Kolmogorov Komplexität

Was unterscheidet die folgenden Zahlenfolgen?

(a) 001011100110101000011100100100101111111011

(b) 010101010101010101010101010101010101010101

(c) 111111111111111111111111111111111111111111

Folgen (b) und (c) weisen leicht zu erkennende Regularitäten auf, während dies bei (a) anscheinend nicht der Fall ist. Tatsächlich ist (a) durch Werfen einer Münze entstanden: Folge (a) erscheint „zufälliger“ als die anderen zwei. Andererseits, im Sinne der Wahrscheinlichkeitsrechnung sind alle drei Folgen gleich (un)wahrscheinlich ($p = 2^{-42}$).

- $\Rightarrow$ Die Wahrscheinlichkeitsrechnung sagt uns nicht, wie „Zufälligkeit“ sinnvoll definiert werden kann.
- Im normalen Sprachgebrauch sind zufällige Ereignisse die, bei denen wir kein Bildungsgesetz finden können, dass es uns erlauben würde, die Eigenschaften der Ereignisse exakt vorauszusagen. Idee $\Rightarrow$ Algorithmik.
- Schreiben wir ein Programm auf, dass die jeweilige Folge erzeugen kann:

(a) **`output`** 001011100110101000011100100100101111111011;

(b) **`for`** $i := 1$ **`to`** 21 **`output`** 01;

(c) **`for`** $i := 1$ **`to`** 42 **`output`** 1;

- Wenn wir von der $Länge 42 abstrahieren, dann hat das Programm für (c) nur eine Länge von $O(1) + \log n$, wobei $\log n$ für die Darstellung der Zahl n benötigt wird.
- Hingegen hat Programm (a) eine Länge von $O(1) + n$, dh um diese zufällige Folge zu beschreiben, müssen wir sie im wesentlichen selbst hinschreiben.
- Zufälligkeit werden wir dann so definieren, dass man im wesentlichen n Bits braucht, um einen zufälligen $ der Länge n zu beschreiben.
- Alternativ dazu kann man auch sagen, dass der $ (a) komplizierter (im Sinne seiner Beschreibung) als (b) oder (c) ist. Das führt zum Begriff der deskriptiven oder Kolmogorov Komplexität endlicher oder unendlicher $s.

Variante: Wenn der Algorithmus eine Eingabeanweisung **`input`** y besitzt, dann gibt die notwendige Programmlänge, um x unter Kenntnis von y zu generieren, gewissermaßen die relative Zufälligkeit von x gegenüber y an, bzw die Information, die y über x enthält.

Definition: Seien $x, y, p \in \{0,1\}^*$. Jede berechenbare partielle Funktion Φ, zusammen mit beliebigem p und y sodass $\Phi(\langle p, y\rangle) = x$, ist eine Beschreibung von x. Die Kolmogorov Komplexität von x relativ zu y in Bezug auf Φ ist

$$K_\Phi(x|y) = \min\{|p| : \Phi(\langle p, y\rangle) = x\},$$

und $K_\Phi(x|y) = \infty$ falls es kein solches p gibt. Wenn $y = \varepsilon$ (der leere $), dann ist $K_\Phi(x|\varepsilon) = K_\Phi(x)$ die (absolute) Kolmogorov Komplexität, dh die Länge des kürzesten Programms in der „Programmiersprache“ Φ (zB Java, eine 1$ TM, ...), das x ausgibt.

Die Wahl der Programmiersprache könnte irgenwie willkürlich erscheinen. Tatsächlich kann man aber durch einen Interpreter jede Programmiersprache auf einer universellen (Turing-)Maschine simulieren.

Theorem: (Invarianz) Es gibt eine berechenbare partielle Funktion Φ sodass für jede andere berechenbare partielle Funktion Φ_n es eine Konstante c_n gibt sodass

$$K_{\Phi}(x|y) \leq K_{\Phi_n}(x|y) + c_n \quad \forall x,y \in \{0,1\}^*$$

Beweis: c_n ist die Länge des Interpreterprogramms. $\surd$

Dh die unterschiedlichen Programmiersprachen unterscheiden sich nur durch eine additive Konstante. Im folgenden fixieren wir eine universelle TM U und eine Funktion Φ und schreiben nur noch $K(x|y)$.

Es kann nicht sehr viele \$s mit kleiner Kolmogorov Komplexität geben: Es gibt ja immer höchstens 2^k Programme der Länge k, und daher auch nur 2^k \$s x mit $K(x) = k$. Die 2^n \$s der Länge teilen sich n wie folgt auf:

höchstens 1 \$ hat K-Komplexität $= 0$
höchstens 2 \$s haben K-Komplexität $= 1$
höchstens 4 \$s haben K-Komplexität $= 2$
⋮
höchstens 2^{n-1} \$s haben K-Komplexität $= n-1$

Allgemein gilt: die Anzahl der \$s mit K-Komplexität $\leq k$ ist höchstens $1 + 2 + \cdots + 2^k = 2^{k+1} - 1$. Umgekehrt:

mindestens 1 \$ hat K-Komplexität $\geq n$
mehr als die Hälfte der 2^n \$s hat K-Komplexität $\geq n-1$
mehr als $3/4$ der 2^n \$s hat K-Komplexität $\geq n-2$
mehr als $7/8$ der 2^n \$s hat K-Komplexität $\geq n-3$, usw. $\triangleright$

Dh, der Großteil aller \$s hat hohe K-Komplexität, ie die ensprechenden \$s sind nicht gut komprimierbar und nicht kompakt beschreibbar, sie sind also zufällig.

Bemerkung: $K(x)$ ist nicht berechenbar: Denn angenommen, $x \mapsto K(x)$ sei berechenbar durch ein (immer stoppendes) Programm M. Dann sei **Programm** P_m:

```
x := ε;
repeat x := Nachfolger von x;
until M(x) gibt einen Wert ≥ m aus;
output x;
```

Jedes P_m beschreibt den lexikographisch ersten \$ mit $K(x) \geq m$. Da P_m aber gerade x_m beschreibt, gilt $K(x_m) \leq |P_m| = 2 \times O(1) + \log m$. Für große m gilt $O(1) + \log m < m$ und daher führt die Annahme, dass ein solches M existiert zu einem Widerspruch. $\surd$

Universelle Wahrscheinlichkeitsverteilung

Für jede \$Länge n definieren wir eine Wahrscheinlichkeitsverteilung m, so dass $m(x)$ die Wahrscheinlichkeit für das Auftreten von x, $|x| = n$ bedeutet. Es muss gelten dass $\sum_{x:|x|=n} m(x) = 1$. Wir wollen m so definieren, dass $m(x)$ proportional zu $2^{-2K(x|n)}$ ist, dh $m(x) = c2^{-2K(x|n)}$. Dazu muss $\sum_{x:|x|=n} 2^{-2K(x|n)} = d$ für konstantes $d = 1/c$ gelten:

$$\begin{aligned} \sum_{x:|x|=n} 2^{-2K(x|n)} &\leq 2^{-2n} + \sum_{i=0}^{n-1} 2^i 2^{-2i} \\ &= 2^{-2n} + \sum_{i=0}^{n-1} 2^{-i} \\ &\leq 2 \qquad \surd \end{aligned}$$

Worst case vs. average case

Beispiel: Quicksort braucht bekanntermaßen im Durchschnitt $O(n \log n)$ Schritte, wobei Eingaben der selben Länge gleichverteilt angenommen sind. Bei sortierter (oder auch umgekehrt sortierter) Eingabefolge (und naiver Implementierung) sind es aber $O(n^2)$ (worst case). Was wäre die durchschnittliche Zeitkomplexität unter der universellen Wahrscheinlichkeitsverteilung?

Sei $\{1,2,\ldots,n\}$ zu sortieren. Die längenbedingte Kolmogorov Komplexität der sortierten Folge ist $K((1,2,\ldots,n)|n) = O(1)$. Unter der universellen Wahrscheinlichkeitsverteilung ist das Auftreten der sortierten Folge besonders groß (Plausibilität?): konstant für alle n: $m((1,2,\ldots,n)) = c2^{-2K((1,2,\ldots,n)|n)} = c2^{-O(1)} =: \alpha$. ▷

Damit ist der Erwartungswert der Rechenzeit unter m:

$$\begin{aligned} & \sum_{x:|x|=n} m(x) T_{\text{Quicksort}}(x) \\ & \quad \geq \quad m((1,2,\ldots,n)) T_{\text{Quicksort}}((1,2,\ldots,n)) \\ & \quad = \quad \alpha \cdot \Omega(n^2) = \Omega(n^2) \end{aligned}$$

Dabei ist $T_A(x)$ die Rechenzeit von Algorithmus A bei Eingabe x. Dh, unter der universellen Wahrscheinlichkeitsverteilung ist der average case gleich dem worst case (bis auf einen konstanten Faktor).

Bemerkung: Die einzige Eigenschaft von $x = (1,2,\ldots,n)$ und $A =$ Quicksort, die für den Beweis wichtig war, war das x eine Eingabe war, bei der A sein worst case Verhalten an den Tag legt. $\rightarrow$ Wir verallgemeinern für A, einen beliebigen immer stoppenden Algorithmus: ▷

`Algorithmus` H
$w := 0$; **`input`** n;
`for` (alle y mit $|y| = n$ — in lexikographischer Ordnung)
`do`
  $v := T_A(y)$;
  `if` $v > w$ **`then`** $w := v; x := y$ **`endif`**;
`enddo`; **`output`** x;

Algorithmus H hat eine feste (aber von A abhängige) Länge c. Für jedes n sei x_n die Ausgabe von diesem Algorithmus. Somit gilt $K(x_n|n) \leq c$ und damit ist $m(x_n)$ proportional zu $2^{-2K(x_n|n)} \geq 2^{-2c}$. Dh, für eine von n unabhängige Konstante α gilt $m(x_n) \geq \alpha$. Außerdem gilt aufgrund der Konstruktion von H, dass $T_A(x_n)$ maximal ist unter allen Eingaben der Länge n. ▷

Dh bei Eingabe x_n legt A sein worst case Zeitverhalten $T_A^{wc}(n)$ an den Tag.

Damit ist der Erwartungswert der durchschnittlichen Rechenzeit (average case) unter der universellen Wahrscheinlichkeitsverteilung m für einen beliebigen Algorithmus A, $T_A^{ac,m}(n)$:

$$\begin{aligned} T_A^{ac,m}(n) &= \sum_{x:|x|=n} m(x) T_A(x) \\ &\geq m(x_n) T_A(x_n) \\ &= m(x_n) T_A^{wc}(n) \\ &\geq \alpha \cdot T_A^{wc}(n) \\ &= \Omega(T_A^{wc}(n)) \qquad \surd \end{aligned}$$

Untere Schranken

"A man, a plan, a canal, Panama!"

Technik: Man nehme einen \$ der Länge n (n genügend groß) mit maximaler Kolmogorov Komplexität, also $K(x) \geq n$. Wenn man nun annimmt, dass es ein Programm gibt, dass weniger als in dem zu beweisenden Theorem behauptet viele Schritte macht, dann gibt es eventuell eine Möglichkeit, den \$ durch Angabe des Programms und weiterer Informationen mit weniger als n Bits zu beschreiben, was ein Widerspruch wäre.

Theorem: Die Sprache $L = \{w0^{|w|}w^R \mid w \in \{0,1\}^*\}$ benötigt auf einer 1\$ TM mindestens $\Omega(n^2)$ Rechenzeit (es ist $(a_1 \dots a_k)^R = (a_k \dots a_1)$).

▷

Sei M eine 1\$ TM und bezeichne i die Schnittstelle zwischen Feld i und $i+1$ auf dem \$. Die Crossing-Sequenz $CS_M(x,i) \in Z^*$ sei die Folge von Zuständen aus Z, die in der Rechnung von $M(x)$ beim Überschreiten von Position i durch den Cursor durchlaufen werden.

Offensichtlich gilt $\sum_{i=-\infty}^{\infty} |CS_M(x,i)| = T_M(x)$.

Wir nehmen oBdA an, dass alle unsere TM nur zu solchen Zeitpunkten in den Endzustand übergehen können, wenn ihr Cursor am Beginn des \$s steht.

Lemma: Sei $|x| = i$. Wenn $CS_M(xy,i) = CS_M(xz,i)$, dann:

$$xy \in L \Leftrightarrow xz \in L\,.$$

▷

Beweis: Da links von i die Rechnungen von M identisch verlaufen müssen und M links stoppt (wegen dem „oBdA") muss der Endzustand in beiden Fällen gleich sein. $\surd$

Lemma: Wenn $c = CS_M(xy,i) = CS_M(x'y',i)$, dann auch $c = CS_M(xy',i) = CS_M(x'y,i)$.

Beweis: Rechts und links Rechnungen zusammenfügen: Es ändert sich nichts am jeweiligen Ablauf (rechts bzw links). $\surd$

Lemma: Sei nun M eine 1\$ TM, die L akzeptiert. Sei $w0^{|w|}w^R$ eine Eingabe und sei $|w| \leq i \leq 2|w|$. Aufgrund der obigen Lemmas gilt, dass für verschiedene w, w' mit $|w| = |w'|$ auch die Crossing-Sequenzen $CS_M(w0^{|w|}w^R)$ und $CS_M(w'0^{|w'|}w'^R)$ verschieden sein müssen.

▷

Beweis: Angenommen, die Crossing Sequenzen wären gleich. Man wende das 2. Lemma an und erhält, dass auch die Crossing Sequenzen (an Position i) von $w0^{|w|}w^R$ und $w0^{|w|}w'^R$ gleich sind. Nun wende man das 1. Lemma an mit $x = w0^{i-|w|}$, $y = 0^{2|w|-i}w^R$ und $z = 0^{2|w|-i}w'^R$. Dabei ist $xy \in L$ und $xz \notin L$. Dies ist ein Widerspruch, daher müssen die Crossing-Sequenzen verschieden sein. $\surd$

Daher ist w aus M, n, i, c eindeutig bestimmt:

Algorithmus Reconstruct(M,n,i,c)
for $w : |w| = n$ **do**
Simuliere M auf $w0^{|w|}w^R$ und notiere dabei die Crossing-Sequenz an der Position i. Falls diese mit c übereinstimmt, dann **output** w;
end

▷

Wir schließen daraus: $K(w) \leq O(\log|w|) + |c|$. Sofern w mit maximaler Kolmogorov Komplexität gewählt wurde, also $K(w) \geq |w|$, so folgt (nun kommt das Kolmogorov-Beweisargument) $|c| \geq |w| - O(\log|w|)$. Daraus folgt:

$$\begin{aligned}
T_M(x) &= \sum_{i=-\infty}^{\infty} |CS_M(x,i)| \\
&\geq \sum_{i=n/3}^{2n/3-1} |CS_M(x,i)| \\
&\geq \sum_{i=n/3}^{2n/3-1} (n/3 - O(\log n)) \\
&\geq n^2/9 - O(n \log n) \\
&= \Omega(n^2) \qquad \checkmark
\end{aligned}$$