Kapitel 0

Organisatorisches

> There is no such thing as a good or bad organizational structure; there are only appropriate or inappropriate ones.
>
> *(Harold Kerzner)*

0.1 Lehrveranstaltung

- Lehrveranstaltung **Algorithmen und Programmierung**
- „Algorithmen und Programmierung“ ist Teil des Moduls „Algorithmen und Datenstrukturen“
- Lehrveranstaltung besteht aus:
  - **Vorlesung**
  - **Übungen**
  - **Tutorium** (Zusatzangebot)
- URL: `http://osg.informatik.tu-chemnitz.de/lehre/aup`

Vorlesung

- **Dozent:** Prof. Matthias Werner
  - `matthias.werner@informatik.tu-chemnitz.de`
  - Professur „Betriebssysteme“
  - Homepage: `osg.informatik.tu-chemnitz.de`
- **Orte & Zeiten:**
  1. Montag, 11.30 Uhr, Raum 1/201
  2. Freitag, 11.30 Uhr, Raum 1/201

Übungen

- Fünf Übungsgruppen
- **Orte & Zeiten:**
  1. Montag, 09:15 - 10:45 Uhr, Raum 1/375, Javar Akhundov
  2. Dienstag, 11:30 - 13:00 Uhr, Raum 1/309, Martin Richter
  3. Donnerstag, 15:30 - 17:00 Uhr, Raum 1/368A, Michael Reißner
  4. Freitag, 09:15 - 10:45 Uhr, Raum 1/368, Jens Pönisch ➡ Dieser Termin ist vorrangig für Bachelor „Wirtschaftsinformatik“
  5. Freitag, 09:15 - 10:45 Uhr, Raum 1/309, Javar Akhundov
- Bitte über ➡ **OPAL** (`mytuc.org/pvmp`) für Gruppe anmelden

Hinweis für künftige Wirtschaftsinformatiker

Teilnahme am **Programmierpraktikum** (aus Modul BM-WIINF) parallel zur Vorlesung wird empfohlen, auch wenn dort andere Sprache genutzt wird.

- **Inhalt:**
  - Klärung von Fragen zur Vorlesung
  - Bearbeiten von Beispielaufgaben
  - Überprüfen von Lösungswegen
- Übungen sind (wie die Vorlesung) ein **Angebot**.
  - **Sie setzen Vorbereitung und Mitarbeit der Teilnehmer voraus.**
  - Bei Mangel an Interesse/Vorbereitung spielt der Übungsleiter **nicht** Alleinunterhalter, sondern beendet die Übung.
- Die Übungen beginnen in KW 42, d.h. nächsten Montag

Tutorium

- Aufgrund guter Erfahrungen bieten wir dieses Semester wieder zusätzlich eine **Frage-Antwort-Stunde** an
  - Im Vorlesungsverzeichnis und auf der Webseite etwas inkorrekt als „Tutorium“ bezeichnet 😊
- **Leiter:** Dr. Peter Tröger
- **Ort & Zeit:** Mittwoch, 15:30 - 17:00 Uhr, 1/201

Zur Beachtung

- In allen Lehrveranstaltungen ist das Handy lautlos zu stellen oder abzuschalten.

Selbststudium

- Neben den Präsenzveranstaltungen sollten Sie selbständig arbeiten
  - Die EU-Richtlinien gehen davon aus, dass für jede Präsenz-Stunde in einer Lehrveranstaltung etwa **zwei** Stunden im Selbststudium verbracht werden

Programmieren kann man letztendlich nur durch Programmieren lernen – Probieren Sie Ihre Erkenntnisse so oft wie möglich **direkt am Computer** aus!

- Sie können dafür die Computer-Pools der Fakultät (FRIZ) und des Universitäts-Rechenzentrums nutzen.

- Für die Nutzung der Pools und viele andere Dienste brauchen Sie einen **Uni-Account**
  - Falls Sie diesen noch nicht haben, sollten Sie ihn schnellstmöglichst beantragen ➡ `https://www.tu-chemnitz.de/urz/nutzerkonto.html`

0.2 Formalia

Voraussetzungen für Teilnahme

- Keinerlei formale Voraussetzungen (außer Studienzulassung 🙂)
- Individuelle Voraussetzungen:
  - Abstraktionsvermögen
  - Abiturstoff, insbesondere anwendungsbereite Mathematik
  - Logisches Denken
  - Bereitschaft, am Rechner „herumzuspielen“
- **Keine** Voraussetzung (schadet aber auch nicht): Programmiererfahrungen / Kenntnis einer Programmiersprache

Anerkennung

- **Anerkennung** in folgenden Studiengängen:
  - Bachelor „Informatik“
  - Bachelor „Angewandte Informatik“
  - Bachelor „MINT“
  - Bachelor „Wirtschaftsinformatik“
  - Bachelor „Wirtschaftsmathematik“
  - Bachelor „Mathematik“ mit Nebenfach „Informatik“ oder „Mathematik“
  - Bachelor „Informatik und Kommunikationswissenschaften“
  - Master „Informatik für Geistes- und Sozialwissenschaftler“
- Die **Art** der Anerkennung ist in den verschiedenen Studiengängen sehr unterschiedlich geregelt:
  - Für die Studierenden der **Wirtschaftsinformatik** ist es eine **Prüfung**, für alle anderen eine **Prüfungsvorleistung**
  - *Merke*: Prüfungen können bei Nichtbestehen maximal zweimal wiederholt werden, Prüfungsvorleistungen / Studienleistungen beliebig oft

Leitungsnachweis und Prüfungsleistungen

- Es gibt zwei verschiedene Arten des Leistungsnachweises:
  - **Bewertete Aufgaben** (i.d.R. Programmierung)
  - **Klausur**
- **Bewertete Aufgaben**
  - Ca. 6-8 Aufgaben über das Semester verteilt
- **Klausur**
  - Dauer: 120 Minuten
  - Anmeldung über das Zentrale Prüfungsamt
- Bachelor Informatik und Bachelor Angewandte Informatik (neue SO) müssen **eine der beiden** Leistungen erbringen
- Bachelor Wirtschaftsinformatik, Bachelor Mathematik und Master Informatik für Geistes- und Sozialwissenschaftler müssen **nur die Klausur** bestehen
- Alle anderen müssen **beide** Leistungen erbringen

Studiengang	Prüfung?	Vorleistung?	Klausur?	Aufgaben?
B. Informatik	✗	✔	✔/✗	✔/✗
B. Angewandte Informatik	✗	✔	✔/✗	✔/✗
B. MINT	✗	✔	✔	✔
B. Wirtschaftsinformatik	✔	✗	✔	✗
B. Wirtschaftsmathematik	✗	✔	✔	✔
B. Mathematik	✗	✔	✔	✗
B. Informatik und Kommunikationswiss.	✗	✔	✔	✔
B. Inf. für Geistes- und Sozialwiss.	✗	✔	✔	✗

Bewertete Aufgaben

- Die Leistung „Bewertete Aufgaben“ wird als erfolgreich gewertet, wenn mindestens 50% der möglichen Punkte erzielt werden
- Außerdem können bis zu 10% Zusatzpunkte für die Abschlussklausur erlangt werden ➡ Das Lösen der Aufgaben lohnt sich also auch für diejenigen, für die dies keine Pflicht ist
- Abgabe über Open Submit (siehe Webseite / Tutorium):

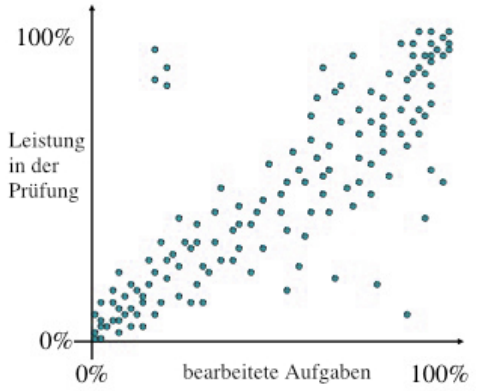
Achtung!

- Alle eingereichten Aufgabenlösungen werden einem **Plagiatscheck** unterworfen.
- Festgestellte **Plagiate** bei bewerteten Aufgaben werden als ***Betrugsversuch*** gewertet!


Bildquellen: Harald Dettenborn / Ralf Roletschek, CC 3.0 Lizenz

Trainingsaufgaben

- Zu Ihrer Selbstkontrolle stellen wir Ihnen Trainingsaufgaben zur Verfügung
- Bearbeitung fakultativ
- Besprechung bei Bedarf in der Übung
- Auch wenn das Lösen dieser Aufgaben nicht erzwungen ist, wird jede(r/m) das **selbstständige** Lösen zu besseren Rezeption des Stoffes **dringend** empfohlen

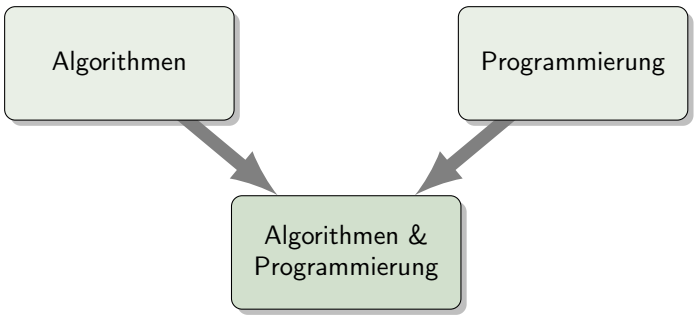


Typische Korrelation zwischen Bearbeitung der Übungsaufgaben und Prüfungsleistung

0.3 Inhalt und Literaturempfehlungen

Themen der Vorlesung

- Diese Lehrveranstaltung berührt zwei große Gebiete[1]



Algorithmen und Programmierung

- Der Entwurf von **Algorithmen** ist vollständig unabhängig von Computern und verlangt das Verstehen von Problemen, Abstraktionsvermögen, Mathematikkenntnisse sowie Kreativität

[1] Wie überraschend! 😉

- **Programmierung** ist (in der praktischen Umsetzung) an Computer gebunden und verlangt (abstraktes) Verständnis der Vorgänge im zu programmierenden Computer, Kenntnisse über Syntax und Semantik[2] von Programmiersprachen sowie über den Umgang mit Entwicklungstools.

Ziele

- Nach erfolgreichen Absolvieren dieser Lehrveranstaltung sollten Sie...
  - ... in Abstraktionen und Algorithmen denken können;
  - ... einige grundlegende Algorithmen kennen und verstehen;
  - ... grundsätzliche Ansätze für algorithmische Lösungen kennen und anwenden können;
  - ... die imperative Programmiersprache C in den Grundzügen[3] beherrschen;
  - ... Algorithmen entwickeln und programmieren können;
  - ... mit den wichtigsten Entwicklungswerkzeuge (Editor, Compiler, Linker, Debugger) sicher umgehen können;
  - ... kleinere Probleme mit Hilfe von C lösen können;
- Zusätzlich sollten Sie natürlich jede Menge mehr oder weniger nützliches Wissen angesammelt und anwendungsbereit parat haben.

Hinweis

Nach allgemeiner Erfahrung sind die Konzepte, die in diesem Kurs die größten Schwierigkeiten bereiten

- **Rekursion** im Algorithmenentwurf
- **Zeiger** (Pointer) beim Programmieren.

Bitte versuchen Sie, diese Konzepte so früh wie möglich zu verstehen, und lernen Sie, sicher damit umzugehen.

- Um Sie darin zu unterstützen, werden diese Konzepte recht früh im Kurs eingeführt und dann wiederholt angewendet.

[2] Diese Begriffe werden wir später noch genauer untersuchen.
[3] ...und auch ein bißchen Python...

Literatur

- Diese Veranstaltung folgt keinem einzelnen Lehrbuch
- Folgende Liste gibt Literatur an, die nützlich sein kann:

[AHU83] Alfred V. Aho, John E. Hopcroft und Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983

[AHU75] Alfred V. Aho, John E. Hopcroft und Jeffrey D. Ullman. *Design and Analysis of Computer Algorithms*. Addison-Wesley, 1975

Anm: *Sind (oder waren zumindest jahrelang) an vielen US-Universitäten die Standardlehrbücher für Studenten im Grund- [AHU83] bzw. Hauptstudium [AHU75].*

Funktion/Bewertung von Algorithmen

[Cor+01] Thomas H. Cormen u. a. *Introduction to Algorithms*. 2. Aufl. MIT Press, 2001

Anm: *Die wahrscheinlich umfangreichste Algorithmensammlung in einem einzelnen Buch; mathematisch fundiert; gut als Nachschlagewerk*

[Sed90] Robert Sedgewick. *Algorithms in C*. Addison-Wesley, 1990

Anm: *Das Buch diskutiert ein breites Spektrum von Algorithmen (wenn auch nicht ganz der Umfang von [Cor+01] erreicht wird). Verzichtet wird auf (zuviel) Mathematik, aber auch auf Aspekte des Algorithmenentwurfs (siehe nächster Abschnitt).*

Neben [Sed90] gibt es vom gleichen Autor noch die Bücher [Sed83] und [Sed03] Alle drei Bücher sind ähnlich und unterscheiden sich im Wesentlichen nur in der Programmiersprache. [Sed83] benutzt Pascal/Pseudocode, [Sed03] Java.

Achtung: Viele Fehler (zumindest in älteren Ausgaben), Errata beachten!

[Knu69] Donald E. Knuth. *Fundamental Algorithms*. Bd. 1. The Art of Computer Programming. Addison-Wesley, 1969

[Knu81] Donald E. Knuth. *Seminumerical Algorithms*. Bd. 2. The Art of Computer Programming. Addison-Wesley, 1981

[Knu98] Donald E. Knuth. *Sorting and Searching*. Bd. 3. The Art of Computer Programming. Addison-Wesley, 1998

Anm: *Donald Knuth schreibt seit über 40 Jahren an einem Standardwerk über Informatik, das auf 5 – 7 Bände angelegt ist und dessen erste drei Bände (plus einiger „Vorschauhefte“ auf Band 4 und Korrekturhefte zu den Bänden 1 – 3) bereits erschienen sind.*

Knuth ist für seine Präzision bekannt. Mathematische Betrachtungen können ggf. übersprungen werden. Benutzte Programmiersprache ist ein künstlicher Assembler.

Entwurf von Algorithmen

*Die folgenden Bücher legen besonderen Wert auf den **Entwurf** von Algorithmen – ein Thema, welches in vielen Algorithmik-Büchern zu kurz kommt.*

[Ski08] Steven S. Skiena. *The Algorithm Design Manual*. 2. Aufl. Springer, 2008

Anm: *Unterhaltsam durch häufigen Bezug auf interessante Probleme aus dem wahren Leben™ („war stories“).*

[Man89] Udi Manber. *Introduction to Algorithms. A Creative Approach*. Addison-Wesley, 1989

Anm: *Autor nutzt ein induktives Vorgehen beim Algorithmenentwurf. Er stellt auch falsche Lösungen vor, um sukzessive zum korrekten Entwurf zu kommen.*

[Edm08] Jeff Edmonds. *How to Think About Algorithms*. Cambridge University Press, 2008

Anm: *Besonderer Schwerpunkt auf Schleifeninvarianten und Rekursion*

Programmiersprache C

Bücher über C gibt es in Hülle und Fülle. Deshalb gibt es hier nur eine sehr kleine Auswahl.

Nutzen Sie ruhig dasjenige Buch, das Ihnen am besten liegt. Achten Sie aber darauf, dass es mindestens die ANSI-Variante von C enthält.

[KR88] Brian W. Kernighan und Dennis M. Ritchie. *The C Programming Language*. 2. Aufl. Prentice Hall, 1988

Anm: *Das Standardwerk über C, von den Erfindern der Sprache.*

[Oua97] Steve Oualline. *Practical C*. 3. Aufl. O'Reilly, 1997

Anm: *Typisches O'Reilly-Buch: Pragmatisch, knapp im Stil, ausführlich im Umfang.*

[Wol16] Jürgen Wolf. *Grundkurs C: C-Programmierung verständlich erklärt*. 2. Aufl. Rheinwerk Computing, 2016

Anm: *Das aktuellest der hier aufgelisteten Bücher, enthält auch C11*

[Reg03] Regionales Rechenzentrum Niedersachsen, Hrsg. *Die Programmiersprache C. Ein Nachschlagewerk*. 18. Aufl. Universität Hannover. 2003

Anm: *Billig (3,50 €); erhältlich über das Rechenzentrum, siehe* `https://www.tu-chemnitz.de/urz/handbuecher_intern.html#bestellformular`

[Sch14] Helmut O.B. Schellong. *Moderne C-Programmierung*. 3. Aufl. Springer, 2014

Anm: *Als E-Book über* `http://dx.doi.org/10.1007/978-3-642-54437-8` *für TU-Studenten kostenlos*

[Goo04] Dan Gookin. *C For Dummies*. 2. Aufl. Wiley Publishing, 2004

Anm: *Falls sonst nichts mehr hilft.*

Programmiersprache Python

Auch über Python gibt es zahlreiche Bücher – auch hier deshalb nur eine kurze Auswahl.

[Zel03] John M. Zelle. *Python Programming: An Introduction to Computer Science*. Franklin Beedle & Associates, 2003

[Chu06] Wesley J. Chun. *Core Python Programming*. 2. Aufl. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006

Anm: *Die Empfehlung von Jens Pönisch* 🙂

[Lut13] Mark Lutz. *Learning Python*. 5th. O'Reilly, 2013

Anm: *Für TUC-Angehörige kostenlos als E-Book über die Bibliothek verfügbar:* `https://katalog.bibliothek.tu-chemnitz.de/Record/0008951133`.

Andere Programmiersprachen

Hier sind nur Bücher gelistet, die keine reine Programmiersprachlehrbücher sind, sondern (auch) Hintergrundwissen über Algorithmenentwurf etc. bieten.

Jedoch sind sie so stark sprachgebunden, dass sie nicht in den allgemeinen Algorithmen-Teil gehören.

[HW94] Brian Harvey und Matthew Wright. *Simply Scheme: Introducing Computer Science*. MIT Press, 1994

Anm: *Didaktisch sehr gut, z.T. unterhaltsam. Sprache: Scheme*

[Wir86] Niklaus Wirth. *Algorithms & Data Structures*. Prentice-Hall, 1986

Anm: *Niklaus Wirth ist der Guru der Strukturierten Programmierung. Sprache: Modula-2*

[Har97c] Brian Harvey. *Symbolic Computing*. 2. Aufl. Bd. 1. Computer Science Logo Style. MIT Press, 1997

[Har97a] Brian Harvey. *Advanced Techniques*. 2. Aufl. Bd. 2. Computer Science Logo Style. MIT Press, 1997

[Har97b] Brian Harvey. *Beyond Programming*. 2. Aufl. Bd. 3. Computer Science Logo Style. MIT Press, 1997

Anm: *Diese drei Bände präsentieren (nahezu) die komplette Informatik mit Hilfe der Programmiersprache Logo*

Vielleicht auch interessant

[HLW06] Helmut Herold, Bruno Lurz und Jürgen Wohlrab. *Grundlagen der Informatik.* Pearson Studium, 2006

Anm: *Auf über 700 Seiten werden theoretische, technische und praktische Grundlagen der Informatik präsentiert; gut als Nachschlagewerk.*

[Are+15] Tilo Arens u. a. *Mathematik.* 3. Aufl. Spektrum Akademischer Verlag, 2015

Anm: *Auf über 1600(!) Seiten werden gut verständlich die Grundlagen der Mathematik dargestellt; allein im ca. ersten Viertel im Wesentlichen der Abiturstoff (der in dieser Vorlesung vorausgesetzt wird).*

[Ben06] Mordechai Ben-Ari. *Understanding Programming Languages.* For personal use free available: `http://www.weizmann.ac.il/sci-tea/benari/books/upl.zip`. John Wiley & Sons, 2006

[Mit02] John C. Mitchell. *Concepts in Programming Languages.* Cambridge University Press, 2002

Anm: *Beide Bücher betrachten Programmiersprachen im Vergleich.*

[Har92] David Harel. *Algorithmics. The Spirit of Computing.* Addison-Wesley, 1992

Anm: *Feuilletonartige Betrachtung algorithmischer Probleme (was dem Einsteiger/Fachfremden das Lesen erleichtert, manche aber stört). Enthält ein Kapitel über Programmiersprachen und viele Bibelzitate.*

[Bia09] Federico Biancuzzi. *Masterminds of Programming. Conversations with the Creators of Major Programming Languages.* O'Reilly, 2009

Anm: *Interviews mit den Schöpfern berühmter Programmiersprachen. Gibt Einblicke in die Motivation und Ziele beim Entwurf von Programmiersprachen.*

[Ben00] Jon Bentley. *Programming Pearls.* 2. Aufl. Addison-Wesley, 2000

[Ben88] Jon Bentley. *More Programming Pearls. Confessions of a Coder.* 1. Aufl. Addison-Wesley, 1988

Anm: *Keine Lehrbücher oder Monographien, sondern ausgewählte Beiträge aus der Zeitschrift „Communications of the ACM" über Probleme beim Entwurf von Software.*

Sehr unterhaltsam, trotzdem lehrreich.

Zur Beachtung

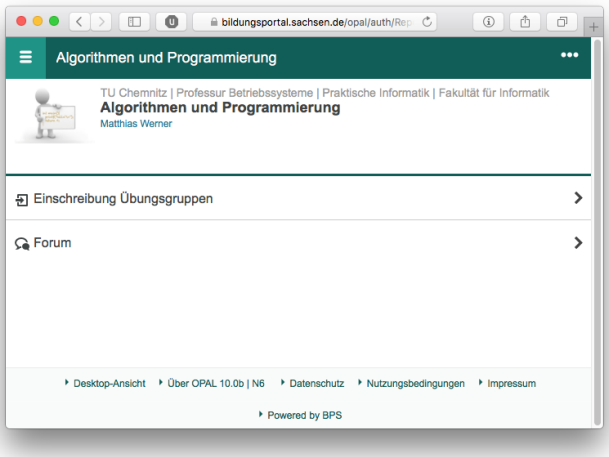
Außer [Sch14] und [HLW06] ist alle hier gelistete Literatur in englischer Sprache.

Von vielen Büchern existiert eine Übersetzung ins Deutsche. Seien Sie aber vorsichtig: Häufig enthalten die Übersetzungen inhaltliche Fehler!

Da Sie für Ihr Studium sowieso nicht um englische Literatur herumkommen, fangen Sie am besten gleich mit Originalliteratur an.

E-Learning

- Neben der Web-Seite an der Professur gibt für den Kurs E-Learning-Tools über OPAL (Bildungsportal Sachsen)
- URL: `https://bildungsportal.sachsen.de/opal`
  - Login mit Uni-Accountname
- Bisher:
  - Einschreibung für Übungsgruppen
  - Forum zur Diskussionen die mit der Lehrveranstaltung in Beziehung stehen
- Weiteres kann während des Semesters hinzukommen



Knobelaufgaben

Am Ende jedes Kapitels werden ein paar Aufgaben stehen. Diese stehen nicht immer im Zusammenhang mit dem unmittelbaren Stoff – dafür gibt es die Trainingsaufgaben auf den Webseiten der Veranstaltung – sondern sollen ergänzende Denkanstöße geben. Teilweise gehen sie über den Stoff hinaus, greifen ihm vor, und/oder verlangen „unkonventionelles“ Denken.

Niemand ist verpflichtet, diese Aufgaben zu lösen (oder sie sich auch nur anzusehen). Wer es aber tut, übt durch die Beschäftigung mit diesen Aufgaben Fähigkeiten, die jede(r) Programmierer(in) nach meiner Meinung nach braucht, wenn sie oder er mehr als ein „Code-Monkey“[4] sein will.

Aufgabe 0.1

Betrachten Sie die Reihe folgender Zahlen:

9 44 32 12 7 42 34 92 35 37 41 8 20 27 83 64 61 28 39 93 29 17 13 14 55 21 66 72 23 73 99 1 2 88 77 3 65 83 84 62 5 11 74 68 76 78 67 75 69 70 22 71 24 25 26

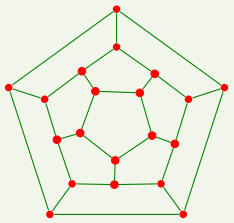
(a) Streichen Sie sowenig Zahlen wie möglich, so dass die verbleibenden Zahlen eine aufsteigende Reihe bilden!

(b) Lösen Sie die Aufgabe erneut, nur soll die Ergebnisreihe diesmal absteigend sein!

(c) Beschreiben Sie wie man vorgehen muss, um für eine beliebige Zahlenreihe zum optimalen Ergebnis bei den beiden vorstehenden Aufgaben zu kommen!

Aufgabe 0.2

Betrachten Sie die nebenstehende Figur.
Finden Sie einen geschlossenen Weg (d.h. Start = Ende) entlang der grünen Linien, so dass jeder rote Punkt genau einmal besucht wird!
Amerkung: *Es muss **nicht** jede grüne Linie genutzt werden.*



[4] https://www.youtube.com/watch?v=qYodWEKCuGg