Seminararbeit zum Thema „Was ist ein Algorithmus?“

Martin Griesser

05. Juni 2013

Inhaltsverzeichnis

1 Vorwort

Ein Algorithmus ist eine schrittweise Lösungsanleitung für eine bestimmte Problemstellung. So könnte zum Beispiel die Anleitung zum Wechseln einer Glühbirne, oder die Zubereitung einer Rezeptur bereits als Algorithmus betrachtet werden. Es gibt allerdings keine generell akzeptierte formale Definition des Begriffes. In dieser Arbeit soll auf den Algorithmusbegriff aus der Sicht der Informatik eingegangen werden.

2 Wortherkunft

Das Wort „Algorithmus“ ist auf den Namen des persisch–arabischen Mathematikers und Universalgelehrten *Abu Ja'far Mohammed ibn Musa al-Khowarizmi*[1] (Mohammed, Vater des Ja'far, Sohn des Mose, geboren in Khowarizm) zurückzuführen. Der in Bagdad lebende Al-Khowarizmi befasste sich mit dem indischen Zahlensystem welches damals bereits die Ziffer 0 verwendete und die Grundlage für unser heutiges Dezimalsystem bildet. Um das Jahr 820 verfasste er sein Werk „Über die indischen Zahlen“. Im 12. Jahrhundert wurde dieses Lehr- und Rechenbuch in Spanien ins Lateinische übersetzt. Die Übersetzung begann mit den Worten: „Dixit Algoritmi:...“(Algoritmi hat gesprochen:...). Die heutige Bezeichnung Algorithmus leitet sich also aus der Ortsbezeichnung des Autors ab. (vgl. Ziegenbalg, *Algorithmen: Von Hammurapi bis Gödel (German Edition)*, S. 19

3 Definition

Ursprünglich wurde das ins Lateinische übernommene Wort *Algorismus* nur für die Kunst des Rechnens mit den indisch-arabischen Zahlen verwendet. Heute bezeichnet man damit eine detaillierte Berechnungsvorschrift zur Lösung eines Problems. Diese besteht aus einer Folge von eindeutigen Elementaranweisungen, welche den Lösungsweg exakt und vollständig beschreiben. Eine Elementaranweisung ist ein Rechenschritt, der nicht mehr weiter unterteilt werden kann. Die Abarbeitung eines Algorithmus ist ein iterativer Prozess. In den diversen Enzyklopädien und Lexika findet sich keine einheitliche, allgemeingültige Definition des Begriffes. Alle Worterklärungen zeigen aber, dass der Algorithmusbegriff sehr eng mit dem Begriff der *Berechenbarkeit* zusammenhängt, weshalb ich im Folgenden näher auf diesen Zusammenhang eingehen möchte.

3.1 Algorithmus und Berechenbarkeit

Intuitiv könnte man berechenbar erklären als „durch einen Kalkül ermittelbar“ oder „maschinell ermittelbar“. Einige bedeutende Mathematiker und Logiker haben sich damit befasst, die Begriffe *Algorithmus* und *Berechenbarkeit* exakter zu definieren, um mathematische Aussagen darüber treffen zu können. Beispiele für die entstandenen Rechenmodelle sind Alan Turings *Turingmaschine*, Alonzo Churchs *Lambda Kalkül*, oder das Konzept der *Rekursiven Funktionen*. Alle diese Modelle stellen formale Fassungen des Begriffes der Berechenbarkeit dar. (vgl ebd., S. 237)

[1] verschiedene Schreibweisen möglich

3.2 Die Turingmaschine und Turing-Vollständigkeit

Die von Alan Turing entwickelte Turingmaschine ist ein Rechenmodell der Theoretischen Informatik, mit dem die Arbeitsweise eines Computers auf mathematisch gut zu analysierende Weise modelliert werden kann. Eine Turingmaschine kann als eine Repräsentation eines Algorithmus bzw. eines Programmes angesehen werden. Jede Funktion, die durch eine Turingmaschine berechnet werden kann heißt *berechenbar*, oder *algorithmisierbar*. Berechnungsmodelle, welche die Eigenschaften einer Turingmaschine nachbilden können, werden als *turingvollständig* bezeichnet. Heutige Computer und Programmiersprachen sind turingvollständig. Sie können also, abgesehen von der Begrenztheit des Speichers, alles berechnen, was mittels einer Turingmaschine berechnet werden kann. (vgl Wikipedia, *Turingmaschine — Wikipedia, Die freie Enzyklopädie*)

3.3 Church-Turing-These

„Die Klasse der turing-berechenbaren Funktionen stimmt mit der Klasse der intuitiv berechenbaren Funktionen überein." (Hoffmann, *Theoretische Informatik*, S. 296)

Diese These ist nicht beweisbar, da da die Aussage „*intuitiv berechenbare Funktion*" nicht exakt formalisiert werden kann, wird aber allgemein als wahr anerkannt. Sollte die Aussage wahr sein, folgt daraus, dass es kein Rechenmodell geben kann, welches mächtiger ist als die bisherigen Modelle. Ein Computer ist ein solches Modell, daher kann auf ihm theoretisch jeder Algorithmus ausgeführt werden (wenn genügend Speicherplatz vorhanden ist). Folglich ist es auch nicht möglich, einen Rechner zu bauen, der mehr berechnen kann, als mit heutigen, turingvollständigen Systemen bereits realisiert wurde. Eine Widerlegung der These wäre die Konstruktion eines Modells, welches Berechnungen anstellen kann, die mit einer Turingmaschine nicht möglich sind. (vgl. Wikipedia, *Church-Turing-These — Wikipedia, Die freie Enzyklopädie*)

3.4 Erweiterte Churchshe These

„Alle bisher bekannten formalen Fassungen des Begriffes der Berechenbarkeit sind gleichwertig. Deshalb kann jede dieser Fassungen als eine angemessene Präzisierung der vorher intuitiv verwendeten Begriffe der Berechenbarkeit, bzw. des Algorithmus angesehen werden"(Ziegenbalg, *Algorithmen: Von Hammurapi bis Gödel (German Edition)*, S. 238)

Aufgrund dieser Aussage kann über die Turingmaschine folgende formale Definition des Begriffes Algorithmus formuliert werden:

„Eine Berechnungsvorschrift zur Lösung eines Problems heißt genau dann *Algorithmus*, wenn eine zu dieser Berechnungsvorschrift äquivalente *Turingmaschine* existiert, die für jede Eingabe, die eine Lösung besitzt, stoppt." (Wikipedia, *Turingmaschine — Wikipedia, Die freie Enzyklopädie*)

3.5 Algorithmen und Programme

In der Informatik werden Algorithmen durch Programme in meist turingvollständigen Programmiersprachen umgesetzt. Joachim Ziegenbalg definiert den Begriff *Programm* wie folgt:

„Ein Programm ist ein Algorithmus, der in einer Sprache formuliert ist, welche die Abarbeitung durch einen Computer ermöglicht." (Ziegenbalg, *Algorithmen: Von Hammurapi bis Gödel (German Edition)*, S. 25)

Somit kann jedes Computerprogramm als Algorithmus betrachtet werden. Darüber gibt es allerdings unterschiedliche Auffassungen, da häufig verlangt wird, dass Algorithmen terminieren müssen, also nach endlich vielen Einzelschritten abbrechen. Dies ist jedoch zum Beispiel bei Betriebssystemprogrammen nicht der Fall.

4 Merkmale anhand eines Beispiels

Abbildung 1 Implementation of Selectionsort

```
procedure SORT(A : List of sortable elements)
    n := length(A)
    left := 0
    repeat
        min := left
        for all i from left + 1 to n do
            if A[i] < A[min] then
                min = i
            end if
        end for
        Swap(A[min], A[left]
        left := left + 1
    while left < n
end procedure
```

4.1 Selectionsort

Selectionsort ist ein einfacher Sortieralgorithmus, der dazu dient, eine Liste von Daten auf- oder absteigend zu sortieren. Obwohl es heute wesentlich effizientere Sortierverfahren gibt, möchte ich diesen Algorithmus aufgrund seiner Einfachheit zur Veranschaulichung verwenden. Abbildung 1 zeigt eine Darstellung in Pseudocode Schreibweise.

4.2 Korrektheit

Um einen realen Nutzen zu erfüllen muss ein Algorithmus zuallererst korrekt sein. Dass dies beim Selectionsort der Fall ist kann mathematisch nachgewiesen werden.

4.3 Endlichkeit

Offensichtlich kann der Beispielalgorithmus exakt durch einen endlich langen Text beschrieben werden. Man spricht hier von *Statischer Endlichkeit*. Wenn ein Algorithmus zu jedem Zeitpunkt der Ausführung nur begrenzt viel Speicherplatz benötigt, spricht man von *Dynamischer Endlichkeit*. Auch dies ist hier gegeben, da der Algorithmus den Speicherplatz für die Eingabeliste und 4 Laufzeitvariablen benötigt

4.4 Determiniertheit

Ein Algorithmus ist determiniert, wenn er bei gleichen Vorbedingungen und Eingaben immer das gleiche Ergebnis liefert.

4.5 Determinismus

Ist zu jedem Ausführungszeitpunkt die als nächstes auszuführende Elementaranweisung eindeutig bestimmt, ist der Algorithmus *deterministisch*. Auch dies trifft im Beispielprogramm zu, da jeder einzelne Schritt eindeutig festgelegt ist. Manche Algorithmen verwenden allerdings Zufallszahlen während der Ausführung. Beim Quicksort-Algorithmus beispielsweise wird das Eingabearray vor der Verarbeitung in zufälliger Reihenfolge durchmischt. Daher ist die Verarbeitungsreihenfolge nicht immer gleich, auch wenn die gleichen Eingabedaten zugrunde liegen. Da dies jedoch nichts an der Korrektheit des Ergebnisses ändert, ist der Quicksort-Algorithmus *determiniert*, aber nicht *deterministisch*.

4.6 Effizienz

Wie effizient ein bestimmter Algorithmus ist, wird in der theoretischen Informatik im Rahmen der Komplexitätstheorie untersucht. Man bezieht sich auf die quantifizierbaren Fragen der Laufzeit- und Speicherkomplexität (vgl. Ziegenbalg, *Algorithmen: Von Hammurapi bis Gödel (German Edition)*, S. 167).

Unter *Laufzeitkomplexität* versteht man die Anzahl an auszuführenden Elementaranweisungen, die für die Abarbeitung notwendig ist. *Speicherkomplexität* misst die Größe des benötigten Speichers. Diese Zahlen sind je nach Algorithmus mehr oder weniger abhängig von der Größe der Eingabe und können mathematisch berechnet werden.

Literatur

Hoffmann, Dirk. *Theoretische Informatik*. Hanser Fachbuchverlag, 2011. ISBN: 3446426396.

Wikipedia. *Church-Turing-These — Wikipedia, Die freie Enzyklopädie*. [Online; Stand 4. Juni 2013]. 2013. URL: `http://de.wikipedia.org/w/index.php?title=Church-Turing-These&oldid=116709005`.

— *Turingmaschine — Wikipedia, Die freie Enzyklopädie*. [Online; Stand 4. Juni 2013]. 2013. URL: `http://de.wikipedia.org/w/index.php?title=Turingmaschine&oldid=118990130`.

Ziegenbalg, Jochen. *Algorithmen: Von Hammurapi bis Gödel (German Edition)*. Spektrum Akademischer Verlag, 1996. ISBN: 3827401143.