule (siehe Kapitel B.5 im Anhang), Beispiel
Hausaufgabe	Vorbereitung: Festlegung der Orte der Knoten und der Kanten auf dem Grundriss/Analyse des Rahmenprogramms für Dijkstra-Algorithmus	–	Kopien mit Grundriss der Schule (siehe Kapitel B.5 im Anhang)

Übersicht über den geplanten Stundenverlauf der 3. Stunde

1.5.3 4./5. Stunde: Vermessen der Schule und Implementierung

Nach den Vorbereitungen in der 3. Stunde kann in dieser Doppelstunde in weiten Teilen selbstständig gearbeitet werden. Nach kurzen Absprachen über die Benennung von Knoten an Schnittpunkten der zu vermessenden Teilgebiete können die entsprechenden Strecken gemessen und notiert werden[19]. Auch die Gruppe, die implementiert, kann aufgrund des Pseudocodes weite Teile selber erarbeiten. Der Lehrer kann an dieser Stelle zu einem Berater werden, der für Fragen und Hilfestellungen zur Verfügung steht, sich aber ansonsten komplett im Hintergrund hält.

[18] Erdgeschoss Innen, Erdgeschoss Außen, 1. Obergeschoss und 2. Obergeschoss; vergleiche Kapitel B.5 im Anhang.

[19] Hierzu bekommen die Schüler eine Tabelle wie im Anhang in Kapitel B.6, die eine Überführung in eine Nachbarschaftsliste erleichtert.

Um die Erfahrungen der unterschiedlichen Gruppen auszutauschen, ist es gegen Ende der Doppelstunde notwendig, dass die Schüler von ihren Erlebnissen berichten. Diese enthalten potenziell auch wieder Fehlvorstellungen, die gelöst wurden, und zeigen eventuelle Absprachefehler auf. Um die Ergebnisse in der nächsten Stunde zusammenfließen zu lassen, ist es noch notwendig, das erarbeitete Anwendungsbeispiel und den implementierten Dijkstra-Algorithmus zu verbinden. Hierzu müssen die Ergebnisse der Messungen als Hausaufgabe digitalisiert werden.

Unterrichtsphase	**Stichworte zum Inhalt**	**Sozialform**	**Material/Medium**
Erarbeitung	Vermessen der Schule *sowie parallel dazu*	Partnerarbeit	Maßbänder, Schulplan (siehe Kapitel B.5 im Anhang), Listen für Ergebnisse (siehe Kapitel B.6 im Anhang)
Erarbeitung	Implementierung des Dijkstra-Algorithmus in Java	Partnerarbeit	Computer, Programm als Grundgerüst (siehe Kapitel B.4 im Anhang)
Sicherung	Gegenseitiges Vorstellen der Ergebnisse/Erfahrungen	Unterrichtsgespräch	Computer/Beamer
Hausaufgabe	Eingeben der Ergebnisse in den Computer	–	–

Übersicht über den geplanten Stundenverlauf der 4./5. Stunde

1.5.4 6. Stunde: Abschluss

Zum Abschluss soll das Thema Routing mit einem Gruppenpuzzle abgerundet werden. An dieser Stelle können die Schüler schon die ersten direkt sichtbaren Früchte ihrer Arbeit genießen: Das Programm ist geschrieben und die Daten sind gesammelt und eingegeben. Die in Kapitel 1.3.2 angesprochenen echten Fragestellungen können nun geklärt werden.

An dieser Stelle gibt es jedoch immer noch ungeklärte Momente, die abschließend betrachtet werden sollten. Mit einem Gruppenpuzzle anhand des Materials im Anhang in Kapitel B.7 können die Aufgaben effizient bearbeitet werden. Diese sind eine Analyse des Algorithmus (Gruppe C des Gruppenpuzzles), eine Beantwortung der im Vortest angelegten Frage nach Anwendungen, angereichert durch historische Informationen (Gruppe A des Gruppenpuzzles) und eine genauere Betrachtung der Anwendung im Internet (Gruppe B des Gruppenpuzzles). Die Aufgabe 3 der Gruppe A wurde aufgrund der Beobachtungen in der Einstiegsstunde (vergleiche Kapitel 2.3) eingeplant.

Ein abschließender Test (siehe Kapitel B.8 im Anhang) überprüft, ob die geforderten Inhalte auch tatsächlich erlernt wurden[20], eine Umfrage (siehe Kapitel B.9 im Anhang) gibt den Schülern die Möglichkeit, ihre Meinungen über die Examensreihe anonym kund zu tun.

Unterrichts-phase	**Stichworte zum Inhalt**	**Sozialform**	**Material/Medium**
Sicherung	Ausprobieren des Programmes	Unterrichts-gespräch	Computer/Beamer, Schulplan mit Knoten und Kanten (siehe Kapitel C.7 im Anhang)
Erarbeitung	Gruppenpuzzle (Dijkstra, Routing im Internet, Analyse des Java-Programms)	Gruppen-puzzle	Arbeitsblätter (siehe Kapitel B.7 im Anhang), Internet
Sicherung	Test/Umfrage	Einzelarbeit	Arbeitsblatt (siehe Kapitel B.8 im Anhang)

Übersicht über den geplanten Stundenverlauf der 6. Stunde

1.6 Lernziele

Global betrachtet sollen die Schüler also mit dem Dijkstra-Algorithmus einen Algorithmus zum Finden eines kürzesten Weges erarbeiten, anwenden und wiedergeben können.

Aus der oben formulierten Phasierung der Examensreihe erwarte ich, dass die Schüler die folgenden Teillernziele erreichen.

Inhaltliche Lernziele: Die Schüler sollen...

- Graphen zur Modellierung von Wegenetzen anwenden können.
- Graphen zur Modellierung von Datennetzen anwenden können.
- ein Maß für Wege angeben können.
- den Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges in einem ungerichteten, kantengewichteten Graphen beschreiben können.
- einen umgangssprachlich formulierten Algorithmus schrittweise formalisieren können.
- die Umsetzung dieses Algorithmus in die Programmiersprache Java nachvollziehen können.
- Algorithmus-Ideen verständlich formulieren können.

[20] Dies hätte auch durchaus in einer Klausur geschehen können, die eine intensivere Evaluation des Lernzuwachses ermöglicht hätte. Da der Termin aufgrund der schulweiten Festlegung von Klausurterminen in der Kursstufe nicht innerhalb der Bearbeitungszeit dieser Arbeit lag, konnten die Ergebnisse keinen Niederschlag in der Arbeit finden.

Soziale Lernziele: Die Schüler sollen. . .

- verschiedene Lösungsansätze durch Kommunikation zu einem gemeinsamen Lösungsansatz umarbeiten können.
- arbeitsteilige Aufträge selbstständig planen und durchführen können.
- Ergebnisse ihrer Arbeit präsentieren können.

Von diesen Lernzielen können die letzten beiden inhaltlichen beziehungsweise sozialen Lernziele als übergeordnet betrachtet werden, die in verschiedenen Bereichen gefördert werden. Ebenso wie die anderen können sie aber konkret auf die Examensreihe bezogen und damit auch überprüft werden.

2 Bericht über die Durchführung des Unterrichts

Dieses Kapitel berichtet über die Durchführung der geplanten Unterrichtsreihe. Anhand der in den Kapitel 1.5 vorgegebenen Phasierung wird der Unterricht chronologisch nachgezeichnet. Dieser Bericht bildet die Grundlage der Reflektion in Kapitel 3.

Die Aussagen in diesem Kapitel werden hauptsächlich durch schriftliche Schüleräußerungen belegt. Diese sind im Anhang in Kapitel C wiedergegeben[1]. Dort sind alle Zitate abgedruckt. Diese gelten exemplarisch für alle schriftlichen Schüleräußerungen, die nicht explizit erwähnt werden.

2.1 Überblick über den Unterrichtsverlauf

Die Examensreihe lief im Großen und Ganzen wie geplant (für eine detaillierte Analyse siehe Kapitel 3.2.2). In der ersten Doppelstunde, die in Kapitel 2.3 dargestellt wird, wurden in einem Vortest das Vorwissen abgefragt und das Problem des Routing aufgeworfen. Die Lösungsideen für das Finden kürzester Wege wurden in einer Gruppenarbeit diskutiert und abschließend im Plenum besprochen. Hierbei wurde unter anderem zum ersten Mal diskutiert, ob ein Algorithmus alle möglichen Wege gehen sollte. In der Hausaufgabe wurden die Schüler mit dem Dijkstra-Algorithmus konfrontiert.

In der dritten Stunde, die in Kapitel 2.4 dokumentiert wird, wurde der Dijkstra-Algorithmus zum Finden kürzester Wege beschrieben und im Pseudocode formalisiert. In der Hausaufgabe wurde das Vermessen der Schule vorbereitet.

In der danach folgenden Doppelstunde, die in Kapitel 2.5 beschrieben wird, wurde von einem Teil der Schüler der Dijkstra-Algorithmus in der Programmiersprache Java implementiert, die anderen Schüler vermaßen die Schule. Am Ende der Doppelstunde wurden die Ergebnisse zusammengetragen und kurz vorgestellt.

In der letzten Stunde, die in Kapitel 2.6 dargestellt wird, wurde in einem Gruppenpuzzle der entwickelte Algorithmus von allen Schülern untersucht, zudem wurde das Thema Routing weiter vertieft.

[1] Die Schülerlösungen sind grundsätzlich ohne Korrektur wiedergegeben. Eine Interpretation der Schülerlösungen ist als Teil dieser Arbeit im Text eingearbeitet.

Die einzelnen Stunden der Examensreihe werden in den folgenden Kapiteln alle detailliert beschrieben. Zuvor werden in Kapitel 2.2 werden für die Examensreihe relevante Unterrichtsinhalte beschrieben, die die Schüler im Vorfeld erarbeitet haben. Am Ende des Berichts wird in Kapitel 2.7 noch beschrieben, was nach der Examensreihe behandelt wurde, um die Einbettung der Examensreihe in die Unterrichtseinheit zu erläutern.

2.2 Vor der Examensreihe

Die vor der Examensreihe behandelten Inhalte sind in Kapitel 1.1 beschrieben. Es sind Grundlagen von Java sowie Grundlagen der Graphentheorie. Dabei wurden, wie dort ebenfalls erwähnt, einige von mir nicht erwartete Defizite deutlich. Daher war es notwendig, Strategien zum Entwickeln von Algorithmen zu erarbeiten, was zum Arbeitsblatt „Entwickeln von Algorithmen“ (siehe Kapitel B.1 im Anhang) führte. Zum Vergleich sind im Anhang in Kapitel D.3 die entsprechenden Einträge im Kursheft abgedruckt. Leider fehlten dann in den Stunden direkt vor der Examensreihe einige Schüler, so dass nicht alle das Arbeitsblatt zu Beginn der Examensreihe schon bearbeitet hatten. Einige relevante Vorkenntnisse wurden gleich zu Beginn der Examensreihe durch einem Vortest überprüft.

2.3 1./2. Stunde (Doppelstunde): Einstieg und erste Begegnung mit Routing

Vortest (ca. 30 Minuten): Die Aufgabe 1 wurde von den meisten Schülern erfolgreich gelöst. Die drei relevanten Punkte, dass ein Graph aus Kanten und Knoten besteht, wobei Knoten irgendwelche Objekte repräsentieren und eine Kante je zwei Knoten miteinander verbindet, wurden korrekt benannt, wie zum Beispiel von Dennis[2] (vergleiche Kapitel C.1.1 im Anhang). Nur Sebastian hat gezeigt, dass bei ihm die Begriffe noch nicht gefestigt waren. Alexander hat die Aufgabe nicht allgemein sondern mit dem Beispiel des Königsberger Brückenproblems (vergleiche Kapitel 1.1) im Hinterkopf gelöst.

Die Aufgabe 2 hingegen ließ schon Leistungsunterschiede erkennen. Wie David (vergleiche Kapitel C.1.2 im Anhang) andeutete, könnte auch die Aufgabenstellung für Verwirrung gesorgt haben. Er schrieb: „In der 11. Klasse habe ich im Informatikunterricht auch nie gelernt etwas in Java zu modellieren“. Da dieses jedoch seit Beginn der zweiten Halbjahres der Jahrgangsstufe 12 wiederholt thematisiert wurde und David aktiv mitgearbeitet hatte, ist davon auszugehen, dass das Wort „modellieren“ ihn und eventuell noch andere verwirrt hat. Im Allgemeinen waren die Grundlagen vorhanden.

Mit der Aufgabe 3 (siehe Kapitel C.1.3 im Anhang) wurde nun das neue Thema eingeführt. Die Aufgabe 3a wurde erwartungsgemäß von allen korrekt gelöst. Interessant ist jedoch die Lösung

[2]Der Name der Schüler wurde geändert, um die Anonymität zu gewährleisten.

der Aufgabe 3b: Hier verwiesen viele Schüler darauf, die „einzelnen Wege“ entlang gegangen zu sein (zum Beispiel Katharina) beziehungsweise „alle Wege“ (Alexander, der dies auch noch farblich veranschaulicht hat) geprüft zu haben. Florian interpretierte sein Vorgehen kaum, zumal er nur von „Betrachten“ schrieb. Dennis zeigte als einziger eine funktionale Sichtweise, indem er ein sequenzielles Vorgehen beschrieb.

Die Anwendungen beschränkten sich auf geläufige Wegenetze, zum Beispiel von Fahrrädern, Autos, Bussen oder Flugzeugen. Nur Kevin nannte ein LAN als mögliche Anwendung.

Die Aufgabe 4 zeigte bei einigen Schülern Defizite auf. Die Antwort von Tobias (siehe Kapitel C.1.4 im Anhang) zum Beispiel machte dies deutlich. Auch wenn die im Arbeitsblatt im Anhang in Kapitel B.1 genannten Schritte nicht vollständig genannt werden mussten, war seine Antwort doch zu kurz. Es gab jedoch auch vollständige Lösungen wie zum Beispiel von Katharina. Hierbei ist jedoch hinzuzufügen, dass Sarah, David und Kevin in der entsprechenden Stunde nicht anwesend waren.

Der Test zeigt, dass die notwendigen Grundlagen größtenteils vorhanden waren. Die Grundbegriffe der Graphentheorie waren vorhanden, auch die beiden Implementationsmöglichkeiten, die die Schüler kennen sollten, waren bekannt. Schwieriger war es mit den Inhalten des Arbeitsblattes „Entwickeln von Algorithmen“.

Der Test war neben dem Überprüfen der Grundlagen auch der Einstieg in das Thema. Hier zeigten sich, wie zu erwarten, unterschiedliche Ansätze zur Lösung des Problems. Den Ansätzen ist gemeinsam, dass sie im Hintergrund „alle möglichen Wege“ haben. Dies war, wie später noch einmal explizit erwähnt, besonders bei Alexander deutlich. Zudem fällt auf, dass die Beschreibungen, wie der kürzeste Weg gefunden wurde, zumeist nicht direkt verwendet werden konnten, um einen Algorithmus zu entwickeln. Aus diesem Grund habe ich das Gruppenpuzzle der letzten Stunde um die Aufgabe 3 der Gruppe A erweitert.

Sicherung (ca. 5 Minuten): Da der kürzeste Weg von allen schon im Test gefunden wurde, war diese Sicherung sehr kurz. Sie diente somit vor allem als Anknüpfungspunkt für den folgenden Advance Organizer.

Advance Organizer (ca. 5 Minuten): Da das Problem den Schülern an dieser Stelle schon bekannt war, konnte das Vorhaben der nächsten Stunden ohne großen Aufwand vom Lehrer dargestellt werden. Ohne besondere Rückfragen konnte in die Erarbeitungsphase übergeleitet werden.

Erarbeitung und Sicherung I (ca. 25 Minuten): Die Schüler wurden so in Gruppen aufgeteilt, dass starke und schwache Schüler gemischt waren und bestimmte Gruppierungen, die erfahrungsgemäß ineffektiv sind, vermieden wurden. Zudem wurden Tobias und Katharina bewusst in eine Gruppe eingeteilt, um die Zusammenarbeit, die später (vergleiche Kapitel 2.5) wichtig sein würde, zu beobachten. Die Gruppen waren: Gruppe A: Sebastian, David und Dennis, Gruppe B: Marcel, Tobias und Katharina, Gruppe C: Florian, Alexander, Kevin und Sarah[3].

Die Ergebnisse der Gruppenarbeit sind im Anhang in Kapitel C.2 vollständig abgedruckt. Auch hier fällt auf, dass die Formulierung „alle möglichen Wege“ bei den Gruppen A und B an erster Stelle stand. Die Gruppe C erwähnte dies nicht. Sie beschrieb eher ein iteratives Vorgehen, das jedoch die notwendige Reaktivierung von Knoten, die auf einem kürzeren Weg als bisher erreicht worden sind, nicht erwähnte. Das verwendete Beispiel gab dieses jedoch auch nicht her. Dieses Resultat ist erstaunlich, da Alexander als Mitglied der Gruppe C sowohl im Test als auch in der Sicherungsphase nach der Gruppenarbeit betont hat, dass er „alle möglichen Wege“ abgehen wolle.

Besonders hinzuweisen ist auch auf die Länge der Gruppenergebnisse. Während die Gruppe A ihre Vorgehensweise ausführlich am Beispiel erklärte, formulierte die Gruppe B ein geradezu minimalistisches Ergebnis in einem einzigen Satz. Die Gruppe C ging einen dritten Weg, indem sie unabhängig vom Beispiel, aber ausführlicher als Gruppe B Handlungsanweisungen formulierte.

Diese Ergebnisse zeigen, dass sich alle Gruppen mit dem Thema auseinandergesetzt haben. Die Ergebnisse sind eine Synthese der verschiedenen Lösungsansätze.

Sicherung II (ca. 20 Minuten): Durch ein gegenseitiges Vorstellen der Ergebnisse wurde die Diskussion der Lösungsansätze fortgeführt. Es entstand eine angeregte Schüler-Schüler-Diskussion, an der sich alle Schüler beteiligten. Hierbei wurde unter anderem auch die Möglichkeit der späteren Verbesserung von zwischenzeitlich gewonnenen kürzesten Wegen erwähnt. Nach einem Impuls des Lehrers wurde diskutiert, ob denn wirklich alle Wege überprüft werden könnten, wie von allen vorgeschlagen. Als Zwischenergebnis wurde hier schon festgehalten, dass Zyklen zu vermeiden seien, da sie keine Vorteile brächten, dafür aber unendlich viele Wege überprüft werden müssten.

Hausaufgabe: Die Hausaufgabe (siehe Kapitel B.3 im Anhang) rundete die Stunde ab und diente gleichzeitig als Scharnier zur nächsten Stunde.

Zusammenfassend kann über die Doppelstunde gesagt werden, dass die Schüler das Problem erkannt haben. Jeder hatte individuelle Lösungsansätze, die sich in vielem mit den Lösungsansätzen der anderen überschnitten haben. Eine wichtige Fehlvorstellung, man könne doch einfach alle Wege betrachten, wurde hinterfragt. Viele Aspekte, die den Dijkstra-Algorithmus ausmachen, wurden erwähnt, wie zum Beispiel die iterative Vorgehensweise, die Unterscheidung zwischen aktiven und nicht-aktiven Knoten und die Verbesserung schon gegangener Wege.

[3] Die detaillierte Angabe der Teilnehmer der Gruppen ermöglicht eine Zuordnung von Schülern zu den Gruppenergebnissen.

2.4 3. Stunde: Der Dijkstra-Algorithmus

Einstieg (ca. 10 Minuten): Eine intensive Besprechung der Hausaufgabe ließ die Schüler darüber diskutieren, wie der dort dargestellte Algorithmus funktioniert. Eine Interpretation der Farben mit rot als aktiv und grün als erreicht fand statt, auch die Bedeutung der an den Knoten gespeicherten Zahlen wurde erfasst. Dies war jedoch nicht sofort für alle einsichtig, so dass Fehlinterpretationen in teilweise kontroversen Diskussionen geklärt werden mussten. Dies liegt unter anderem daran, dass das Beispiel trotz seiner in Kapitel 1.4.3 aufgezeigten Vorzüge nicht alle relevanten Aspekte des Dijkstra-Algorithmus darstellt.

Erarbeitung (ca. 15 Minuten): In denselben Gruppen wie in der vorhergehenden Stunde arbeiteten die Schüler den noch umgangssprachlich formulierten Algorithmus der Hausaufgabe in einen strukturierten Algorithmus um (vergleiche Kapitel C.3 im Anhang). Die Ergebnisse zeigten zwar eine deutliche Strukturierung, ließen aber noch keine Schleifen erkennen.

Sicherung (ca. 15 Minuten): Das Ergebnis der Gruppenarbeit, insbesondere das der Gruppe C, diente als Grundlage für das gemeinsame Erstellen von Pseudocode. Da die Schüler, wie in Kapitel 1.1 und 1.5.2 erwähnt, bisher keine Vorkenntnisse im Erstellen von Pseudocode hatten, war das konkrete Erstellen lehrergeleitet. Insbesondere die Formalisierung, zum Beispiel Schleifen durch Einrücken zu kennzeichnen, wurde vorgegeben. Das Ergebnis beinhaltete keinerlei Informationen über Datenstrukturen, da diese für den Dijkstra-Algorithmus nicht von großer Bedeutung sind.

Im Anhang in Kapitel C.4 ist das Tafelbild als Hefteintrag abgedruckt, wobei die Nummern „1.“ und „2.“ vom Schüler selber hinzugefügt wurden. Der Doppelpfeil „⇒“ stand fälschlicherweise an der Tafel, hier hätte „→“ stehen müssen, um Bedeutungsüberschneidungen zu mathematischen Notationen zu vermeiden. An dieser Stelle wurde deutlich, dass bisher noch nicht thematisiert wurde, wie denn der gefundene kürzeste Weg tatsächlich ausgegeben werden kann. Dieses Problem stellte sich erst bei der konkreten Implementierung in der nächsten Stunde.

Hausaufgabe (ca. 5 Minuten): Durch Anknüpfen an den Advance Organizer in der ersten Stunde (vergleiche Kapitel 1.5.1) und den momentanen Kenntnisstand der Schüler resultierte direkt die Formulierung der Hausaufgabe zur nächsten Stunde. Die Notwendigkeit für das Festlegen von Knoten und Kanten im Übersichtsplan der Schule war sofort einsichtig und auch das Vorbereiten der Implementierung durch Tobias und Katharina, indem sie sich in das vorgegebene Programmfragment einarbeiten sollten, war offensichtlich. Ein kurzes Beispiel auf Folie verdeutlichte die Art der Knotenfestlegung.

Zusammenfassend kann festgehalten werden, dass die Stunde, in all ihrem Anspruch von intensivem Arbeiten und fruchtbaren Diskussionen geprägt war.

2.5 4./5. Stunde (Doppelstunde): Vermessen der Schule und Implementierung

Einstieg (ca. 10 Minuten): Wie vorbereitet starteten Tobias und Katharina mit der Implementierung (siehe unten). Währenddessen klärten die anderen Schüler durch Zusammenlegen der Ergebnisse der Hausaufgabe ab, wie die Knoten denn nun konkret bezeichnet werden sollten. Hierbei wurde auch zugeteilt, wer Verbindungswege zwischen den vier Vermessungsbereichen jeweils messen sollte.

Vermessen der Schule (ca. 60 Minuten): Die erste Kante zwischen Knoten 16 und Knoten 19 wurde gemeinsam vermessen, um Details wie zum Beispiel den genauen Messort zu klären. Danach verteilten sich die Schüler eigenverantwortlich in der Schule und maßen ihre Kanten. Hierbei wurde deutlich, dass zum Beispiel die Gruppe von Marcel und David, die das zweite Obergeschoss vermessen sollte, schneller vorankam als die Gruppe von Dennis und Sarah, die den Außenbereich vermessen sollte. Hierbei ist auch festzuhalten, dass die Wahl der Kanten von Dennis und Sarah nicht optimal war. Dazu hätte man kleine Ungenauigkeiten in Kauf genommen und man nicht alle von ihnen angegebenen Kanten messen müssen (vergleiche Kapitel C.6 im Anhang).

Sichern der Ergebnisse (ca. 10 Minuten): Die ursprünglich als Hausaufgabe gedachte Sicherung durch das Eingeben der Ergebnisse in den Computer ließ sich noch in der Stunde durchführen. Die Schüler erstellten den im Anhang in Kapitel C.6 abgebildeten Überblick der festgelegten Knoten und Kanten. Zudem gaben sie die gemessenen Entfernungen mit den entsprechenden Knoten als Nachbarschaftsliste in den Computer ein. Dies hätte sonst zu Hause geschehen müssen, und zwar mit angemessener Zeit vor dem Beginn der nächsten Stunde, um eine Zusammenführung von Messdaten und Programm zu ermöglichen. Beim Eintragen der Knoten fiel den Schülern zudem auf, dass sie vergessen hatten, drei Entfernungen zu messen. Diese konnten sie direkt nachholen.

Implementierung des Dijkstra-Algorithmus (ca. 80 Minuten): Während der Vermessung der Schule arbeiteten Tobias und Katharina am Algorithmus[4]. Dies geschah in weiten Teilen selbstständig. Katharina hatte darüber hinaus schon einiges vorbereitet. Die beiden diskutierten über Lösungsansätze zur Umsetzung des Pseudocodes. Es fand eine echte Partnerarbeit statt, bei der sich beide mit ihrem Wissen und ihren Ideen ergänzen konnten. Sie benötigten jedoch einige Hilfestellungen. Diese waren teilweise technischer Art, es wurden aber auch konzeptionelle Tips für das Ermitteln des Rückweges benötigt. Das im Anhang in Kapitel C.5 abgedruckte Programm ist das, was von Tobias und Katharina entwickelt wurde, mit folgenden kleinen Änderungen:

[4]Ich habe diese beiden für diese Aufgabe ausgewählt, weil ich denke, dass sie für diese Aufgabe geeignet sind. Katharina zeichnet sich durch engagiertes und gewissenhaftes Arbeiten aus, Tobias kann durch einige Vorkenntnisse und Erfahrungen seine gut überlegten Ideen einbringen. Diese bewusste Einteilung ermöglichst somit eine innere Differenzierung.

- Der Name der Operation wurde von kurzesterWeg in berechneKuerzestenWeg geändert, um Verwechselungen mit dem Klassennamen zu vermeiden.
- Die Abfrage „**if** (kuerzesterWeg[i] < kuerzesterWeg[j])“ in Zeile 77 wurde durch die folgende korrekte Abfrage ersetzt: „**if** (kuerzesterWeg[i] + matrix[i][j] < kuerzesterWeg[j])“.
- Die Ausgaben in den Zeilen 104–114 wurde lesbarer gemacht.
- In die Zeilen 127–207 wurden die von den Schülern eingegebenen Messungen kopiert. Diese wurden auf die unten genannten Änderungen der Knotennamen reduziert.
- In den Zeilen 234–263 wurde eine dynamische Eingabe ermöglicht.
- Der Code wurde neu formatiert, damit er gefälliger lesbar ist.

Austausch (ca. 10 Minuten): Der Austausch über die Erfahrungen beim Messen und Implementieren gestaltete sich kurz. Da keine Besonderheiten vorkamen, gab es nur zu berichten, welche aufgekommenen Probleme zu lösen waren, zum Beispiel wie in der Implementierung der Rückweg gespeichert werden konnte.

Hausaufgabe: Da die geplante Hausaufgabe schon wie oben beschrieben in der Stunde von den Schülern bearbeitet werden konnte, und es für die folgende Stunde keine sinnvolle Aufgabe gab, wurde keine Hausaufgabe gestellt.

2.6 6. Stunde: Abschluss

Sicherung (ca. 10 Minuten): Durch die Unpünktlichkeit einiger Schüler konnte die Stunde nicht wie gewohnt ruhig begonnen werden. Trotzdem konnte der erarbeitete Algorithmus mit verschiedenen Wegen gemeinsam ausprobiert werden. Dabei beobachteten die Schüler einige Unzulänglichkeiten ihrer Lösung, insbesondere die Tatsache, dass der kürzeste Weg vom Knoten 29 (Lehrerzimmer) zum Knoten 12 (Musiksaal) durch mehrere Türen führt. Die Zeit, die man braucht, um durch sie hindurch zu gehen, rechtfertigt die (minimale) Zeitersparnis durch den kürzeren Weg nicht. Auch führten die Schüler weitere Aspekte an, die einen längeren Weg als besser darstellen lassen können, zum Beispiel die Tatsache, dass Wege ausserhalb des Schulgebäudes durch das Wetter suboptimal sein könnten. Andere Wege wie zum Beispiel vom Lehrerzimmer zum Sprachlabor wurden als optimal bezeichnet.

Gruppenpuzzle „Hintergrund und Transfer“ (ca. 30 Minuten): Das Gruppenpuzzle hingegen ließ eine intensive Arbeitsphase zu. Die Schüler kannten die Sozialform und konnten die Zeit nutzen, um im Sinne des Gruppenpuzzles sich gegenseitig die Lösungen der Aufgaben zu erklären. Da hier eine schriftliche Lösung der Aufgaben nicht vorgesehen war, ist eine Überprüfung der Ergebnisse schwer, ein Teilbereich wurde in der nachfolgenden Stunde in einem Test abgefragt. Das Gruppenpuzzle hat mehr Zeit beansprucht als veranschlagt, da die Schüler zwar intensiv, aber nicht so schnell wie erwartet gearbeitet haben. Daher wurde der Test in die folgende Stunde verschoben.

Umfrage (ca. 5 Minuten): Die Umfrage im Anhang in Kapitel B.9 wurde mit dem Kommentar ausgeteilt, dass die Ergebnisse anonym ausgewertet werden. Die Reaktionen der Schüler[5] zeigten, dass sie die Umfrage trotzdem ernst nehmen würden. Im Anhang in Kapitel C.8 sind die Ergebnisse als Mittelwert mit Standardabweichung angegeben. Dabei sind die üblichen Vorsichtsmaßnahmen zu bedenken. Das bedeutet unter anderem, dass die Ergebnisse untereinander nicht vergleichbar sind, dass also zum Beispiel eine Rangfolge der höchsten und niedrigsten Werte nicht erstellt werden sollte. Aussagekräftig sind insbesondere Aussagen, die eine geringe Standardabweichung haben, hier scheinen sich die Schüler in ihrer Einschätzung einig gewesen zu sein. Eine genaue Analyse der Umfrage wird in Kapitel 3.3 vorgenommen.

2.7 Nach der Examensreihe/Ausblick

Nach der Examensreihe wurden weitere Inhalte thematisiert, die mit ihr in Zusammenhang stehen. Ein Überblick lässt sich leicht anhand der Kurshefteinträge im Anhang in Kapitel D.3 gewinnen.

Nachtest (ca. 15 Minuten): Der erste Teil der Aufgabe 1 soll prüfen, inwieweit die Schüler die Funktionsweise des Dijkstra-Algorithmus wiedergeben können. Eine vollständige Wiedergabe des Algorithmus ist keinem Schüler gelungen[6]. Sebastian hat die beste Antwort gegeben (siehe Kapitel C.9 im Anhang), ihm fehlt nur der Hinweis darauf, dass kürzere Wege bisher gefundene Lösungen verbessern können. Durch kürzere Wege brauchen längere Wege nicht weiter verfolgt zu werden. Seine Beschreibung von „gierig“, dem zweiten Teil der Aufgabe 1, ist ebenfalls einleuchtend, auch wenn dies nicht der üblichen Definition von „gierig“ entspricht. Da das Wort für Schüler jedoch im Zusammenhang mit Algorithmen nicht geläufig ist, ist dies als sinnvolle Erklärung positiv zu werten. Die Lösung von Tobias hingegen kann als typisch angesehen werden. Seine Definition von „gierig“ kann als gut bezeichnet werden, sie trifft auch den Kern der üblichen Definition. Die Lösung von Florian interpretiere ich dahingehend, dass der Begriff „alle Wege“ differenzierter gesehen wird, auch wenn er nicht präzise von dem Begriff „allen möglichen Wegen“ abgrenzt, sondern das Argument alleine stehen lässt.

Die Aufgabe 2 überprüft einen Teil des Gruppenpuzzles: In der Aufgabe 1 der Gruppe B des Gruppenpuzzles (siehe Kapitel B.7 im Anhang) wurde gefordert, Kriterien für einen „sinnvollen“ Weg zu benennen. Da momentan bei im Internet weitergeleiteten Paketen nicht nach Herkunft unterschieden wird, ist ein sinnvoller Weg immer auch der kürzeste. Da die Schüler davon ausgehen können, dass der Graph zum Zeitpunkt der Messung vollständig bekannt ist, was auch die

[5] Sarah, die an dem Tag krank war, hat die Umfrage in der nächsten Stunde, in der sie anwesend war, ausgefüllt.
[6] Am Tag des Nachtests waren zwei Schüler krank, Dennis und Sarah.

Karte von 1974 nahelegt, ist vor allem die Dauer ein Kriterium. Der Datendurchsatz spielt ebenso eine Rolle, da eine Überlastung einer Leitung zusätzliche Latenzen und somit einen längeren Weg erzeugt. Sebastian beschreibt diesen Zusammenhang richtig. Katharina benennt auch diese beiden Kriterien, obwohl sie den Datendurchsatz fälschlicherweise mit „Datenvolumen“ bezeichnet, das sie konsequent in Byte messen möchte und nicht in Byte/Sekunde. Kevin bestreitet, dass das Datenvolumen relevant ist, er fordert hingegen stabile Leitungen. In der Gesamtheit werden viele relevante Kriterien genannt, teilweise aber auch zusätzlich irrelevante Kriterien.

Stellwand „Routing in der Schule“: Da die Schüler während der 4./5. Stunde beim Vermessen von vielen Kollegen auf ihre Arbeit angesprochen wurden und ich von mehreren Seiten gebeten wurde, die Ergebnisse der Unterrichtsreihe zu präsentieren, habe ich mich entschlossen, dass die Schüler Ergebnis an einer Stellwand der Schulöffentlichkeit präsentieren sollen. Diese Stellwand, die als Foto im Anhang in Kapitel D.1 zu sehen ist, haben die Schüler selber konzipiert und erstellt. Sie wurde in der Schule ausgestellt und am Tag der offenen Tür von den Schülern zusammen mit dem Programm präsentiert (siehe unten). Sie enthält ein Fragment des erstellten Programmcodes, die Festlegung der Knoten und Kanten sowie erläuternde Texte. Das Erarbeiten der Stellwand stellte auch einen guten Abschluss dar.

Brückenbauen: Im Anschluss an die Osterferien bearbeiteten die Schüler das Material von Zürcher (2006). Dort wird eine weiter optimierte Variante des Dijkstra-Algorithmus verwendet. Dies schärfte das Verständnis des Dijkstra-Algorithmus.

Stationenlernen „Klassische Probleme der Graphentheorie“: Weitere klassische Probleme der Graphentheorie (Problem eines Handlungsreisenden, Briefträgerproblem, Hamiltonkreise und das Vier-Farben-Problem) erweiterten den Bereich der Anwendungen von Graphen. Die von mir zusammengestellten und als Stationenlernen aufbereiteten Materialien sind im Anhang in Kapitel D.2 abgedruckt. Ergänzt werden sie durch eine Station zum Thema Routing in Datennetzen. Ein weiterer Fokus liegt auf der Einschätzung der Komplexität von Algorithmen.

Präsentation am Tag der offenen Tür: Der Kollege, der für den „Tag der offenen Tür“ für die zukünftigen Schüler der Jahrgangsstufe 5 verantwortlich war, beobachtete die Schüler auch bei ihren Messungen. Er fragte an, ob die Ergebnisse nicht dort präsentiert werden könnten, um einen Einblick in das Fach Informatik zu ermöglichen. Dieser Bitte bin ich gefolgt, indem die Schüler zusätzlich zur Stellwand mit Computer und Beamer das erarbeitete Programm demonstrierten und erklärten. Obwohl Inhalte der Jahrgangsstufe 12 nicht primär für zukünftige Schüler der Jahrgangsstufe 5 interessant sind, entwickelten sich doch interessante Gespräche. Neben den obligatorischen Fragen nach dem Weg zu einem bestimmten Ort, die die Schüler bereitwillig und ausführlich beantworten konnten, ging es dabei auch um Art und Umfang des Informatikunterrichts. Erstaunen rief manchmal die Aussage hervor, dass die Schüler keine Informatik-AG mit besonders begabten Schülern seien, sondern ein normaler Kurs der Jahrgangsstufe 12.

3 Kritische Auswertung

In diesem Kapitel werde ich die in Kapitel 2 berichtete Durchführung der Examensreihe analysieren und mit den Ansprüchen der Planung aus Kapitel 1 in Verbindung setzen. Um dies zu erleichtern habe ich am Anfang und am Ende der Examensreihe je einen Test schreiben lassen, um Vorkenntnisse und Ertrag zumindest teilweise messbar zu machen. Eine anonyme Umfrage, in der die Schüler offen ihre Meinung sagen können, rundet die Analyse ab.

In diesem Kapitel werden unter anderem einzelne Aspekte der Ergebnisse der Umfrage (siehe Kapitel B.9 im Anhang), die im Anhang in Kapitel C.8 vollständig dargestellt sind, kommentiert. Dazu muss darauf hinwiesen werden, dass zumindest prinzipiell die einzelnen Ergebnisse aufgrund unterschiedlicher Maßstäbe nicht miteinander verglichen werden können, da die Maßstäbe unterschiedlich sind[1]. Als Anhaltspunkt kann man jedoch eine zufällige Verteilung nehmen: Der Durchschnitt wäre hierbei 0,00 und die Standardabweichung 2,16.

Als erste Beobachtung lässt sich sagen, dass die Schüler den Unterricht durchweg positiv erlebt haben, da die Aussagen alle positiv gestellt sind und alle bis auf eine Aussage überdurchschnittlich bewertet werden. Dies wird auch in der Bewertung der letzten Aussage, „Insgesamt bin ich mit unserem Unterricht zufrieden“ deutlich: Mit einem Durchschnitt von 1,80 und bei einer Standardabweichung von 0,79 ist sich die Lerngruppe recht einig (vergleiche Kapitel C.8 im Anhang).

3.1 Erreichen der Lernziele

Die schon im Titel aufgeworfene Frage, ob es möglich ist, Routingalgorithmen in einem regulären Informatikkurs der Jahrgangsstufe 12 mit grundlegendem Anforderungsniveau erfolgreich zu unterrichten (vergleiche Einleitung auf S. III und Kapitel 1.3.2), muss als zentraler Bestandteil dieser Examensarbeit an vorderster Stelle kritisch betrachtet werden. Hierzu ist der Lernzuwachs zu betrachten und zu bewerten.

3.1.1 Erreichen der inhaltlichen Kompetenzen

In Kapitel 1.6 habe ich Lernziele für die Unterrichtsreihe formuliert. Das Erreichen der einzelnen Lernziele werde ich nun sukzessive betrachten und nach Gründe dafür suchen, warum sie erreicht

[1] Vergleiche hierzu die Beobachtungen von Schusser (1996, S. 26 ff.), der sich intensiv mit der Frage der Vergleichbarkeit von Schulnoten auseinandersetzt. Die Ergebnisse sind auf die Umfrage übertragbar.

beziehungsweise nicht erreicht worden sind. Hierzu analysiere ich neben den Berichten in Kapitel 2 insbesondere den Vortest (siehe Kapitel B.2 im Anhang) und den Nachtest (siehe Kapitel B.8 im Anhang), deren Ergebnisse ich schon in den Kapiteln 2.3 und 2.7 beschrieben habe.

Die Schüler sollen Graphen zur Modellierung von Wegenetzen anwenden können: Dieses Lernziel beinhaltet zwei Aspekte. Zunächst ist die Verwendung eines Graphen als Modell für ein Wegenetz den Schülern am Ende der Unterrichtsreihe selbstverständlich erschienen. Das sieht man daran, dass sie in der Aufgabe 1 des Nachtests den vorgegebenen (und aus dem Vortest bekannten) Graphen problemlos lesen und verstehen konnten. Dies kann man sogar schon für die Aufgabe 3a des Vortests festhalten. Der selbstverständliche Umgang mit dem Graphen als Modell für das Wegenetz der Schule in der 4./5. Stunde (vergleiche Kapitel 2.5) lässt dies auch zweifelsfrei deutlich werden. An dieser Stelle bleibt jedoch auch festzuhalten, dass dies keine besonders anspruchsvolle Leistung ist. In Zeiten von Navigationssystemen (vergleiche Kapitel 1.2) und mit minimalen Vorkenntnissen über Graphen ist es kein abwegiges Modell. Zugleich beinhaltet dieses Lernziel aber auch einen kreativen Umgang mit Graphen zur Modellierung von Wegenetzen. Hier lässt sich als Beleg für das Erreichen dieses Lernzieles die Bearbeitung der Hausaufgabe von der 3. Stunde auf die 4./5. Stunde (vergleiche zusätzlich Kapitel 2.4) heranziehen. Wie dort beschrieben gab es keine prinzipiellen Verständnisschwierigkeiten. Neben kleinen Absprachproblemen, die auf fehlende Gewissenhaftigkeit der Schüler zurückzuführen sind, ist nur das Maß an Genauigkeit des Modells an einigen Stellen uneinheitlich gewesen. Dies wurde jedoch gleich zu Beginn der 4. Stunde homogenisiert. Die Schüler konnten wie in Kapitel 2.6auch Unzulänglichkeiten ihres Modells erkennen.

Insgesamt kann dieses Lernziel also als vollständig erreicht angesehen werden.

Die Schüler sollen Graphen zur Modellierung von Datennetzen anwenden können: Ähnlich wie bei Wegenetzen verhält es sich mit Datennetzen, wobei Datennetze im Vergleich zu Wegenetzen einen Schritt abstrakter sind. Zudem standen Datennetze nicht im Zentrum der Unterrichtsreihe. Jedoch klang auch diese Anwendungsmöglichkeit von Graphen immer wieder an, insbesondere im Gruppenpuzzle (siehe Kapitel B.7 im Anhang) in der 6. Stunde. Wie in Kapitel 2.6 erläutert war eine vollständige Ergebnisüberprüfung des Gruppenpuzzles nicht geplant. Im Nachtest wird genau diese Modellierung für eine erfolgreiche Bearbeitung der Aufgabe 2 vorausgesetzt. Die Schülerlösungen, die zwar nicht ganz korrekt sind, aber viel Richtiges beinhalten, zeigen indirekt, dass auch diese Modellierung verstanden wird. Ein weitergehendes Modellieren durch Angabe möglicher Kriterien für die „Länge“ eines Weges ist für viele Schüler möglich (vergleiche Kapitel 2.7). Hier lässt sich auch im Vergleich zur Aufgabe 3c des Vortests ein deutlicher Lernzuwachs sehen, in dem die meisten Schüler Datennetze nicht als einen möglichen Anwendungsbereich von Graphen gesehen haben (vergleiche Kapitel 2.3).

Auch für dieses Lernziel kann festgehalten werden, dass es erreicht wurde.

Bei der Modellierung bietet sich eine mögliche Erweiterung an. Gallenbacher (2006, S. 4) fragt explizit, welche Informationen denn nun relevant für kürzeste Wege sind. Dieses betont noch stärker den Modellierungsprozess, der ein wesentlicher Bestandteil der Informatik ist[2].

Die Schüler sollen ein Maß für Wege angeben können: Dieses Lernziel ist bei Wegen wie zum Beispiel in der Schule trivial. Für Graphen, die jedoch keine Wegstrecken angeben, für die gar die Dreiecksungleichung nicht gilt, ist dies nicht immer einfach. Genau diese Kompetenz zu überprüfen war Ziel der Aufgabe 2 des Nachtests. Wie in Kapitel 2.7 beschrieben ist dies zwar vielen Schülern, aber eben nicht allen möglich. Dies mag zunächst erstaunlich klingen, da in der Gruppe B des Gruppenpuzzles in der 6. Stunde in der Aufgabe 1 verlangt wurde, *Kriterien für einen sinnvollen Weg* anzugeben. Ein Hindernis könnte aber sein, dass in der Aufgabe 2 des Nachtests nun nach *sinnvollen Kriterien für den kürzesten Weg* gefragt wurde, womit die Aufgabe das Anforderungsniveau II hat.

Für dieses Lernziel muss also festgehalten werden, dass es nicht in vollem Umfang erreicht worden ist: Einige Schüler sind nicht immer in der Lage, in einem eigentlich bekannten Anwendungsbereich ein geeignetes Maß anzugeben. Dies kann mehrere Ursachen haben. Zunächst kann es sein, dass der Anwendungsbereich doch nicht so bekannt ist, wie angenommen, und dass die Sachkenntnis fehlt um sinnvolle Kriterien anzugeben. Die Lösung von Marcel (siehe Kapitel C.9.2 im Anhang) legt dieses nahe, da er von einer festen Zeit zwischen zwei Knoten ausgeht. Dann kann es sein, dass die Vorstellung „kürzester Weg“ automatisch mit „Länge in Metern“ verknüpft ist. Dies ist jedoch in keiner Schülerlösung zu erkennen. Zuletzt könnte noch sein, dass die Schüler nicht verstanden haben, dass für einen kürzesten Weg allein die Länge der Wege zwischen je zwei Knoten relevant ist, so denn ein Weg existiert. Wenn dies so wäre, müsste man davon ausgehen, dass die Schüler das Wesen des Dijkstra-Algorithmus nicht verstanden haben. Eine solche Lösung existiert jedoch auch nicht. Es bleibt also die Interpretation, dass die Sachkenntnis über das Internet fehlt, dass also die Ergebnisse des Gruppenpuzzles nicht detailliert genug waren, um bei allen die Grundlage für die Beantwortung dieser Frage zu legen.

Die Schüler sollen den Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges in einem ungerichteten, kantengewichteten Graphen beschreiben können: Dieses Lernziel wurde explizit in der Aufgabe 1 des Nachtests abgefragt. Wie in Kapitel 2.7 beschrieben, ist die korrekte Beantwortung nicht bei allen in vollem Umfang gelungen.

Dies könnte entweder daran liegen, dass den Schülern der Dijkstra-Algorithmus nicht so geläufig ist, dass sie ihn vollständig wiedergeben können, oder dass sie es nicht für nötig empfunden haben, ihn in allen Details zu beschreiben. Wie im Anhang in Kapitel C.9.2 an den beispielhaften Lösungen der Schüler zu erkennen ist, sind die fehlenden Details oftmals durch vage Formulierungen verdeckt. So schreibt Sebastian zum Beispiel: „Von diesen neuen Knoten [...] findet eine weitere Überprüfung statt“. Was genau er mit „Überprüfen“ meint, schreibt er nicht. Ähnlich

[2] „Der Vorgang der Abstraktion, bei dem die reale Welt auf ein Modell reduziert wird, das die wesentlichen Bestandteile hervorhebt, ist für die Informatik von zentraler Bedeutung“ (Hartmann et al., 2006, S. 115).

Florian, der schreibt: „Der Algorithmus überprüft alle von A aus erreichbaren Knoten und hält den bis dahin kürzesten Weg fest. Dies wird fortgesetzt, bis alle von A aus erreichbaren Knoten mindestens einmal aktiv waren“, wie aber Knoten aktiv werden, schreibt er nicht.

Zudem bleibt zu beobachten, dass immer noch Formulierungen wie „alle Möglichkeiten“ (zum Beispiel bei Florian) zu lesen sind. Dies ist ein Hinweis darauf, dass eventuell ein wesentliches Detail des Dijkstra-Algorithmus, nämlich dass nur kürzere Wege berücksichtigt werden, nicht von allen verstanden wurde[3]. Dennoch gehe ich davon aus, dass das Prinzip des Dijkstra-Algorithmus von den meisten Schülern verstanden worden ist. Den Hauptgrund für eine verkürzte Darstellung sehe ich darin, dass der zur Verfügung stehende Platz sehr gering war: In zehn Zeilen sollten die Schüler den Dijkstra-Algorithmus beschreiben und zusätzlich die Eigenschaft „gierig“ erläutern.

Dass einige Schüler immer noch in Betracht ziehen, alle möglichen Wege zu gehen, erscheint angesichts der Diskussion in der 1./2. Stunde (vergleiche Kapitel 2.3) zunächst erstaunlich. Wie in Kapitel 2.3 aber ebenfalls festgehalten, wurde diese Frage dort nicht abschließend thematisiert. Das liegt unter anderem daran, dass den Schülern die Voraussetzungen für eine richtige Beurteilung fehlten (vergleiche Kapitel 1.1).

Die Schüler sollen einen umgangssprachlich formulierten Algorithmus schrittweise formalisieren können: Dieses Lernziel beschreibt einen zentralen Prozess der Examensarbeit. Entlang des Arbeitsblattes „Entwickeln von Algorithmen“ (siehe Kapitel B.1 im Anhang) haben die Schüler ihre umgangssprachlich formulierten Ideen präzisiert und formalisiert.

Hier ist als Ausgangspunkt zunächst die Gruppenarbeit der 1./2. Stunde zu sehen, deren Ergebnisse im Anhang in Kapitel C.2 festgehalten sind. Sie stellt nach ersten Gedanken der Schüler in der Aufgabe 3b (vergleiche Kapitel C.1.3 im Anhang sowie die Interpretation in Kapitel 2.3) eine erste Einigung der Schüler auf einen gemeinsamen Algorithmus dar. Die Ergebnisse der drei Gruppen sind umgangssprachlich formuliert.

Die Ergebnisse der Gruppenarbeit in der 3. Stunde (vergleiche Kapitel C.3 im Anhang) haben hingegen schon eine deutlich erkennbare Struktur. Abläufe sind erkennbar, wenn auch Kontrollstrukturen noch nicht benannt worden sind. Damit wird deutlich, dass das Material, das die Schüler in der Hausaufgabe zur 3. Stunde bearbeiten sollten (siehe Kapitel B.3 im Anhang), auch eine Strukturierung der Vorstellung zur Folge hatte.

Im darauf folgenden Unterrichtsgespräch wurden diese Ergebnisse gemeinsam in Pseudocode umgewandelt. Die starke Lenkung durch den Lehrer ist, wie in Kapitel 2.4 dargestellt, durch die fehlenden Vorkenntnissen in Pseudocode (vergleiche Kapitel 1.1) bedingt[4].

---

[3] Dieser Einwand scheint jedoch auf hohem Niveau. Hartmann et al. (2006, S. 118) schreiben über den Dijkstra-Algorithmus: „Die Grundidee hinter dem Algorithmus: Um den kürzesten Weg vom Startknoten S zum Zielknoten Z zu bestimmen, berechnet man *alle Wege* und wählt den kürzesten aus“ (Hervorhebung durch den Verfasser dieser Arbeit).

[4] Hartmann et al. (2006, S. 120) schreiben hierzu: „Die Beschreibung des Algorithmus in Pseudocode hat zudem den großen Nachteil, dass zuerst eine formale Notation eingeführt werden muss. Die Studierenden müssen erst die Notation verstehen, bevor sie sich auf die Kernidee hinter dem Algorithmus konzentrieren können“. Das trifft den Kern des Problems „auf den Kopf“. Zudem schreiben sie aber auch: „Einleuchtender als ein Programm

Insgesamt ist aber dennoch deutlich geworden, dass die Schüler mit Hilfe des Arbeitsblattes aus Kapitel B.1 im Anhang einen umgangssprachlich formulierten Algorithmus formalisieren können.

Die Schüler sollen die Umsetzung dieses Algorithmus in die Programmiersprache Java nachvollziehen können: Wie in Kapitel 1.5.2 begründet haben nur zwei Schüler (Tobias und Katharina) den Dijkstra-Algorithmus implementiert. Daher ist es für die anderen Schüler wichtig, die Umsetzung des Algorithmus nachvollziehen zu können. Die Gruppe C des Gruppenpuzzles in der 6. Stunde sollte dies leisten. Da Tobias und Katharina in die Expertengruppe der Gruppe C eingeteilt wurden, konnten sie ihr Erarbeitetes weitergeben. Wie in Kapitel 2.6 allerdings schon dargestellt, konnte das Ergebnis des Gruppenpuzzles – und somit des Lernziels – nicht direkt überprüft werden. Da dieses Lernziel auch nicht im Nachtest überprüft wurde, wäre jede bewertende Aussage über das Erreichen des Lernziels Spekulation. Dadurch wird deutlich, dass eine Verschriftlichung der Ergebnisse des Gruppenpuzzles eine Bewertung verbessert hätten.

Die Schüler sollen Algorithmus-Ideen verständlich formulieren können: In der Examensreihe wird das Reden über Informatik an vielen Stellen ausdrücklich von den Schülern verlangt. Sowohl in den Gruppenarbeiten als auch im Plenum sind die Schüler aufgefordert, ihre Ideen und Vorstellungen über Algorithmen so zu formulieren, dass ihre Mitschüler sie verstehen können. Ein Lernzuwachs ist hier deutlich zu erkennen. Während zum Beispiel Florian im Vortest in der Aufgabe 3c noch schreibt, er habe den kürzesten Weg „durch simples Betrachten“ (siehe Kapitel C.1.3 im Anhang) gefunden, sind die ebenso umgangssprachlich geforderten Beschreibungen der danach folgenden Gruppenarbeit schon präziser (vergleiche Kapitel C.2 im Anhang). Auch die Lösungen der Aufgabe 1 des Nachtests (siehe Kapitel C.9.1 im Anhang) sind neben ihrer inhaltlichen Verbesserung verständlicher. Damit ist auch diese Kompetenz stärker ausgeprägt.

3.1.2 Erreichen der sozialen Kompetenzen

Neben den inhaltlichen Lernzielen sollte auch das Erreichen der sozialen Lernziele reflektiert werden.

Die Schüler sollen verschiedene Lösungsansätze durch Kommunikation zu einem gemeinsamen Lösungsansatz umarbeiten können: Aufbauend auf die Kommunikationsfähigkeit zum Austausch über ihre Vorstellungen sollen die Schüler sich auf einen gemeinsamen Algorithmus einigen. Dies bildet den Kern der Gruppenarbeiten in der 1./2. und 3. Stunde sowie des Unterrichtsgesprächs zur Formulierung des Pseudocodes. Auch hier lässt sich feststellen, dass die Schüler diese Kompetenz erreicht haben: Die schriftlichen Lösungen der Gruppenarbeit (siehe Kapitel C.2 und C.3 im Anhang) sowie die Diskussion bei der Formulierung des Pseudocodes belegen dies.

---

wäre eine Beschreibung des Ablaufs durch eine Bilderfolge [. . .], zusammen mit einer textuellen Beschreibung“. Die dadurch angedeutete parallele Repräsentation (vergleiche auch Kapitel 1.3.1) eines Algorithmus auf verschiedene Art und Weise wurde auch von mir entsprechend umgesetzt.

Die Schüler sollen arbeitsteilige Aufträge selbstständig planen und durchführen können: Dieses übergeordnete Lernziel möchte ich hauptsächlich anhand der 4./5. Stunde evaluieren. Was eine selbständige Planung angeht, ist dieses Lernziel nicht voll erreicht, insbesondere weil ich zu viele Vorgaben gemacht habe. Die grobe Struktur war schon vorgegeben, so dass die Schüler keine großen Planungen machen mussten. Erst in der Planung der konkreten Umsetzung konnten die Schüler aktiv werden. Dies liegt hauptsächlich daran, dass ich den Zeitaufwand für das Vermessen der Schule nicht gut einschätzen konnte und daher eine stärkere Lenkung gewählt habe. Bei der Umsetzung durch die Schüler zeigte sich, dass sie eigenverantwortlicher hätten arbeiten können. Beim Kontrollgang während der 4./5. Stunde zeigte sich auch, dass die Schüler die gestellte Aufgabe gewissenhaft und systematisch bearbeiteten. Dieses wurde auch bei der Bearbeitung des Gruppenpuzzles in der 6. Stunde deutlich: Innerhalb des vorgegebenen Zeitrahmens arbeiteten die Schüler zumeist effektiv und erreichten einen deutlichen Lernzuwachs (wenn auch keinen optimalen, vergleiche hierzu die Auswertung der inhaltlichen Lernziele in Kapitel 3.1.1).

Die Schüler sollen Ergebnisse ihrer Arbeit präsentieren können: Dieses Lernziel steht in engem Zusammenhang mit den anderen sozialen Lernzielen. Es stellt eine Voraussetzung für gelungene Kommunikation über Inhalte der Informatik dar. An den verschiedenen Stellen, wo Schüler Ergebnisse präsentieren sollten, zeigte sich, dass dies den Schülern keine Mühe bereitete. Insbesondere in den Stammgruppen des Gruppenpuzzles der 6. Stunde war zu beobachten, dass die Vorträge unaufgeregt und sachlich abliefen. Somit ist bei den Schülern auch diese Kompetenz entwickelt.

Auch wenn die Präsentation am „Tag der offenen Tür“ nicht Teil der Examensreihe ist, sollte auch hier festgehalten werden, dass die Schüler sich und ihre Arbeit dort gut präsentiert haben.

3.1.3 Lernzuwachs aus Sicht der Schüler

In der Umfrage sind einige Fragen zu finden, die die Schüler ihren Lernzuwachs einschätzen lassen. Dabei möchte ich zunächst mit der Aussage „Der Lehrer hat ein hohes Maß an Fachwissen“ anfangen, die die Schüler mit einem extrem hohen Mittelwert von 2,50 (Standardabweichung 0,71) bewertet haben. Das an sich ist für einen Lernzuwachs zunächst nicht relevant, jedoch ist diese Einschätzung der Schüler notwendig, um sich bei Fragen zuversichtlich an den Lehrer wenden zu können.

Die Aussage „Ich lerne viel im Unterricht“ hingegen lässt die Schüler selber einschätzen, ob sie viel lernen oder nicht. Der Wert von 1,00 bei einer Standardabweichung von 1,15 zeigt eine verhaltene Zustimmung der Schüler. Wenn die Schüler sie hätten umformulieren müssen, hätten sie eventuell geschrieben: „Ich lerne einiges im Unterricht“. Da die Schüler dies explizit auf die Stunden der Examensreihe beziehen sollten, könnte man aus Sicht der Schüler sagen, dass die Zeit hätte effektiver genutzt werden können. Ich gehe jedoch davon aus, dass die Schüler „viel“ im Sinne von „viele verschiedene Dinge“ verstehen und nicht im Sinne von „eine Sache besonders intensiv“. Zudem sind Lernziele, die sich nicht direkt aus dem Stoff ableiten, für Schüler oftmals nicht transparent[5].

„Die Unterrichtsinhalte sind aktuell“ wurde ähnlich wie die vorige Aussage bewertet: Der Mittelwert ist 0,90 bei einer Standardabweichung von 0,99. Gründe hierfür sind darin zu finden, dass trotz aktuellen Bezügen (zum Beispiel in Routenplanern, vergleiche Kapitel 1.2) keine aktuelle Diskussion darüber stattfindet. Es wird anders als andere Themen der Informatik zum Beispiel nicht in Zeitungen diskutiert. Dass diese Aussage jedoch nicht komplett verneint wird, ist vermutlich darauf zurückzuführen, dass die im Unterricht verwendeten Beispiele einen Alltagsbezug für die Schüler haben.

Dieser Alltagsbezug ist auch ein Grund dafür, dass die Aussage „Im Unterricht werden Theorie und Praxis angemessen miteinander verknüpft“ mit 1,50 durchaus Zustimmung erfährt, auch wenn die Standardabweichung von 1,27 nicht gerade gering ist. Aus Sicht der Schüler geschieht dies aber noch längst nicht so stark, wie sie es sich wünschen würden. Hierbei muss gefragt werden, was denn aus Schülersicht Theorie und Praxis ist. Ein Indiz dafür bietet eine freie Schüleräußerung in der Umfrage, in der der Schüler das Erlernen der Sprache Java als Theorie bezeichnet (vergleiche Kapitel C.8 im Anhang). Nach Leuders (2001, S. 100) hingegen ist aber auch die Aufgabe der Vermessung der Schule nicht authentisch, da es unglaubwürdig scheint, dass jemand die Schule so wie die Schüler vermessen würde, um dort kürzeste Wege zu finden. Aus dieser Sicht ist verständlich, dass der Aussage nicht in vollem Umfang zugestimmt wird. Die dennoch erfolgte Zustimmung mag daher kommen, dass ein Realitäts- und Anwendungsbezug vorhanden ist und dass das Erarbeitete stellvertretend für komplexere aber dafür auch authentischere Zusammenhänge steht.

Alleinig wird der Aussage „Die Unterrichtsinhalte haben Bezüge zu anderen Fächern“ nicht zugestimmt. Auch wenn der Mittelwert mit −0,40 (Die Standardabweichung ist 1,35) sehr knapp nur von einer Zustimmung entfernt ist, sollte dies betont werden. In Bezug auf die Examensreihe stimmt dies auch. Es wurden keine expliziten Querverweise getätigt und keine Anleihen aus anderen Fächern getätigt. Dies wäre zwar möglich, würde aber nicht dem Hauptanliegen dienen.

Insgesamt bleibt festzuhalten, dass die Schüler während der Examensreihe aus ihrer Sicht einiges an Kompetenzen und Kenntnissen gewonnen haben. Ein teilweise aktuelles Thema und eine merkbar angemessene Verknüpfung von Theorie und Praxis haben trotz teilweise mangelnder Bezüge zu anderen Fächern dazu geführt, dass die Schüler mit dem Unterricht zufrieden waren.

3.1.4 Konstruktivismus

In Kapitel 1.4.1 habe ich einen pragmatischen Konstruktivismus gefordert. Inwieweit dies umgesetzt worden ist, soll in diesem Kapitel evaluiert werden.

[5] Dies gilt nicht zuletzt wegen der Aussage „Mir ist klar, welche Ziele im Unterricht erreicht werden sollen“, der mit 1,70 zwar durchaus, aber auch nicht in vollem Umfang zugestimmt wird.

Zunächst soll hier die Einschätzung der Schüler wiedergegeben werden. Drei Aussagen der Umfrage bewerten die Umsetzung des Konstruktivismus. An vorderster Stelle steht die Aussage „Ich muss keine Angst haben, im Unterricht etwas falsch zu machen“. Dies ist gleichfalls Grundvoraussetzung und wichtigstes Element des Konstruktivismus. Deshalb ist der sehr hohe Mittelwert von 2,40 bei einer geringen Standardabweichung von 0,52 eine starke Unterstützung der Forderung.

Den Aussagen „Der Unterricht in meiner Klasse wird abwechslungsreich gestaltet“ sowie „Im Unterricht werden Theorie und Praxis angemessen miteinander verknüpft“ wird mit Mittelwerten von 1,20 sowie 1,50 jeweils zugestimmt, auch wenn die Standardabweichungen mit 1,55 und 1,27 nicht besonders gering sind. Wenn diese beiden Aussagen gelten, wird das Lernen im konstruktivistischen Sinne gefördert, da eine Konfrontation mit bisherigen Vorstellungen durch verschiedene Anregungen die Schärfung des Verstehens bewirkt. Eine Vernetzung mit schon vorhandenen Vorstellungen wie zum Beispiel durch Anwendungsbeispiele aus der Lebenswelt der Schüler ist so ebenso möglich. Die Zustimmung lässt auch hier auf eine richtige Umsetzung der Forderung des Konstruktivismus schließen.

Eine Umsetzung des Konstruktivismus gilt aus meiner Sicht nicht in vollem Umfang. Dafür hätten Fehlvorstellungen bewusster aufgenommen werden müssen, zum Beispiel durch Aufgaben, die die Schüler direkt mit typischen Fehlvorstellungen konfrontieren. So wurden Fehlvorstellungen nur implizit in Gruppenarbeiten und Unterrichtsgesprächen aufgenommen. Dies wird am Beispiel der Fehlvorstellung deutlich, dass man einfach alle möglichen Wege ausprobieren kann. Diese Fehlvorstellung hat dazu geführt, dass das Lernziel „Die Schüler sollen den Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges in einem ungerichteten, kantengewichteten Graphen beschreiben können“ in genau diesem Punkt nicht voll erreicht wurde (vergleiche Kapitel 3.1.1).

Zusammenfassend kann trotzdem davon gesprochen werden, dass zentrale Momente des Konstruktivismus erfüllt wurden.

3.2 Umsetzung der methodischen Planungen

Ich habe in Kapitel 1.4 mit entdeckendem Lernen und Selbstorganisiertem Lernen starke methodische Erwartungen geweckt. In welchem Maße diese umgesetzt wurden, muss kritisch hinterfragt werden. In diesem Zusammenhang ist unter anderem auch das Lehrer-Schüler-Verhältnis relevant, das in Kapitel 3.3 besonders betrachtet wird.

3.2.1 Offenheit des Unterrichts

In Kapitel 1.4.1 habe ich ein gewisses Maß an Offenheit des Unterrichts gefordert. Hartmann et al. (2006, S. 91 ff.) beschreibeny Offenheit und ihre Konsequenzen wie folgt: „Die Lernenden erlangen neues Wissen, indem sie persönliche Erfahrungen machen und Dinge hinterfragen. Sie entwickeln durch Staunen, sich Wundern und Zweifeln ihre eigenen (vieleicht naiven) Theorien und müssen dabei mitunter alte Vorstellungen und zuvor aufgestellte Hypothesen verwerfen“

(Hartmann et al., 2006, S. 92). Diesen zentralen Punkt ich versucht umzusetzen, indem ich die Schüler zunächst eigene Ideen habe entwickeln lassen. Bei der Konfrontation mit dem Dijkstra-Algorithmus musste zum Beispiel die Idee, man könne doch einfach alle Wege abgehen, verworfen werden. Dies war unter anderem möglich, weil sich die Schüler trauten, Fehler zu machen (vergleiche die Ergebnisse der Umfrage im Anhang in Kapitel C.8). Hartmann et al. (2006, S. 92) schreiben dazu sogar: „Es gibt a priori kein Richtig und Falsch auf der Suche nach Neuem“. Dem stimme ich zwar in Bezug auf von Hentig (1998) nicht vollständig zu, sage aber dennoch, dass es nicht schlimm ist, zunächst Wege beschreiten zu wollen, die nicht zum Ziel führen.

Auch wenn es also möglich gewesen ist, „ein Thema explorativ zu erkunden, Hypothesen aufzustellen, zu überprüfen und mit anderen Schülern auszutauschen“ (Hartmann et al., 2006, S. 92), so ist der Unterricht nicht vollständig offen gewesen: Der Lerngegenstand ist, wie in Kapitel 1.4.1 dargestellt, sehr konkret und nicht unbedingt „mehrschichtig oder vielfältig, verschiedene Aspekte umfassend und verschiedene Entdeckungswege zulassend“ (Hartmann et al., 2006, S. 92).

Nach Herold und Landherr (2003, S. 11) gilt dies ebenso. Dort wird gefordert: „Die Schüler werden nicht mehr durch segmentierte Gestaltung des Unterrichts zum Ziel geführt. Vielmehr werden sie durch offene Aufgabenstellungen zum selbsttätigen Handeln aufgefordert“. Die Hausaufgabe zur 3. Stunde (siehe Kapitel B.3 im Anhang) zum Beispiel aber stellt hier eine eindeutige Segmentierung dar, auch das Vorgehen nach dem Arbeitsblatt „Enwickeln von Algorithmen“ (siehe Kapitel B.1 im Anhang) ist eine Vorgabe, die dieser Forderung diametral entgegensteht.

Die Schüler scheinen dies anders zu sehen. Der Aussage „Im Unterricht wird mir Gelegenheit zu eigenständigem Lernen und Arbeiten gegeben“ stimmen sie mit einem Mittelwert von 1,90 und einer Standardabweichung von 0,99 klar zu. Da sie aber auch den Aussagen „Der Unterricht ist so gestaltet, dass ich ihm in der Regel gut folgen kann“, „Mir ist klar, welche Ziele im Unterricht erreicht werden sollen“ sowie „Im Unterricht ist eine klare Struktur erkennbar“ mit 1,90, 1,70 und 1,89 zustimmen (Standardabweichungen 0,57, 1,06 und 0,78), muss diese Bewertung interpretiert werden.

Zunächst ist die Aussage vorsichtig formuliert. Die Formulierung „Mir wird Gelegenheit gegeben“ ist um vieles schwächer als zum Beispiel die Formulierung „Ich muss häufig“. Zudem muss bedacht werden, dass sich Schüler in einer rezeptiven Rolle durchaus einrichten können und von daher zu viel Offenheit und damit verbunden mehr Selbstverantwortung und Aktivität scheuen könnten. Daher interpretiere ich die Zustimmung zu der Aussage so, als ob dort stünde: „Ich als Schüler akzeptiere das Maß an Offenheit und Struktur, das ich im Unterricht erlebt habe. Es bietet mir die Möglichkeit, selber zu denken und zu handeln. Ich bin aber auch drüber froh, dass klare Strukturen existieren, an denen ich mich orientieren kann“. Basis für einen Überblick über die Struktur des Unterrichts ist unter anderem der Advance Organizer, der im Sinne von Herold und Landherr (2003) am Anfang der Examensreihe einen Ausblick auf das Vorhaben bot. Dies verdeutlicht das Wechselspiel zwischen Offenheit und Lenkung.

Zusammenfassend ist zu sagen, dass der Unterricht in den Binnenstrukturen offen gestaltet ist, jedoch im großen Zusammenhang der Forderung nach Offenheit nicht standhält. Dies liegt hauptsächlich an der didaktischen Entscheidungf, als Lerngegenstand den Dijkstra-Algorithmus zu behandeln, weil dadurch viel vorgegeben werden musste.

3.2.2 Die äußere Organisation des Unterrichts

Neben den inhaltlichen Fragestellungen an den Unterricht stellt sich auch die Frage, inwiefern der Ablauf des Unterrichts gut geplant und durchgeführt ist. Es bleibt zunächst einmal festzuhalten, dass die in Kapitel 1 vorgestellte Planung im Großen und Ganzen so umgesetzt werden konnte, wie sie gedacht war. Im Detail gab es jedoch auch einige Schwierigkeiten. Während die beiden Doppelstunden (1./2. Stunde sowie 4./5. Stunde) von der Zeitplanung jeweils gut umsetzbar waren, war die 3. Stunde sehr gefüllt und die 6. Stunde so voll, dass der eigentlich geplante Test in die Folgestunde der Examensreihe verschoben werden musste.

Für die 3. Stunde bleibt festzuhalten, dass das Vorhaben, sowohl in Gruppenarbeit eine strukturierte Version des Dijkstra-Algorithmus als auch danach dasselbe in Pseudocode zu erstellen (vergleiche Kapitel 1.5.2), sehr ambitioniert war. Aufgrund der Tatsache, dass die Schüler bisher keinen großen Kontakt mit Pseudocode hatten, war dies nur mit einer verstärkten Lenkung durch den Lehrer möglich.

In der 6. Stunde hätte die Durchführung des geplanten Gruppenpuzzles optimiert werden können. Wäre in jeder Gruppe vorgesehen gewesen, einen Zeitwächter zu bestimmen, der genau auf die Einhaltung vorher festgelegter Zeiten achtet, hätte die Zeit effizienter genutzt werden können, auch wenn ich hier festhalten möchte, dass die Zeit ohne den Zeitwächter dennoch effektiv genutzt wurde, um in dem einen oder anderen Punkt mehr ins Detail gehen zu können. Aber auch diese Vorgehensweise hätte möglicherweise nicht die für den Test fehlende Zeit gebracht. Ich hätte die Stunde durchaus vorentlasten können, indem ich die erste Phase nur im Gruppenpuzzle hätte stattfinden lassen.

Eine durchaus mögliche Ursache hingegen trifft nicht zu. Die Schüler bestätigen die Aussage „Mein Lehrer erscheint pünktlich zum Unterricht“ schon fast zu deutlich mit einem Mittelwert von 2,90 bei einer geringen Standardabweichung von 0,32. Hingegen wird der Aussage „Im Unterricht wird kaum Zeit für Nebensächlichkeiten verschwendet“ mit einem Mittelwert von 0,90 (Standardabweichung 1,37) nur mäßige Zustimmung gegeben. Was für Nebensächlichkeiten könnten dies sein? Zunächst kann man natürlich festhalten, dass das, was Schüler für nebensächlich hielten, nicht unbedingt für den Unterricht nebensächlich war. Eventuell fühlen sich die Schüler auch unter- oder überfordert, da sie der Aussage „Mein Lehrer stellt angemessen hohe Leistungsanforderungen an die Klasse“ bei einem Mittelwert von 1,33 und einer hohen Standardabweichung von 1,32 nur mäßig zustimmen. Auch dies könnte zu einer solchen Beobachtung führen. Hier besteht also tatsächlich Optimierungsbedarf.

Auch wenn ich es nicht so extrem wie die Schüler einschätze, so möchte ich mich ihnen doch anschließen, wenn sie der Aussage „Mein Lehrer ist immer gut auf den Unterricht vorbereitet“ mit einem Mittelwert von 2,70 bei einer Standardabweichung von 0,48 deutlich zustimmen. Ebenso stimmen sie der Aussage „Im Unterricht wird der PC sinnvoll eingesetzt“ mit einem Mittelwert von 2,00 (Standardabweichung 0,67) zu.

Zur äußeren Organisation von Unterricht zähle ich auch die Bewertung der Schüler. Da der Aussage „Die Bewertungsmaßstäbe zur Beurteilung meiner Leistungen sind mir bekannt und nachvollziehbar“ bei einem Mittelwert von 0,50 und der sehr hohen Standardabweichung von 1,78 nur schwach zugestimmt wird, drängt sich die Frage auf, woran das liegt. Hinzu kommt, dass die Schüler mit ihren Beurteilungen bisher zufrieden gewesen sind. Es ist also kein Ausdruck dafür, dass sie sich schlecht bewertet fühlen. Dass sie dennoch die Bewertungsmaßstäbe nicht nachvollziehen können, könnte aber daran liegen, dass das bewusste Zulassen von Fehlern (vergleiche Kapitel 3.1.4) einen scheinbar bewertungsfreien Raum schafft, aber dennoch aufgrund der mündlichen Leistungen im Unterricht eine Note erteilt wird. Hier sehe ich Bedarf, mit den Schülern zu reden und diesen scheinbaren Widerspruch zu klären.

Die letzte Aussage, deren Bewertung ich in diesem Kapitel interpretieren möchte, sollte nicht zu stark gewichtet werden. „Mein Lehrer ist an meiner schulischen/beruflichen Weiterentwicklung interessiert“ ist eine Frage, die ihren Ursprung hauptsächlich darin hat, dass die Umfragen von Schudak (2007a) und Schudak (2007b) an berufsbildenden Schulen durchgeführt wurden. Dennoch ist natürlich auch ein Gymnasium an der Entwicklung seiner Schüler interessiert. In diesem Zusammenhang kann eine mittlere Zustimmung von 1,33 bei einer Standardabweichung von 1,51 als angemessen angesehen werden.

3.3 Das Lehrer-Schüler-Verhältnis

Ein gutes Verhältnis zwischen Lehrer und Schülern ist grundlegend für erfolgreichen Unterricht. Eine Atmosphäre, die von Freundlichkeit und gegenseitigem Respekt geprägt ist, erlaubt ein entspanntes sachliches Arbeiten. Um die Interpretationen hierzu nicht im Ungefähren zu lassen, habe ich in einer anonymen Umfrage die Schüler ihre Meinung kundtun lassen.

Dabei beziehen sich sechs Fragen direkt auf das Lehrer-Schüler-Verhältnis. Die beiden Aussagen „Ich respektiere meinen Lehrer“ und „Ich fühle mich von meinem Lehrer ernst genommen“ werden durchweg positiv bewertet. Man kann also begründet sagen, dass ich in meiner Rolle akzeptiert werde. Die Schüler fühlen ihre Anliegen offenbar angemessen aufgenommen.

In dem Zusammenhang passt auch die Aussage „Ich muss keine Angst haben, im Unterricht etwas falsch zu machen“, die mit einem hohen Durchschnittswert von 2,40 und einer sehr geringen Standardabweichung von 0,52 bewertet wurde. Die Bewertungen sind alle entweder eine +2 oder eine +3. Diese Aussage finde ich besonders hervorhebenswert, nicht nur weil sie einen offenen Umgang mit Fehlvorstellungen – einem Grundprinzip des Konstruktivismus (vergleiche Kapitel 3.1.4) – ermöglicht, sondern auch, weil sie Menschlichkeit ermöglicht. Ein Klima der Angst, eventuell gar im Zusammenhang mit dem Empfinden einer ungerechten Benotung, hemmt Schüler, sich mit den Inhalten des Unterrichts auseinanderzusetzen. Daher ist es besonders gut, dass eben diese Ängste nicht existieren.

Leider scheint dies keine großen Auswirkungen auf die Bewertung der Aussage „Mein Lehrer trägt dazu bei, dass ich gerne lerne“ zu haben. Mit einem Mittelwert von 0,89 ist sie zwar noch positiv, jedoch keinesfalls deutlich. Laut der Standardabweichung von 1,69 sind sich auch die Schüler durchaus nicht einig. Ein genauerer Blick auf die Zahlen (einmal -2, einmal -1, einmal 0, dreimal 1, einmal 2 und zweimal 3 sowie eine Enthaltung) zeigt, dass nur drei Schüler der Aussage eher nicht zustimmen. Die große Standardabweichung hingegen ist hauptsächlich darauf zurückzuführen, dass zwei Schüler der Aussage voll und ganz zustimmen.

Gründe für diese auffällige Uneinigkeit sind möglicherweise darin zu suchen, wie ich persönlich auf die Schüler wirke. Da einer meiner zentralen pädagogischen Überzeugungen ist, dass Schüler eigenständig lernen sollen, stehen Maßnahmen, die die extrinsische Motivation fördern, nicht im Zentrum des Unterrichts[6]. Angesichts der Ergebnisse sollte ich mir ernsthaft überlegen, ob nicht gerade in Kursen mit grundlegendem Anforderungsniveau[7] der Aspekt der Motivation verstärkt werden sollte. Eine starke positive Korrelation zwischen Motivation und Leistung ist hinlänglich bekannt[8] und sollte von daher von mir ernster genommen werden. Ich denke, dies hätte vor allem Vorteile für Schüler, die nicht sowieso schon intrinsisch motiviert sind[9].

Der Aussage „Mein Lehrer reagiert angemessen auf Unterrichtsstörungen“ wurde mit 1,10 nur mäßig zugestimmt. Die Uneinigkeit – zumal die Standardabweichung 1,45 beträgt – und die Tatsache, dass der Unterricht aus meiner Sicht im Grunde störungsfrei verläuft, lässt mich darauf schließen, dass eine Unsicherheit aufgrund mangelnder Beispiele existiert, so dass die Schüler also nicht wirklich einschätzen können, wie ich auf Unterrichtsstörungen reagieren würde. Ebenso könnte es auch sein, dass die Schüler etwas anderes unter Unterrichtsstörungen verstehen als ich und von daher anders urteilen als ich es würde.

Die letzte Aussage, die ich in diesem Zusammenhang betrachten möchte, lässt die Schüler das Festgestellte gut zusammenfassen: Die Aussage „Mein Lehrer fördert eine freundliche Atmosphäre in der Klasse“ bewerteten die Schüler bei einem Durchschnitt von 2,00 und einer geringen Standardabweichung von 0,67. Man kann also konstatieren, dass in der Lerngruppe ein positives Lernklima herrscht, in dem die Schüler sich wohl fühlen. Da die Aussage auf den Lehrer direkt bezogen ist, fühle ich mich auch darin bestätigt, dass meine Bemühungen, dieses Klima zu erhalten und zu pflegen, Früchte tragen[10].

[6] Die intrinsische Motivation wird hauptsächlich durch die Auswahl des Gegenstandes und den Lebensweltbezug durch ein geeignetes Beispiel gefördert. Vergleiche hierzu die Überlegungen in Kapitel 1.3.2.

[7] Florian sprach an einer Stelle sogar mal von einem „Abdeckerkurs“.

[8] „Leichter, anregender Stress, Belohnungserwartung und Konzentration fördernde Atmosphäre sind förderliche Bedingung für die Verankerung von Erfahrungen im Langzeitgedächtnis“ (Leuders, 2003, S. 44).

[9] Leider kann ich keinen Zusammenhang zwischen intrinsischer beziehungsweise extrinsischer Motivation und „gerne lernen“ nachweisen, da mir dazu die entsprechenden Fragen in der Umfrage fehlen. Zumindest ein relevanter Zusammenhang, intrinsische Motivation $\Rightarrow$ „gerne lernen“, dürfte jedoch außer Frage stehen.

[10] Gerne zitiere ich an dieser Stelle Dennis, der gesagt hat: „Muss man denn bei der Umfrage auch etwas negativ beurteilen?“

3.4 Fazit

Die Analyse der Examensreihe lässt mich zu dem Schluss kommen, dass in den durchgeführten sechs Stunden das Hauptanliegen (vergleiche Kapitel 1.3.2) umgesetzt wurde. Es ist also möglich in einem regulären Informatikkurs der Jahrgangsstufe 12 mit grundlegendem Anforderungsniveau Routing zu vermitteln. Die Schüler haben einen klaren und angemessenen Lernzuwachs erreicht. Die beschriebenen Einschränkungen sind aus meiner Sicht hauptsächlich darauf zurückzuführen, dass einige Voraussetzungen gefehlt haben, insbesondere die fehlende Erfahrung im Umgang mit Pseudocode sowie mit Komplexität. Dies hat zu Fehlvorstellungen geführt, die teilweise schwer zu korrigieren waren, wie zum Beispiel die Vorstellung, man könne einfach alle möglichen Wege ausprobieren. Dies ist aus meiner Sicht jedoch kein großes Problem. Auch wenn dies schon bei der Planung hätte berücksichtigt werden müssen, hat es den Erfolg nicht verhindert.

Damit kann die in der Einleitung auf Seite III gestellte Frage positiv beantwortet werden: Es ist möglich, in einem Informatikkurs der Jahrgangsstufe 12 Routingalgorithmen erfolgreich zu unterrichten.

Dennoch werde ich im folgenden Ausblick Konsequenzen darstellen, die ich im Anschluss an die Examensreihe gezogen habe, sowie weitere Alternativen aufzeigen, die bei einer erneuten Durchführung berücksichtigt werden sollten.

3.5 Ausblick

3.5.1 Konsequenzen aus der Examensreihe

Die Beobachtung, dass der Dijkstra-Algorithmus nicht vollständig von allen Schülern verstanden worden ist (vergleiche mit den Abwägungen in Kapitel 3.1.1), hat dazu geführt, dass ich die Schüler das Material von Zürcher (2006) habe bearbeiten lassen. Der spielerische und interaktive Umgang ließ die Schüler verschiedene Test-Szenarien aufbauen und so den Dijkstra-Algorithmus besser verstehen, auch in Abgrenzung der Details der beiden Varianten.

Im danach folgenden Stationenlernen zu klassischen Problemen der Graphentheorie (siehe Kapitel D.2 im Anhang) habe ich zudem eine Station hinzugenommen, die Routing im Internet thematisiert. Hier wird insbesondere thematisiert, in welchen Bereichen der Dijkstra-Algorithmus sinnvoll eingesetzt werden kann, und wo es Bereiche gibt, in denen er nicht eingesetzt werden kann.

Den fehlenden Vorstellungen über Komplexität von Algorithmen bin ich vor allem in den Stationen 1, 2 und 3 begegnet, in denen zum Einen für konkrete Algorithmen Angaben zur Komplexität gemacht werden sollen, und wo zum Anderen typische Klassen von Funktionen in Bezug auf ihr Verhalten für große Funktionsargumente verglichen werden sollen. Dies zusammen ermöglicht eine qualifizierte Beurteilung von Algorithmen.

3.5.2 Alternativen

Die fehlenden Voraussetzungen in Bezug auf Pseudocode und Komplexitätsvorstellung sollten bei einem erneuten Durchlauf vorher erarbeitet werden[11]. Damit könnte in der 3. Stunde die Lehrerlenkung (vergleiche Kapitel 2.4) stark reduziert werden. Ein besseres Beispiel als das der Hausaufgabe (siehe Kapitel B.3 im Anhang), eventuell sogar ein animiertes Beispiel, könnte den Dijkstra-Algorithmus einsichtiger machen.

Es gibt mehrere Möglichkeiten, die Unterrichtsreihe zu eröffnen. Im Vortest wurde gleich vorgegeben, dass Knoten Kreuzungen darstellen und Kanten Wege sind, die ein bestimmtes Gewicht haben. Dies könnte zugunsten einer echten Modellierung weggelassen werden. Überhaupt kann der Aspekt der Modellierung nach Hartmann et al. (2006, S. 115) wie bei Gallenbacher (2006, S. 4) stärker betont werden.

Wie in Kapitel 1.2 schon dargestellt verstehe ich unter „Nachbarschaftsliste“ etwas anderes, als üblicherweise darunter verstanden wird. Da Nachbarschaftslisten eine geläufige Übersetzung für Adjazenzlisten ist, würde ich eine Nachbarschaftsliste demnächst anders benennen, beispielsweise „Kantenliste“. Diese Datenstruktur halte ich aber weiterhin für sinnvoll, da sie im Gegensatz zur Nachbarschaftsmatrix und dem, was normalerweise unter Nachbarschaftslisten verstanden wird, anschaulich und praktisch verwendbar ist.

Zuletzt würde ich das Gruppenpuzzle der 6. Stunde etwas erweitern, um Elemente der Station 5 des Stationenlernens mit aufzunehmen. Dadurch würde das Gruppenpuzzle länger, es würde wohl eine ganze Stunde ausfüllen. Zeitwächter in jeder Gruppe würden eine Umsetzung in diesem Zeitrahmen ermöglichen.

Die genannten Alternativen hätten den Erfolg der Examensreihe vergrößert. Trotzdem erscheint mir unter Berücksichtigung der Kriterien diese Unterrichtsreihe als ein lohnenswertes und erfolgreiches Unternehmen.

[11] Hartmann et al. (2006, S. 120) machen dies sehr deutlich.

A Literaturverzeichnis

BELLMAN, RICHARD (1958): *On a Routing Problem. Quarterly of Applied Mathematics*, 16(1), S. 87–90.

BIBLIOGRAPHISCHES INSTITUT IN LEIPZIG (1880-1898): *Karte von Königsberg.*
URL: `http://www.hicleones.com/callmap-e.php?tekst=10071&map=Koenigsberg%2FK%F6nigsberg%20-%20City%20%2F%20Ville`. [Online; Stand 22. Mai 2007].

BRITT, MATT (2007): *Internet map 4096.*
URL: `http://en.wikipedia.org/wiki/Image:Internet_map_4096.png`. [Online; Stand 22. Mai 2007].

BRUNER, JEROME SEYMOUR (1960): *The Process of Education.* Harvard University Press, Cambridge.

DALGETY, JAMES (2007): *Sir William Hamilton's Icosian Game and Traveller's Dodecahedron Puzzle.*
URL: `http://puzzlemuseum.com/month/picm02/200207icosian.htm`. [Online; Stand 22. Mai 2007].

DIESTEL, REINHARD (2006): *Graphentheorie.* Springer, Heidelberg.
URL: `http://www.math.uni-hamburg.de/home/diestel/books/graphentheorie/`. [Online; Stand 22. Mai 2007].

DIJKSTRA, EDSGER W. (1959): *A Note on Two Problems in Connexion with Graphs. Numerische Mathematik*, 1, S. 269–271.
URL: `http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf`. [Online; Stand 22. Mai 2007].

DODGE, MARTIN (1974): *An Atlas of Cyberspaces – Historical Maps: Logical network map of ARPANET.*
URL: `http://www.cybergeography.org/atlas/historical.html`. [Online; Stand 22. Mai 2007].

FACHKONFERENZ INFORMATIK GYMNASIUM CAROLINUM OSNABRÜCK (2007): *Curriculum in Informatik, Gymnasium Carolinum Osnabrück.*

FLOYD, ROBERT W. (1962): *Algorithm 97 (SHORTEST PATH). Communications of the ACM*, 5(6), S. 345.

GALLENBACHER, JENS (2006): *Abenteuer Informatik – IT zum Anfassen von Routenplaner bis Online-Banking.* Spektrum Akademischer Verlag, Heidelberg.

GUDJONS, HERBERT (2003): *Didakdik zum Anfassen – Lehrer/in-Persönlichkeit und lebendiger Unterricht.* Klinkhardt, Bad Heilbrunn.

HART, PETER E., NILSSON, NILS J. und RAPHAEL, BERTRAM (1968): *A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics*, SSC4 (2), S. 100–107.

HARTMANN, WERNER, NÄF, MICHAEL und REICHERT, RAIMOND (2006): *Informatikunterricht planen und durchführen.* Springer, Berlin.

von Hentig, Hartmut (1998): *Kreativität – Hohe Erwartungen an einen schwachen Begriff.* Carl Hanser, München.

Herold, Martin und Landherr, Birgit (2003): *SOL - Selbstorganisiertes Lernen – Ein systemischer Ansatz für Unterricht.* Schneider, Hohengehren.

Herzberg, Joachim (2005): *Wissensbasierte Robotik – Vorlesung SS 2005.* Universität Osnabrück (unveröffentlicht).

Humbert, Ludger (2005): *Didaktik der Informatik – mit praxiserprobtem Unterrichtsmaterial.* Teubner, Wiesbaden.

Kultusministerkonferenz (2004): *Einheitliche Prüfungsanforderungen in der Abiturprüfung – Informatik, Beschluss vom 1.12.1989 i.d.F. vom 5.2.2004.* Beschlüsse der Kultusministerkonferenz. Luchterhand, München.

Leuders, Timo (2001): *Qualität im Mathematikunterricht der Sekundarstufe I und II.* Cornelsen, Berlin.

Leuders, Timo (2003): *Mathematik-Didaktik – Praxishandbuch für die Sekundarstufe I und II.* Cornelsen, Berlin.

Lutz-Westphal, Brigitte (2004): *Wie komme ich optimal ans Ziel? Unterricht über Kürzeste-Wege-Algorithmen für Graphen.*
URL: `http://www.zib.de/Publications/Reports/ZR-04-35.pdf`. [Online; Stand 22. Mai 2007].

Meyer, Hilbert (1987a): *UnterrichtsMethoden – I: Theorieband.* Cornelsen, Berlin.

Meyer, Hilbert (1987b): *UnterrichtsMethoden – II: Praxisband.* Cornelsen, Berlin.

Müller, Matthias M. und Padberg, Frank (2007): *Extreme Programming from an Engineering Economics Viewpoint.*
URL: `http://www.ipd.uka.de/Tichy/uploads/publikationen/41/edser02.pdf`. [Online; Stand 22. Mai 2007].

Niedersächsisches Kultusministerium (1993): *Rahmenrichtlinien für das Gymnasium – gymnasiale Oberstufe, die Gesamtschule – gymnasiale Oberstufe, das Fachgymnasium, das Abendgymnasium, das Kolleg – Informatik.* Schroedel, Hannover.

Niedersächsisches Kultusministerium (2007): *Themen der Schriftliche Abiturprüfung 2010 – Informatik.* (Bisher unveröffentlicht).

Reed, Isaac (1994): *Famous Problems in the History of Mathmatics.*
URL: `http://mathforum.org/isaac/mathhist.html`. [Online; Stand 22. Mai 2007].

Röthlisberger, Urs und Wittmann, Armin (1994): *Routing Algorithmen(Puzzle).*
URL: `http://www.educeth.ch/lehrpersonen/informatik/unterrichtsmaterialien_inf/kommuniation_kryptographie/routing_algorithmus/index`. [Online; Stand 22. Mai 2007].

Sanders, Peter, Singler, Johannes und Kupferer, Stephan (2006): *7. Algorithmus der Woche – Kürzeste Wege – Wie komme ich am schnellsten von einem Ort zum anderen?*
URL: `http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo7.php`. [Online; Stand 22. Mai 2007].

Schubert, Sigrid und Schwill, Andreas (2004): *Didaktik der Informatik.* Spektrum Akademischer Verlag, Heidelberg.

Schudak, Axel (2007a): *Schulumfragen Demo.*
URL: `http://www.schulumfragen.de/demo/`. [Online; Stand 22. Mai 2007].

SCHUDAK, AXEL (2007b): *Schülerumfrage 2007.*
URL: http://www.schulumfragen.de/till/schulumfrage2007/. [Online; Stand 22. Mai 2007].

SCHUSSER, GERHARD (1996): *Leistungsbeurteilung in der Schule – Zur Notwendigkeit veränderter Leistungsbeurteilungspraxis als konstitutives Moment überfälliger Schulreformmaßnahmen.* Staperfeld, Osnabrück.

SJUTS, JOHANN (1999): *Mathematik als Werkzeug zur Wissensrepräsentation – Theoretische Einordnung, konzeptionelle Abgrenzung und interpretative Auswertung eines kognitions- und konstruktivismustheoriegeleiteten Mathematikunterrichts*, Band 35 von *Schriftenreihe des Forschungsinstituts für Mathematikdidaktik.* Forschungsinstitut für Mathematikdidaktik, Osnabrück.

STAMM, TOBIAS (2007): *Dijkstra-Algorithmus.*
URL: http://mandalex.manderby.com/d/dijkstra.php?id=93. [Online; Stand 22. Mai 2007].

VORNBERGER, OLIVER (1997): *Graphenalgorithmen.*
URL: http://www-lehre.inf.uos.de/~graph/skript/skript.html. [Online; Stand 22. Mai 2007].

WARSHALL, STEPHEN (1962): *A Theorem on Boolean Matrices. Journal of the ACM*, 9(1), S. 11–12.

WIKIPEDIA (2007a): *A*-Algorithmus.*
URL: http://de.wikipedia.org/wiki/A*-Algorithmus. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007b): *Algorithmus von Floyd und Warshall.*
URL: http://de.wikipedia.org/wiki/Algorithmus_von_Floyd_und_Warshall. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007c): *Bellman-Ford-Algorithmus.*
URL: http://de.wikipedia.org/wiki/Bellman-Ford-Algorithmus. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007d): *Dijkstra-Algorithmus.*
URL: http://de.wikipedia.org/wiki/Dijkstra-Algorithmus. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007e): *Edsger Wybe Dijkstra.*
URL: http://de.wikipedia.org/wiki/Edsger_Wybe_Dijkstra. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007f): *Problem eines Handlungsreisenden.*
URL: http://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007g): *Routing.*
URL: http://de.wikipedia.org/wiki/Routing. [Online; Stand 22. Mai 2007].

WIKIPEDIA (2007h): *Vier-Farben-Satz.*
URL: http://de.wikipedia.org/wiki/Vier-Farben-Satz. [Online; Stand 22. Mai 2007].

ZÜRCHER, SAMUEL (2006): *Graphenalgorithmen: Brückenbauen (Entdeckendes Lernen).*
URL: http://www.educeth.ch/lehrpersonen/informatik/unterrichtsmaterialien_inf/algorithmen_datenstrukturen/brueckenbauen/index. [Online; Stand 22. Mai 2007].

B Verwendete Materialien

In diesem Kapitel sind in chronologischer Reihenfolge alle Materialien aufgeführt, die den Schülern ausgeteilt wurden. Die Arbeitsblätter sind zur besseren Lesbarkeit auf die Hälfte der Fläche verkleinert worden.

B.1 Material „Entwickeln von Algorithmen“

Entwickeln von Algorithmen

Mit diesem Arbeitsblatt lernst Du Möglichkeiten kennen, wie man komplexere Algorithmen planen, umsetzen und programmieren kann. Die einzelnen Schritte müssen nicht jedes Mal ausgeführt werden – je schwerer ein Algorithmus ist, desto mehr lohnt es sich, kleinschrittig vorzugehen.

1. Beschreibe die Aufgabenstellung mit eigenen Worten (Ziel abklären).

2. Beschreibe umgangssprachlich, wie Du die Aufgabe lösen kannst. Stell Dir vor, Du würdest die Lösung einem Mitschüler erklären (Lösung finden).

3. Formuliere Deine umgangssprachliche Lösung in eine Ablaufbeschreibung um. Du darfst dabei die Arbeitsweise eines Computers im Hinterkopf haben (Lösung strukturieren).

4. Formuliere Deine Ablaufbeschreibung um, indem Du sie in Schritte umwandelst, die auch im Computer in einem Schritt verarbeitet werden können (Pseudocode).

5. Übersetze die Schritte des Pseudocode in Java-Code.

B.2 Vortest

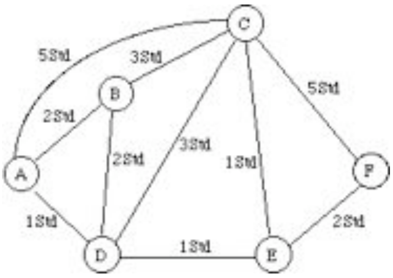
Kurs if20 **Kurztest** 9. März 2007

Name: ___________________________

Aufgabe 1. Definiere die Begriffe „Graph“, „Knoten“ und „Kante“.

Aufgabe 2. Benenne und beschreibe zwei Arten, wie man einen Graphen in Java modellieren kann. Modelliere das Haus des Nikolaus in einer der beiden Arten.

Aufgabe 3. Der folgende Graph beschreibt wie lange ein Zug braucht, um von einem Ort zu einem anderen zu kommen.



a) Gib den kürzesten Weg von A nach C an.

b) Beschreibe, wie Du den kürzesten Weg gefunden hast.

c) Benenne mindestens zwei andere Anwendungen für solche Graphen.

Aufgabe 4. Du sollst einen komplizierten Algorithmus entwickeln. Stelle sinnvolle Schritte dar, die Dir helfen können, den Algorithmus zu erarbeiten.

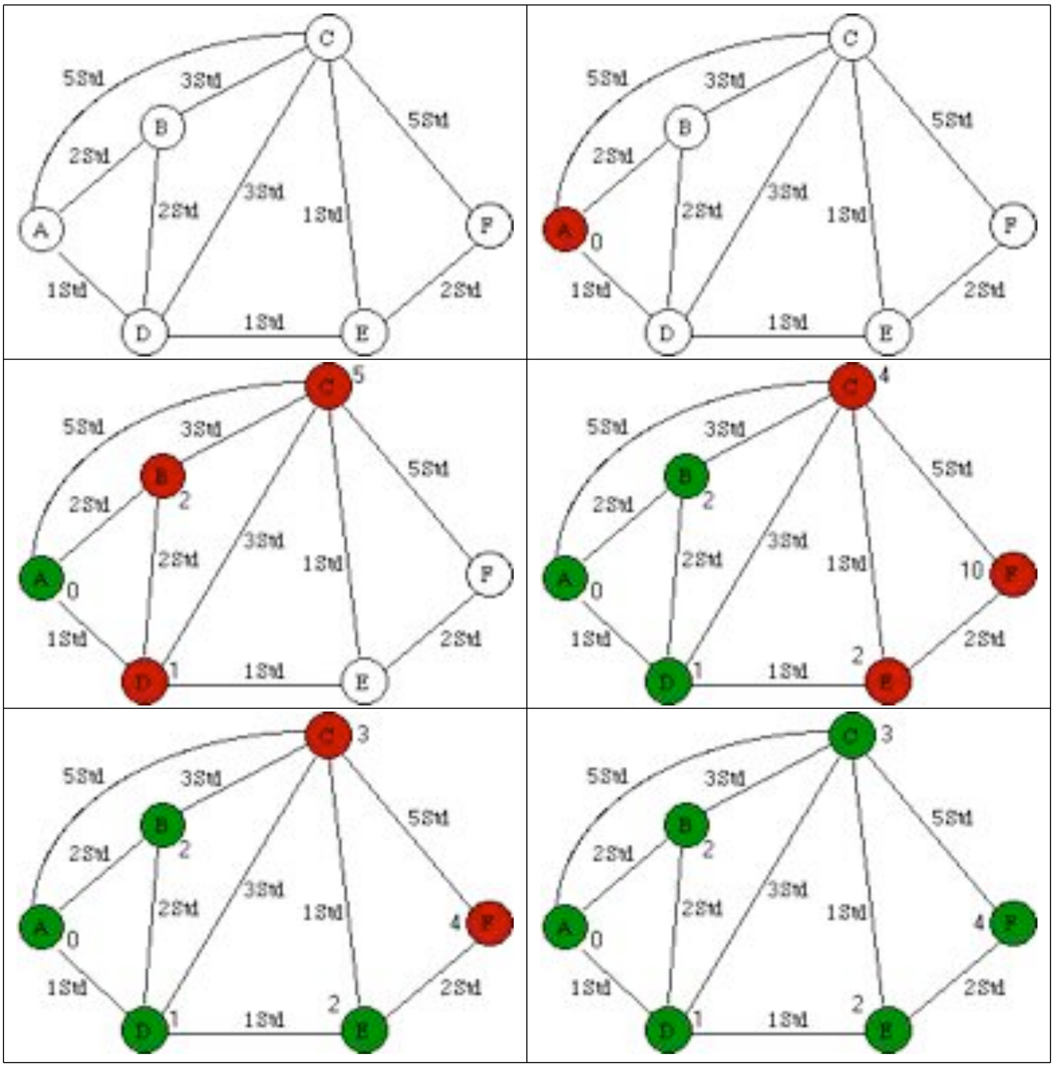
Die Graphik auf S. 1 ist Röthlisberger und Wittmann (1994) entnommen.

B.3 Arbeitsblatt Hausaufgabe von der 1./2. auf die 3. Stunde

Die folgende Abfolge von Graphen ist eine Möglichkeit, den kürzesten Weg von A nach C herauszufinden.

1) Erläutere, wie wohl der Algorithmus funktioniert.

2) Vergleiche diesen Algorithmus mit dem Beispiel, das Du in Deiner Gruppe erarbeitet hast. Nenne dazu Gemeinsamkeiten und Unterschiede. Hat der eine oder andere Algorithmus Vor- und Nachteile?



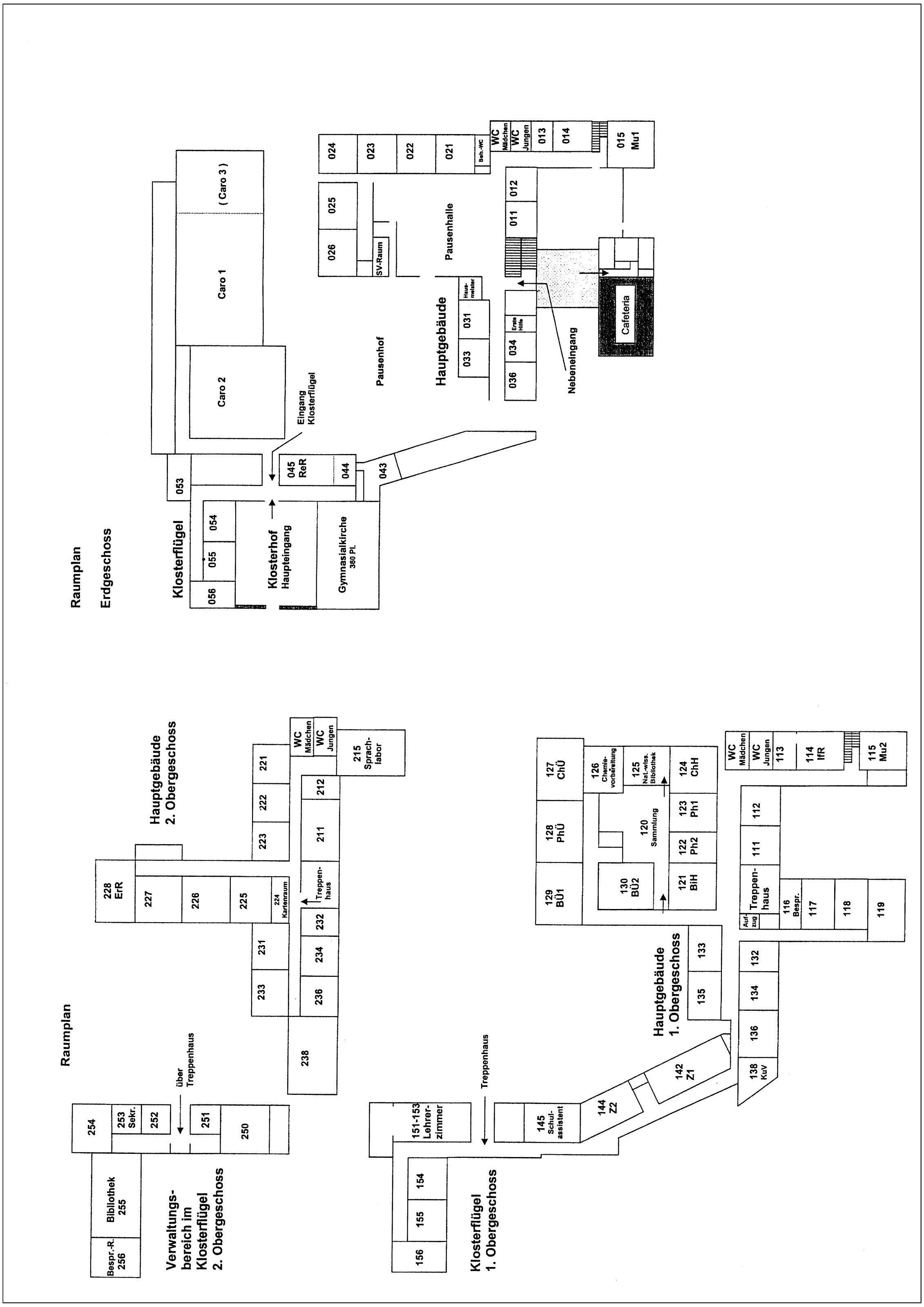
Die Graphiken sind Bearbeitungen einer Graphik aus dem Material von Röthlisberger und Wittmann (1994).

B.4 Vorgegebenes Programm

```java
public class KuerzesterWeg {

  /**
   * Ermittelt die höchste Zahl der Knotennamen.
   */
  public static int hoechsterKnoten(int[][] liste) {
    // der bisher höchste gefundene Knoten ist 0
    int maximum = 0;
    // für jeden Eintrag in der Liste
    for (int i = 0; i < liste.length; i++) {
      // ersetze das (bisherige) Maximum, wenn ein Knoten
      // mit höherer Zahl gefunden wurde
      if (liste[i][0] > maximum)
        maximum = liste[i][0];
      if (liste[i][1] > maximum)
        maximum = liste[i][1];
    }
    return maximum;
  }

  /**
   * Wandelt eine Nachbarschaftsliste in eine Nachbarschaftsmatrix um.
   */
  public static int[][] nachbarschaftsListe2Matrix(int[][] liste) {
    int[][] matrix =
      new int[hoechsterKnoten(liste) + 1][hoechsterKnoten(liste) + 1];
    for (int i = 0; i < matrix.length; i++)
      for (int j = 0; j < matrix.length; j++)
        matrix[i][j] = 0;
    for (int i = 0; i < liste.length; i++) {
      matrix[liste[i][0]][liste[i][1]] = liste[i][2];
      matrix[liste[i][1]][liste[i][0]] = liste[i][2];
    }
    return matrix;
  }

  /**
   * Berechnet den kürzesten Weg in matrix zwischen den Knoten a und b.
   * Gibt die Länge des Weges und den Weg selber auf dem Bildschirm aus.
   */
  public static void kuerzesterWeg(int[][] matrix, int von, int nach) {




  }

  /**
   * Haupt-Programm. Initialisiert alle Daten und führt die Suche aus.
   */
  public static void main(String[] argumente) {
    // zum Eingeben der Entfernungen: KnotenA, KnotenB, Entfernung in cm
    int[][] nachbarschaftsListe = {{0,1,120},{0,2,140}};

    // wird automatisch aus der Nachbarschaftsliste generiert
    int[][] nachbarschaftsMatrix = null;

    // Nachbarschaftsliste konvertieren
    nachbarschaftsMatrix = nachbarschaftsListe2Matrix(nachbarschaftsListe);
```

```

        // Kürzesten Weg von a nach b
        int a = 0;
        int b = 1;
        kuerzesterWeg(nachbarschaftsMatrix, a, b);
    }
}
```

B.5 Material „Schulplan“



B.6 Tabelle für Messergebnisse

Knoten 1	Knoten 2	Entfernung [cm]

Knoten 1	Knoten 2	Entfernung [cm]

B.7 Gruppenpuzzle

GRUPPE A

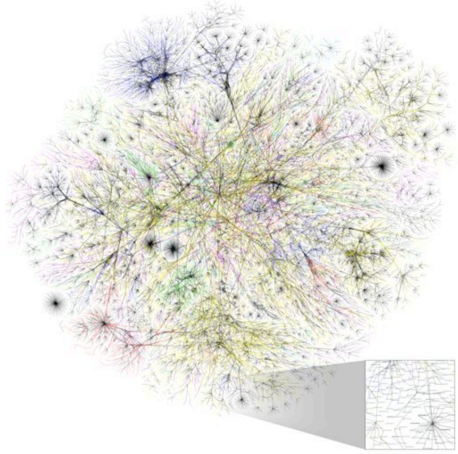
Schau Dir die folgenden Artikel an:
`http://de.wikipedia.org/wiki/Edsger_Wybe_Dijkstra`
`http://de.wikipedia.org/wiki/Dijkstras_Algorithmus`

Aufgaben:

1) Nenne relevante Punkte zum Leben von Edsger Wybe Dijkstra.
2) Beschreibe mehrere typische Anwendungen des Dijkstra-Algorithmus. Vergleiche mit Deinen Vermutungen im Test.
3) Es wurde vermutet, dass ein sinnvoller Algorithmus zum Finden eines kürzesten Weges alle Wege von A nach B überprüft. Überprüft der Dijkstra-Algorithmus alle Wege oder lässt er bestimmte Wege (wenn ja, welche) aus?

GRUPPE B

Das folgende Bild zeigt einen Ausschnitt des Internets, wie es sich uns heute darstellt. Daneben ist die Situation im Jahr 1974 (Das ARPANET ist ein direkter Vorgänger des Internet).



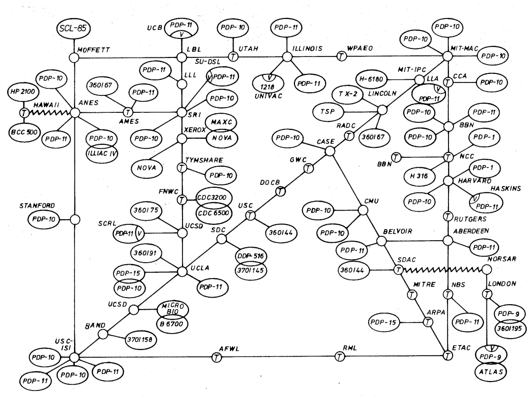


Abb. 4 ARPA NETwork, topologische Karte. Stand Juni 1974.

Aufgaben:

1) Datenpakete im Internet sollten einen sinnvollen Weg im Internet zurücklegen. Benenne Kriterien für einen „sinnvollen" Weg. Hat sich die grundlegende Situation seit 1974 geändert?
2) Wie kann der Dijkstra-Algorithmus helfen? Erläutere anhand eines selbst gewählten Beispiels anhand der Karte des ARPANET.
3) Wo hat der Dijkstra-Algorithmus Begrenzungen, wenn er im Internet angewendet werden soll?

GRUPPE C

Schau Dir den Quelltext des erarbeiteten Programms an.

Aufgaben:

1) Teste ein paar Verbindungen.
2) Benutze die Funktionen „Einzelne Anweisung" und „Ganze Routine", um Schritt für Schritt ein einfaches Beispiel nachvollziehen zu können.
3) Erkläre den Sinn und die Verwendung der Variablen `kuerzesterWeg`, `rueckweg` und `erreicht`.

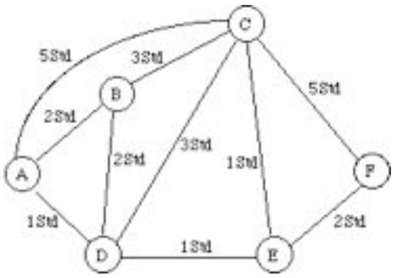
In der Gruppe A wird auf Wikipedia (2007e) und Wikipedia (2007d) verwiesen. Die Graphiken der Gruppe B sind eine Bearbeitung von Britt (2007) und aus Dodge (1974).

B.8 Nachtest

Kurs if20 **Kurztest** 21. März 2007

Name: ______________________________

Aufgabe 1. Der folgende Graph beschreibt, wie lange ein Zug braucht, um von einem Ort zu einem anderen zu kommen.



Beschreibe, wie der Dijkstra-Algorithmus in diesem Graphen den kürzesten Weg von A nach C finden kann. Der Dijkstra-Algorithmus ist ein so genannter „gieriger“ Algorithmus (greedy). Erläutere dies in Deinen Beschreibungen.

Aufgabe 2. Welche Kriterien sind im Internet für einen „kürzesten“ Weg sinnvoll?

Die Graphik ist Röthlisberger und Wittmann (1994) entnommen.

B.9 Umfrage

Bitte bewerte die folgenden Aussagen. Sie sollen dazu dienen, den Unterricht vom 9. März bis heute einzuschätzen. Sie sollen anonym beantwortet werden, sei dabei bitte ehrlich. Denk nicht zu lange über jede Aussage nach, sondern entscheide „aus dem Bauch heraus“. +3 bedeutet dabei: „Dem stimme ich voll und ganz zu“, -3 bedeutet: „Dem kann ich ganz und gar nicht zustimmen.“

Unterricht

	+3	+2	+1	0	-1	-2	-3	Enthaltung
Der Unterricht ist so gestaltet, dass ich ihm in der Regel gut folgen kann.								
Mir ist klar, welche Ziele im Unterricht erreicht werden sollen.								
Im Unterricht werden Theorie und Praxis angemessen miteinander verknüpft.								
Die Unterrichtsinhalte sind aktuell.								
Im Unterricht wird mir Gelegenheit zu eigenständigem Lernen und Arbeiten gegeben.								
Im Unterricht ist eine klare Struktur erkennbar.								
Mein Lehrer ist immer gut auf den Unterricht vorbereitet.								
Der Unterricht in meiner Klasse wird abwechslungsreich gestaltet.								
Die Unterrichtsinhalte haben Bezüge zu anderen Fächern.								
Mein Lehrer stellt angemessen hohe Leistungsanforderungen an die Klasse.								
Ich muss keine Angst haben, im Unterricht etwas falsch zu machen.								
Im Unterricht wird der PC sinnvoll eingesetzt.								
Die Bewertungsmaßstäbe zur Beurteilung meiner Leistungen sind mir bekannt und nachvollziehbar.								
Ich fühle mich von meinem Lehrer gerecht beurteilt.								
Im Unterricht wird kaum Zeit für Nebensächlichkeiten verschwendet.								

Lehrerpersönlichkeit

	+3	+2	+1	0	-1	-2	-3	Enthaltung
Der Lehrer hat ein hohes Maß an Fachwissen.								
Ich respektiere meinen Lehrer.								
Ich fühle mich von meinem Lehrer ernst genommen.								
Mein Lehrer erscheint pünktlich zum Unterricht.								
Mein Lehrer trägt dazu bei, dass ich gerne lerne.								
Ich lerne viel im Unterricht.								
Mein Lehrer ist an meiner schulischen/beruflichen Weiterentwicklung interessiert.								
Mein Lehrer reagiert angemessen auf Unterrichtsstörungen.								
Mein Lehrer fördert eine freundliche Atmosphäre in der Klasse.								
Insgesamt bin ich mit unserem Unterricht zufrieden.								

Was noch gesagt werden sollte:

Die Fragen sind größtenteils Schudak (2007a) und Schudak (2007b) entnommen.

B.10 Verwendete Version des Dijkstra-Algorithmus

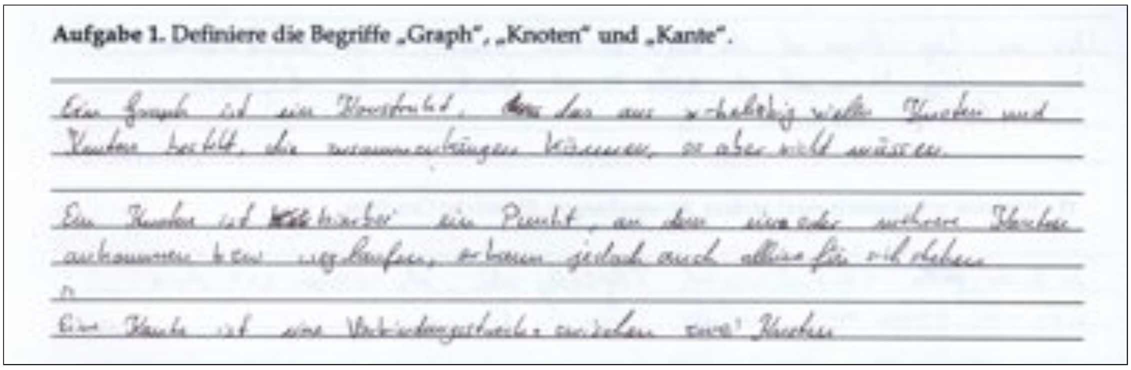
```
Dijkstra-Algorithmus(Matrix, Startknoten, Zielknoten)
for i ← 0 to Matrix.length
    do
        Entfernung[i] ← ∞
        Vorgaenger[i] ← −1
        Aktiv[i] ← false
Entfernung[StartKnoten] ← 0
Aktiv[StartKnoten] ← true
while ∃i(Aktiv[i] = true)
    do
        for i ← 0 to Aktiv.length
            do DemnaechstAktiv[i] ← false
        for i ← 0 to Aktiv.length
            do if Aktiv[i]
                do
                    for j ← 0 to Matrix.length
                        do if Entfernung[j] > Entfernung[i] + Matrix[i][j]∧
                            Matrix[i][j] ≠ 0
                                do
                                    DemnaechstAktiv[j] ← true
                                    Entfernung[j] ← Entfernung[i] + Matrix[i][j]
                                    Vorgaenger[j] ← i
        Aktiv ← DemnaechstAktiv
System.out.println("Entfernung: " + Entfernung[Zielknoten])
MomentanerKnoten ← Zielkoten
while MomentanerKnoten ≠ Zielknoten
    do System.out.println(MomentanerKnoten)
```

C Schülerlösungen

C.1 Vortest

C.1.1 Aufgabe 1

Dennis:


Aufgabe 1. Definiere die Begriffe „Graph", „Knoten" und „Kante".

Ein Graph ist ein Konstrukt, das aus x-beliebig vielen Knoten und Kanten besteht, die zusammenhängen können, es aber nicht müssen.

Ein Knoten ist hierbei ein Punkt, an dem eine oder mehrere Kanten ankommen bzw. weglaufen, er kann jedoch auch alleine für sich stehen.

Eine Kante ist eine Verbindungsstrecke zwischen zwei Knoten.

Sebastian:

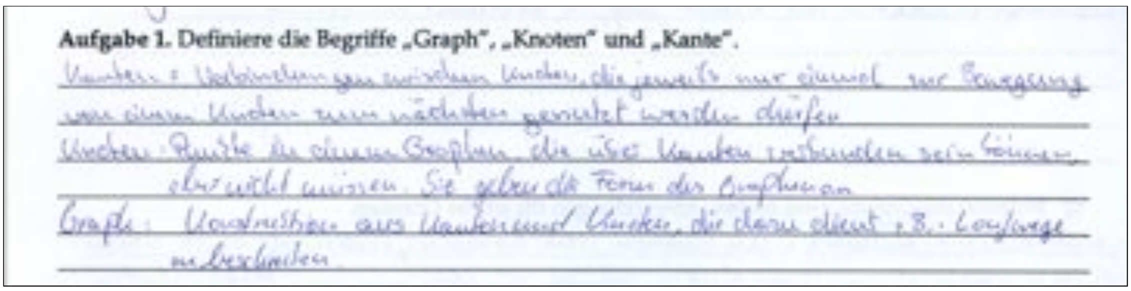

Aufgabe 1. Definiere die Begriffe „Graph", „Knoten" und „Kante".

Ein Graph ist ein Gebilde aus Kanten und [illegible], wobei die Anzahl der Kanten zählen, die von einem Knoten abgehen/hinführen.

Knoten sind die Verbindungen zwischen dem Graph und die Verbindung der Kanten.

Kanten dienen als Verbindung des Graphen und können beliebig oft von einem Knoten zum anderen verbunden sein.

Alexander:


Aufgabe 1. Definiere die Begriffe „Graph", „Knoten" und „Kante".

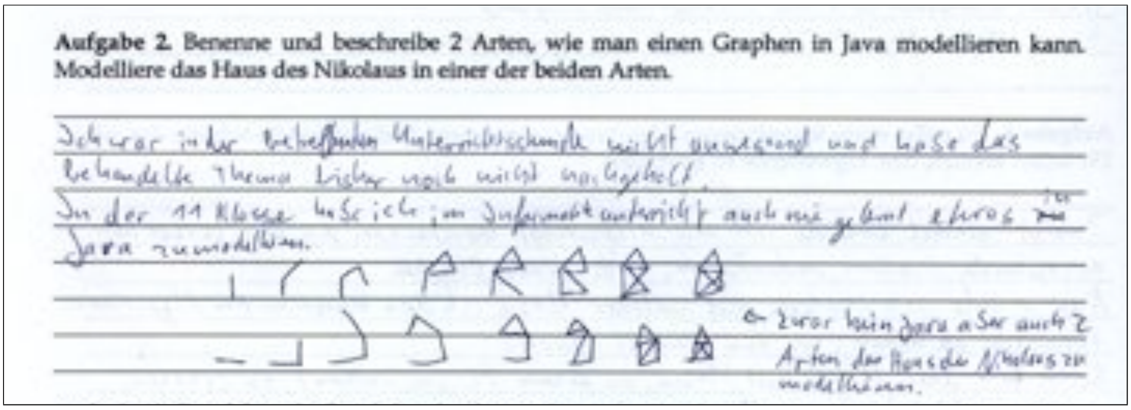
Kanten = Verbindungen zwischen Knoten, die jeweils nur einmal zur Bewegung von einem Knoten zum nächsten genutzt werden dürfen

Knoten: Punkte in einem Graphen, die über Kanten verbunden sein können, aber nicht müssen. Sie geben die Form des Graphen an

Graph: Konstruktion aus Kanten und Knoten, die dazu dient z.B. Laufwege zu beschreiben

C.1.2 Aufgabe 2

David:

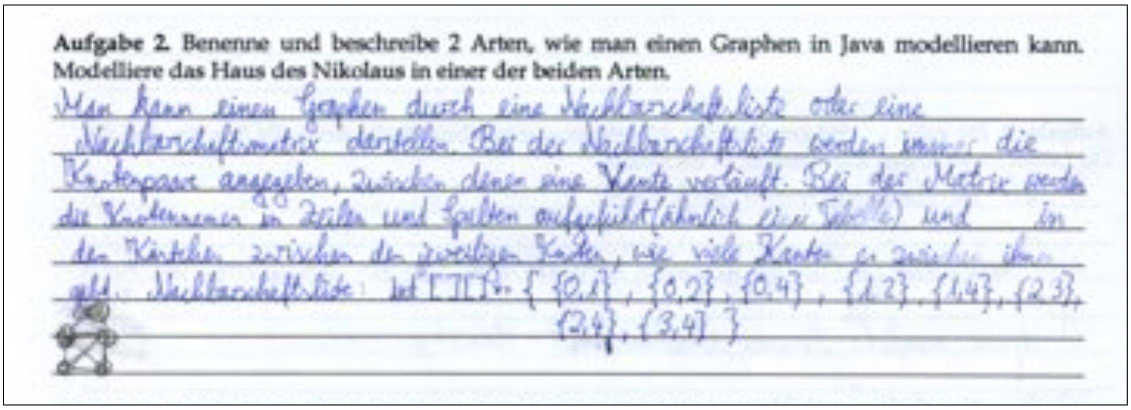


Aufgabe 2. Benenne und beschreibe 2 Arten, wie man einen Graphen in Java modellieren kann. Modelliere das Haus des Nikolaus in einer der beiden Arten.

Ich war in der betreffenden Unterrichtsstunde nicht anwesend und habe das behandelte Thema bisher noch nicht nachgeholt. In der 11 Klasse habe ich im Informatikunterricht auch nie gelernt etwas in Java zu modellieren.

← zwar kein Java aber auch 2 Arten das Haus des Nikolaus zu modellieren.

Katharina:

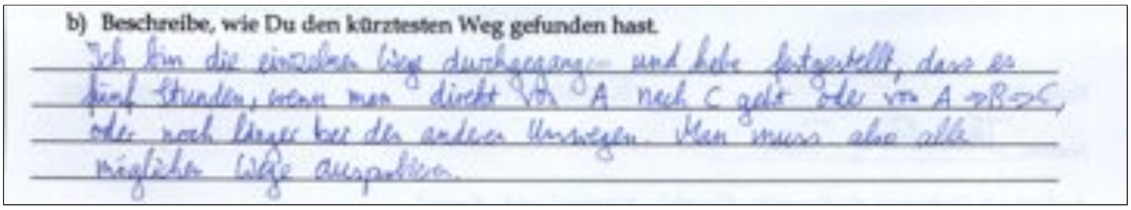


Aufgabe 2. Benenne und beschreibe 2 Arten, wie man einen Graphen in Java modellieren kann. Modelliere das Haus des Nikolaus in einer der beiden Arten.

Man kann einen Graphen durch eine Nachbarschaftsliste oder eine Nachbarschaftsmatrix darstellen. Bei der Nachbarschaftsliste werden immer die Knotenpaare angegeben, zwischen denen eine Kante verläuft. Bei der Matrix werden die Knotennamen in Zeilen und Spalten aufgeführt (ähnlich einer Tabelle) und in den Kästchen zwischen den jeweiligen Knoten, wie viele Kanten es zwischen ihnen gibt. Nachbarschaftsliste: int [][] l = { {0,1}, {0,2}, {0,4}, {1,2}, {1,4}, {2,3}, {2,4}, {3,4} }

C.1.3 Aufgabe 3

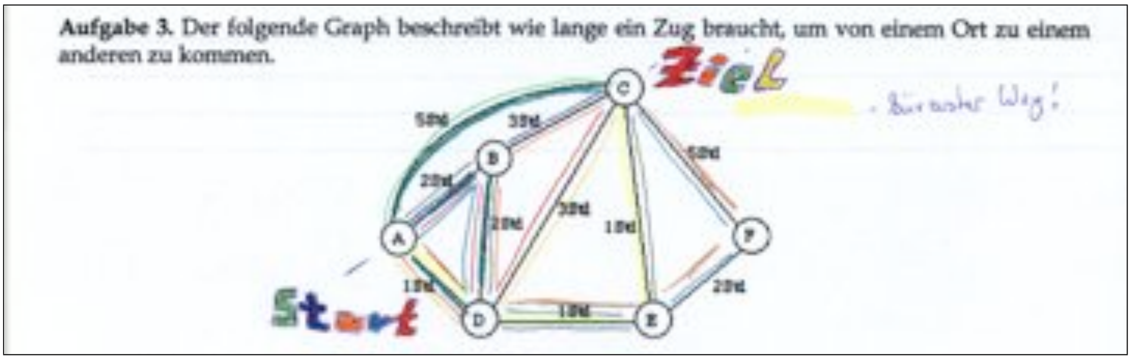
Katharina:



b) **Beschreibe, wie Du den kürztesten Weg gefunden hast.**

Ich bin die einzelnen Wege durchgegangen und habe festgestellt, dass es fünf Stunden, wenn man direkt von A nach C geht oder von A → B → C, oder noch länger bei den anderen Umwegen. Man muss also alle möglichen Wege ausprobieren.

Alexander:



Aufgabe 3. Der folgende Graph beschreibt wie lange ein Zug braucht, um von einem Ort zu einem anderen zu kommen.

b) Beschreibe, wie Du den kürztesten Weg gefunden hast.

Den kürzesten Weg habe ich gefunden indem ich von Punkt A alle Möglichkeiten zu den nächsten Punkten gesucht habe und von diesen wiederum alle Möglichkeiten zu weiteren Punkten, bis ich schließlich den bei „C" angekommen bin. Dann habe ich überprüft, welcher der kürzeste Weg in Anbetracht der Zeit ist.

Florian:

b) Beschreibe, wie Du den kürztesten Weg gefunden hast.

Gefunden habe ich den Weg ~~dass~~ durch simples betrachten des Graphens unter Berücksichtigung der Kanten-Attribute. Gesucht habe ich zuerst aber nach der schnellsten Verbindung über möglichst wenige Knoten dazwischen.

Dennis:

b) Beschreibe, wie Du den kürztesten Weg gefunden hast.

Ich habe vom Knoten A ausgehend alle Kanten „abgefahren", jeweils die Zeit addiert und geguckt, welcher Weg der schnellste sei.

War ein Weg länger als der bisher kürzeste, konnte ich ihn ausschließen. War ein Weg kürzer als der bisher kürzeste, konnte ich ihn als neuen kürzesten Weg annehmen.

Kevin:

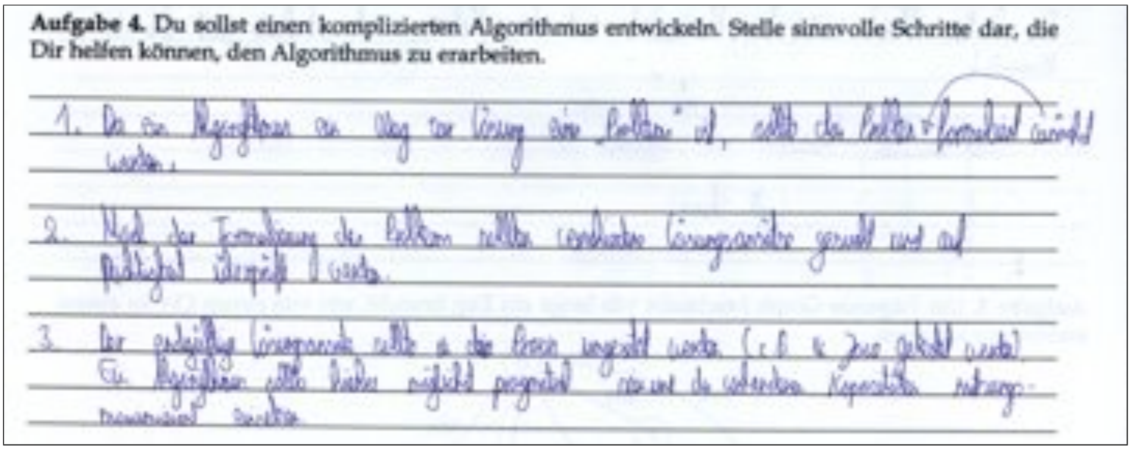
c) Benenne mindestens zwei andere Anwendungen für solche Graphen.

(Wandern =))

Zwei andere Anwendungsgebiete könnten z.B. die Vernetzung von verschiedenen Flughäfen sein oder aber ein „Computer-Netzwerk". → „LAN"

C.1.4 Aufgabe 4

Tobias:



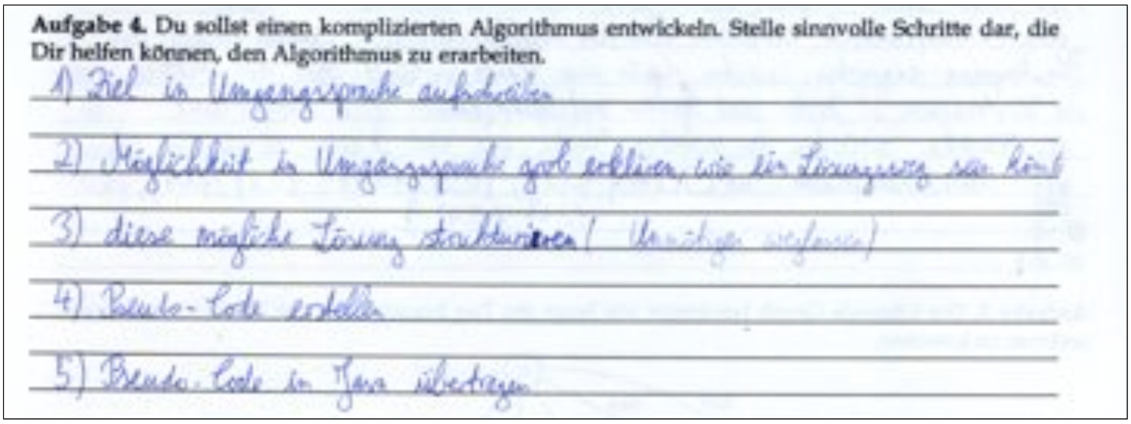
Aufgabe 4. Du sollst einen komplizierten Algorithmus entwickeln. Stelle sinnvolle Schritte dar, die Dir helfen können, den Algorithmus zu erarbeiten.

1. Da ein Algorithmus ein Weg zur Lösung eines „Problems" ist, sollte das Problem [illegible] werden.

2. Nach der Formulierung des Problems sollten verschiedene Lösungsansätze gesucht und auf Richtigkeit überprüft werden.

3. Der geeignetste Lösungsansatz sollte in die Praxis umgesetzt werden (z.B. in Java geschrieben werden). Ein Algorithmus sollte hierbei möglichst [illegible] [illegible] die [illegible] Kapazitäten [illegible] werden.

Katharina:



Aufgabe 4. Du sollst einen komplizierten Algorithmus entwickeln. Stelle sinnvolle Schritte dar, die Dir helfen können, den Algorithmus zu erarbeiten.

1) Ziel in Umgangssprache aufschreiben

2) Möglichkeit in Umgangssprache grob erklären, wie ein Lösungsweg aussehen könnt

3) diese mögliche Lösung strukturieren (Unnötiges weglassen)

4) Pseudo-Code erstellen

5) Pseudo-Code in Java übertragen

C.2 Gruppenarbeit 1./2. Stunde

Gruppe A:

Routing

1) Einigt euch auf eine gemeinsame Vorgehensweise, den kürzesten Weg in einem Graphen zu finden.

2) Notiert die Vorgehensweise ausführlich (vgl. Punkt 2 AB) „Entwickeln eines Algorithmus".

1) Wir fangen bei Knoten A (Startpunkt) an und gehen alphabetisch zum Knoten C (Endpunkt) alle möglichen Wege ab. Man startet mit dem direkten Weg zu C und sieht, dass die Strecke 5 Stunden beansprucht. Dies ist unser aktuell kürzester Weg. Danach läuft man über Punkt B zu Punkt C, merkt aber das sich die Wegstrecke nicht ver~~längert~~ kürzt. Wir fahren fort, über Punkt B und D zu C zu gelangen, erkennen aber das die Strecke deutlich länger wird. Nun versuchen wir über Punkt B und D nach zu Punkt E zu gelangen. Da wir aber bereits jetzt die bisher kürzeste Wegstrecke erreicht hat, können alle von hier weiterführenden Wege vernachlässigt werden, sodass wir uns einen neuen „Brückenpunkt" suchen können. Dies geschieht mit dem sofortigen anfahren vom Punkt D, ~~[illegible]~~ gefolgt vom direkten ansteuern des Zielpunktes. Hiermit haben wir eine neue kürzeste Streckenzeit von 4 Stunden. Als nächstes laufen wir über Punkt D und E zu Punkt C und bekommen bereits einen erneuten kürzesten ~~Punkt~~ Weg von 3 Stunden. Wenn man jetzt den nächsten Weg abgehen möchte, merkt man nach dem Weg D→E→F, dass dieser Weg gar nicht kürzer werden kann, womit wir unseren endgültigen kürzesten Weg herausgefunden haben.

Gruppe B:

Routing

1) Einigt Euch auf eine gemeinsame Vorgehensweise, den kürzesten Weg in einem Graphen zu finden.

2) Notiert die Vorgehensweise ausführlich (vgl. Punkt 2 des AB „Entwickeln eines Algorythmus").

2) ~~1)~~ Wir gehen alle möglichen Wege (Kanten) durch und ermitteln den kürzesten Weg anhand der nebenstehenden ~~Ka~~ Stundenanzahl zwischen den beiden Knoten, die wir als Start- und Zielknoten festgelegt haben.

Gruppe C:

Informatik 9.3.07

Routing

1) Einigt euch eine gemeinsame Vorgehensweise, den kürzesten Weg in einem Graphen zu finden.

2) Notiert die Vorgehensweise ausführlich (Vgl. Punkt 2 des AB „Entwickeln eines Algorithmus")

zu 2) Startpunkt suchen/festlegen (in diesem Falle A)

Legen einen Zielpunkt fest.

Knoten aktivieren und alle Punkte markieren die vom Startpunkt erreichbar sind.

Von den erreichten Knoten weiter zu den nächsten erreichbaren Knoten

Verfahren wiederholen bis wir schließlich an unserem Zielpunkt angelangt kommen

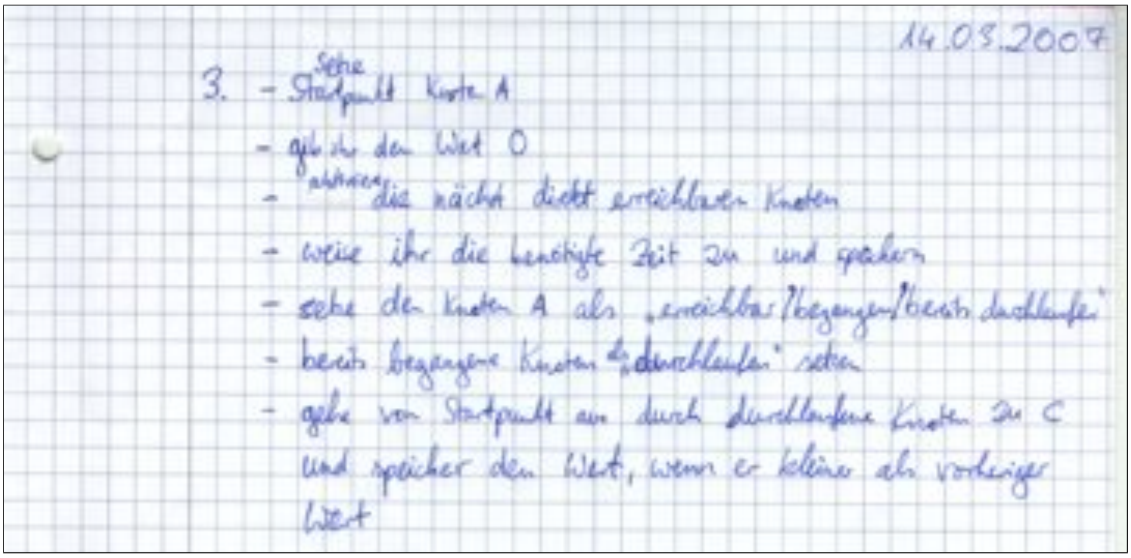
Beim Durchlaufen der Kanten fassen wir die Informationen der Kanten zusammen umso die kleinste, kürzeste etc Weg, Strecke zu ermitteln.

C.3 Gruppenarbeit 3. Stunde

Gruppe A:

Die Gruppe hatte aufgrund umso intensiverer Diskussionen keine schriftliche Lösung.

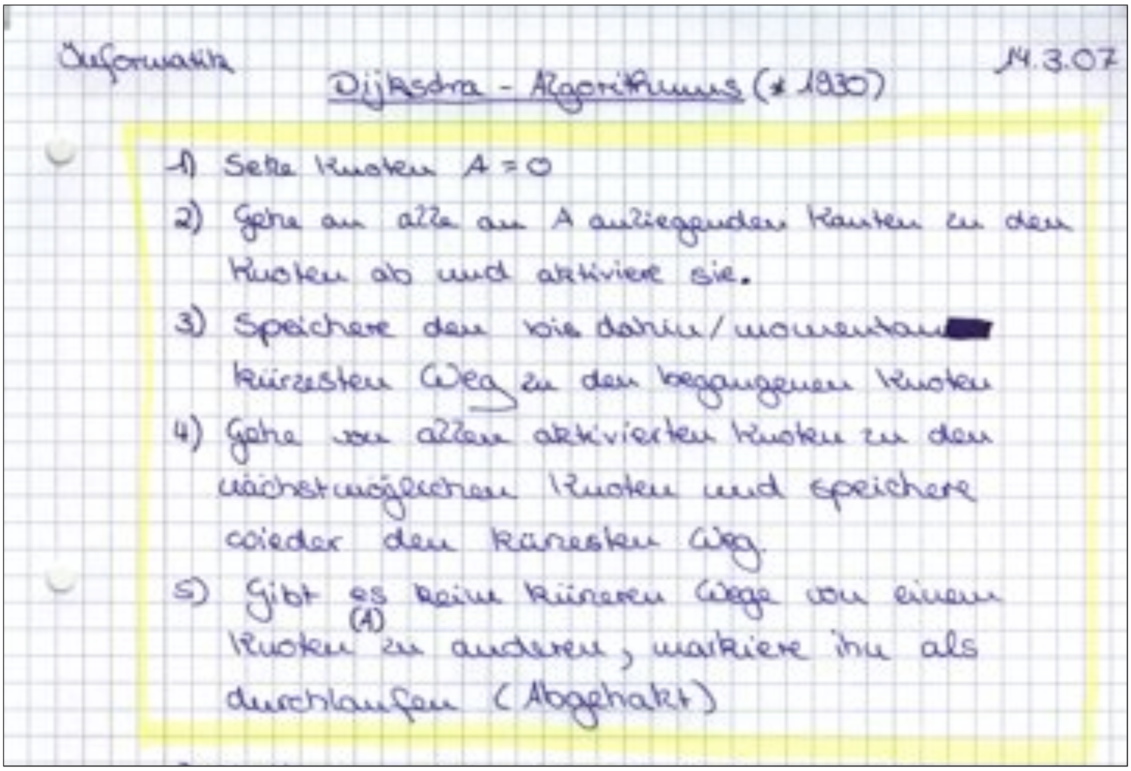
Gruppe B:



14.03.2007

3. - Setze Startpunkt Knoten A
- gib ihm den Wert 0
- aktiviere die nächst direkt erreichbaren Knoten
- weise ihr die benötigte Zeit zu und speichern
- setze den Knoten A als „erreichbar/begangen/bereits durchlaufen"
- bereits begangene Knoten als „durchlaufen" setzen
- gehe von Startpunkt aus durch durchlaufene Knoten zu C und speicher den Wert, wenn er kleiner als vorheriger Wert

Gruppe C:



Informatik 14.3.07

Dijkstra - Algorithmus (* 1930)

1) Setze Knoten A = 0
2) Gehe an alle an A anliegenden Kanten zu den Knoten ab und aktiviere sie.
3) Speichere den bis dahin/momentan kürzesten Weg zu den begangenen Knoten
4) Gehe von allen aktivierten Knoten zu den nächstmöglichen Knoten und speichere wieder den kürzesten Weg.
5) Gibt es keine kürzeren Wege von einem (A) Knoten zu anderen, markiere ihn als durchlaufen (Abgehakt)

C.4 Hefteintrag mit Pseudocode des Dijkstra-Algorithmus

Alexander:

Dijkstra - Algorithmus (* 1930)

1) Startknoten wird aktiviert; Entfernung ist 0.

2) Solange es noch aktive Knoten gibt:

  Für alle aktiven Knoten:

    Für alle von diesem Knoten direkt erreichbaren Knoten

      Ist der neue Weg kürzer => Knoten aktivieren, Entfernung speichern

    Knoten als erreicht markieren

Die Nummern „1.“ und „2.“ wurden von Alexander selbst hinzugefügt.

C.5 Schülerlösung des Programmes

```java
import java.io.*;

public class KuerzesterWeg {

  /**
   * Ermittelt die höchste Zahl der Knotennamen.
   */
  public static int hoechsterKnoten(int[][] liste) {
    // der bisher höchste gefundene Knoten ist 0
    int maximum = 0;
    // für jeden Eintrag in der Liste
    for (int i = 0; i < liste.length; i++) {
      // ersetze das (bisherige) Maximum, wenn ein Knoten
      // mit höherer Zahl gefunden wurde
      if (liste[i][0] > maximum)
        maximum = liste[i][0];
      if (liste[i][1] > maximum)
        maximum = liste[i][1];
    }
    return maximum;
  }

  /**
   * Wandelt eine Nachbarschaftsliste in eine Nachbarschaftsmatrix um.
   */
  public static int[][] nachbarschaftsListe2Matrix(int[][] liste) {
    int[][] matrix =
      new int[hoechsterKnoten(liste) + 1][hoechsterKnoten(liste) + 1];
    for (int i = 0; i < matrix.length; i++)
      for (int j = 0; j < matrix.length; j++)
        matrix[i][j] = -1;
    for (int i = 0; i < liste.length; i++) {
      matrix[liste[i][0]][liste[i][1]] = liste[i][2];
      matrix[liste[i][1]][liste[i][0]] = liste[i][2];
    }
    return matrix;
  }
```

```java
  /**
   * Berechnet den kürzesten Weg in matrix zwischen den Knoten a und b.
   * Gibt die Länge des Weges und den Weg selber auf dem Bildschirm aus.
   */
  public static void berechneKuerzestenWeg(int[][] matrix, int von, int nach) {

    boolean[] aktiv = new boolean[matrix.length + 1];
    boolean[] demnaechstaktiv = new boolean[matrix.length + 1];
    boolean[] erreicht = new boolean[matrix.length + 1];
    int[] kuerzesterWeg = new int [matrix.length + 1];
    int[] rueckweg = new int [matrix.length + 1];

    // alle Knoten nicht aktiv und nicht erreicht
    for (int h = 0; h <= matrix.length; h++) {
      erreicht[h] = false;
      aktiv[h] = false;
      demnaechstaktiv[h] = false;
      kuerzesterWeg[h] = 2147483647; // der größtmögliche Wert
      rueckweg[h] = 0;
    }
    // Startknoten aktivieren
    aktiv[von] = true;

    // Entfernung ist 0
    kuerzesterWeg[von] = 0;

    // solange es noch aktive Knoten gibt:
    boolean nochaktiv = true;
    while(nochaktiv) {

      for (int i = 0; i < matrix.length; i++) {

        // für alle aktiven Knoten:
        if (aktiv[i]) {
          // für alle von diesem Knoten direkt erreichbaren Knoten
          for (int j = 0; j < matrix.length; j++) {
            if (matrix[i][j] > 0) {
              // ist der neue Weg kürzer
              if (kuerzesterWeg[i] + matrix[i][j] < kuerzesterWeg[j]) {
                // Knoten aktivieren
                demnaechstaktiv[j] = true;

                // Entfernung speichern
                kuerzesterWeg[j] = kuerzesterWeg[i] + matrix[i][j];
                rueckweg[j] = i;
              }
            }
          }
          // Knoten als erreicht markieren
          erreicht[i] = true;
        }
      }
      aktiv = demnaechstaktiv;
      demnaechstaktiv = new boolean [matrix.length + 1];
      for (int k = 0; k < matrix.length + 1; k++) {
        demnaechstaktiv[k] = false;
      }
      nochaktiv = false;
      for (int k = 0; k < matrix.length; k++) {
        if (aktiv[k])
          nochaktiv = true;
```

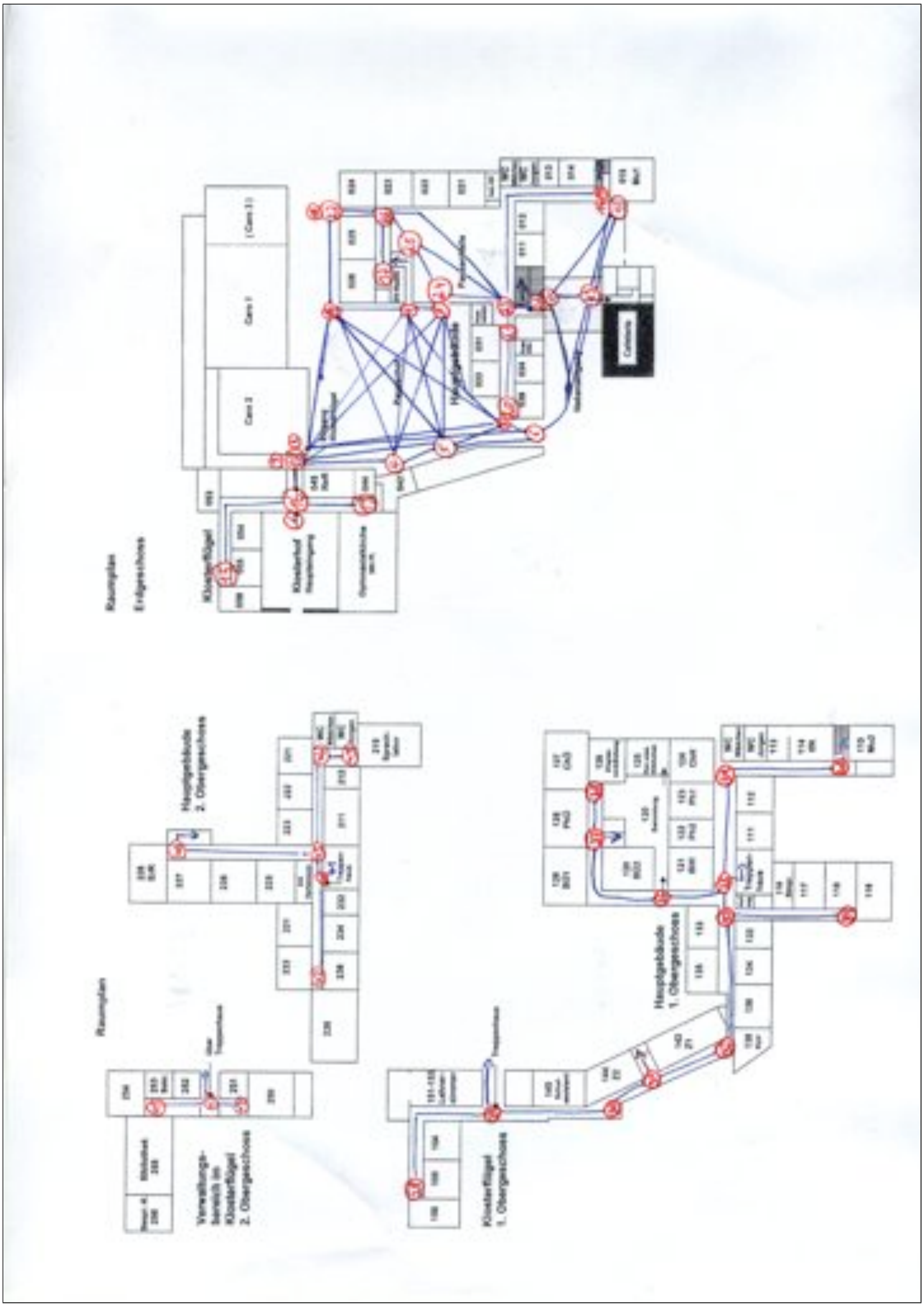
```
      }
    }
    int momentaneposition = nach;

    if (!erreicht[nach]) {
      System.out.println("Dieser Ort ist von " + von + " nicht erreichbar.");
    }
    else {
      System.out.print("Dieser Weg muss gegangen werden: " + nach);
      while(momentaneposition != von) {
        System.out.print(" <- " + rueckweg[momentaneposition]);
        momentaneposition = rueckweg[momentaneposition];
      }
      System.out.println(".");
      System.out.println("Das sind insgesamt " + kuerzesterWeg[nach] + " cm.");

    }
  }

  /**
   * Haupt-Programm. Initialisiert alle Daten und führt die Suche aus.
   */
  public static void main(String[] argumente) {
    System.out.println("Dieses Programm berechnet die Entfernung zwischen " +
      "zwei Orten im Carolinum.");

    // zum Eingeben der Entfernungen: KnotenA, KnotenB, Entfernung in cm
    int[][] nachbarschaftsListe = {
      // Erdgeschoss Innen
      {19,1,1920},
      {1,15,2450},
      {1,2,780},
      {26,10,910},
      {26,17,1250},
      {26,25,1510},
      {25,24,1110},
      {25,23,3120},
      {23,24,2050},
      {22,21,450},
      {21,7,2200},
      {24,23,4975},
      {23,26,3860},
      // Erdgeschoss Aussen
      {1,2,780},
      {2,3,320},
      {2,4,2300},
      {2,5,3000},
      {2,6,5650},
      {2,7,5100},
      {2,8,5200},
      {2,9,5200},
      {2,0,3600},
      {4,5,1400},
      {4,8,4250},
      {4,9,4200},
      {4,0,3800},
      {5,6,2400},
      {5,7,1950},
      {5,8,3700},
      {5,9,3600},
      {5,0,4000},
      {6,7,700},
```

```
      {6,11,3500},
      {6,12,5750},
      {6,13,3270},
      {7,8,3800},
      {7,9,4040},
      {7,0,4650},
      {8,9,650},
      {8,0,2250},
      {9,0,1800},
      {10,0,2440},
      {11,12,2860},
      {11,13,1360},
      {12,13,2580},
      // 1. Obergeschoss
      {28,29,3470},
      {29,1,1210},
      {29,30,2810},
      {30,31,1170},
      {31,5,1250},
      {31,32,1590},
      {32,33,2890},
      {33,34,2350},
      {33,35,545},
      {35,22,1630},
      {35,36,1555},
      {36,37,2355},
      {37,25,1910},
      {37,38,1150},
      {35,39,2650},
      {39,40,2370},
      {40,12,1450},
      // 2. Obergeschoss
      {35,41,1720},
      {41,42,2340},
      {41,43,780},
      {43,44,1900},
      {44,45,650},
      {43,46,3070},
      {37,46,1390},
      {47,49,1300},
      {47,48,850},
      {29,47,1500},
      // Zusätzliche Verbindungen
      {8,24,50},
      {11,22,200},
      {22,23,100}
    };

    // wird automatisch aus der Nachbarschaftsliste generiert
    int[][] nachbarschaftsMatrix = null;

    // Nachbarschaftsliste konvertieren
    nachbarschaftsMatrix = nachbarschaftsListe2Matrix(nachbarschaftsListe);

    // Auskommentieren, um die komplette Nachbarschaftsmatrix auszugeben
    // for (int i = 0; i < nachbarschaftsMatrix.length; i++) {
    //   for (int j = 0; j < nachbarschaftsMatrix.length; j++) {
    //     System.out.print(nachbarschaftsMatrix[i][j] + ", ");
    //   }
    //   System.out.println("");
    // }
```

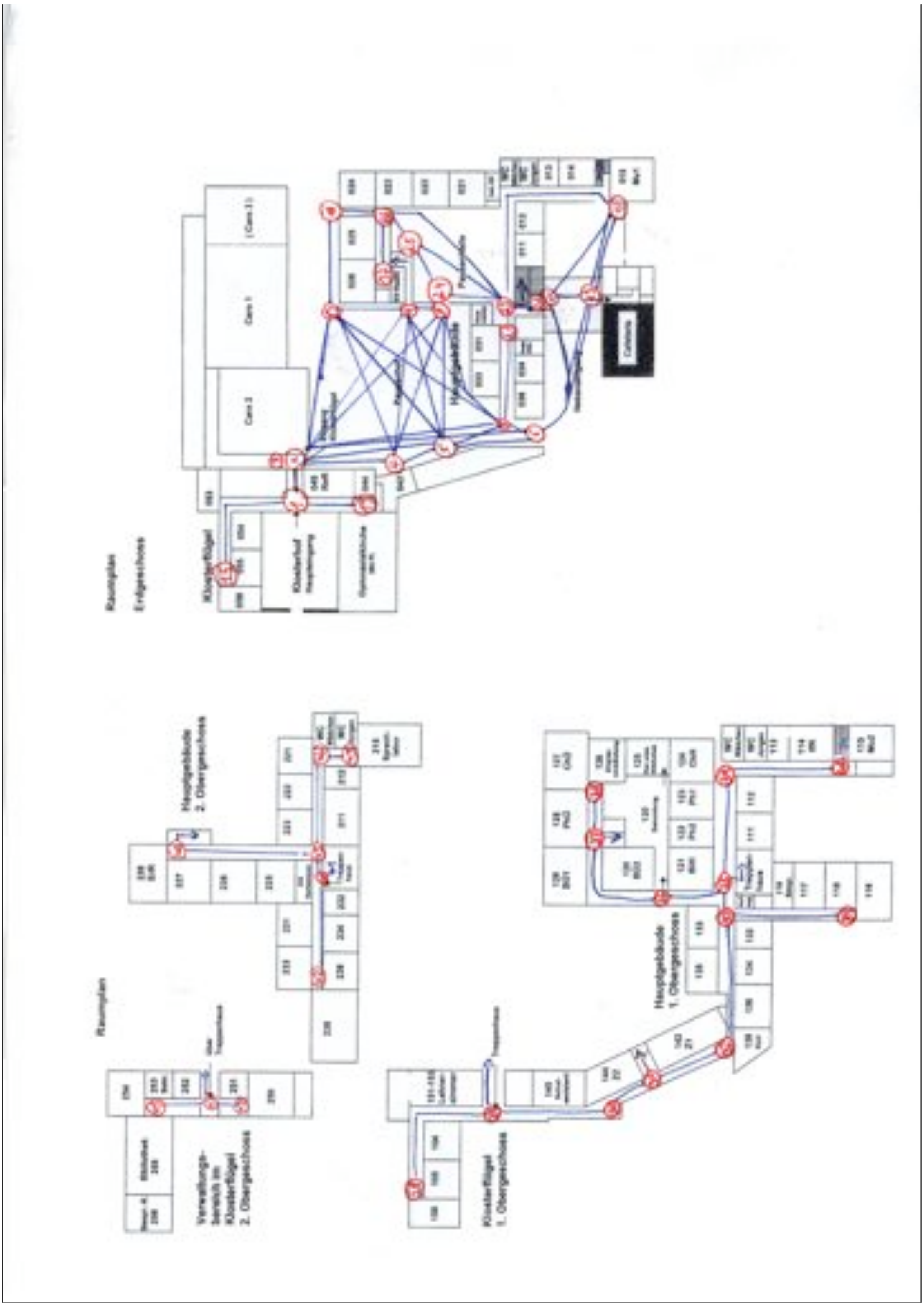
```java
    // Kürzester Weg von a nach b
    int a = -1;
    int b = -1;

    try {
      InputStreamReader eingabe = new InputStreamReader(System.in);
      BufferedReader in = new BufferedReader(eingabe);

      // Lese den Startpunkt ein...
      while (a == -1) {
        System.out.print("Bitte geben Sie den Startpunkt an: ");
        try {
          a = Integer.parseInt(in.readLine());
        } catch (NumberFormatException e) {
          System.out.print("Fehler, bitte geben Sie den Startpunkt an: ");
          a = -1;
        }
        if (a < 0 || a > nachbarschaftsMatrix.length) {
          System.out.print("Fehler, bitte geben Sie den Startpunkt an: ");
          a = -1;
        }
      }

      // Lese den Zielpunkt ein...
      while (b == -1) {
        System.out.print("Bitte geben Sie den Endpunkt an: ");
        try {
          b = Integer.parseInt(in.readLine());
        } catch (NumberFormatException e) {
          System.out.print("Fehler, bitte geben Sie den Endpunkt an: ");
          b = -1;
        }
        if (b < 0 || b > nachbarschaftsMatrix.length) {
          System.out.print("Fehler, bitte geben Sie den Endpunkt an: ");
          b = -1;
        }
      }
    } catch (IOException e) {
      System.out.println("Allgemeiner Fehler beim Lesen der Eingabe.");
    }

    berechneKuerzestenWeg(nachbarschaftsMatrix, a, b);

  }
}
```

Die Änderungen der Schülerlösung durch den Lehrer, die hier schon eingearbeitet worden sind, sind in Kapitel 2.5 beschrieben.

C.6 Knotenfestlegungen auf dem Schulplan (Schülerversion)



C.7 Knotenfestlegungen auf dem Schulplan (bearbeitete Version)



C.8 Umfrageergebnisse

Frage	$\bar{x}$	$\sigma_{\bar{x}}$
Der Unterricht ist so gestaltet, dass ich ihm in der Regel gut folgen kann.	1,90	0,57
Mir ist klar, welche Ziele im Unterricht erreicht werden sollen.	1,70	1,06
Im Unterricht werden Theorie und Praxis angemessen miteinander verknüpft.	1,50	1,27
Die Unterrichtsinhalte sind aktuell.	0,90	0,99
Im Unterricht wird mir Gelegenheit zu eigenständigem Lernen und Arbeiten gegeben.	1,90	0,99
Im Unterricht ist eine klare Struktur erkennbar.	1,89	0,78
Mein Lehrer ist immer gut auf den Unterricht vorbereitet.	2,70	0,48
Der Unterricht in meiner Klasse wird abwechslungsreich gestaltet.	1,20	1,55
Die Unterrichtsinhalte haben Bezüge zu anderen Fächern.	−0,40	1,35
Mein Lehrer stellt angemessen hohe Leistungsanforderungen an die Klasse.	1,33	1,32
Ich muss keine Angst haben, im Unterricht etwas falsch zu machen.	2,40	0,52
Im Unterricht wird der PC sinnvoll eingesetzt.	2,00	0,67
Die Bewertungsmaßstäbe zur Beurteilung meiner Leistungen sind mir bekannt und nachvollziehbar.	0,50	1,78
Ich fühle mich von meinem Lehrer gerecht beurteilt.	1,50	1,43
Im Unterricht wird kaum Zeit für Nebensächlichkeiten verschwendet.	0,90	1,37
Der Lehrer hat ein hohes Maß an Fachwissen.	2,50	0,71
Ich respektiere meinen Lehrer.	2,20	0,63
Ich fühle mich von meinem Lehrer ernst genommen.	2,10	0,88
Mein Lehrer erscheint pünktlich zum Unterricht.	2,90	0,32
Mein Lehrer trägt dazu bei, dass ich gerne lerne.	0,89	1,69
Ich lerne viel im Unterricht.	1,00	1,15
Mein Lehrer ist an meiner schulischen/beruflichen Weiterentwicklung interessiert.	1,33	1,51
Mein Lehrer reagiert angemessen auf Unterrichtsstörungen.	1,10	1,45
Mein Lehrer fördert eine freundliche Atmosphäre in der Klasse.	2,00	0,67
Insgesamt bin ich mit unserem Unterricht zufrieden.	1,80	0,79

Umfrageergebnisse

Freie Texte („Was noch gesagt werden sollte“):

„Alles in Allem muss ich sagen, dass der Unterricht doch ganz gut strukturiert ist, und [...][1] sehr viel Spaß macht.“

„Theoretische Inhalte (das Erlernen der ‚Sprache‘ Java) werden (sehr selten) nicht intensiv vermittelt. Ich hätte mir einen intensiveren Lernprozess gewünscht. Der folgende Unterrichtsinhalt war mir zuweilen nicht immer klar.“

„Insgesamt guter Unterricht – machen Sie weiter so!“

„Der Unterricht sollte auf keinen Fall verändert werden, sondern so bleiben wie er ist!“

[1] Diese Textstelle, die eine Aussage über einen anderen Lehrer enthält, habe ich zur Wahrung der Persönlichkeitsrechte ausgelassen. Sie ändert nicht den Sinn der restlichen Aussage.

C.9 Nachtest

C.9.1 Aufgabe 1

Sebastian:

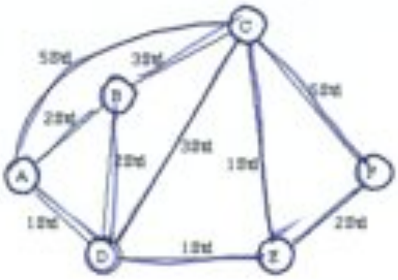
Beschreibe, wie der Dijkstra-Algorithmus in diesem Graphen den kürzesten Weg von A nach C finden kann. Der Dijkstra-Algorithmus ist ein so genannter „gieriger" Algorithmus (greedy). Erläutere dies in Deinen Beschreibungen.

Es wird zunächst immer der Startpunkt A festgelegt, nun werden alle Kanten, die von A zu einem anderen Punkt führen (also B, C, D), abgefahren und die Zeit wird gespeichert. Von diesen neuen Knoten, die nun auf aktiv gesetzt werden, findet nun erneut Überprüfung der zu erreichenden Knoten statt. Die Zeiten werden wiederum gespeichert. So fährt der Algorithmus weiter fort, bis er alle Knoten überprüft hat. Die Zeit, um einen Punkt vom Startpunkt A zu erreichen wird an jedem ~~[illegible]~~ Knoten notiert.

Gieriger Algorithmus bedeutet nun, dass alle Knoten überprüft werden, und eben nicht nur eine Verbindung von A zu C gesucht wird, sondern eine Verbindung von A zu allen Punkten des Graphen, wenn sie denn mit diesem in Verbindung durch eine Kante stehen.

Florian:

Aufgabe 1. Der folgende Graph beschreibt wie lange ein Zug braucht, um von einem Ort zu einem anderen zu kommen.



Beschreibe, wie der Dijkstra-Algorithmus in diesem Graphen den kürzesten Weg von A nach C finden kann. Der Dijkstra-Algorithmus ist ein so genannter „gieriger" Algorithmus (greedy). Erläutere dies in Deinen Beschreibungen.

Der Algorithmus überprüft alle von A aus erreichbaren Knoten und hält den bis dahin kürzesten Weg fest. Dies wird fortgesetzt, bis alle von A aus erreichbaren Knoten mindestens einmal aktiv waren. Da ~~die~~ der kürzeste Weg zu jedem Knoten immer mit abgespeichert wurde, ist nun der kürzeste Weg zu jedem von A ausgehenden Knoten bekannt. So wird sichergestellt, dass durch die Ausschöpfung aller Möglichkeiten, auch wirklich der kürzeste Weg zu C gefunden wird.

Tobias:

Beschreibe, wie der Dijkstra-Algorithmus in diesem Graphen den kürzesten Weg von A nach C finden kann. Der Dijkstra-Algorithmus ist ein so genannter „gieriger" Algorithmus (greedy). Erläutere dies in Deinen Beschreibungen.

Der ~~Dijk~~ Dijkstra-Algorithmus geht [illegible] nacheinander alle von einem beliebigen Startpunkt aus erreichbaren Knoten, der Pfad [illegible], um auf diese Weise den kürzesten Weg zu ermitteln. [illegible] er [illegible] nacheinander die erreichbaren Knoten und speichert sie in einer Tabelle oder [illegible].

Da der Dijkstra-Algorithmus alle [illegible] vom Startpunkt aus erreichbaren Knoten und die von dort aus erreichbaren Knoten abgeht, um den kürzesten Weg zu ermitteln, spricht man auch von einem „gierigen" (engl. greedy) Algorithmus.

C.9.2 Aufgabe 2

Sebastian:

Aufgabe 2. Welche Kriterien sind im Internet für einen „kürzesten" Weg sinnvoll?

Für einen kürzesten Weg im Internet ist es also sinnvoll, dass dieser Weg von einem Datenpaket in [illegible] kürzester Zeit absolviert wird. [illegible] also [illegible] die Millisekunden [illegible], um dann eine sehr geringe Transportzeit zu erreichen, [illegible] eine große [illegible] des Netzes, sodass es, auch wenn viele Pakete unterwegs sind, nicht langsamer wird. Da das Internet aus vielen Verzweigungen besteht, bedeutet der „kürzeste" Weg auch, so wenig Knotenpunkte wie möglich zu passieren, und so eben den schnellsten und kürzesten Weg zu nutzen.

Katharina:

Aufgabe 2. Welche Kriterien sind im Internet für einen „kürzesten" Weg sinnvoll?

Im Internet sind Kriterien wie Zeit (gemessen in ms) und Datenvolumen (gemessen in MB[Megabyte] oder KB[Kilobyte]) für einen „kürzesten" Weg sinnvoll

Kevin:

Aufgabe 2. Welche Kriterien sind im Internet für einen „kürzesten" Weg sinnvoll?

Wichtig ist die Messung des „kürzesten" Weges im Internet. Nicht die tatsächliche Entfernung oder das Datenvolumen sind ausschlaggebend, sondern die Zeit, ~~[illegible]~~ angegeben in ms. ~~Sinnvolle Kriterien sind~~ Abhängig ist die Zeit u.a. von der Anzahl der Knoten. Außerdem sollte der Weg, also die Leitungen stabil und sicher sein, um zu gewährleisten, dass der Weg auch wirklich die kürzeste Zeit in Anspruch nimmt.

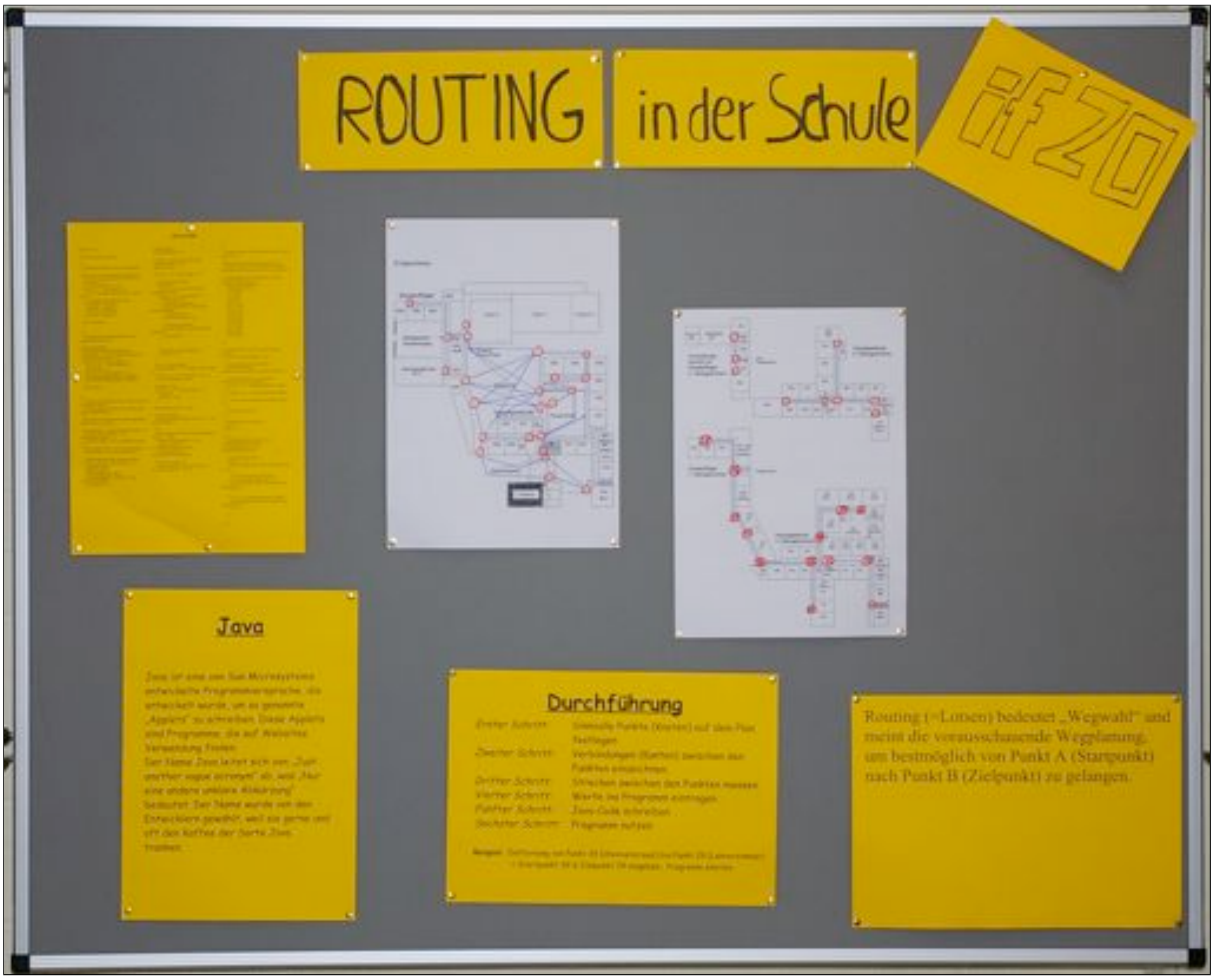
Marcel:

Aufgabe 2. Welche Kriterien sind im Internet für einen „kürzesten" Weg sinnvoll?

Sinnvoll wäre eine direkte Verbindung von Server A zu Server B. Wenn ein Datenpaket erst über 5 Server zu Server B gelangt und dafür 20 ms benötigt, dann ~~[illegible]~~ wäre es sinnvoller einen direkten Weg zu wählen, der nur 10 ms in Anspruch nimmt.

D Materialien nach der Examensreihe

D.1 Stellwand „Routing in der Schule“



D.2 Stationenlernen „Klassische Probleme der Graphentheorie“

Station 1: Problem des Handlungsreisenden – Travelling Salesman Problem

Problem: Handlungsreisende müssen bestimmte Städte bereisen. Dies sollten sie möglichst schnell machen können.

Aufgaben:

1. Die Städte Adorf (Index 0), Bestadt (Index 1), Cecity (Index 2) und Dedesheim (Index 3) sollen besucht werden. Die Entfernungen in Kilometern sind in der folgenden Nachbarschaftsmatrix gespeichert:

$$N = \begin{pmatrix} 0 & 20 & 42 & 35 \\ 20 & 0 & 30 & 34 \\ 42 & 30 & 0 & 12 \\ 35 & 34 & 12 & 0 \end{pmatrix}$$

   a) Skizziere die Lage der Städte (nicht maßstabsgetreu) und trage die Entfernungen an den Wegen ein.
   b) Gib zwei verschiedene Rundwege an, die über alle vier Städte gehen.
   c) Vergleiche die Längen.
   d) Gib den kürzesten Rundweg an, den du finden kannst.

2. a) Gib an, wie viele verschiedene Rundwege es bei vier Städten gibt.
   b) Findest Du eine allgemeine Formel in Abhängigkeit der Anzahl der Städte n, mit der Du berechnen kannst, wie viele verschiedene Rundwege es gibt? Geh davon aus, dass man von jeder Stadt direkt zu einer anderen Stadt gehen kann, dass also zwischen je zwei Knoten immer eine Kante existiert.
   c) Berechne mit Hilfe Deiner Formel, wie viele Rundwege es bei 15 Städten und wie viele es bei 18 Städten gibt. Bist Du erstaunt?
   d) Geh mal davon aus, dass das Berechnen eines Weges eine Millisekunde dauert. Wie lange dauert es, alle Wege bei 15 bzw. 18 Städten zu berechnen? 18 Städte sind 20% mehr als 15 Städte. Um wie viel Prozent dauert es länger, alle Wege zu berechnen?

3. Hast Du die Station 2 (Briefträgerproblem) schon gemacht? Dann bearbeite noch die folgenden Aufgaben:
   a) Beschreibe die Unterschiede zwischen dem Briefträgerproblem und dem Problem eines Handlungsreisenden.
   b) Beschreibe die Unterschiede der Komplexität der Probleme.

Das Beispiel ist Wikipedia (2007f) entnommen.

Station 2: Briefträgerproblem – Chinese Postman Problem

Problem: Ein Briefträger möchte möglichst schnell seine Briefe verteilt haben. Dazu muss er alle Straßen mindestens einmal abgegangen sein und wieder am Ausgangspunkt ankommen.

Aufgaben:

1. In der folgende Nachbarschaftsliste sind acht Straßen zwischen fünf Knoten gespeichert.

Knoten A	Knoten B	Distanz [km]
0	1	1,12
0	2	1,12
1	2	1,00
1	3	1,00
1	4	1,41
2	3	1,41
2	4	1,00
3	4	1,00

   a) Zeichne eine (nicht maßstabsgetreue) Skizze der Straßen.
   b) Gib zwei verschiedene Rundwege an und berechne die Gesamtlänge.
   c) Ist es egal, wo man anfängt?
   d) Gib den kürzesten Rundweg an, den Du finden kannst.

2. Bei bestimmten Graphen ist es ganz einfach, einen optimalen Rundweg zu finden. Du kennst solche Graphen sogar schon: Graphen, die einen Eulerkreis besitzen.
   a) Beschreibe, was ein Eulerkreis ist.
   b) Besitzt das Beispiel aus Aufgabe 1 einen Eulerkreis? Begründe bzw. widerlege!
   c) Beweise, dass ein Eulerkreis tatsächlich ein kürzester Rundweg ist.

3. Hast Du die Station 1 (Problem des Handlungsreisenden) schon gemacht? Dann bearbeite noch die folgenden Aufgaben:
   a) Beschreibe die Unterschiede zwischen dem Briefträgerproblem und dem Problem eines Handlungsreisenden.
   b) Beschreibe die Unterschiede der Komplexität der Probleme.

Station 3: Hamiltonkreise

Problem: *Soll in Aufgabe 1.d definiert werden.*

Aufgaben:

1. Der Irische Mathematiker William Rowan Hamilton (1805–1865) erfand das folgende Spiel *„Icosian Game“* 1857:

Dabei sollen alle Ecken des Dodekaeders genau einmal abgelaufen werden, bevor man zum Ausgangspunkt zurückkehrt.

   a) Zeichne einen Dodekaeder wie auf dem Spielbrett.
   b) Finde eine Lösung für das Problem.
   c) Beschreibe Deine Strategie zur Lösung des Problems.
   d) Definiere das Problem mit Begriffen der Graphentheorie.

2. *„Zunächst erscheint die Aufgabenstellung ähnlich dem 1736 von L. Euler (verneinend) gelösten Königsberger Brückenproblem, einem Spezialfall des Eulerkreisproblems und Grundsteinlegung der Graphentheorie. Während für das Eulerkreisproblem aber besonders effiziente Lösungs-Algorithmen existieren, ist bekannt, dass beide Varianten des Hamiltonkreisproblems besonders schwer algorithmisch lösbare Probleme sind. Sowohl die gerichtete als auch die ungerichtete Variante des Hamiltonkreisproblems gehört zur Liste der 21 klassischen NP-vollständigen Probleme, für die Richard Karp 1972 in seinem berühmten Artikel die Zugehörigkeit zu dieser Klasse von Problemen nachgewiesen hat.“*[1]

   a) Vergleiche das Hamiltonkreisproblem mit dem Eulerkreisproblem. Wo sind Unterschiede, wo sind Gemeinsamkeiten?
   b) Finde Gründe, warum das Hamiltonkreisproblem wohl so schwer durch einen Algorithmus zu lösen ist.
   c) „NP-vollständig“ bedeutet, dass die Laufzeit in Abhängigkeit von der Anzahl der Knoten x nicht durch eine ganzrationale Funktion ($f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$) beschrieben werden kann. Zeichne die folgenden Funktionen und ordne sie danach, wie sie sich bei beliebig großen x verhalten: $f(x) = 1$, $g(x) = x^3$, $h(x) = x \cdot \ln(x)$, $i(x) = \ln(x)$, $j(x) = x$, $k(x) = x^{10}$, $l(x) = \sqrt{x}$, $m(x) = x!$, $n(x) = 2^x$, $o(x) = x + 1$.

[1] Wikipedia (2007): *Hamiltonkreisproblem.* URL: `http://de.wikipedia.org/wiki/Hamiltonkreisproblem` [Online; Stand 12. April 2007].

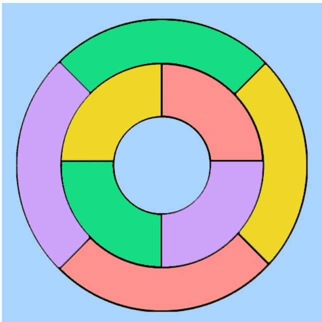
Die Graphik ist Dalgety (2007) entnommen.

Station 4: Vier-Farben-Problem

Problem: Es sollen möglichst wenige Farben verwendet werden, um eine Landkarte (ohne Enklaven wie z. B. West-Berlin in der ehemaligen DDR) zu färben.

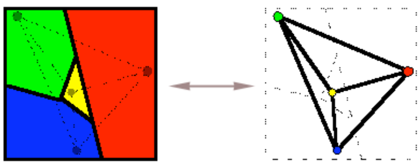
Aufgaben:

1. a) Zeichne eine fiktive Landkarte. Ordne den einzelnen Ländern möglichst wenig Farben zu. Dabei sollen zwei aneinander stoßende Länder verschiedene Farben haben.
   b) Konstruiere je ein Beispiel, das zeigt, dass es notwendig ist, mindestens drei bzw. vier Farben zu verwenden.
   c) Es wurde immer wieder vermutet, dass sogar fünf Farben notwendig sind. Die folgende Graphik wird dabei häufig als Beweis herangezogen. Widerlege diese Behauptung, indem Du nur vier Farben verwendest, um die Landkarte einzufärben.

   Der Beweis, dass tatsächlich nur vier Farben ausreichen, ist viel schwerer, als an einem konkreten Beispiel zu zeigen, dass nur vier Farben ausreichen. Hierfür wurde 1977 ein Computer verwendet. Das Vier-Farben-Problem ist damit das erste große mathematische Problem, das mit Hilfe eines Computers gelöst wurde.

2. a) Das Vier-Farben-Problem lässt sich auch mit Graphen modellieren. Dabei werden die Knoten entsprechend gefärbt, während die Kanten die „Grenzen“ zwischen zwei Ländern darstellen.

   Zeichne einen Graphen für die von Dir in Aufgabe 1.c erstellte Landkarte.
   b) Diese Graphen haben eine besondere Eigenschaft, die man „planar“ nennt. Konstruiere einen Graphen, der sich nicht mit vier Farben färben lässt, ohne dass benachbarte Knoten die gleiche Farbe haben.
   c) Definiere die Eigenschaft „planar“ eines Graphen.

Die Graphiken sind Wikipedia (2007h) entnommen.

Station 5: Echtes Routing

Problem: Der Dijkstra-Algorithmus findet zwar gut kürzeste Wege, hierzu muss aber der komplette Graph bekannt sein. Das Internet verändert sich jedoch ständig, es ist viel zu ineffizient, zunächst einen kompletten Graphen des Internets zu erstellen.

Aufgaben:

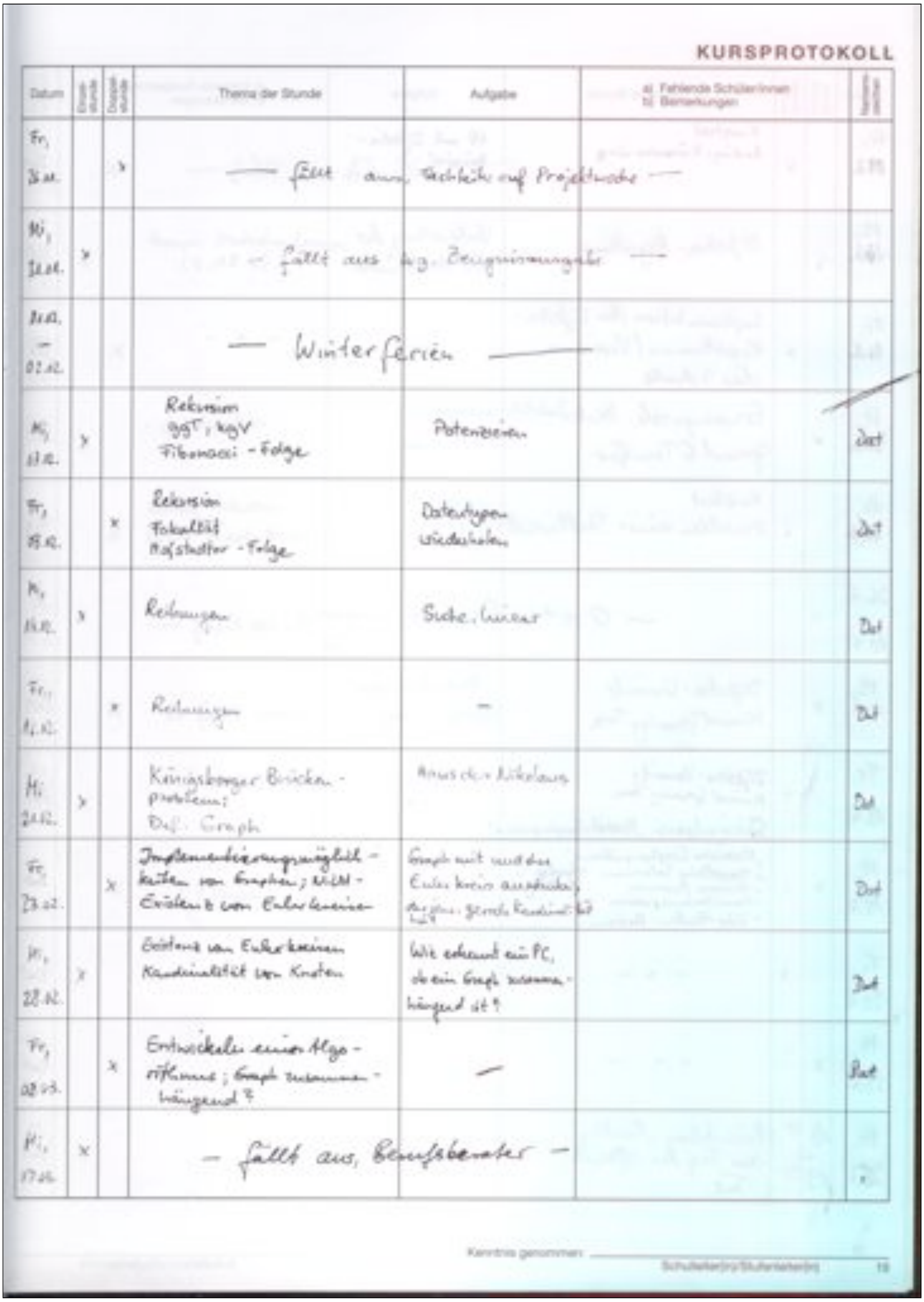
1. Intradomain-Routing kommt in autonomen Systemen wie z. B. einem Internetprovider oder einer Universität vor. Interdomain-Routing findet zwischen autonomen Systemen statt.
   a) Gib sinnvolle Übersetzungen für die Worte „Intradomain“ und „Interdomain“ an. Achtung: „domain“ wird nicht wie „Domäne“ wie z. B. in www.caro-os.de verwendet.
   b) Was könnten Gemeinsamkeiten und Unterschiede zwischen Intra- und Interdomain-Routing sein?
   c) Der Befehl traceroute vollzieht die Route eines Pakets zu einem bestimmten Rechner nach und misst die Entfernung (in ms) zu den jeweiligen Knoten. Das Folgende ist eine beispielhafte Ausgabe des Befehls:

```
Computer:~ benutzer$ traceroute www.uni-osnabrueck.de
traceroute to www.uni-osnabrueck.de (131.173.244.4), 64 hops max, 40 byte packets
 1  iserv.caro-os.de (10.0.0.222)  39.497 ms  20.310 ms  1.284 ms
 2  * * *
 3  217.0.71.170 (217.0.71.170)  95.724 ms  80.312 ms  48.952 ms
 4  h-eb1.h.de.net.dtag.de (62.154.49.170)  49.906 ms  49.423 ms  49.529 ms
 5  cr-hannover1-po1-0.x-win.dfn.de (188.1.62.1)  65.649 ms  66.068 ms  65.324 ms
 6  xr-han1-ge8-24.x-win.dfn.de (188.1.145.253)  68.514 ms  68.472 ms  109.201 ms
 7  xr-bie1-te2-1.x-win.dfn.de (188.1.145.10)  76.716 ms  67.895 ms  98.189 ms
 8  kr-uni-osnabrueck.x-win.dfn.de (188.1.231.90)  90.861 ms  69.105 ms  69.695 ms
 9  * * *
10  virt-www.serv.uni-osnabrueck.de (131.173.244.4)  83.628 ms 69.942 ms 70.840 ms
```

   Welche Knotenpunkte könnten wohl zu je einem autonomen System gehören?

2. Es gibt viele Möglichkeiten des Routings, selbst diese Liste ist nicht vollständig: manuelle Konfiguration eines Routers (jeder Router wird manuell konfiguriert), zentralisiertes Routing (eine Zentrale konfiguriert die Router), isoliertes Routing (der Router ist auf sich alleine gestellt und schickt das Paket z. B. einfach an alle seine Nachbarn), verteiltes adaptives Routing (der Router versucht selber herauszufinden, welche Netze er erreichen kann, indem er z. B. regelmäßig seine Nachbarrouter fragt), Hierarchisches Routing (der Router verwaltet nur einen kleinen Ausschnitt, den er selber kennt, alle anderen Pakete werden an einen höheren Router weitergeleitet).
   a) Beschreibe Vor- und Nachteile der verschiedenen Routing-Möglichkeiten.
   b) Bei welcher/n Möglichkeit(en) könnte der Dijkstra-Algorithmus sinnvoll angewendet werden?
   c) Ordne die Routing-Möglichkeiten dem Intra- und Interdomain-Routing zu. Begründe Deine Entscheidung und gib jeweils ein Beispiel an, wo diese Möglichkeit sinnvoll eingesetzt werden kann.

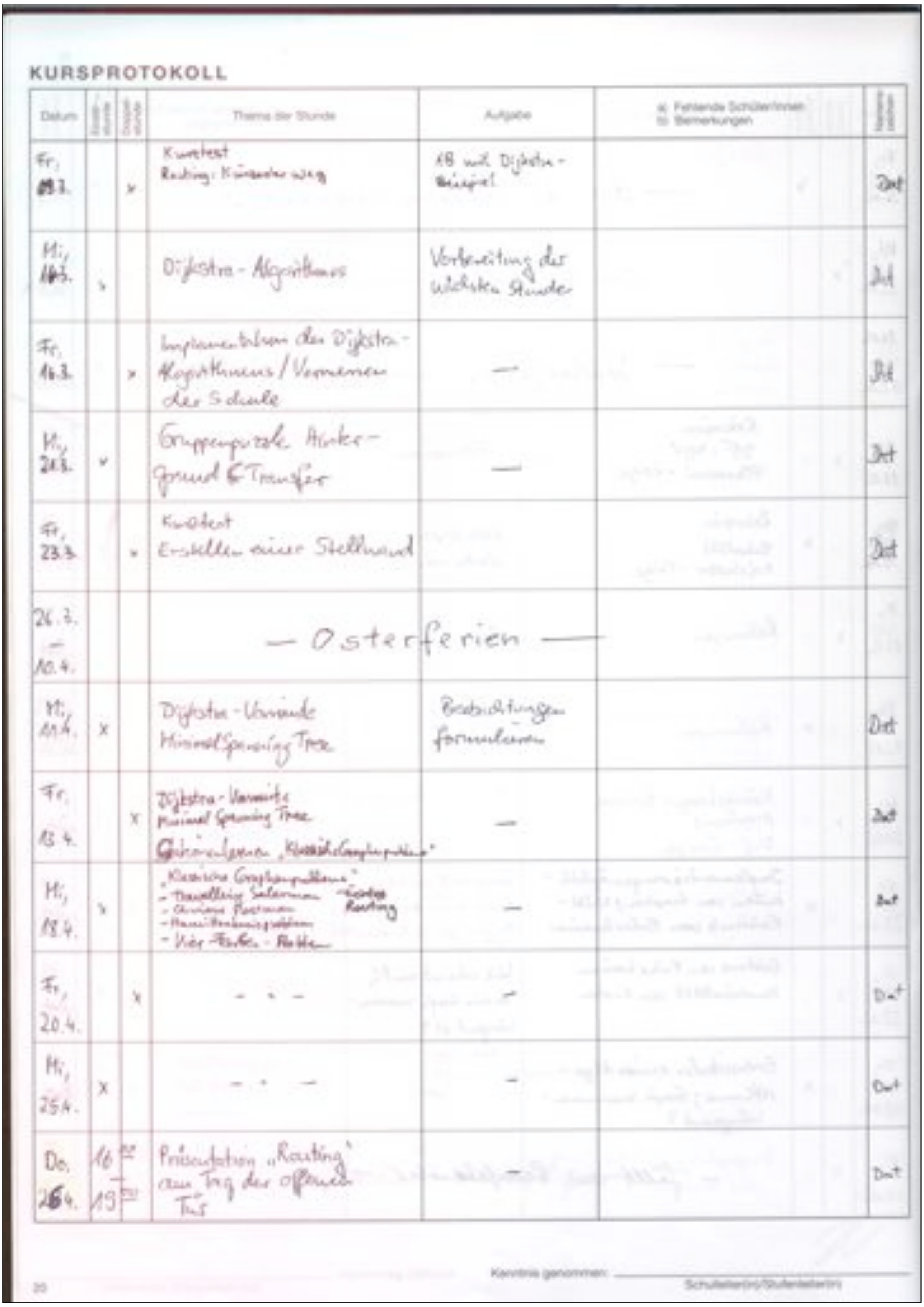
D.3 Einträge ins Kursheft



KURSPROTOKOLL

Datum	Einzelstunde	Doppelstunde	Thema der Stunde	Aufgabe	a) Fehlende Schüler/innen b) Bemerkungen	Namenszeichen
Fr, [illegible]		x	— fällt aus, Technik auf Projektwoche —			
Mi, [illegible]	x		— fällt aus wg. Zeugnisausgabe —			
[illegible] – [illegible]			— Winterferien —			
Mi, 13.02.	x		Rekursion ggT, kgV Fibonacci-Folge	Potenzieren		Dat
Fr, 15.02.		x	Rekursion Fakultät Hofstadter-Folge	Datentypen wiederholen		Dat
Mi, [illegible]	x		Reihungen	Suche, linear		Dat
Fr, [illegible]		x	Reihungen	–		Dat
Mi, [illegible]	x		Königsberger Brückenproblem; Def. Graph	Haus des Nikolaus		Dat
Fr, 23.02.		x	Implementierungsmöglichkeiten von Graphen; NLM - Existenz von Eulerkreisen	Graph mit und ohne Eulerkreis ausdrucken, [illegible] gerade Kardinalität hat?		Dat
Mi, 28.02.	x		Existenz von Eulerkreisen Kardinalität von Knoten	Wie erkennt ein PC, ob ein Graph zusammenhängend ist?		Dat
Fr, 02.03.		x	Entwickeln eines Algorithmus; Graph zusammenhängend?	–		Dat
Mi, 17.04.	x		— fällt aus, Berufsberater —			

Kenntnis genommen: ______ Schulleiter(in)/Stufenleiter(in) 19



KURSPROTOKOLL

Datum			Thema der Stunde	Aufgabe	a) Fehlende Schüler/innen b) Bemerkungen	
Fr, 9.3.		x	Kurztest Routing: Kürzester Weg	AB mit Dijkstra-Beispiel		Dnt
Mi, 14.3.	x		Dijkstra-Algorithmus	Vorbereitung der nächsten Stunde		Dnt
Fr, 16.3.		x	Implementation des Dijkstra-Algorithmus / Vernetzen der Schule	—		Dnt
Mi, 21.3.	x		Gruppenpuzzle Hintergrund & Transfer	—		Dnt
Fr, 23.3.		x	Kurztest Erstellen einer Stellwand			Dnt
26.3. – 10.4.			— Osterferien —			
Mi, 11.4.	x		Dijkstra-Umwandlung Minimal Spanning Tree	Beobachtungen formulieren		Dnt
Fr, 13.4.		x	Dijkstra-Variante Minimal Spanning Tree Gehörend zum „Klassische Graphenprobleme"	—		Dnt
Mi, 18.4.	x		„Klassische Graphenprobleme" - Travelling Salesman - Chinese Postman - Hamiltonkreisproblem - Vier-Farben-Problem Routing	—		Dnt
Fr, 20.4.		x	– – –	–		Dnt
Mi, 25.4.	x		– – –	–		Dnt
Do, 26.4.	16[00] – 19[00]		Präsentation „Routing" am Tag der offenen Tür	–		Dnt

20 Kenntnis genommen: Schulleiter(in)/Studienleiter(in)

E Versicherung

Ich versichere, dass ich die Arbeit selbstständig verfasst und bei ihrer Anfertigung keine anderen als die angegebenen Hilfsmittel benutzt habe.

Ort, Datum, Unterschrift