Freie Universität Berlin
Friedrich-Meinecke-Institut
Studiengang: Lehramtsmaster (Geschichte, Informatik)
Seminartitel: Fachdidaktik Informatik –
Entwicklung, Evaluation und Forschung
Seminarnummer: 19323411
Dozentin: Teresa Busjahn

Was das Verteilen eines Kuchens mit Rekursion zu tun hat

Entwurf eines Lernmaterials

27.07.2017

Inhalt

1 Einleitung

Als ich im Informatikunterricht das erste Mal von Rekursion hörte, wurde erklärt, dass sich bei dieser eine Methode immer wieder von selbst aufruft, bis eine Abbruchbedingung erfüllt ist. Das klang erstmal ein wenig nach einer neuen Art von Schleife. Als mein Lehrer dann begann, etwas von Kisten zu erzählen, die in immer größere Kisten gepackt würden, bis sie dann wieder ausgepackt würden, half das nicht, die Verwirrung im Klassenraum zu beenden. Im Gegenteil: diese Schleife, die in zwei Richtungen verlief, wurde für viele immer geheimnisvoller. Da ich das Fach sehr mochte, versuchte ich dieses Geheimnis zu lüften und kam recht schnell dahinter, dass es sich bei der Rekursion gar nicht um eine Schleife handelt. Andere Schüler*innen[1] in meiner Klasse hatten bis zum Ende der Schulzeit Probleme mit der Rekursion.

Mittlerweile sind einige Jahre vergangen, ich stehe selber kurz davor, als Lehrer zu arbeiten und ich frage mich, wie ich solche Themen wie die Rekursion verständlicher erklären kann. Als ich versuchte, die Rekursion meiner Freundin verständlich zu machen, bemerkte ich schnell, dass ich ein Beispiel mit Lebensweltbezug bräuchte. Ich erklärte es am Verteilen eines Kuchens und die Idee für das – in der vorliegenden Arbeit beschriebene – Lernmaterial war geboren. Um dieses jedoch genauer zu konzipieren und zu erklären, soll zuerst die Fehlvorstellung einer Rekursion als Schleife betrachtet werden, um anschließend darzustellen, welche Lernmaterialien es bereits gibt und welche Schwächen diese in Bezug auf das Vorhaben des Behebens der Fehlvorstellung haben, sodass die Entwicklung eines neuen Lernmaterials begründet ist. Im Rahmen einer Anleitung sollen Aufbau

1 In dieser Arbeit werden Begriffe mithilfe des Gender Gap gegendert, da das generische Maskulinum allzu schnell vergessen lässt, dass Menschen nicht nur männlich, sondern von unterschiedlichem sozialen und/oder biologischen Geschlecht sein können.

und Einsatz des Lernmaterials erklärt werden, damit zum Schluss reflektiert werden kann, weshalb sich das neu entworfene Lernmaterial eignet, um die Fehlvorstellung, eine Rekursion sei eine Schleife, zu entkräften.

2 Von der Fehlvorstellung zum Lernmaterial

2.1 Fehlvorstellung – die Rekursion als Schleife

Wie bereits in der Einleitung dargestellt, läuft die Rekursion Gefahr, als Schleife missverstanden zu werden. Von Juha Sorva wurde diese Fehlvorstellung im Buch »Visual Program Simulation in Introductory Programming Education« mit der Nummer 61 versehen und dem Thema »Rec« für Recursion zugeordnet. In einer Kurzbeschreibung heißt es, dass die Rekursion bei dieser Fehlvorstellung lediglich als ein Konstrukt zur Produktion von Wiederholungen verstanden würde. Dass es sich hierbei um eine Selbstreferenz, also einen Aufruf der Methode durch sich selbst, handelt, wird dagegen ausgeblendet. [1, S. 370]

Michael Weigend widmet sich dem Thema in »Intuitive Modelle der Informatik« leicht umfangreicher. Er schreibt, dass das rekursive Aufrufen der Methode bei dieser Fehlvorstellung einfach als Start der nächsten Runde innerhalb einer Schleife wahrgenommen würde, bei der nun ein neues Argument mitgegeben wird. Solange die Methode endrekursiv ist, also kein Objekt oder nur ein leeres Objekt zurückgegeben wird, wirft diese Vorstellung einer Rekursion als Schleife somit keine Widersprüche auf. Weigend merkt an, dass diese Fehlvorstellung durch die Interpretation der Rekursion als Iteration[2] entsteht. [2, S. 100]

Es wird jedoch hervorgehoben, dass Schüler*innen, die sich mit

2 Bei einer Iteration wird die gleiche Methode mehrfach wiederholt, um ein Problem auf diesem Weg schrittweise zu verkleinern.

iterativen Problemlösungen gründlich auseinandergesetzt hätten, auch die Rekursion besser verstehen würden. Es kann somit von einem positiven Transferprozess ausgegangen werden. Dieser setzt aber voraus, dass das Konzept der Iteration tatsächlich verstanden wurde. Ein Transfer von Wissen über die Rekursion auf die Programmierung iterativer Methoden sei dagegen schwieriger. [2, S. 100]

2.2 Darstellung – Was für Lernmaterialien gibt es bereits

Das große Problem existierender Lernmaterialien ist, dass es nicht gelingt, die Rekursion anschaulich zu verdeutlichen. In verschiedenen Materialien wurde eine hohe Realitätsferne festgestellt. Die Anwendung von Rekursion zur Lösung des Rätselspiels »Türme von Hanoi« stellt bereits den Ansatz mit dem ausgeprägtesten Lebensweltbezug dar. [3, S. 259–72], [4, S. 74–92], [5, S. 166–209] Zudem sind die gewählten Darstellungen sehr abstrakt gehalten [3, S. 166–209], [4, S. 74–92], [6, S. 30] und extrem mathematisch formuliert. [7, S. 438–454] Häufig wird sich zum Beispiel auf die Berechnung der Fibonaccizahlen gestützt, die im Alltag der Schüler*innen weniger Bedeutung haben dürften. Auch bei online gefundenen Materialien setzen sich genau diese Probleme fort. Die Frage muss gestellt werden, wie derart abstrakte Erklärungen eine Fehlvorstellung beheben sollen, die gerade aus der Abstraktheit von Konzepten wie Rezeption und Iteration geboren sind. Gerade dort, wo die Schüler*innen versuchen müssen, die Abstraktion ohne anschauliche Materialien zu durchschauen, kann die Wurzel von Fehlvorstellungen vermutet werden.

Jene Ansätze, die weniger abstrakt sind und als »Beispiele aus dem Alltag« deklariert werden – wie etwa das Lied »Ein Mops kam in die Küche« oder ein sich in sich selbst wiederholendes Fernsehbild – laufen Gefahr, den Eindruck der Rekursion als Schleife zu festigen, obwohl beide Beispiele durch den Aufruf der ›Wiederholung‹ in sich selbst

durchaus rekursiv sind und sich beide leicht im Klassenzimmer nachstellen ließen. Als Einführung in das Thema erscheinen sie als durchaus geeignet. [8, S. 4–5] Ob das Zeichnen von Schneeflocken mithilfe einer Turtle-Grafik tatsächlich motivierend auf heutige Schüler*innen wirkt und als lebensweltnah wahrgenommen wird, kann zumindest hinterfragt werden. [8, S. 23]

2.3 Eine Anleitung

Im Folgenden soll der Aufbau sowie der Einsatz des Lernmaterials erklärt werden. Dabei wird von der materiellen Ebene, in der benannt wird, welche Komponenten das Lernmaterial beinhaltet, auf den methodischen Aufbau übergeleitet werden, um anschließend zu klären, zu welchem Unterrichtszeitpunkt es eingesetzt wird und wie dabei vorgegangen werden kann.

2.3.1 Aufbau des Lernmaterials

Das Material liegt in Form eines einfachen Arbeitsblattes vor, welches durch Zettel mit zusätzlichen Hilfestellungen zur Aufgabe 3) des Arbeitsblattes ergänzt wird. Außerdem existiert digital ein Programm »kuchen.py« dessen Quelltext sich auch in der ersten Aufgabe des Arbeitsblattes wiederfindet.[3] Die vier Aufgaben des Arbeitsblattes bauen aufeinander auf.

Zuerst wird ein recht einfacher Python-Code gegeben, der sich in der Benennung von Methode und Variablen bereits klar an dem Kuchenbeispiel orientiert. Dieser Code enthält eine rekursive Methode `kuchenVerteilen`, welche einen bereits geschnittenen `kuchen` (eine Variable vom Typ Integer), der noch mehr als `1` Kuchentück enthält, in zwei Hälften teilt, welche bei einer ungeraden Stückzahl ebenfalls ungleich groß sein können. Mit jeder Hälfte ruft sich die Methode selbst

3 Siehe »Arbeitsblatt« (S. II), »Hilfszettel« (S. IV) & »Lösungsskizze« (S. V).

auf. Ist die Zahl der Kuchenstücke gleich `1`, wird das Stück gegessen. Um das Verständnis des – durch die Schüler*innen zu untersuchenden – Codes zu erleichtern, ist darunter ein Diagramm abgebildet, welches die Baumstruktur des Ablaufs der Methode darstellt. In die Knoten des Diagramms, die jeweils einen Aufruf der Methode darstellen, soll die aktuelle Zahl der Kuchenstücke eingetragen werden. Sie sollen sich die Arbeitsweise des Programms und somit der Rekursion auf diese Weise selbst erarbeiten.

Im Anschluss wird das Programm in Aufgabe 2) über den Befehl `python kuchen.py` in der Konsole ausgeführt und die Schüler*innen sollen überprüfen, ob die Ausgabe ihren Erwartungen entspricht. Ist dies nicht der Fall, wird zurück auf die Aufgabe 1) verwiesen. Die zweite Aufgabe dient somit einer Selbstüberprüfung und um mögliche Fehler zu finden.

Im Rahmen der dritten Aufgabe soll der Code nun so umgearbeitet werden, dass es den Kuchen bei mehr als 2 Stücken in drei Drittel aufteilt und diese jeweils an einen rekursiven Aufruf weitergegeben werden. Das Programm soll als `kuchen3.py` gespeichert werden. Zur Unterstützung können bei dieser Aufgabe zwei verschiedene Hilfszettel ausgegeben werden, um sie je nach Fähigkeit der Schüler*innen zu differenzieren.[4]

Zum Schluss sollen zwei Ablaufdiagramme – ähnlich dem aus der ersten Aufgabe – für das von den Schüler*innen erarbeitete Programm kuchen3.py gezeichnet werden. Das Erste soll aufzeigen, wie das Programm 6 Kuchentücke aufteilt und das Zweite soll den Ablauf mit 10 Stücken wiedergeben. An dieser Aufgabe soll offengelegt werden, ob die Schüler*innen das Prinzip des Programms und damit die Rekursion verstanden haben.

4 Siehe »Hilfszettel« (S. IV).

2.3.2 Einsatz des Lernmaterials

Optimaler Weise wird das Lernmaterial eingesetzt, nachdem die Schüler*innen einen ersten Kontakt mit dem Thema Rekursion hatten. Hierzu könnten zum Beispiel die weiter oben benannten »Beispiele aus dem Alltag« herangezogen werden. [8, S. 4–5] Im Rahmenlehrplan ist das Thema im Themenfeld »Algorithmisches Problemlösen« im Rahmen des Inhalts »Variablenkonzept und Prozeduren« [9, S. 25] beziehungsweise im Rahmen der Teilkompetenz »Programme entwerfen und realisieren« der Kernkompetenz »Problemlösen – Probleme erfassen und mit Informatiksystemen lösen« unter der Kompetenzstufe H angesiedelt. [9, S. 18]

Das Material kann von den Schüler*innen selbstständig bearbeitet werden. Bei Bedarf können die Hilfszettel ausgegeben werden. Zudem ist eine gemeinsame Bearbeitung der Aufgaben durch zwei Partner*innen denkbar.

Außerdem ist eine enaktive Demonstration[5] des Programms – zum Beispiel zwischen den Aufgaben 2) und 3) oder im Anschluss an die vierte Aufgabe – möglich. So könnte ein echter Kuchen entweder nach dem Code aus `kuchen.py` oder dem aus `kuchen3.py` verteilt werden. Dies könnte das Ziel der dritten Aufgabe verdeutlichen oder zum Schluss der Arbeit mit dem Lernmaterial die beiden Versionen des Programms vergleichbar darstellen. Hierzu kann willkürlich eine Zahl von Kuchenstücken – möglicherweise 6 – als der Kuchen für die erste Version und die selbe Anzahl Stücke für die zweite Version verwendet werden. Die restlichen Kuchenstücke lassen sich an die Schüler*innen verteilen, die bisher leer ausgegangen sind.

5 Eine enaktive Demonstration ist dadurch genkennzeichnet, dass durch Handlung Wissen erschlossen wird.

2.4 Umgang mit der Fehlvorstellung

> *»Löse ein Problem, indem du es auf ein oder mehrere gleichartige, aber ›kleinere‹ Teilprobleme zurückführst, bis das Problem so klein geworden ist, daß die Lösung trivial ist.«*
>
> (Klaus Kusche über das Ziel einer Rekursion)[6]

Das Ziel des Lernmaterials muss es sein, den Schüler*innen zu verdeutlichen, dass es bei der Rekursion nicht um die Wiederholung einer Methode sondern um die Aufgliederung eines Problems in Teilprobleme geht. Dies ist es auch, die Art, wie die Methode `kuchenVerteilen` funktioniert. Der Kuchen wird immer wieder halbiert, bis am Ende bei jeder Methode nur noch ein Stück vorliegt und »die Lösung« mit den Worten von Klaus Kuschke »trivial ist«. [10, S. 1]

Gerade die benannte Anschaulichkeit durch den Lebensweltbezug ist es, die das Material bei gleichzeitiger Reduktion auf den Kern dessen, was Rekursion ist, von dem gesichteten Material abhebt. Es legt wert darauf, nicht einfach den Aufruf der Methode aus sich selbst in den Vordergrund zu stellen, sondern die Zergliederung des Problems, dass die Kuchenstücke verteilt werden sollen. Das es sich hierzu nicht einfach sondern zweifach – je Hälfte – selbst aufruft, widerspricht dem Wirken einer Schleife. In der dritten Aufgabe wird hierzu auch noch vorgeführt, dass sich diese Zergliederung durch weitere Aufrufe beliebig erweitern lässt.

Dies ist natürlich ein Ansatz, der sich auch mit mathematischen Ansätzen beweisen und demonstrieren lässt. Auch auf diese Weise könnte die Fehlvorstellung ausgeräumt werden. [7, S. 438–454] Es ist jedoch hinreichend bekannt, dass abstrakte, mathematische Beweisführungen nicht immer leicht nachvollziehbar sind. Für ein neues Problem kann dies – wie weiter oben bereits benannt[7] – zu

6 [10, S. 1]

7 Siehe »Darstellung – Was für Lernmaterialien gibt es bereits« (S. 3).

Verständnisschwierigkeiten führen, aus denen Resignation oder neue Fehlvorstellungen folgen.

Das vorliegende Lernmaterial versucht dem mit einem Beispiel aus dem tatsächlichen Alltag entgegenzuwirken. Es kann angenommen werden, dass alle Schüler*innen schon einmal vor der Aufgabe standen, einen Kuchen aufzuteilen. Auch wenn dabei wohl selten derart schematisch gehandelt wird, wie es in dem Beispielprogramm des Lernmaterials formalisiert ist, bietet dies einen Ansatz, die Schüler*innen an genau dieser Stelle abzuholen. Es wird somit an einem Beispiel mit Lebensweltbezug aufgezeigt, dass Rekursion keine Schleife ist, sondern der Kern des Prinzips im Selbstaufruf liegt, der dazu dient, komplexe Probleme in Teilprobleme aufzulösen. Das Lernmaterial tut damit – eventuell gestützt durch die erwähnte enaktive Demonstration – genau das, was es tun soll.

Schwierigkeiten können bei der Analyse des Code-Beispiels in der ersten Aufgabe auftreten. Um die Wahrscheinlichkeit von Verständnisproblemen zu minimieren, wurde der Code ausführlich kommentiert und das Diagramm, welches den Ablauf darstellt, als Orientierung gegeben. Dennoch kann der Transfer vom Quellcode auf das Ablaufdiagramm für Schüler*innen, die mit dem Kompetenzerwerb Probleme haben, eine Schwierigkeit darstellen. Ganz ist dies beim Umgang mit Code-Beispielen nie auszuschließen. Im Fall der Fälle, kann unter Umständen eine Zusammenarbeit mit informatikaffinen Partner*innen helfen.

Auch bei der dritten Aufgabe, in der der Programmcode umgearbeitet werden soll, könnte eine solche Zusammenarbeit helfen. Wichtig ist bei der Zusammenarbeit, dass nicht nur ein*e Schüler*in den Code schreibt, während der*die andere daneben sitzt. Es sollte auf jeden Fall ein Austausch gewährleistet werden. Dieser sollte die Voraussetzung für die Zusammenarbeit sein. Zusätzlich versuchen die beiden

Hilfszettel Problemen vorzubeugen. Der eine gibt dazu lediglich die Struktur der Methode vor, während der andere die Zerlegung in drei Drittel aufzeigt.

3 Fazit

Im Rahmen der Arbeit wurde die Fehlvorstellung der Rekursion als Schleife erklärt und verschiedenes Lernmaterial gesichtet. Dabei konnte festgestellt werden, dass durchaus Bedarf für ein Lernmaterial besteht, welches sich der behandelten Fehlvorstellung mithilfe eines Lebensweltbezugs nähert. Ein solches wurde im Folgenden dargestellt, indem dessen Aufbau und Einsatz in einer Anleitung beschrieben wurde.

Anschließend wurde betrachtet, warum das Material für das Beheben der Fehlvorstellung geeignet ist. Hierbei konnte auf die eingangs getätigte Beschreibung der Fehlvorstellung Bezug genommen und die Schwächen bestehender Materialien bedacht werden, um aufzuzeigen, dass das entwickelte Lernmaterial genau die hieraus gewonnenen Kenntnisse aufgreift. In Hinblick auf die Frage im Titel dieser Arbeit kann somit geantwortet werden, dass das Verteilen eines Kuchens zum Lernmaterial für die Erklärung der Grundprinzipien der Rekursion werden kann.

Anhang

Literaturverzeichnis

[1] J. Sorva, *Visual Program Simulation in Introductory Programming Education*. Helsinki: Aalto University – Department of Computer Science and Engineering, 2012.

[2] M. Weigend, *Intuitive Modelle der Informatik*. Potsdam: Universitätsverlag Potsdam, 2007.

[3] G. Harbeck *u. a.*, *Metzler Informatik. Grundband*, 1. Stuttgart: J.B. Metzlersche Verlagsbuchhandlung, 1984.

[4] V. Heidemann und M. Foegen, *Bausteine Informatik. Problem, Algorithmus, Lösung. Lehr- und Übungsbuch. Sprachenunabhängig*. Ferd: Dümmlers Verlag, 1991.

[5] G. Harbeck *u. a.*, *Metzler Informatik. Grundband*, 2. Stuttgart: J.B. Metzlersche Verlagsbuchhandlung, 1990.

[6] P. Hubwieser, M. Spohrer, M. Steinert, und S. Voß, *Informatik 3. Algorithmen, Objektorientierte Programmierung, Zustandsmodellierung*, 1. Stuttgart: Ernst Klett Verlag, 2008.

[7] H.-D. Burkhard, A. Burmeister, L. Engelmann, H.-J. Laabs, G. Paulin, und C. Wagenknecht, *Informatik bis zum Abitur*, 1. Berlin: paetec Gesellschaft für Bildung und Technik mbH, 2004.

[8] C. Kreß, *Das Thema „Rekursion“ im Informatikunterricht*. Frankfurt am Main: Hessischer Bildungsserver, 2004.

[9] Senatsverwaltung für Bildung, Jugend und Familie und Ministeriums für Bildung, Jugend und Sport Land Brandenburg, *Rahmenlehrplan Berlin-Brandenburg. Teil C. Informatik. Jahrgangsstufen 7-10*, 1. Auflage. Berlin, 2015.

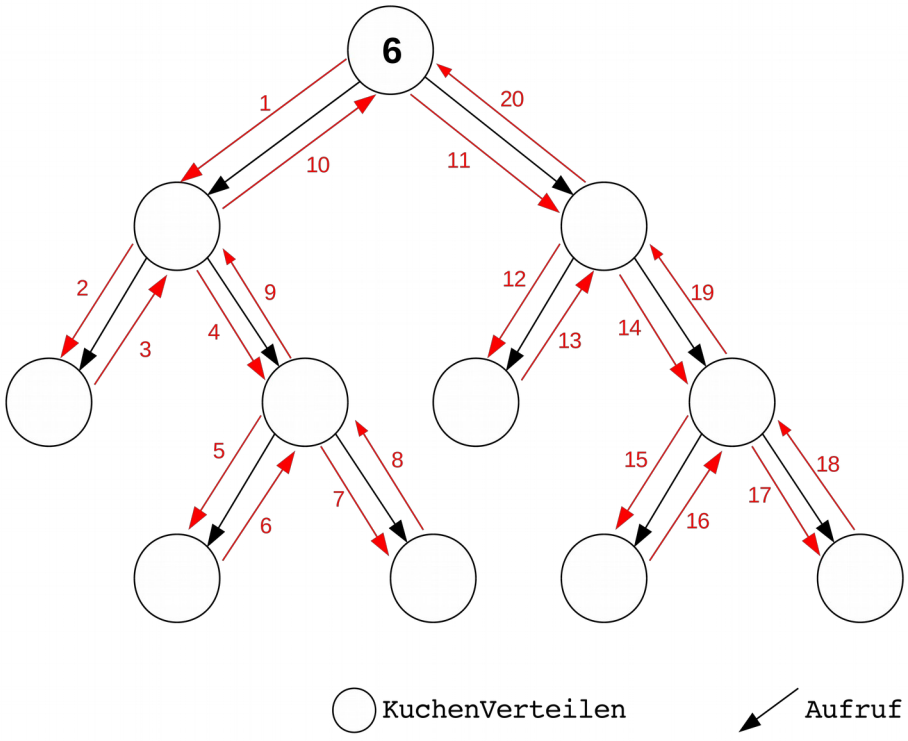
[10] K. Kusche, „Die Rekursion – ein Vorlesungsskript“. 2003.

Arbeitsblatt

Aufgabe 1) Das folgende Python-Programm verteilt die Stücken eines Kuchens. Betrachtet das Programm genau. Unter dem Programmcode findet ihr ein Diagramm, welches den Ablauf des Programms für einen Kuchen mit 6 Stücken darstellt. Tragt in die Kreise, die einen rekursiven Aufruf von `kuchenVerteilen` darstellen, die jeweils aktuelle Anzahl an Kuchenstücken ein. *(Hinweis: Die nummerierten, roten Pfeile geben die Reihenfolge des Verlaufs an.)*

```
def kuchenVerteilen(kuchen):
    print("Es wurden " + str(kuchen) + " Kuchenstücken weitergegeben")
    if kuchen > 1:
        haelfte = kuchen//2
            #Nimmt abgerundete Hälfte (z.B.: 3/2=1)
        kuchen = kuchen-haelfte
            #Die anderen Stücken bleiben am Kuchen (andere Hälfte)
        kuchenVerteilen(haelfte)
            #kuchenVerteilen wird mit der ersten Hälfte rekursiv aufgerufen
        kuchenVerteilen(kuchen)
            #kuchenVerteilen wird mit der anderen Hälfte rekursiv aufgerufen
    else:
        print("Das Stück wurde gegessen")

kuchenVerteilen(6)
            #ein Kuchen bestehend aus 6 Stücken soll verteilt werden
```



Aufgabe 2) Führt das Programm `kuchen.py` mithilfe des Befehls `python kuchen.py` im Terminal des Computers aus. Entspricht die Ausgabe euren Erwartungen? Falls nein überprüft die Ausgabe mit eurem Ergebnis aus Aufgabe 1) und korrigiert mögliche Fehler.

Aufgabe 3) Schreibt das Programm `kuchen.py` so um, dass es den Kuchen bei mehr als 2 Stücken in drei Drittel aufteilt und diese jeweils an einen rekursiven Aufruf weitergibt. *(Speichert es unter dem Namen* `kuchen3.py`*)*

Aufgabe 4) Zeichnet auf, wie das Ablaufdiagramm vom ursprünglichen Programm abgeändert werden müsste, wenn das abgeänderte Programm mit 6 Stücken ausgeführt würde. Wie würde es bei einem Kuchen mit 10 Stücken aussehen?

Hilfszettel

Hilfszettel 1:

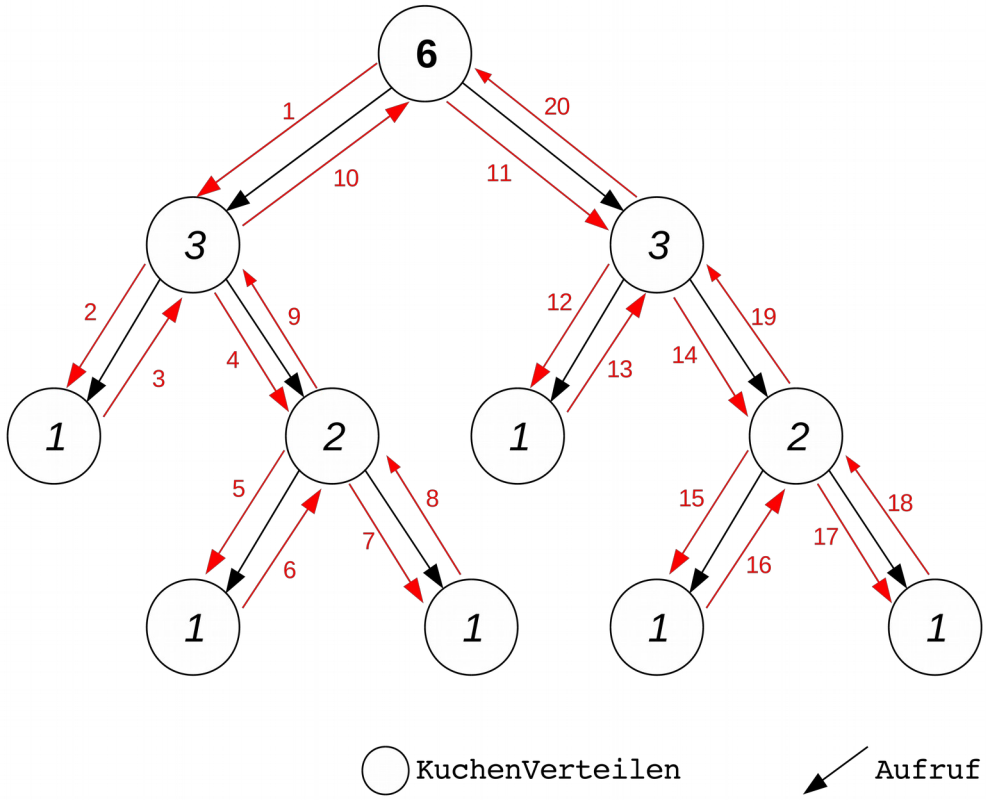
```
def kuchenVerteilen(kuchen):
    print("Es wurden " + str(kuchen) + " Kuchenstücken weitergegeben")
    if kuchen > 2:
        #Code muss noch geschrieben werden
    elif kuchen > 1:
        #Code muss noch geschrieben werden
    else:
        #Code muss noch geschrieben werden
```

Hilfszettel 2:

```
def kuchenVerteilen(kuchen):
    print("Es wurden " + str(kuchen) + " Kuchenstücken weitergegeben")
    if kuchen > 2:
        drittel1 = kuchen//3
            #Nimmt abgerundetes Drittel (z.B.: 7/3=2)
        kuchen = kuchen-drittel1
            #Die anderen Stücken bleiben am Kuchen (die anderen zwei Drittel)
        drittel2 = kuchen//2
            #Von den verbliebenen zwei Dritteln wird eine abgerundete Hälfte
             genommen (z.B.: 3/2=1)
        kuchen = kuchen-drittel2
            #Die anderen Stücken bleiben am Kuchen (andere Hälfte)
        #Was passiert nun mit den drei Dritteln?
    elif kuchen > 1:
        #Hier sieht es aus, wie im ursprünglichen Programm
    else:
        #Code muss noch geschrieben werden
```

Lösungsskizze

Lösungsskizze zu Aufgabe 1:



Lösungsskizze zu Aufgabe 3:

```
def kuchenVerteilen(kuchen):
    print("Es wurden " + str(kuchen) + " Kuchenstücken weitergegeben")
    if kuchen > 2:
        drittel1 = kuchen//3
            #Nimmt abgerundetes Drittel (z.B.: 7/3=2)
        kuchen = kuchen-drittel1
            #Die anderen Stücken bleiben am Kuchen (die anderen zwei Drittel)
        drittel2 = kuchen//2
            #Von den verbliebenen zwei Dritteln wird eine abgerundete Hälfte
             genommen (z.B.: 3/2=1)
        kuchen = kuchen-drittel2
            #Die anderen Stücken bleiben am Kuchen (andere Hälfte)
        kuchenVerteilen(drittel1)
            #kuchenVerteilen wird mit dem ersten Drittel rekursiv aufgerufen
        kuchenVerteilen(drittel2)
            #kuchenVerteilen wird mit dem zweiten Drittel rekursiv aufgerufen
        kuchenVerteilen(kuchen)
            #kuchenVerteilen wird mit dem letzten Drittel rekursiv aufgerufen
    elif kuchen > 1:
        haelfte = kuchen//2
            #Nimmt abgerundete Hälfte (z.B.: 3/2=1)
        kuchen = kuchen-haelfte
            #Die anderen Stücken bleiben am Kuchen (andere Hälfte)
        kuchenVerteilen(haelfte)
            #kuchenVerteilen wird mit der ersten Hälfte rekursiv aufgerufen
        kuchenVerteilen(kuchen)
            #kuchenVerteilen wird mit der anderen Hälfte rekursiv aufgerufen
    else:
        print("Das Stück wurde gegessen")

kuchenVerteilen(6)
            #ein Kuchen bestehend aus 6 Stücken soll verteilt werden
```

Lösungsskizze zu Aufgabe 4: