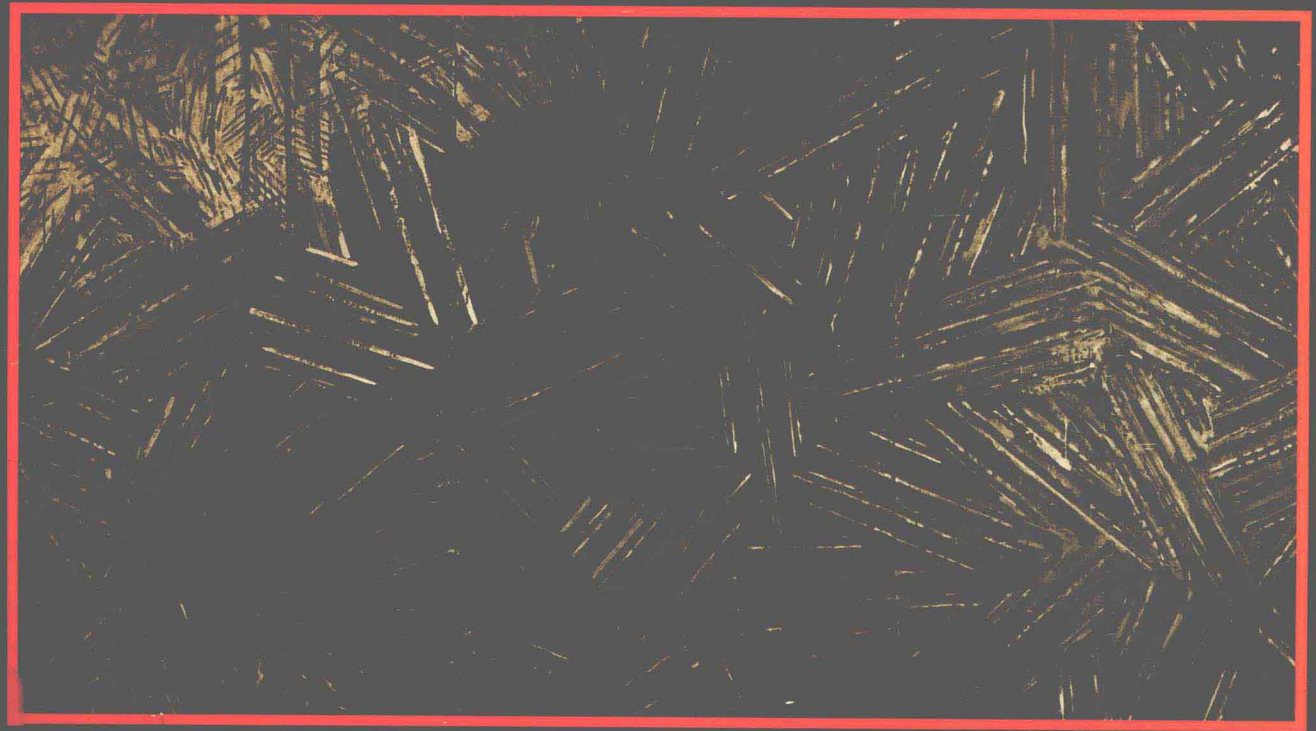
EXPLORING DISCRETE MATHEMATICS WITH MAPLE

Kenneth H. Rosen



EXPLORING DISCRETE MATHEMATICS

With Maple

Kenneth H. Rosen
AT&T Bell Laboratories

John S. Devitt
Waterloo Maple, Inc.

Troy Vasiga
University of Waterloo

James McCarron
Waterloo Maple, Inc.

Eithne Murray
INRIA, Rocquencourt

Ed Roskos
AT&T Bell Laboratories

Boston, Massachusetts Burr Ridge, Illinois Dubuque, Iowa
Madison, Wisconsin New York, New York San Francisco, California St. Louis, Missouri

McGraw-Hill

A Division of The McGraw-Hill Companies

Exploring Discrete Mathematics with Maple

3 4 5 6 7 8 9 0 FGR FGR 9 0 2 1 0 9 8 7

ISBN 0-07-054128-0

The editor was Maggie Rogers.
The production supervisor was Louis Swaim.
Fairfield Graphics was printer and binder.

Preface

This book is a supplement to Ken Rosen's text *Discrete Mathematics and its Applications, Third Edition*, published by McGraw-Hill. It is unique as an ancillary to a discrete mathematics text in that its entire focus is on the computational aspects of the subject. This focus has allowed us to cover extensively and comprehensively how computations in the different areas of discrete mathematics can be performed, as well as how results of these computations can be used in explorations. This book provides a new perspective and a set of tools for exploring concepts in discrete mathematics, complementing the traditional aspects of an introductory course. We hope the users of this book will enjoy working with it as much as the authors have enjoyed putting this book together.

This book was written by a team of people, including Stan Devitt, one of the principle authors of the Maple system and Eithne Murray who has developed code for certain Maple packages. Two other authors, Troy Vasiga, and James McCarron have mastered discrete mathematics and Maple through their studies at the University of Waterloo, a key center of discrete mathematics research and the birthplace of Waterloo Maple Inc.

To effectively use this book, a student should be taking, or have taken, a course in discrete mathematics. For maximum effectiveness, the text used should be Ken Rosen's *Discrete Mathematics and its Applications*, although this volume will be useful even if this is not the case. We assume that the student has access to Maple, Release 3 or later. We have included material based on Maple shareware and on Release 4 with explicit indication of where this is done. (Where to obtain Maple shareware is described in the Introduction.) We do not assume that the student has previously used Maple. In fact, working through this book can teach students Maple while they are learning discrete mathematics. Of course, the level of sophistication of students with respect to programming will determine their ability to write their own Maple routines. We make peripheral use of calculus in this book. Although all places where calculus is used can be omitted, students who have studied calculus will find this material of interest.

This volume contains a great deal of Maple code, much based on existing Maple functions. But substantial extensions to Maple can be found throughout the book; new Maple routines have been added in key places, extending the capabilities of what is currently part of Maple. An excellent example is new Maple code for displaying trees, providing functionality not currently part of the network package of Maple. All the Maple code in this book is available over the Internet; see the Introduction for details.

This volume contains an Introduction and ten chapters. The Introduction describes the philosophy and contents of the chapters and provides an introduction to the use of Maple, both for computation and for programming. This chapter is especially important to students who have not used Maple before. (More material on programming with Maple is found throughout the text, especially in Chapters 1 and 2.) Chapters 1 to 10 correspond to the respective chapters of *Discrete Mathematics and its Applications*. Each chapter contains a discussion of how to use Maple to carry out computation on the subject of that chapter. Each chapter also contains a discussion of the of the Computations and Explorations found at the end of the corresponding chapter of *Discrete Mathematics and its Applications*, along with a set of exercises and projects designed for further work.

Users of this book are encouraged to provide feedback, either via the postal service or the Internet. We expect that students and faculty members using this book will develop material that they want to share with others. Consult the Introduction for details about how to download Maple software associated with this book and for information about how to upload your own Maple code and worksheets.

Acknowledgments

Thanks go to the staff of the College Division of McGraw-Hill for providing us with the flexibility and support to put together something new and innovative. In particular, thanks go to Jack Shira, Senior Sponsoring Editor and Maggie Rogers, Senior Associate Editor, for their strong interest, enthusiasm, and frequent inquiries into the status of this volume, and to Denise Schanck, Publisher, for her overall support. Thanks also goes to the production department of McGraw-Hill for their able work.

We would also like to express thanks to the staff of Waterloo Maple Inc. for their support of this project. In particular, we would like to thank Benton Leong and Ha Quang Le for their suggestions. Furthermore, we offer our appreciation to Charlie Colbourn of the University of Waterloo for helping bring this working team together as well as for his contributions as on of the authors of Maple's networks package which is heavily used in parts of this book.

As always, one of the authors, Ken Rosen, would like to thank his management at AT&T Bell Laboratories, including Steve Nurenberg, Ashok Kuthyar, Hrair Aldermishian, and Jim Day, for providing the environment and the resources that have made this book possible. Another author, Troy Vasiga, would like to thank his wife for her encouragement and support during the preparation of this book.

Contents

Contents

Introduction

An introduction to discrete mathematics is usually taught in a traditional way. Concepts, applications, and problem solving techniques are presented with worked examples provided to illustrate key points. The questions included in the textbook exercise sets to reinforce these key points are designed to be solved without a large amount of computation. But today, with modern mathematical computation software, such as Maple, complicated computations can be carried out easily and quickly. Having such computational tools provides a new dimension to a course in discrete mathematics course, namely an *enquiring* and *experimental* approach to learning. This book is designed to bridge the traditional approach to learning discrete mathematics and this new enquiring and experimental approach.

Using computational software, students can experiment directly with many objects important in discrete mathematics. These include sets, large integers, combinatorial objects, graphs, and trees. Furthermore, by using interactive computational software to do this, students can explore these examples more thoroughly thereby fostering a deeper understanding of concepts, applications, and problem solving techniques.

This supplement has two main goals. The first is to help students learn how to carry out computations in discrete mathematics using Maple, a widely used software package for interactive mathematical computation. The second is to guide students through the process of mathematical discovery through the use of computational tools. This exploration is based on the use of Maple.

Structure of This Volume

This supplement begins with a brief introduction to Maple, its capabilities and its use. Our goal is to provide students new to Maple with the requisite background. The material in this introduction explains the philosophy behind working with Maple, how to use Maple to carry out computations, and the basic structure of Maple. Maple is more than just a computational engine; it also is a programming language. Consequently, this introduction continues by explaining the basic constructs for programming with Maple. For those new to Maple, this material is important for understanding Maple procedures and their use.

Besides the introduction, the main body of this book contains ten chapters. Each chapter is based on a chapter of *Discrete Mathematics and its Applications*, third edition, by Kenneth H. Rosen, published by McGraw Hill (henceforth referred to as the *text*). A chapter begins with comprehensive coverage explaining how Maple can be used to explore the topics of the corresponding chapter of the text. This coverage includes a discussion of relevant Maple commands, as well as many new procedures, written expressly for this book. Many worked examples illustrating how to use Maple to explore topics in that chapter are provided. Additionally, a discussion of many of the Computations and Explorations in the corresponding chapter of the text is provided. Often, these exercises ask students to carry out a series of computations to explore a concept or study a problem. Here, we provide guidance, partial solutions, and sometimes complete solutions to these exercises. Finally, each chapter concludes with a set of additional questions for the students to explore. Some of these are straightforward computational exercises, while others are more accurately described as projects requiring substantial additional work, including programming. Consequently, this set of exercises is labeled Exercises/Projects.

The backmatter of this book includes an extensive index. Programming examples appear throughout the book, but even so, some supplementary programs were developed to facilitate the exposition and discussion. These supplementary programs are also listed.

This volume has been designed to help students achieve the main goals of a course in discrete mathematics. These goals, as described in the preface of the text, are the mastery of mathematical reasoning, combinatorial analysis, discrete structures, applications and modeling, and algorithmic thinking. This supplement demonstrates how to use the interactive computational environment of Maple to enhance and accelerate the achievement of these goals.

Interactive Maple

Exploring mathematics with Maple is like exploring a mathematical topic with an expert assistant at your side. As you investigate a topic you should always be asking questions. In many cases the answer to your question lies in an experiment. Maple, your highly trained mathematical assistant, can often carry out these directed experiments quickly and accurately. Often this requires only a few simple directives (commands).

By hand, the magnitude and quantity of work required to investigate

even one reasonable test case may be prohibitive. By delegating the detail to your mathematical assistant your efforts are much more focussed on choosing the right the mathematical questions and on interpreting their results than in a traditional computational environment.

The reasons an interactive system such as Maple supports such rapid investigation include the fact that:

1. The types of objects you are investigating already exist as part of the basic infrastructure provided by the system. This includes sets (ordered and unordered), variables, polynomials, graphs, arbitrarily large integers and rational numbers, and most importantly, support for exact computations.

2. Tools for manipulating those objects already exist, or can be created easily by essentially mimicking the interactive solution to a particular problem.

Current day calculators are appropriate for simple numerical investigations, but they do not allow you to effectively investigate more complicated mathematical structures or to quickly prototype multistep methods for manipulating those complicated structures.

On the other hand, more traditional programming languages, the kind used to build Maple, almost all of your effort goes into just building an appropriate environment (potentially years of work).

An Example

The use of Maple is merely a means to the end of achieving the goals of a course in discrete mathematics. Yet, with any tool, to use it effectively you must have some basic understanding of the tool and its capabilities. In this section we introduce Maple by working through a sample interactive session.

A new Maple session begins by starting the Maple program. It presents you with a blank *Maple worksheet*, much as any word processor would begin by presenting you with a blank page. You can continue an old session by "opening" an old document, or by specifying the name of such a document at the time you start Maple.

The Maple session then proceeds by entering Maple comments, and studying Maple's responses (which appear in the document) and inserting comments. The comments are entered as paragraphs, much as in any word processor. The document is called a Maple worksheet.

For the most part, a Maple worksheet behaves much like any word processor document. You create descriptions or commands by typing, backspacing, and generally editing the contents of the document using all the standard operations of a word processor.

It differs from a word processor document in that selected paragraphs (or lines of input) can be designated as being *command input regions.* These command input regions are somewhat restricted in that they must contain only valid Maple commands.

However, if you press **ENTER** anywhere on a line containing a complete Maple command the command is handed off to the Maple computational engine, the requested computations are carried out, and then inserted into the document in the paragraph immediately following the command.

Thus, it is just like editing a document with an eager mathematical assistant at your side, waiting for you to delegate various mathematical tasks.

Typically, upon starting Maple, the cursor is placed on a new line beginning with a special prompt character > . This "prompt" indicates that you are in a command region. You proceed by typing your command and pressing the Enter key.

In the sample session which follows, the Maple commands are shown in a distinct `font` on lines beginning with the special prompt character > . Typically, they consist of mathematical formulae together with a function indicating what action or transformation is to be performed, and ending in a semi-colon.

Complete commands end in semi-colons (;). This is so that more than one maybe grouped on one line. In the absence of a terminating semi-colon Maple normally expects the command to be completed on the next line.

To execute a command, make sure that the insertion point is some where on the line containing the command, and press Enter to compute (or re-compute) the result and display the answer.

Symbolic algebra systems are impressive in terms of the range and type of computations they can do.

Three typical commands[1] and their results are as follows.

```
> Sum( (i+5)^3,i=1..n);
```

[1]The semi-colons are essential parts of the commands (serving to indicate completion). Also, In the last two commands, the double quote " is used to indicate the "value of the last computed result".

$$\sum_{i=1}^{n} (i+5)^3$$

```
> expand(");
```

$$\sum_{i=1}^{n} (i^3 + 15\,i^2 + 75\,i + 125)$$

```
> value(");
```

$$\frac{1}{4}(n+1)^4 + \frac{9}{2}(n+1)^3 + \frac{121}{4}(n+1)^2 + 90\,n - 35$$

The result produced by each command is inserted immediately following the actual command.

A First Encounter with Maple

As already indicated, working with Maple is like working with an expert mathematical assistant. This requires a subtle change in the way you think about a problem. Instead of focusing on "how do I do XYZ?" your primary role becomes that of deciding "what" needs to be done next.

Much of discrete mathematics is about understanding the relationship between actual objects or sets of objects and mathematical models that are set up to capture some property of these objects. Understanding these relationships often requires that you view either the objects or the associated mathematical model in different ways.

Maple allows you to manipulate the mathematical models almost casually. For example, the polynomial $(x + (x + z)y)^3$ can be entered into Maple as the command

```
> (x + (x+z)*y )^3;
```

$$(x + (x + z)\,y)^3$$

The result of executing this "command" is displayed immediately after the command. In this case Maple simply echoes the polynomial as no special computations were requested.

The power of having a mathematical assistant now becomes apparent because a wide range of standard operations become immediately available. For example, you can expand, differentiate, integrate simply by deciding that this is what is needed. Maple uses (") to refer to the value of the previous computation. Thus a full expansion of the previous polynomial takes place in response to the command

```
> expand(");
```

$$x^3 + 3\,y\,x^3 + 3\,x^2\,y\,z + 3\,y^2\,x^3 + 6\,y^2\,x^2\,z + 3\,x\,y^2\,z^2 + y^3\,x^3 + 3\,y^3\,x^2\,z + 3\,y^3\,x\,z^2 + y^3\,z^3$$

Perhaps you wish to view it as if it were actually a polynomial in x with the y's and z's placed in the coefficients. To collect the previous polynomial as a polynomial in x, use the command

```
> collect( " , x );
```

$$(y^3 + 3\,y^2 + 1 + 3\,y)\,x^3 + (3\,y\,z + 3\,y^3\,z + 6\,y^2\,z)\,x^2 + (3\,y^3\,z^2 + 3\,y^2\,z^2)\,x + y^3\,z^3$$

To return to a factored form, simply request that the previous result be factored.

```
> factor(");
```

$$(x + y\,x + y\,z)^3$$

The flexibility to move quickly between different representations of the same object will prove to be extremely useful when looking for a solution to a problem.

Armed with this kind of expert mathematical assistant, experiments (even complicated ones) can be run quickly so that you are freer to explore ideas and models.

A second very important benefit is that the particular computations that you choose to perform (even lengthy ones) are performed accurately. Thus, the feedback you do get from your experiments is much more likely to be feedback on the model you had chosen rather than nonsense arising because of a simple arithmetic error.

Finally, the sheer computational power of such an assistant allows you to run much more extensive experiments. This can be important when trying to establish or identify a relationship between a mathematical model and a collection of discrete objects.

An Interest Rate Problem

To see how working with a computational assistant like Maple can help with the problem solving process let us build a model for computing compound interest at (say) 10% interest per annum, compounded annually.

In almost every investigation it is helpful to start with a particular example. Thus we consider a case where the initial balance is given by:

```
> p(0) = 1000;
```

$$p(0) = 1000$$

We will want to refer to this equation again so we give it a name, say eq0 by using the name assignment operator `:=` . (Remember that (") refers to the previously evaluated expression!)

```
> eq0 := " ;
```

$$eq0 := p(0) = 1000$$

The interest rate can be given similarly by the (named) equation eq1 as in

```
> eq1 :=   ( i = .1 );
```

$$eq1 := i = .1$$

The total deposit p(1) at the end of the first interest period is related to p(0) by the equation

```
> p(1) =  p(0) + i * p(0);
```

$$p(1) = p(0) + i\,p(0)$$

To compute a numerical value for p(1), we simply use the equation eq0 and eq1 to substitute in a specific values for i and $p(0)$. This is easily done via the `subs` command as in

```
> subs( {eq0, eq1} , " );
```

$$p(1) = 1100.0$$

We now have a specific solution for one interest period. Also, we have invested some effort in setting up the problem using the correct notation for the mathematical assistant. How does this help us?

1. We can now easily re-compute specific examples for different values of $p(0)$ and i. Such recomputations are often used to gain more insight or to test the model for extreme cases. For example if the interest rate is 0, we would expect p(1) and p(0) to be equal. (This kind of test helps us to verify that our model makes sense.) To verify this here, go back to the command line defining eq1 and by selecting and typing, change the value of i to 0. Then simply hit the ENTER key several times until a new value for p(1) is computed.

   Note that without further typing or data entry, each time you hit ENTER, the current command is executed and the cursor advances to the next command. This allows you to run through the steps of your emerging model quickly.