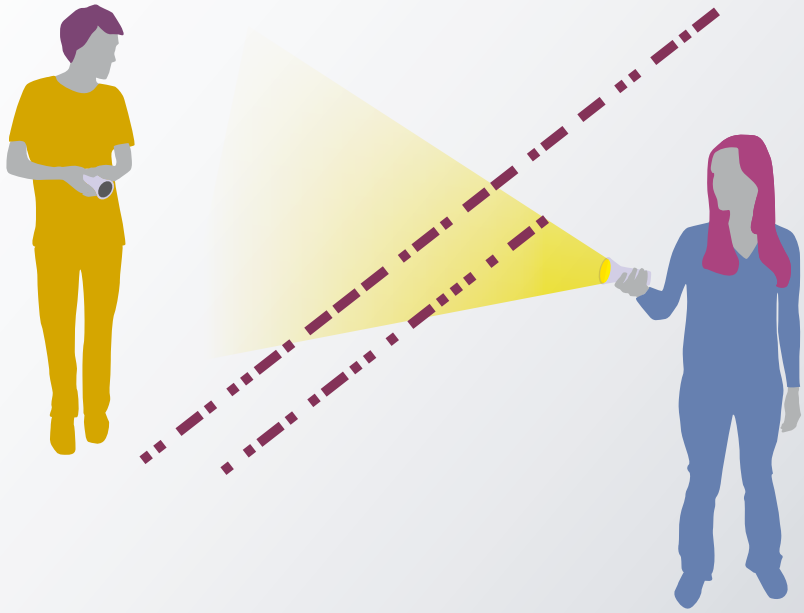
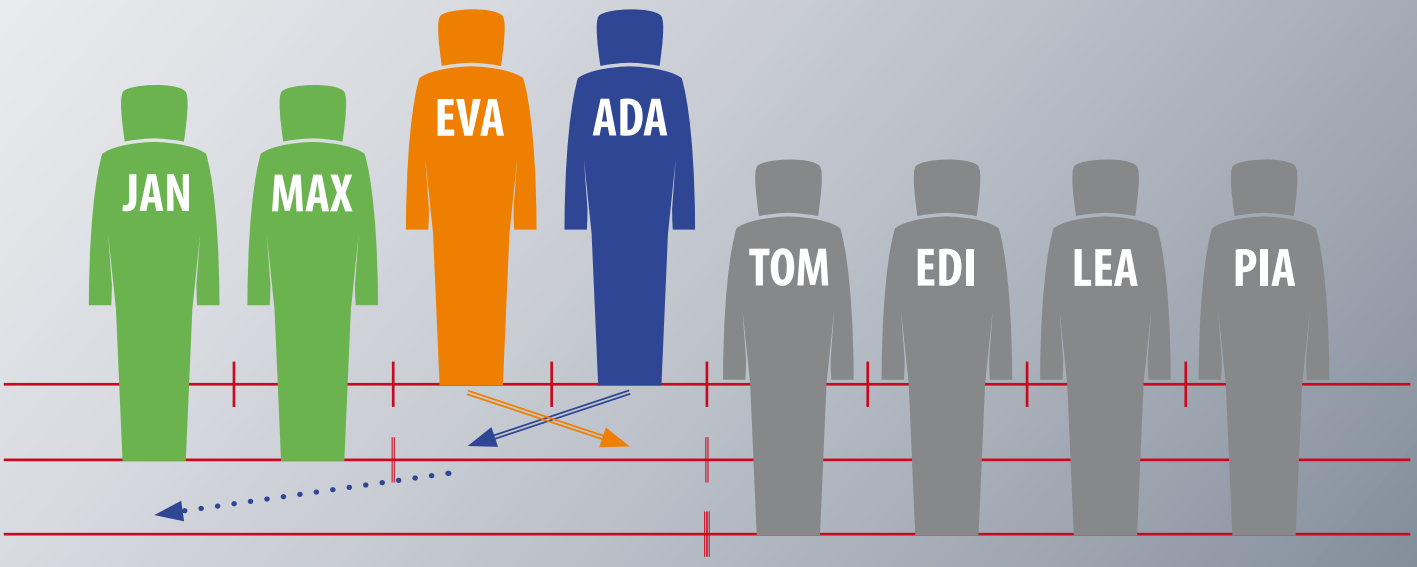


Ernestine Bischof
Roland Mittermeir

INFORMATIK erLeben

Beispiele für schülerinnen- und schüleraktivierenden Informatikunterricht



Autoren:
Ernestine Bischof
Roland T. Mittermeir

Institut für Informatik-Systeme
Alpen-Adria Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt
Österreich

E-Mail:
ernestine@isys.uni-klu.ac.at
roland@isys.uni-klu.ac.at

Die Autoren danken dem IT-Campus Kärnten für die Initialförderung des Projektes *Informatik erLeben* und dem KWF | Kärntner Wirtschaftsförderungs Fonds für die Finanzierung der Drucklegung dieser Broschüre sowie für die Unterstützung zur Weiterführung des Projekts.

Inhalt

	SEITE
Warum *Informatik erLeben*?	3
Schnuppereinheiten - Einleitung	5
C – Codierung	
C1 – Morsespiel	6
C3 – Codierung und Codebäume	11
V – Verschüsselung	
V1 – Verschlüsselung mit der Cäsar-Chiffre	16
So – Sortieren	
E-So1 – Schuften oder Denken?	19
So2 – Selection-Sort	22
So4 – Mergesort	26
Ein Blick auf die informatische *erLebens*-Landschaft	32
Einladung	35
Glossar	36

In Informatik steckt mehr als man denkt.

IN
INFO
IN FORM
INFORMATIK
INFORMATIKER
INFORMATIKER LEBEN
INFORMATIK ERLEBEN
INFORMATIKERINNEN LEBEN
INFORMATIKERIN IN FORM
IN INFORMATIK IN FORM
INFORMATIK
IN FORM
INFO
IN

Informatik bietet viele kreative Möglichkeiten.

Peter Holub

Warum *Informatik erLeben*?

Die Schule bereitet Jugendliche auf ihr nachschulisches Leben vor. Dies geschieht durch Wissensvermittlung, aber auch über die Vermittlung von Einstellungen. Wir sehen in der Entwicklung von Einstellungen gegenüber Technik und insbesondere gegenüber Informationstechnik und Informatik Defizite, die durch Erarbeiten von Lehreinheiten, die hier exemplarisch vorgestellt werden, punktuell abgebaut werden sollen.

Technik ist aus der Perspektive heutiger Jugendlicher etwas Vorhandenes. Man nutzt und benutzt sie, ohne viel darüber zu reflektieren, wie und warum sie entstand. Sie ist einfach hier. Vielleicht bestehen noch Vorstellungen darüber, wie sich diese Technik weiter entwickeln könnte. Derartige Vorstellungen werden meist von Science-Fiction inspiriert und dann noch von manchen technikfaszinierten Experten verstärkt. Wie ein Leben ohne die heute als selbstverständlich angesehene Technik aussehen würde, darüber wird kaum reflektiert. Allenfalls wird thematisiert, dass die heute verfügbare und die für morgen an die Wand gezeichnete Technik auch bedenkliche Nebenwirkungen hat. Somit besteht Raum für Technikkritik und Technikskepsis. Dies wäre an sich gut und berechtigt, wenn diese Technikkritik mit einem gehörigen Maß an Technikverständnis gepaart wird.

Letzteres wird unseren Jugendlichen im Rahmen des alltäglichen Schulunterrichts allerdings kaum vermittelt. Klassische Fächer in denen Technikverständnis vermittelt werden könnte, sind zweifellos Physik und Chemie, die aber in der Beliebtheitsskala eher am unteren Ende liegen. Technikgeschichte ist im Rahmen des Geschichtsunterrichts – wenn überhaupt – nur ein Nebenaspekt. Verbleibt der Informatikunterricht, der weder von Lehrenden noch von Lernenden unmittelbar als Technikunterricht gesehen wird. Allerdings ist auch hier ein Wandel festzustellen. Während viele Eltern, wohl nicht zuletzt, weil sie diesen Unterricht in ihrer Schulzeit selbst noch nicht angeboten bekamen, hohe Erwartungen in diesen Unterricht setzten und Schulen sich durch ein Wahlangebot in Informatik zu profilieren versuchen, nimmt die Faszination der Schüler und insbesondere der Schülerinnen an diesem Fach zusehends ab[1,2]. In der von Antonitsch und Kolleginnen präsentierten Studie[1] zeigte sich, dass insbesondere bei weiblichen Jugendlichen Informatik mit Klischees assoziiert wird, die teils in Filmen, teils in Kriminalromanen aufgebaut und transportiert werden. Was tun? Da Klischees meist unbewusst verankert sind, ist es schwierig gegen sie anzukämpfen.

Mein *heureka* entstand, als eine Krankenschwester, Sr. Martina, der ich mithin meinen Anteil an dieser Publikation widme, verwundert gestand, dass sie es gar nicht verstehen kann, wie ich mich für ein so fades Fach wie Informatik so sehr interessieren kann. Informatik ein fades Fach? Warum? Sie berichtete mir dann von dem Informatikunterricht, den sie genoss und der, laut der von Micheuz durchgeführten Umfrage[3] an Österreichs Schulen dominiert. Geleitet durch das ECDL-Curriculum lernen Kinder, die sich selbst beibrachten oder von älteren Geschwistern und Freunden gelernt haben, wie man ein Handy bedient, einen Videorecorder programmiert und möglicherweise auch wie man mit anderen via E-Mail in Kontakt tritt und sich spannende Dinge aus dem Internet „herunterlädt", wie man mit Computern bzw. mit spezifischen Computer-Anwendungsprogrammen

1 Antonitsch A., Krainer L., Lerchster R., Ukowitz M.: *Kriterien der Studienwahl von Schülerinnen und Schülern unter spezieller Berücksichtigung von IT-Studiengängen an Fachhochschule und Universität;* IFF-Forschungsbericht, Universität Klagenfurt, März 2007.

2 Schulte C., Knobelsdorf M.: *Das informatische Weltbild von Studierenden;* in: Schubert S. (Hrsg.): *Didaktik der Informatik in Theorie und Praxis;* Proc. INFOS 2007, LNI, Gesellschaft für Informatik, Bonn, 2007, Seite 69-79.

3 Micheuz P.: *Some Findings on Informatics Education in Austrian Academic Secondary Schools,* in: Informartics in Education, 2008, Vol. 7(2), pp. 221-236; Institute of Mathematics and Informatics, Vilnius.

umgeht. Hier besteht die Gefahr, dass Fertigkeiten unterrichtet werden, die intellektuell nicht reizvoll sind. Schlimmer noch, Jugendliche könnten meinen, dass mit Blick auf bestimmte Berufsgruppen diese Fertigkeiten durch Auswendiglernen ohne zu Grunde liegendem Verständnis auf eine Ebene gezogen werden, die befürchten lässt, dass bloß Prüfungsstoff vermittelt würde. Man sieht sich also gezwungen, Details über den Umgang mit einer Maschine zu lernen, zu der man eigentlich keine Beziehung aufgebaut hat und mit der man daher auf dieser Detailebene gar nicht umgehen möchte. Unterricht dieser Art kann also als Beweis dafür herangezogen werden, dass man doch für die Schule und nicht für's Leben lernt.

Mein Gespräch mit Sr. Martina zeigte mir, dass Informatikunterricht – ja eigentlich Technikunterricht insgesamt – unter ein neues Grundthema gestellt werden muss, um die Neugier und Kreativität jener Jugendlichen zu erwecken, die durchaus gerne nachdenken, aber sich nicht den Eigenheiten der Bedienungsschnittstelle einer Maschine unterwerfen wollen. Ich entwickelte erst einmal ein Konzept für eine Didaktik-Lehrveranstaltung, in der gemeinsam mit den Studierenden Formen des Informatikunterrichts ohne Verwendung des Computers ausgearbeitet werden sollten. Während dieser Planungen erreichte mich die Bitte von Dr. Gabriele Meßner-Mitteregger, für den IT-Campus ein Konzept für Informatikeinheiten für Volksschulkinder zu entwickeln. Erst etwas skeptisch, ob dies nicht zu früh ist, versicherte ich mich der Unterstützung von Mag. Ernestine Bischof, die eben eine Arbeit über Informatikunterricht an Volksschulen abgeschlossen hatte. Unter Mithilfe eines Lenkungsausschusses bestehend aus Mag. Peter Holub, Dr. Elvira Sematon und Mag. Beatrix Schönet sowie Dekan Martin Hitz arbeiteten Frau Bischof und ich ein Konzept von Lehreinheiten unterschiedlichen Anspruchsniveaus aus, die in der Folge an diversen Schultypen und Schulstufen erprobt und optimiert wurden. Unser Dank gilt den Mitgliedern des Lenkungsausschusses und insbesondere Herrn Holub für den graphischen Denkanstoß auf Seite 2 sowie allen Lehrkräften und Direktionen, die in der Erprobungsphase jeweils zwei Unterrichtsstunden für unsere Schnuppereinheiten zur Verfügung stellten.

Das vorliegende Heft, *„Informatik erLeben"* ist das Ergebnis dieser im Sommersemester 2008 durchgeführten Probeläufe. Es stellt einige dieser Einheiten als Demonstrationsbeispiele vor. Weitere Einheiten sind über ein Internet-Portal für Lehrerinnen und Lehrer beziehbar[4]. Sie finden dort auch eine Beschreibung der Grundprinzipien, auf denen unser Ansatz basiert, sowie einen Aufsatz, der Diskussionsanregungen über die Wechselwirkungen zwischen Technik und Gesellschaft enthält.

Mit der Bezeichnung *„Informatik erLeben"* wollen wir klar machen, dass mit den in diesem Projekt vorgestellten Lehreinheiten kein umfassender Informatikunterricht angeboten wird und auch kein neuer Prüfungsstoff zur Verfügung gestellt werden soll. Es soll hier Technik in der Form von Informatik als Ergänzung zum Normalunterricht *erlebbar* vermittelt werden. Vor allem soll zum Nachdenken über grundlegende technische Prinzipien angeregt werden. Dies kann in der Form „heute machen wir das einmal anders" erfolgen. Wir erhoffen allerdings, dass dieses „heute einmal anders" als Anregung dient, ähnliche didaktische Überlegungen, in Ihren normalen Informatikunterricht einfließen zu lassen.

Roland Mittermeir
Klagenfurt, im August 2008

4 Sie können in diesem Portal unter http://informatik-erleben.uni-klu.ac.at schnuppern und Materialien beziehen. Wenn Sie sich als Lehrperson registrieren, haben Sie darüber hinaus die Möglichkeit mit der Community der Nutzerinnen und Nutzer dieser Einheiten wie auch mit uns direkt in Kontakt zu treten.

Schnuppereinheiten

Die folgenden „Schnuppereinheiten“ sollen Ihnen den „Geschmack“ des in *Informatik erLeben* entwickelten Materials vermitteln. Dieses gliedert sich in drei Ebenen:

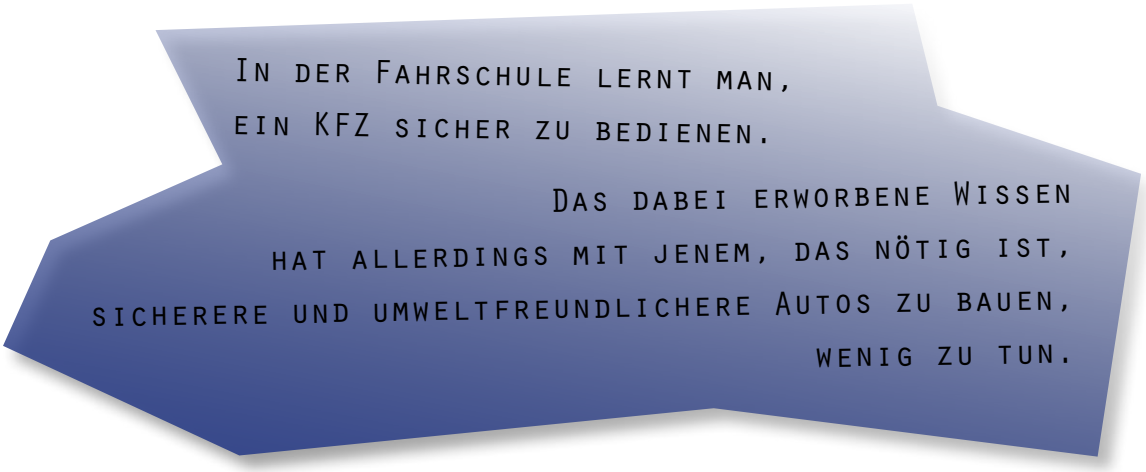
- fachliche Stränge,
- Lehreinheiten,
- Module.

Die Entwicklung der Materialien ging von jeweils zweistündigen Lehreinheiten aus. Die Mehrstufigkeit erlaubt Ihnen jedoch die Zusammenstellung eigener Einheiten aus den vorgegebenen kleineren Modulen in Abhängigkeit von bereits vermittelten Inhalten und in Abhängigkeit von dem bereits mit Ihrer Klasse behandelten Stoff.

Im ersten Teil stellen wir zwei Module aus dem Themenbereich **Codierung** vor, die zu einer Einheit verbunden werden können. Die Module C1 und C3 wurden gewählt, weil C3 etwas anspruchsvoller ist als der über das Web-Portal zugreifbare Modul C2, der ebenso mit C1 kombiniert werden kann. Wir stellten allerdings fest, dass die hier vorgestellte Kombination durchaus bereits von Volksschulklassen erarbeitet werden konnte und auf Interesse stieß.

Anschließend präsentieren wir das Themengebiet **Verschlüsselung**. Obzwar dies ein komplexes Thema ist, kann es in dieser Form bereits mit Volksschulkindern durchgespielt werden, da das spielerische Element dominiert. Der Modul kann sehr gut in eine Einheit über Datenschutz und Schutz der Privatsphäre eingebunden werden. „Wollt ihr, dass jede Nachricht die ihr sendet, von eurer Lehrerin gelesen werden kann?“ oder für Ältere „Wie kann man sicherstellen, dass bei einem Internet-Einkauf per Kreditkarte die Kreditkartennummer nicht in falsche Hände gerät?“.

Der letzte Block ist eine Einheit zu **Sortieren**. Hier nähern wir uns einem informatischen Kernthema, der Algorithmik und der Bewertung von Algorithmen. In ihrer Grundform kann diese Einheit durchaus bereits in der Sekundarstufe I gebracht werden, sie kann aber so ausgebaut werden, dass sie bis in höhere Klassen der Sekundarstufe II interessant ist. Wir präsentieren hier nicht nur zwei unterschiedliche Sortiermodule, sondern auch die übergeordnete Einheit um zu betonen, dass insbesondere bei älteren Jugendlichen die Motivation für die jeweilige Problemstellung ein wesentliches Element der Gesamtkonzeption ist, das entsprechend berücksichtigt werden sollte.



Modul C1 Morsespiel

Zeitrahmen
20 Minuten

Zielgruppe
Volksschule

Lehrziel
Verständnis, dass Zeichen aus unserem Alphabet anderen Zeichen zugeordnet werden können (Grundprinzip der Codierung)

Requisiten
- Eine Taschenlampe je Kind (möglich auch mit Holzstäben, wobei hier allerdings die Unterscheidung zwischen lang und kurz schwieriger ist und die Kinder ihre Zeichen nicht gleichzeitig „senden" können)
- Abschnitte mit der Zuordnung Buchstabe-Morsezeichen

Der Morseapparat wurde 1833 von Samuel Morse erfunden.

Partizipanden
Spielgruppe: 8-12 TN (gerade Anzahl!)
Beobachter: alle anderen Kinder der Klasse

Unterlagen
C_AB1 und C_AB2

Vorgehensweise
1. Sollten mehr als 12 TN sein, stellen die restlichen Kinder eine Beobachtergruppe dar, die die Arbeitsanweisungen nicht erfährt und in der Beobachtung herausfinden soll, was geschieht (→ siehe Arbeitsblatt C_AB1_Arbeitsanweisungen). Die Beobachter verlassen inzwischen die Klasse und hören somit auch die Arbeitsanweisungen für das Spiel nicht!
2. Jeder TN aus der Spielgruppe bekommt eine Taschenlampe und einen Buchstaben aus dem Morsealphabet zugeordnet und muss diesen zunächst geheim halten. Jeweils zwei TN haben den gleichen Buchstaben. Idealerweise wird das Klassenzimmer abgedunkelt.
3. Die TN prägen sich ihren Buchstaben ein und verteilen sich wahllos im Klassenraum. Möglicherweise ist ein Probedurchgang mit einem Beispielcode notwendig, der vom Übungsleiter (ÜL) vorgezeigt wird.
4. Jeder TN gibt nun das ihm zugeordnete Morsezeichen mit der Taschenlampe. Während die Zeichen gegeben werden, darf nicht gesprochen werden.
   Achtung: Damit kein Chaos entsteht, sollte das Übermitteln der Zeichen vom ÜL synchronisiert werden (etwa durch ein Handsignal/Klopfzeichen).
5. Wenn der Partner mit dem gleichen Zeichen gefunden wurde, stellen sich die beiden TN nebeneinander und warten, bis alle fertig sind.

6. Die Teilnehmer der Beobachtergruppe schreiben gemeinsam mit der/dem ÜL einige Beispiele für die Zuordnung von Morsecode und Buchstabe auf.
7. Als Abschluss erhalten die TN eine kurze Erklärung zum Morsecode, etwa wie die Zuordnung funktioniert, und wo er verwendet wird/wurde.

Morsecode: *Der Morsecode ist ein Verfahren zur Übermittlung von codierten Buchstaben und Zeichen mit einfachen technischen Mitteln. Dabei wird ein konstantes Signal ein- oder ausgeschaltet. Wie nebenstehend ersichtlich, hat jeder Buchstabe unseres Alphabets eine Codierung/Zuordnung in Morsezeichen. Dabei haben häufiger verwendete Buchstaben einen möglichst kurzen Code.*
Der Code kann als Tonsignal, als Funksignal, als elektrischer Impuls mit einer Morsetaste über eine Telefonleitung, mechanisch oder optisch (etwa mit blinkendem Licht) übertragen werden – oder auch mit jedem sonstigen Medium, mit dem zwei verschiedene Zustände (wie etwa Ton oder kein Ton), eindeutig und in der zeitlichen Länge variierbar dargestellt werden können. Man spricht auch von Morsetelegrafie.

A	· —	N	— ·
B	— · · ·	O	— — —
C	— · — ·	P	· — — ·
D	— · ·	Q	— — · —
E	·	R	· — ·
F	· · — ·	S	· · ·
G	— — ·	T	—
H	· · · ·	U	· · —
I	· ·	V	· · · —
J	· — — —	W	· — —
K	— · —	X	— · · —
L	· — · ·	Y	— · — —
M	— —	Z	— — · ·

Morsecodetabelle (ohne Sonderzeichen)

8. Den TN wird dann noch ein Bezug zu ihrer Lebenswelt vermittelt:

Der Morsecode wurde in der Telegrafie, im Funk und in der Schifffahrt verwendet. Heute findet man ihn nicht nur in der Luft- und Schifffahrt, sondern auch in Melodien. Ein bekannter Morse-Klingelton ist der bei Nokia verwendete SMS-Ton „Spezial", der – entsprechend seinem Einsatzzweck – die Buchstabenfolge S M S symbolisiert.
Ähnlich sieht der Morsecode für S O S aus.

Tafel: Schreiben Sie die beiden Buchstabenfolgen inklusive zugehörigem Morsecode untereinander, damit die TN sehen, dass das S immer gleich codiert ist.

S O S · · · — — — · · ·
S M S · · · — — · · ·

Zusätzlich könnten jetzt noch gemeinsam aus dem Morsealphabet Wörter herausgesucht werden, etwa einige Namen der TN.
Unser *Informatik erLeben* Logo (··|·) stellt beispielsweise die Initialen i und e dar, getrennt durch ein Trennzeichen, das im folgenden Abschnitt noch erläutert wird.

9. Zusätzlich zu den beiden Zeichen kurz oder lang ist noch eine Zeichentrennung nötig. Machen Sie die TN darauf aufmerksam, indem Sie ein Beispiel verwenden:

AEIOU	(·—|·|··|——|—·—)
vs	
ENEWDT	(·|—·|·|·——|—··|—)

Praktisch verwendet man hierfür ein Trennintervall. Für das Arbeiten in der Klasse ist es aber sinnvoll etwa durch einen senkrechten Strich ein explizites Trennzeichen einzuführen.

Schreiben Sie die Folge von Signalen zuerst ohne Trennzeichen auf und lassen Sie die TN selbst die Interpretationsmöglichkeiten entdecken.

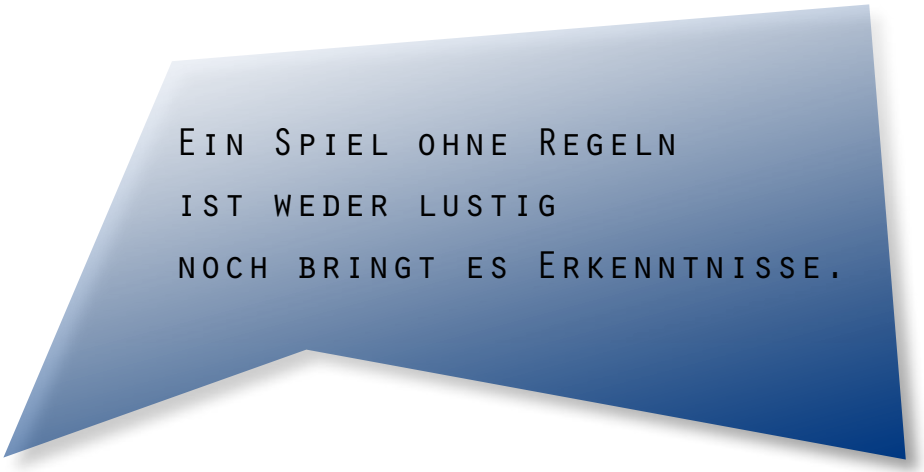
10. Das Morsealphabet kennt nur zwei Zustände – lang und kurz, sowie Trennintervalle (hier: Trennzeichen) wenn der Code übertragen wird.

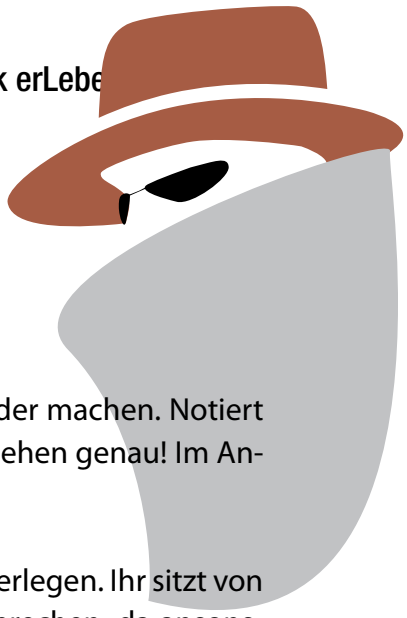
Fallen euch andere Beispiele ein, wo sich zwei unterschiedliche Zustände ergeben?
Denkt daran, wie wir den Morsecode übermittelt haben?
Lichtschalter: ein und aus. Ton: ein und aus, kurz und lang, hoch oder niederfrequent.

Weiterführende Informationen

Morsecode-Übersetzer:
http://morsecode.scphillips.com/jtranslator.html

Vollständiges Morsealphabet:
http://jumk.de/alphabete/morse.shtml





C_AB1_Beobachterrolle

Morsespiel

Beobachtergruppe

Ihr seid Geheimagenten und sollt herausfinden, was die anderen Kinder machen. Notiert euch dafür in der Tabelle einige Merkmale und beobachtet das Geschehen genau! Im Anschluss werden wir dann besprechen, was ihr beobachtet habt.

Bevor ihr mit der Beobachtung beginnt, solltet ihr aber noch etwas überlegen. Ihr sitzt von anderen Geheimagenten etwa drei Meter entfernt und dürft nicht sprechen, da ansonsten die geheime Botschaft abgehört wird. Wie könnt ihr also den anderen Agenten Nachrichten schicken, wenn ihr zusätzlich auch keinen Bleistift und kein Papier zur Verfügung habt.

Vorschläge

.....

.....

.....

.....

Beobachtung

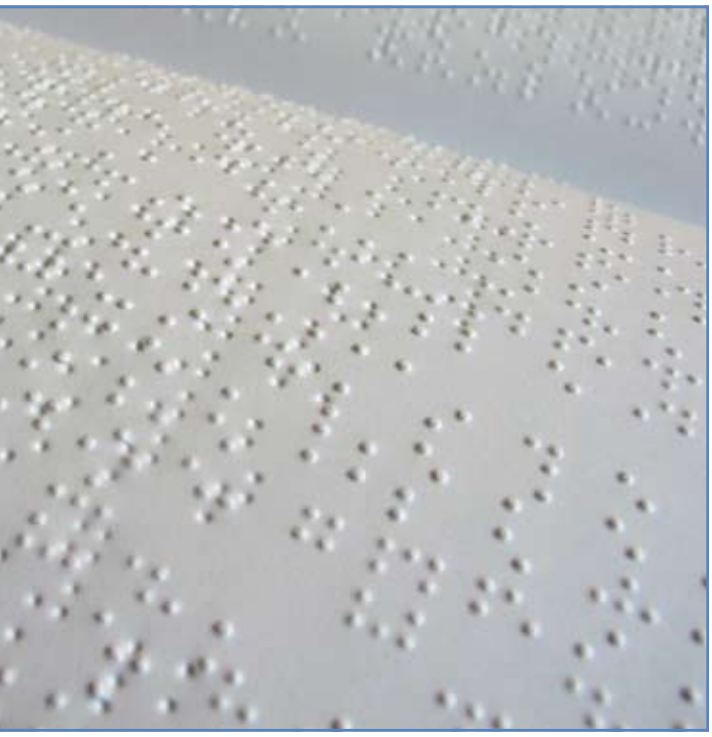
Wie stellen sich die Kinder auf?	
Wie verhalten sie sich? Sprechen sie miteinander?	
Wofür verwenden sie die Taschenlampe?	
Namen der beteiligten Kinder:	

C_AB2

Morsealphabet

A	· —	N	— ·
B	— · · ·	O	— — —
C	— · — ·	P	· — — ·
D	— · ·	Q	— — · —
E	·	R	· — ·
F	· · — ·	S	· · ·
G	— — ·	T	—
H	· · · ·	U	· · —
I	· ·	V	· · · —
J	· — — —	W	· — —
K	— · —	X	— · · —
L	· — · ·	Y	— · — —
M	— —	Z	— — · ·

Modul C3 Codierung und Codebäume



Zeichen tragen die Bedeutung, die wir ihnen geben.
Foto: Braille-Schrift.

Zeitrahmen
20 Minuten

Zielgruppe
Volksschule

Lehrziel
Verständnis, dass Zeichen aus unserem Alphabet anderen Zeichen zugeordnet werden können (Grundprinzip der Codierung), Untermauerung durch bekannte „Codierung" aus dem Alltag

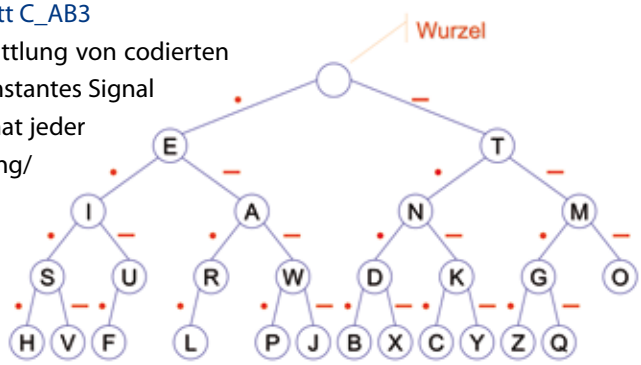
Inhaltliche Voraussetzung
Morsespiel

Requisiten
Codebaum auf Plakat

Unterlagen
C_AB2, C_AB3

Vorgehensweise

1. Gehen wir noch einmal auf den Morsecode ein, um zu sehen, welche andere Möglichkeit es gibt, diesen noch darzustellen. → Arbeitsblatt C_AB3
   Der Morsecode ist ein Verfahren zur Übermittlung von codierten Buchstaben und Zeichen. Dabei wird ein konstantes Signal ein- oder ausgeschaltet. Wie oben gezeigt, hat jeder Buchstabe unseres Alphabets eine Codierung/ Zuordnung in Morsezeichen.
   Wir sehen hier, welche Codezeichen des Codebaumes den einzelnen Buchstaben zugeordnet sind. Man startet bei der Wurzel. Die einzelnen Kreise mit den Buchstaben werden Knoten bzw. am Ende jedes Pfades auch Blätter genannt, so wie bei einem richtigen Baum. Folgt man also den Knoten bis zum gewünschten Buchstaben, so erhält man den zugeordneten Morsecode. Knoten am Ende des Baumes (in unserem Fall also H, V, F, L, P, J, B, X, C, Y, Z, Q, O) werden Blätter genannt.
   Nun werden die TN durch Fragen zur oben erklärten Baumstruktur hingeführt.



Codebaum des Morsecodes
Beispiel: Codewort für D = – · ·

- Welcher Code ergibt sich für den Buchstaben E?
- Welcher Code ergibt sich für Q?
- Was fällt euch dabei auf? Warum sind die Codes unterschiedlich lang?

In jeder Sprache ist das Auftreten gewisser Buchstaben wahrscheinlicher als das von ande-

ren Buchstaben. E ist einer der häufigsten Buchstaben. Somit wurde darauf geachtet, dass E einen möglichst kurzen Morsecode hat. Q hingegen kommt sehr selten vor und hat somit einen längeren Morsecode zugeordnet.

- Welcher Vorteil ergibt sich dadurch? Denkt an die Person, die die Zeichen übertragen muss.
  Weniger Aufwand bei der Übertragung; insgesamt gesehen sind somit die Codefolgen kürzer, als wenn es umgekehrt wäre, dass also Q nur ein Zeichen und E vier Zeichen hätten.

2. Als Hinführung zur Informatik wird hier erstmals das binäre Zahlensystem angesprochen:

Zwei Zustände werden auch im Computer genutzt. Dargestellt werden diese zwei Zustände durch 0 und 1. Diese bedeuten, dass entweder Strom fließt oder nicht, ähnlich wie beim Lichtschalter. Der Computer arbeitet nur mit 0 und 1. Man könnte also auch die Buchstaben vom Morsecode mit 0 und 1 darstellen, anstatt lang und kurz zu verwenden. Der Morsecode wurde entwickelt, da es nur die Möglichkeit gab mit Strom/elektrischen Impulsen Zeichen über längere Strecken zu senden.

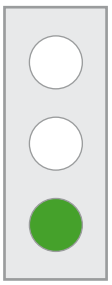
Ergänzung (falls das Morsespiel nicht durchgeführt wurde): Zusätzlich zu den beiden Zeichen kurz oder lang ist noch ein Trennintervall nötig. (Es ist hier durch einen senkrechten Strich als Trennzeichen angedeutet). Machen Sie die TN darauf aufmerksam indem Sie ein Beispiel verwenden:

AEIOU	(·—|·|··|———|··—)
vs	
ENEWDT	(·|—·|·|·——|—··|—)

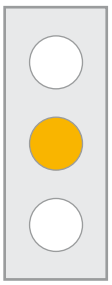
Schreiben Sie die Folge von Signalen zuerst ohne Trennzeichen auf und lassen Sie die TN selbst die Interpretationsmöglichkeiten entdecken.

3. Ein weiteres Beispiel für die Zuordnung von Zeichen und Bedeutungen sind Verkehrsampeln.
   → Arbeitsblatt C_AB3
   Versuchen wir einmal gemeinsam herauszufinden, welche Bedeutung die Farben haben können.

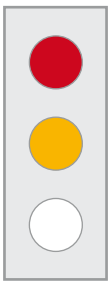
Rot:
Stehenbleiben



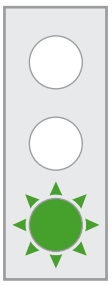
Grün:
Freie Fahrt



Gelb:
Stopp, schaltet auf Rot



Rot + Gelb:
Bereitmachen zum Losfahren



Grün blinkend:
Schaltet bald auf Gelb

Gelb blinken gibt es auch noch.

Gäbe es hier noch Möglichkeiten, die nicht im jetzigen System berücksichtigt sind?
Ja, z.B. Rot und Grün, Rot blinken

Auch diesen Möglichkeiten könnte man wieder eine Bedeutung zuordnen. Hier habe ich drei Lichter für vier Botschaften.

Könnte ich mit weniger Lichtern auch auskommen?

Die untenstehende Tabelle zeigt die Möglichkeiten.

0 = leuchtet nicht

1 = leuchtet

	Licht gelb	Licht grün
Fahren	1	1
Vorsicht	1	0
Bereit	0	1
Stehen	0	0

Warum wählt man nicht diese einfachere Codierung bei Verkehrsampeln?

Redundanz bringt Sicherheit, hilft bei schlechter Sicht, Ausfall einer Lampe, …

Quellen/Weiterführende Literatur

Gallenbacher, Jens: *Abenteuer Informatik. IT zum Anfassen von Routenplaner bis Online-Banking.* Spektrum Akademischer Verlag, München, 2007.

Bollin, Andreas; Eder, Johann; Hitz, Martin: *Skriptum Einführung in die Informatik*, Institut für Informatik-Systeme, Universität Klagenfurt, 2008.

Gumm, Heinz-Peter; Sommer, Manfred: *Einführung in die Informatik.* Oldenbourg Wissenschaftsverlag, München, 2002.

WANN KANN MAN EINER FEHLERHAFTEN NACHRICHT DENNOCH DIE RICHTIGE BOTSCHAFT ENTNEHMEN?

C_AB3

Codierung

Mein Name: .....

Mein Name in Morsecode: .....

Mein Name in der eigenen Codierung: .....

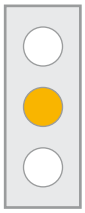
Hier erstellen wir unseren eigenen Code:

A		J		S	
B		K		T	
C		L		U	
D		M		V	
E		N		W	
F		O		X	
G		P		Y	
H		Q		Z	
I		R			

Zuordnung von Zeichen und Bedeutungen am Beispiel einer Verkehrsampel:

Signal: .....

Bedeutung: .....



Signal: .....

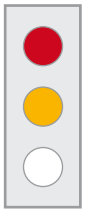
Bedeutung: .....

Signal: .....

Bedeutung: .....

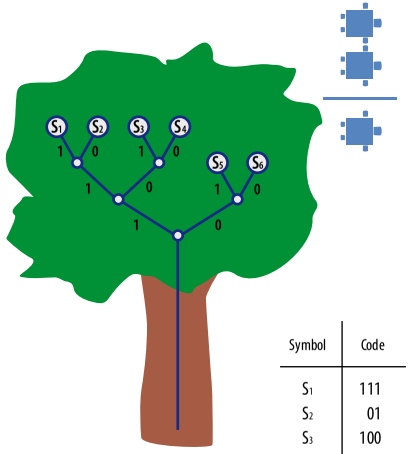
Signal: .....

Bedeutung: .....

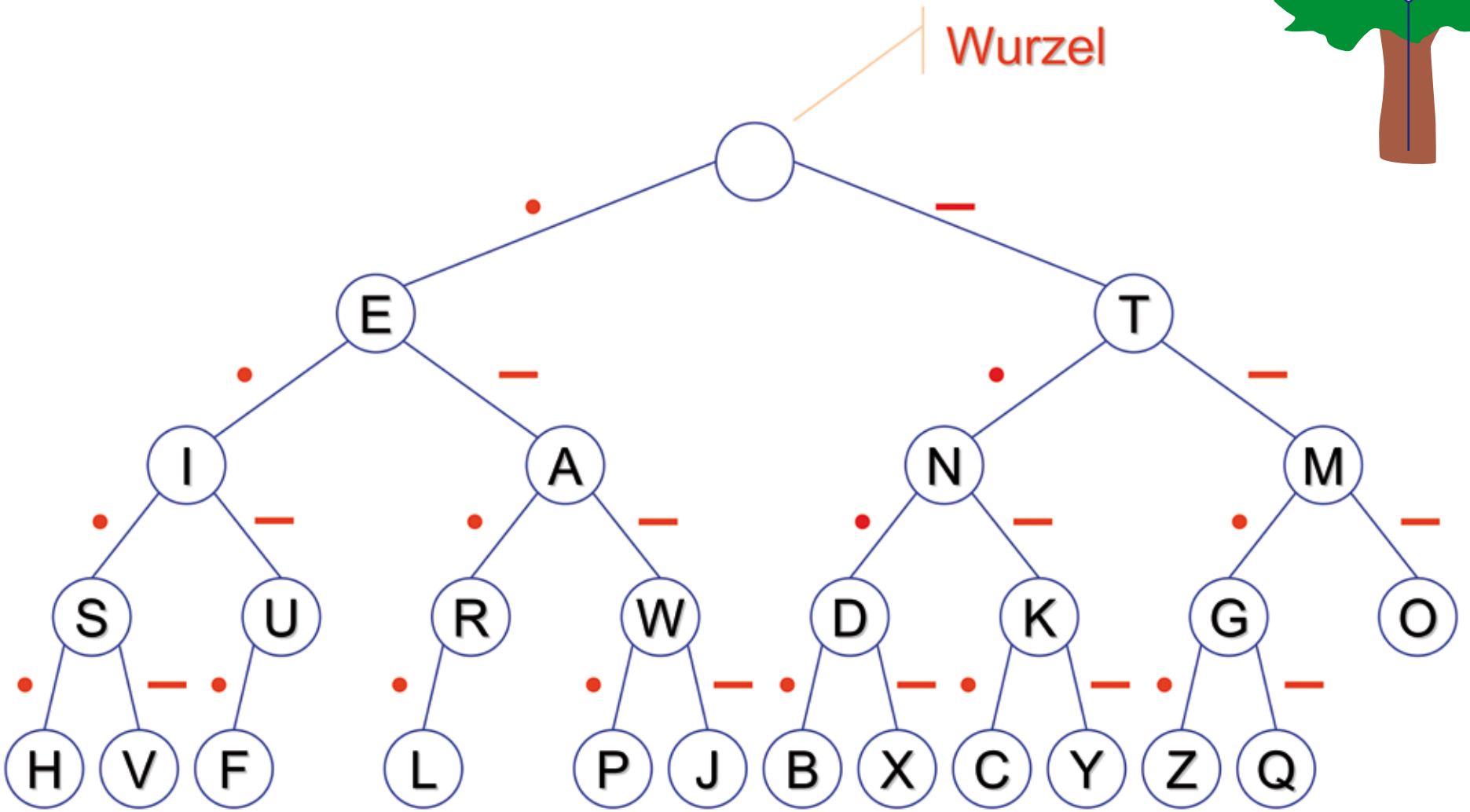


Signal: .....

Bedeutung: .....



Symbol	Code
S_1	111
S_2	01
S_3	100
S_4	101
S_5	110
S_6	00



Beispiel: Codewort von D = — · ·

Modul V1 Verschlüsselung mit der Cäsar-Chiffre

Zeitrahmen
30 Minuten

Zielgruppe
- Volksschule
- Sekundarstufe I

Requisiten
Kärtchen mit dem Alphabet in zweifacher Ausführung auf jeweils unterschiedlicher Grundfarbe.

Lehrziel
- Verstehen des Prinzips der symmetrischen Verschlüsselung
- Symmetrische Verschlüsselungsverfahren kennen lernen

Motivation
Geheime Nachrichten wecken die Neugierde. Die Kryptografie ist ein zentrales Thema der Mathematik/Informatik, war historisch gesehen schon früh von Bedeutung und wird in der heutigen virtuellen Kommunikation auch immer wichtiger.

Mit diesem Verfahren wurden lateinische Nachrichten im römischen Reich verschlüsselt.

Vorgehensweise
1. Als Einstieg wird den TN ein verschlüsselter Text gezeigt, und gefragt, was dieser bedeuten könnte.
   Beispieltext: **Tfgvm Nlitvm Prmwvi!** *(Guten Morgen Kinder!)*
   Die TN werden darüber informiert, dass es sich um einen Geheimtext handelt, so wie etwa bei der Spiegelschrift. Es wurden nach gewissen Regeln Buchstaben vertauscht.
2. Nun werden Verschlüsselungsverfahren betrachtet, die symmetrische Schlüssel verwenden und der obige Text wird aufgelöst.

   Einfache Verschlüsselung: Die Buchstaben des Alphabets werden verschlüsselt, indem sie von hinten beginnend aufgeschrieben werden. Wir verwenden dazu Kärtchen, grüne für den Klartext, rote für den verschlüsselten Text, das Chiffrat.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Ausgangssituation

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Gespiegeltes Alphabet

Die TN erhalten die Aufgabe, die grünen Kärtchen in der richtigen Reihenfolge aufzulegen und darunter dann die roten Kärtchen von hinten beginnend. Somit erhält man jeweils die Zuord-

nungspaare der Buchstaben. A würde also durch das Chiffre Z ersetzt werden, B durch Y usw.

Bei Entschlüsseln des Textes auf der Tafel suchen wir das zur jeweiligen Chiffre (rot) passende Klartextzeichen (grün). Wir finden etwa zum roten *T* das darüber stehende grüne *G*. Nach Entschlüsselung der Worte an der Tafel wird ein Beispielswort gemeinsam verschlüsselt indem wir zu den Klartextbuchstaben (grün) die passende Chiffre (rot) suchen und anschreiben.

Ihr habt einen verschlüsselten Text vor euch liegen, welche Informationen würdet ihr benötigen, um ihn zu entschlüsseln?

Wenn man weiß, dass auf diese Art verschlüsselt wurde, kann man sich die Entschlüsselungsbuchstaben selbst zuordnen.

Diese Art der Verschlüsselung ist also nicht sehr sicher und leicht zu brechen, daher sehen wir uns dann eine etwas komplexere Art der Verschlüsselung an.

3. Wozu wird überhaupt verschlüsselt?

Was würdet ihr sagen, wenn eure SMS, die ihr an Freunde schickt, von den Eltern oder anderen Personen mitgelesen oder sogar verändert werden könnten? Wo könnte eine Verschlüsselung von Nachrichten wichtig sein, wo war sie früher wichtig, sucht konkrete Beispiele?

- Unbefugte dürfen Nachrichten/Daten nicht lesen, z.B. electronic banking, Industriespionage, Kriegsführung, Geheimdienste
  - o Beispielsweise dürfen Informationen über ein neues Produkt einer Firma nicht an die Konkurrenzfirma gelangen.
  - o Nachrichten über geplante Angriffe oder Angriffstaktiken dürfen nicht dem Gegner in die Hände fallen.
  - o Und natürlich würde man sich auch beim privaten E-Mail-Verkehr wünschen, dass der Text nicht für alle mitlesbar ist.

  Entspricht ein E-Mail eher einer Postkarte oder einem Brief?

  Eine Postkarte kann der Briefträger leicht lesen… Aber auch Briefe können unerlaubt geöffnet und wieder verschlossen werden, allerdings ist der Zugriff zumindest erschwert.
- Unbefugte dürfen Daten nicht ändern können (Integrität)
- Damit im Zusammenhang auch der Nachweis der Urheberschaft (Authentifikation)

Verschlüsselung steht somit auch in einem engen Zusammenhang mit dem Themenbereich Sicherheit.

4. **Cäsar-Chiffre**

Den TN wird nun ein Beispieltext gezeigt. Sie werden gefragt, ob dieser mit der oben genannten Methode entschlüsselt werden kann. Bevor das Grundprinzip anhand von Kärtchen erläutert wird, werden die TN gefragt, ob sie bereits von der Cäsar-Verschlüsselung gehört haben, wenn ja sollen diese TN bereits eine Erklärung dazu abgeben, was sie darunter verstehen (eventuelle Richtigstellung durch ÜL). Für die weitere Veranschaulichung werden jeweils zwei Kärtchenreihen mit dem Alphabet durch 2 TN aufgelegt. Die zweite Reihe wird von den TN um 2 Stellen verschoben, wobei die letzten Kärtchen jeweils wieder am Anfang eingefügt werden. Sind genug TN vorhanden, also mindestens 26 so kann jedes Kind zwei Karten halten und die Karten werden an den übernächsten Nachbarn weitergegeben. Der Schlüssel ist also in diesem Fall 2.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Ausgangssituation

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

Verschobene Kärtchen

Somit erhält man nun die Zuordnung für die Ver- und Entschlüsselung. Wieder wird damit ein kurzes Beispielswort ver-/entschlüsselt (an der Tafel).

Cäsar-Chiffre (Schlüssel 2)

guten tag ▶ iwvgp vci
hallo kinder ▶ jcnnq mkpfgt
guten morgen kinder ▶ iwvgp optigp mkpfgt
Wir sind eine brave klasse ▶ Ykt ukpf gkpg dtcxg mncuug
das ist eine geheime nachricht ▶ Fcu kuv gkpg igjgkog pcejtkejv

Was muss man wissen, um einen Text, der auf diese Art und Weise verschlüsselt wurde, wieder zu entschlüsseln?

Die Zahl, um wie viel Positionen verschoben wurde, also den Schlüssel.

Wie könnte man die Verschlüsselung auch ohne den Schlüssel zu wissen, brechen (wenn Zeit keine Rolle spielt)?

a) Durch Ausprobieren, um wie viel Stellen verschoben wurde. Das ist in endlicher Zeit machbar
b) Kryptoanalytiker gehen von der Häufigkeit der Buchstaben aus und schließen daraus auf die Verschlüsselung. Man sucht hierbei aus dem verschlüsselten Text den häufigsten Buchstaben heraus und nimmt an, dass dieser dem häufigsten Buchstaben im Alphabet entspricht, dann sucht man den zweithäufigsten usw. Die drei häufigsten Buchstaben im deutschen Alphabet sind E (17,4%), N (9,78%) und I (7,55%).
Auch anhand von Wortanfängen bzw. Wortlängen kann man versuchen Wörter zu entschlüsseln. Beispielsweise kann man die 3 Artikel *der*, *die*, *das* sehr leicht aufgrund der Wortlänge und der gleichen Anfangsbuchstaben erkennen. Aus den Wortanfängen kann man dann darauf schließen, welcher Buchstabe es sein könnte: der-die-das ergäbe FGT-FKG-FCU, lässt man die Leerzeichen weg, kann man die Wörter auf diese Art und Weise nicht mehr so leicht analysieren.

OHNE EINHALTEN GEMEINSAMER KONVENTIONEN KANN KEINE KOMMUNIKATION ENTSTEHEN!

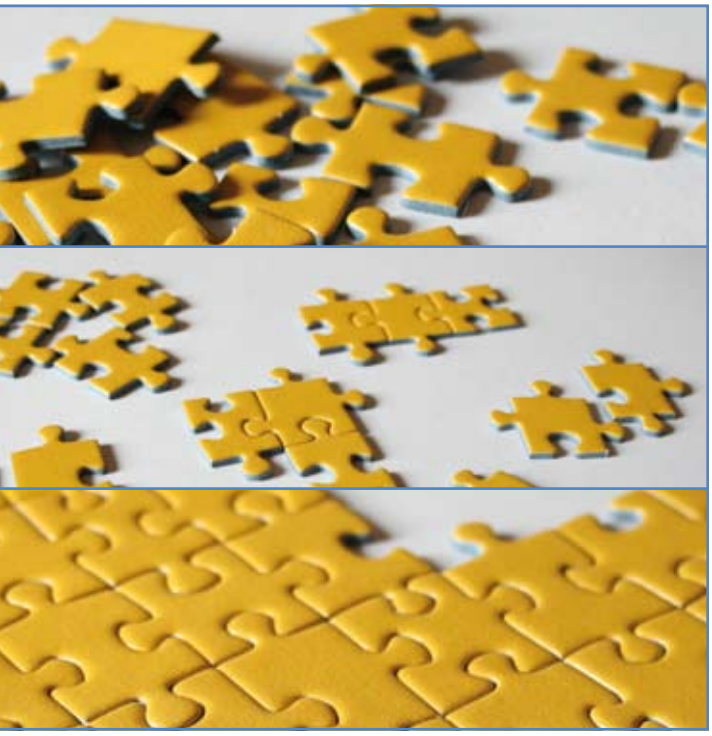
Quellen/Weiterführende Literatur

Hier können Sie eigene Texte mit der Cäsar-Chiffre verschlüsseln lassen:
http://willy.chemie.uni-konstanz.de/fotos/caesar.htm

Singh, Simon: Geheime Botschaften. *Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet*. München: dtv 2004.

Schulz, Ralph-Hardo: *Codierungstheorie. Eine Einführung*. Wiesbaden: Vieweg Verlag 2003.

Lehreinheit E-So1 *Schuften oder Denken?*



Nach welchem System geht man beim Puzzle-bauen vor?

Zeitrahmen
110 Minuten

Zielgruppe
- Sekundarstufe I;
- Sekundarstufe II mit geringen Informatik-Vorkenntnissen

Lehrziel
- Effizienz von Algorithmen
- Zusammenwirken von Datenstrukturen und Algorithmen
- Informatik ::= zuerst denken, dann erst implementieren

Motivation
In vielen Lebenssituationen müssen wir Dinge sortieren. Oft findet dies unter erschwerenden Rahmenbedingungen statt. Wir sortieren Rechnungen oder Belege einmal nach Betrag und kurz darauf nach Datum und haben dazu meist zu wenig Tischfläche vor uns, um alle Objekte mit gebührendem Abstand nebeneinander zu legen. Vor dem Bau eines Dachstuhls sollten die Balken so aufgelegt werden, dass sie passend nach Länge und Verwendungszeitpunkt liegen und beim Bau einer Steinstützmauer sollten die Steine auch so liegen, dass man die nach Fertigstellung einer Reihe noch verfügbaren Steine wieder in einer Anordnung hat, aus der leicht ersichtlich ist, welche Steine als nächste in die Mauer passen.
Die Bewegung großer Blöcke ist aufwändig. Handelt es sich um Gesteinsblöcke ist dies offensichtlich, handelt es sich um Datenblöcke, ist das nicht so offensichtlich. Aber es ist dennoch der Fall und im Computer finden sehr oft Umordnungen von Datenbeständen statt, um effizienter auf einzelne Elemente zugreifen zu können. *(Siehe auch Module und Einheiten zu Suche)*.
Man wird sich in beiden Fällen überlegen, wie man die Zahl der kraftraubenden (Zeit raubenden) Operationen minimiert. Bei den Gesteinsblöcken wird dies die Bewegung der Felsteile sein. Wenn man sie zur Bestimmung der richtigen Anordnung allerdings zuerst vermessen oder abwiegen muss, damit man sie zueinander in Beziehung setzen kann, ist dies auch mit einem nicht zu vernachlässigenden Aufwand verbunden.
Freilich bedeutet dies, dass man die Messergebnisse irgendwie auf die Steine schreiben wird. – Wir nähern uns daher von einem physischen Problem, bei dem die Steine einfach so lange bewegt werden, bis sie in der richtigen Reihenfolge liegen, einem informatischen Problem.
- Die Steine werden vermessen und das Ergebnis angeschrieben.
- Nun wird eine Ordnung zwischen den Vermessungsergebnissen hergestellt und diese Ordnung auf die Position der aktuellen Lage der Steine zurück abgebildet.
- Schließlich werden die Steine so angeordnet, wie es der vorher erstellten Reihenfolge der Messergebnisse entspricht.

Bleibt man bei Steinen, ist klar, dass das Problem einfacher ist, wenn man die Steine vorerst abseits jener Stellen gelagert hat, an denen sie letztlich platziert werden sollen. Dies gilt auch für das Sortieren von Daten. Doch während Steine in der Regel abseits des Bauplatzes abgekippt werden, sind Daten meist in der Struktur, in der sie auch sortiert werden (in situ Sortierverfahren). Dieser Unterschied ist dadurch begründet, dass man bei zu sortierenden Steinblöcken meist wenig Objekte sortieren muss (10, 100, oder vielleicht auch 1.000) während bei Daten meist sehr viele Datensätze (10.000, 100.000, mehrere Millionen) zu sortieren sind.
Die besondere Rolle von Sortieralgorithmen in der Informatik besteht darin, dass man innerhalb vieler Verfahren erst einmal mit Sortierproblemen beschäftigt ist, da es sich mit sortierten Datenbeständen viel einfacher und effizienter arbeiten lässt als mit unsortierten. *(Siehe auch Module und Einheiten zu Suche)*.

Requisiten
Geburtstagsordnung: keine

Partizipanden
Siehe Beschreibung der Partizipanden bei den einzelnen Algorithmen.

Grobstruktur
1. Motivation
2. Aufgabenstellung, ein Sortierverfahren zu entwickeln
   a) für große Gesteinsblöcke
   b) für „normale“ Datensätze
3. Durchspielen eines elementaren Sortierverfahrens
   siehe So1– Bubble-Sort, So2 – Selection-Sort, So3 – Insertion-Sort

Aufgrund des Gesteinsblocks-Paradigmas ist es wahrscheinlich, dass die Schüler für die Gesteinsblöcke von sich aus auf ein Verfahren kommen, das im Wesentlichen dem Selection-Sort entspricht. Man kann hier aber flexibel jenes Verfahren üben, das den Schülervorschlägen am ehesten entspricht. Wesentlich ist bloß, dass im nächsten Schritt die Kosten von Vergleich und Bewegung getrennt analysiert werden und hier für die Systemumgebung der Klasse realistische Zeitangaben getroffen werden.

4. Komplexitätsabschätzung des elementaren Sortierverfahrens
5. Durchspielen eines $O(n \cdot ld(n))$ Sortierverfahrens. Vorschlag: So4 – Mergesort

Für die Gesteinsblock-Analogie passt hier Mergesort (So4) sicherlich besser als Quicksort (So5). Für die informatischen Zielsetzungen kann man jedoch sowohl mit Mergesort oder Quicksort als kontrastierendem Verfahren fortsetzen. Mergesort hat wohl auch den Vorteil, dass es für Jugendliche leichter nachvollziehbar ist und auch in Alltagssituationen, etwa Sortieren von Belegen auf einem Tisch mit relativ kleiner Oberfläche in der Variante eines m-Wege-Merges (Vergleichsoperation nicht zwischen 2 sondern zwischen m Alternativen) stattfinden kann. Quicksort ist demgegenüber das elegantere und intellektuell anspruchsvollere Verfahren, durch das man schön die der Rekursion innewohnende Eleganz zeigen kann.

6. Analyse der Komplexität von Mergesort
7. Abschlussdiskussion mit Vergleich des Sortieraufwands

Detailüberlegungen zur Komplexitätsabschätzung (4. und 6.)
Ausgehend von der Motivation, überlegt man, wie viel die beiden mit Sortieren verbundenen elementaren Operationen an Zeit kosten:

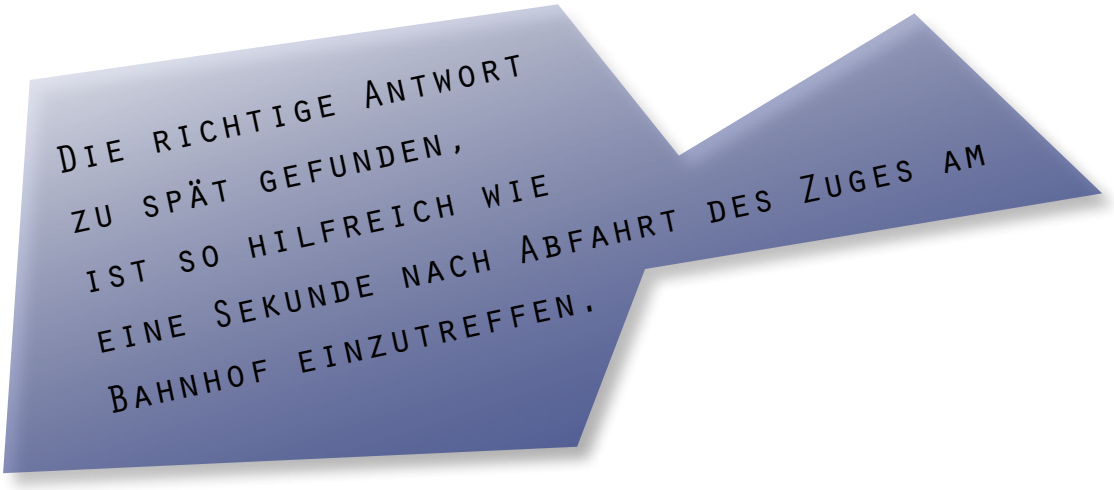
- Vergleich (Wir können im Computer immer nur 2 Objekte vergleichen!) Es handelt sich um eine Elementaroperation im Mikrosekundenbereich. Weniger als 10 μs.
- Tausch (benötigt 3 Schritte *(wenn unklar ist, warum 3 Schritte, dann Dreieckstausch kurz ansprechen)*, der Einfachheit halber, ebenfalls etwa 10 μs.

Unter Rückgriff auf die in Schritt 3 bzw. Schritt 5 entwickelten Tafelskizzen berechnet man auf der Tafel in Schritten, die Nachvollziehbarkeit durch Kopfrechnen erlauben:

Für 100.000 Datensätze:

n Operationen:	$100.000\ 10\mu s = 10^5 \cdot 10 \cdot 10^{-6} sec = 1 sec$
n^2 Operationen:	$100.000 \cdot 100.000 \cdot 10\mu s = 10^5 \cdot 10^5 \cdot 10 \cdot 10^{-6} sec = 10^5 sec =$ $= 100.000 sec = 27{,}7$ Stunden ≈ 1 Tag, 3 Stunden
Zwischenrechnung:	$100.000 \leq 1024 \cdot 128 = 2^{10} \cdot 2^7 = 2^{17}$, daher $ld(100.000) \leq 17$
$n \cdot ld(n)$ Operationen:	$100.000 \cdot 17 \cdot 10\mu s = 10^5 \cdot 10 \cdot 2 \cdot 10 \cdot 10^{-6} sec = 20 sec$

Vergleichen wir dieses Ergebnis (20 sec) mit dem Ergebnis des zuerst angewendeten Sortierverfahrens, sieht man, dass sich der Aufwand, der in die Konstruktion eines leistungsfähigen Verfahrens gesteckt wurde, bei 100.000 und mehr Datensätzen jedenfalls lohnt. Wir können auch abschätzen, dass er sich schon früher (10.000) sicherlich lohnt, während bei Feldern kleinen Umfangs ein rasch entwickeltes elementares Verfahren wohl ausreichend leistungsfähig ist.



Modul So2 Selection-Sort

Zeitrahmen
20 - 30 Minuten (je nach Tiefe der Diskussion)

Zielgruppe
- Volksschule, Sekundarstufe I
- Sekundarstufe II als kontrastierendes Sortierverfahren oder in Erweiterung mit Diskussion über Invarianten.

Lehrziel
- Vermittlung eines einfachen relativ leicht durchschaubaren Sortierverfahrens.
- Basis für eine Diskussion über die algorithmische Komplexität dieses Verfahrens und Frage, ob es nicht bessere Möglichkeiten geben könnte.

Requisiten
Geburtstagsordnung: keine
Sonst: Selbst angefertigte Kärtchen, die einer Halbordnung entsprechen und vor dem Austeilen an die Experimentalgruppe durchmischt werden (zufallsverteilt).

Partizipanden

Experimentalgruppe:	n ≥ 10 Jugendliche
Algoritm. Unterstützung:	1 Taktgeber (sichert, dass sich der Algorithmus in beobachtbarer Langsamkeit entwickelt). 1 Sucher, der jene Person auswählt, die in der noch unsortierten (Rest-) Menge den jeweils kleinsten Geburtstag hat.
Beobachtungs(gruppe):	
Bewegungszähler	Zähler der aufzeichnet, wie oft Personen innerhalb des jeweiligen Durchlaufs den Platz tauschen mussten.
Fragen-Zähler	Zähler, der aufzeichnet, wie oft Sucher in den einzelnen Durchläufen nach den jeweiligen Geburtsdaten fragen musste.
Tiefenzähler	Zähler, der aufzeichnet, wie oft der Sucher die Struktur insgesamt durchlaufen musste *(Anzahl der Durchläufe der äußeren Schleife)*.

Vorgehensweise (Durchspielen des Algorithmus)
Aufbau
1. Experimentalgruppe stellt sich wie im Turnunterricht in einer nach Größe geordneten Riege auf.

Selection-Sort (Riege)
2. Wenn die Riege aus nur einer Person besteht, ist die Riege sortiert. Also, das Aufteilungsverfahren ist FERTIG.

Auf die Sortiertheit der einelementigen Menge muss hier noch nicht besonders abgehoben werden. Wichtig ist hier vielmehr, darauf hinzuweisen, dass jedes Verfahren ein Problem bestimmter Größenordnung benötigt, um überhaupt angewandt zu werden. Beim Sortieren müssen wir vergleichen und dazu bedarf es eben mindestens zweier Elemente.
Während diese Überlegung bei rekursiven Verfahren und Bubble-Sort wegen der Endbedingung motiviert ist, ist dies hier eingangs noch nicht der Fall. Daher kann diese für rekursive Verfahren und auch für Bubble-Sort wichtige Feststellung bei Selection-Sort (wie auch bei Insertion-Sort) ggf. an dieser Stelle entfallen, da man in der Experimentalgruppe ja jedenfalls mehr als eine Person hat.
Allerdings geht die Abbruchbedingung der äußeren Schleife (Schritt 4) sowie die Diskussion über Invarianz und Variante des Verfahrens letztlich doch wieder davon aus, dass eine ein-elementige Menge (Minimum am Anfang, Maximum am Ende des Verfahrens) sortiert ist. Erweitert man das Verfahren allerdings bis zum Programmieren, ist sie auch wichtig, weil sonst der Aufruf der Sortierprozedur mit einer leeren oder einelementigen Menge unweigerlich zu Problemen führen würde.

3. Mit der unsortierten (Teil-)Riege werden folgende Anweisungen ausgeführt:
   3.1. Die an der linksten Position stehende Person tritt einen Schritt vor. Damit wird diese Position frei.
   3.2. Der Sucher erfragt das Geburtsdatum der vorgetretenen Person und merkt sich dieses als vorläufig aktuell kleinstes sowie die zugehörige Person als aktuellen Tauschkandidat.
   3.3 Der Sucher wandert sodann am noch unsortierten Teil der Riege nach rechts indem er jede einzelne Person nach dem Geburtsdatum fragt.
   Wenn dieses kleiner ist als das aktuell kleinste, so merkt sich der Sucher nun dieses als aktuell kleinstes sowie die zugehörige Person als aktuellen Tauschkandidat.
   3.4. Sobald der Sucher das Geburtsdatum der am weitesten rechts stehenden Person erfragt hat und auch geprüft hat, ob diese Person allenfalls neuer aktueller Tauschkandidat ist, bricht er seine Wanderschaft ab und ersucht den nunmehr feststehenden aktuellen Tauschkandidaten an das linke Ende der unsortierten (Teil-)Riege zu treten.
   Merke: Es kann auch sein, dass die von der am weitesten links liegenden Position hervorgetretene Person bis zum kompletten Durchlauf der Riege aktueller Tauschkandidat bleibt. Dies ändert nichts am Verfahren, dann tritt eben diese Person in die Lücke ihrer ehemaligen Position zurück.

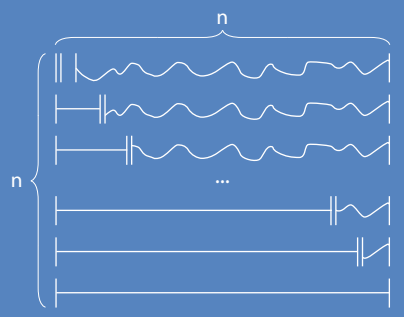
   Sollte sich dieser Fall nicht zufällig im Laufe des Gesamtalgorithmus ergeben, empfiehlt es sich, diese Situation mit der Klasse im Rahmen der Diskussion des Verfahrens zu besprechen.

   3.5. Dadurch ist nun die Position des aktuellen Tauschkandidaten (also der Person mit dem nächst kleinsten Geburtstag) frei geworden. Sie wird von der Person eingenommen, die in Schritt 3.1 die linkste Position frei gemacht hat.

Bei der Überleitung zum Pseudocode wird zu besprechen sein, dass Schritt 3.4, der hier entfallen kann, wenn die linkste Person auch jene mit dem kleinsten noch verfügbaren Geburtstagsdatum war, durchaus als Dreieckstausch ohne Wirkung durchgeführt wird. Dies ist effizienter als eine explizite Behandlung des Sonderfalls.
Die Sonderfallbehandlung entsteht hier lediglich dadurch, dass der Dreieckstausch zwischen Linkstem und Kleinstem auf mehrere Schritte (3.1, 3.4 und 3.5) aufgeteilt wurde. Wir empfehlen dies allerdings, da durch 3.1 die besondere Rolle des jeweils linksten Elements, sowie die Trennung zwischen sortiertem und unsortiertem Teil der Riege für alle unmittelbar ersichtlich wird.

4. Der Sucher geht nun zur Position jener Person, die unmittelbar rechts von der in 3.4 eingefügten Person ist.
   Wenn diese Person bereits die am rechtesten Rand stehende Person ist, gilt für den unsortierten Bereich die in Schritt 2 zu prüfende Bedingung und wir sind FERTIG;
   Sonst, führen wir das Verfahren ab Schritt 3.1 nochmals durch.
5. Wir beobachten, dass durch die Kombination 3.3 und 3.4 beim ersten Durchlauf dieses Verfahrens die Person mit dem insgesamt kleinsten Geburtstag an den Anfang der Riege gestellt wurde. Ab diesem Zeitpunkt war die Riege stets in einen sortierten und einen noch unsortierten Teil gegliedert, wobei durch jeden Durchlauf der sortierte Teil um eine Person gewachsen, der unsortierte entsprechend um eine Person geschrumpft ist.

Tafelskizze zu Selection-Sort



		Beweg.	Vergleiche
	0	3	n-1
	1	3	n-2
	2	3	n-3
	...	...	...
	n-2	3	1
	n-1	3	0
=	n	-	-
		3·(n-1)	(n-1)·(n-1+1)/2

Analyse des Algorithmus

- Schritt 5 des Verfahrens ist ja kein Verfahrensschritt mehr, sondern ein Analyseschritt. Da davon auszugehen ist, dass nicht alle TN diese Beobachtung gemacht haben, empfiehlt es sich, die Analyse des Algorithmus mit einer Diskussion über diese Beobachtung zu beginnen.

Stimmt diese Aussage/Beobachtung? Warum stimmt sie?
Warum musste im letzten Durchlauf die am weitesten rechts stehende Person nicht mehr geprüft werden? Warum konnten wir ohne weitere Prüfung annehmen, sie sei jene mit maximalem Geburtstagswert?

- Welche Beobachtungen wurden von der Beobachtungsgruppe gemacht:

- Wie viel Fragen mussten beantwortet werden?
- Wie viel Bewegungen haben stattgefunden?
- Hat sich die Anzahl der Bewegungen oder der Fragen im Zuge der Durchläufe geändert? Tafelskizze zur Veranschaulichung!
- Wie oft musste der in 3. beschriebene Ablauf durchlaufen werden? Wird das immer so sein? Wie oft muss er mindestens, wie oft maximal wiederholt werden? Tafelskizze zur Veranschaulichung!
- Wie lassen sich diese Beobachtungen auf die Zahl der Personen in der Experimentalgruppe umlegen? Wie würden die Zahlen aussehen, wenn in der Experimentalgruppe 10, 100, 1000 Personen gestanden wären.

Aus der Tafelskizze (gerade Linien für sortierten Bereich, Wellenlinie für unsortierten Bereich mit schräg absteigender Verkürzung der (unsortierten) Wellenlinien und gleichzeitig wachsenden geraden (sortierten) Linien sollte eine Basis für das Eintragen der Werte der Beobachter und letztlich die Basis für die Erklärung von $O(n^2)$ Abfragen bei $O(n)$ Bewegungen gegeben werden.

Wenn sich die Fragen anhand der physischen Übung und der schematischen Tafelskizze nicht ausreichend beantworten lassen, kann man noch eine weiteres Tafelbild mit Zahlen (oder Namen) wie rechts dargestellt nachreichen.

- Anhand der Tafelskizze feststellen, dass unabhängig vom Grad der Vorsortierung zwischen $O(n^2)$ Abfragen und $O(n)$ Platztauschoperationen benötigt werden.
- Skizzierung des Algorithmus in Pseudocode mit expliziter Schleifenstruktur. Erkennen der Wiederholungsgesetzmäßigkeiten aus den geschachtelten Schleifen. Daraus:
  - o Erkennen der zunehmenden Sortiertheit der Riege und der Rolle der Endbedingung.
  - o Besprechen, warum es sinnvoll ist, den finalen Dreieckstausch auf alle Fälle durchzuführen.
  - o Wenn entsprechende Vorkenntnisse aus Mengenlehre gegeben sind, könnte man anhand der Skizze des Algorithmus in Pseudocode auch in eine Diskussion über Invarianten eintreten.

Pseudocode für Selection-Sort

```
Selectionsort (Feld, links, rechts) {
        VAR:    sucher, posAktuellesMinimum: INDEXTYP
                hilfsStelle: FELDELEMENTTYP

        SOLANGE links < rechts {
                posAktuellesMinimum := links;
                sucher := links;
                SOLANGE sucher ≤ rechts {
                        WENN Feld[posAktuellesMinimum] > Feld[sucher] DANN
                                posAktuellesMinimum := sucher;
                        sucher := sucher + 1
                }
                hilfsStelle := Feld[links];
                Feld[links] := Feld[posAktuellesMinimum];
                Feld[posAktuellesMinimum] := hilfsStelle;
        }
}.
```

Weiterführende Literatur

Ottmann Th., Widmayer P.: *Algorithmen und Datenstrukturen*; BI Wissenschaftsverlag, Mannheim, 1993.

Li Liwu.: *Java: Data Structures and Programming*; Springer Verlag, 1998.

Modul So4 Mergesort

Zeitrahmen
40 Minuten

Zielgruppe
Sekundarstufe I

Lehrziel
Vermittlung eines effizienten, relativ leicht durchschaubaren Sortierverfahrens mit Ausblicken auf Rekursion und rekursive Datenstrukturen (natürlicher Binärbaum).

Requisiten
Geburtstagsordnung: keine

Partizipanden

Experimentalgruppe:	n ≥ 10 Jugendliche
Algoritm. Unterstützung:	1 Aufteiler (stellt jeweils die Mitte fest und sichert, dass sich beide Hälften solange dies möglich ist, weiter teilen) 1 Vergleicher (Steuert die Mischoperation)

Beobachtungs(gruppe):

Mischzähler	Zähler der aufzeichnet, wie oft eine Person im Zuge der Misch- und Rücksetzoperation den Platz tauschen musste.
Fragen-Zähler	Zähler, der aufzeichnet, wie oft Personen nach den jeweiligen Geburtsdaten fragen mussten.
Tiefenzähler	Zähler, der aufzeichnet, wie oft die Struktur insgesamt durchlaufen werden musste (Rekursionstiefe).

Vorgehensweise (Durchspielen des Algorithmus)
Aufbau

1. Experimentalgruppe stellt sich wie im Turnunterricht in einer nach Größe geordneten Riege auf.

Mergesort (Riege)

2. Wenn die Riege aus nur einer Person besteht, ist die Riege sortiert. Also, das Aufteilungsverfahren 2a. FERTIG. Setze bei Schritt 3 fort.

Auf das theoretisch und am ersten Blick abstrus wirkende Faktum, dass eine einelementige Liste oder Riege notwendigerweise auch sortiert ist, sollte hier unbedingt hingewiesen werden, damit die Jugendlichen erkennen, dass es stets eines gewissen Nukleus bedarf, damit eine größere, einem bestimmten Kriterium entsprechende Struktur entstehen kann. Wenn man am Ende des Verfahrens prüft, ob und warum wirklich sortiert wurde, wird auf diesen ersten Schritt, der an dieser Stelle zu behandeln ist zurückgegriffen.

Sonst:

2b. Die Mitte der Riege wird festgestellt und die Riege teilt sich in zwei gleich große Teil-Riegen. Beide Riegen treten nach Teilung einen Schritt zurück (linke nach links hinten, rechte nach rechts hinten, sodass in der Mitte anfangs viel, im Laufe des fortgesetzten Verfahrens immer weniger, aber doch etwas Platz bleibt).

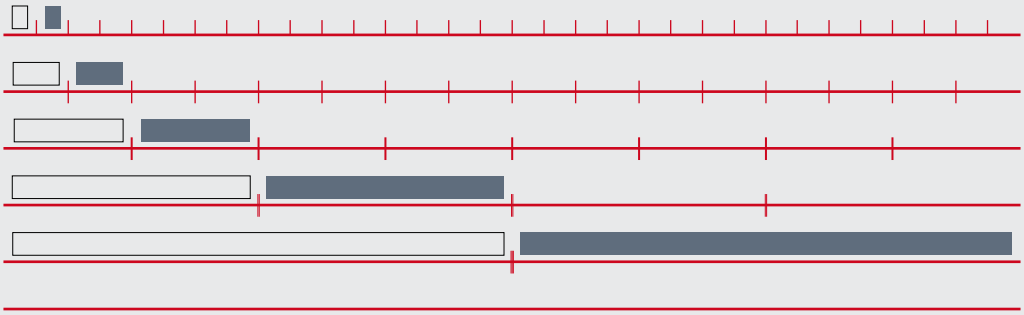
2c. ***Mergesort*** (linke Teilriege)

2d. ***Mergesort*** (rechte Teilriege)

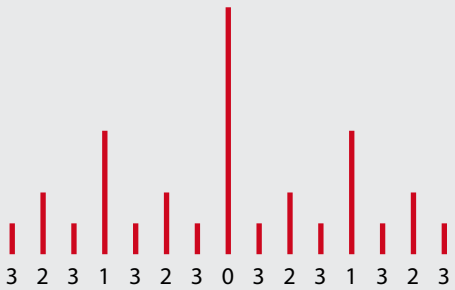
3. ***Mische*** (linke Teilriege, rechte Teilriege) so dass jeweils die Person vom Kopf der jeweiligen Teilriege mit dem „jüngeren" Geburtstag sich an das Ende der vor den beiden Teilriegen stehenden neuen „Gesamt(teil)riege" stellt. Der Kopf der „Gesamt(teil)riege" steht vor der Position auf der der Kopf der linken Teilriege stand oder noch steht.

Um den Aufteilungs- und Mischvorgang gut zu beobachten, wird es sinnvoll sein, am Boden Kreidestriche für jene Positionen/Linien zu machen, auf welche die sich teilenden Gruppen zurück treten sollen. Damit hat man auch jene Linie, auf welche die sich mischenden Gruppen vorkommen sollen.

Bodenskizze (Rote Linien mit Andeutung der sich rekursiv teilenden Riege, bis nur mehr Einzelpersonen übrig bleiben.)



In den meisten Klassen wird sich diese zu obiger Beschreibung passende Anordnung empfehlen. Wenn viel Platz vorhanden ist, kann man bei jedem Teilungsschritt die beiden Subriegen auch jeweils rechts und links der zu teilenden (Sub-)riege aufstellen lassen.



Die alternative Anordnung veranschaulicht das Teilungsverfahren etwas besser, insinuiert allerdings, dass zusätzlicher Speicherplatz benötigt wird. Dies ist allerdings nicht der Fall. Beim Verfahren mit Zurücktreten ist es einfacher, nachher feststellen zu lassen, wie viel zusätzlicher Speicherplatz für „Buchhaltungsoperationen" zur Feststellung der Grenzen der eben geteilten Subriegen nötig ist, wenn man nicht zurück tritt (zur Seite tritt) sondern nur die Grenzen der jeweiligen Bereiche markiert werden.

Details zu Mische(linkeTR, rechteT)

A. Die beiden aus dem Aufteilungsteil von Mergesort entstandenen Teilriegen stehen hinter der Linie, auf die sie durch Trennen und Zurücktreten gekommen sind. Die Linie auf der sie vorher standen, ist daher nun frei.

B. Der „Vergleicher" erhebt die Geburtsdaten der beiden an den Kopfpositionen der hinteren Linie stehenden Subgruppen und ersucht jenen mit dem „jüngeren" Geburtsdatum an den Kopf

der vorderen Linie zu treten.

C. Nun erhebt der „Vergleicher" weiter die Geburtsdaten der nunmehr an den (reduzierten) Kopfpositionen der hinteren Linie stehenden Personen und ersucht die „jüngere" sich an das Ende der sich eine Linie weiter vorn bildenden Gruppe anzuschließen.

D. Sobald eine der beiden hinteren Gruppen erschöpft ist, kann die verbliebene Gruppe wie sie ist an das Ende der vorderen Linie angeschlossen werden.

Analyse des Algorithmus

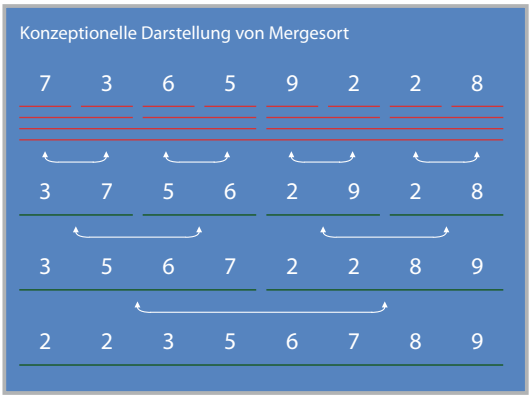
- Welche Beobachtungen wurden von der Beobachtungsgruppe gemacht:

  - o Welche Gesetzmäßigkeit gilt zwischen linker und rechter Teilriege?
    - A. nach dem Aufspalten?
    - B. nach dem Mischen?
  - o Welche Gesetzmäßigkeit gilt innerhalb linker und rechter Teilriege?
    - A. nach dem Aufspalten?
    - B. nach dem Mischen?
  - o Endet das Aufspalten in jedem Fall?
  - o Endet das Mischen in jedem Fall?
  - o Wie viel Linien mussten gebildet werden?
  - o Wie viel Fragen mussten beantwortet werden?
    - A. pro Gruppe?
    - B. über alle Gruppen hinweg in einer Linie?
    - C. über alle Linien hinweg?
  - o Wie viel Bewegungen mussten durchgeführt werden?
  - o Wie lassen sich diese Beobachtungen auf die Zahl der Personen in der Experimentalgruppe umlegen? Wie würden die Zahlen aussehen, wenn in der Experimentalgruppe 10, 100, 1000 Personen gestanden wären?

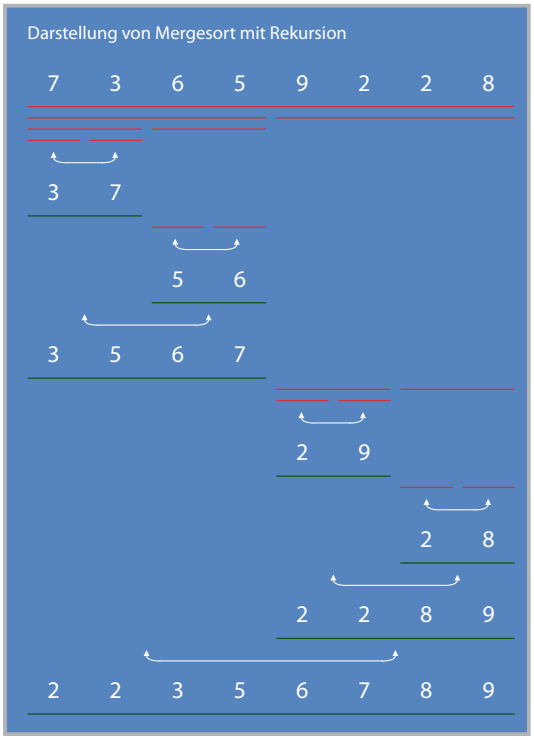
  - o Tafelskizze der entstandenen Teilsequenzen

Für die Tafelskizze zu Mergesort bieten sich zwei Varianten an:

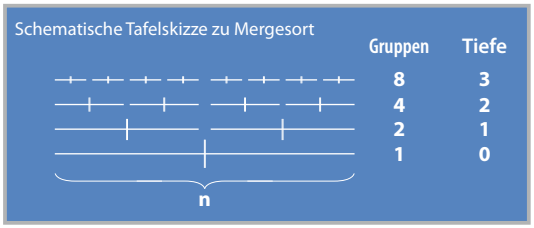
- *Für Klassen unterer Schulstufen, in denen noch nicht programmiert wurde bzw. für Kinder, denen man Rekursion im Detail noch nicht zumuten möchte, bietet sich nebenstehende konzeptionelle Darstellung an.*



- *Für Oberstufen-Klassen wird es demgegenüber naheliegend sein, das Verfahren (auch) in seiner Verschränkung zwischen teilen und mischen zu zeigen. Diese zweite Form entspricht der angegebenen Lösung in Pseudocode (S. 30). Wichtig ist freilich, dass diese rekursive Lösung schrittweise (hier zeilenweise) als Tafelbild entsteht. Aus der Dynamik der Präsentation können die Jugendlichen die Dynamik des Verfahrens erkennen und auch sehen, dass die reale Lösung letztlich der konzeptuellen Lösung entspricht. Die rekursive Lösung statisch zu projizieren (wie es im Druck leider der Fall ist) verwirrt demgegenüber wohl mehr als es hilft.*



- Anhand einer schematischen Tafelskizze feststellen, dass O(n · ld(n)) Schritte benötigt werden, da die Strukturen wenn man sie gesamtheitlich betrachtet, zwar immer n Elemente haben, aber in die Tiefe gestaffelt, nur ld(n) Ebenen entstehen.



- o Gilt O(n · ld(n)) immer?
- o Ist der Aufwand davon abhängig, ob die Riege sortiert war oder ob die Elemente zufällig angeordnet waren?
- Wodurch entsteht dies?

→ „Halbierungen" der Riege, hier immer! (Bei Quicksort nur im günstigsten Fall und auch im Schnitt)

$$\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \dots \frac{1}{2^n}, \dots$$

- Skizzierung des Algorithmus in Pseudocode mit Rekursion und, wenn Zeit ist, auch in Pseudocode mit expliziter Schleifenstruktur. Erkennen der Wiederholungsgesetzmäßigkeiten aus der Rekursion. Erkennen der Endbedingung und der O(ld(n)) Schritte in die Tiefe aus der Rekursion und O(n) Schritte aus der Kombination der Schleifenstruktur.

Pseudocode für Mergesort

Die Idee ist hier, dass ÜL (oder TN) quasi zur Protokollierung des durchgespielten Algorithmus sich den Pseudocode von der Klasse diktieren lässt. Dazu muss man nicht im Vorfeld bereits Programmierung in irgend einer Programmiersprache oder Pseudocode kennen gelernt haben. Die TN fassen dies als eine Art formalisiertes Stichwortprotokoll auf.
Der Algorithmus für Mergesort sollte jedenfalls skizziert werden, jener für Mische ist etwas komplizierter, obzwar das Konzept des systematischen Mischens unmittelbar klar ist. Daher ist er nur dann anzugeben, wenn die Zeit dafür reicht.

```
Mergesort (Feld, links, rechts) {
        WENN links < rechts DANN {
                bestimme mitte;
                Mergesort (Feld, links, mitte-1);
                Mergesort (Feld, mitte, rechts);
        } SONST {} %nur mehr 1 oder kein Element, daher sortiert
        Mische (Feld, links, mitte, rechts, Hilfsfeld);
        überschreibe Feld mit Hilfsfeld
}.
```

```
Mische (Feld, links, mitte, rechts; Hilfsfeld) {
        i := links;
        kl := links; km := mitte;
        SOLANGE kl < mitte UND km ≤ rechts {
                WENN Feld[kl] ≤ Feld[km] DANN {
                        Hilfsfeld[i] := Feld[kl];
                        kl := kl+1
                } SONST {
                        Hilfsfeld[i] := Feld[km];
                        km := km+1
                }
                i := i+1;
        }
        SOLANGE kl < mitte {
                Hilfsfeld[i] := Feld[kl]; kl := kl+1; i := i+1
        }
        SOLANGE km < rechts {
                Hilfsfeld[i] := Feld[km]; km := km+1; i := i+1
        }
}.
```

Vorsicht: Bei rekursiven Algorithmen grundsätzlich gleich Platz lassen für den bzw. die nichtrekursiven Trivialteile. Diese sodann mit Farbkreide einfügen. Man sollte sie auch dann einfügen, wenn sie, so wie oben, leer sind, damit die Jugendlichen diesen Teil nicht vergessen.

- Hinweis auf Rekursivität: Maler, der ein **Selbstportrait** vor seiner Staffelei malt (kann, Moderator einbeziehend, in bis zu 3 Rekursionsschritten auf der Tafel gezeichnet werden). *Eine ausführlichere Behandlung von Rekursion finden Sie in Modul A-Rekursion.*
- Diskussion von Rekursion in Mathematik: f(n) = n!

$$f(n) = \begin{cases} 1 & \text{wenn } n = 0 \\ n \cdot f(n-1) & \text{sonst} \end{cases}$$

Weiterführende Literatur

Ottmann Th., Widmayer P.: *Algorithmen und Datenstrukturen*, BI Wissenschaftsverlag, Mannheim, 1993.

Cormen Th. H., Leiserson Ch. E., Rivest R.L.: *Introduction to Algorithms*, MIT Press / McGraw-Hill Book Company, 1989.

Ein Blick auf die informatische *erLebens*-Landschaft

Tabelle 1 zeigt die für *Informatik erLeben* konzipierten Einheiten und Materialien, die Sie unter http://informatik-erleben.uni-klu.ac.at abrufen können.

Die Spalten von Tabelle 1 beschreiben Themengruppen, die der Beirat des Projekts als besonders attraktiv für Kinder in jenem Alter empfand, in dem die Neugierde für technische Zusammenhänge noch besteht. Der Komplexitätsgrad der Themen steigt dabei so an, dass, wenn man die entsprechende Personengruppe in mehreren Jahrgängen unterrichtet, in späteren Jahrgängen auf bereits früher durchgespielte Einheiten zurückgegriffen werden kann. Daher können die Spalten auch als fachliche Stränge, die sich über unterschiedliche Reifestufen bzw. Schulstufen[5] der Kinder und Jugendlichen erstrecken, angesehen werden. Aus der Komplexität der Thematik ergibt sich allerdings, dass nicht alle Spalten bis zum Oberstufen-Niveau gefüllt sind und auch nicht alle Spalten am Volksschul-Niveau beginnen.

Die Alters- bzw. die daraus ableitbare Schulstufen-Gliederung (Zeilen der Matrix) dienen lediglich der groben Orientierung. Bei Einheiten für das Volksschulalter war es uns wichtig, dass die Einheit gegebenenfalls auch als Spiel aufgefasst werden kann. Von Maturanten und Fast-Maturanten wird man nicht erwarten, dass sie einen Ant-Colony-Suchalgorithmus auf einer privaten Feier zur Belustigung aller durchspielen.

Freilich konnten wir bei der Konzeption der Einheiten grundsätzlich nicht davon ausgehen, dass *erLeben*-Einheiten in unmittelbarer Folge aufeinander aufbauen. Die Einheiten sind daher so konzipiert, dass man grundsätzlich in der jeweiligen Schulstufe bzw. Alterskategorie mit einer fachlich zum jeweiligen Unterricht passenden Einheit beginnen kann und dann in weiterer Folge entweder ein anderes Thema der Informatik ebenfalls mit einer *erLeben*-Einheit untermauert oder eben im Umfeld der erstgewählten Einheit weitere dazu passende Einheiten mit der Klasse erarbeitet. Das Aufbrechen der Einheiten in Module soll es Ihnen ermöglichen, individuelle Kombinationen aufzubauen und so ihre eigene Einheit zu entwickeln, die dem Wissensstand Ihrer Klasse und Ihren Lehrzielen am besten entspricht.

Die fachlichen Stränge sind allerdings nicht auf Spalten beschränkt. So kann etwa die Einheit, die mit dem Morsespiel beginnt, einerseits als Ausgangspunkt für Codierung und andererseits auch als Ausgangpunkt für Kommunikation und Protokolle verwendet werden. Ebenso ist sie in weiterer Folge in Bezug auf spezifische Codes und von dort wieder in Querverbindung zu CPU und zum Addierwerk (Halbaddierer, Volladdierer) weiter ausbaubar. Da hier nur ein Ausschnitt der insgesamt ausgearbeiteten bzw. noch in Ausarbeitung befindlichen Einheiten vorgestellt wird, zeigt die Einheit E-So1, Schuften oder Denken mit Mergesort, wie etwa auf Ebene einer Einheit Elemente aus A-Algorithmen (A3 Rekursion) in die Diskussion unterschiedlich leistungsfähiger Sortieralgorithmen eingewoben werden.

5 Auch wenn schulorganisatorisch die naheliegendste Gliederung jene in Volksschule, Sekundarstufe I, Sekundarstufe II wäre und wir ursprünglich auch von dieser Gliederung ausgegangen sind, zeigte sich im Verlauf der Arbeit, dass eine reine Altersangabe mit relativ großer Bandbreite zielführender ist. Dies liegt einerseits daran, dass selbst in Klassen gleichen Jahrgangs teils sehr unterschiedliches Leistungspotential vorliegt. Weiters ist durch den Unterricht im 5. Jahrgang AHS (Pflichtunterricht in Informatik) eine Zäsur gesetzt, die eigentlich eine sehr unterschiedliche Behandlung des Stoffs vor oder nach diesem Pflichtunterricht nahe legen würde. Doch die Ausprägungen des Unterrichts in diesem Jahrgang sind so divers, dass man auch diesen nicht als verlässliche Trennlinie auffassen darf.

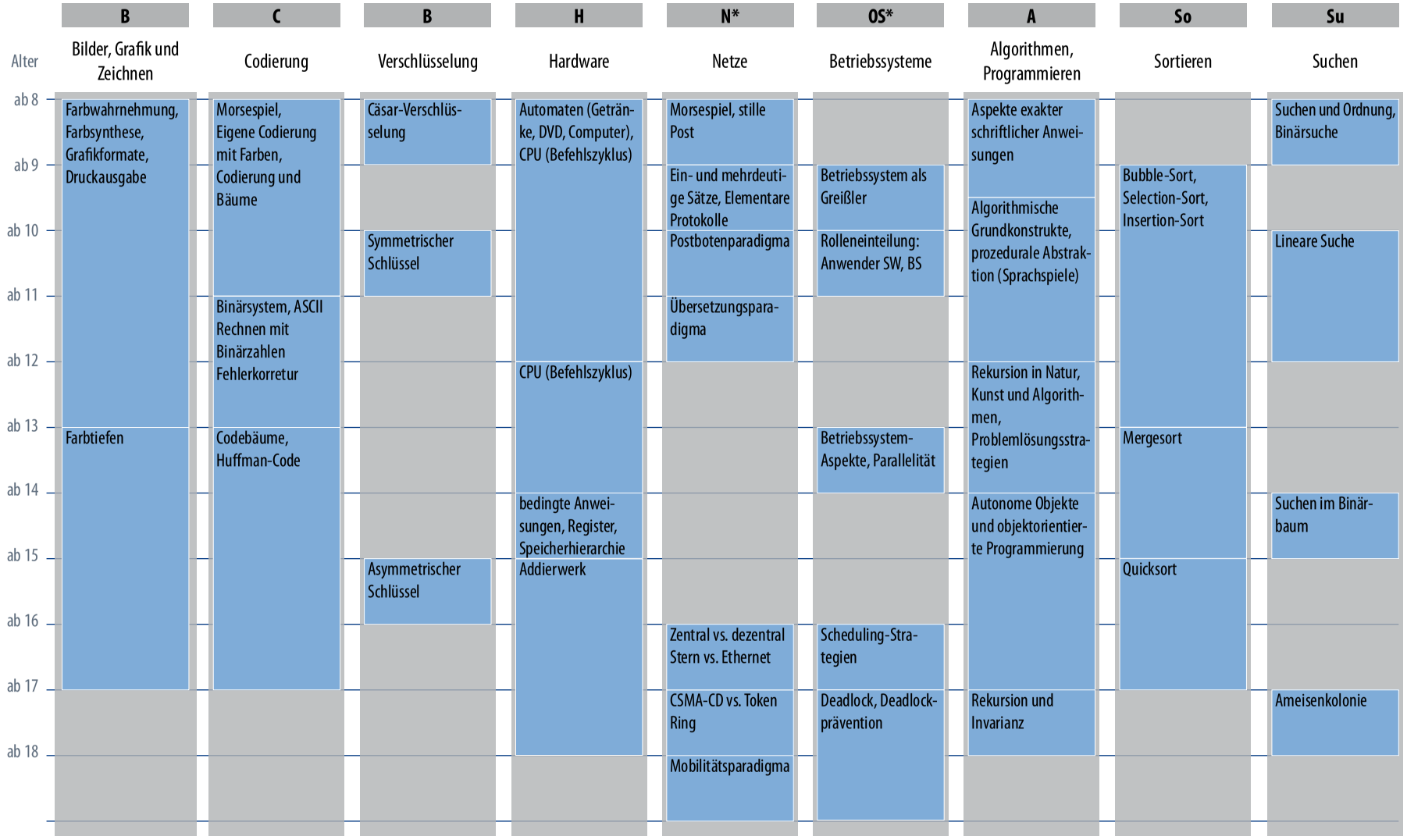


Tabelle 1: Zuordnung von Modulen und Einheiten zu Altersgruppen
*Mit *) versehene Stränge sind derzeit noch in Ausarbeitung*

Am Beispiel *Sortieren und Komplexität* lässt sich auch zeigen, dass durch die Modularisierung Raum für Spontanität bleiben kann. Ziel der Einheit ist, Schülerinnen und Schülern zu zeigen, dass es sich lohnt, einige Gedanken in die Qualität eines Algorithmus (hier in seine algorithmische Komplexität) zu investieren, bevor man sich an die Umsetzung macht. Dazu sollte zuerst ein elementares Sortierverfahren mit einer Komplexität von $O(n^2)$ verwendet werden und anschließend ein Verfahren mit algorithmischer Komplexität $O(n \cdot ld(n))$ vorgestellt werden. Anhand realistischer Berechnungszeiten für Einzelschritte, allerdings bei großem *n*, wird gezeigt, dass bei großem Datenumfang in einem Fall Stunden oder Tage vergehen, bis man das Ergebnis hat, während im anderen Fall Sekunden oder Minuten ausreichen.

Dabei ist es gleichgültig, ob man als naive Strategie Insertion-Sort, Selection-Sort oder Bubble-Sort verwendet und ebenso ist es gleichgültig, ob man als verbesserte Strategie Mergesort oder Quicksort verwendet. In einer Evaluierungseinheit begannen wir nach kurzer Einleitung mit der Frage „Wie sortiert man 1000 Karteikarten?". Als Antwort erhielten wir, dass man „die Nachbar-Karten solange tauscht, bis alle in der richtigen Reihenfolge sind". Dies ist zwar noch kein vollständiger Algorithmus, aber die Antwort zeigt, dass dieser Schüler in Richtung Bubble-Sort dachte. Wir haben daher spontan nicht den vorbereiteten Selection-Sort durchspielen lassen, sondern ersuchten, die Antwort dieses Schülers noch so weit zu präzisieren, dass sich daraus Anweisungen ableiten ließen und spielten sodann nach diesen Anweisungen Bubble-Sort als Beispiel für einen elementaren $O(n^2)$-Sortieralgorithmus. Auf Ebene der Lehreinheit blieben wir trotz dieses spontanen Wechsels eines Moduls bei unserem Konzept und führten im zweiten Teil wie vorbereitet Quicksort und die entsprechenden Analysen aus.

Wir hoffen, Sie durch obige Ausführungen neugierig gemacht zu haben, die vorgestellten *erLebens*-Einheiten mit Ihrer Klasse zu erproben und über die Web-Page weitere Einheiten zu finden.

Einladung

Werte Kolleginnen und Kollegen,
Sie haben im vorliegenden Heftchen nun einige Module kennen gelernt, die das Wesen der Informatik als technisches Fach vorstellen und die versuchen, bei den Schülerinnen und Schülern Interesse und Verständnis für diese Disziplin zu wecken. Weitere Vorschläge dieser Art finden Sie unter

http://informatik-erleben.uni-klu.ac.at

Wenn Ihnen dieses Konzept gefällt, haben Sie nun zwei Möglichkeiten, Einheiten dieser Art in Ihren Unterricht einzubauen:

- Sie entscheiden sich, eine (oder mehrere) dieser Einheiten selbst mit Ihrer Klasse durchzuführen oder

- Sie kontaktieren uns mit der Bitte, eine (vielleicht gelegentlich auch einige) dieser Einheiten mit Ihrer Klasse/Ihren Klassen durchzuführen.

Sollte die Hürde zu groß sein, oder möchten Sie das Konzept einfach zuerst einmal in einer Beobachterrolle kennen lernen, so können Sie gerne mit uns Kontakt aufnehmen. Wir halten in Ihrer Klasse gerne eine *Informatik erLeben* Einheit.

Bitte kontaktieren Sie uns unter:

O.Univ.-Prof. Dr. Roland Mittermeir roland@isys.uni-klu.ac.at
++43 (0) 463 2700 3513

Mag. Ernestine Bischof: ernestine@isys.uni-klu.ac.at
++43 (0) 463 2700 3517

Glossar

Algorithmus
Eine endliche, exakt beschriebene Handlungsvorschrift (Standardbeispiel: Kochrezept, Euklid'scher Algorithmus)

Algorithmische Komplexität
Gibt an, wie viel elementare Schritte ein Algorithmus benötigt, um eine vorgegebene Aufgabe zu bewältigen.

Baum (im informatischen Sprachgebrauch)
Eine azyklische zusammenhängende graphische Struktur, die von einem eindeutig bestimmten Element (Wurzel) ausgeht. Aus den Knoten (also insbesondere auch aus der Wurzel) gehen Kanten hervor, an denen weitere Knoten angeordnet sind. Gehen aus diesen Knoten keine weiteren Kanten hervor, spricht man von Endknoten oder Blättern. Hängt an solch einem inneren Knoten noch eine weitere Struktur (hat er also Nachfolger-Knoten), bildet dieser innere Knoten gemeinsam mit den ihm zugeordneten Substrukturen einen Teilbaum.
Bäume als Graphen werden in der Regel von der Wurzel ausgehend nach unten gezeichnet.

binärer Baum
Spezialfall eines Baumes, bei dem jeder Knoten maximal zwei Nachfolger hat. Sie spielen in der Informatik als Such- und Entscheidungsbäume eine besondere Rolle.

Bubble-Sort
Sortierverfahren, bei dem durch wiederholten Vergleich und wenn nötig durch Positionstausch der Nachbarelemente sortiert wird.

Graph
Mathematische Struktur, die aus Knoten (Elementen) und Kanten (Verbindungen) besteht. Je nach Aufbau der Verbindungselemente können Graphen zyklisch (Netze) oder azyklisch (Bäume, Sequenzen) sein.

Insertion-Sort
Sortierverfahren, bei dem das jeweils erste Element des noch unsortierten Teilbereichs an die aktuell richtige Position im bereits sortierten Teilbereich gebracht wird.

Mergesort
Effizientes Sortierverfahren, das durch Halbierung und anschließendes vergleichendes Mischen von Teilfolgen Sortiertheit erzielt. Eignet sich grundsätzlich auch für externes Sortieren.

$O(n)$, $O(n \cdot ld(n))$, $O(n^2)$
Komplexitätsangabe von Algorithmen in Abhängigkeit vom Umfang n der zu bewältigenden Aufgabe. Ein Algorithmus hat lineare (oder O(n)) Komplexität, wenn die Bearbeitungsdauer im Durchschnitt proportional zur Anzahl der zu bearbeitenden Elemente ist. Er hat quadratische ($O(n^2)$) Komplexität, wenn die Bearbeitungsdauer mit dem Quadrat der Anzahl der zu bearbeitenden Elemente steigt. Bei logarithmischer Komplexität steigt sie nur mit dem Logarithmus der Zahl der Elemente.
Konstante Glieder oder Faktoren spielen dabei eine vernachlässigbare Rolle. Wenn eine Analyse etwa zeigt, dass $3\,n^2 + 270\,n + 1000$ Schritte nötig sind, spricht man dennoch von einem $O(n^2)$, also quadratischen Verfahren, weil bei hinreichend großem n das Faktum, dass dieses quadratisch in die Formel eingeht, dominiert.
Für Sortierverfahren sind $O(n \cdot ld(n))$ Verfahren von besonderer Bedeutung, weil man zeigen kann, dass der allgemeine Fall von in-situ Sortierung ohne Parallelität im Schnitt mindestens n x Logarithmus-Dualis von n Schritte benötigt.

Quicksort
Effizientes Sortierverfahren, das eine Folge anhand eines Vergleichselements in eine Teilfolge mit Elementen, die kleiner(gleich) dem Vergleichselement und eine Teilfolge mit Elementen, die größer(gleich) dem Vergleichselement sind, aufspaltet. Durch rekursive Wiederholung entsteht Sortiertheit.

Selection-Sort
Sortierverfahren, bei dem das jeweils minimale Element des noch unsortierten Teilbereichs an das Ende des bereits sortierten Teilbereichs gebracht wird.