# Ein Algorithmus zur Bewältigung von Zahlenfolgen-Fortsetzungsaufgaben

**Johannes Knabe** JKNABE@UNI-OSNABRUECK.DE
Cognitive Science Bachelor Programm, Universität Osnabrück

**Hauke Strasdat** HSTRASDA@UNI-OSNABRUECK.DE
Cognitive Science Bachelor Programm, Universität Osnabrück

## Abstract

Diese Ausarbeitung beschäftigt sich mit der Frage, in wie weit sich Intelligenztests, insbesondere Zahlenfolgenfortsetzungsaufgaben, algorithmisch lösen lassen.
Wir stellen einen Algorithmus vor, mit dem Zahlenreihen fortgesetzt werden koennen. Solche Aufgaben zum induktiven Schliessen mit Zahlen sind Bestandteil vieler Intelligenztests. Nach einer kurzen Einfuehrung in die Intelligenzmessung mit Tests, gehen wir auf das Problem der Fortsetzung von Zahlenreihen ein. Es werden einige generelle Probleme aufgezeigt - insbesondere das Fehlen eines objektiven Einfachheitsmasses zur Auswahl der plausibelsten Fortsetzung. Nicht zuletzt zeigen wir auch Parallelen zum menschlichen Verhalten beim lösen derartiger Probleme auf und die Grenzen des Verfahrens bei Folgen wie bspw. den Primzahlen, die Menschen mit Weltwissen lösen.

## 1. Einführung: Intelligenztest, Zahlenfolgen, Generalisierbarkeit

### 1.1. Intelligenztest

"Intelligenz ist das, was ein Intelligenztest misst" wurde bereits Anfang des 20. Jahrhunderts festgestellt. Diese Aussage ist seitdem genauso umstritten wie die diversen erhältlichen Tests; da wir aber dennoch ein (partiell) intelligentes Programm schreiben wollten, mussten wir einen Test stellvertretend auswählen. Zu den gebräuchlichsten und bekanntesten IQ-Tests zählen der Stanford-Binet-Test, der Wechsler-Intelligenztests mit seinen deutschen Varianten HAWIE (für Erwachsene) und HAWIK (für Kinder), der Intelligenz-Struktur-Test von Amthauer – der IST-70 – sowie verschiedene Tests, die auf Raymond Cattel zurückgehen. Der für unsere Untersuchungen gewählte Test ist der IST-2000 R, eine überarbeitete Version des vorgenannten IST-70[1]. Dieser ist wie - außer dem Wechsler-Test - alle obenstehenden Tests ein reiner Papier- und Bleistift-Test. Ein weiteres wichtiges Auswahlkriterium war, dass sich Teilbereiche dieses Tests nur um Zahlen drehen - und da wir so wenig wie möglich modellieren bzw. Objekterkennung betreiben (man denke an geometrische Figuren) wollten, eignen sich diese Aufgaben - repräsentativ für viele ähnliche - besonders gut für unser Projekt. Desweiteren handelt es sich bei Zahlenreihenaufgaben um gute Beispiele für induktives schliessen, d.h. es wird von einer gefundenen Regelmaessigkeit angenommen, dass diese auch die Fortsetzung der Folge bestimmt.

### 1.2. Zahlenfolgen[2]

Ein spezieller Typ von Aufgabe in diesem Bereich sind die Fortsetzungsfolgen: Eine Sequenz von Zahlen ist gegeben und die Testperson muss eine Regelmäßigkeit hinter den Zahlen erkennen, um den Nachfolger (oder mehrere) herauszufinden. Ein einfaches Beispiel:

$$3, 5, 7.$$

Ziemlich schnell sieht man, dass 3, 5 und 7 aufeinander folgende, ungerade Zahlen sind. Man könnte also meinen, die Lösung sei

$$3, 5, 7, 9.$$

[1] Amthauer, R., Bröcke, D., Liepmann, D., Beaducel, A. (2001) I-S-T 2000 R - Test Manual. Göttingen: Hogrefe Verlag

[2] Die Beispiele aus diesem Kapitel stammen von: Bousquet, Olivier (2003) Statistical Learning Theory. Kurs beim Interdisziplinären Kolleg 2003. Günne am Möhnesee

Dies ist jedoch nicht die einzige Möglichkeit. 3, 5 und 7 sind nämlich auch Primzahlen, also wäre

$$3, 5, 7, 11$$

auch eine mögliche Lösung. Nun stellt sich die Frage, ob sich diese Ambiguität aufhebt, wenn man die Größe der Zahlenfolge erhöht. Geben sei:

$$3, 5, 7, 13.$$

Aber auch hier lassen sich wieder verschiedene Regelmäßigkeiten entdecken. Was also ist die beste Fortsetzung unserer Reihe? Schnell ist man versucht nach Aristoteles und Occam's Razor die 'einfachst' Möglichkeit zu fordern. Jemand, der nicht besonders mit der Mathematik vertraut ist, könnte zum Beispiel auf folgende simple Idee kommen: Die Folge enthält alle Zahlen, die auf 3, 5 und 7 enden:

$$3, 5, 7, 13, 15, 17, 23, ...$$

Eine andere Regel wäre: Die Folge besteht aus allen Primzahlen, die nicht auf 1 enden:

$$3, 5, 7, 13, 17, 19, 27, ...$$

Die beiden letzten Beispiele beruhen stark auf der Repräsentation (Dezimalsystem). Dies fällt besonders auf, wenn man die Repräsentation ändert. Die Zahlenfolge 3, 5, 7 und 13 im Dualsystem:

$$11, 101, 111, 1101.$$

Hier gibt es eine besonders einfache Regularität: Schiebe eine 0 in die vorletzte Position. Ersetze die 0 durch eine 1 usw.:

$$11, 101, 111, 1101, 11101, 11111, 111101, ...$$

Die selbe Folge im Dezimalsystem:

$$3, 5, 7, 13, 15, 29, 31, ...$$

Die Regelmäßigkeit, die im Dualsystem so offensichtlich war, ist hier ziemlich versteckt. Mit anderen Worten: Diese Regelmäßigkeit ist im Dualsystem viel einfacher zu entdecken als im Dezimalsystem. Dies ist ein Indiz für ein grundlegendes Problem: Die Einfachheit einer Struktur ist relativ zur Repräsentation der Daten und dem Vorwissen des Betrachters! Um also die Fortsetzung einer Folge zu finden, die in unserer Kultur am ehesten als sinnvoll erachtet wird, müssen wir gewisse Annahmen über das Vorgehen von Menschen bei der Lösung von Zahlenfolgenproblemen machen.

## 2. Schätzen einer Folgezahl

Im folgenden werden wir ein Algorithmus vorstellen, der Intelligenztests mit Zahlenfolgen lösen kann.
Der Basisalgorithmus beruht auf folgender Grundannahme:

> Die Zahlen in der Folge ergeben sich aus einer mathematischen Grundrechenoperationen, und zwar jeweils zwischen zwei unmittelbar aufeinanderfolgenden Zahlen - für die Erweiterung zur Erkennung alternierender Zahlenfolgen siehe 2.2.

Für Folgen, bei denen diese Grundannahme nicht zutrifft, da entweder die angewendete Operation nicht bekannt ist, mehrere Vorgänger in Betracht gezogen werden oder ganz allgemein Weltwissen vonnöten ist, scheitert der Algorithmus.

### 2.1. Der Algorithmus

In diesem Abschnitt wollen wir den Algorithmus an einigen Beispielen vorstellen. Als bekannte Operation setzen wir zunächst nur die Addition und deren Umkehrfunktion, die Subtraktion voraus. Im folgenden haben wir den Algorithmus in folgende, immer wieder referenzierte Schritte unterteilt:

**algo**(Folge A)

1. WENN Folge konstant DANN RETURN lastElement(Folge) SONST
   *Wenn die Folge konstant ist, gebe ein Element der Folge als Ergebnis zurück, ansonsten:*

2. WENN length(Folge) $\leq 2$ DANN STOP SONST
   *Wenn die Folge aus nur zwei Elementen besteht, brich ab, wenn nicht:*

3. neue-Folge $\leftarrow$ combine-elements(Folge, -)
   *Verknüpfe mittels Subtraktion alle Zahlen der Folge (außer die erste) mit ihrem Vorgänger(!). Aus den Ergebnissen entsteht die neue Folge.*

4. RETURN lastElement(Folge) + **algo**(neue-Folge)
   *Verknüpfe mit der Addition den Rückgabewert von algo(Folge B) und das letzte Element von Folge. Gebe diese Zahl als Ergebnis zurück.*

Der Algorithmus versucht also mit Hilfe einer Umkehrfunktion (in diesem Beispiel die Subtraktion) die gegebene Folge in eine konstante Folge

umzuwandeln. Gelingt dies, so kann er mit Hilfe der Umkehrfunktion der Umkehrfunktion (hier die Addition) eine Regel generalisieren und so das nachfolgende Element bestimmen.
Zur Veranschaulichung betrachte man folgendes Beispiel:

$$1, 3, 5, 7.$$

Schritt 1,2: Die Folge ist nicht konstant und besteht aus mehr als zwei Elementen, also gehen wir über zu Schritt drei. Verknüpft man jedes Element mit Hilfe der Subtraktion erhält man diese neue Folge: $2, 2, 2$. Im vierten Schritt addiert man $algo(2, 2, 2)$ mit 7. Da die Folge $2, 2, 2$ konstant ist, ergibt $algo(2, 2, 2) = 2$. Also ist das Ergebnis $2 + 7 = 9$ und wir erhalten folgende Zahlenfolge:

$$1, 3, 5, 7, 9.$$

Dieses Vorgehen des Algorithmus scheint uns durchaus intuitiv, da viele Menschen auch zuerst die stark verinnerlichten Grundrechenarten paarweise auf die Elemente der Folge anwenden.

### 2.2. Weitere Operationen

Der Algorithmus lässt sich um beliebig viele Operationen erweitern. In unserer Implementation haben wir neben der Addition/Subtraktion die Multiplikation und deren Umkehrfunktion, die Division, implementiert. Ist die Liste durch wiederholte Rekursion auf zwei Elemente geschrumpft, so bricht der Algorithmus noch nicht ab, sondern versucht das Problem mit der Multiplikation/Division zu lösen. Ein Beispiel:

$$2, 4, 8.$$

Als erstes wird Schritt drei mit der Subtraktion probiert, die resultierende Folge sieht dann so aus:

$$2, 4.$$

Diese Folge ist nicht konstant und besteht aus zwei Elementen. Also ist noch keine Regelmäßigkeit gefunden. Mit der Division gelingt dies jedoch: $4/2 = 2$ und $8/4 = 2$. Wir erhalten die konstante Folge:

$$2, 2.$$

In Schritt vier (beim Rekursionsaufstieg) multiplizieren wir 2 mit 8 und erhalten die gesuchte Folge:

$$2, 4, 8, 16.$$

Auch Folgen, die sowohl auf Addition/Subtraktion und Multiplikation/Division beruhen, kann der Algorithmus lösen, da er in jeden rekursiven Aufruf von *algo* via Backtracking entscheidet, welche Operation angewendet wird. Auch hierfür noch ein Beispiel:

$$2, 4, 12, 48.$$

Schritt drei mit der Subtraktion:

$$2, 8, 36.$$

Dieser Weg wird scheitern, da man im rekursiver Aufruf mit der Subtraktion die Folge $6, 28$ und mit der Division die Folge $4, 4.5$ erhalten würde. Also versuchen wir Schritt drei mit der Division und erhalten:

$$2, 3, 4.$$

Nun wenden wir im rekursiver Aufruf die Subtraktion an:

$$1, 1.$$

Über den Rekursionsaufstieg konstruieren wir die gesuchte Zahl: $4+1 = 5$ und $5*48 = 240$. Die gesuchte Folge ist also:

$$2, 4, 12, 48, 240.$$

### 2.3. Alternierende Zahlenfolgen

Der im letzten Kapitel vorgestellte Basisalgorithmus kann keine alternierenden Zahlenfolgen erfassen. Bei alternierende Zahlenfolgen wechseln sich mehrere Operationen ab. Ein Beispiel wäre

$$2, 5, 10, 13, 26$$

Hier wird abwechselnd 3 addiert und mit 2 multipliziert.
Um alternierende Zahlenreihen zu berücksichtigen, wird das Programm folgendermaßen erweitert: Findet der Algorithmus keine Regelmäßigkeit in einer Folge, wird die sogenannte Schrittweite um eins erhöht und der Algorithmus nochmal ausgeführt. Die Erhöhung der Schrittweite auf n bedeutet zweierlei:

1. Der Algorithmus wird n mal ausgeführt. Einmal mit der Folge in Orginallänge und n-1 mal mit Folgen, bei denen je ein Element mehr vorne fehlt. Das Ergebnis aller Aufrufe lässt sich dann als n-Tupel zusammen fassen.

2. Es wird im dritten Schritt des Algorithmus nicht mehr jedes Paar der Folge betrachtet, sondern nur noch jedes n-te.
   Beim rekursiven Aufruf von *algo* reduziert sich die Schrittweite auf eins. (Sonst würde die Laufzeit unnötig erhöht, da Menschen solch komplexe Beziehungen sicher nicht bemerken würden.)

Um dies zu veranschaulichen, betrachte man unser Beispiel:

$$2, 5, 10, 13, 26$$

Da der Algorithmus bei Schrittweite 1 (in 2.1 beschriebenen Basisalgorithmus) keine Regelmäßigkeit finden kann - dies nachzurechnen ist dem Leser überlassen - erhöht sich die Schrittweite auf 2. Das bedeutet nun folgendes:

1. Der Algorithmus wird zweimal aufgerufen: Einmal mit $algo(2, 5, 10, 13, 26)$ und einmal mit $algo(5, 10, 13, 26)$. Das Ergebnis beider Aufrufe läßt sich als Tupel zusammenfassen.

2. Es wird im dritten Schritt des Algorithmus nur noch jedes 2-te Paar betrachtet. Das bedeutet, dass bei $algo(2, 5, 10, 13, 26)$ nur die Zahlenpaare 2;5 und 10;13 berücksichtigt werden. Daraus läßt sich mit der Subtraktion die konstante Folge $3, 3$ ermitteln. Da die Schrittweite beim rekursive Aufruf von $algo$ auf eins zurückgesetzt wird, ist $algo(3, 3) = 3$ und das Ergebnis wäre $3 + 26 = 29$. Beim zweiten Aufruf $algo(5, 10, 13, 26)$ würden dann die Paare 5;10 und 13;26 berücksichtigt und mit der Division die Folge $2, 2$, herauskommen. Das Ergebnis beim zweiten Aufruf ist dann $2 * 26 = 52$. Die Rückgabewerte beider Aufrufe lassen sich als Tupel zusammenfassen:

$$(29, 52)$$

Nun stellt sich die Frage, welches Skalar aus dem Tupel der 'richtige' Nachfolger der Folge ist. In unserem Beispiel ist es die 29; da von 13 nach 26 die Multiplikation 'dran' war, ist nun von 26 nach 29 wieder die Addition am Zuge.
Welches Element aus dem n-Tupel genommen wird, lässt sich allgemein mit folgender Formel beschreiben:

$$Tupelindex = (Folgelaenge - 1) \bmod Schrittweite$$

Folgenlänge bedeutet die Länge der ursprünglichen Folge. Tupelindex bezeichnet die Position des richtigen Wertes im Tupel, angefangen zu zählen mit der 0 (in untenstehendem Ergebnistriple als Subskripte angegeben).
Beispiel: Bei der Folge

$$5, 6, 4, 6, 7, 5, 7, 8$$

wird abwechselnd 1 addiert, 2 subtrahiert und 2 addiert. Die Schrittweite, die zur Lösung benötigt wird, beträgt also drei. Füttert man den Algorithmus mit dieser Zahlenfolge, so erhält man als Ergebnis folgendes Triple:

$$(9_0, 6_1, 10_2)$$

Um herauszufinden, welches Element des Triples das gesuchte Element ist, benutzt man die oben vorgestellt Formel:

$$Tupelindex = ((8 - 1) \bmod 3) = (7 \bmod 3) = 1$$

6 ist also das gewünschte nachfolgende Element:

$$5, 6, 4, 6, 7, 5, 7, 8, 6$$

## 3. Schlussbetrachtung

Der vorgestellte Algorithmus kann nicht für jede Zahlenfolge den gewünschten Nachfolger angeben;

$$3, 5, 7, 11, 13$$

beispielsweise würde sicher von jeder Person mit mathematischen Vorkenntnissen als Primzahlen erkannt und mit 17 fortgesetzt. Dies ist jedoch nicht das Ergebnis von Berechnungen im Stile 'Sieb der Eratosthenes', sondern einfach Erfahrungswerte. Solches Weltwissen 'hart' zu implementieren scheint sinnlos. Flexibles aneignen von Wissen ist wäre die Lösung. Dies ist jedoch ein allgemeines Problem, welches den Rahmen von Zahlnfortsetzungsproblemen bei weitem übersteigt.
Die Lösung anderer Probleme hingegen, wie etwa Operationen zwischen mehreren Vorgängerwerten (Fibonacci) in Betracht zu ziehen, ist durchaus mit der Grundidee des Algorithmus' kompatibel. Es müsste lediglich nach einer erfolglosen Ausführung die Anzahl der zu betrachtenden Vorgänger um eins erhöht werden und der Algorithmus erneut ausgeführt werden.

## Referenzen


Amthauer, R., Bröcke, D., Liepmann, D., Beaducel, A. (2001) I-S-T 2000 R - Test Manual. Göttingen: Hogrefe Verlag

Bousquet, Olivier (2003) Statistical Learning Theory. Kurs beim Interdisziplinären Kolleg 2003. Günne am Möhnesee