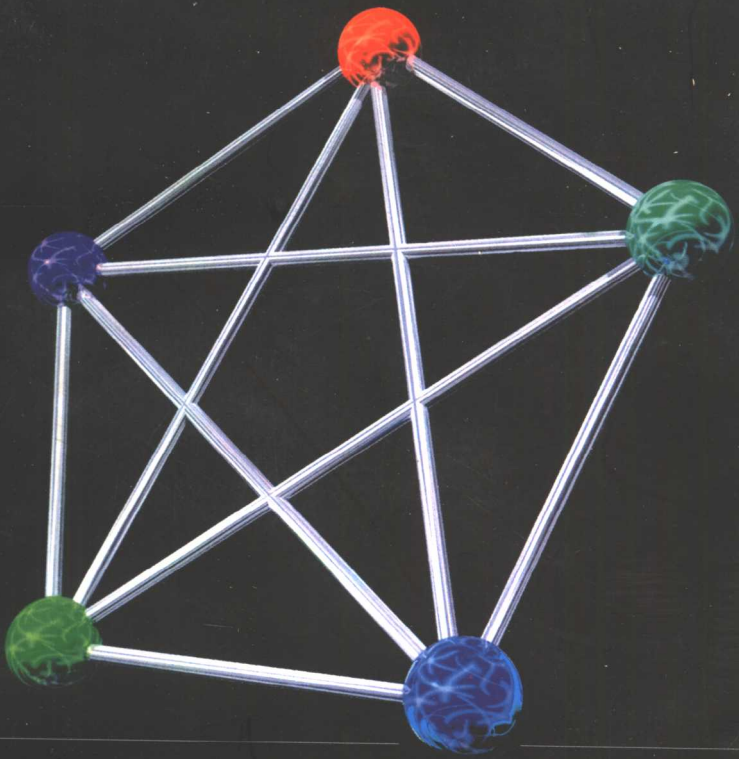


DISCRETE MATHEMATICS

FOR COMPUTER SCIENTISTS

离散数学

JOHN TRUSS

SECOND EDITION
第2版

世界图书出版公司

Discrete Mathematics for Computer Scientists

2nd edition

J.K. Truss
University of Leeds

世界图书出版公司
北京·广州·上海·西安

书　　名：Discrete Mathematics for Computer Scientists
作　　者：J. Truss
中 译 名：离散数学
出 版 者：世界图书出版公司北京公司
印 刷 者：北京中西印刷厂
发　　行：世界图书出版公司北京公司（北京市朝阳门内大街 137 号　100010）
开　　本：大 32 开　850 × 1168　　印　张：19
版　　次：1999 年 4 月第 1 版　　1999 年 4 月第 1 次印刷
书　　号：7-5062-4117-X/ O•237
版权登记：图字 01-99-0744
定　　价：98.00 元

Preface

Discrete mathematics and mathematical logic lie at the heart of any modern study of computer science. The rise of the digital computer over the second half of the twentieth century has coincided with a growth of interest in these fields, and discrete mathematics has now become a major area of mathematics in its own right. Of course it is possible to make excellent and effective use of computers without involving oneself in mathematical considerations. But any understanding of how computers operate, in terms of either their hardware or software, inevitably involves mathematics, even ignoring the fact that many of the applications of computing lie in facilitating numerical and combinatorial calculations.

The sort of mathematics that arises in a computing context is not necessarily what most people would consider to be mathematics at all. Its character may seem more like that of 'mere' organization, symbol management, or data manipulation. Can one really justify calling this subject, where one may not even mention numbers, *mathematics*? It is quite true that the mathematics required for computing is not the classical mathematics of Newton and Leibniz. But mathematics is much more than that, and in its role of searching out patterns, treating them systematically and rigorously, passing from the particular to the general, is bound to adapt to changing circumstances. The particular discipline which more than any other has come into its own as a result of this change of emphasis is mathematical logic, which now has a similar status in computer science to that of classical mathematics in newtonian physics.

Mathematics is a fascinating and significant subject in its own right which, irrespective of any applications, deserves attention from anyone wishing to understand the world we live in, and the complicated and beautiful patterns it contains. The fact that mathematics is also useful can be regarded as a genuine bonus, though it is perhaps an inevitable consequence of its deep underlying logic and elegance. I would like to feel that, although many who read this book may do so because they have to, they will nevertheless obtain some pleasure, and that mathematics will be seen as it should be: not as a complicated and incomprehensible collection of irrelevant abstractions, but as a living and powerful subject, one full of delight and interest for those who study it.

Readership and prerequisites

The principal intended readership for the book are first- or second-year computer science undergraduates at freshman or sophomore level. Much of the material here is appropriate for courses in discrete mathematics and theoretical computer science at this stage, in particular most of the first eight chapters. Other topics have been included which certainly come under the heading of mathematics suitable for a computer science readership, but which are pitched at a slightly higher level. Whereas some of these can, one hopes, be tackled by first-year students, many are normally taught at a later stage, in the second or even third year (junior level). This, it is hoped, will make the book more widely useful, and will make it possible for an enthusiastic student to see where the more basic work leads, while still being sufficiently approachable for relative beginners. A small number of sections have been 'starred', to indicate that they are rather more advanced than the main part of the text, and they are not directly used in what follows them. Careful note has been taken of the ACM recommendations for a discrete mathematics syllabus; for example, the syllabuses proposed by Bertziss (1987) are fully covered by what is included in the book.

The presentation is intended to be essentially self-contained: it should in principle be possible for a reader with very little mathematical background to understand almost everything. In particular, it should be emphasized that a knowledge of the differential and integral calculus is *not* a prerequisite. They are referred to and used occasionally, but their role is relatively minor, and the one or two sections where they are needed can be omitted without great detriment. (The differential and integral calculus are not to be confused with the propositional, predicate, and lambda calculus, which most certainly *do* occur, and are very important.)

Some mathematical maturity is nevertheless desirable, particularly when coping with the more advanced material in the later chapters. Throughout the text, knowledge of basic high-school algebra will be assumed, and some knowledge of programming concepts would be a help. To assist readers who are less familiar with the necessary background material, I have included in the second edition a short appendix as revision of some basic algebra, polynomials, factorization, rational functions, and complex numbers, with some exercises. Where sections of algorithms or 'programs' have been included, the notation should be clear; the style adopted is mostly that of a PASCAL-like language.

Outline of topics

The first chapter begins at the beginning with the basic data types of computer science (or for that matter, mathematics), which are examined in some detail. They are the natural numbers, integers, rationals, reals, and radix systems. Pride of place is given to mathematical induction, one of the most important topics for a computer scientist, in view of looping and recursion in computer programs, and there is some discussion of simple correctness proofs. It is here that the central notion of an *algorithm* is first described.

Mathematical logic is the cornerstone of contemporary computer science. This may be familiar to the student as applied to switching circuits, but the impact of logic goes much deeper than this. Computer programs are themselves 'formal proofs' in a sense (or at least formal *computations*) so their construction and verification is, or is akin to, formal logical manipulation. Two chapters are devoted to logic, Chapters 2 and 7. In Chapter 2 we introduce the two main kinds of language considered, propositional logic, where the basic (atomic) building blocks are 'propositions' linked by the connectives 'and', 'or', 'not', and 'if...then...', and first-order predicate logic, where variables which are intended to range over a possible domain of interpretation may be quantified as 'there exists' or 'for all'. This chapter includes some familiar computing applications, as well as explaining how to write appropriate sentences of natural language in a formal way.

Topics which are more central in mainstream logic are presented in Chapter 7, principally the notion of a formal proof, here given using a natural deduction system, and the links with interpretations (the semantics) via the completeness theorem for propositional or predicate logic are explained. These ideas are also central in the study of programming languages, where one can consider the formal, syntactic aspects, as well as the meaning of a program, either how it is intended to behave (its operational semantics), or what it is intended to stand for (its denotational semantics). As an illustration of these connections, some introductory material on logic programming is given in Section 7.3. Topics allied to logic, but which are introduced before Chapter 7, are boolean algebras and complete lattices (the latter used in denotational semantics). These are covered in Chapter 6, together with other important notions associated with orderings of various kinds, the most pervasive of which is that of a tree.

In Chapter 3 the modern language for handling sets, relations, and functions is introduced. The view is taken that programs are more or less the same as functions, or at any rate that functions are realized by implementations of programs.

Chapter 4 serves principally as an introduction for other parts of the book. Linear algebra occurs widely in applications, for example in combinatorics, graphs, and coding theory. The last two sections of the chapter treat groups and semigroups, also topics of wide applicability. Only the basics are covered here, but enough of a grounding is given to meet the requirements of later chapters.

Combinatorics, which is discussed in Chapter 5, is an attractive and popular part of the subject, and one could say that this is what really constitutes the core of discrete mathematics. This chapter also includes a section on probability (the ideas and arguments are very closely related to those of combinatorics) and one on the solution of difference equations. Difference equations are required for analysis of algorithmic complexity, and since computers are frequently used in the numerical solution of differential equations via their discretized versions (which *are* difference equations), there is a dual purpose in including this material.

A central topic in discrete mathematics is graph theory, and this is covered in Chapter 8. The language of graph theory allows us to visualize combinatorial problems diagramatically, and many issues which arise in computing correspond to geometrical features of the resulting graphs or digraphs, for instance connectedness and accessibility, colourings of vertices, and the existence of paths of various

sorts. There is space here to include only an introduction to the subject, but the topics chosen are ones which form strong links with those in the rest of the book, such as trees, logic, and algorithms.

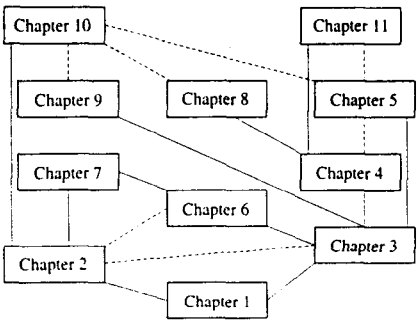
The final three chapters examine slightly more advanced topics, which nevertheless are of central importance. Ever since Alan Turing gave his analysis of the notion of computability in 1936, even before the first electronic digital computer had been built, the study of formal machines of various sorts has proved an active area, and has provided the appropriate theoretical background for the study of the capabilities and limitations of real machines. Chapter 9 studies finite automata and the corresponding regular languages they recognize, and then looks at the wider notions of computability on register machines and Turing machines (which turn out to be equivalent). The theme of feasibility – the question not only of whether something can be calculated, but also of how long it might take – is taken up in Chapter 10. Apart from some analysis of well-known arithmetical algorithms, methods for sorting and searching are described. The most important topic of Chapter 10, however, is that of NP-completeness, associated as it is with so many combinatorial problems arising in practice, for instance in operational research.

Chapter 11, on coding theory, is devoted mainly to error-correcting codes (cryptography is also mentioned at the end of Chapter 10). The modern theory of error-correcting codes, apart from its evident utility, provides a striking illustration of the application of algebraic and combinatorial methods in computer science. Specific topics covered are linear block codes, variable-length and Huffman codes, and the chapter concludes with an introduction to information theory, which among other things studies the limitations on the amount of information it is possible to convey over a channel.

Structure and use

The pedagogical approach used is fairly standard. I have attempted to aim above all for clarity and an interesting and relevant choice of material. The chapters are arranged in sections with their own introductions and exercises. There are over 700 exercises in all, and within each section these have been arranged as far as possible in increasing order of difficulty, rather than in order of presentation of the topics. Answers to selected exercises are provided. Answers to the remaining exercises are available in a *Solutions Manual* from Addison Wesley Longman. There is a summary at the end of each chapter. Terminology is for the most part standard and special symbols have been listed at the beginning of the book. In order to distinguish clearly between logical symbols and the corresponding operations in a boolean algebra, I have used $\wedge$, $\vee$, and $\neg$ (not) for the former, and $\wedge$, $\vee$, and $-$ for the latter. I have used hcf (highest common factor) as opposed to gcd (greatest common divisor), and this and other usages are clearly indicated in the index and list of symbols.

The reader is of course at liberty to read (some or all of) the chapters in any order he or she wishes. Although I have adopted what seems to me a sensible and logical development of the material, many other ways of organizing the work could also have been followed. It is certainly possible to take things in other orders



or to treat topics selectively. The principal pattern of chapter dependences is shown in the diagram, with strong and weak links indicated, though there are occasional other cross-references between chapters not shown as connected. Weak dependences typically occur where only a section of an earlier chapter is a prerequisite for the later one; this makes for greater flexibility than is indicated in the diagram. For instance, the only essential dependence of Chapter 5 on Chapter 4 comes about in the section on permutation groups (Section 5.2), and it is possible to understand the rest of Chapter 5 on the basis of the first three chapters alone. I illustrate four possible courses which could be built from parts of the material covered:

1. Introductory course on discrete mathematics: Sections 1, 2.1, 3.1–3.3, 4.1–4.5, 5.1–5.2, 6.1–6.2, 8, and 10.1–10.2.
2. Course on the foundations of computability and logic: Sections 1, 2, 3, 4.6, 6, 7, and 9.
3. Course on algebraic methods in computer science: Sections 1, 3, 4, 5.2, 6.3, and 11.
4. Course on discrete mathematics, with emphasis on algorithms: Sections 1, 2.1, 3, 5, 8, 9, and 10.

One could envisage other selections. Also, if students had the necessary prior knowledge, it would be possible to omit some of the introductory material in planning a more advanced course. Best of all though is to use the whole book.

Acknowledgments

I would like to acknowledge the helpful comments on drafts of this book made by R.A. Duke, R.O. Gandy, J.C. Higgins, J. van Leeuwen, A. McGettrick, D.M. Spresser, and S.S. Wainer. I would particularly like to thank R.O. Gandy for permission to use unpublished material of his on register machines (Section 9.2). R.R. Burnside and A.R.G. Macdivitt provided encouragement in the early stages when I was teaching a course at Paisley College which included much of the material used

here. I would like to thank the editorial staff at Addison Wesley Longman, in particular S. Mallen and S. Plumtree, for helpful suggestions and encouragement throughout the project, and E. Mitchell and anonymous referees during preparation of the second edition.

J.K. Truss, Leeds
March 1998

Contents

List of symbols

(Note that some symbols are used in more than one sense, but the context should make it clear which applies.)

D_4	group of symmetries of a square	183
D_n	dihedral group of order $2n$	184
dom f	domain of function f	99
$d(x, y)$	distance from x to y; Hamming distance between x and y	327, 466
e	base of natural logarithms, $= 2.718\,28\ldots$	4
E	shift operator, $E(f)(n) = f(n+1)$; edge set of a graph	245, 320
$e(C)$	average error probability of code C	507
$\dot{e}(C)$	maximum error probability of code C	507
eor	exclusive *or*	73
$E(X)$	expected value (mean) of X	220
exp	exponential operator, $\exp(x) \equiv e^x$	234, 240
F	false; set of faces of a planar representation of a graph; set of final states of a finite automaton or Turing machine	53, 331, 372, 414
f'	derivative of f (sometimes)	108
f^{-1}	inverse of function f	102
fA	image of set A under function f	100
$f^{-1}A$	inverse image of set A under function f	101
$f(e)$	final vertex of edge e	320
$f \approx g$	f and g have equal growth rates	426
fix (σ)	fixed-point set of permutation σ	210
F^n	n-dimensional vector space over field F	144
${}^rF_n$	number of functions from r-element set onto n-element set	196
F_q	field with q elements	149
$f \mid X$	restriction of function f to X	104
$f \mid \rightarrowtail y$	function f with x overridden by y	105
$f(x)$	image of x under function f	100
$f : X \to Y$	f is a function from X into Y	100
$f : x \mapsto y$	equivalent to $f(x) = y$	107
$\overline{G}$	complement of graph G	463
gcd	greatest common divisor (= hcf)	36
$g \circ f$	composition of functions f and g	102
$\lvert G : H \rvert$	index of subgroup H in G	182
gp (X)	subgroup generated by X	180
G_x	stabilizer of x in permutation group G	209
$H(A, B)$	joint entropy of A and B	508
hcf	highest common factor	36
$H(X)$	entropy of random variable X	506
$H(X\|Y)$	conditional entropy of X, given Y	514
i	square root of -1	5, 525
i	program input	103
$\tilde{i}$	instruction number i	383
$\bar{i}$	contents of register i	383

$I(A\|B)$	system mutual information	508
$I(a_i\|b_j)$	gain in information about a_i given by b_j	508
id_X	identity on X	102
$i(e)$	initial vertex of edge e	320
$I(E)$	information conveyed by event E	504, 505
I_n	integrating factor; $n \times n$ identity matrix (written as I if n is understood)	167, 239
$K_{m,n}$	complete bipartite graph on m, n vertices	328
K_n	complete graph on n vertices	323
$\overline{K}_n$	graph with n vertices and no edges	337
lcm	lowest common multiple	45
$lim_{n-\infty} a_n$	limit of a_n as n tends to ∞	426
$L(M)$	language accepted by machine M	372
$\log n$	logarithm of n to base 2	248
$\log_b a$	logarithm of a to base b	429
$\max(a, b)$	greater of a and b	188
$\min(a, b)$	smaller of a and b	346
$m \dot{-} n$	$m - n$ if $m \geqslant n$, 0 otherwise	405
$\mathcal{N}$	model of arithmetic with domain $\mathbb{N}$	304
$\mathbb{N}$	set of natural numbers $\{0, 1, 2, \ldots\}$	2
$n!$	n factorial $= n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1 (0! = 1)$	45
nand	formal symbol for 'not ... and ...'	65
$\mathbb{N}_N$	semigroup of integers $\geqslant N$ under +	185
nor	formal symbol for 'not ... or ...'	66
NP	class of problems soluble in polynomial time on non-deterministic Turing machine	454
$\binom{n}{r}$	number of ways of choosing r elements out of n	194
o	program output	103
$f = O(g)$	f has growth rate less than or equal to that of g	425
$f = o(g)$	f has slower growth rate than g	426
P	class of problems soluble in polynomial time on deterministic Turing machine	455
$\{P\}_m(x_1, \ldots, x_m)$	output on running program P on input $(x_1, \ldots, x_m)$	391
$\{P\}_m(x_1, \ldots, x_m)\downarrow$	computation by P on input $(x_1, \ldots, x_m)$ converges	391
$\{P\}_m(x_1, \ldots, x_m)\uparrow$	computation by P on input $(x_1, \ldots, x_m)$ diverges	391
$P(n, r)$	number of r-tuples of distinct members of n-element set	194
$\mathcal{P}^n(X)$	nth power set of X	204
$\Pr(E)$	probability of event E	215
$\Pr(X\|Y)$	conditional probability of X given Y	218
$[P](S)$	final state of computation by P from state S	391
$[P](S)\downarrow$	computation by P starting from state S converges	391
$[P](S)\uparrow$	computation by P starting from state S diverges	391
$\mathcal{P}(X)$	power set of X	88

$\{x : P(x)\}$	set of x having property P	86
Q	set of states of finite automaton or Turing machine	372, 413
$\mathbb{Q}$	set of rational numbers	3
q_0	initial state of finite automaton or Turing machine	372, 414
$\mathbb{R}$	set of real numbers	4
$\mathcal{R}$	class of regular languages	371
R^{-1}	inverse of relation R	95
range f	range of function f	100
r.e.	recursively enumerable	402
rev (n)	number whose radix 10 expansion is reverse of that of n	21
$\text{rev}_r(n)$	number whose radix r expansion is reverse of that of n	412
$\mathbb{R}^n$	n-dimensional real vector space	144
$R \mid X$	domain restriction of relation R to X	104
RTC (R)	reflexive transitive closure of relation R	338
s	assignment of values in domain to variables	301
S	register machine state; successor function	390, 403
S'	next state after S	390
$\underline{S}$	formal symbol for successor	75
sg x	1 if $x > 0$, 0 if $x = 0$	408
S_n	Stirling's approximation to $n!$	198
sp (X)	subspace spanned by X	146
$S \circ R$	composition of relations R and S	101
Sym (X)	group of all permutations of X	179
$\mathcal{S}_n$	symmetric group on n letters	179
t/x	result of substituting term t for variable x	312
T	true	53
TC (R)	transitive closure of relation R	338
$T_m(e, x, y)$	Kleene's T-predicate	396
U_i^n	projection function $U_i^n(x_1, \ldots, x_n) = x_i$	403
$U + \boldsymbol{x}$	translated subspace of U by vector $\boldsymbol{x}$	164
V	vertex set of a graph	320
v	assignment of truth values	56
V_n	set of vertices of a graph having degree n	335
(x), $(x)_\sim$	equivalence class containing x	117
x^{-1}	inverse of x (under some binary operation)	176
$-x$	boolean inverse of x	272
$[x]$	integer part of x	269
$\boldsymbol{x}$	vector	143
$\lvert x \rvert$	absolute value of real number x	426
$[X]^n$	set of n-element subsets of X	229
$x\,R\,y$	relation R holds between x and y ('infix' notation)	94
$X - Y$	difference of two sets	88

$[x, y]$	closed interval $\{z : x \leqslant z \leqslant y\}$	97
$[x, y)$	semi-open interval $\{z : x \leqslant z < y\}$	97
(x, y)	open interval $\{z : x < z < y\}$; ordered pair of x and y	91, 97
$x \hat{} y$	(sometimes $x \cdot y$) x concatenated with y	185, 371
$\|z\|$	modulus of complex number z	526
$\bar{z}$	complex conjugate of complex number z	526
$\mathbb{Z}$	set of integers	2
Z	zero function	403
$\mathbb{Z}_n$	set of integers modulo n	7
$\Gamma \models \varphi$	φ is logical consequence of Γ	55, 301
$\Gamma \vdash \varphi$	φ is provable from Γ	287
δ	transition function for finite automaton or Turing machine; $\delta(x) = x \dot{-} 1$	372, 414, 405
Δ	symmetric difference, $A \Delta B = (A - B) \cup (B - A)$; difference operator, $(\Delta f)(n) = f(n+1) - f(n)$	97, 234
ϵ	empty string; (small) positive real number	127, 427
$\lambda x . P(x)$	function taking x to $P(x)$ for each x	108
μ	least number operator	410
$(\Omega, \Pr)$	sample or probability space	215
π	ratio of circumference of circle to its diameter	4
π	partition of a set	118
$\pi_\sim$	partition defined by equivalence relation $\sim$	118
$\pi(n, k)$	number of partitions of n-element set into k pieces	204
$\rho(n, k)$	number of ordered partitions of n-element set into k pieces	204
σ	standard deviation	221
$\varphi \equiv \psi$	formulae φ and ψ are logically equivalent	63, 80
Σ	alphabet (list of symbols)	127
Σ^*	set of strings over alphabet Σ	127
χ_A	characteristic function of A	106
$\models \varphi$	formula φ is valid	55, 301
$\vdash \varphi$	formula φ is provable	291
$\mathbf{0}$	zero vector	143
$\mathbb{0}$	'zero' of a boolean algebra	272
$\underline{0}$	formal symbol for 0	75
$\mathbb{1}$	'one' of a boolean algebra	272
$\prod$	product	188
$\sum$	summation	11
$\int$	integral	108
$\sim$	general equivalence relation	117
$\forall$	for all	22
$\exists$	there exists	75
$\wedge$	and (formal use)	50
$\vee$	formal symbol for 'or'	50