Rekursive Algorithmen und ihre Veranschaulichung

von Andreas Koch

Das Thema „Rekursion“ ist Inhalt der Informatik-Lehrpläne vieler Bundesländer und der Empfehlungen zu Bildungsstandards der Gesellschaft für Informatik (vgl. GI, 2016, S.10). In diesem Beitrag wird exemplarisch untersucht, welche konkreten Algorithmen sich zum Einstieg in dieses Thema in der Informatik-Oberstufe eignen und wie sie sich veranschaulichen lassen.

Wissenschaftlicher Hintergrund

Rekursion ist eine Beschreibung, eine Methode oder ein Vorgang mit Anweisungen oder Regeln, die auf sich selbst beruhen. Der Begriff *Rekursion* stammt vom lateinischen Verb *recurrere* für *zurücklaufen* und ist vom Wort *Iteration* abzugrenzen, das sich vom lateinischen Verb *iterare* für *wiederholen* ableitet.

Mithilfe der obigen Definition lässt sich der Begriff *Algorithmus* wie folgt für die Schulinformatik spezialisieren: Ein rekursiver Algorithmus ruft sich in der Implementierung selbst auf, ein iterativer Algorithmus verwendet hingegen Schleifen, um Anweisungen mehrfach auszuführen.

Jeder rekursive Algorithmus kann in einen äquivalenten iterativen Algorithmus überführt werden und umgekehrt. Rechner führen rekursive Algorithmen iterativ aus, indem sie die Informationen der Selbstaufrufe auf einem Stapel speichern.

Das Laufzeitverhalten iterativer Algorithmen ist in der Regel optimaler als das von rekursiven Algorithmen, da weniger Redundanz auftritt und weniger Zwischenspeicherungen notwendig sind. Dessen ungeachtet sind rekursive Algorithmen bei einigen Problemstellungen eine sinnvolle Lösungsstrategie, insbesondere wenn die Formulierung eines iterativen Algorithmus nicht gelingt.

Lehrpläne

Im Berliner Rahmenlehrplan für die gymnasiale Oberstufe wird für die Schülerinnen und Schüler im Leistungskursfach Informatik folgendes Lernziel festgelegt: Sie „wenden rekursive Verfahren an“ (SenBJS, 2006, S.14). Das Land Baden-Württemberg schreibt als Bildungsstandard für vierstündige Informatikkurse an allgemeinbildenden Gymnasien „Rekursion als Lösungsprinzip“ vor (KM BW, 2008, S.1). Der Lehrplan des Freistaats Bayern enthält für die Jahrgangsstufe 11 im Fach Informatik des achtjährigen Gymnasiums u.a. den Inhaltsstichpunkt „Rekursive Abläufe: rekursiver Methodenaufruf, Abbruchbedingung, Aufrufsequenz“ (ISB, 2004, Abschnitt „Inf 11.1.1 Listen (ca. 29 Std.)“). Der Informatik-Kernlehrplan für die Sekundarstufe II an Gymnasium und Gesamtschule in Nordrhein-Westfalen legt folgende Lernziele fest: Die Schülerinnen und Schüler „stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar“ und „implementieren rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen“ (MSB NRW, 2014, S.29).

Die Kompetenzformulierungen und Inhaltsstichpunkte zeigen, dass im Informatikunterricht der Oberstufe konkrete rekursive Algorithmen zu behandeln sind.

Klassen rekursiver Algorithmen

Rekursive Algorithmen lassen sich in vier Klassen einteilen. Manche Algorithmen können auch mehreren Klassen zugeordnet werden.

Logische Algorithmen

Algorithmen zur Lösung logischer Probleme wie z.B. *Die Türme von Hanoi*, Palindrom-Test, Permutation einer Zeichenkette (bei ABC also ABC, ACB, BAC, BCA, CAB, CBA) und *Sudoku* führen zumeist in natürlicher Weise auf eine rekursive Lösungsmethode. Algorithmen dieser Klasse haben für den Schulunterricht die größte Relevanz, da sie die Daseinsberechtigung rekursiver Algorithmen neben iterativen Algorithmen motivieren.

Mathematische Algorithmen

Algorithmen, die mathematische Funktionen bestimmen, wie z.B. Fibonacci-Zahlenfolge, Potenzfunktion,

Summenfunktion und Verzinsungsfunktion, bieten sich aufgrund ihres zumeist geringen Schwierigkeitsgrads an, um das Implementieren konkreter Methoden zu üben. Anhand dieser Algorithmen kann folgendes allgemeines Vorgehen zur Aufstellung rekursiver Methoden erarbeitet werden: Bestimmung der Parameter und ihrer Datentypen, Identifizierung aller zu ändernden Parameter samt der Art der Änderung, Bestimmung des Basisfalls bzw. der Abbruch- oder Ausführungsbedingung, Aufstellung der Rekursionsvorschrift und Implementierung der Methode.

Oftmals gibt es zu einem rekursiven mathematischen Algorithmus eine einfachere äquivalente iterative Variante. Die Gegenüberstellung beider Algorithmen schärft die Beurteilungskompetenz der Schülerinnen und Schüler und sensibilisiert sie dafür, über eigene Lösungen zu reflektieren.

Grafische Algorithmen

Algorithmen mit grafischer Ausgabe – wie z. B. Zeichenmethoden für die Fraktale *Kochsche Schneeflocke*, *Sierpinski-Dreieck* und *Y-Baum* – besitzen einen hohen motivationalen Charakter. Sie erleichtern das Auffinden von Programmierfehlern, da sie diese wortwörtlich sichtbar machen. Für die Implementierung grafischer Algorithmen bietet sich die didaktische „Turtle"-Bibliothek an, die für viele Entwicklungsumgebungen verfügbar ist. Sie stellt rudimentäre Operationen bereit, um die Kriechspur einer Schildkröte zu zeichnen. Gängige Methoden sind `Forward(double strecke)`, um das Turtle-Objekt ausgehend von der aktuellen Position um die Länge `strecke` in Blickrichtung geradeaus zu bewegen und `Left(double x)` bzw. `Right(double x)`, um die Blickrichtung des Turtle-Objekts um `x` Grad nach links bzw. rechts zu drehen.

Komplexere Algorithmen

Algorithmen, denen ein komplexes Prinzip oder ein eigenständiges Problem zugrunde liegt, wie beispielsweise das Sortierverfahren QuickSort, sind sinnvolle Lerninhalte für separate Unterrichtseinheiten, die sich idealerweise an eine Einheit zum Thema *Rekursion* unmittelbar anschließen.

Problemstellungen

Die Auswahl einer geeigneten Problemstellung für den Einstieg in eine Einheit zum Thema *Rekursion* sollte sich an zwei Leitfragen orientieren: Ist das Prinzip der Rekursion für die Schülerinnen und Schüler unmittelbar ersichtlich und verständlich? Führt die Problemstellung in natürlicher Weise auf eine implementierbare Rekursionslösung?

Im Folgenden werden vier konkrete Einstiege dargestellt und auf ihre Geeignetheit hin überprüft.

Rollenspiel „Summenbildung"

Vier Schülerinnen bzw. Schüler stellen sich in eine Reihe. Sie erhalten jeweils – verdeckt für die anderen – ein Blatt Papier, das mit einer natürlichen Zahl bedruckt ist, und zwar beginnend mit der Zahl 1 und fortlaufend aufsteigend. Der vierte Schüler soll nun die Summe aller Zahlen bestimmen. Dabei darf jeder Schüler ausschließlich der rechts neben ihm stehenden Person eine einzige Frage stellen, auf die diese – auch mit Zeitverzögerung – antworten darf.

Das beschriebene Problem lässt sich rekursiv lösen, indem der vierte Schüler den dritten nach der ihm bekannten Teilsumme fragt, der den zweiten fragt, der wiederum den ersten fragt. Der erste Schüler kann dem zweiten Schüler mit der Zahl 1 antworten, da rechts neben ihm keine weitere Person steht. Der zweite Schüler kann dem dritten mit der Zahl 3 antworten, der dritte Schüler dem vierten mit der Zahl 6 und der vierte kann so das Ergebnis 10 bestimmen. Dieses Vorgehen führt unmittelbar auf eine rekursive Methode (siehe Bild 1), wobei der erste Schüler den Basisfall repräsentiert und der vierte Schüler den Selbstaufruf `Summe(3)` ausführt, um das Ergebnis von `Summe(4)` zu bestimmen.

Allerdings ist das Szenario des Rollenspiels künstlich – der rekursive Algorithmus entsteht aufgrund der Vorgaben. Eine iterative Methode ist naheliegender, denn sie beruht auf dem aus dem Alltag bekannten Prinzip des fortlaufenden Abzählens bzw. Durchnummerierens, bei dem die Objekte einmal und nacheinander erfasst werden. Daher eignet sich das Rollenspiel nicht als Einstieg in eine Unterrichtseinheit. Allerdings kann es im Rahmen einer Übungsphase zur Veranschaulichung der Summe-Methode verwendet werden und als Ausgangspunkt dienen, um Rekursion der Iteration gegenüberzustellen.

Fibonacci-Zahlenfolge

Folgendes Szenario geht auf den Mathematiker Leonardo Fibonacci (1170–1240) zurück: Eine Kaninchenpopulation besteht zu Beginn aus einem einzigen Paar, einem Männchen und einem Weibchen. Ein Paar wird nach einem Monat geschlechtsreif und wirft ein weiteres Kaninchenpaar pro Monat. Die Kaninchen können sich unbeschränkt vermehren und sind unsterblich. Die zugehörige Problemstellung besteht darin, einen Algorithmus zur Berechnung der Anzahl der Paare innerhalb der Kaninchenpopulation im n-ten Monat zu bestimmen.

Im ersten und zweiten Monat gibt es jeweils ein Paar. Ab dem dritten Monat erhöht sich die Anzahl der Paare des Vormonats exakt um die Anzahl des Vorvormonats, da letzteres der Anzahl der geschlechtsreifen

```
int Summe(int n)
{
  if (n == 1) return 1;
  return n + Summe(n-1);
}
```

Bild 1: JAVA-Quelltext der rekursiven Summe-Methode.

```
int Fibonacci(int n)
{
  if (n<=2) return 1;
  return Fibonacci(n-2) + Fibonacci(n-1);
}
```

Bild 2 (oben): JAVA-Quelltext der rekursiven Fibonacci-Methode.

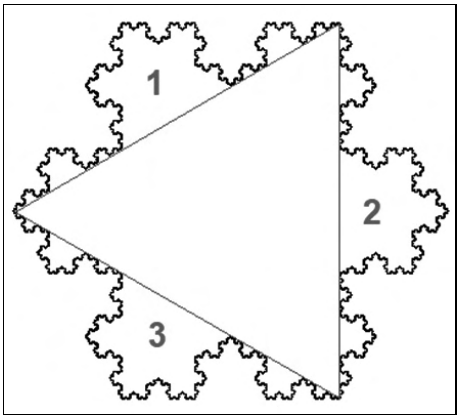

Bild 3 (links): Kochsche Schneeflocke.

Kaninchen entspricht. So entsteht die folgende nach Fibonacci benannte Zahlenfolge: 1, 1, 2, 3, 5, 8, 13, 21, 34, … Eine rekursive Lösungsmethode ergibt sich unmittelbar aus der Problemstellung (siehe Bild 2), weshalb sich die Fibonacci-Zahlenfolge als Einstieg eignet.

Kochsche Schneeflocke

Die Kochsche Schneeflocke (siehe Bild 3) ist ein Fraktal, d.h. ein grafisches Objekt mit einem hohen Grad an Selbstähnlichkeit. Sie besteht aus drei identischen Teilen, den Koch-Kurven. Das Konstruktionsprinzip stammt vom schwedischen Mathematiker Helge von Koch (1870–1924).

Die Koch-Kurven (siehe Bild 4) entstehen nach und nach durch Drittelung einer Strecke der Länge l und Zeichnen eines gleichseitigen Dreiecks der gedrittelten Streckenlänge.

Die Problemstellung, eine Koch-Kurve bzw. Kochsche Schneeflocke beliebiger Stufe n zu zeichnen, führt in natürlicher Weise auf einen rekursiven Algorithmus, da das Zeichenprinzip rekursiv ist. Eine Kochkurve der Stufe 1 besteht aus vier Teilstücken, nämlich Koch-Kurven der Stufe 0, d.h. vier Strecken der Länge 90/3 = 30. Die Koch-Kurve der Stufe 2 besteht wiederum aus vier Teilstücken, nämlich Kochkurven der Stufe 1 bzw. allgemein: Eine Kochkurve der Stufe n besteht aus vier Teilstücken, nämlich Kochkurven der Stufe n–1. Dabei beträgt die Winkelweite der Richtungsänderungen zwischen erstem und zweitem Teilstück 45°, zwischen zweitem und drittem Teilstück 90° und zwischen drittem und viertem Teilstück 45°. Unter Zuhilfenahme der Turtle-Bibliothek ergibt sich die *zeichneKochkurve-Methode* (siehe Bild 5).

```
void zeichneKochkurve(int n, double l)
{
  if (n == 0) Forward(l);
  else
  {
    zeichneKochkurve(n-1, l/3);
    Left(45);
    zeichneKochkurve(n-1, l/3);
    Right(90);
    zeichneKochkurve(n-1, l/3);
    Left(45);
    zeichneKochkurve(n-1, l/3);
  }
}
```

Bild 5 (oben): JAVA-Quelltext der zeichneKochkurve-Methode.

Bild 6 (unten): JAVA-Quelltext der zeichneKochflocke-Methode.

```
void zeichneKochflocke(int n, double l)
{
  zeichneKochkurve(n, l);
  Right(120);
  zeichneKochkurve(n, l);
  Right(120);
  zeichneKochkurve(n, l);
}
```

Eine Kochsche Schneeflocke besteht aus drei Koch-Kurven, wobei die Winkelweite der Richtungsänderung zwischen den Kurven jeweils 120° beträgt. Aus diesem Zusammenhang folgt die *zeichneKochflocke-Methode* (siehe Bild 6).

Die Kochsche Schneeflocke ist ein geeignetes Einstiegsbeispiel, da das Konstruktionsprinzip per Definition rekursiv ist.

Die Türme von Hanoi

In einem Tempel in der indischen Stadt Benares liegen 64 kostbare Scheiben aus Diamant zu einem Turm aufgeschichtet. Jede Scheibe ist ein wenig kleiner als die Scheibe, auf der sie ruht. Ein Priesterorden hat nun die Aufgabe erhalten, den Turm unter Beachtung heiliger Regeln zu einer anderen Stelle im Tempel zu bewegen. Wenn der Turm an einer Stelle abgebaut und an anderer

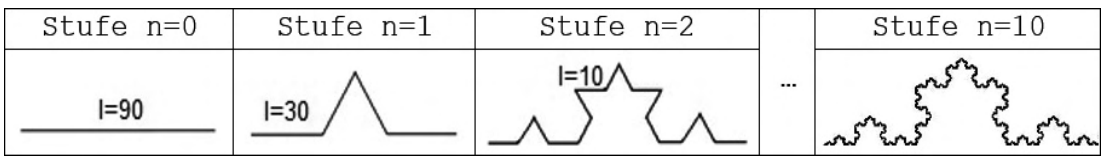

Bild 4: Konstruktionsprinzip der Koch-Kurve.

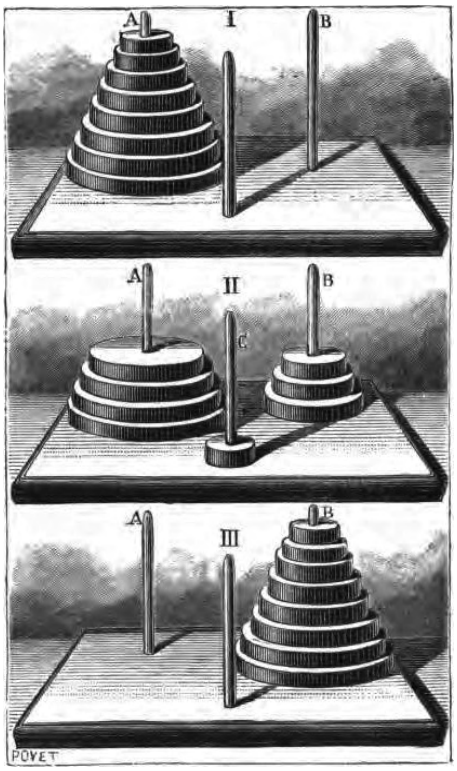

Bild 7: Modell für „Die Türme von Hanoi“ von Édouard Lucas (1893, S. 56).

Stelle wieder ganz aufgebaut wurde, wird der Tempel und mit ihm die ganze Welt zu Staub zerfallen.

Die obige Sage „La Tour de Hanoï“ (ursprünglich: „Der Turm von Benares“) geht auf den französischen Mathematiker Eìdouard Lucas (1842–1891) zurück (vgl. Lucas, 1893, S. 55 ff.) und führt auf die naheliegende Frage: „Wann zerfällt die Welt zu Staub?“ Dazu muss ein optimaler Lösungsalgorithmus bestimmt und dessen Laufzeit analysiert werden. Die formalisierte Problemstellung mit konkretisierten „heiligen Regeln“ ist, einen Turm mit n Scheiben mithilfe von drei Stäben vom ersten auf den dritten Stab zu verschieben, indem immer nur eine Scheibe von einem auf den anderen Stab bewegt wird, sodass niemals eine größere auf einer kleineren Scheibe liegt. Die Schülerinnen und Schüler können die Lösung des Problems handlungsorientiert erarbeiten, wenn ihnen ein Modell zur Verfügung gestellt wird (siehe Bild 7).

Eine Lösung ist, zuerst den Turm der Höhe n–1 rekursiv vom ersten Stab („Startstab“) auf den mittleren Stab („Hilfsstab“) zu verschieben, dann die verbleibende unterste Scheibe vom ersten auf den dritten Stab zu legen und schließlich den Turm der Höhe n–1 rekursiv vom zweiten auf den dritten Stab („Zielstab“) umzuschichten. Die Optimalität der *verschiebe-Methode* (siehe Bild 8) folgt aus der Überlegung, dass jede unterste Scheibe erst dann bewegt werden kann, wenn die darüber liegenden Scheiben verschoben wurden.

Der Aufruf von verschiebe(3, "Stab 1", "Stab 2", "Stab 3") gibt auf der Konsole die nötigen Legeoperationen aus: Lege Scheibe von Stab 1 auf Stab 3; Lege Scheibe von Stab 1 auf Stab 2; Lege Scheibe von Stab 3 auf Stab 2; Lege Scheibe von Stab 1 auf Stab 3; Lege Scheibe von Stab 2 auf Stab 1; Lege Scheibe von Stab 2 auf Stab 3; Lege Scheibe von Stab 1 auf Stab 3. Am Beispiel dieses Aufrufs lässt sich das Rekursionsprinzip veranschaulichen (siehe Bild 9).

Die Aufrufe von verschiebe(1, …) und verschiebe (0, …) sind unter dem Aufruf verschiebe(2, …) subsumiert, damit das verallgemeinerte Rekursionsprinzip erarbeitet werden kann, indem die Schülerinnen und Schüler

```
void verschiebe(int n, String startstab, String hilfsstab, String zielstab)
{
  if (n > 0)
  {
   verschiebe(n-1, startstab, zielstab, hilfsstab);
   System.out.println("Lege Scheibe von " + startstab + " auf " + zielstab);
   verschiebe(n-1, hilfsstab, startstab, zielstab);
  }
}
```

Bild 8 (Mitte): JAVA-Quelltext der rekursiven verschiebe-Methode.

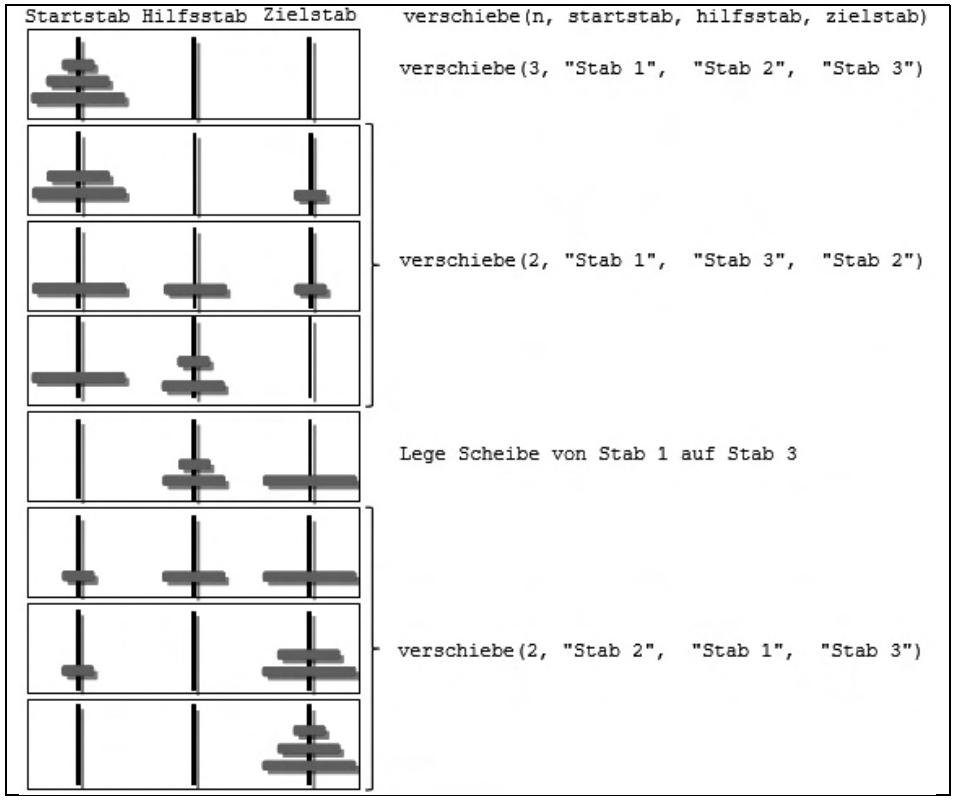

Bild 9 (rechts): Veranschaulichung des Rekursionsprinzips.

```
Summe(4)  = 4 + Summe(3)
          = 4 + 3 + Summe(2)
          = 4 + 3 + 2 + Summe(1)
          = 4 + 3 + 2 + 1
              recurrere
          = 4 + 3 + 3
          = 4 + 6
          = 10
```

Bild 10: Veranschaulichung der Summe-Methode in linearer Notation.

von den Turmhöhen 3 und 2 auf die Turmhöhen n und n–1 abstrahieren können.

Der herausforderndste Schritt bei der Erarbeitung der *verschiebe-Methode* ist die Semantik der Parameter und deren korrekte Verwendung bei den beiden Selbstaufrufen. Der „Hilfsstab"-Parameter der Rekursionsstufe n wird zum „Zielstab"-Parameter der Stufe n–1, um den Turm der Höhe n–1 auf dem „Hilfsstab" der Stufe n zu „parken". Nach Umlegen der verbliebenen untersten Scheibe vom „Startstab" auf den „Zielstab" wird der „Startstab"-Parameter der Rekursionsstufe n zum „Hilfsstab"-Parameter der Stufe n–1, um den auf dem „Hilfsstab" der Stufe n geparkten Turm der Höhe n–1 auf den „Zielstab" der Stufe n zu verschieben.

Die Anzahl der Legeoperationen in Abhängigkeit der Turmhöhe n beträgt 2^n–1 und entwickelt sich demzufolge exponentiell. Damit sind mehr als 18,4 Trillionen Scheibenbewegungen bei einem Turm der Höhe 64 nötig. Unter der Annahme, dass die Priester aus der Sage eine Sekunde pro Scheibenbewegung benötigen, ist die Einstiegsfrage „Wann zerfällt die Welt zu Staub?" mit ca. 585 Milliarden Jahren zu beantworten. Die folgende Überlegung liegt einem formalen Beweis der Formel für die Anzahl der Legeoperationen zugrunde: Pro *verschiebe*-Aufruf der Rekursionsstufe n fallen neben einer Legeoperation zwei weitere Aufrufe der Stufe n–1 an. Der Ansatz Legeoperationen(n) = 2·Legeoperationen(n–1) + 1 kann für einen Induktionsbeweis verwendet werden oder mittels sukzessivem Einsetzen, geometrischer Summe und Teleskopsumme zur obigen Formel vereinfacht werden. In der Regel ist ein Beweis nicht notwendig, da die Schülerinnen und Schüler an den ersten Zahlen der Folge (1, 3, 7, 15, 31, 63) unmittelbar erkennen, dass sie um eins erniedrigten Zweierpotenzen entsprechen.

Die Problemstellung „Die Türme von Hanoi" ist ein geeignetes Einstiegsbeispiel, da die Schülerinnen und Schüler die Rekursionslösung selbstständig am Modell erarbeiten können, und eine Implementierung mithilfe einer Veranschaulichung der Rekursionsaufrufe möglich ist.

Veranschaulichungen

Die obigen Problemstellungen zeigen: Veranschaulichungen, die der Erarbeitung eines rekursiven Algorithmus dienen, hängen vom individuellen Problem ab. Allerdings gibt es gleichartige Veranschaulichungsmöglichkeiten für die Abläufe rekursiver Algorithmen, die sich im Wesentlichen durch die Anzahl der Selbstaufrufe unterscheiden. Methoden mit zahlenwertigen Rückgabewerten bieten sich aufgrund ihrer Quantifizierbarkeit zur Veranschaulichung besonders an und zeigen, wie rekursive Methoden vom Rechner ausgeführt werden.

Lineare Notation

Für Methoden mit einem einzigen Rekursionsaufruf im Rückgabeausdruck wie bei der *Summe-Methode* ist eine linear strukturierte Notation ausreichend (siehe Bild 10).

Baumnotation

Für Methoden mit zwei Rekursionsaufrufen im Rückgabeausdruck wie bei der *Fibonacci-Methode* bietet sich eine Baumnotation an (siehe Bild 11).

Anhand des Baums (siehe Bild 11) erkennen die Schülerinnen und Schüler, in welcher Reihenfolge die Methodenaufrufe stattfinden und wie durch Zurücklaufen („recurrere") die Ergebnisbildung ausgehend von den Basisfällen der Baumblätter abläuft. Außerdem wird deutlich, dass Teilergebnisse wie Fibonacci(3) in obigem Beispiel redundant berechnet werden – ein häufiger Nachteil gegenüber äquivalenten iterativen Methoden.

Aufrufstapel

Bei jeder rekursiven Methode kann das Setzen von Haltepunkten im Quelltexteditor der Entwicklungsumgebung und eine Beobachtung des Aufrufstapels, in dem die Ergebnisse der rekursiven Aufrufe zwischengespeichert werden, zur Veranschaulichung verwendet werden. Details wie Rücksprungadressen können aus-

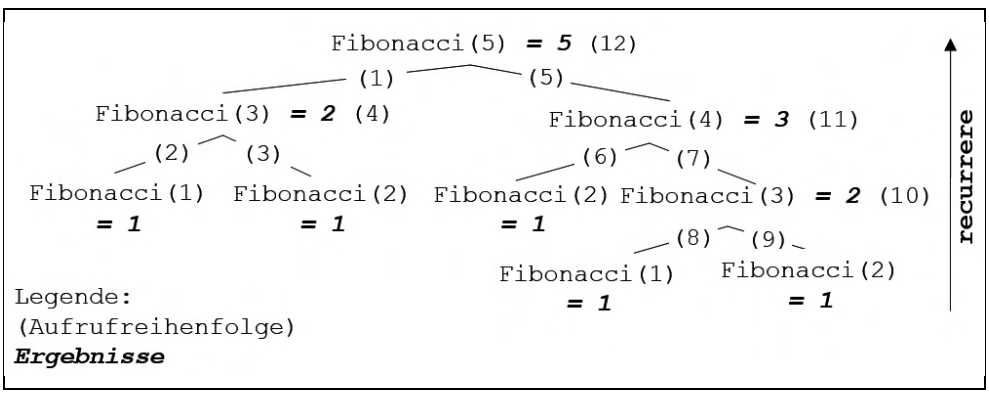

Bild 11: Veranschaulichung des Aufrufs Fibonacci(5) in Baumnotation.

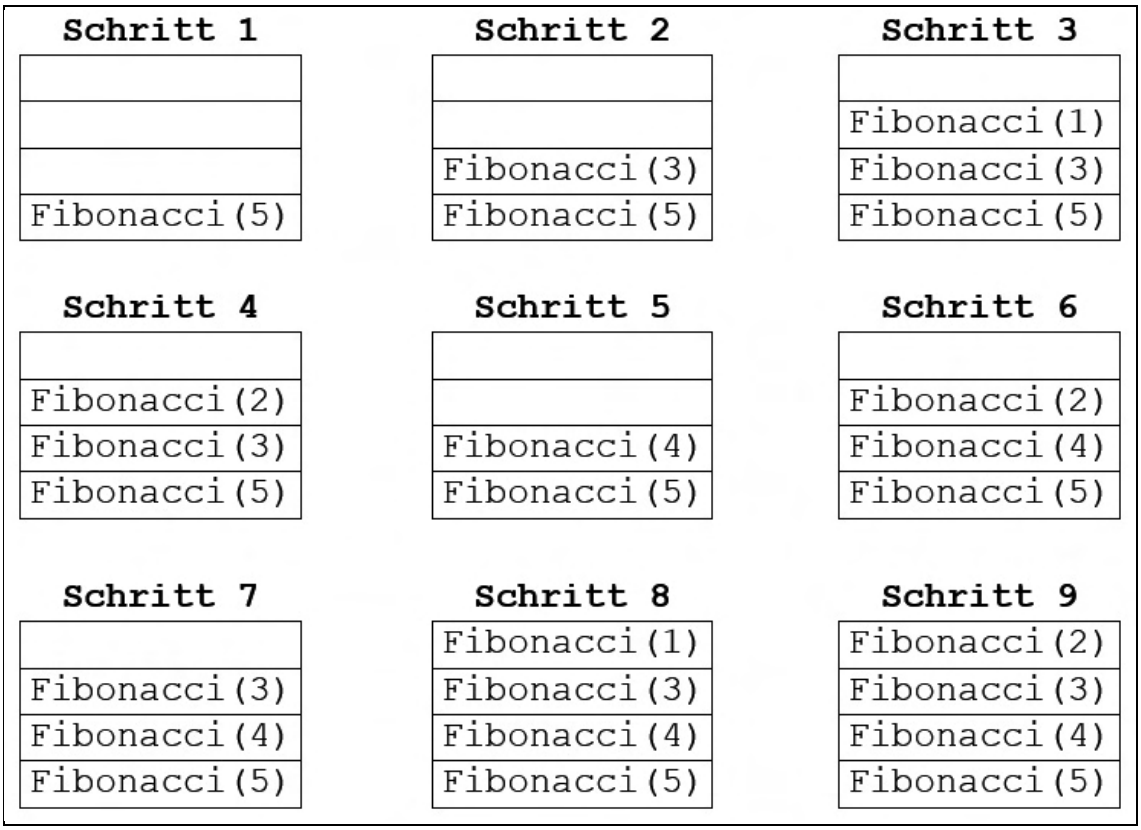

Bild 12: Veranschaulichung des Aufrufs Fibonacci(5) am Aufrufstapel.

gespart werden, um die Reihenfolge der Rekursionsaufrufe in den Vordergrund zu stellen (siehe Bild 12).

Fazit

Rekursive Algorithmen, denen eine Problemstellung zugrunde liegt, die die Schülerinnen und Schüler zur Einsicht führt, dass Rekursion eine probate Problemlösungsstrategie in der Informatik darstellt, sind grundsätzlich für den Einstieg in das Thema *Rekursion* geeignet. Dabei darf die Rekursionslösung weder Zufallsprodukt noch Konsequenz eines künstlichen Szenarios sein, sondern muss sich aus den Rahmenbedingungen der Problemstellung ergeben. Andernfalls verschließen sich den Lernenden Bedeutung und Nutzwert rekursiver Algorithmen. Spätestens bei der Gegenüberstellung zu äquivalenten iterativen Algorithmen würde die Notwendigkeit der Rekursion zwangsläufig infrage gestellt werden.

Es stand in LOG IN …

Ausgewählte Beiträge zum Thema „Rekursion“

Baumann, R.: Grundfragen der Algorithmen-Theorie im Informatikunterricht – Teil 2: Rekursive Funktionen. In: LOG IN, 3. Jg. (1983), Heft 4, S. 42–48.

Baumann, R.: Rekursion – Beispiele aus der Grafik. In: LOG IN, 5. Jg. (1985), Heft 3, S. 30–34.

Baumann, R.: Rekursion – Beispiele aus der Kombinatorik. In: LOG IN, 8. Jg. (1988), Heft 5/6, S. 58–63.

Baumann, R.: Rekursion und Grafik in PROLOG. In: LOG IN, 15. Jg. (1995), Heft 5/6, S. 65–69.

Baumann, R.: Ein selbstreproduzierendes Programm in JAVA. In: LOG IN, 24. Jg. (2004), Nr. 130, S. 20.

Fothe, M.: Rekursion – Ein Thema für den Informatikunterricht. In: LOG IN, 25. Jg. (2005), Nr. 133, S. 46–54.

Fothe, M.; Ludwig, H.; Küspert, K.; Wenzel, M.: Unterrichtsreflexion mit ungewöhnlichen Mitteln – Eine Studie zu Möglichkeiten der externen Unterstützung von Informatiklehrerinnen und -lehrern am Beispiel „Rekursion und Iteration“. In: LOG IN, 26. Jg. (2006), Nr. 141/142, S. 52–63.

Knöß, P.: Rekursive Prozeduren – eine Unterrichtsreihe (Teil 1). In: LOG IN, 5. Jg. (1985), Heft 5/6, S. 83–86.

Knöß, P.: Rekursive Prozeduren – eine Unterrichtsreihe (Teil 2). In: LOG IN, 6. Jg. (1986), Heft 1, S. 35–37.

Künzell, St.; Lehmann, E.; Matzanke, St.: Grafische Darstellung der Baumrekursion. In: LOG IN, 16. Jg. (1996), Heft 4, S. 38–40.

Röhner, G.: Suchbaum-Modellierung – Mit Unterrichtsbeispielen in JAVA. In: LOG IN, 24. Jg. (2004), Nr. 131/132, S. 62–69.

Schwill, A.: Leben in der rekursiven Welt – Selbstreproduzierende Automaten und Programme. In: LOG IN, 24. Jg. (2004), Nr. 130, S. 15–20.

Veranschaulichungen helfen bei der Erarbeitung rekursiver Algorithmen. Ihre Konkretisierung hängt von der jeweiligen Problemstellung ab. So kann das Konstruktionsprinzip der *Kochschen Schneeflocke* durch Vergleich der Koch-Kurven aufeinanderfolgender Stufen verbildlicht werden. Die Lösungsstrategie zu *Die Türme von Hanoi* lässt sich mithilfe eines Modells veranschaulichen.

Für die Visualisierung von Rekursionsabläufen bieten sich drei Klassen an: lineare Notation, Baumnotation und Aufrufstapel.

StD Andreas Koch
Seminar für Ausbildung und Fortbildung
der Lehrkräfte Tübingen (Gymnasium)
Mathildenstraße 32
72072 Tübingen

E-Mail: andreas.koch@seminar-tuebingen.de

Literatur und Internetquellen

GI – Gesellschaft für Informatik (Hrsg.): Bildungsstandards Informatik für die Sekundarstufe II. Erarbeitet vom Arbeitskreis „Bildungsstandards SII“ unter Koordinierung von Gerhard Röhner – Empfehlungen der Gesellschaft für Informatik e. V. vom 29.01.2016. In: LOG IN, 36. Jg. (2016), Nr. 183/184, Beilage.
https://t1p.de/2myc

ISB – Staatsinstitut für Schulqualität und Bildungsforschung München: Lehrplan (Pflicht-/Wahlpflichtfächer) – III Jahrgangsstufen-Lehrplan – Jahrgangsstufen 11/12 – Informatik. München, 2004.
https://t1p.de/beww

KM BW – Ministerium für Kultus, Jugend und Sport Baden-Württemberg: Bildungsplan Informatik Baden-Württemberg. Kursstufe (4-stündig) – Entwurfsfassung Juli 2008 – Änderungen vom 14.07.2010 / 21.10. 2014.
https://t1p.de/szto

Lucas, É.: Récréations mathématiques – vol. III. Paris: Gauthier-Villars et fils, 1893.
https://archive.org/details/rcrationsmathma07lucagoog

MSB NRW – Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen: Kernlehrplan für die Sekundarstufe II Gymnasium/Gesamtschule in Nordrhein-Westfalen – Informatik. Düsseldorf: 2014.
https://t1p.de/q58a

SenBJS – Senatsverwaltung für Bildung, Jugend und Sport Berlin: Rahmenlehrplan für die gymnasiale Oberstufe – Gymnasien, Gesamtschulen mit gymnasialer Oberstufe, Berufliche Gymnasien, Kollegs, Abendgymnasien – Informatik. Berlin: Oktoberdruck, 2006.
https://t1p.de/94yl

Alle Internetquellen wurden zuletzt am 15. August 2019 geprüft und können auch aus dem Service-Bereich des LOG IN Verlags (https://www.login-verlag.de/) heruntergeladen werden.