TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Informatik - Professur Didaktik der Informatik

GRAPHBENCH UND SEIN EINSATZ IN DER SEKUNDARSTUFE II

DIDAKTIK AUSGEWÄHLTER ASPEKTE

Ulf Mögling

30. September 2019

Dozent: Dr. Thiemo Leonhardt

INHALTSVERZEICHNIS

1 EINLEITUNG

Das Werkzeug Graphbench ist ein an der Eidgenössischen Technischen Hochschule Zürich von unter anderem Markus Brändle entwickeltes Programm zur Visualisierung von Algorithmen der theoretischen Informatik. Es behandelt NP-vollständige Probleme der Komplexitätstheorie, das heißt, eine Lösung kann durch eine Turingmaschine in polynomialer Zeit überprüft werden. Integriert sind verschiedene Probleme, unterschiedliche Lösungsalgorithmen und eine Java-Toolbox, um eigene Algorithmen zu schreiben und zu testen.
In dieser Arbeit wird behandelt, ob ein Einsatz in der Sekundarstufe II sinnvoll ist und wie dieser aussehen könnte. Es werden Unterrichtseinheiten vorgestellt, mit denen die Schüler mit Hilfe des Werkzeuges solche Probleme und Lösungsalgorithmen sowie Laufzeiten, Extremfälle, Korrektheit und reale Anwendung untersuchen.

1.1 EINORDNUNG

Die Recherche von vorhandenen Unterrichtsmaterialien zur Vermittlung NP-vollständiger Algorithmen wird sehr schnell ausgebremst. Vorhandene Materialien richten sich in fast allen Fällen an Studenten und sind damit für die Sekundarstufe II nicht geeignet. Die Präsentationen und Videos sind oft auf spezielle Vorlesungsinhalte abgestimmt und damit nicht im Rahmen eines Schuljahres vermittelbar. Einige im Gespräch mit dem Fachberater für Informatik ausgemachten Medien sollen im folgenden Abschnitt erwähnt und kurz analysiert werden.
An der Hochschule Zittau/Görlitz wurde im Fachbereich Informatik die Lern- und Arbeitsumgebung AtoCC entwickelt[11]. Diese soll motivierende praxisrelevante Anwendungen der theoretischen Inhalte an insbesondere Studierende vermitteln. Durch konsequente Weiterentwicklung entstand daraus die interaktive Lernumgebung FLACI[12].

Das Akronym steht für "Formale Sprachen, abstrakte Automaten, Compiler und Interpreter". Dabei können Grundkenntnisse der theoretischen Informatik vermittelt, formale Sprachen evaluiert, Automaten konstruiert und simuliert werden. Die Simulation von Zustandsautomaten und die Erzeugung von Zustandsgraphen kann in Bezug auf die Untersuchung von Laufzeitklassen eine Rolle spielen. Zumindest die Anwendung von Mealy- und Moore oder Turing-Maschinen kann einen Ausblick in unterschiedliche Problemklassen ermöglichen. Zustandsgraphen sind in Verbindung mit den Automaten vermittelbar oder bekannt aus der Programmierumgebung Kara. Der Übergang von Automaten zu Algorithmen ist nicht darstellbar und damit die Simulation NP-vollständiger Probleme problematisch. Zumindest die Visualisierung der abstrakten Theorie der Automaten ist ein positiver Aspekt dieses Werkzeuges, für die Untersuchung der Berechenbarkeit ist es nur umständlich zu gebrauchen.
Bei der Programmierung endlicher Automaten in der Schule ist der Marienkäfer Kara ein Thema[7]. Dieser basiert auf dem Konzept endlicher Automaten, ist alltagsnah und bietet spielerische Zugänge zu grundlegenden Programmierkonzepten. Die Lehrbücher des Duden-Paetec-Verlages beinhalten verschiedene Aufgaben zum Marienkäfer[3][2]. Durch Entwicklung verschiedener Umgebungen, z.B. Kara-Python, Kara-Turing usw. sind Übergänge zu realen Programmiersprachen und Algorithmen leicht umsetzbar. Aufgaben zu Fragen der Berechenbarkeit sind zumindest für endliche Funktionen erstellt und Präsentationen wie "der fleißige Biber" mit 4 Zuständen verfügbar. Fleißige Biber sind Turing Maschinen, die Markierungen auf einem Band hinterlassen, bevor sie anhalten. Ein Biber mit 4 Zuständen hinterlässt 13 Markierungen, ein Biber mit 5 Zuständen 4098 und bei 6 Zuständen sind es bereits $1,29 * 10^{865}$ Markierungen[7]. Leider sind diese Demonstrationen nicht besonders geeignet, den Unterschied zwischen P- und NP-Problemen zu verbildlichen. Wurden Algorithmen in der Sekundarstufe I bereits mit Kara erstellt und deren Grundstrukturen kennengelernt, ist ein fortgesetzter Einsatz durchaus sinnvoll und berechtigt. Laufzeiten sind untersuchbar, Lehrmaterial ist dazu leider nur begrenzt vorhanden.
Die Bergische Universität Wuppertal hat bereits im Jahr 2000 begonnen, eine Online-Sammlung von verschiedenen Modulen zu mathematischen und informatischen Inhalten zu erstellen[5]. Zu finden sind unter anderem Aufgaben und Informationen zu Sortierverfahren, Vierfarbenproblem, Graphenproblem, Backtracking und Turingmaschine. Aufgrund der schon in die Jahre gekommenen Java-Scripte und der Optimierung für Internet-Explorer 4.0 sind Teile der Module nicht mehr funktionsfähig, so dass diese nicht mehr in vollem Umfang genutzt werden können. Trotzdem ist diese Seite eine Quelle für Motivation und Inspiration. Mit ein wenig Aufwand können Präsentationen und Aufgaben in den eigenen Unterricht integriert werden. Keine der Simulatoren konnte getestet werden, so dass keine Aussage bezüglich der tatsächlichen Nutzbarkeit getroffen werden kann.

Veranschaulichungen von Algorithmen in Graphen sind auch von der Technischen Universität München zur Verfügung gestellt worden. Unter anderem wurden der Dijkstra-Algorithmus und der Algorithmus von Ford und Fulkerson mit Pseudocode, Beschreibung und Forschungsaufgaben auf universitärem Niveau implementiert [4]. Es können sowohl eigene, als auch zufällige Graphen erstellt und der Ablauf der Algorithmen beobachtet werden. Es sind Beispiele für die Anwendung zu entdecken und anhand des Pseudocodes bekommen interessierte Schülerinnen und Schüler einen Einblick in die Implementierung. Da sich diese Darstellungen an Studenten richten, sind die Forschungsaufgaben nicht für den Unterricht verwendbar. Die Beschreibungen und der Pseudocode sind verständlich und laden zum Ausprobieren und Entdecken ein. Vom Aufbau erinnern die Graphen stark an das Werkzeug Graphbench. Schwierig bleibt die Verwendung, falls Graphentheorie im Unterricht keine Rolle spielt.
Sortieralgorithmen, das Problem des kürzesten Weges und das Rucksackproblem wurden auf einem schülergerechten Niveau von Jens Gallenbacher im Buch Abenteuer Informatik aufgearbeitet [6]. Anhand von Kopiervorlagen, Karten- und Puzzlespielen werden diese ausführlich erläutert, auf grundlegende Fragen der Informatik eingegangen, Anwendungen und Beispiele bebildert sowie Aufgaben und Lösungen dargestellt. Es werden verschiedene didaktisch aufgearbeitete Methoden vorgegeben, Flussdiagramme zur späteren Algorithmisierung angegeben, auf Aufwands- und Laufzeitabschätzungen eingegangen und der Unterschied zwischen P und NP verständlich erläutert. Das Buch ist eine Fundgrube haptischer Herangehensweisen an schwierige informatische Sachverhalte- zu großen Teilen ohne Nutzung von Software. Auch Grenzen der Berechenbarkeit sind neben anderen informatischen Inhalten abgebildet. Vom Problem des kürzesten Weges abgesehen ist dabei keine zusätzliche Einarbeitung notwendig. Die Vorlagen ermöglichen einen direkten Einsatz im Unterricht und die Beschreibungen sind zum Nachlesen geeignet.

2 MOTIVATION

Ziel des Informatikunterrichts in der Sekundarstufe I ist eine zeitgemäße und fachlich substanzielle informatische Bildung zu befördern. Darauf aufbauend werden in der Sekundarstufe II die Struktur der Prozess- und Inhaltsbereiche aus den Standards der Unterstufe übernommen und um die Anforderungsbereiche der einheitlichen Prüfungsanforderungen in der Abiturprüfung Informatik ergänzt. Dazu zählt auch die Analyse von praktischen und theoretischen Grenzen der Algorithmisierung.
Um informatische Probleme zu visualisieren wurde in vielen Bereichen Lernsoftware entwickelt. Programme zur Veranschaulichung mathematisch abstrakter Themen sind allerdings sehr rar. Insbesondere NP-Vollständigkeit ist unterrepräsentiert. Graphbench füllt in diesem Bereich eine Lücke und besitzt allein deshalb seine Daseinsberechtigung. Bei der Entwicklung der Lernumgebung war ein Ziel die "intuitive Einführung in ein sonst abstraktes und komplexes Thema. Es wurden didaktische und computergestützte Ansätze verbunden."[1]
Graphbench ermöglicht die Untersuchung von Problemen der Graphentheorie: Graphfärbbarkeit, Vertex Cover, Clique, Traveling Salesman, Erfüllbarkeit logischer Formeln und ähnlicher Probleme. Zunächst "müssen die Lernenden die Problemstellungen als solche verstehen. Mit der Lernumgebung Graphbench können sie beliebige Probleminstanzen erkunden und erhalten ein intuitives Gefühl für die Beschaffenheit der Probleme. Anschließend können sie verschiedene Lösungsalgorithmen durch Beobachten des animierten Ablaufes entdecken."[14] Durch Änderung der Ausgangskonfiguration können Effizienz und Komplexität der Algorithmen beurteilt und verglichen werden.
Dem Lehrer bieten sich in Graphbench einige Möglichkeiten, die Erfahrungen der Schülerinnen und Schüler zu lenken. Es kann eine bestimmte Problemstellung exakter untersucht, alle integrierten Probleme kennengelernt, der Fokus auf algorithmische Aspekte gelegt, oder die Berechenbarkeit erfasst werden. Ob nur oberflächlich oder tiefer in die Materie eingetaucht und Algorithmen näher untersucht oder sogar selbst entwickelt werden, bleibt der Lehrperson überlassen.

Das Werkzeug lädt zum "Erkunden verschiedener Graphenprobleme, zum Erzeugen von Probleminstanzen sowie zum Betrachten der animierten Lösungsalgorithmen und Problemreduktionen"[14] ein. Die Voraussetzungen für den Einsatz zur Vermittlung NP-vollständiger Probleme in der Sekundarstufe II sind also durchweg gegeben.

3 REIHENPLANUNG

Im Inhaltsbereich Algorithmen der Sekundarstufe II der Bildungsstandards der Gesellschaft für Informatik sind Analyse, Vergleich und Beurteilung von Algorithmen verortet. In Beispielaufgaben wird immerhin auf Zustandsgraphen eingegangen. Die Nutzung von Graphen oder Untersuchung von Graphenproblemen ist nicht zu finden.
Laut dem sächsischen Lehrplan Informatik am Gymnasium sollen in der Kursstufe im Lernbereich fünf Beispielalgorithmen bezüglich ihrer Effizienz und Komplexität beurteilt werden. Dazu wird zwischen Algorithmen mit polynomialen und exponentiellem Aufwand unterschieden. Weiterhin soll die Zeitkomplexität experimentell ermittelt und theoretisch nachgewiesen werden, sowie Grenzen der Berechenbarkeit kennengelernt werden. Im Lernbereich sieben wurden die Wissenschaftsbereiche der Informatik angesiedelt und dabei in der theoretischen Informatik auf Sprachen und Automaten sowie Probleme der Berechenbarkeit hingewiesen. Graphentheorie lässt sich dabei weder in den Lern- noch den Wahlbereichen finden. Das Werkzeug Graphbench bezieht sich explizit auf Anwendung in Graphenproblemen- ohne Grundlagen zu schaffen bleiben die visualisierten Lösungsalgorithmen also im luftleeren Raum hängen. Im realen Alltag einer Lehrperson ist es also zunächst die Aufgabe, Raum zur Vermittlung der Graphentheorie zu finden. Im Folgenden wird davon ausgegangen, dass dieser Raum gefunden wurde und die Unterrichtsplanung daran orientiert.
Im Rahmen der Belegarbeit sollen bis zu fünf Doppelstunden ausführlich vorgestellt werden. Ausgehend vom Lernbereich 5: Algorithmen und den bei Schülerinnen und Schülern vorhandenen Vorwissen aus der Sekundarstufe I wird angenommen, dass bereits verschiedene Sortieralgorithmen von Schülergruppen in einer Programmierumgebung umgesetzt wurden. Der sich unten anschließende tabellarische Stoffeinheitenplan bezieht sich auf die Effizienz von Algorithmen, die Einführung von Graphen und das Nutzen von Graphbench.

	Thema	Inhalt
1	Laufzeit von Sortieralgorithmen	Experimentelles Ermitteln, theoretischer Nachweis
2	Einführung Graphentheorie	Beispiel: Navigation, Knoten, Kanten, Richtung, Gewichtung
3	Ein Lösungsalgorithmus: Dijkstra	Lösung auf dem Papier, Erstellen von Struktogramm
4	**Nutzung Graphbench**	**Schrittweise Einführung in die Nutzung, Erstellung eigener und vorgegebener Graphen**
5	Übungsstunde, Zusammenfassung	Lösung von Anwendungsaufgaben, Laufzeituntersuchung, Erläuterung NP-vollständig

Diese fünf Doppelstunden werden im folgenden genauer betrachtet. Damit sind 10 von 18 Stunden des Lernbereiches abgedeckt. Die vierte und fünfte Doppelstunde werden in Kapitel 4 ausführlich vorgestellt.

3.1 LAUFZEIT VON SORTIERALGORITHMEN

Die Schülerinnen und Schüler messen in dieser Unterrichtsstunde die Laufzeit bereits implementierter Sortieralgorithmen für vorgegebene unsortierte Listen unterschiedlicher Länge. In Gruppen werden die Listenlängen den Laufzeiten der verschiedenen Algorithmen in einem Diagramm gegenübergestellt. Die Lernenden erkennen unterschiedliche Laufzeiten und stellen eine Vermutung über die Abhängigkeit von der Problemgröße auf. Zuletzt notieren Sie den theoretischen Nachweis anhand zweier Beispiele.
Die Doppelstunde wird mit einem Video zu Selection-Sort eingeleitet. Da die Schülerinnen und Schüler bereits einen Algorithmus implementiert haben, genügen wenige Erläuterungen um dieses Video auf Bekanntes zu übertragen. Als nächstes sollen in einem Wettbewerb die implementierten Algorithmen gegeneinander antreten. Dazu wird mit einer Stoppuhr die Laufzeit für die Sortierung unterschiedlich langer Listen gemessen und zunächst tabellarisch und anschließend grafisch festgehalten. In heterogenen Klassen bietet sich an, herausragenden Schülern die Aufgabe zu stellen, eigene Algorithmen zu entwerfen. Diese müssen dann gegen die Bekannten antreten. Im Unterrichtsgespräch werden Vermutungen aufgestellt, warum sich diese Laufzeiten ergeben, wovon sie abhängig sind und es wird darauf eingegangen, dass es worst- und best-case-Szenarien gibt. Der Aufwand wird als Funktion in Abhängigkeit von der Problemgröße eingeführt und der theoretische Nachweis und damit der exakte Aufwand

für Selection- und Bubble-Sort im Lehrervortrag hergeleitet. Zuletzt wird darauf verwiesen, dass es sich dabei um polynomiale Laufzeiten handelt und dass es auch andere gibt. Dies dient als Ausblick auf die nächsten Unterrichtsstunden.

3.2 EINFÜHRUNG GRAPHENTHEORIE

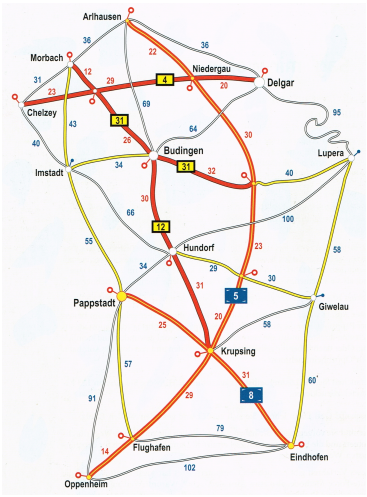


Abbildung 3.1: Straßenplan [6]

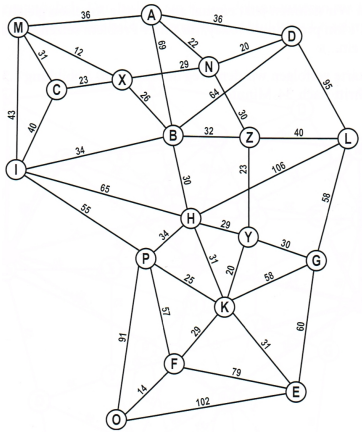


Abbildung 3.2: Graph zum Plan [6]

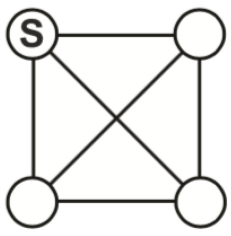


Abbildung 3.3: Vollst.Graph mit 4 Knoten [6]

Die Schülerinnen und Schüler abstrahieren von einem vorgegebenen Straßenplan die wichtigen Elemente und entdecken dadurch das Modell eines Graphen. Sie ordnen Knoten, Kanten und deren Wichtung den Elementen des Planes zu und erkennen, dass Einbahnstraßen in Graphen in Form von Pfeilen darstellbar sind und damit zu gerichteten Graphen führen. Die Lernenden können vollständige und unvollständige Graphen unterscheiden und die Anzahl möglicher Wege von einem Startpunkt aus bestimmen. Sie erkennen, dass das Vergleichen aller möglichen Wege in einem vollständigen Graphen mit extrem hohem Aufwand verbunden ist.

Diese Doppelstunde wird mit dem Alltagsbezug auf einen Routenplaner eingeleitet. Ein Programm benötigt zum Finden des kürzesten Weges nicht alle Informationen, dass heißt eine Landkarte muss zunächst vereinfacht und die wichtigen Daten heraus gefiltert werden. Angelehnt an das erste Kapitel [6] entscheiden die Schülerinnen und Schüler in Partnerarbeit, ob Name, Position und Größe der Städte, Verlauf, Länge, Typ, Start und Ziel einer Straße und sonstige landschaftliche Informationen wichtig sind. Eine erstes Modell wird von der Lehrperson vorgestellt. Im Unterrichtsgespräch werden weitere Verbesserungen vorgenommen, bis ein akzeptabler Graph entstanden ist. Diese entdeckende Einführung von Graphen wird als nächste mit der Theorie unterlegt. Städte und Kreuzungen sind Knoten, Straßen heißen Kanten. Weitere Eigenschaften von Graphen werden theoretisch vorgestellt und mit Beispielen versehen. Im schlimmsten Fall sind alle Knoten mit allen anderen durch Kanten verbunden- es entsteht ein vollständiger Graph. Die Schülerinnen und Schüler bestimmen nach Anleitung die Anzahl der möglichen Wege in einem

vollständigen Graphen mit 3 bis 7 Knoten. Sie erfahren, dass die Anzahl der Wege exponentiell steigt und erkennen, dass das Vergleichen aller Wege zur Ermittlung des Kürzesten keine sinnvolle Lösungsstrategie darstellt.

3.3 EIN LÖSUNGSALGORITHMUS: DIJKSTRA

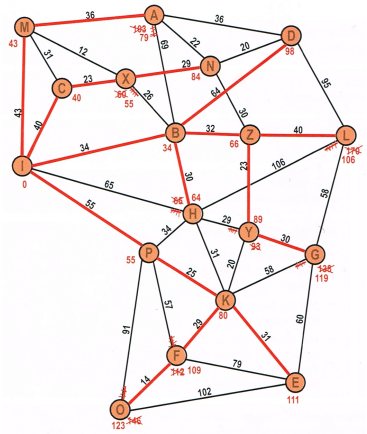


Abbildung 3.4: Startstadt Imstadt [6]

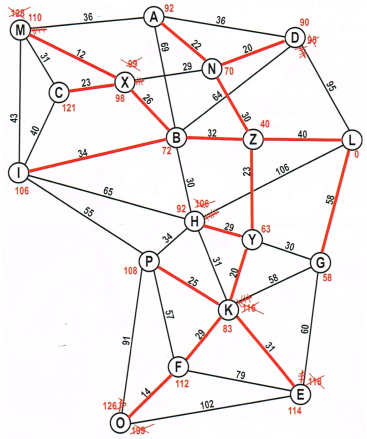


Abbildung 3.5: Startstadt Lupera [6]

Die Schülerinnen und Schüler entdecken den Algorithmus von Dijkstra und nutzen ihn zum Lösen des kürzesten Wege-Problems in einem Graphen. Sie bearbeiten einen Graphen auf Papier und üben die Umsetzung an weiteren Aufgaben. Die Schülerinnen und Schüler notieren ein Struktogramm für den Dijkstra-Algorithmus und vergleichen den Aufwand mit einem Brute-Force Algorithmus.

Mit dem in der vorherigen Stunde erarbeitetem Graphen und dazugehörigen Kopiervorlagen soll zunächst der kürzeste Weg von Imstadt nach Oppenheim gefunden werden. Das verwendete Prinzip ist die Bewegung von Ameisentrupps von der Anfangsstadt in alle benachbarten Städte, das Merken der zurückgelegten Strecke, die Aufteilung des Trupps und wieder die Bewegung in alle benachbarten Städte. Bis das Ziel erreicht ist oder der Algorithmus abbricht, wird ein Weg gestrichen, falls sich zwei Ameisentrupps treffen. Eine computerbasierte Umsetzung geht nicht von Ameisen aus, sondern liest die kürzeste Strecke ein und geht von dieser Stadt als neuer Anfangsstadt aus. In jeder Stadt wird mit dem bisherigen kürzesten Weg verglichen und der aktuell kürzeste Weg gemerkt. Schrittweise wird durch den Lehrer der Algorithmus von Dijkstra eingeführt. Sind die ersten drei Städte im Unterrichtsgespräch erarbeitet, führen die Schülerinnen und Schüler den Versuch in Partnerarbeit fort, bis sie zu einer Lösung gelangen. Diese wird verglichen und danach in Einzelarbeit der kürzeste Weg von Lupera nach Eindhofen, Giwelau und Morbach gesucht. Nach dieser Übung wird das Struktogramm für den Algorithmus von Dijkstra zusammen getragen und notiert. Natürlich kann anstatt der Länge der Strecke auch die geringste benötigte Zeit für eine Strecke betrachtet werden. In der letzten Stunde wurden alle möglichen Wege in einem vollständigen Graphen mit sieben Knoten notiert. Darauf Bezug nehmend demonstriert die Lehrperson wie viele Wege bei der

Nutzung des Dijkstra-Algorithmus verglichen werden müssen. Diese Wegeanzahl wird als Aufwand notiert. Die Schülerinnen und Schüler erfahren, dass der Algorithmus von Dijkstra zur Klasse der P-Probleme gehört, aber sehr viel weniger Wege vergleicht als der Brute-Force Algorithmus.

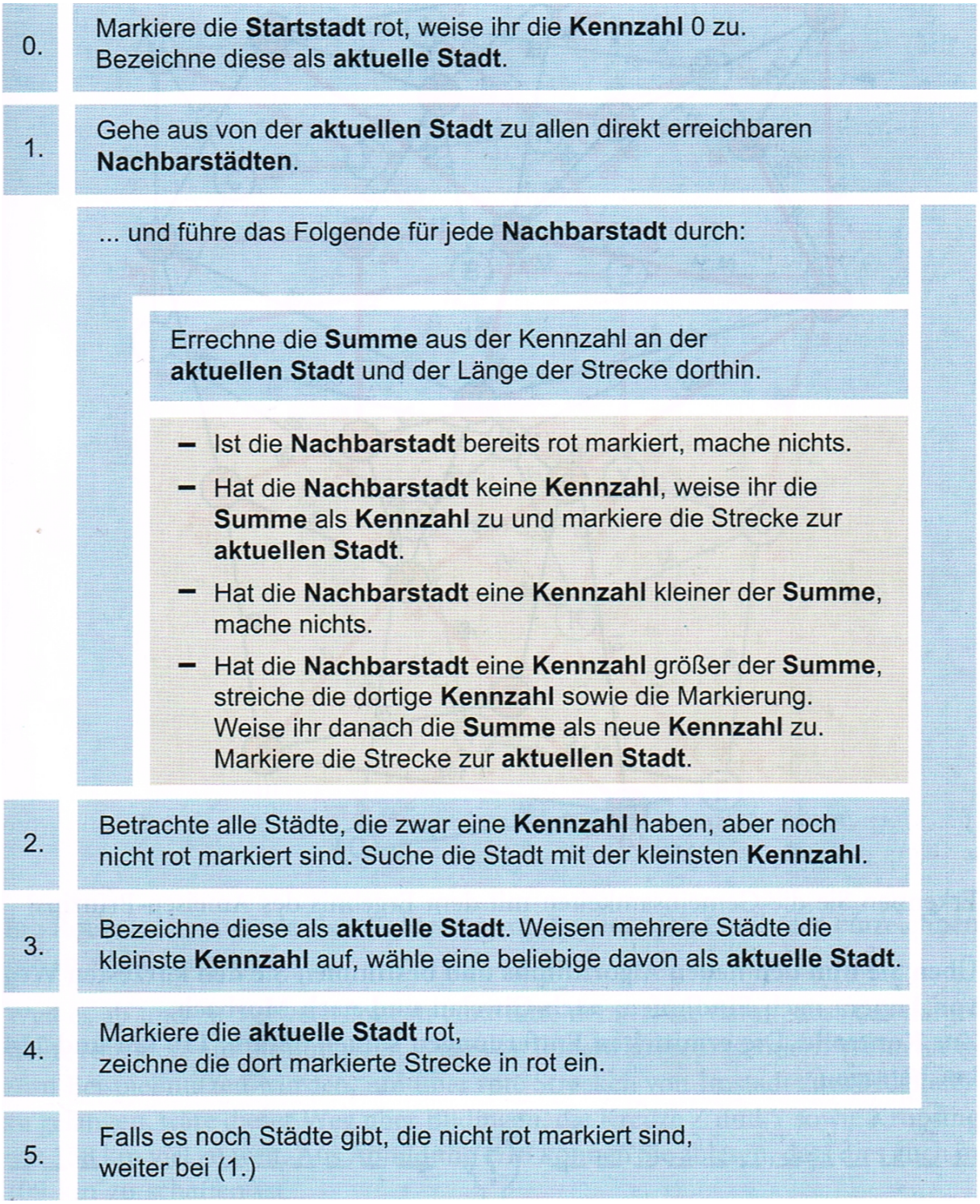


Abbildung 3.6: Struktogramm [6]

4 LEHR-LERN-MATERIAL

Abbildung 4.1: Innenstadt Bern [7]

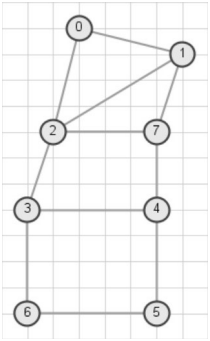


Abbildung 4.2: Graph mit 8 Knoten

Von der Aufgabenstellung, Material für eine Doppelstunde zur Verfügung zu stellen, wird im folgenden Kapitel abgewichen. Aufgrund des Umfanges des Werkzeuges Graphbench ist eine Doppelstunde zu wenig, um mehr als eine Aufgabenstellung zu bearbeiten. Aus den bisher gehaltenen Stunden kennen die Schülerinnen und Schüler Graphen und wissen, wie ein Straßenplan auf einen Graph übertragen werden kann. Diese Voraussetzungen erleichtern das Nutzen des Werkzeuges.

In der ersten Doppelstunde erstellen die Schülerinnen und Schüler im Programm einen Graphen als Abbild einer Straßenkarte. Sie ermitteln experimentell die Zeit, die der Backtracking-Algorithmus für das Finden eines Hamilton-Kreises mit acht, zehn und zwölf Knoten benötigt und übertragen das Problem auf eine Anwendungsaufgabe.

Zunächst werden die Merkmale eines Graphen wiederholt. Im Anschluss wird die Aufgabe vorgestellt: Nicht immer muss der kürzeste Weg zwischen zwei Knoten gefunden werden. In der Innenstadt von Bern soll ein Kinderumzug stattfinden und an jeder Kreuzung genau einmal vorbeiziehen. Die Aufgabe soll diesmal mit dem Werkzeug Graphbench gelöst werden. Zunächst wird die Oberfläche im Lehrervortrag erläutert und ein gemeinsamer Graph mit 8 Knoten in Schritt für Schritt Anleitung angefertigt. Es wird zusätzlich vorgeführt, wie die Anzahl der Berechnungen im Pseudo-Code festgestellt werden kann und wie zusätzliche Informationen zu den Algorithmen zu finden sind. Die Schülerinnen und Schüler ermitteln anschließend die benötigte Zeit und die Anzahl der Durchläufe im Pseudoalgorithmus. Anschließend erweitern sie den Graphen

auf 10 und 12 Knoten und wiederholen die Messung. Danach erzeugen sie eigene Graphen mit 10 Knoten und messen noch einmal Zeit und Anzahl der Durchläufe. Der Schüler der den Graphen mit den meisten Durchläufen erzeugt hat, stellt seinen Graph vor und bekommt einen Preis - z.B. eine Packung Gummibären. Im Unterrichtsgespräch wird analysiert, warum dieser Graph die höchste Schwierigkeit aufweist und andere aufgetretene Lösungen diskutiert.

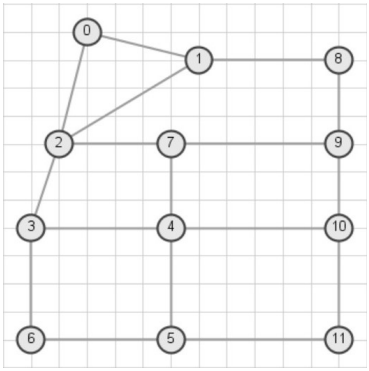


Abbildung 4.3: Graph mit 12 Knoten

In den letzten Minuten der Stunde wird eine weitere Anwendungsaufgabe vorgestellt: Ein Werbezeppelin fliegt über die wichtigsten Plätze und Sehenswürdigkeiten Berns, die Runde soll dabei so kurz wie möglich ausfallen. Es wird besprochen, welcher Unterschied zur ersten Aufgabe besteht- die Straßen spielen keine Rolle sondern nur der Abstand per Luftlinie. Die Runde soll außerdem möglichst kurz sein. Mit der Erläuterung, dass diese Aufgabe einem weiteren Problem - dem Traveling Salesman Problem - entspricht werden die Schülerinnen und Schüler in die nächste Stunde entlassen.

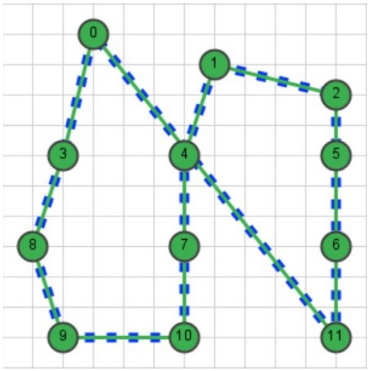


Abbildung 4.4: Lösung TSP mit Greedy: Länge 1219

Zu Beginn der letzten Doppelstunde dieser Einheit bietet sich die Zusammenfassung der bis hierher bekannten Komplexitätsklassen an. Sortieralgorithmen und Dijkstra sind polynomial, der Hamiltonkreis ist komplexer und geht darüber hinaus. Er gehört, wie das nun zu besprechende TSP, zur Klasse der NP-vollständigen Probleme. Die Schülerinnen und Schüler implementieren den bekannten Stadtplan in Graphbench, sie lösen das Traveling Salesman Problem mit unterschiedlichen Algorithmen, ermitteln experimentell die benötigte Zeit und überprüfen die Lösung auf Optimalität. Sie notieren die theoretische Zusammenfassung der Komplexitätsklassen, üben die Bestimmung des Aufwandes an einem Beispiel und diskutieren, ob P=NP gilt.
Die Schülerinnen und Schüler wenden sich nun selbstständig der Aufgabe zu, die Effizienz der verschiedenen in Graphbench implementierten Algorithmen zur Lösung des Rundreiseproblems zu untersuchen. Sie ermitteln die Zeit und die Anzahl der benötigten Durchläufe für die Beispielaufgabe experimentell. Dabei ist es wiederholt notwendig, darauf hinzuweisen, dass in unserer speziellen Anwendung nur vollständige Graphen betrachtet werden können und das Richtung und Wichtung der Kanten keine Rolle spielen. Als Zusatzaufgabe für starke Schülerinnen und Schüler ist eine Erweiterung des Graphen, die Wichtung der Kanten und die erneute Untersuchung der Effi-

zienz denkbar. Die Algorithmen sollen anschließend nach der besten Lösung und der benötigten Zeit geordnet werden. Dabei liegt der Schwerpunkt nicht im Kennen der Algorithmen. Ein tieferer Einblick in deren Ablauf ist nicht vorgesehen.

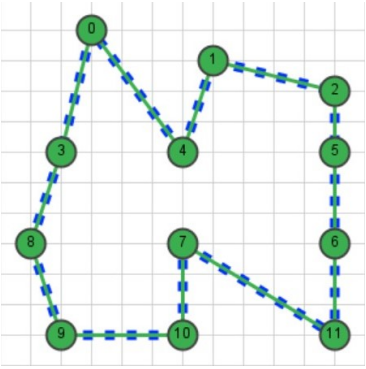


Abbildung 4.5: Lösung mit Convex Hull: Länge 1089

Die Betrachtung des Graphenfärbungsproblems und einer entsprechenden Anwendungsaufgabe ist eine weitere mögliche Zusatzaufgabe. Ein alle Aufgaben umfassendes Arbeitblatt ist im Anhang (7) zu finden. Die theoretische Zusammenfassung beinhaltet den Effizienzbegriff: Ein Problem wird mit möglichst geringem Ressourcenverbrauch gelöst. Es gibt linearen, quadratischen, kubischen, polynomialen oder exponentiellen Aufwand. Alle Probleme die in polynomialer Zeit gelöst werden können, gehören zur Klasse P. Alle Probleme, deren Laufzeit sich nicht durch ein Polynom ausdrücken lässt gehören zur Klasse NP. Die Lösung eines solchen Problems kann von einem Turing-Automaten in polynomialer (endlicher) Zeit überprüft werden. Die dabei betrachteten Algorithmen sind keine Näherungsverfahren. Darüber hinaus gibt es Probleme, die nicht berechenbar sind.

Zum Abschluss der Stoffeinheit berechnen die Schülerinnen und Schüler die Laufzeit von zwei angegebenen Algorithmen in Abhängigkeit von "n" und stellen sich unter Anleitung der schwierigen Frage, ob die Klassen P und NP gleich sind. Sie äußern Vermutungen und versuchen Beispiele aus ihrer nun bekannten Erfahrungswelt anzugeben. Damit ist das Thema der Komplexität und Berechenbarkeit ausführlich erörtert und die Stoffeinheit abgeschlossen.

5 EVALUATION

Neben anderen Graphenproblemen und noch weiteren Lösungsalgorithmen bildet Graphbench auch die Lösbarkeit von logischen Gleichungen und 3D-Matching an. Diese Themen sind noch abstrakter als Hamiltonkreis oder Traveling Salesman Problem und haben deshalb in der Sekundarstufe II keinen Anwendungsbereich.
Die in Kapitel 4 beschriebenen Stunden sind für die Schülerinnen und Schüler der Kursstufe bereits schwierig nachzuvollziehen. Sollen sie sich selbstständig in das Werkzeug einarbeiten und die Algorithmen erforschen, verlieren sie sich sehr schnell in der Kompliziertheit der Thematik. Ohne Anleitung verlören sich die Lösungsalgorithmen im luftleeren Raum, tiefere Einsichten blieben verborgen. Da die Algorithmen bis zu diesem Zeitpunkt noch unbekannt waren, werden einige unter den Heranwachsenden auch so schon überfordert sein.
Die notwendige Anleitung in ein neues Programm bringt zusätzlich einen erhöhten Zeitaufwand mit sich. Ob die abgebildeten Stunden ausreichend sein werden, um das Thema Komplexität zu erfassen, ist fragwürdig. Das Werkzeug ist nicht selbsterklärend und die Einführung notwendig.
Ein Einsatz in der Oberschule ist komplett auszuschließen, außerhalb der Kursstufe findet sich kein Anwendungsbereich.
Graphbench ist also für Schüler in den meisten Fällen schwer zu begreifen, selbst unter Anleitung und Beschreibung werden wenige Schüler hinter die Konzepte und Ideen blicken können. Dennoch ist das Entwickeln eines Gefühles für die Komplexität eines Algorithmus möglich, die Visualisierung der einzelnen Komponenten gut und im Sinne eines Demonstrationstools die Nutzung nicht sehr zeitaufwendig.
Vergleichbar ist der Einsatz mit der Demonstration von Traceroute oder dem Vorzeigen eines Videos über einen Sortieralgorithmus. Es erklärt visuell den Algorithmus, ohne das ein Begreifen nötig oder erforderlich wäre.
Einige Probleme hat die Software bei der Nutzung: Der Start mit allen Funktionen ist umständlich, laden und speichern von Graphen hakt. Mit Sicherheit lassen sich noch mehr

Schwächen finden, das Programm ist nicht aktuell, die Pflege lässt zu wünschen übrig. Dass in jedem PC-Pool die benötigte Java-Version verfügbar ist, ist kaum zu glauben. Damit verbunden ist zusätzliche Zeit, bis alle Schülerinnen und Schüler das Programm auch tatsächlich nutzen können.
Gegen den Gebrauch des Werkzeuges in einer Unterrichtseinheit sprechen die schlechte Implementierung von Branch and Bound, dass das Traveling Salesman Problem nur an vollständigen Graphen gelöst werden kann, das Graphen nicht gerichtet sein können und dass das Rucksackproblem nicht abbildbar ist. Außerdem werden die meisten Probleme am effektivsten mit dem Greedy-Algorithmus gelöst - dies setzt einen komplett falschen Anreiz.
Das aktuell größte Problem bei der Nutzung von Graphbench ist die fehlende Nische im Lehrplan. Graphenprobleme sind dort nicht zu finden, die Lernzielebene für theoretische Informatik ist "Kennen", zum Beurteilen der Effizienz von Algorithmen eine 8 Unterrichtsstunden umfassende Stoffeinheit viel zu ausführlich.

6 FAZIT

Mit den Alternativen, die am Anfang dieser Arbeit aufgezeigt wurden, ist der einzige Vorteil den Graphbench aufzuweisen hat, dass es umsonst ist. Umsonst ist auch die Zeit, die in die Vorbereitung von speziell angepassten Aufgaben investiert wurde. Kartenspiele, Puzzle und Anwendung von Algorithmen auf dem Papier gehen schneller von der Hand, sind einprägsamer und visuell anspruchsvoller. In seinem Kernelement, der Visualisierung von Graphenproblemen, ist schlampig vorgegangen worden: Bei der Modifizierung gerät man schnell an seine Grenzen und Greedy ist mit Sicherheit kein bester Löser. Der Wunsch NP-vollständige Probleme zu visualisieren und damit zugänglicher zu machen ist fantastisch, doch dazu sollte ein Programm auch eine entsprechende Zugänglichkeit besitzen. Was bleibt ist eine veraltete Software, die den Anspruch der Universität auf die Schule übertragen soll.

7 ANHANG

Von [7] wurde das folgende Aufgabenblatt zur Verfügung gestellt.

Schwierige Probleme in der Informatik

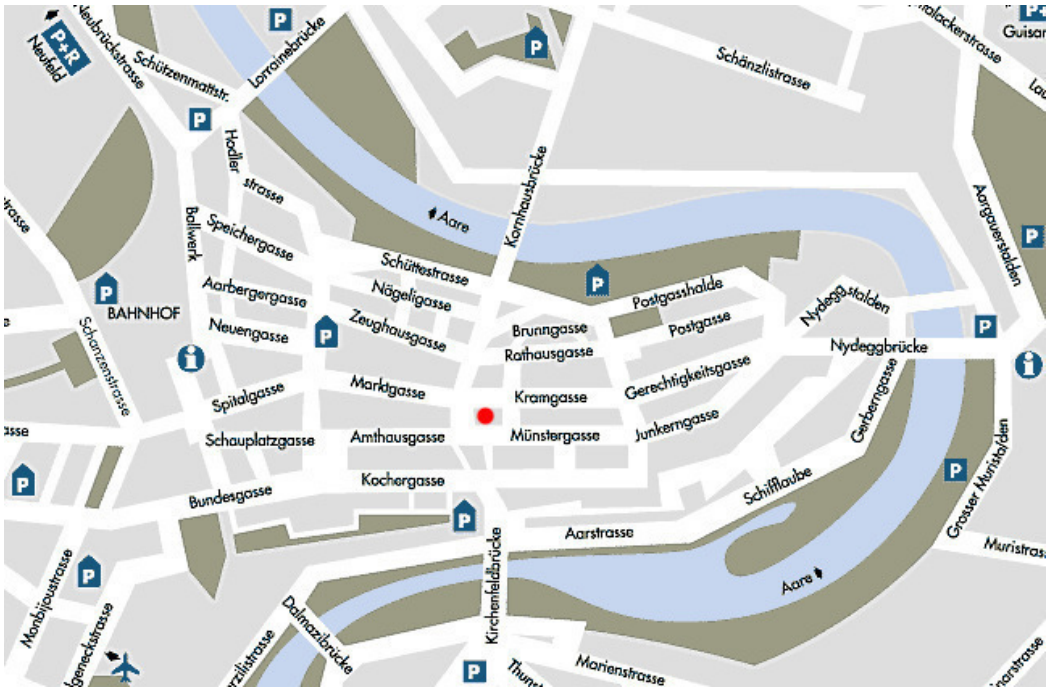
Ablauf

- Geschichte durchlesen 10 min
- Einführung in GraphBench 10 min
- Entdeckungsphase 60 min
- Diskussion mit Mitschüler
- Vorbereitung des Vortrags 30 min
- Kurzvorträge

Organisationsprobleme

Das letzte *Zürifäscht* war ein Riesenerfolg. Fast eine Million Leute strömten in die Metropole. Bern möchte, angetrieben durch diesen Erfolg, ein *Bärnfescht* organisieren.

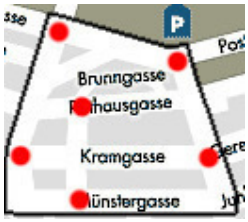
Der Stadtpräsident der Stadt Bern hat dafür ein Organisationskomitee einberufen, das Eure Klasse unterstützen soll.



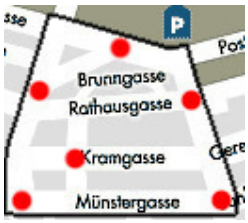
Berns Altstadt

Das *Bärnfescht* beginnt mit einem Kinderumzug durch Berns Altstadt. Der Umzug startet und endet beim *Zytglogge* (Punkt in der Mitte) und soll an jeder Kreuzung der Altstadt genau einmal vorbeiziehen. Die Route muss möglichst rasch bestimmt werden, um die Polizei zwecks Strassensperrung zu informieren.

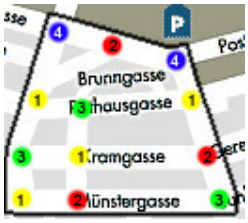
Nebst Attraktionen finden an diesem Fest viele Konzerte statt. Bühnen können aufgrund ihrer Grösse nur an Kreuzungen aufgestellt werden, sollen sich aber gegenseitig nicht beschallen. Das bedeutet, dass Bühnen nicht direkt durch eine Strasse verbunden sein dürfen. Die OK Mitglieder wissen nicht, wieviele Bühnen sie mieten müssen!



Die Polizei interessiert, wieviele Leute sie einsetzen muss. Die Polizisten werden auf Kreuzungen positioniert. Du denkst, dass es ausreicht, wenn jeder Strassenabschnitt von einem Polizisten überwacht wird. Durch Ausprobieren findest du die Lösung nach einer Stunde. Geht das effizienter?



Nebst Bühnen gibt es an den Kreuzungen auch je einen Stand (für Essens-, Getränke-, Vergnügungsstände und Toiletten). An benachbarten Kreuzungen sollen verschiedene Stände stehen. Ist das mit vier verschiedenen Ständen im Berner Strassennetz möglich?



Das Regionalfernsehen will das Fest live übertragen. Der Sender besitzt drei geeignete Kameras. Diese sind so auf Kreuzungen zu positionieren, dass jede Kamera mit den zwei anderen in Sichtkontakt steht, um die Strassen von beiden Seiten zu filmen. Welche Kreuzungen eignen sich dafür?

Ausserdem fliegt ein Werbezeppelin vor dem Fest über die wichtigsten Plätze und Sehenswürdigkeiten. Eine Runde soll so kurz wie möglich sein.

Du siehst, die Planung des *Bärnfeschts* wirft Probleme auf und ist eine spannende Angelegenheit.

All diese Fragen sind wichtige Informatikthemen und nicht einfach zu lösen. Man weiss zwar, wie man die Lösung bestimmt, je nach Grösse (Anzahl Knoten) kann es aber Tage dauern, bis die Lösung gefunden wird. Selbst mit dem schnellsten Computer!

Solche Probleme werden mit dem Programm GraphBench betrachtet. Starte das Programm und lies die folgende Einführung.

Deine Aufgabe

Mach dich mit dem Problem vertraut: Spiel mit der Umgebung und finde heraus, worum es hier geht. Du sollst die Lösungsidee einem Nicht-Informatiker erklären können. Hinweis: Auf der letzten Seite ist jedes Problem kurz beschrieben.

Überlege dir Fragen zu diesem Problem und versuche sie zu beantworten. Eine Auswahl von Fragen ist aufgeführt. Es gibt viele andere Gesichtspunkte, die du untersuchen kannst:

- Wie funktionieren die Algorithmen?
- Kann ich selbst einen Algorithmus finden, der das Problem löst?
- Wieviele Schritte braucht der Algorithmus, bis er eine Lösung findet?
- Welcher Algorithmus ist der schnellste (d.h. benötigt am wenigsten Schritte)?
- Findet das Programm immer die richtige Lösung?
- Wie siehts aus mit Extremfällen?
- Gibt es praktische Anwendungen für dieses Problem?
- Gibt es Grenzen in der praktischen Nutzung?

Formuliere deine Antworten und Erkenntnisse in kurzen, einfachen Sätzen. Eine Person, die sich nicht mit Informatik beschäftigt, sollte deine Erkenntnisse verstehen.

Nach 60 Minuten setzt du dich mit einem Mitschüler zusammen, der dasselbe Problem studiert hat. Ihr diskutiert, was ihr entdeckt habt.

Anschliessend überarbeitest du deine Erkenntnisse. Formuliere 3-5 Erkenntnisse und halte sie schriftlich fest.

Bereite einen 3-minütigen Kurzvortrag zu deinem Thema vor. Er beinhaltet die Schilderung des Problems, einen möglichen Lösungsalgorithmus und deine Erkenntnisse. Der Vortrag muss von deinen Mitschülern verstanden werden. Dafür hast du ca. 30 Minuten Zeit.

Viel Spass beim Entdecken!

LITERATURVERZEICHNIS

[1] Markus A Brändle. *GraphBench: Exploring the Limits of Complexity with Educational Software: Exploring the limits of complexity with educational software*. PhD thesis, ETH Zurich, 2006.

[2] L.Engelmann et al. *Informatik SII, Gymnasiale Oberstufe*. DUDEN PAETEC GmbH, Berlin, 2006.

[3] L.Engelmann et al. *Informatik SI, Informatische Grundbildungl*. Cornelson Verlag GmbH, Berlin, 2017.

[4] P.Gritzmann et al. Graphen-algorithmen. "Website". http://www-m9.ma.tum.de.

[5] Andreas Frommer, Stefanie Krivsky-Velten, and Karsten Blankenage. Matheprisma. "Website". http://www.matheprisma.uni-wuppertal.de.

[6] Jens Gallenbacher. *Abenteuer Informatik*. Spektrum, 2008.

[7] Werner Hartmann and Raimond Reichert. Swisseduc. "Website". https://www.swisseduc.ch.

[8] T.Leuders (Hrsg.). *Mathematik Didaktik, Praxishandbuch für die Sekundarstufe I und II*. Cornelson Scriptor, 2011.

[9] Wikimedia Foundation Inc. Wikipedia. "Website". http://www.wikipedia.org/.

[10] Google Ireland Limited. Youtube. "Website". https://www.youtube.com.

[11] Christian Wagenknecht and Michael Hielscher. Atocc. "Website". http://www.atocc.de.

[12] Christian Wagenknecht and Michael Hielscher. Flaci. "Website". https://flaci.com.

[13] Christian Wagenknecht and Michael Hielscher. *Formale Sprachen, abstrakte Automaten und Compiler*. Springer, 2014.

[14] W.Hartmann, M.Näf, and R.Reichert. *Informatikunterricht planen und durchführen*. Springer-Verlag Berlin Heidelberg, 2006.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich den vorliegenden ausführlichen Unterrichtsentwurf selbstständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entstammen, durch Angabe der Quellen als Entlehnung kenntlich gemacht habe.

Ulf Mögling,..... .....