String-Matching Algorithmen

Sebastian Kreinecker

Institut für Algebra

Februar, 2018

JOHANNES KEPLER UNIVERSITY LINZ

Begriffserklärung

- Ein *String* ist eine Zeichenkette (z.Bsp.: $'fadfnja'$ oder auch $'ruena\ afeasdf'$).
- 'matching' heißt übereinstimmen.
- Ein *Algorithmus* ist ein Verfahren.

Problem

Wir wollen Wege finden, sodass man am Computer automatisch ein Teilwort (*Substring*) eines Wortes (*String*) erkennen kann und die Positionen ausgibt, an denen das Teilwort auftritt.

Problem

Wir wollen Wege finden, sodass man am Computer automatisch ein Teilwort (*Substring*) eines Wortes (*String*) erkennen kann und die Positionen ausgibt, an denen das Teilwort auftritt.
Beispiel:

Wort: **Blumentopferde**.
Suchen: **pferd**.

Problem

Wir wollen Wege finden, sodass man am Computer automatisch ein Teilwort (*Substring*) eines Wortes (*String*) erkennen kann und die Positionen ausgibt, an denen das Teilwort auftritt.
Beispiel:

Wort: **Blumentopferde**.
Suchen: **pferd**.
Unser *Algorithmus* soll uns als Antwort 9 liefern.

Beispiel aus der Praxis

Activities Document Viewer Wed 09:24

7 of 565 Cox, Little, O'Shea - Ideals, varieties and algorithms.pdf 149.59%

We wrote this book to introduce undergraduates to some interesting ideas in algebraic geometry and commutative algebra. Until recently, these topics involved a lot of abstract mathematics and were only taught in graduate school. But in the 1960s, Buchberger and Hironaka discovered new algorithms for manipulating systems of polynomial equations. Fueled by the development of computers fast enough to run these algorithms, the last two decades have seen a minor revolution in commutative algebra. The ability to compute efficiently with polynomial equations has made it possible to investigate complicated examples that would be impossible to do by hand, and has changed the practice of much research in algebraic geometry. This has also enhanced the importance of the subject for computer scientists and engineers, who have begun to use these techniques in a whole range of problems.

It is our belief that the growing importance of these computational techniques warrants their introduction into the undergraduate (and graduate) mathematics curriculum. Many undergraduates enjoy the concrete, almost nineteenth-century, flavor that a computational emphasis brings to the subject. At the same time, one can do some substantial mathematics, including the Hilbert Basis Theorem, Elimination Theory, and the Nullstellensatz.

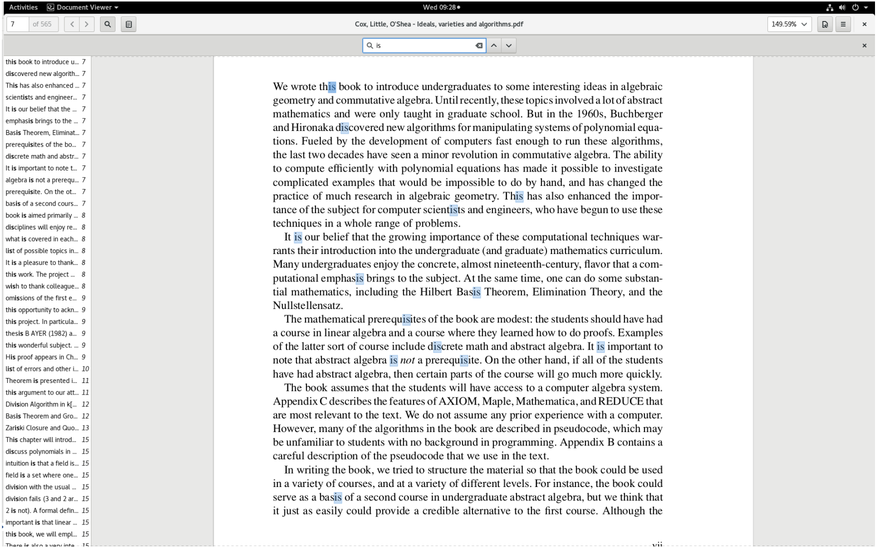
The mathematical prerequisites of the book are modest: the students should have had a course in linear algebra and a course where they learned how to do proofs. Examples of the latter sort of course include discrete math and abstract algebra. It is important to note that abstract algebra is *not* a prerequisite. On the other hand, if all of the students have had abstract algebra, then certain parts of the course will go much more quickly.

The book assumes that the students will have access to a computer algebra system. Appendix C describes the features of AXIOM, Maple, Mathematica, and REDUCE that are most relevant to the text. We do not assume any prior experience with a computer. However, many of the algorithms in the book are described in pseudocode, which may be unfamiliar to students with no background in programming. Appendix B contains a careful description of the pseudocode that we use in the text.

In writing the book, we tried to structure the material so that the book could be used in a variety of courses, and at a variety of different levels. For instance, the book could serve as a basis of a second course in undergraduate abstract algebra, but we think that it just as easily could provide a credible alternative to the first course. Although the

vii

Beispiel aus der Praxis



We wrote this book to introduce undergraduates to some interesting ideas in algebraic geometry and commutative algebra. Until recently, these topics involved a lot of abstract mathematics and were only taught in graduate school. But in the 1960s, Buchberger and Hironaka discovered new algorithms for manipulating systems of polynomial equations. Fueled by the development of computers fast enough to run these algorithms, the last two decades have seen a minor revolution in commutative algebra. The ability to compute efficiently with polynomial equations has made it possible to investigate complicated examples that would be impossible to do by hand, and has changed the practice of much research in algebraic geometry. This has also enhanced the importance of the subject for computer scientists and engineers, who have begun to use these techniques in a whole range of problems.

It is our belief that the growing importance of these computational techniques warrants their introduction into the undergraduate (and graduate) mathematics curriculum. Many undergraduates enjoy the concrete, almost nineteenth-century, flavor that a computational emphasis brings to the subject. At the same time, one can do some substantial mathematics, including the Hilbert Basis Theorem, Elimination Theory, and the Nullstellensatz.

The mathematical prerequisites of the book are modest: the students should have had a course in linear algebra and a course where they learned how to do proofs. Examples of the latter sort of course include discrete math and abstract algebra. It is important to note that abstract algebra is *not* a prerequisite. On the other hand, if all of the students have had abstract algebra, then certain parts of the course will go much more quickly.

The book assumes that the students will have access to a computer algebra system. Appendix C describes the features of AXIOM, Maple, Mathematica, and REDUCE that are most relevant to the text. We do not assume any prior experience with a computer. However, many of the algorithms in the book are described in pseudocode, which may be unfamiliar to students with no background in programming. Appendix B contains a careful description of the pseudocode that we use in the text.

In writing the book, we tried to structure the material so that the book could be used in a variety of courses, and at a variety of different levels. For instance, the book could serve as a basis of a second course in undergraduate abstract algebra, but we think that it just as easily could provide a credible alternative to the first course. Although the

Programm

- Wie kann man einen Algorithmus anschreiben?
- Wie können wir ein solches Verfahren (Algorithmus) finden?
- Wie können wir dieses Verfahren in einem Computer benutzen?
- Gibt es mehrere Verfahren für ein Problem? Sind diese schneller?