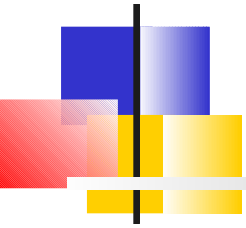
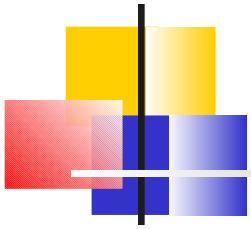


Algorithmische Komplexität

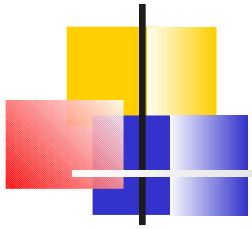
Lucas Champollion



Ein paar Folgen

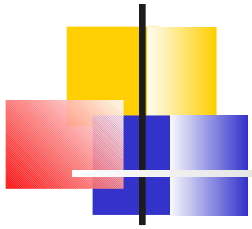
1. 11111111111111111111111111111111111111111111111111111111111111111...
2. 01101010000010011110011001100111111100111011100110010010000100000...
3. 11011111111111111010111111111110011111101011011011111111111011111111111...
4. _____

Wie läßt sich Komplexität formalisieren?



Kolmogorov-Komplexität

- Die *Kolmogorov-Komplexität* $K_U(x)$ des Strings x in Bezug auf einen universellen Computer U ist die Länge (in Bits) des kürzesten Programms, das x ausdruckt und dann anhält.
- Bei der *bedingten Komplexität* $K_U(x|I(x))$ ist die Länge der Ausgabe I(x) im Computer voreingestellt und braucht nicht als Teil des Programms übermittelt zu werden
- Im folgenden wählen wir einen Computer U, der eindeutige Kommandos auf Englisch versteht.



Beispiele

1. 11111111111111111111111111111111111111111111111111111111111111...

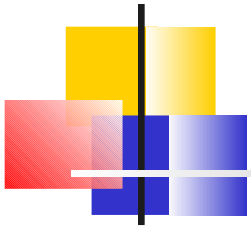
   Print “1” the specified number of times.

2. 01101010000010011110011001100111111001110111001100100100001000...

   Print the specified number of digits in binary format of sqrt(2)-1.

3. 11011111111111111010111111111100111111010111011011111111111011111111111...

   ???



Die dritte Folge

3.1101111111111111101011111111110011111101011
10110111111111110111111111...

- Diese Folge ist *"1-lastig":* insgesamt k Einsen bei Länge n

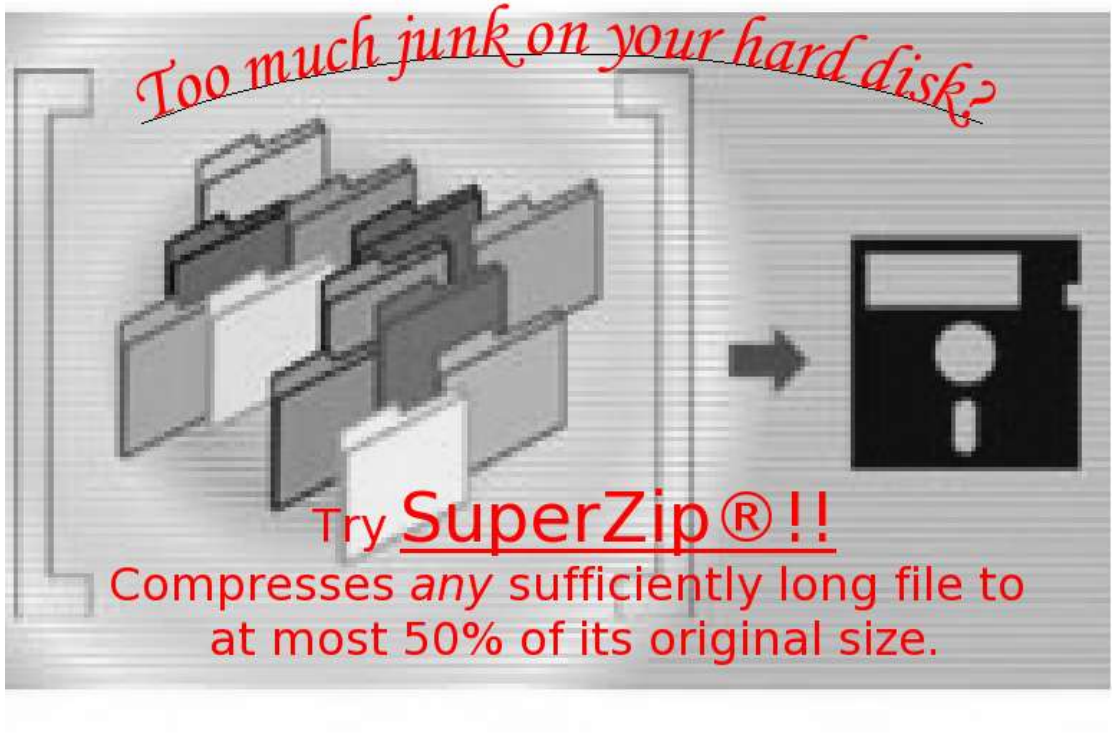
  `Print the i`th `sequence in a lexicographic ordering of all sequences with k 1's`

- Länge dieses Programms: etwa *log n* + *nH(k/n)* Bits (deutlich weniger als n)
  - Länge des Programms ist proportional zu n

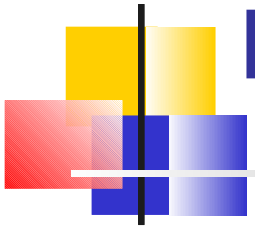
... und was ist mit “unserer” Folge?

4. _____

Kurze Werbeunterbrechung



Wieviel Geld würden Sie für mein Programm bieten?



Die Stiftung Warentest rät...

Kompressionsfaktor unter 50%?

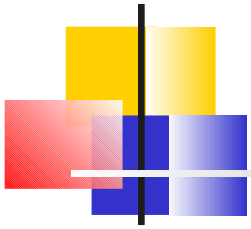
- Programme sind auch nur Folgen!
- Es gibt $1+2+4+8+...+2^{n/2-1} = 2^{n/2}-1$ Folgen mit weniger als n/2 Bits (nicht alle sind Programme)
- Aber “4.” ist nur eine von 2^n Folgen der Länge n
- Höchstens jede $2^{n/2}$te Folge der Länge n ist auf unter 50% komprimierbar!
- Nur 1/1000 aller Programme mit Länge 1.000.000 läßt sich auf unter 999.990 Bits komprimieren!

Die wahrscheinlichste Lösung...

Print the number
“_____”.

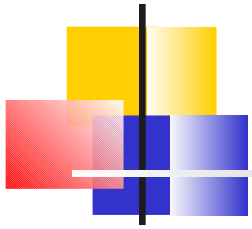
- Falls K(x) ≥ x, nennen wir x *inkompressibel.*
- *Maximale* Komplexität von String der Länge n ist gleich n (plus konstanter Faktor c), wenn n bekannt ist (bedingte Komplexität) - immerhin!
- Ansonsten müssen wir dem Computer auch die Länge n des Strings mitteilen
- Die Bit-Darstellung von n hat Komplexität

$$\log_2 n + \log_2 \log_2 n + \log_2 \log_2 \log_2 n + \ldots \text{ (addiere solange Summanden positiv sind)} \in O(\log n)$$



“Aber bei Winzip klappt es doch...”

- Auch Winzip kriegt nicht alles klein! (z.B. eine andere Winzip-Datei)
- Regelmäßigkeiten können ausgenutzt werden (Byteschema, Buchstaben-, Worthäufigkeiten)
- Verschiedene Kompressionsraten: Mit Winzip kann man Sprachen und Musikstile erraten...
- Die meisten Folgen sind inkompressibel, aber nicht so die meisten von Menschenhand erzeugten Folgen

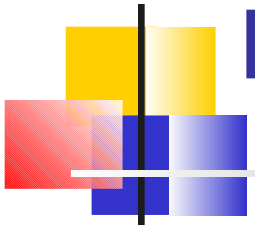


Komplexität ist universell

- DEFINITION. Ein Dekompressionsalgorithmus U ist “asymptotisch nicht schlechter” als ein anderer Algorithmus V, wenn für jeden Datensatz x gilt:

$$K_U(x) \leq K_V(x) + C$$

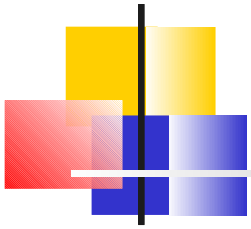
- THEOREM. Es gibt einen Dekompressionsalgorithmus U, der asymptotisch nicht schlechter als jeder andere Algorithmus V ist. (Warum? Das sehen wir gleich.)
- Im Folgenden statt $K_U(x)$ nur $K(x)$



Nochmals: Komplexität ist universell!

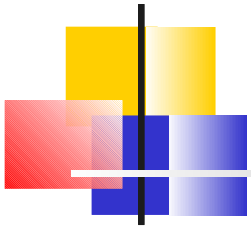
- Länge des kürzesten Programms für x hängt nicht von der Programmiersprache ab (bis auf konstante Faktoren, bei großen x vernachlässigbar)
- Für Informatiker: (Fast) alle Programmiersprachen sind ineinander überführbar
- Das SuperZip-Argument ist unabhängig vom verwendeten Kompressionsverfahren
- Kolmogoroff-Komplexität ist *inhärente Eigenschaft* von Strings (und auch von Zahlen)

Wie basteln wir uns unser U?



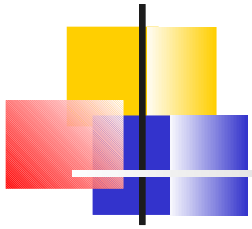
- Angenommen, wir wollen eine Datei in komprimierter Form an einen Freund verschicken, aber wir wissen nicht genau, ob er das Dekompressionsprogramm hat.
- Was tun? Wir schicken das Programm selbst und hängen die Datei ans Ende. (vgl. Selbstextrahierendes Zip-Archiv .EXE)
- U funktioniert wie der Computer des Freundes: Es sucht im Eingabestring x nach einen Programm p in einer festen, selbstbeschränkenden Programmiersprache (z.B. Java oder TM-Nummern) und wendet es auf x an.

Warum ist U universell?

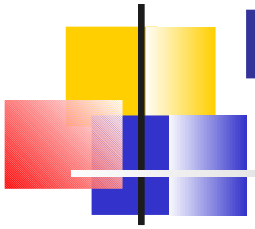


- Betrachte jedes andere Dekompressionsverfahren V. Sei `V.java` ein C Bits langes Programm, das V implementiert. Sei x ein beliebiger Datensatz. Sei `X.zip` die Eingabe, das von V nach x dekomprimiert wird.
- $K_V(x)$ = Länge(`X.zip`)
- U dekomprimiert die Eingabe `V.java+X.zip` zu x. $K_U(x)$ = Länge(`V.java+X.zip`)
- `V.java+X.zip` ist nur C Bits länger als `X.zip` (C hängt nicht von x ab)
  - **$K_U(x) \leq K_V(x) + C$** **--- für alle x !!!**

K(x) als zweiteiliger Code

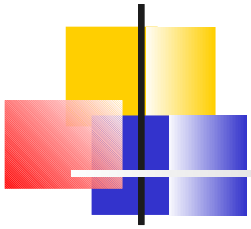


- Betrachte x = 10101010101010101010101010
- K(x) besteht aus:
    - einem Teil, der “10” enthält (Daten), und
    - einem Teil, der die Anweisung zum 13maligen Drucken enthält (Algorithmus)
- “Wertvolle” Information “`10`”+”`Repeat 13 times`” wird durch Funktion K(x) von “wertloser” herausdestilliert
- *Wie* das geht, darüber wissen wir nichts



Beachte...

- Im Rahmen von Kolmogorov-Komplexität gibt es nur Dekompressionsverfahren
- Für (fast) jeden Datensatz wäre das optimale Kompressionsverfahren ein anderes
- Es gibt keine zeitliche Beschränkung für den Dekompressionsalgorithmus (die Maschine U)
- K(x) mißt nicht “logische Kompliziertheit” - zufälliger Buchstabensalat erhält höhere Werte als menschliche DNA

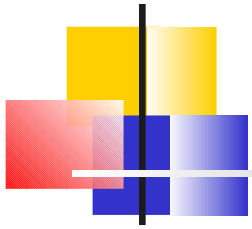


Shannon vs. Kolmogorov

http://www.cwi.nl/~pdg/presentations/NVTI6.PDF

- Beide Theorien messen *Informationsgehalt* in *Bitfolgen*
- Shannons Entropie H(P) mißt *erwarteten* Informationsgehalt beim Beobachten des Ergebnisses einer Zufallsvariable
  - geht von Wahrscheinlichkeitsverteilung P aus
  - Funktion von Verteilungen nach $\mathfrak{R}$
- Kolmogorov-Komplexität K(x) mißt *tatsächlichen* Informationsgehalt in einer *bestimmten* Folge
  - keine Annahmen über Wahrscheinlichkeiten
  - Funktion von Symbolfolgen (oder Zahlen) nach N

Pro und kontra Kolmogorov



- Kontra:
  - K(x) ist nicht berechenbar
  - asymptotische Theorie: für “kurze” Sequenzen nicht aussagekräftig
- Pro:
  - K(x) ist von oben (theoret.) approximierbar
  - es ist erstaunlich, daß “Informationsgehalt einer Folge” überhaupt objektiv definiert werden kann!
  - unabhängig von Wahrscheinlichkeits-verteilungen (praktisch z.B. in Statistik) und verwendetem Formalismus

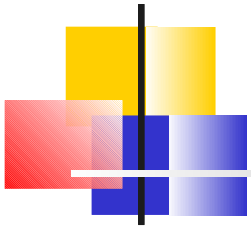
Erweiterung: Unendliche Folgen

- Bisher haben wir K(x) nur für endlich lange Folgen x definiert
- DEFINITION. Eine unendliche Folge x ist *inkompressibel*, wenn

$$\lim_{n\to\infty} \frac{K(x_1x_2x_3\ldots x_n|n)}{n} = 1$$

- THEOREM. In jeder inkompressiblen unendlichen Folge strebt das Verhältnis von Nullen und Einsen gegen 1:1.

Zusammenfassung



- Die *Kolmogorov-Komplexität* K(x) des Strings x ist die Länge (in Bits) des kürzesten Programms, das x ausdruckt und dann anhält.
- Die weitaus meisten Strings sind *inkompressibel*.
- K(x) ist bis auf einen konstanten Faktor *unabhängig* von der verwendeten Programmiersprache.
- K(x) ist *nicht berechenbar*.
- K(x) ist *unabhängig* von Wahrscheinlichkeiten.
- Die meisten “interessanten” Strings haben *geringe*, aber *nicht minimale* Komplexität.

Vielen Dank!

Andrei Nikolaevich Kolomogorov
(1903–1987)