4 - Algorithmen & Objektivität

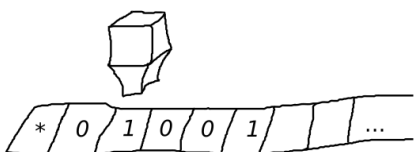
Marianne Maertens

Technische Universität Berlin

Wintersemester 2025/2026

Themen

1. Wiederholung
2. Was ist ein Algorithmus
3. Repräsentation von Algorithmen
4. Turing-Maschinen
5. Algorithmen und Objektivität



Geschichte der Informatik

- enge Verbindung von:

Automatik

automatos (altgriech.) - selbst (autos) + denken (men)

Rechengeräte - praktische Fragen, Arbeitserleichterung

Ingenieurdisziplin - Rechnen mechanisieren

→ Entwicklung des Computers (Hardware)

Information

informare (lat.) - durch Unterweisung bilden

Erkenntnisse - theoretische, philosophische Fragen

Mathematik - Denken formalisieren

→ **Entwicklung von Algorithmen (Software)**

Lesen Sie den Text

Wie werden Algorithmen im Text definiert?

Welche Bedeutung wird ihnen zugeschrieben?

Lesen Sie den Text

Wie werden Algorithmen im Text definiert?

- klar definierte Folge von Schritten, die eindeutig beschreibt, wie alle Probleme eines bestimmten Typs gelöst werden können

Welche Bedeutung wird ihnen zugeschrieben?

Lesen Sie den Text

Wie werden Algorithmen im Text definiert?

- klar definierte Folge von Schritten, die eindeutig beschreibt, wie alle Probleme eines bestimmten Typs gelöst werden können

Welche Bedeutung wird ihnen zugeschrieben?

- die Intelligenz, die zur Lösung des Problems erforderlich ist, steckt im Algorithmus

Lesen Sie den Text

Wie werden Algorithmen im Text definiert?

- klar definierte Folge von Schritten, die eindeutig beschreibt, wie alle Probleme eines bestimmten Typs gelöst werden können

Welche Bedeutung wird ihnen zugeschrieben?

- die Intelligenz, die zur Lösung des Problems erforderlich ist, steckt im Algorithmus
- das Intelligenznivau von Maschinen hängt davon ab, wie viel Intelligenz durch Algorithmen abgebildet werden kann

Lesen Sie den Text

Wie werden Algorithmen im Text definiert?

- klar definierte Folge von Schritten, die eindeutig beschreibt, wie alle Probleme eines bestimmten Typs gelöst werden können

Welche Bedeutung wird ihnen zugeschrieben?

- die Intelligenz, die zur Lösung des Problems erforderlich ist, steckt im Algorithmus
- das Intelligenznivau von Maschinen hängt davon ab, wie viel Intelligenz durch Algorithmen abgebildet werden kann
- Maschinen können nur solche Aufgaben ausführen, für die es einen Algorithmus gibt. Wenn für eine Problemlösung kein Algorithmus existiert, liegt die Lösung des Problems außerhalb der Fähigkeit von Maschinen.

Euklidischer Algorithmus (300 v. Chr.)

- Algorithmus zum Finden des größten gemeinsamen Teilers zweier Zahlen
- ältester Algorithmus, der den Namen verdient
- wird noch heute benutzt, weil es keinen besseren gibt

Gegeben: 2 Zahlen

Gesucht: deren größter gemeinsamer Teiler (ggT)

Wie funktioniert der klassische Euklidische Algorithmus?

Der Euklidische Algorithmus - geometrisch

gegeben: 75 und 27

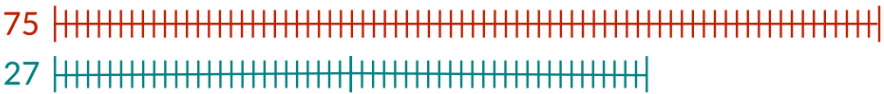
Der Euklidische Algorithmus - geometrisch

gegeben: 75 und 27

Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

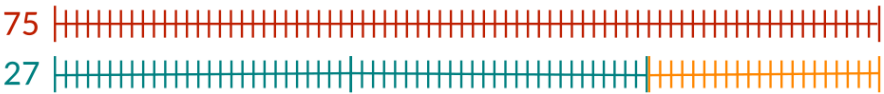
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

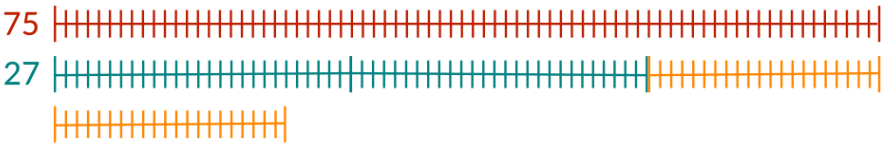
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

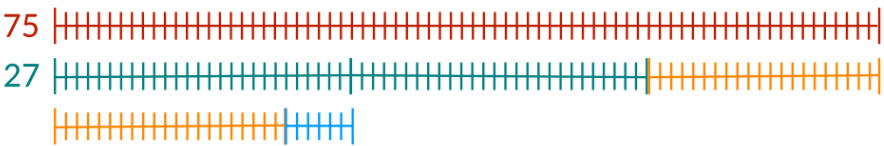
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

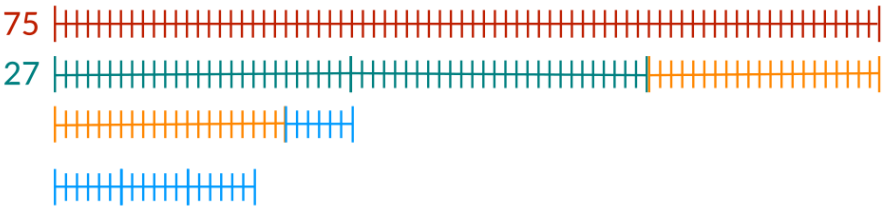
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

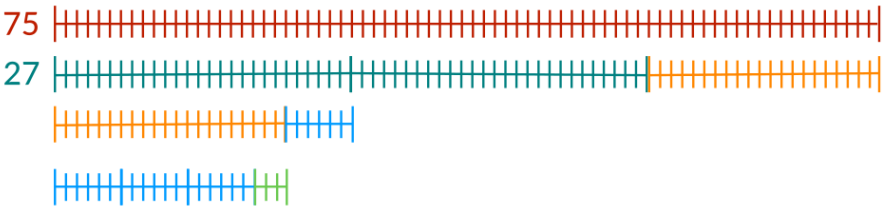
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

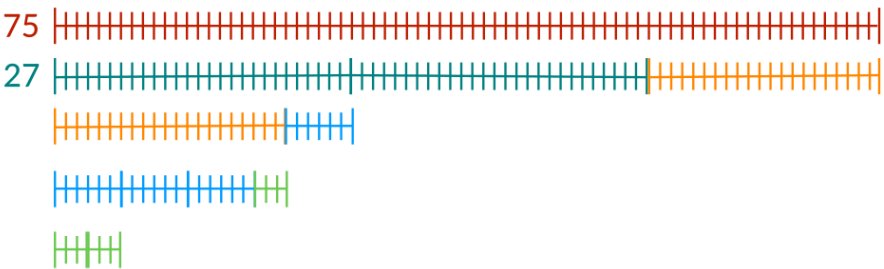
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

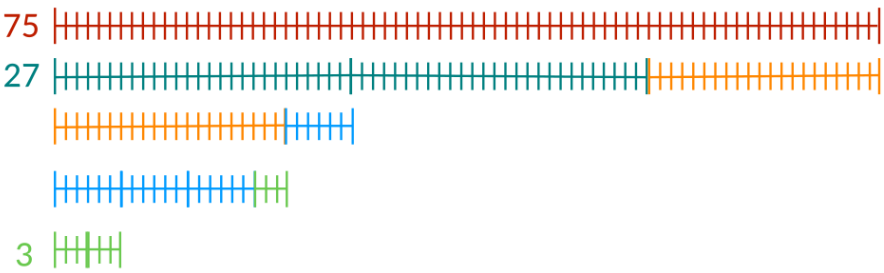
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

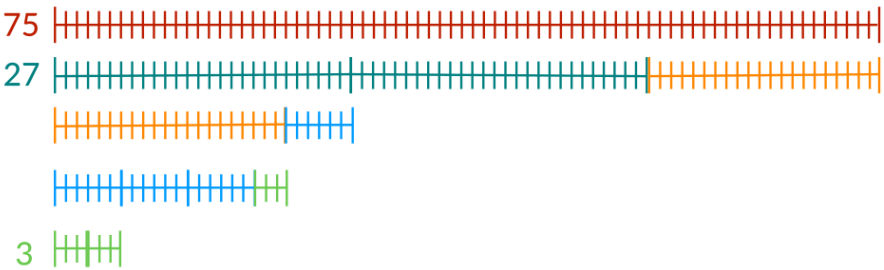
gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

Der Euklidische Algorithmus - geometrisch

gegeben: 75 und 27



Quelle https://www.youtube.com/watch?v=i_Zb43TEOAc

ggT(75, 27) = 3

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

1 2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

~~1~~ 2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

~~1~~ 2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~

9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15

~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~

23 ~~24~~ 25 ~~26~~

- streiche Vielfache der aktuellen Primzahl (“sieben”)

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~

~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~

~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~

23 ~~24~~ 25 ~~26~~

- streiche Vielfache der aktuellen Primzahl (“sieben”)

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~

~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~

~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~

23 ~~24~~ ~~25~~ ~~26~~

- streiche Vielfache der aktuellen Primzahl (“sieben”)

Das Sieb des Eratosthenes (276-194 v.Chr.)

- Algorithmus zum Finden aller Primzahlen in einer Liste von Zahlen der Länge n
- Primzahl ist eine ganze Zahl, $x \in \mathbb{N} > 1$, die ausschließlich durch sich selbst und durch 1 **teilbar** ist

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~

~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~

~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~

23 ~~24~~ ~~25~~ ~~26~~

- streiche Vielfache der aktuellen Primzahl (“sieben”)
- größte Siebzahl einer Liste mit n Zahlen ist $\sqrt{n}$
- übrige Zahlen sind Primzahlen

Herkunft des Begriffs Algorithmus

- $\sim$ 780 – 850 Abu Dscha'far Muhammad ibn Musa al-Chwarizmi
- "*Dixit Algorizmi*" = "also sprach al-Chwarizmi"
- als Gütezeichen einer Rechnung (ab ca. 1200)

Herkunft des Begriffs Algorithmus

- $\sim$ 780 – 850 Abu Dscha'far Muhammad ibn Musa al-Chwarizmi
- "*Dixit Algorizmi*" = "also sprach al-Chwarizmi"
- als Gütezeichen einer Rechnung (ab ca. 1200)

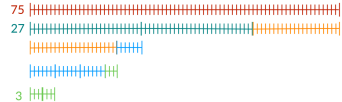
Definition

- Eine eindeutige Handlungsanweisung zur Lösung eines Problems. Dabei wird eine bestimmte Eingabe in eine bestimmte Ausgabe überführt.
- Intelligenz steckt in der Anweisung. Der Ausführende muss nicht verstehen, was er tut.

(Hisāb al-ğābr wa-'l-muqābala - Algebra)

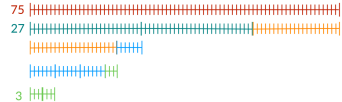
Übung

1. Bestimmen Sie ggT(8,5) mit Hilfe des klassischen Euklidischen Algorithmus. Schreiben Sie die Schritte in Form von Gleichungen auf.
2. Beschreiben Sie in Worten, was der Algorithmus macht.



Übung

1. Bestimmen Sie ggT(8,5) mit Hilfe des klassischen Euklidischen Algorithmus. Schreiben Sie die Schritte in Form von Gleichungen auf.
2. Beschreiben Sie in Worten, was der Algorithmus macht.

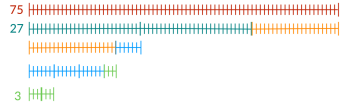


$$8 = 1 * 5 + 3$$
$$5 = 1 * 3 + 2$$
$$3 = 1 * 2 + 1$$
$$2 = 2 * 1 + 0$$

ggT(8,5) = 1

Übung

1. Bestimmen Sie ggT(8,5) mit Hilfe des klassischen Euklidischen Algorithmus. Schreiben Sie die Schritte in Form von Gleichungen auf.
2. Beschreiben Sie in Worten, was der Algorithmus macht.



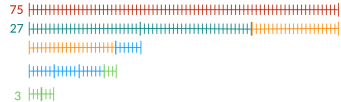
$8 = 1 * 5 + 3$
$5 = 1 * 3 + 2$
$3 = 1 * 2 + 1$
$2 = 2 * 1 + 0$

ggT(8,5) = 1

1. größere Zahl als Vielfaches der kleineren Zahl darstellen plus Rest

Übung

1. Bestimmen Sie ggT(8,5) mit Hilfe des klassischen Euklidischen Algorithmus. Schreiben Sie die Schritte in Form von Gleichungen auf.
2. Beschreiben Sie in Worten, was der Algorithmus macht.



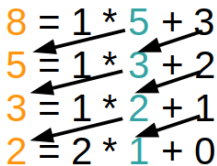
$8 = 1 * 5 + 3$
$5 = 1 * 3 + 2$
$3 = 1 * 2 + 1$
$2 = 2 * 1 + 0$

ggT(8,5) = 1

1. größere Zahl als Vielfaches der kleineren Zahl darstellen plus Rest
2. wenn der Rest null ist, dann ist die kleinere Zahl der ggT

Repräsentation von Algorithmen

mathematisch - arithmetisch



natürliche Sprache

1. größere Zahl als Vielfaches der kleineren Zahl plus Rest
2. wenn der Rest null ist $\rightarrow$ kleinere Zahl ist der ggT

Repräsentation von Algorithmen

mathematisch - arithmetisch

$$8 = 1 * 5 + 3$$
$$5 = 1 * 3 + 2$$
$$3 = 1 * 2 + 1$$
$$2 = 2 * 1 + 0$$

mathematisch - geometrisch

natürliche Sprache

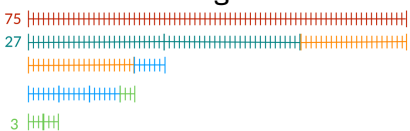
1. größere Zahl als Vielfaches der kleineren Zahl plus Rest
2. wenn der Rest null ist $\rightarrow$ kleinere Zahl ist der ggT

Repräsentation von Algorithmen

mathematisch - arithmetisch

$$8 = 1 * 5 + 3$$
$$5 = 1 * 3 + 2$$
$$3 = 1 * 2 + 1$$
$$2 = 2 * 1 + 0$$

mathematisch - geometrisch



natürliche Sprache

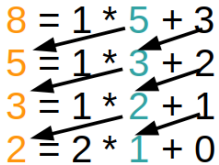
1. größere Zahl als Vielfaches der kleineren Zahl plus Rest
2. wenn der Rest null ist → kleinere Zahl ist der ggT

Pseudocode

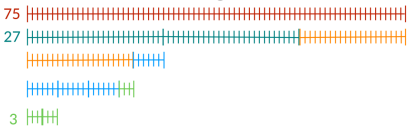
```
ggt_mod(a,b)
   while a % b != 0 # so lange der Rest nicht null ist
     if b > a:
       a, b = b, a
     a = a % b # größere Zahl als Vielfaches der Kleineren
   return b # kleinere Zahl ist ggT
```

Repräsentation von Algorithmen

mathematisch - arithmetisch



mathematisch - geometrisch



Ablaufdiagramme

(Flowcharts), Programmcode

natürliche Sprache

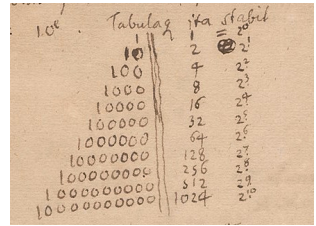
1. größere Zahl als Vielfaches der kleineren Zahl plus Rest
2. wenn der Rest null ist → kleinere Zahl ist der ggT

Pseudocode

```
ggt_mod(a,b)
   while a % b != 0 # so lange der Rest nicht null ist
     if b > a:
       a, b = b, a
     a = a % b # größere Zahl als Vielfaches der Kleineren
   return b # kleinere Zahl ist ggT
```

Formalisieren des Denkens: Zeichensysteme

- Leibniz (1646-1716) träumte von einer Maschine, die Symbole manipulieren kann, um den Wahrheitswert mathematischer Aussagen zu bestimmen
- Voraussetzung: klare, formale Sprache
- 3. Jh. v. Chr. indischer Mathematiker Pingala beschreibt Zahlensysten bestehend aus 2 Zeichen
- Ende 17Jh. entwirft Leibniz das binäre Zahlensystem
  - kritisch bei der Entwicklung elektronischer Rechenmaschinen
  - Zahlen werden durch elektrische Zustände dargestellt (Strom an/aus)

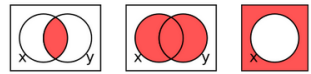


https://de.wikipedia.org/wiki/Dualsystem#Entwicklung_des_Dualsystems

Formalisieren des Denkens: Boolesche Logik

- 1847 entwickelt George Boole (1815-1864) das nach ihm benannte Logikkalkül
- es war die Grundlage für die Zwei-Elementige Boolesche Algebra
- verallgemeinert Eigenschaften der logischen Operatoren UND (Konjunktion), ODER (Disjunktion), NICHT (Negation) sowie der mengentheoretischen Verknüpfungen Durchschnitt, Vereinigung, Komplement

  → FORSA - Formale Sprachen und Automaten



x∧y x∨y ¬x

Venn-Diagramme für Konjunktion, Disjunktion und Negation

https://de.wikipedia.org/wiki/Boolesche_Algebra

Konjunktion

∧	**0**	**1**
0	0	0
1	0	1

Disjunktion

∨	**0**	**1**
0	0	1
1	1	1

Negation

	¬
0	1
1	0

Programmierbare Maschinen

- Ada Lovelace (1815-1852) schreibt 1842: *"Die Grenzen der Arithmetik wurden in dem Augenblick überschritten, in dem die Idee zur Verwendung der [Programmier]Karten entstand … die Analytical Engine hat keine Gemeinsamkeit mit schlichten Rechenmaschinen."*

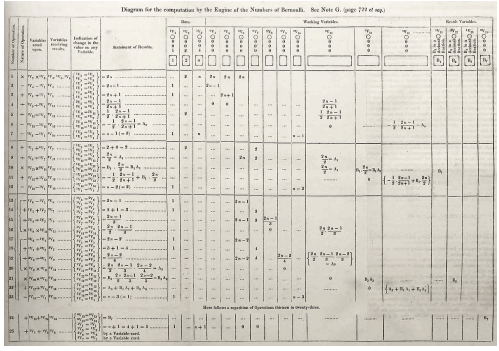
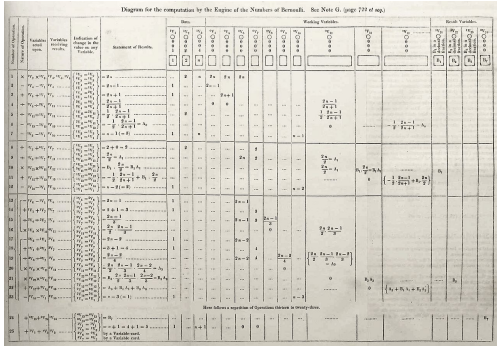


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 720 et seq.)

Programmierbare Maschinen

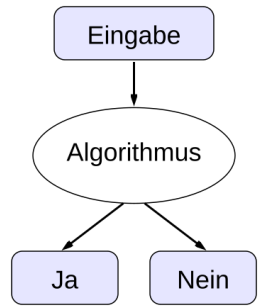
- Ada Lovelace (1815-1852) schreibt 1842: *“Die Grenzen der Arithmetik wurden in dem Augenblick überschritten, in dem die Idee zur Verwendung der [Programmier]Karten entstand … die Analytical Engine hat keine Gemeinsamkeit mit schlichten Rechenmaschinen.”*



“Die Analytical Engine hat keine Ambitionen etwas (Neues) hervorzubringen. Sie kann [nur] das tun, was wir ihr befehlen; sie hat jedoch keine Fähigkeit zur Erkenntnis analytischer Verhältnisse oder Wahrheiten.”
(Lady Lovelace’s objection, Turing, 1950)

Das “Entscheidungsproblem”

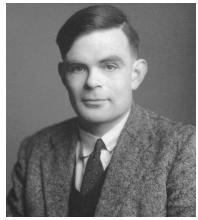
- Anfang 20. Jh.: Kann es einen Algorithmus geben, der für eine Aussage (Eingabe) entscheidet, ob sie universell wahr oder falsch ist (Hilbert & Ackermann, 1928)
- Church (1935) und Turing (1936) haben bewiesen, dass es so einen Agorithmus nicht geben kann



https://de.wikipedia.org/wiki/Entscheidbarkeit

Konzept der Berechenbarkeit

Alan Turing (1912-1954)



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

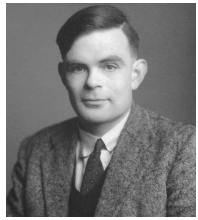
By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

- Wenn man schon nicht alles **beweisen** kann, was kann man dann **berechnen**? (Und was nicht?)
- Was heißt das überhaupt, Dinge zu berechnen?

Konzept der Berechenbarkeit

Alan Turing (1912-1954)



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

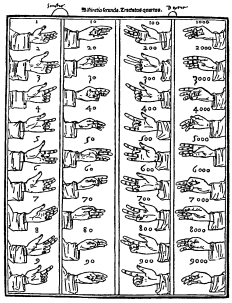
[Received 28 May, 1936.—Read 12 November, 1936.]

- Wenn man schon nicht alles **beweisen** kann, was kann man dann **berechnen**? (Und was nicht?)
- Was heißt das überhaupt, Dinge zu berechnen?

→ versucht zu **formalisieren** wie ein Mensch rechnet

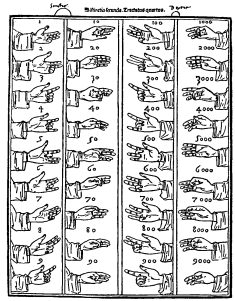
→ **Turingmaschine** - Idee war nicht, Modell für Computer aufzustellen

Was brauchen Menschen, um etwas zu berechnen?



Was brauchen Menschen, um etwas zu berechnen?

- Liste von Anweisungen (Algorithmus/Programm)
- Lesen, Schreiben (radieren/überschreiben)
- Schmierpapier für Zwischenergebnisse
- Gedächtnis
- “Eingabe” & “Ausgabe”
- Speichermedium



Turing's Regeln

1. Alphabet: endliche Menge von Zeichen

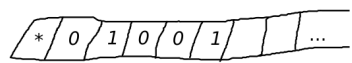
   $\Sigma = \{0, 1\}$

Turing's Regeln

1. Alphabet: endliche Menge von Zeichen
   $\Sigma = \{0, 1\}$
2. Speicherband, das in Felder unterteilt ist, 1 Zeichen pro Feld, beliebig lang - "Tickertape"

Turing's Regeln

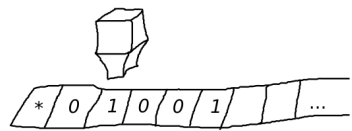
1. Alphabet: endliche Menge von Zeichen
   $\Sigma = \{0, 1\}$
2. Speicherband, das in Felder unterteilt ist, 1 Zeichen pro Feld, beliebig lang - "Tickertape"



3. Leerzeichen □ oder "B" (blank)
4. Sonderzeichen '*' für den Anfang des Tapes (* und $\square \notin \Sigma$)

Turing's Regeln

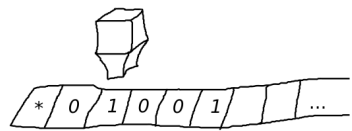
1. Alphabet: endliche Menge von Zeichen
   $\Sigma = \{0, 1\}$
2. Speicherband, das in Felder unterteilt ist, 1 Zeichen pro Feld, beliebig lang - "Tickertape"



3. Leerzeichen □ oder "B" (blank)
4. Sonderzeichen '*' für den Anfang des Tapes (* und $\square \notin \Sigma$)
5. Schreib-/Lesekopf, der jeweils ein Feld lesen oder überschreiben kann (darf * nicht überschreiben und nicht weiter nach links gehen)

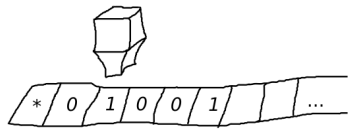
Turing's Regeln

1. Alphabet: endliche Menge von Zeichen
   $\Sigma = \{0, 1\}$
2. Speicherband, das in Felder unterteilt ist, 1 Zeichen pro Feld, beliebig lang - "Tickertape"



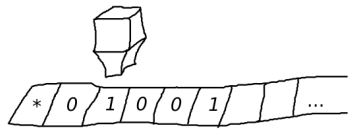
3. Leerzeichen □ oder "B" (blank)
4. Sonderzeichen '*' für den Anfang des Tapes (* und $\square \notin \Sigma$)
5. Schreib-/Lesekopf, der jeweils ein Feld lesen oder überschreiben kann (darf * nicht überschreiben und nicht weiter nach links gehen)
6. "Programm", das den Kopf steuert - Berechnungsvorschrift

Wie soll so ein “Programm” aussehen?



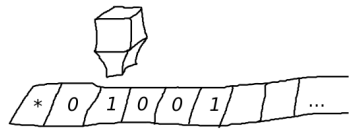
Wie soll so ein “Programm” aussehen?

read	write	move	go to
1	1	R	0



Wie soll so ein “Programm” aussehen?

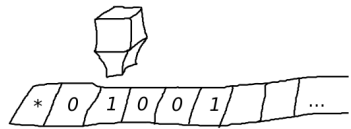
read	write	move	go to
1	1	R	0



1. `read:` Lies aktuelles Feld

Wie soll so ein “Programm” aussehen?

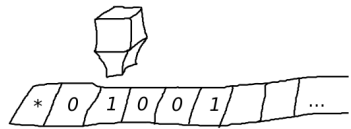
read	write	move	go to
1	1	R	0



1. `read:` Lies aktuelles Feld
2. `write:` Schreibe etwas auf das Feld

Wie soll so ein “Programm” aussehen?

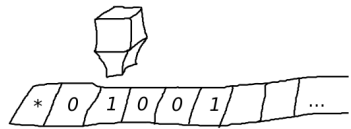
read	write	move	go to
1	1	R	0



1. `read:` Lies aktuelles Feld
2. `write:` Schreibe etwas auf das Feld
3. `move`: Gehe 1 Schritt nach links, rechts oder halte an

Wie soll so ein “Programm” aussehen?

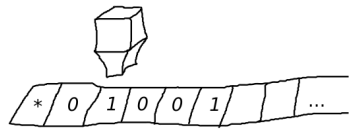
read	write	move	go to
1	1	R	0



1. `read:` Lies aktuelles Feld
2. `write:` Schreibe etwas auf das Feld
3. `move`: Gehe 1 Schritt nach links, rechts oder halte an
4. `go to:` wähle den nächsten Programmschritt aus

Wie soll so ein “Programm” aussehen?

read	write	move	go to
1	1	R	0

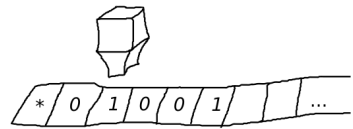


1. `read:` Lies aktuelles Feld
2. `write:` Schreibe etwas auf das Feld
3. `move`: Gehe 1 Schritt nach links, rechts oder halte an
4. `go to:` **wähle** den nächsten Programmschritt aus

(2, 3 und 4 hängen von 1 ab)

Wie soll so ein “Programm” aussehen?

read	write	move	go to
1	1	R	0
0	1	R	2
*	*	R	0
B	0	L	-1



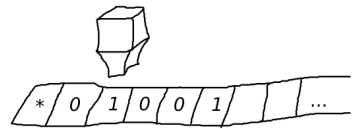
1. `read:` Lies aktuelles Feld
2. `write:` Schreibe etwas auf das Feld
3. `move`: Gehe 1 Schritt nach links, rechts oder halte an
4. `go to:` **wähle** den nächsten Programmschritt aus

(2, 3 und 4 hängen von 1 ab)

Wie soll so ein “Programm” aussehen?

read	write	move	go to
1	1	R	0
0	1	R	2
*	*	R	0
B	0	L	-1

= Zustand



1. `read:` Lies aktuelles Feld
2. `write:` Schreibe etwas auf das Feld
3. `move`: Gehe 1 Schritt nach links, rechts oder halte an
4. `go to:` **wähle** den nächsten Programmschritt aus

(2, 3 und 4 hängen von 1 ab)

Turing-Maschine: Subtraktion von 1 von Binärzahl

10-1=9

$$\begin{array}{rr} & 1010 \\ - & 1 \\ \hline & 1001 \end{array}$$

17-1=16

$$\begin{array}{rr} & 10001 \\ - & 1 \\ \hline & 10000 \end{array}$$

Turing-Maschine: Subtraktion von 1 von Binärzahl

- wenn Input 0, dann 0

10-1=9

$$\begin{array}{rr} & 1010 \\ - & 1 \\ \hline & 1001 \end{array}$$

17-1=16

$$\begin{array}{rr} & 10001 \\ - & 1 \\ \hline & 10000 \end{array}$$

Turing-Maschine: Addition von 1 zu einer Binärzahl

Überlegen Sie sich eine Turing-Maschine zur Addition von 1 zu einer Binärzahl für die nachfolgenden Beispielrechnungen!

4+1=5

$$\begin{array}{r} 100 \\ +\quad 1 \\ \hline 101 \end{array}$$

5+1=6

$$\begin{array}{r} 101 \\ +\quad 1 \\ \hline 110 \end{array}$$

Turing-Maschine: Addition von 1 zu einer Binärzahl

Überlegen Sie sich eine Turing-Maschine zur Addition von 1 zu einer Binärzahl für die nachfolgenden Beispielrechnungen!

4+1=5

$$\begin{array}{r} 100 \\ +\quad 1 \\ \hline 101 \end{array}$$

5+1=6

$$\begin{array}{r} 101 \\ +\quad 1 \\ \hline 110 \end{array}$$

Wenden Sie Ihre Turing-Maschine nun auf folgende Zahl an! Ändern Sie Ihre Turing-Maschine, so dass sie auch für diese Eingabe die korrekte Ausgabe liefert!

7+1=8

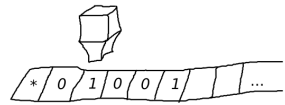
$$\begin{array}{r} 111 \\ +\quad 1 \\ \hline 1000 \end{array}$$

Turing-“Maschine”

read	write	move	go to
1	1	R	0
0	1	R	2
*	*	R	0
B	0	L	-1

= Zustand

- theoretisches Modell einer Rechenmaschine
- Modell der theoretischen Informatik, das den Begriff der Berechenbarkeit formalisiert
- Berechnung besteht aus schrittweisen Manipulationen von Zeichen/Symbolen, Zeichen können u.a. als Zahlen interpretiert werden

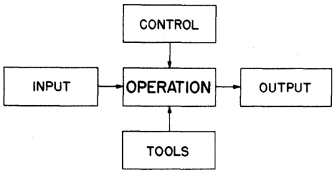


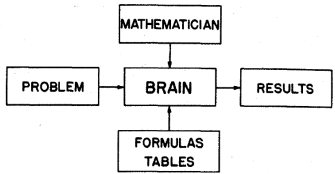
1940/50er Jahre

Grace Hopper (1906-1992)

- 1952 The education of a computer

While the materialization is new, the idea of mechanizing mathematical thinking is not new. Its lineage starts with the abacus and descends through Pascal,Leibnitz, and Babbage. More



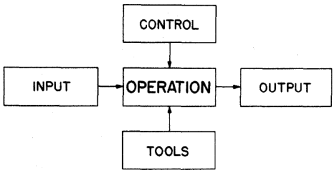


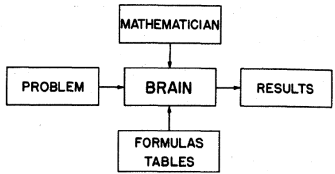
1940/50er Jahre

Grace Hopper (1906-1992)

- 1952 The education of a computer

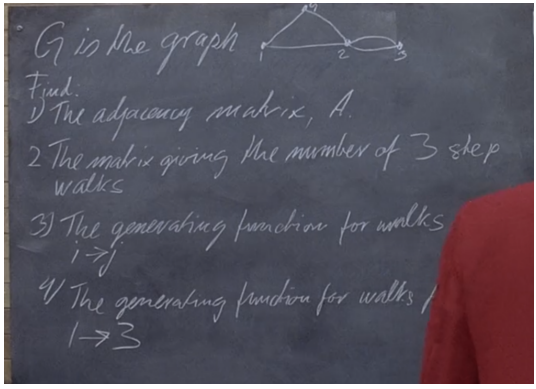
While the materialization is new, the idea of mechanizing mathematical thinking is not new. Its lineage starts with the abacus and descends through Pascal,Leibnitz, and Babbage. More





"I was lazy and hoped that the programmer may return to being a mathematician

Algorithmen & Objektivität



aus Good Will Hunting (Drama 1997), Quelle `https://bthmas.wordpress.com/2016/03/26/good-will-hunting-problem/`

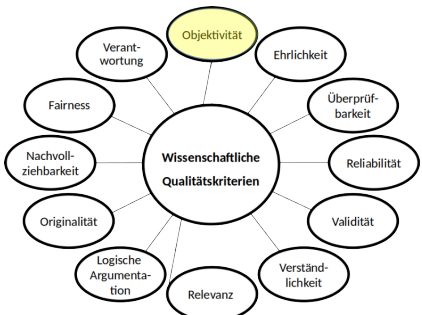
- so beschrieben, dass bei gleicher Eingabe (Frage) immer dasselbe Ergebnis herauskommt
- jede/r kann sie ausführen
- unabhängig von Nutzerin

→ Objektivität

Objektivität

1. Was bedeutet Objektivität?
2. Was steht Objektivität im Wege?
3. Welches Problem wird in dem Affenexperiment beschrieben?

Objektivität

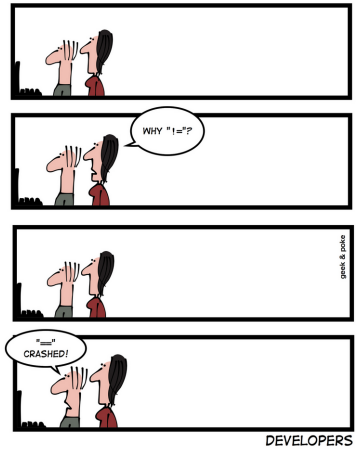


1. Ergebnisse sind unabhängig vom Beobachter, d.h. frei von persönlichen Urteilen, Einstellungen, nicht frei von fachlichem Vorwissen
2. menschliche Wahrnehmungs- und Urteilstendenzen z.B. Cognitive Biases, da sie die Unvoreingenommenheit beeinträchtigen können
3. Affenexperiment: Bestätigungsfehler (confirmation bias, selektive Wahrnehmung, Wunschdenken)

`https://www.visualcapitalist.com/50-cognitive-biases-in-the-modern-world/`

Intelligente Maschinen - Wo stehen wir?

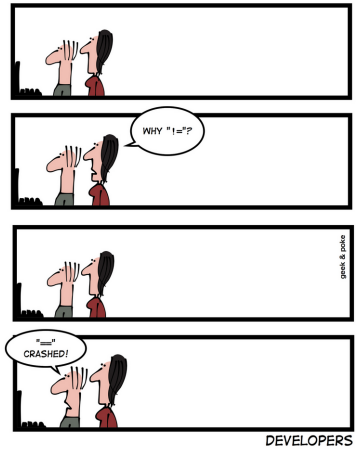
- Denken zu definieren ist subjektiv oder indirekt - von außen beobachtbar machen, messbar machen - wie gut ist die Messung?



(Quelle: `http://geek-and-poke.com/`)

Intelligente Maschinen - Wo stehen wir?

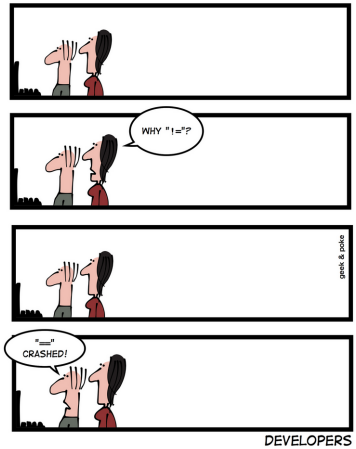
- Denken zu definieren ist subjektiv oder indirekt - von außen beobachtbar machen, messbar machen - wie gut ist die Messung?
- Intelligenz haben wir noch nicht definiert



(Quelle: `http://geek-and-poke.com/`)

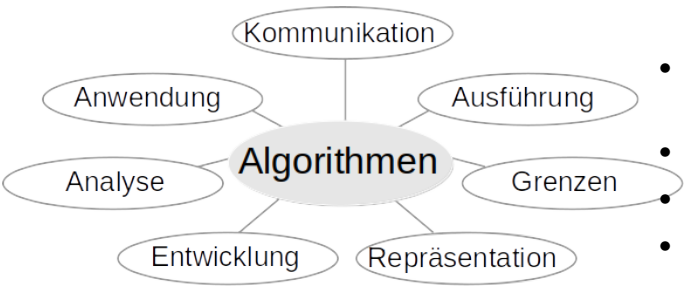
Intelligente Maschinen - Wo stehen wir?

- Denken zu definieren ist subjektiv oder indirekt - von außen beobachtbar machen, messbar machen - wie gut ist die Messung?
- Intelligenz haben wir noch nicht definiert
- klassische Maschinen und Algorithmen sind bereichsspezifisch



(Quelle: `http://geek-and-poke.com/`)

Algorithmen als Forschungsgegenstand



- Welche Probleme lassen sich algorithmisch lösen?
- Wie kann man Algorithmenentwicklung vereinfachen?
- Wie repräsentiert man Algorithmen?
- Wie kann man Algorithmen analysieren?
- Können Algorithmen intelligentes Verhalten erzeugen?
- Wie steigert man ihre Effizienz?