ZPG IMP – KLASSE 9

DATEN UND CODIERUNG

UNTERRICHTSVERLAUF

Inhaltsverzeichnis

Übersicht

Der Bereich Daten und Codierung ist in zwei etwa gleich große Teile gegliedert, die unabhängig voneinander sind und auch in umgekehrter Reihenfolge unterrichtet werden können:

Teil 1: Die *Digitalisierung* analoger Signale mittels *Diskretisierung* sowie die *verlustbehaftete* und *verlustfreie Datenkompression*.

Hier sind verschiedene sinnvolle Unterrichtsgänge möglich:

(i) Der Einstieg in die Digitalisierung analoger Signale erfolgt über den Bereich Bilder. Diskretisierung, Rasterung, Bildauflösung, Farbtiefe werden thematisiert und Datenmengen berechnet. Darauf aufbauend folgt einerseits die verlustfreie Datenkompression am Beispiel der Lauflängencodierung und andererseits die verlustbehaftete Datenkompression bei Farbbildern. Optional kann das Thema danach auf die Digitalisierung von Audio übertragen werden. Folgende Arbeitsblätter sind relevant:

01_duc_ab_Digit_Bild.odt

02_duc_ab_Digit_Kompression.odt

optional: 03_duc_ab_Digit_Audio.odt

(ii) Die in Klasse 7 in Kapitel *Daten und Codierung* erworbenen Kompetenzen bezüglich der Binärdarstellung von Bildern werden aufgegriffen. Die Diskretisierung wird veranschaulicht anhand von Pixelgrafiken im Dateiformat Portable Anymap: PBM für Schwarzweiß-Bilder, PGM für Graustufenbilder und PPM für Farbbilder. Hier werden Bildauflösung und Farbtiefe thematisiert und die entstehenden Datenmengen berechnet. Relevant sind dann folgende Arbeitsblätter:

01e_duc_ab_Digit_Ergänz_Dateiformate.odt

01_duc_ab_Digit_Bild.odt

02_duc_ab_Digit_Kompression.odt

(iii) Der Einstieg in die Digitalisierung analoger Signale erfolgt über den Bereich Audio. Hier können Diskretisierung, Abtastrate, Samplingtiefe und Datenmengen behandelt werden. Die verlustbehaftete Kompression lässt sich bei Audio-Dateien gut veranschaulichen. Die Durchführung der Lauflängencodierung erfolgt dann auf Basis von Bitfolgen und ist damit nicht mehr so eng mit dem Thema Audio verknüpft wie bei Bildern, was aber kein Nachteil sein muss. Danach kann ggf. ein Vergleich mit Diskretisierung bei Bildern und Grafiken erfolgen. Relevant sind folgende Arbeitsblätter:

03_duc_ab_Digit_Audio.odt

02_duc_ab_Digit_Kompression.odt

optional: 01_duc_ab_Digit_Bild.odt

In den hier vorgestellten Materialien wird der erste Weg beschrieben, es sind aber auch Materialien für die Digitalisierung von Audio vorhanden, die bei Bedarf entsprechend angepasst werden können.

Die Arbeitsblätter zu den Portablen Bitmaps orientieren an den Seiten von inf-schule.de[1].

1 *https://www.inf-schule.de/information/darstellunginformation/binaerdarstellungbilder* (abgerufen am 9.3.19)

Teil 2: Das Problem des kürzesten Pfades und der Algorithmus von Dijkstra.

Zunächst wird das Versagen der Brute-Force-Methode aufgezeigt, indem die SuS erkennen dass die Anzahl aller möglichen Wege bei großen n exponentiell ansteigt. Dann wird aufgezeigt, dass ein spezifisches Verhalten von Ameisen zum Finden des kürzesten Weges sehr effizient ist und darauf aufbauend, dass der Algorithmus von Dijkstra wesentliche Elemente dieses Verhaltens aufgreift und ein effektiver Algorithmus zur Berechnung des kürzesten Weges ist.

Dieses Vorgehen orientiert sich an dem Buch 'Abenteuer Informatik' von Jens Gallenbacher[2].

Voraussetzungen:

Der zweite Themenbereich greift die Kompetenzen aus Klasse 8 bezüglich *Aussagenlogik und Graphen* (IMP-Mathematik) auf. Notwendige Voraussetzung sind aber nur die Begriffe *Knoten* und *Kante.*

Differenzierung:

Zum jeweiligen Themengebiet passende Differenzierungsmöglichkeiten finden sich direkt bei der Beschreibung der Unterrichtsstunde. Weiterführende Aufgaben sind mit ** markiert.

2 Jens Gallenbacher: Abenteuer Informatik. Springer, Heidelberg, 2008. Kapitel 1.

Digitalisierung von Bildern

Material:

- *01_duc_ab_Digit_Bild.odt*

Beim Vorgang der Digitalisierung wird das Abbild eines realen Objekts als Bild gespeichert. Anhand des Fotos eines Hundes werden auf dem Arbeitsblatt die Begriffe Diskretisierung, Rasterung, Pixel, Farbtiefe, Bildauflösung veranschaulicht. Dabei ist zu beachten, dass das abgebildete Originalfoto des Hundes nicht das Original ist, sondern bereits digitalisiert.

Die **Farbtiefe** ist der *Speicherplatz*, der benötigt wird, um die Farbinformation für ein Pixel zu speichern. Gelegentlich wird die *Anzahl der möglichen Farben* mit der Farbtiefe verwechselt. Beispiel: Eine Farbtiefe von 3 Bit bedeutet, dass 3 Bit Speicherplatz je Pixel zur Verfügung steht. Damit können $2^3 = 8$ verschiedene Farben dargestellt werden.

Die **Bildauflösung** wird in dpi (Pixel je Inch) angegeben, und macht damit eine Aussage über die Qualität des Bildes. Bei Kameras hat sich die Angabe in MP (MegaPixel) etabliert und bezeichnet die Gesamtzahl der Pixel des Kamerasensors.

Die Gesamtzahl der Pixel des Kamerasensors liefert nur dann einen Hinweis über die Bildqualität, wenn die Größe des Sensors genannt wird. 12 MP bei einer Spiegelreflexkamera bedeuten beispielsweise etwas anderes als 12 MP bei einer Kompaktkamera. Bei einer größeren Fläche ist die Abgrenzung der einzelnen Messpunkte trennschärfer als bei einer kleinen Fläche. Da in (fast) allen Handykameras dieselbe Sensorgröße verbaut wird, verzichten die Handyhersteller meist auf die Nennung der Sensorgröße und geben nur die Gesamtzahl der Pixel als Qualitätsmerkmal an. Weiterhin ist die Verarbeitung der Messwerte ausschlaggebend für die Bildqualität, und die kann bei derselben Pixelanzahl sehr unterschiedlich sein. Ein weiterer Qualitätsfaktor ist die Optik der Linse. Wichtig für die SuS ist an dieser Stelle, dass die Angabe der Bildauflösung bei Kameras nur einen ungefähren Hinweis auf die Qualität des Bildes gibt.

Da viele SuS keine analoge Fotografie mehr kennen, kann die Herstellung von analogen Bildern thematisiert und mit der von digitalen Bildern verglichen werden (Kamera – Film belichten – Negativstreifen – Belichten des Fotopapiers). In der analogen Fotografie bestimmt das Bildkorn das Auflösungsvermögen des Films. Je lichtempfindlicher der Film ist, desto grobkörniger ist er.

In Aufgabe 2 wird die Bildauflösung für zwei verschieden große Bilder ermittelt, die dieselbe Gesamtzahl an Pixeln haben. Die entstehende Datenmenge wird in Aufgabe 2(c), 4 und 5 berechnet. Eine allgemeine Formel zur Berechnung Datenmenge erstellen die SuS in Aufgabe 3.

In Aufgabe 6 recherchieren die SuS die Bildauflösung bei Drucker, Bildschirm, Kamera und Fotoausdruck. Hier kann thematisiert werden, dass man unterscheiden muss zwischen Medien zur Digitalisierung und Wiedergabemedien. Eine gute Qualität bei der Digitalisierung (Kamera) liefert auf einem schlechten Wiedergabemedium (z. B. Bildschirm) kein optimales Ergebnis.

In Aufgabe 5 wird die Datenmenge des Origianalfotos berechnet, wobei den SuS auffallen wird, dass der berechnete Wert viel größer ist als normalerweise bei Fotos üblich. Damit könnte man zur JPEG-Kompression überleiten. (Arbeitsblatt Datenkompression, ab Aufgabe 11)

In Aufgabe 7 wird ausgehend von Schwarzweiß-Bildern zur Kompression hingeleitet. Je nach Antworten und Ideen der SuS kann zuerst verlustbehaftete und/oder verlustfreie Kompression vertieft werden.

Verlustfreie Kompression: Arbeitsblatt Datenkompression, ab Aufgabe 1

Verlustbehaftete Kompression: Arbeitsblatt Datenkompression, ab Aufgabe 11

Datenkompression

Material:

- *02_duc_ab_Digit_Kompression.odt*
- *kompression_1.png, kompression_2.png*
- *Luna.jpg*

Verlustfreie Datenkompression – Lauflängencodierung

Bei der *Datenkompression* (auch: *Datenkomprimierung*) wird die Menge an Daten reduziert, also verringert. Wenn die Verringerung der Datenmenge mit einem Qualitätsverlust einher geht, spricht man von *verlustbehafteter* Datenkompression, ansonsten von *verlustfreier* Kompression. Dabei ist es unwesentlich, ob der Qualitätsverlust wahrnehmbar ist oder nicht. Relevant ist nur, ob die ursprüngliche Datei aus den reduzierten Daten wiederhergestellt werden kann oder nicht.

In Aufgabe 1 werden zwei Bilder verglichen, die dasselbe Format, aber ein ganz unterschiedliches Datenvolumen haben. Der Inhalt der Datei beeinflusst also deren Größe.

Die SuS analysieren in Aufgabe 2 eine Lauflängencodierung (engl.: RLE, Run Length Encoding) (9s 6w 2s 1w 1s 2w 1s 1w 2s 6w 2s 1w 4s 1w 2s 6w 9s) und ermitteln daraus die Bilddaten.

Bei Schwarzweiß-Bildern kann die Information, dass auf 9 schwarze Pixel 6 weiße folgen, weggelassen werden, weil schwarze und weiße Pixel sich abwechseln. Die Bilddaten können weiter reduziert werden zu: 9 6 2 1 1 2 1 1 2 6 2 1 4 1 2 6 9. Wichtig ist hierbei, dass eine Vereinbarung getroffen werden muss, mit welcher Farbe begonnen wird.

Im Kompressionsalgorithmus ist festgelegt, mit wie vielen Bit eine Zahl codiert wird. Diese Bitanzahl ist nicht abhängig vom der jeweiligen Datei. Denkbar wäre allerdings auch, dass ein Kompressionsalgorithmus die Datei analysiert und die kleinstmögliche Codierung für eine Zahl festlegt – individuell für jede Datei.

In Aufgabe 3 wird die Lauflängencodierung geübt und die Speicherplatzersparnis berechnet. Eventuell erkennen die SuS bereits an dieser Stelle, dass nicht immer eine Ersparnis erreicht wird (Aufgabe 4).

Im Bild aus Aufgabe 2 wird beispielsweise keine Reduzierung erreicht. Ein einzelnes Pixel benötigt (bei binärer Speicherung) 1 Bit Speicher, 9 Pixel benötigen 9 Bit Speicher. Kommen 9 gleiche Pixel vor, benötigt man 4 Stellen, um die Zahl 9 binär darzustellen, also 4 Bit. Allerdings werden dann auch '2 weiße Pixel' mit 4 Bit codiert. Erst ab 5 gleichartigen Pixel ergibt sich also eine Einsparung durch die Lauflängencodierung.

Nach dieser Erkenntnis kann man mit den SuS Verbesserungsvorschläge sammeln, um die Lauflängencodierung effektiver zu gestalten (Aufgabe 7 b).

Ein Verbesserungsvorschlag könnte sein, dass die Lauflängencodierung nur angewendet wird, wenn es eine Verbesserung bringt. Im Beispiel aus Aufgabe 2 sähe das so aus:

9 6 sswswwswss 6 sswssssswss 6 9 bzw.: 9 6 1101001011 6 1101111011 6 9

Hinweis: Die entstehende Problematik erkennen die SuS eventuell erst, wenn die binäre

Darstellung betrachtet wird: Wie kann unterschieden werden, ob es sich um eine Folge von s/w Bits handelt (z.B. swws => 1001_2) oder um eine Binärzahl (z.B. 9 => 1001_2)?

Kommen in den Daten nicht nur zwei sich abwechselnde Zeichen (s, w) vor, sondern mehrere verschiedene, muss zusätzlich zur Zahl auch das Zeichen genannt werden, das wiederholt wird. Bsp.: AAABBBBBCCDDDDDDEEF wird codiert zu: A3 B5 C2 D6 E2 F1. (Aufgabe 5, Aufgabe 9)

Damit die komprimierte Datei nicht länger wird als die originale, könnte man nur Zeichen codieren, die häufiger als z.B. dreimal hintereinander auftreten.

AAABBBBBCCDDDDDDEEF => AAAB5CCD6EE

Hierbei muss aber unterschieden werden können, ob es sich um ein Einzelzeichen oder um eine Wiederholung handelt. Das kann durch ein Sonderzeichen geschehen, welches sonst nie verwendet wird. z.B. AAA#B5CC#D6EE. Falls alle Zeichen im Text vorkommen können, dann kann man ein möglichst seltenes Zeichen verwenden. Kommt es nun im Text vor, muss es dort als Wiederholung gekennzeichnet werden. Aus # wird dann ##1. Bsp.: ABBB#CDDDDE## => ABBB##1C#D3E##2

Die Lauflängencodierung kann auf verschiedenen Ebenen angewendet werden: Auf Bit-Ebene wie in Aufgabe 1 und 2, auf Zeichen-Ebene wie in Aufgabe 4 oder auch auf Pixel-Ebene wie in Aufgabe 5.

Hinweis: Bei den Aufgaben wird nicht unterschieden, ob die Zahlen binär- oder ASCII-codiert werden. Im Allgemeinen ergibt sich das aus der Aufgabenstellung. Wenn genau dieses Thema im Unterricht vertieft werden soll, dann sollten die Aufgabenstellungen angepasst und konkretisiert werden.

Hinweis:

Die Huffman-Codierung, die sich logisch anschließen würde, ist im Bildungsplan nicht vorgesehen.

Verlustbehaftete Datenkompression

Bei der verlustbehafteten Kompression[3] wird aus einer Datei Information entfernt, die später nicht wiederhergestellt werden kann. Ziel ist es nun, genau solche Information zu entfernen, die nicht *notwendig* ist – man spricht von *Irrelevanzreduktion.* Verlustbehaftete Kompression findet meist Anwendung in der Bild-, Video- und Audio-Übertragung.[4]

Die Entscheidung, welche Information für die Qualität des Bildes (oder auch z.B. einer Audiodatei) wesentlich ist, hängt von der menschlichen Wahrnehmung ab.

Bei Audiodateien könnten z.B. Töne, die im nicht hörbaren Bereich liegen, entfernt werden. Allerdings ist der hörbare Bereich nicht bei allen Menschen exakt gleich. Das Audio-Format MP3 liefert für einige eine völlig ausreichende Musik-Qualität, für andere ist MP3 inakzeptabel.

Die Netzhaut des menschlichen Auges kann Helligkeitsunterschiede viel stärker wahrnehmen als Farbunterschiede. Daher können Farben eher reduziert werden als Helligkeitsstufen, ohne dass ein qualitativer Unterschied bemerkt wird. Diesen Effekt macht sich bereits das Farbfernsehen mit der *YUV-422 Reduzierung* zunutze.

3 https://de.wikipedia.org/wiki/Datenkompression (abgerufen am 02.01.2019)

4 Theoretische Grundlage bildet die Shannonsche *Rate-Distortion-Theorie*. Sie beschreibt, welche Datenübertragungsrate mindestens nötig ist, um Informationen mit einer bestimmten Güte zu übertragen.

Linien und Kanten sehr wichtig für die Wahrnehmung von Objekten, so wird z.B. eine Strichzeichnung oft schneller erkannt als ein Farbbild desselben Objekts.

Weiterhin verstärkt die menschliche Bildverarbeitung Kontraste. Beim Effekt der Machschen Streifen werden an den Grenzen von einheitlich gefärbten Flächen die Kontraste viel stärker wahrgenommen als real vorhanden.[5] Bild[6]

Machsche Streifen

Der JPEG-Algorithmus macht sich solche Effekte (unter anderem) zunutze und erreicht eine sehr starke Reduzierung der Datenmenge, ohne deutliche Qualitätsverringerungen.

Neben der individuellen Wahrnehmung muss bei der Qualitätsentscheidung natürlich auch berücksichtigt werden, für welchen Zweck ein Bild benötigt wird (z.B. Betrachtung auf dem Handy oder auf einem großen Bildschirm; Ausdruck als Postkarte oder Poster) sowie die Qualität des Darstellungsmediums (z.B. sehr 'gutes' Foto in Verbindung mit einem 'schlechten' Beamer, Bildschirm bzw. Drucker).

Die SuS untersuchen in Aufgabe 11 das Aussehen und Datenmenge eines JPEG-Fotos, nachdem sie es in unterschiedlichen JPEG-Kompressionsgraden gespeichert haben. Wahrscheinlich sprechen die SuS von einer besseren oder schlechteren 'Qualität' des Bildes. Dann muss man thematisieren, dass zuvor definiert wurde, dass die 'Qualität' abhängig ist von der Auflösung und der Farbtiefe.

Bildausschnitt. Datenmenge: 459 KB (Original: 3.447 KB)

Bildausschnitt. Datenmenge: 272 KB (Original: 3.447 KB)

(Bild: M. Klein)

Kompressionsartefakte bei starker Vergrößerung der Bäume rechts oben

In Aufgabe 13 kann (optional oder differenzierend) auf den Effekt der Kompressionsartefakte eingegangen werden. Das sind sichtbare Blockstrukturen (daher *Blockartefakte*), die durch eine Unterteilung des Bildes in

5 https://de.wikipedia.org/wiki/Machsche_Streifen#/media/File:Maschsche_Streifen.svg (abgerufen am 12.3.19)

6 https://de.wikipedia.org/wiki/Machsche_Streifen#/media/File:Maschsche_Streifen.svg (abger. 12.3.19) Polini 14:14, 30.3.08 (CEST) cc-by-sa Vers. 3.0

(meist 8•8 Pixel) große Blöcke bei der JPEG-Kompression entstehen.

Kompressionsartefakte treten auch bei der Audio-Kompression als hörbare Störungen auf.

Ergänzung: Verlustbehaftete Audio-Kompression[7]

Wie bei Bildern spielt die Wahrnehmung eine große Rolle (Psychoakustik). Zum Beispiel werden Frequenzen oberhalb von ca. 20 kHz nicht wahrgenommen und können entfernt werden.

Zwei Töne werden erst dann als getrennte Töne wahrgenommen, wenn der Frequenzbereich weit genug auseinander liegt. Und ein leiser Ton ist nach einem lauten Ton zunächst nicht hörbar.

Durch solche Phänomene können die Ausgangsdaten ohne hörbaren Qualitätsverlust weiter reduziert werden.

Werden Musik, Sprache oder Geräusche auf Werte um etwa 192 kbit/s reduziert, können die meisten Menschen kaum Qualitätsunterschiede zum unkomprimierten Ausgangsmaterial feststellen.

Ein Standardverfahren zur Audiokompression ist das 1982 entwickelte MP3-Verfahren. Der maßgebliche Entwickler dieses Verfahrens ist der deutsche Mathematiker und Elektrotechniker Karlheinz Brandenburg.

Bei der Qualität von Audio-Aufnahmen muss unterschieden werden zwischen der Qualität der Aufnahme und der Qualität der Wiedergabe.

7 https://de.wikipedia.org/wiki/Datenkompression (abgerufen am 2.1.2019)

Digitalisierung von Audio

Material:

- *03_duc_ab_Digit_Audio.odt*
- *Audio-Datei zum Experimentieren*

Bei diesem Arbeitsblatt kann das Gelernte auf den Bereich Audio übertragen werden.

Die Darstellung von Tönen durch Schwingungslinien ist den SuS aus dem Physikunterricht (Klasse 8) bekannt. Ebenso der Zusammenhang zwischen Amplitude und Lautstärke und zwischen Enge der Amplitude und Tonhöhe. Der Begriff *Frequenz* ist jedoch nicht notwendigerweise bekannt.

Auf dem Arbeitsblatt wird das entstehende Raster aus Samplingtiefe und Samplingrate (Abtastrate) erklärt und veranschaulicht. Die entstehende Datenrate und Datenmenge werden berechnet.

Die SuS vergleichen in Aufgabe 1 die Vorgänge bei der Digitalisierung von Bildern und von Audio und setzen die Begriffe in Relation zueinander.

Hinweis: Der Begriff des *Rasters* kann zur Verwirrung führen: Bei einem Bild ist mit 'Rasterung' meist die Bildauflösung, also die Anzahl der Pixel gemeint. Ein akustisches Signal ist im Gegensatz zu einem Bild eindimensional. Hier ist mit 'Raster' meist das entstehende Raster aus Samplingtiefe und Samplingrate gemeint.

In Aufgabe 2 experimentieren die SuS mit dem Tool Audacity[8] (oder einem anderen Tool zur Bearbeitung von Audio-Dateien). Hier hilft es, wenn die SuS eigene Kopfhörer mitbringen. Sie zoomen in die Aufnahme bis man die einzelnen Abtastpunkte erkennt. Dann exportieren sie die Datei im MP3-Format und experimentieren mit verschiedenen Qualitätsstufen. Bitte stellen Sie den SuS dazu eine geeignete Audio-Datei zur Verfügung.

In den Aufgaben 4 und 5 werden Datenmengen berechnet.

In Aufgabe 6 recherchieren die SuS verschiedene Datenübertragungsraten. Dabei kann der Unterschied zwischen Datenrate und Übertragungsrate herausgestellt werden.

in Aufgabe 7 wird eine Übersicht über gängige Audioformate erstellt.

8 www.audacityteam.org (abgerufen am 5.5.2019)

Ergänzung: Dateiformate von Bildern

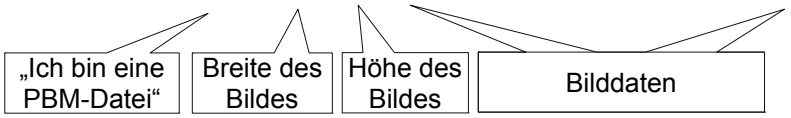
Material:

- *01e_duc_ab_Digit_Ergänzung_Dateiformate.odt*
- *pixelBeispiel.txt*
- *greymapBeispiel.pgm*
- *ev. svg_Grafik.txt*
- *Bildbetrachtungsprogramm, z.B. IrvanView*

Portable Bitmap für Schwarzweiß – Bilder

In Klasse 7 entwickelten die SuS Ideen, wie man ein Schwarzweiß-Bild binär darstellen kann. Es wurde die Notwendigkeit eines Protokolls thematisiert, also einer Vereinbarung über das Format, in dem das Bild übertragen wird. Sehr wahrscheinlich einigten sich viele SuS auf ein Format, das dem Portablen Bitmap sehr ähnlich ist.

Aufbau einer PBM-Datei: **P1 4 5 01110100011101000111**



Zunächst decodieren die SuS eine Bitfolge im PBM-Format und ermitteln das zugehörige Schwarzweiß-Bild. Zur besseren Erkennbarkeit sind Kopfdaten im Beispiel nicht binär angegeben.

Hinweis: Die Angabe der Bildhöhe ist im PBM-Format immer enthalten, sie ist aber nicht zwingend notwendig, weil sie sich aus der Breite und der Pixelzahl ergibt.

In Aufgabe 2 wird eine vorgegebene Textdatei durch Ändern der Endung in ein Portables Bitmap umgewandelt werden. Ergänzend können die SuS in der Textdatei einzelne Bits ändern und das Ergebnis mit einem Bildbetrachtungsprogramm (z.B. IrvanView) betrachten.

Optional können die SuS ein eigenes PBM-Bild erstellen.

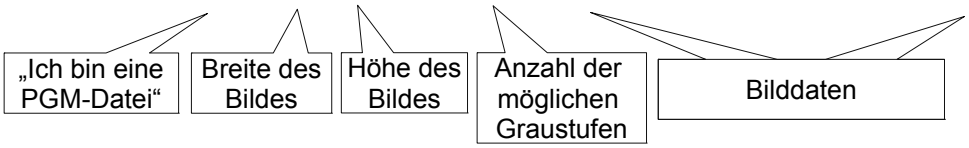
Bei der Betrachtung der Datenmenge bzw. des benötigten Speicherplatzes wird zwischen Header und Bilddaten unterschieden. Der Header P1_24_36_ besteht aus 9 Stellen (incl. der drei Leerzeichen). Je Stelle wird 1 Byte (8 Bit) benötigt, weil Buchstaben und Zahlen mit der ASCII-Tabelle codiert werden. Die Bilddaten benötigen so viele Bits wie das Bild Pixel hat.

Hinweis: Tatsächlich wird jedes Pixel mit 1 Byte codiert (00000001 für schwarz, 00000000 für weiß). Die SuS können dies selber herausfinden, indem sie den berechneten Wert mit dem im Dateimanager vergleichen.

Portable Greymap für Graustufen–Bilder

Für Graustufen-Bilder ist eine PGM-Datei abgebildet, bzw. es liegt die PGM-Datei bereit. Die SuS ermitteln daraus das allgemeine PGM-Format.

Aufbau einer PGM-Datei : `P1 24 36 255 129 258 178 ...`



kleine Zahlen: dunkle Pixel

große Zahlen helle Pixel

Anhand der Abbildung über verschiedene Graustufen wird veranschaulicht, wie viele Abstufungen jeweils möglich sind, wenn 1, 2, 4, bzw. 8 Bit zu Darstellung eines Pixels verwendet werden.

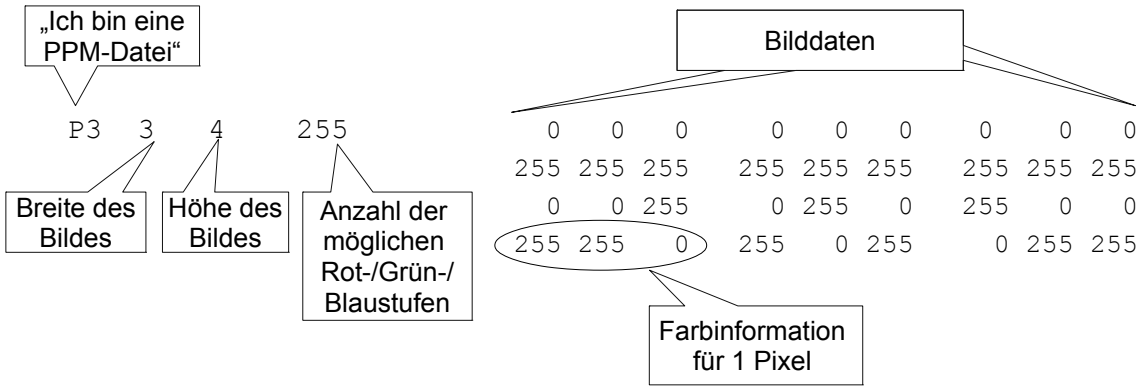
Die 1, 2, 4, 8 Bit entsprechen jeweils der Farbtiefe.

Farbtiefe = 1 bit: 0,1→ 2 Farbstufen

Farbtiefe = 2 bit: 00,01,10,11→ 4 Farbstufen

Farbtiefe = 3 bit: 000, 001,010, 011, 100, 101, 110, 111 → 8 Farbstufen usw.

Portable Pixmap für Farb–Bilder



Nach den qualitativ sehr einfachen Schwarzweiß-Bildern und den etwas realistischeren Graustufen-Bildern bilden Farbbilder die Realität viel besser ab. Allerdings wird die Datenmenge auch sehr viel größer.

Bei Farbbildern wird häufig der RGB-Farbraum verwendet mit je 8 Bit für den Rot-, den Blau- und den Grün-Wert. (256^3 = 16.777.216 verschiedene Farben). Bei 16 Bit ergeben sich ca. 281 Billionen Farben.

Es können weitere, den SuS bekannte Bildformate gesammelt werden, um bei der Datenkompression wieder aufgegriffen zu werden, z.B. JPG, BMP, RAW, GIF, PNG, SVG.

Alternativen/Ergänzungen:

Nicht im Bildungsplan enthalten sind SVG-Grafiken. Zur Differenzierung oder als Ergänzung kann die Vektordarstellung als Alternativ-Konzept zur Pixeldarstellung betrachtet werden. In Aufgabe 6 ist dazu ein Beispiel zum Selbsterkunden.

Das Problem des kürzesten Weges – Dijkstra und die Ameisen

Material:

- *04_duc_ab_kürzesterWeg.odt*
- *04_duc_kürzesterWeg.odp*

Graphen sind den SuS aus IMP-Mathematik Klasse 8 bereits geläufig. Voraussetzung für dieses Kapitel sind aber lediglich die Begriffe *Knoten* und *Kante*.

Als Einstieg eignet sich eine Straßenkarte, verbunden mit dem Auftrag, den kürzesten Weg von einem Ort zu einem anderen zu finden (siehe Präsentation). Dabei kann thematisiert werden:

- Die Länge der gezeichneten Straßen muss nicht der tatsächlichen Länge entsprechen.
- Der Graph kann als Modell der Wirklichkeit gesehen werden, und damit die Problemstellung weiter vereinfachen. Alle Dinge, die nicht zur Lösung des Problems notwendig sind, werden weggelassen. Wirklichkeit → Straßenkarte → Graph.
- Jede Straßenkarte kann als Graph gezeichnet werden. Dabei ist jede Straßenkreuzung als Knoten zu betrachten, auch wenn sich dort keine Ortschaft befindet.
- Die Suche nach dem kürzesten bzw. dem schnellsten Weg unterscheiden sich nicht. Der Unterschied liegt nur in der Einheit der Kantenbeschriftung (z.B. km, min).

.Nachfolgend wird das Problem auf dem Arbeitsblatt strukturiert betrachtet.

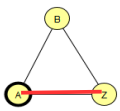
In Aufgabe 1 wird der kürzeste Weg in zwei übersichtlichen Graphen gesucht. Die Frage, wie man sicher sein kann, dass es keinen kürzeren Weg gibt, lässt sich bei dem ersten Graphen einfach durch Aufschreiben aller möglichen Wege beantworten. Bei dem zweiten Graphen werden die meisten SuS argumentieren, dass ihr Weg der kürzeste sein *muss*. Falls einige Schüler nicht auf ABEF (12) kommen, sondern auf z.B. ADCF (15), ADCEF (14) oder ADEF (13), erkennen sie unmittelbar, wie leicht man vom eigenen Ergebnis überzeugt ist und dennoch falsch liegt. Bei diesem zweiten Graphen ist die Brute-Force-Methode, nämlich die Länge aller Wege zu ermitteln, sehr viel schwieriger, dennoch könnten sich manche SuS herausgefordert fühlen, alle Wege zu finden.

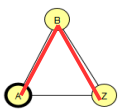
In Aufgabe 2 wird definiert, was genau die Brute-Force-Methode zum Finden des kürzesten Weges beinhaltet, nämlich:

> Suche alle Wege, die bei A beginnen und die jeden Knoten höchstens einmal besuchen.
> Für jeden dieser Wege berechne die Entfernung von A bis F.

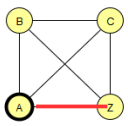
In Aufgabe 3 ermitteln die SuS nun die Anzahl aller möglichen Wege in einem vollständigen Graph, die bei A beginnen und bei Z enden:

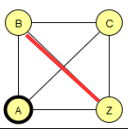
3 Knoten: **2 Wege**

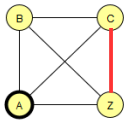




4 Knoten: Z kann über drei Wege erreicht werden:





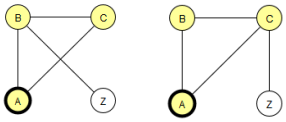


direkt von A, von B kommend, von C kommend.

- Um von A nach Z zu gelangen, gibt es genau einen Weg.

- Um im Graph ABC von A nach B zu gelangen, gibt es 2 Wege.

- Um im Graph ABC von A nach C zu gelangen, gibt es 2 Wege.



Also gibt es 1+2+2 = **5** verschiedene Wege von A nach Z.

5 Knoten: Z kann über 4 verschiedene Wege erreicht werden: Direkt von A aus, von B, von C, von D kommend.

- Um von A nach Z zu gelangen, gibt es genau einen Weg.

- Um im Graph ABCD von A nach B zu gelangen, gibt es 5 Wege.

- Um im Graph ABCD von A nach C zu gelangen, gibt es 5 Wege.

- Um im Graph ABCD von A nach D zu gelangen, gibt es 5 Wege.

Also gibt es 1+5+5+5 = **16** verschiedene Wege von A nach Z.

6 Knoten: 1+16+16+16+16 = 1+3•16 = **65** Wege

7 Knoten: ...

Bei 100 Koten sind es bereits $2{,}56 \cdot 10^{154}$ verschiedene Wege.

In Deutschland gibt es zum Vergleich 10.848 Gemeinden (Stand 2019). Jeder Ort ist mit einigen Orten über Wege verbunden, aber nicht allen. Trotzdem ist leicht einsichtig, dass Zahl der möglichen Wege zwischen zwei Orten sehr groß sein kann und die Berechnung aller möglichen Wege (viel) zu lange dauert.

Die Brute-Force-Methode ist also grundsätzlich ein guter Ansatz, um mit Sicherheit den kürzesten Weg zu finden, aber in der Realität nicht geeignet, um z.B. in einem Navigationsgerät verwendet zu werden.

Bevor nun der Dijkstra-Algorithmus eingeführt wird, wird das Ameisen-Verhalten zum Finden des kürzesten Weges behandelt, das das Prinzip sehr anschaulich darstellt.[9] Das bietet neben der Veranschaulichung eine Vernetzung mit der Biologie, sowie die Erkenntnis, dass Vorgänge, wenn sie in einen völlig anderen Bereich übertragen werden, zu erstaunlichen Ergebnissen führen können.

Wenn Ameisen von ihrem Bau zu einer Futterquelle laufen, dann laufen sie nach einer Weile auf dem kürzesten Weg. Sie zeigen dabei folgendes Verhalten: sie teilen sich auf und laufen vom Bau aus in alle möglichen Richtungen los. Kommen sie zu einem Abzweig, dann teilt sich der Trupp auf und in jede mögliche Richtung laufen Ameisen-Trupps weiter. So verfahren sie an jedem Knoten-Punkt, wenn mehrere Richtungen möglich sind. Kommen Ameisen zu einem Knoten-Punkt, an dem bereits Ameisen sind, dann wissen sie, dass die anderen Ameisen einen schnelleren Weg gefunden haben (und sie selber können umkehren). Treffen zwei Ameisentrupps aufeinander, während sie einen Weg entlanglaufen, so ist klar, dass dieser Weg aus beiden Richtungen betrachtet, ungeeignet ist, um schnell zum Ziel zu kommen. Beide Trupps können umkehren und einen anderen Weg suchen. Die Ameise, die als erste am Ziel ankommt, hat den kürzesten Weg gefunden.

9 Dieses Vorgehen orientiert sich an Jens Gallenbacher: Abenteuer Informatik. Springer, Heidelberg, 2017

Voraussetzung ist, dass es sehr viele Ameisen gibt, und dass sich alle gleich schnell bewegen.

Dieser 'Algorithmus' kommt der Realität sehr nahe. In der Natur markieren Ameisen ihre Wege mit Pheromonen. Auf einem sehr langen Weg kommen sie seltener entlang – weil sie dafür länger benötigen – und die Pheromon-Spur verfliegt schneller. Auf einem kurzen Weg, kommen Ameisen häufiger entlang, weil die Entfernung kürzer ist, deshalb ist die Pheromon-Spur intensiver.

Auf dem Arbeitsblatt sind die wesentlichen Elemente des Algorithmus aufgeschrieben. Die Fallunterscheidung, falls mehrere Trupps aufeinandertreffen notieren die SuS selber. Zunächst erweckt es den Eindruck, dass es drei verschiedene Fälle gibt: (1.) ein Trupp trifft, während er sich auf einem Weg befindet auf einen anderen. (3.) ein Trupp kommt zu einem Ort, an dem bereits Ameisen sind. (2.) Zwei Trupps kommen gleichzeitig an einem Ort an.

Der Fall 2, bei dem zwei Trupps gleichzeitig an einem Ort ankommen, bedeutet, dass beide Wege gleich schnell sind. Dann vereinen sich die beiden Trupps und teilen sich, wie ein einzelner Trupp, auf die verbleibenden Wege auf. Daher muss dieser Fall nicht gesondert betrachtet werden.

Der Fall 3 *kann* gar nicht auftreten (obwohl er zur Erklärung des Verhaltens als wesentlich erscheint). Wenn ein Trupp zu einem Ort kommt, dann teilt er sich auf und läuft auf allen möglichen Wegen weiter, also auch auf dem Weg, auf dem die langsameren Ameisen gerade anmarschieren, und damit handelt es sich um Fall 1.

In Aufgabe 3 simulieren die SuS das Ameisen-Verhalten Schritt für Schritt. Das kann zunächst gemeinsam an der Tafel begonnen werden und dann von den SuS selbstständig zu Ende geführt werden.

Die Frage, wie man das Ameisen-Verhalten in einen computertauglichen Algorithmus überführen kann, führt zum Dijkstra-Algorithmus.

Hinweis: Der Ameisen-Algorithmus ist nur zur Veranschaulichung gedacht, relevant ist der Dijkstra-Algorithmus.

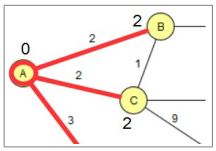
Die SuS analysieren zunächst den vorgegebenen Algorithmus, indem sie ihn auf einen einfachen Graphen anwenden. Das kann gemeinsam an der Tafel erfolgen oder in Kleingruppen.

Hinweis: Beim Übergang vom Ameisenverhalten zum Dijkstra-Algorithmus können einige Schüler Schwierigkeiten haben, weil die Algorithmen nicht exakt gleich sind: Die Ameisen laufen alle gleichzeitig. Es gibt sehr viele Ameisen, die gleichzeitig in alle Richtungen laufen. Der Dijkstra-Algorithmus hingegen wertet immer nur einen Knoten aus und bestimmt die Entfernungen zu den Nachbarknoten. Dabei kann es durchaus vorkommen, dass man zunächst zu große Zahlen bei einem Knoten einträgt. Das passiert bei den Ameisen nicht, da den Ameisen schon unterwegs andere Ameisen entgegenkommen. Daher tritt der Arbeitsschritt des Vergleichens mit den schon eingetragenen Werten und ggf. Anpassung des bisher besten Weges bei den Ameisen überhaupt nicht auf. Man kann der Verwirrung vorbeugen, indem man darauf hinweist, dass Dijkstras Algorithmus auf der prinzipiellen Idee des Ameisenverhaltens aufbaut, allerdings so gestaltet ist, dass er sinnvoll auf einem Computer ausgeführt werden kann. Eventuell kann man mit den SuS die Unterschiede zwischen beiden Algorithmen herausarbeiten.

Beide Algorithmen liefern eine Baum-Struktur, nachdem die unbrauchbaren Wege entfernt bzw. als unbrauchbar markiert wurden. Dieser Baum zeigt, beginnend bei der Wurzel (Startort), die kürzesten Wege zu allen anderen Orten.

Hinweise zum Dijkstra-Algorithmus:

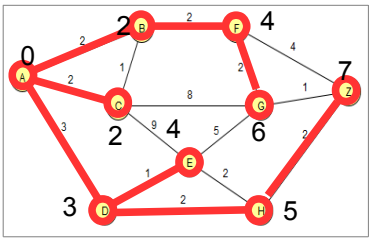
Um die Darstellung möglichst überschaubar zu halten und Redundanz zu vermeiden, wurde bei der Beschreibung des Dijkstra-Algorithmus auf den Eintrag der unendlichen Entfernung an jedem Knoten zu Beginn verzichtet. Ebenso auf den Eintrag des jeweiligen Vorgänger-Knotens. Wenn man mehr Nähe zu der programmtechnischen Umsetzung herstellen möchte, kann man z.B. die relevanten Wege als Pfeil darstellen (gerichtete Kanten). Alternativ kann an jeden Knoten auch zusätzlich der Vorgängerknoten vermerkt werden.



Haben mehrere Knoten dieselbe Entfernung, wählt man davon einen beliebigen als aktuellen Knoten. In der Abbildung kann man B oder C als nächsten aktuellen Knoten wählen.

Falls es mehrere kürzeste Wege gibt, findet der Algorithmus nur *einen*, nicht alle. Die Ameisen finden im Gegensatz dazu immer alle kürzesten Wege.

In der Abbildung ist A-D-H-Z (7) ist der gefundene kürzeste Weg zu Knoten Z. Der Weg A-B-F-G-Z (7) ist gleich kurz, wird aber nicht gefunden. (Siehe dazu Aufgabe 2 b.)



Grundsätzlich ist der Dijkstra-Algorithmus dann zu Ende, wenn es keine nichtmarkierten Knoten mehr gibt. Damit findet der Algorithmus die kürzesten Wege vom Start-Knoten zu allen anderen Knoten. Bei der Suche des kürzesten Weges vom Start zum Ziel, kann man den Dijkstra-Algorithmus vorzeitig beenden, nämlich sobald der aktuelle Knoten gleich dem Zielknoten ist.

Der Dijkstra-Algorithmus liefert nur dann den kürzesten Weg, wenn die die Kantengewichte positiv sind. Da die SuS wahrscheinlich nicht auf die Idee von negativen Kantengewichten kommen, muss das nicht weiter thematisiert werden.

In Aufgabe 2 stehen zwei Graphen zur Anwendung des Algorithmus zur Verfügung. Man kann zusätzlich die beiden Graphen des letzten Arbeitsblattes verwenden und damit ggf. Unterschiede zum Ameisenverhalten herausarbeiten. Ein zusätzlicher komplexerer Graph (2(c)) ist optional und etwas herausfordernd.

In Aufgabe 3 wird ein Straßennetz dargestellt, in dem 'Einbahnstraßen' enthalten sind.

Der Algorithmus kann wie gewohnt ausgeführt werden, es müssen lediglich die Richtungen der Pfeile beachtet werden.

Hinweis: Bei gerichteten Graphen sind die Kanten als Pfeile dargestellt, die angeben, in welche Richtung die Beziehung gilt. Dies kann als 'Einbahnstraße' interpretiert werden. Gehen die Pfeile in beide Richtungen, gilt auch die Beziehung in beide Richtungen. Bei ungerichteten Graphen, gehen die Pfeile immer in beide Richtungen und werden daher weggelassen. Die Kante wird als Linie dargestellt. Als weitere Differenzierung kann die Darstellung eines gerichteten bzw. eines ungerichteten Graphen als Tabelle thematisiert werden.

In Aufgabe 4 werden kürzeste Wege übertragen auf Beziehungen in sozialen Netzwerken.

Die Aufgaben 5-7 stehen zur Differenzierung oder Ergänzung zur Verfügung.

In Aufgabe 5 wird in einem Tool des Lehrstuhls M9 der TU München der Dijkstra-Algorithmus

Schritt für Schritt ausgeführt und erklärt[10]. Es sind leichte Kontrollfragen zu beantworten. Unter 'Weiteres' werden weitere Problemstellungen vertieft: Wie sieht der (Pseudo-)Code des Algorithmus aus? Wie schnell ist der Algorithmus? Berechnet der Algorithmus wirklich die kürzesten Wege? Wie löst man das Problem negativer Kantenkosten? Wo finde ich noch mehr Informationen zu Graphalgorithmen?

In Aufgabe 6 wird der A*-Algorithmus visualisiert. Zwischen Start und Ziel kann man zunächst ein vertikales Hindernis setzen und führt dann nacheinander den Dijkstra- und der A*-Algorithmus aus.[11]

In Aufgabe 7 kann der A*- Algorithmus von den SuS näher untersucht werden.[12]

Zusatzmaterial

- Eine weitere Seite zum A*-Algorithmus:
  http://www.geosimulation.de/methoden/a_stern_algorithmus.htm[13]
- Sehr anschaulicher Vergleich von mehreren kürzester-Weg-Algorithmen:
  https://www.redblobgames.com/pathfinding/a-star/introduction.html[14]
- Tool zum Testen des Dijkstra-Algorithmus: Material → Software → Dijkstra.jar[15]

  Tool zum Erstellen von Graphen: Material → Software → GraphEditor.jar

10https://www-m9.ma.tum.de/graph-algorithms/spp-dijkstra/index_de.html (abgerufen am 28.04.19)
11 https://qiao.github.io/PathFinding.js/visual/ (abgerufen am 28.04.19)
12 https://www-m9.ma.tum.de/graph-algorithms/spp-a-star/index_de.html (abgerufen am 28.04.19)
13 Abgerufen am 28.04.19
14 Abgerufen am 28.04.19
15 Erstellt von Dirk Zechnal, Tom Schaller