

Michael A. Henning
Jan H. van Vuuren

Graph and Network Theory

An Applied Approach using
Mathematica®

Springer Optimization and Its Applications

Volume 193

Series Editors

Panos M. Pardalos , *University of Florida*
My T. Thai , *University of Florida*

Honorary Editor

Ding-Zhu Du, *University of Texas at Dallas*

Advisory Editors

Roman V. Belavkin, *Middlesex University*
John R. Birge, *University of Chicago*
Sergiy Butenko, *Texas A&M University*
Vipin Kumar, *University of Minnesota*
Anna Nagurney, *University of Massachusetts Amherst*
Jun Pei, *Hefei University of Technology*
Oleg Prokopyev, *University of Pittsburgh*
Steffen Rebennack, *Karlsruhe Institute of Technology*
Mauricio Resende, *Amazon*
Tamás Terlaky, *Lehigh University*
Van Vu, *Yale University*
Michael N. Vrahatis, *University of Patras*
Guoliang Xue, *Arizona State University*
Yinyu Ye, *Stanford University*

Aims and Scope

Optimization has continued to expand in all directions at an astonishing rate. New algorithmic and theoretical techniques are continually developing and the diffusion into other disciplines is proceeding at a rapid pace, with a spot light on machine learning, artificial intelligence, and quantum computing. Our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in areas not limited to applied mathematics, engineering, medicine, economics, computer science, operations research, and other sciences.

The series **Springer Optimization and Its Applications (SOIA)** aims to publish state-of-the-art expository works (monographs, contributed volumes, textbooks, handbooks) that focus on theory, methods, and applications of optimization. Topics covered include, but are not limited to, nonlinear optimization, combinatorial optimization, continuous optimization, stochastic optimization, Bayesian optimization, optimal control, discrete optimization, multi-objective optimization, and more. New to the series portfolio include Works at the intersection of optimization and machine learning, artificial intelligence, and quantum computing.

Volumes from this series are indexed by Web of Science, zbMATH, Mathematical Reviews, and SCOPUS.

More information about this series at <https://link.springer.com/bookseries/7393>

Michael A. Henning • Jan H. van Vuuren

Graph and Network Theory

An Applied Approach using Mathematica®



Michael A. Henning
Department of Maths and Applied Maths
University of Johannesburg
Auckland Park, South Africa

Jan H. van Vuuren
Department of Industrial Engineering
Stellenbosch University
Matieland, South Africa

ISSN 1931-6828 ISSN 1931-6836 (electronic)
Springer Optimization and Its Applications
ISBN 978-3-031-03856-3 ISBN 978-3-031-03857-0 (eBook)
<https://doi.org/10.1007/978-3-031-03857-0>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Dedication

We dedicate this book with love and appreciation to our mothers.



Joyce Henning-Smith



Annalene van Vuuren

Contents

Preface xiii

List of Algorithms xxv

List of Biographical Notes xxvii

Part 1 Topics in network optimisation

Chapter 1	An introduction to graphs 1
1.1	Introduction 1
1.2	What is a graph? 2
1.3	Special graphs 6
1.4	Operations on graphs 9
1.5	Degree sequences 11
1.6	Isomorphisms 14
1.7	Directed graphs 16
1.8	Representing a (di)graph on a computer 18
1.9	Multigraphs and pseudographs 19
	Exercises 20
	Computer exercises 25
	Projects 29
	Further reading 32

Chapter 2	Graph connectedness 35
2.1	Introduction 35
2.2	Connected graphs 36
2.3	Distance in graphs 38
2.4	Cut-vertices and bridges 40
2.5	Directed graphs 41
2.6	Further study of connectivity in graphs 43
	Exercises 43
	Computer exercises 45
	Projects 48
	Further reading 53

Chapter 3	Algorithmic complexity 55
3.1	Introduction 55
3.2	“Big \mathcal{O} ” notation 60
3.3	A classification scheme for decision problems 63
3.4	Polynomial-time reducibility and NP-completeness 64
3.5	Decision problems vs. computation problems 70

	Exercises 73
	Computer exercises 76
	Projects 78
	Further reading 85
Chapter 4	Optimal paths 87
4.1	Introduction 87
4.2	Distance in weighted graphs 90
4.3	Shortest paths in weighted graphs 91
4.4	Longest paths in weighted digraphs 99
	Exercises 103
	Computer exercises 105
	Projects 107
	Further reading 113
Chapter 5	Trees 115
5.1	Introduction 115
5.2	Properties of trees 116
5.3	Constructing minimum spanning trees 119
5.4	Rooted trees 127
5.5	Depth-first tree searches 130
	Exercises 134
	Computer exercises 136
	Projects 138
	Further reading 144
Chapter 6	Location problems 145
6.1	Introduction 145
6.2	The centre of a graph 148
6.3	The median of a graph 152
6.4	General centres and medians 153
6.5	Absolute centres and medians 155
6.6	General absolute centres and medians 159
	Exercises 166
	Computer exercises 168
	Projects 168
	Further reading 171
Chapter 7	Maximum flow networks 173
7.1	Introduction 173
7.2	Preliminary concepts 174
7.3	The max-flow min-cut theorem 177
7.4	The max-flow min-cut algorithm 179
	Exercises 183
	Computer exercises 184
	Projects 187
	Further reading 191

Chapter 8	Minimum-cost network flows 193
8.1	Introduction 193
8.2	Network flow and linear programming theory 195
8.3	Basic feasible solutions 198
8.4	The network simplex algorithm 202
	Exercises 211
	Computer exercises 212
	Projects 214
	Further reading 219
Part 2	Topics in classical graph theory
Chapter 9	Matchings 223
9.1	Introduction 223
9.2	Maximum matchings 224
9.3	Vertex covers 227
9.4	Tutte's theorem 232
9.5	The Tutte-Berge formula 234
9.6	The binding number of a graph 246
9.7	Matching algorithms for bipartite graphs 250
9.8	A matching algorithm for general graphs 252
9.9	A weighted matching algorithm 262
	Exercises 281
	Computer exercises 284
	Projects 285
	Suggestions for further background reading 289
	Further reading 289
Chapter 10	Eulerian graphs 293
10.1	Introduction 293
10.2	Finding eulerian circuits and trails 294
10.3	The Chinese postman problem 300
10.4	Other postman problems 306
10.5	Eulerian digraphs 308
10.6	Fleury's algorithm for digraphs 309
	Exercises 311
	Computer exercises 314
	Projects 317
	Further reading 320
Chapter 11	Hamiltonian graphs 323
11.1	Introduction 323
11.2	Which graphs are hamiltonian? 325
11.3	The closure function 329
11.4	The travelling salesman problem 334
11.5	Almost traceability and almost hamiltonicity 344
	Exercises 348

	Computer exercises 350
	Projects 352
	Further reading 357
Chapter 12	Graph connectivity 359
12.1	Introduction 359
12.2	Cuts and separating sets 360
12.3	Blocks 361
12.4	Connectivity and edge connectivity 368
12.5	Menger's Theorem 370
12.6	Computing the connectivity of a graph 377
	Exercises 380
	Computer exercises 384
	Projects 386
	Further reading 392
Chapter 13	Planarity 393
13.1	Introduction 393
13.2	Properties of planar graphs 395
13.3	Which graphs are planar? 398
13.4	The crossing number of a graph 415
13.5	Other parameters related to planarity 420
13.6	Embedding on surfaces other than the plane 425
13.7	The Robertson-Seymour theorems 430
	Exercises 432
	Computer exercises 437
	Projects 438
	Further reading 444
Chapter 14	Graph colouring 449
14.1	Introduction 449
14.2	Vertex colouring 451
14.3	Edge colouring 477
	Exercises 482
	Computer exercises 486
	Projects 488
	Further reading 497
Chapter 15	Oriented graphs 501
15.1	Introduction 501
15.2	Strong orientations 501
15.3	Tournaments 503
15.4	Higher-order rankings in tournaments 506
15.5	Almost traceable and almost hamiltonian orient digraphs 512
	Exercises 517
	Computer exercises 518
	Projects 520

Further reading 523

Part 3 Topics in modern graph theory

Chapter 16	Domination in graphs 527
16.1	Introduction 527
16.2	The domination number 529
16.3	The independent domination number 555
16.4	The irredundance number 569
16.5	Total domination 576
	Exercises 594
	Computer exercises 595
	Projects 599
	Further reading 615
Chapter 17	Ramsey theory 625
17.1	Introduction 625
17.2	The classical two-colour Ramsey problem 628
17.3	Two-colour extensions 633
17.4	The multi-colour Ramsey problem 635
17.5	Multipartite Ramsey theory 636
17.6	Requirements other than that of a subgraph 641
	Exercises 644
	Computer exercises 645
	Projects 647
	Further reading 649
Chapter 18	Extremal graph theory 655
18.1	Introduction 655
18.2	Cycles in graphs 656
18.3	Turán's theorem 657
18.4	Zarankiewicz numbers 659
18.5	Framing numbers 662
18.6	Diameter two graphs of minimum size 667
18.7	Diameter two critical graphs of maximum size 673
	Exercises 678
	Further reading 678
Chapter 19	Graph enumeration 681
19.1	Counting labelled graphs 681
19.2	Counting unlabelled trees 692
19.3	Pólya's enumeration theorem 700
19.4	Using Pólya's theorem to enumerate graphs 708
	Exercises 714
	Computer exercises 717
	Projects 720
	Further reading 722

Chapter 20	The probabilistic method 725
20.1	Introduction 725
20.2	Ramsey theory 726
20.3	Linearity of expectation 727
20.4	Random graphs 734
	Exercises 746
	Projects 746
	Further reading 747
	Index 751
	Subject index 751
	Author index 763

Preface

Being a powerful language capable of describing discrete structure succinctly, graph theory is an area of mathematics that is particularly well-suited to model building and practical problem solving. That said, it is also a beautiful theory in its own right, even if model building is not the primary objective, with many intriguing and long-standing open theoretical questions that are easily posed and understood by lay people, but which have hitherto evaded solution by prominent mathematicians in the field.

The two main themes of this book echo the aforementioned facets of graph theory. These themes are the development of graph theory as a mathematical subject in its own right, as well as the development of useful model building tools and effective model solution techniques which are amenable to computer implementation and which are aimed at solving practical problems. Our aim in this book is to cover a wide variety of topics in graph and network theory — both from a theoretical standpoint and from an applied modelling point of view (also demonstrating the use of Wolfram's **MATHEMATICA**[®] during the latter endeavour).

About this book

The origins of this book date back to the period 2003–2004. During this time, the first author compiled a set of lecture notes while teaching a graph theory course at the University of KwaZulu-Natal, South Africa. At the same time, the second author put together course notes on network optimisation and graph theory for operations research and applied mathematics students at Stellenbosch University, South Africa. We subsequently combined forces and used our respective lecture notes as a foundation for the contents of this book.

A notable aspect of this book, distinguishing it from many other graph theory texts, is the way in which we have sought to integrate the theoretical and applied aspects of the material covered into a coherent whole. Where possible, chapters open with real-life problem descriptions which serve as motivation for the theoretical development of subject matter. Detailed proofs are given of almost all theoretical results. All algorithms are, furthermore, verified for correct working, demonstrated by means of numerical examples, and analysed in respect of computational complexity. Each chapter also closes with three different sets of exercises. The first set of exercises contains so-called classical or standard exercises to cater for the needs of the more mathematically inclined reader. Many of these are routine exercises, designed to test understanding of the material in the text, but some are more challenging and less routine. The second set of exercises is earmarked

for the computer technologically savvy reader and offer computer exercises (using **MATHEMATICA**). The final set of exercises consists of larger projects aimed at equipping those readers with backgrounds in the applied sciences to apply the necessary skills learned in the chapter in the context of real-world problem solving.

For the historically inclined reader, each chapter offers biographical notes as well as pictures of graph theorists and mathematicians who have contributed significantly to the development of the results documented in the chapter. These notes allow the reader to associate faces, as opposed to merely names, with some of the important discoveries and results presented in the book. In total, approximately one hundred biographical notes are presented throughout the book. These notes are intended to render the results presented more interesting and to bring to life the topics covered, thereby hopefully exciting and inspiring the reader.

The time span of the graph theoretic contributions described in the book is wide indeed — covering virtually the entire period over which graph theory has evolved into the modern subject that it is today. This claim may be substantiated by citing the following feature exemplars from the book:

- The seminal work on (di)graph edge traversal by [Leonhard Euler](#) in 1736,
- an elegant solution to the well-known knight's tour problem proposed by [Euler](#) in 1759,
- the influential questions about chess board coverage by chess pieces raised by Max Bezzel in 1848,
- the catalytic questions on vertex traversal posed by [Sir William Rowan Hamilton](#) in 1856,
- the ever-popular puzzle *Towers of Hanoi* proposed by [Édouard Lucas](#) in 1883,
- a proof that every planar graph is 5-colourable put forward by [Percy John Heawood](#) in 1890,
- the origins of a theory of graphs as distilled by [Dénes König](#) in 1916,
- the establishment of what is now known as [Ramsey theory](#) by [Frank Plumpton Ramsey](#) in 1930,
- characterisations of planar graphs by [Kazimierz Kuratowski](#) in 1930 and [Klaus Wagner](#) in 1937,
- early work on matchings and vertex covers by [Dénes König](#) and Jenő Egerváry in 1931,
- the seminal work on graph connectivity by [Hassler Whitney](#) in 1932,
- the ground-breaking combinatorial enumeration theorem by [George Pólya](#) in 1937,
- an upper bound on the chromatic number of a graph established by [Rowland Brooks](#) in 1941,
- an algorithm for computing maximum flows by [Lester Ford Jr](#) and [Delbert Ray Fulkerson](#) in 1956,
- algorithms for finding shortest spanning trees by [Joseph Kruskal](#) in 1956 and [Robert Prim](#) in 1957,
- algorithms for finding shortest paths in weighted graphs by [Edsger Dijkstra](#) in 1956 and [Robert Floyd](#) in 1962,
- the foundational texts on graph theory by [Claude Berge](#) in 1957 and [Øystein Ore](#) in 1962,

- the influential work on a variety of planar graph embeddings by [William Tutte](#) in 1963,
- a conjecture on the domination number of graph products by [Vadim Vizing](#) in 1963,
- the formalisation of basic notions in algorithmic complexity by [Jack Edmonds](#) in 1965,
- proof of the NP-completeness of the satisfiability problem by [Stephen Cook](#) in 1971,
- formalisation of probabilistic methods in combinatorics by [Paul Erdős](#) and [Joel Spencer](#) in 1974,
- the computer proof of the four colour theorem by [Kenneth Appel](#) and [Wolfgang Haken](#) in 1976,
- a relationship between parameters related to graph domination, independence, and irredundance established by [Béla Bollobás](#) and [Ernie Cockayne](#) in 1979, and
- a result on the existence of a finite collection of forbidden minors in embeddings of graphs in surfaces of arbitrary genus by [Neil Robertson](#) and [Paul Seymour](#) in 2004.

The book also contains numerous theoretical developments and graph theoretic results dating from the period 2005–2022. In fact, it also contains a sprinkling of previously unpublished results.

Organisation of material in the book

The material in this book has been organised into three distinct parts, each with a different focus. The first part is devoted to topics in network optimisation, with a focus on basic notions in algorithmic complexity and the computation of optimal paths, shortest spanning trees, maximum flows and minimum-cost flows in networks, as well as the solution of network location problems. The second part is devoted to a variety of classical problems in graph theory, including problems related to matchings, edge and vertex traversal, connectivity, planarity, edge and vertex colouring, and orientations of graphs. Finally, the focus in the third part is on modern areas of study in graph theory, covering graph domination, Ramsey theory, extremal graph theory, graph enumeration, and application of the probabilistic method.

Part 1: Topics in network optimisation

[Part 1](#) of the book contains eight chapters and opens in [Chapter 1](#) with an introduction to basic notions and terminology in graph and network theory. These concepts and the related terminology are essential prerequisites for an understanding of the material presented in the remainder of the book.

[Chapter 2](#) is devoted to a study of graph connectedness and the fundamental concepts of paths and of distance in graphs.

Fundamental concepts in the study of algorithmic complexity are introduced and illustrated in [Chapter 3](#). In this chapter, the reader is introduced to a classification scheme for decision problems, as well as the notions of polynomial-time

reducibility and of NP-completeness. The particular decision problems underlying computation problems are also studied.

In Chapter 4, we consider the problem of computing optimal paths in graphs. For this purpose, the notion of distance in a weighted graph is defined. Two algorithms are presented for computing shortest paths in weighted graphs, namely the 1956 [algorithm of Dijkstra](#) and the 1962 [algorithm of Floyd](#). An [algorithm for computing longest paths in weighted directed graphs](#) is also put forward in the context of project scheduling. All the algorithms presented in this chapter (and indeed in the remainder of the book) are described in general, verified in terms of correct working, and illustrated by means of numerical examples.

The focus in Chapter 5 is on the simplest class of graphs, called trees. After having established a number of basic properties of trees, the problem of constructing minimum spanning trees in weighted graphs is considered. Two algorithms are documented for solving this problem, the 1956 [algorithm of Kruskal](#) and the 1957 [algorithm of Prim](#). Rooted trees are also studied and properties of such trees are established. The chapter closes with a discussion on the ubiquitous notion of a depth-first tree search.

In Chapter 6, we turn our attention to network location problems. Upon considering the classical problems of finding the centre and median of an unweighted graph, based on vertex eccentricity values, we study generalisations of the notions of a centre and a median in the context of weighted graphs. In particular, we describe procedures for computing the classical centre and median (vertex demand & vertex location), the general centre and median (unrestricted demand & vertex location), the absolute centre and median (vertex demand & unrestricted location), and the absolute general centre and median (unrestricted demand & unrestricted location) in weighted (directed or undirected) graphs. All of these procedures are illustrated by detailed numerical examples.

Maximum flow networks are the topic of Chapter 7. Preliminary concepts of networks and flows in these networks are established, as well as important results on network flows. As a consequence of these results, we present the 1962 [max-flow min-cut theorem](#) due to [Ford](#) and [Fulkerson](#) which utilises the well-known augmenting path technique. A 1972 algorithm of [Edmonds](#) and [Karp](#) is also presented which provides a systematic method for [finding an \$f\$ -augmenting path](#) in a network with flow f , if such a path exists.

In Chapter 8, the last chapter of Part 1, we turn our attention to finding minimum-cost flows in networks. After formulating a minimum-cost network flow model as a linear programming problem, we study its associated dual problem using the notion of basic feasible solutions and other fundamental concepts from the realms of linear algebra and linear programming. We also present the well-known [network simplex algorithm](#) for computing minimum-cost flows which is a simplification of the celebrated 1947 [simplex algorithm](#) due to [Dantzig](#) and which has a worst-case time complexity that is significantly faster than the earlier, general-purpose [simplex algorithm](#).

Part 2: Topics in classical graph theory

Part 2 of the book contains a further seven chapters, opening in Chapter 9 with a presentation of the classical 1957 characterisation of maximum matchings in terms

of augmenting paths (although observed by Petersen in 1891 and König in 1931), as well as an important matching result in bipartite graphs due to König in 1931 and Hall in 1935. The concept of vertex covers is also introduced, and relationships between the matching number and vertex covering number are obtained. In addition, the König-Egerváry Theorem, also of 1931, showing that the matching number and vertex covering number are equal in bipartite graphs, is derived. An upper bound is also presented on the matching number in terms of the order, size and the number of vertex disjoint odd cycles in a graph. We, furthermore, describe the famous 1947 characterisation by Tutte of general graphs that admit perfect matchings. As a consequence of this result, we prove a result first observed in 1891 by Petersen that a connected cubic graph with at most two bridges has a perfect matching. We also present an important max-min formula, called the Tutte-Berge formula, for the matching number, and apply this formula to determine a tight lower bound on the matching number of a connected regular graph. Matching algorithms for bipartite graphs are presented, one utilising network flows and the other utilising augmenting paths. Moreover, an algorithm for finding a maximum matching in a general graph is presented, which relies heavily on the so-called “blossom shrinking” method devised by Edmonds in 1961. A polynomial-time algorithm for finding a maximum weight matching in a weighted graph is finally presented.

The topic of study in Chapter 10 is eulerian graphs. After considering the problem of finding eulerian circuits and trails in (multi)graphs, a characterisation of connected eulerian multigraphs, due to Euler in 1736 and Hierholzer in 1873, is presented. This is followed by a similar characterisation of connected multigraphs that possess eulerian trials. We present Fleury’s 1883 algorithm for computing an eulerian trial or circuit in a connected multigraph with at most two vertices of odd degree. Thereafter, we introduce the well-known Chinese postman problem, dating back to 1960, in which a shortest tour (visiting each edge) in a connected, weighted graph of order n is sought, and we determine the exact weight of such a shortest tour. We also present an efficient algorithm for computing a solution to the general Chinese postman problem in $\mathcal{O}(n^4)$ time. A number of more recent variations on the Chinese postman problem are also recounted in this chapter. Eulerian circuits and trails in directed graphs are finally studied, and an algorithm for determining such circuits and trails is presented.

In Chapter 11, we turn our attention to hamiltonian graphs. We start by considering the question of which graphs are hamiltonian. Several necessary conditions for a graph to be hamiltonian are presented. Sufficient conditions for a graph to be hamiltonian are also given, including a simple degree condition established by Dirac in 1952 as well as more complex degree condition statements due to Chvátal in 1972 and Posa in 1976. Using results first observed in 1976 by Bondy and Chvátal, we continue by studying the closure function of a graph. The chapter closes with a discussion on the celebrated travelling salesman problem popularised by Merrill Flood in 1937. Various heuristics are presented for approximating the weight of a shortest hamiltonian cycle in a weighted complete graph, including the well-known nearest-neighbour heuristic. Various lower bounds on such a shortest hamiltonian cycle are given. We finally present the constant-factor approximation algorithm proposed in 1976 by Christofides for finding a hamiltonian cycle of

weight at most $\frac{3}{2}$ times the weight of a shortest hamiltonian cycle in a weighted complete graph satisfying the triangle inequality.

We consider measures of graph connectivity and their relation to communication networks in Chapter 12. We define the notions of cut-vertices and bridges, and, more generally, of vertex cuts and edge cuts, as well as the notion of separating sets in graphs. Properties of blocks in graphs are studied, as are characterisations of blocks. Adopting a depth-first search paradigm, we present an algorithm for efficiently computing the blocks of a connected graph. The concepts of connectivity and of edge connectivity of a graph are discussed next, and we present Whitney's 1932 theorem which relates the connectivity, edge connectivity and minimum degree of a graph. We prove two versions of a classic result of Menger dating back to 1927, namely the vertex form of Menger's Theorem and the edge form of Menger's Theorem. Whitney's 1932 characterisation of k -connected graphs is given, as well as fundamental properties of k -connected graphs established in 1960 by Dirac. We finally demonstrate how to apply the max-flow min-cut theorem of Chapter 7 to determine the connectivity of a connected graph.

In Chapter 13, we study the notion of planarity in graphs. Fundamental properties of planar graphs are established, after which the classical 1758 formula due to Euler, relating the order, size, and number of faces of a connected plane graph, is recounted. Thereafter, the concepts of a contraction, a subdivision and a graph minor are defined, and this is followed by two classic characterisations of planar graphs, one due to Kuratowski in 1930 and the other due to Wagner in 1937. An efficient algorithm for planarity testing is presented and important results related to graph planarity due to Tutte and Thomassen are presented. The crossing number of a graph is considered next, and fundamental results of among others, Turán, Zarankiewicz and Kleitman, are given. Other parameters closely related to planarity are also reviewed, such as the rectilinear crossing number, the thickness of a graph and the coarseness of a graph, and results on these topics due to Beineke and Chartrand are presented. Embeddings of graphs on surfaces other than the plane are finally studied, and results are presented on 2-cell embeddings of graphs and the genus of a graph. The chapter closes with a discussion on a remarkable family of theorems, due to Robertson and Seymour over the period 1983–2004, which generalise the notion of the existence of a finite collection of forbidden minor embeddings in the characterisation by Wagner of planar graphs to surfaces of higher genus.

Chapter 14 is devoted to the topic of vertex and edge colouring in graphs. The chapter opens with a discussion on the notions of a proper (vertex) colouring and properties of graphs that are k -colourable. Heawood's 1890 proof that every planar graph is 5-colourable is presented and it is explained how an earlier attempt by Kempe in 1879 at proving that every planar graph is, in fact, 4-colourable had failed. The rich history of this four-colour problem is recounted, including its eventual (computer) proof by Appel and Haken in 1976. The notions of subgraphs that are critical with respect to k -colourability and of the chromatic number of a graph are introduced. Various bounds on the chromatic number of a graph are established, including the result by Brooks dating back to 1941 which shows that the chromatic number of any graph (other than a complete graph or a cycle) is bounded from above by its maximum degree. Various other results related to the chromatic number of a graph are given, including a 1955 proof by Mycielski that,

for every positive integer k , there is a triangle-free graph with chromatic number k , as well as Nordhaus-Gaddum type bounds on the chromatic number of a graph. A number of heuristics for vertex colouring are presented, including those of [Welsh and Powell](#) (1967) and of [Brélaz](#) (1979). [Brown's exact colouring algorithm](#) (dating from 1972) is also presented. The chapter closes with a study of edge colourings in graphs. In particular, we introduce the notion of the chromatic index of a graph and prove [Vizing's](#) 1964 result that this index is either the maximum degree (for so-called *class 1* graphs) or one more than this number (for so-called *class 2* graphs).

In [Chapter 15](#), the last chapter of [Part 2](#), we study directed graphs that can be obtained from a graph by assigning a direction to each edge (that is, orienting each edge) of the graph. Such directed graphs are called oriented graphs. We present a characterisation, proposed by [Robbins](#) in 1939, of graphs that have a strong orientation. We explore fundamental properties of tournaments, which are orientations of complete graphs. The chapter closes with a discussion on player rankings of various orders in tournaments.

[Part 3: Topics in modern graph theory](#)

[Part 3](#) of the book contains a further five chapters, opening in [Chapter 16](#) with a study of dominating sets in graphs. Properties of minimal dominating sets and bounds on the domination number of a graph are presented. We prove that all the essential upper bounds on the domination number of a graph without isolated vertices and isolated edges can be written in a unified form. As a consequence of this result, [Ore's](#) 1962 upper bound of one-half the order on the domination number of a graph with no isolated vertices follows. This is followed by a description of the improved 1989 upper bound by [McCuaig](#) and [Shepherd](#) of two-fifths the order, except for seven exceptional graphs, when the minimum degree is at least two, and [Reed's](#) further improved 1996 upper bound of three-eighths the order when the minimum degree is at least three. Bounds on the domination number in terms of other parameters, such as the maximum degree and connectivity, are also given. We present [Vizing's](#) result showing, if a graph of a given order has sufficiently many edges, that it is guaranteed to have a dominating set of some specified order, and we describe improvements on this result due to [Sanchis](#) and [Rautenbach](#) during the period 1990–1991. We also discuss [Vizing's 1963 conjecture](#) that the domination number of the Cartesian product of two graphs is at least the product of their domination numbers, and present the double-projection approach by [Clark](#) and [Suen](#), dating from 2000, for results established to date on the conjecture, which still remains open. Nordhaus-Gaddum type bounds on the domination number are discussed. The complexity of determining the domination number and algorithmic results on this parameter are also presented. The upper domination number of a graph is defined and, using edge weight arguments, bounds on the upper domination of regular graphs are established. Similar results are also presented for independent dominating sets (where the dominating set is required to be independent) and for total dominating sets (where every vertex in the dominating set is required to have at least one neighbour in the dominating set). The irredundance number of a graph is defined, and a domination inequality chain

is recounted which relates the domination number, the independent domination number and the irredundance number.

The topic of [Chapter 17](#) is graph Ramsey theory. We present the classical two-colour Ramsey problem, and show that the problem is well-posed by deriving the 1935 closed-form upper bound on its solution due to [Erdős](#) and [Szekeres](#). The values of small Ramsey numbers due to, among others, Greenwood, Gleason, [McKay](#) and [Radziszowski](#) are also presented. This is followed by a discussion on two-colour extensions of the classical Ramsey number (in which subgraphs other than cliques are sought in random edge bi-colourings of complete graphs), with results due to, among others, Faudree, Schelp, and [Chvátal](#) presented. An extension to the multi-colour Ramsey problem is considered, as is an extension to multipartite Ramsey theory. In both the latter cases, fundamental properties of the generalised Ramsey numbers are discussed, general bounds are established and the values of small Ramsey numbers are presented.

We turn our attention in [Chapter 18](#) to topics in extremal graph theory other than the important topic of Ramsey theory covered in the previous chapter. We present [Mantel's 1907 theorem](#) which determines the maximum number of edges in a graph of given order that contains no triangle. The maximum number of edges in a graph of given order that contains no 4-cycle is also given. We present a classic result of [Turán](#) dating back to 1941 which characterises those graphs of given order with the largest possible number of edges that do not contain a clique of specified order. In addition, we investigate bounds on the maximum number of edges in a subgraph of the complete bipartite graph without a specified biclique, and define the so-called [Zarankiewicz](#) numbers. We present results on the [Zarankiewicz](#) problem in which the maximum number of edges in a bipartite graph not containing a given bipartite graph as a subgraph is sought. We also define the framing number of a graph, a term coined in 1992 by [Chartrand](#) and others, and present results on the framing number of a clique and an independent set, and of a clique and biclique. Moreover, we consider an extremal problem due to [Erdős](#) and [Rényi](#) dating from 1962 in which the minimum size of a graph with diameter 2 is sought such that no vertex is adjacent to every other vertex. Diameter-two critical graphs of maximum size are finally studied, and the [Murty-Simon Conjecture](#), which bounds the size of a diameter-2-critical graph, is discussed.

[Chapter 19](#) is devoted to the topic of graph enumeration which involves counting the number of distinct graphs satisfying some property, without having to identify or draw all these graphs explicitly. We develop the necessary combinatorial tools for carrying out such enumerative analyses. These tools include recurrence relations, generating functions and results from basic group theory. We count so-called labelled graphs of various structures. We present [Cayley's 1889 tree formula](#) for counting the number of distinct labelled trees of a given order, and give a proof of [Cayley's formula](#) due to Prüfer, dating back to 1918. In applications to genealogical ancestry, we model a family tree as a special kind of graph called a genealogical register. Using recurrence relations and generating functions, we count the number of distinct genealogical registers of a given order; that is, we enumerate the genealogical registers of a given order and show that these numbers are the well-known [Catalan](#) numbers. We consider the more complex problem of counting various unlabelled graph types and structures, that is, graphs in which vertices are considered indistinguishable. In particular, we count the number of distinct,

rooted, unlabelled trees using the notion of a partition of a positive integer. We also consider the problem of enumerating unlabelled trees of a fixed order that are not rooted. We enumerate more general (unlabelled) graph structures than trees using a more powerful enumeration tool, known as [Pólya's Enumeration Theorem](#), which dates back to 1937. We finally consider the problem of enumerating unlabelled digraphs of given order and size.

In [Chapter 20](#), the last chapter of [Part 3](#), we consider the probabilistic method in graph theory, a powerful proof technique for obtaining structural results of graphs. We present Erdős's 1947 proof of a lower bound on Ramsey numbers that gave birth to the probabilistic method. Using linearity of expectation, we go on to determine a lower bound on the number of hamiltonian paths in a tournament, and we also determine a lower bound on the independence number of a graph, a result established independently by Caro in 1979 and Wei in 1981, known as the [Caro-Wei Theorem](#). Moreover, we apply linearity of expectation to determine an upper bound on the domination number of a graph in terms of its order and minimum degree using the deletion method, also called the alteration method. We study the theory of random graphs introduced by Erdős and Rényi in 1960. We present a surprising 1959 result of Erdős showing that by choosing the order n sufficiently large and by choosing a probability value p carefully, there exists a random graph of order n (where each edge is chosen to be included in the graph with probability p , independently of the choice of any other edges) which contains an induced subgraph with arbitrarily large girth and arbitrarily large chromatic number. A lower bound on the chromatic number of a random graph is also determined. We close the chapter by determining the domination number of a random graph.

The intended audience

We have written this book with three primary audiences in mind. The first potential audience consists of students of mathematics who wish to familiarise themselves with the subject areas of graph theory and network optimisation, including prevalent techniques and major accomplishments in these fields. The large variety of proofs presented in this book are anticipated to help strengthen these students' use of different proof techniques to solve mathematical problems. Many of the chapters (especially in [Parts 2](#) and [3](#)) cover the latest developments in the field, and our hope is that such students will be sufficiently motivated by the book to pursue the resolution of open questions and further development of the topics covered. For students of mathematics we propose the following parts of the book as a possible syllabus in an introductory course on graph theory:

Part	Chapter(s)	Relevant content
1	1, 2, 5	Entire chapters
2	9 10 11 13 14, 15	Sections 9.1 to 9.6 10.1 , 10.3 , 10.4 Entire chapter Sections 13.1 to 13.5 Entire chapters
3	16–20	Entire chapters

The second potential audience comprises students of applied mathematics, operations research or industrial engineering who wish to equip themselves in the application of mathematical skills from the realms of graph theory and discrete mathematics in order to be able to solve real-world problems. Our hope is that such students will be able to draw from this book and build a solid foundation in graph and network theory from which to model and solve practical problems in their fields of applied study. For students of applied mathematics, operations research or industrial engineering we propose the following parts of the book as a possible syllabus in an introductory course on graph and network theory:

Part	Chapter(s)	Relevant content
1	1–8	Entire chapters
2	9 10–12 13 14 15	Sections 9.1, 9.2, 9.7 Entire chapters Sections 13.1, 13.2, 13.6 Entire chapter Sections 15.1 to 15.4
3	16	Sections 16.1, 16.2

The third potential audience contains students of computer science who wish to expose themselves to algorithmic solution procedures for solving graph theoretic or network models that can be implemented on a computer, and to learn how to apply Mathematica to solve problems in discrete mathematics. Our hope is that such students will develop the necessary computer and algorithmic skills offered in this book to solve problems in discrete mathematics. For students of computer science we propose the following parts of the book as a possible syllabus in an introductory course on graph and network theory:

Part	Chapter(s)	Relevant content
1	1–6, 8	Entire chapters
2	9 10 11 13 14 15	Sections 9.1, 9.2 Entire chapter Sections 11.1, 11.4 Sections 13.1, 13.2, 13.6 Sections 14.1, 14.2.1, 14.2.4, 14.2.5 Sections 15.1 to 15.3
3	16	Computer exercises

Notation

While we endeavoured to introduce all notation as we proceeded with the main exposition in the book, there are a small number of exceptions. These exceptions pertain to the set of natural numbers (positive integers), denoted throughout by \mathbb{N} , the set of counting numbers, denoted by $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, the set of real numbers, denoted by \mathbb{R} , and the set of nonnegative real numbers, denoted by \mathbb{R}^+ . In addition, we denote the set containing the first n positive integers by $[n]$, which is a shorthand notation for the set $\{1, \dots, n\}$.

Acknowledgements

We wish to express our sincere thanks to Marc Bailey, Gerdus Benadé, Ronald Benjamin, Wayne Bossenger, Melissa Cunliffe, Daniël de Wet, Antoinette Erasmus, Robert Hagspihl, Günther Hüselmann, Clinton Jacobs, Johan Janse van Rensburg, Amirkiarash Kiani, Martin Kidd, Byron Kilian, Jacqui Kotzee, Mattie Landman, Tobias Liljedahl, Benedikt Meylahn, My-Quan Ngo, Thorsten Schmidt-Dumont, Maryke Thom, Ben Turnbull and Christian van der Walt for their helpful comments, suggestions and remarks which have led to improvements in the general presentation and have enhanced the exposition of the book.

A special acknowledgement is due to Wyatt Desormeaux for proofreading the entire manuscript for us.

Our sincere thanks also go to Werner Gründlingh for his tireless efforts in typesetting the book and for producing superb graphics. We very much appreciate his expertise, and the enormous time sacrifice he has made in assisting us during the typesetting process.

We finally extend a hearty thanks to Alewyn Burger for assisting us with the **MATHEMATICA** exercises at the end of each chapter.

Invitation

We have tried to eliminate errors, but surely several remain. We therefore invite readers to notify us of any typographical errors. Suggested corrections of errors and suggestions for additional material inclusion can be sent to our email addresses below. In short, we would welcome any feedback the reader might have.

Research Professor
Department of Mathematics and
Applied Mathematics
University of Johannesburg

Michael A Henning
e-mail: mahenning@uj.ac.za

Professor of Operations Research
Department of Industrial Engineering
Stellenbosch University

Jan H van Vuuren
e-mail: vuuren@sun.ac.za

List of Algorithms

1	An algorithm for computing the degree sequence of a graph	56
2	An algorithm for solving the puzzle <i>Towers of Hanoi</i>	58
3	An algorithm for computing the clique number of a graph	71
4	An iterative algorithm for computing Fibonacci numbers	74
5	A binary search algorithm	79
6	The Bubble sort algorithm	81
7	The Merge sort algorithm	81
8	The Merge algorithm	82
9	A recursive algorithm for computing Fibonacci numbers	83
10	The euclidean algorithm for computing greatest common divisors . .	83
11	Dijkstra's algorithm for computing shortest paths	93
12	Floyd's algorithm for computing shortest distances	98
13	An algorithm for computing longest paths in a weighted digraph .	100
14	Kruskal's algorithm for computing a shortest spanning tree	121
15	Prim's algorithm for computing a shortest spanning tree	125
16	A depth-first search tree construction algorithm	131
17	An algorithm for computing the centre of a tree	152
18	f -Augmenting path algorithm for solving maximum flow problems .	181
19	The simplex algorithm for solving linear programming problems .	203
20	The network simplex algorithm for solving transportation problems	208
21	A maximum matching algorithm for bipartite graphs	252
22	A maximum matching algorithm for general graphs	258
23	Maximum-weight matching algorithm	267
23a	Function in Step 9 of the Maximum-weight matching algorithm .	268
23b	Function in Step 10 of the Maximum-weight matching algorithm .	268
23c	Function in Step 11 of the Maximum-weight matching algorithm .	269
24	Fleury's algorithm for finding an eulerian circuit in a graph	298
25	An algorithm for finding an eulerian circuit in a digraph	310
26	The Nearest neighbour heuristic	336
27	An algorithm for computing a short hamiltonian cycle	340

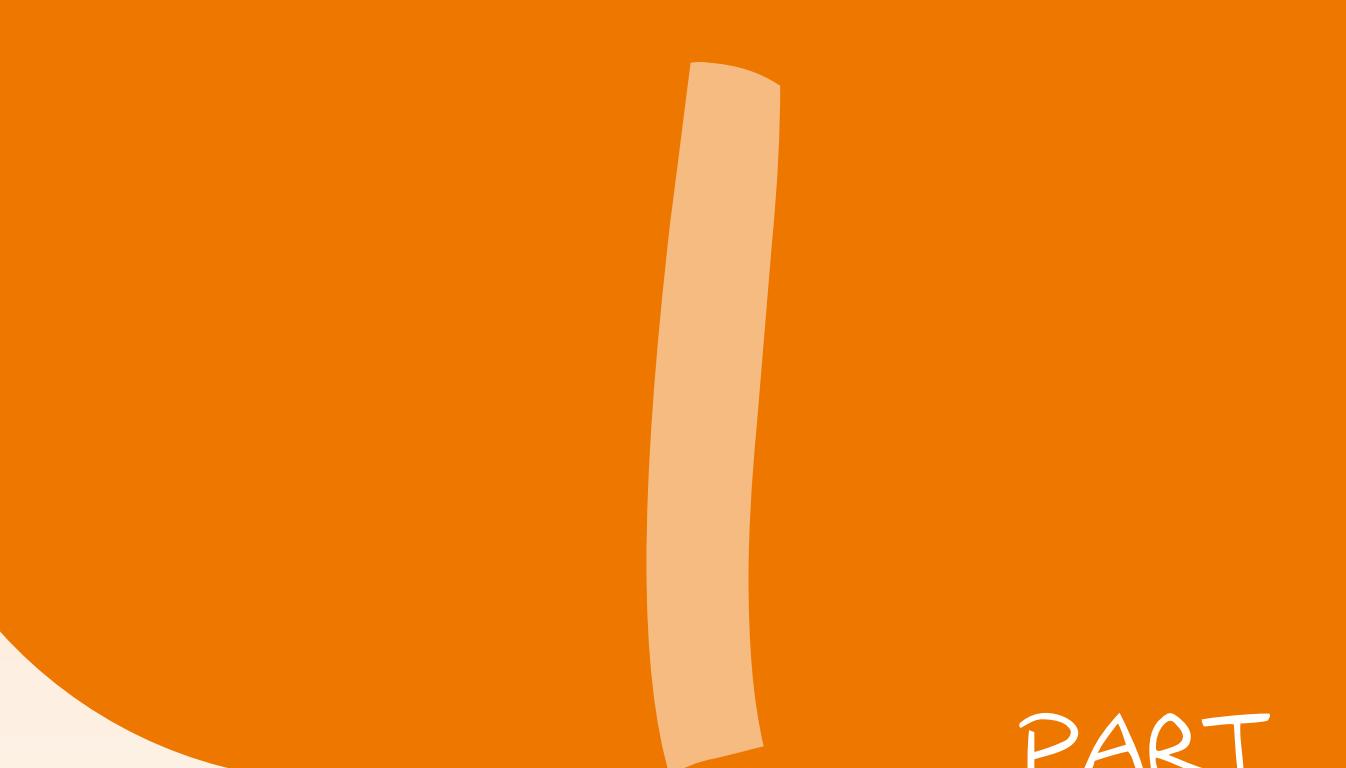
28	An algorithm for computing the blocks of a connected graph	364
29	An algorithm for computing the connectivity of a connected graph .	378
30	An algorithm for planarity testing	411
31	The Sequential colouring heuristic	467
32	The Welsh-Powell colouring heuristic	468
33	Brélaz's colouring heuristic	469
34	The Smallest-last colouring heuristic	471
35	Brown's algorithm for computing the chromatic number of a graph	474
36	An algorithm for computing the domination number of a tree	550

List of Biographical Notes

1	Claude Berge (1926–2002)	5
2	Øystein Ore (1899–1968)	8
3	Seifollah Hakimi (1932–2005)	12
4	Dénes König (1884–1944)	39
5	Édouard Lucas (1842–1891)	57
6	Paul Bachmann (1837–1920)	60
7	Edmund Landau (1877–1938)	62
8	Stephen Cook (1939–present)	65
9	George Boole (1815–1865)	66
10	Leonid Levin (1948–present)	70
11	Leonardo Fibonacci (c 1170–c 1250)	73
12	John von Neumann (1903–1957)	80
13	Euclid of Alexandria (third to fourth century BC)	84
14	Edsger Dijkstra (1930–2002)	92
15	Robert Floyd (1936–2001)	97
16	Joseph Kruskal (1928–2010)	122
17	Robert Prim (1921–present)	124
18	Edward Moore (1925–2003)	135
19	Stephen Hedetniemi (1939–present)	150
20	Lester Ford Jr (1927–2017)	177
21	Delbert Fulkerson (1924–1976)	178
22	Richard Karp (1935–present)	180
23	George Dantzig (1914–2005)	198
24	Philip Hall (1904–1982)	226
25	Jack Edmonds (1934–present)	253
26	Leonard Euler (1707–1783)	295
27	Carl Hierholzer (1840–1871)	296
28	Sir William Hamilton (1805–1865)	326
29	Paul Dirac (1902–1984)	328

30	John Bondy (1944–present)	330
31	Václav Chvátal (1946–present)	331
32	Lajos Pósa (1947–present)	333
33	Nicos Christofides (1942–2019)	342
34	Tibor Gallai (1912–1992)	344
35	Joseph Horton (dates unknown)	345
36	Martin Grötschel (1948–present)	347
37	Yoshiko Wakabayashi (1950–present)	348
38	Hassler Whitney (1907–1989)	369
39	Karl Menger (1902–1985)	372
40	Kazimierz Kuratowski (1896–1980)	401
41	William Tutte (1917–2002)	403
42	Carsten Thomassen (1948–present)	404
43	Klaus Wagner (1910–2000)	406
44	Paul Turán (1910–1976)	418
45	Kazimierz Zarankiewicz (1902–1959)	419
46	Daniel Kleitman (1934–present)	420
47	Lowell Beineke (1939–present)	423
48	Gary Chartrand (1936–present)	424
49	Richard Guy (1916–2020)	428
50	Neil Robertson (1938–present)	430
51	Paul Seymour (1950–present)	432
52	Francis Guthrie (1831–1899)	455
53	Augustus de Morgan (1806–1871)	456
54	Alfred Kempe (1849–1922)	457
55	Percy Heawood (1861–1955)	458
56	Kenneth Appel (1932–2013)	459
57	Wolfgang Haken (1928–present)	459
58	Julius Petersen (1839–1910)	460
59	Robin Thomas (1962–2020)	461
60	Rowland Brooks (1916–1993)	463
61	Daniel Brélaz (1950–present)	469
62	Vadim Vizing (1937–2017)	479
63	Robin Wilson (1943–present)	481
64	Herbert Robbins (1915–2001)	502
65	Oskar Perron (1880–1975)	509
66	Ferdinand Frobenius (1849–1917)	510
67	USR Murty (1940–present)	513
68	Ingo Schiermeyer (1960–present)	514
69	Marietjie Frick (1948–present)	515
70	Susan van Aardt (1966–present)	516
71	Alewyn Burger (1968–present)	517
72	Béla Bollobás (1943–present)	532
73	Ernie Cockayne (1940–present)	532

74	Bruce Reed (1962–present)	537
75	Alexandr Kostochka (1951–present)	538
76	Dieter Rautenbach (1972–present)	541
77	Douglas Rall (1949–present)	545
78	Sandi Klavžar (1962–present)	546
79	Wayne Goddard (1965–present)	547
80	Boštjan Brešar (1968–present)	555
81	Lutz Volkmann (1944–present)	559
82	Odile Favaron (1938–present)	562
83	Kieka Mynhardt (1953–present)	573
84	Anders Yeo (1970–present)	583
85	Stéphan Thomassé (1968–present)	587
86	Werner Gründlingh (1978–present)	601
87	David Erwin (1972–present)	610
88	Peter Dankelmann (1962–present)	613
89	Adriana Roux (1982–present)	614
90	Teresa Haynes (1953–present)	615
91	Peter Slater (1946–2016)	615
92	Frank Ramsey (1903–1930)	626
93	George Szekeres (1911–2005)	630
94	Brendan McKay (1951–present) Stanisław Radziszowski (1953–present)	632
95	Frank Harary (1921–2005)	639
96	Johannes Hattingh (1962–present)	642
97	Paul Erdős (1913–1996)	668
98	Alfréd Rényi (1921–1970)	669
99	Zoltán Füredi (1954–present)	674
100	Arthur Cayley (1821–1895)	686
101	Eugène Catalan (1814–1894)	692
102	George Pólya (1887–1985)	708
103	Joel Spencer (1946–present)	732
104	Noga Alon (1956–present)	734



PART

Topics in
Network Optimisation



An introduction to graphs

Contents

1.1	Introduction	1
1.2	What is a graph?	2
1.3	Special graphs	6
1.4	Operations on graphs	9
1.5	Degree sequences	11
1.6	Isomorphisms	14
1.7	Directed graphs	16
1.8	Representing a (di)graph on a computer	18
1.9	Multigraphs and pseudographs	19
	Exercises	20
	Computer exercises	25
	Projects	29
	Further reading	32

1.1 Introduction

Recent years have seen increased demand for the application of mathematics. *Graph theory* has proven to be particularly useful in this respect within a large number of rather diverse fields. The exciting and rapidly growing area of graph theory is rich in theoretical results as well as applications to real-world problems.

One of the primary activities of an applied mathematician is *modelling*. In a mathematical model, we represent or identify a real-life situation or problem with a mathematical system. One of the best-known examples of this representation is plane Euclidean geometry which provides useful results for describing small regions, such as measuring distances.

For graph theorists, the modelling process involves formulating a practical problem in such a way that it can be solved by techniques of graph theory. This is not always easy; the way in which the modelling process is carried out, and the degree to which the mathematical model accurately represents the original problem, varies considerably from problem to problem.

Many real-life situations can be described by means of a diagram consisting of a set of points with lines joining certain pairs of points. Loosely speaking (we shall

be more precise shortly), such a diagram is what we mean by a graph. Instances of graph applications abound. For example, the points might represent cities with lines representing direct flights between certain pairs of these cities in some airline system, or perhaps the lines represent pipelines between certain pairs of cities in an oil network. The points, on the other hand, might represent factories with lines representing communication links between them. Electrical networks, multiprocessor computers or switching circuits may also clearly be represented by graphs. In chemistry, graphs may sometimes be helpful in describing the structure of molecules. A chemical molecule may be represented as a graph whose “points” correspond to the atoms and whose “lines” correspond to the chemical bonds connecting them. The problem of counting the alkane (paraffin) hydrocarbons C_nH_{2n+2} is essentially a tree counting problem. (Trees are a class of graphs which we shall study in [Chapter 5](#).) For the sociologist, a graph may depict the manner in which people (or groups of people) exert influence over one another. Graphs may arise in several seemingly unrelated contexts, such as genetics, ecology, archeology, music, and the phasing of traffic lights. In order to investigate such relationships more deeply, we need to study the theory of graphs in greater detail.

Although various isolated sojourns were made into what is today considered the realm of graph theory prior to the twentieth century (such as [Leonard Euler](#)'s famous pursuit of a solution to the celebrated Königsberg Bridges Problem in 1736), graph theory is, perhaps surprisingly, a relatively young subdiscipline of mathematics. In fact, the first textbook on the subject, due to [Dénes Kónig](#), only appeared in 1936. Two later seminal texts by the French mathematician [Claude Berge](#) (in 1958) and the Norwegian mathematician [Øystein Ore](#) (in 1962) have placed the discipline of graph theory on a firm foundation. Today graph theory is an active field of research in mathematics.

1.2 What is a graph?

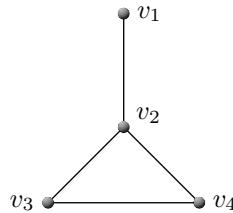
The common feature in all the preceding examples is that in each case we have a collection of “objects” (cities, people, factories, atoms, *etc.*) which are interrelated in some way. These ideas are easily abstracted to produce the concept of a graph.

A **graph** G is a finite nonempty set of objects, called **vertices** (the singular is **vertex**), together with a (possibly empty) set of unordered pairs of distinct vertices, called **edges**. The set of vertices of a graph G is called the **vertex set** of G , denoted by $V(G)$, and the set of edges is called the **edge set** of G , denoted by $E(G)$. The edge $e = \{u, v\}$ is said to **join** the vertices u and v . If $e = \{u, v\}$ (or simply $e = uv$) is an edge of G , then u and v are **adjacent vertices**, while u and e are **incident**, as are v and e . Furthermore, if e_1 and e_2 are distinct edges of G incident with a common vertex, then e_1 and e_2 are **adjacent edges**. It is convenient henceforth to denote an edge by uv or vu rather than by $\{u, v\}$. The cardinality of the vertex set of a graph G is called the **order** of G and is denoted by $n(G)$, or more simply by n when the graph under consideration is clear, while the cardinality of its edge set is the **size** of G , denoted by $m(G)$ or m . An (n, m) **graph** has order n and size m . The graph of order $n = 1$ is called the **trivial graph**. A **nontrivial graph** has at least two vertices.

We remark that graph theory terminology is not consistent. For example, some graph theory textbooks call a vertex a **node** or a **point**, and an edge a **line**.

It is customary to define or describe a graph by means of a diagram in which each vertex is represented by a point (which we draw as a dot or small circle) and each edge $e = uv$ is represented by a line segment or curve joining the points corresponding to u and v .

To illustrate these definitions, consider a diagram of the graph $G_{1.1}$ defined by the vertex set $V(G_{1.1}) = \{v_1, v_2, v_3, v_4\}$ and edge set $E(G_{1.1}) = \{v_1v_2, v_2v_3, v_2v_4, v_3v_4\}$, shown in [Figure 1.1](#). This graph has order 4 and size 4. Furthermore, observe that v_1 and v_2 are adjacent, but v_1 and v_3 are not adjacent. The vertex v_2 is incident to the edge v_2v_3 , but v_4 is not incident to v_2v_3 . The edges v_1v_2 and v_2v_3 are adjacent, but v_1v_2 and v_3v_4 are not adjacent.

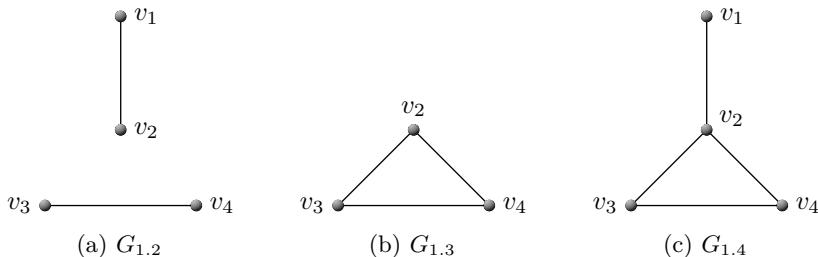


[Figure 1.1:](#) The graph $G_{1.1}$.

It is a common feature of mathematics that we study complicated objects by looking at simpler objects of the same type contained inside them, and these smaller objects are often indicated by the prefix “sub.” For example, we study subsets of sets, subsystems of systems, and so on. Consequently, in graph theory we adopt the following definition.

A **subgraph** of a graph G is a graph all of whose vertices belong to $V(G)$ and all of whose edges belong to $E(G)$. If H is a subgraph of G , then we write $H \subset G$. If a subgraph H of G contains all the vertices of G , then H is called a **spanning subgraph** of G . (Note that G is a subgraph of itself.)

For example, if $G_{1.1}$ is the graph shown in [Figure 1.1](#), then the graphs $G_{1.2}$, $G_{1.3}$ and $G_{1.4}$ shown in [Figure 1.2](#) are all subgraphs of $G_{1.1}$. However, only $G_{1.2}$ and $G_{1.4}$ are spanning subgraphs of $G_{1.1}$.



[Figure 1.2:](#) Subgraphs of the graph $G_{1.1}$.

An important type of subgraph of a graph that we shall encounter later in this book is an induced subgraph. If W is a nonempty subset of vertices of a graph G , then the subgraph $G[W]$ of G **induced** by W is the graph having vertex set W and whose edge set consists of all those edges of G incident with two vertices in W .

A subgraph H of G is called a **vertex-induced subgraph**, or simply an **induced subgraph**, of G if $H = G[W]$ for some subset W of $V(G)$. Hence, if H is an induced subgraph of G , then every edge of G incident with two vertices in $V(H)$ belongs to $E(H)$ (so two vertices are adjacent in H if and only if they are adjacent in G). Similarly, if F is a nonempty subset of edges of G , then the subgraph $G[F]$ of G **induced** by F is the graph whose vertex set consists of all those vertices of G incident with edges in F and whose edge set is F . A subgraph H of G is called an **edge-induced subgraph** of G if $H = G[F]$ for some subset F of $E(G)$.

For example, if $G_{1.1}$ is the graph shown in Figure 1.1, and $G_{1.2}$ and $G_{1.3}$ are the subgraphs of $G_{1.1}$ shown in Figure 1.2, then $G_{1.2} = G_{1.1}[F]$ and $G_{1.3} = G_{1.1}[W]$, where $F = \{v_1v_2, v_3v_4\}$ and $W = \{v_2, v_3, v_4\}$.

We have already introduced two parameters associated with a graph G , namely the order and the size of G . Now we define a collection of additional parameters associated with G . Let v be a vertex of a graph G . The **degree** of v is the number of edges of G incident with v . The degree of v is denoted by $d_G(v)$, or simply $d(v)$ if G is clear from the context. The **minimum degree** of G is the minimum degree among all the vertices of G and is denoted by $\delta(G)$, while the **maximum degree** of G is the maximum degree among all the vertices of G and is denoted by $\Delta(G)$.

In Figure 1.3, a graph $G_{1.5}$ is shown together with the degrees of its vertices. For this graph, we have $\delta(G_{1.5}) = 0$ and $\Delta(G_{1.5}) = 5$. A vertex is called **odd** or **even** depending on whether its degree is odd or even. A vertex of degree 0 in G is called an **isolated vertex** of G (while also being considered even) and a vertex of degree 1 is an **end-vertex** of G . A vertex adjacent to an end-vertex in G is called a **support vertex** of G .

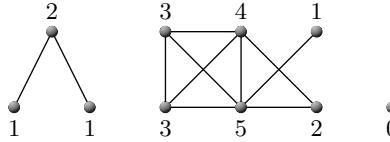


Figure 1.3: The degrees of the vertices of the graph $G_{1.5}$.

The **open neighbourhood**, denoted by $N_G(v)$ or simply $N(v)$ if G is clear from the context, of a vertex v of G is the set

$$N_G(v) = \{u \in V(G) \mid uv \in E(G)\}.$$

The **closed neighbourhood**, denoted by $N_G[v]$ or $N[v]$, of a vertex v of G is the set

$$N_G[v] = N_G(v) \cup \{v\}.$$

For every vertex v in a graph G we observe that $d_G(v) = |N_G(v)|$. The open neighbourhood of the vertex v_2 in the graph $G_{1.1}$, shown in Figure 1.1, is $N_{G_{1.1}}(v_2) = \{v_1, v_3, v_4\}$, while its closed neighbourhood is $N_{G_{1.1}}[v_2] = \{v_1, v_2, v_3, v_4\}$. Hence, the degree of v_2 is $d_{G_{1.1}}(v_2) = 3$.

Observe that for the graph $G_{1.5}$ shown in Figure 1.3, $n(G_{1.5}) = 10$ and $m(G_{1.5}) = 11$, while the sum of the degrees of its ten vertices is 22. The fact that this last number equals $2m(G_{1.5})$ is no mere coincidence, as we now show.

Theorem 1.1 *In any graph, the sum of all the vertex degrees is equal to twice the number of edges.*

Claude Berge was a French mathematician, recognised as one of the modern founders of combinatorics and graph theory. He obtained his PhD in mathematics (game theory) from the University of Paris in 1953. While still a doctoral student, he was appointed as research assistant at the French National Centre for Scientific Research. In 1957 he took up a position as professor at the Institute of Statistics at the University of Paris. From 1965 to 1967 he directed the International Computing Centre in Rome. He was also associated with the *Centre d'Analyse et de Mathématique Sociales* at the *École des hautes études en sciences sociales*. He held various visiting professor positions at pre-eminent universities, including Princeton University (1957), Pennsylvania State University (1968), and New York University (1985), and he was also a frequent visitor to the Indian Statistical Institute, Calcutta. He wrote five books, one on game theory (1957), one on graph theory and its applications (1958), one on topological spaces (1959), one on principles of combinatorics (1968) and one on hypergraphs (1970). Each of these works proved seminal and were translated into several languages. Apart from mathematics, Berge was also interested in chess, literature, sculpture, and art in general. He even co-authored a murder mystery based on a mathematical theorem, entitled *Who killed the Duke of Densmore?*



Biographic note 1: Claude Berge (1926–2002)

Proof Every edge is incident with two vertices. Hence, when the degrees of the vertices are summed, each edge is counted twice. ■

The above theorem is sometimes called the *Handshaking Lemma*. This name arises from the fact that a graph may be used to represent a group of people shaking hands at a party. In such a graph, the people are represented by the vertices, and an edge is included whenever the corresponding people have shaken hands. With this interpretation, the number of edges represents the total number of handshakes, the degree of a vertex is the number of hands shaken by the corresponding person, and the sum of the degrees is the total number of hands shaken. So the *Handshaking Lemma* states simply that the total number of hands shaken (the sum of the vertex degrees) is equal to twice the number of handshakes (edges) — the reason being, of course, that exactly two hands are involved in each handshake. There are some important consequences of the *Handshaking Lemma*, such as the following result.

Corollary 1.2 *In any graph, there is an even number of odd vertices.*

Proof Let G be an (n, m) graph. If G has no odd vertices, then the result follows immediately. Suppose, therefore, that G contains k (≥ 1) odd vertices v_1, \dots, v_k . If G contains even vertices as well, then denote these by v_{k+1}, \dots, v_n . By [Theo-](#)

rem 1.1,

$$\sum_{i=1}^k d(v_i) + \sum_{i=k+1}^n d(v_i) = 2m.$$

Since each of the numbers $d(v_{k+1}), \dots, d(v_n)$ is even, $\sum_{i=k+1}^n d(v_i)$ is even, and so we have that

$$\sum_{i=1}^k d(v_i) = 2m - \sum_{i=k+1}^n d(v_i)$$

is even. Each of the numbers $d(v_1), \dots, d(v_k)$ is, however, odd. Since the sum of an odd number of odd numbers is odd, it follows that k must be even; that is, G has an even number of odd vertices. If G has no even vertices, then we have $d(v_1) + \dots + d(v_k) = 2m$, from which we again conclude that k is even. ■

❖ The reader should now be able to attempt Exercises 1.1–1.6 and Project 1.1.

1.3 Special graphs

There are certain classes of graphs that occur so often that they deserve special mention and in some cases, special notation.

A graph G is called **r -regular** or **regular of degree r** if the degree of every vertex of G is r . A graph is **regular** if it is r -regular for some $r \in \mathbb{N} \cup \{0\}$. Some examples of graphs that are regular of degree $r \in \{1, 2, 3\}$ are shown in Figure 1.4. A 3-regular graph is sometimes called a **cubic graph**.

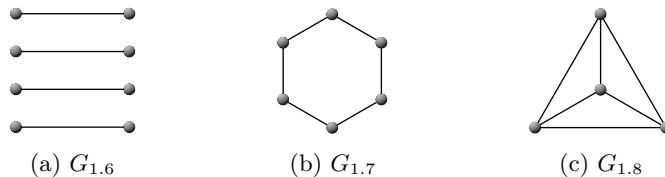


Figure 1.4: Regular graphs of degree (a) $r = 1$, (b) $r = 2$ and (c) $r = 3$.

A **complete graph** or **clique** is a graph in which every two distinct vertices are adjacent. A complete graph of order n is denoted by K_n and is sometimes called an **n -clique**. The degree of every vertex in a complete graph K_n is $n - 1$. Therefore, K_n is an $(n - 1)$ -regular graph. The complete graphs K_n , for $n \in [5]$, are illustrated in Figure 1.5(a)–(e).

An **empty graph** is a graph containing no edges. An empty graph of order n is denoted by \bar{K}_n . The empty graphs \bar{K}_n , for $n \in [5]$, are illustrated in Figure 1.5(f)–(j).

Of particular importance in applications are bipartite graphs. A **bipartite graph** is a graph whose vertex set can be partitioned into two subsets V_1 and V_2 (called **partite sets**) in such a way that each edge of the graph joins a vertex of V_1 to a vertex of V_2 . Hence, there is no adjacency amongst vertices of the same partite set. We can distinguish the vertices in V_1 from those in V_2 by drawing the former in black and the latter in white, so that each edge is incident with a black vertex and a white vertex. Some examples of bipartite graphs are shown in Figure 1.6.

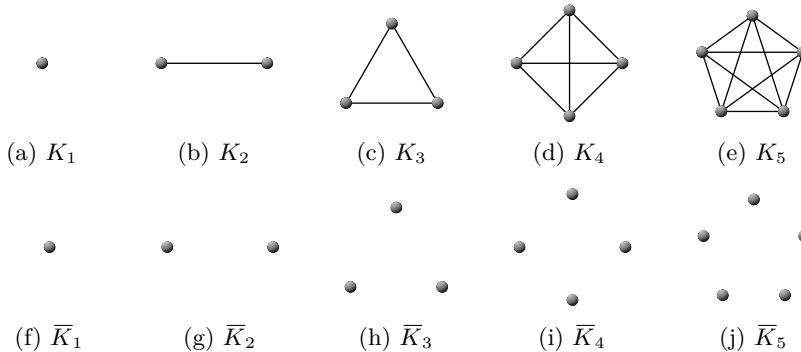


Figure 1.5: (a)–(e) The complete graphs K_n for $n \in [5]$. (f)–(j) The empty graphs \bar{K}_n for $n \in [5]$.

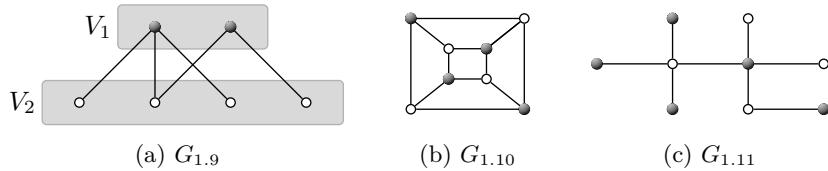


Figure 1.6: Bipartite graphs.

A **complete bipartite graph** is a bipartite graph with partite sets V_1 and V_2 having the added property that every vertex of V_1 is adjacent to every vertex of V_2 (so each black vertex is adjacent to each white vertex). If $|V_1| = r$ and $|V_2| = s$, then this graph is denoted by $K(r, s)$ or more commonly $K_{r,s}$. A complete bipartite graph of the form $K_{1,s}$ is called a **star**, while a complete bipartite graph of the form $K_{n,n}$ is called an **n -biclique**. Some examples of complete bipartite graphs are shown in Figure 1.7.

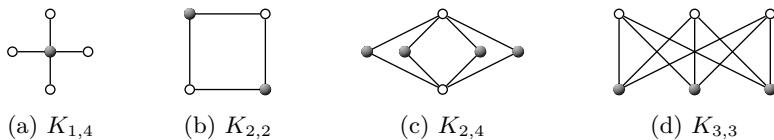
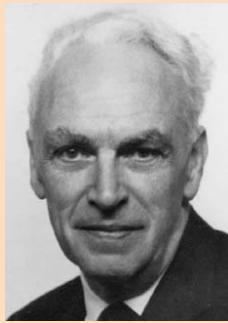


Figure 1.7: Complete bipartite graphs.

A graph G is **k -partite**, $k \geq 2$, if its vertex set $V(G)$ can be partitioned into k subsets V_1, \dots, V_k (called **partite sets**) in such a way that no two vertices of V_i are adjacent for all $i \in [k]$. A **complete k -partite graph** is a k -partite graph with partite sets V_1, \dots, V_k having the added property that every vertex of V_i is adjacent to every vertex not in V_i for all $i \in [k]$. If $|V_i| = n_i$, then this graph is denoted by $K(n_1, \dots, n_k)$ or by K_{n_1, \dots, n_k} . If $n_i = n$ for all $i \in [k]$, then the graph is called a **balanced complete k -partite graph** and is denoted by $K_{k \times n}$. A graph is a **(balanced) complete multipartite graph** if it is a (balanced) complete k -partite graph for some $k \in \mathbb{N} \setminus \{1\}$. A graphical representation of the complete multipartite

Øystein Ore was a Norwegian mathematician who obtained his PhD in mathematics (abstract algebra) from the University of Oslo in 1922. He also studied at Göttingen University, was a fellow at the Mittag-Leffler Institute in Sweden, and spent some time at the University of Paris. In 1925, he was appointed research assistant at the University of Oslo and in 1927 he moved to Yale University as an assistant professor of mathematics. He was promoted to associate professor in 1928 and to full professor in 1929. In 1931, he became a Sterling Professor (Yale's highest academic rank), a position he held until his retirement in 1968. Although known for his contributions to algebraic ring theory and Galois connections, he is best known for his work in graph theory. He was the author of the seminal book *The Theory of Graphs* in 1962 and also wrote eight other books. He was elected to the American Academy of Arts and Sciences and the Oslo Academy of Science. He was also a founder member of the Econometric Society. In gratitude for the services rendered to his native country during the second world war, he was decorated in 1947 with the Order of St Olav. Ore had a passion for art (painting and sculpture), was an avid collector of ancient maps, and spoke a number languages.



Biographic note 2: Øystein Ore (1899–1968)

graph $K_{2,5,3}$ is shown in Figure 1.8(a), while Figures 1.8(b) and (c) illustrate the difference between the balanced complete multipartite graphs $K_{2 \times 3}$ and $K_{3 \times 2}$.

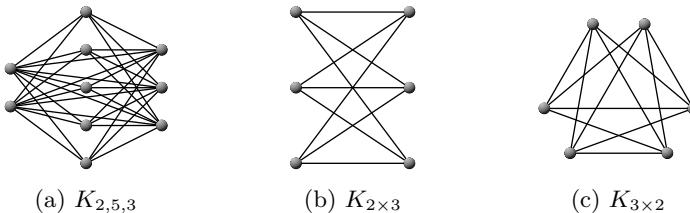


Figure 1.8: Some complete multipartite graphs.

Let $n > z \geq 1$ be integers. Also, let $1 \leq i_1, \dots, i_z \leq n-1$ be z distinct integers. Then the **circulant** $C_n\langle i_1, \dots, i_z \rangle$ of order n is a graph with vertex set

$$V(C_n\langle i_1, \dots, i_z \rangle) = \{v_0, \dots, v_{n-1}\}$$

and edge set

$$E(C_n\langle i_1, \dots, i_z \rangle) = \{v_\alpha v_{(\alpha+\beta) \pmod{n}} \mid \alpha \in \{0, \dots, n-1\} \text{ and } \beta \in \{i_1, \dots, i_z\}\},$$

where the notation $a \pmod{b}$ denotes the remainder when a is divided by b . The set $\{i_1, \dots, i_z\}$ is called the **connection set** of the circulant $C_n\langle i_1, \dots, i_z \rangle$. The inherent symmetry of the circulant $C_n\langle i_1, \dots, i_z \rangle$ may be visualised by arranging the vertex set $\{v_0, \dots, v_{n-1}\}$ on the edge of an imaginary circle and then joining every i_1 -th vertex, \dots , every i_z -th vertex on the circle by means of edges. This

is illustrated in Figures 1.9 and 1.10. Note that the complete graph K_n is the circulant $C_n\langle 1, \dots, \lceil \frac{n}{2} \rceil \rangle$.

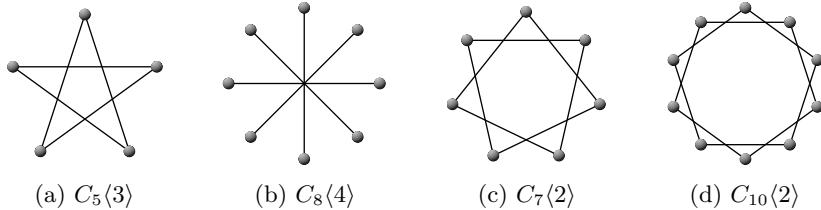


Figure 1.9: Elementary circulants: The circulants in (a), (c) and (d) are nonsingular, while the circulant in (b) is singular.

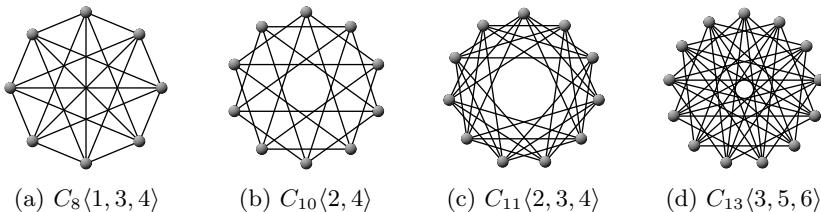


Figure 1.10: Composite circulants: The circulants in (b)–(d) are nonsingular, while the circulant in (a) is singular.

If $z = 1$ the circulant $C_n\langle i_1 \rangle$ is said to be **elementary**, else it is called **composite**. If n is even and $i = \frac{n}{2}$, then $C_n\langle i \rangle$ is called a **singular** (elementary) circulant. A composite circulant $C_n\langle i_1, \dots, i_z \rangle$ is said to be singular if $i_j = \frac{n}{2}$ for some $j \in [z]$. Figure 1.9(b) depicts the elementary, singular circulant $C_8\langle 4 \rangle$, while the composite, singular circulant $C_8\langle 1, 3, 4 \rangle$ is shown in Figure 1.10(a).

❖ The reader should now be able to attempt Exercises 1.7–1.13.

1.4 Operations on graphs

There are several operations one can perform on graphs in order to form new ones. In the following definitions, we assume that F and H are two graphs with disjoint vertex sets. The **union** $G = F \cup H$ of F and H has vertex set $V(G) = V(F) \cup V(H)$ and edge set $E(G) = E(F) \cup E(H)$. If a graph G consists of the union of k (≥ 2) disjoint copies of a graph H , then we write $G = kH$. The graph $2K_1 \cup 3K_2 \cup K_{1,3}$ is shown in Figure 1.11.

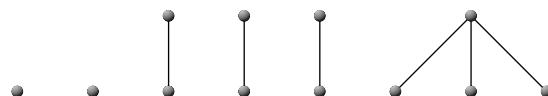


Figure 1.11: The union of the graphs $2K_1$, $3K_2$ and $K_{1,3}$.

The **join** $G = F + H$ of F and H has vertex set $V(G) = V(F) \cup V(H)$ and edge set $E(G) = E(F) \cup E(H) \cup \{uv \mid u \in V(F) \text{ and } v \in V(H)\}$. Using the join operation, we see that $K_{r,s} = \bar{K}_r + \bar{K}_s$. Another illustration of the join of two graphs, $G_{1.12}$ and $G_{1.13}$, is provided in Figure 1.12.

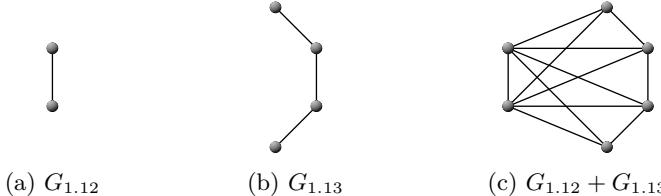


Figure 1.12: The join of two graphs.

The **cartesian product** $G = F \square H$ of F and H has vertex set $V(G) = V(F) \times V(H)$, where \times denotes the cartesian product between sets, and two vertices (u_1, u_2) and (v_1, v_2) are adjacent in G if and only if either

$$u_1 = v_1 \text{ and } u_2 v_2 \in E(H) \quad \text{or} \quad u_2 = v_2 \text{ and } u_1 v_1 \in E(F).$$

(Recall that the cartesian product $X \times Y$ of two sets X and Y is the set of all ordered pairs of the form (x, y) such that x belongs to X and y belongs to Y .) The cartesian product of $F \square H$ may be constructed by placing a copy of F at each vertex of H and adding the appropriate edges. For example, $K_2 \square K_3$ is shown graphically in Figure 1.13.

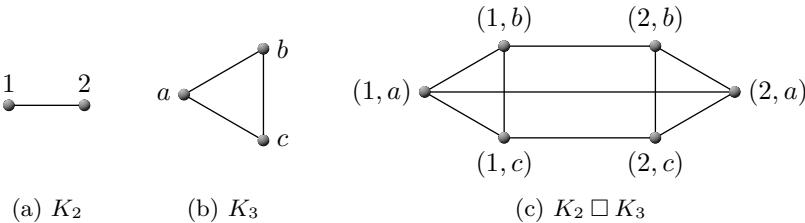


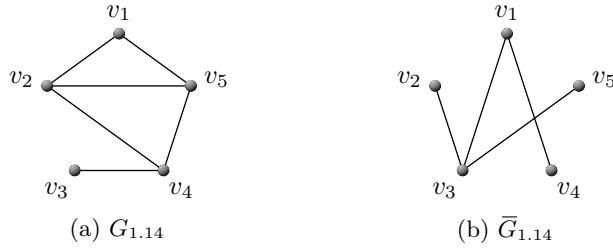
Figure 1.13: The cartesian product of two graphs.

If G is a graph, we form its **complement** \bar{G} by taking the vertex set of G and joining two vertices by an edge whenever they are not joined in G . For example, the complement of K_n is \bar{K}_n . Figure 1.14 contains an illustration of a graph $G_{1.14}$ and its complement $\bar{G}_{1.14}$. Note that if we take the complement of \bar{G} , we obtain the original graph G .

Let G_1, \dots, G_t be t nonempty graphs with the same vertex set V that are pairwise edge-disjoint. The **edge union**

$$G = \bigoplus_{i=1}^t G_i = G_1 \oplus \cdots \oplus G_t \tag{1.1}$$

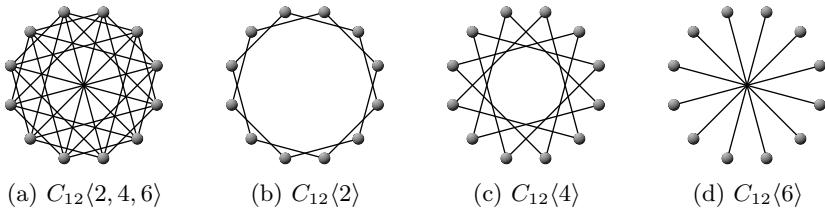
has vertex set $V(G) = V$ and edge set $E(G) = \bigcup_{i=1}^t E(G_i)$. Each graph G_i in (1.1) is called a **factor** of G , and G is said to be **factorable** into the factors G_1, \dots, G_t .

**Figure 1.14:** A graph and its complement.

The whole expression (1.1) is called a **factorisation** of G . A composite circulant may therefore be constructed from two or more elementary circulants, and may be written as the edge union

$$C_n \langle i_1, \dots, i_z \rangle = \bigoplus_{s=1}^z C_n \langle i_s \rangle$$

of elementary circulants. **Figure 1.15(a)** contains a graphical representation of the composite circulant $C_{12}\langle 2, 4, 6 \rangle$ together with its factors $C_{12}\langle 2 \rangle$, $C_{12}\langle 4 \rangle$ and $C_{12}\langle 6 \rangle$ in **Figure 1.15(b)–(d)**.

**Figure 1.15:** The graph (a) $C_{12}\langle 2, 4, 6 \rangle$ together with its factors (b) $C_{12}\langle 2 \rangle$, (c) $C_{12}\langle 4 \rangle$ and (d) $C_{12}\langle 6 \rangle$.

❖ The reader should now be able to attempt Exercises 1.14–1.15 and Project 1.2.

1.5 Degree sequences

It is often convenient to list the degrees of the vertices in a graph — this is usually done by writing them in nonincreasing order (that is, in decreasing order, but allowing “repeats” where necessary). The resulting list is called the **degree sequence** of a graph. Given a graph G , the degree sequence of G may easily be determined (we simply find the degree of each of its vertices and order these degrees appropriately). For example, the graph $G_{1.5}$ in **Figure 1.3** has degree sequence $s : 5, 4, 3, 3, 2, 2, 1, 1, 1, 0$. Several interesting questions about degree sequences now come to mind.

The first question one might think of is: “Can we reverse this process?” By this we mean: “Given a degree sequence s , can we determine a graph with s as degree sequence?” Perhaps a better question is: “Can we determine when a sequence of integers represents the degree sequence of some graph?” This leads us to the

following definition. A sequence of integers is called **graphical** if it is the degree sequence of some graph. A graph G with degree sequence \mathbf{s} is called a **realisation** of \mathbf{s} .

Let us begin with the question: “When is a sequence $\mathbf{s} : d_1, \dots, d_n$ of integers the degree sequence of some graph?” Certain conditions are clearly important. First, degrees are nonnegative integers, so $d_i \geq 0$ for all $i \in [n]$. Next, $d_i \leq n - 1$, because no vertex in a graph of order n can be adjacent to more than $n - 1$ other vertices. Furthermore, Theorem 1.1 tells us that the sum of the degrees of the vertices in any graph must be an even number, and so $\sum_{i=1}^n d_i$ is even. The above three conditions are all necessary for a sequence to be graphical, but these conditions are not sufficient. For example, the sequence 3, 3, 3, 1 satisfies all three of these necessary conditions, but is not graphical. A necessary and sufficient condition for a sequence to be graphical was found by Havel [8] in 1955 and later rediscovered by Hakimi [7] in 1962.

Seifollah Louis Hakimi was an American mathematician who was born in Meshed, Iran in 1932. He obtained his PhD in electrical engineering from the University of Illinois at Urbana-Champaign in 1959, where he was appointed as assistant professor of electrical engineering upon graduating. He also fulfilled the role of associate project director of the Circuit Theory Group of the University of Illinois from 1959 until 1961 when he was appointed as associate professor of electrical engineering at Northwestern University in Evanston, Illinois. He chaired the Department of Electrical Engineering from 1973 to 1978 at Northwestern University, where he later became an emeritus professor. His main research interests were in network and systems theory, but he is best known for characterising the degree sequences of undirected graphs, for formulating the *Steiner Tree Problem* on networks, and for his work on facility location problems on networks. He died on 23 June 2005.



Biographic note 3: Seifollah Hakimi (1932–2005)

Theorem 1.3 ([7, 8]) *A sequence $\mathbf{s} : d_1, \dots, d_n$ of nonnegative integers with $d_1 \geq \dots \geq d_n$, $n \geq 2$, is graphical if and only if the sequence $\mathbf{s}_1 : d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n$ is graphical.*

Proof Suppose that the sequence \mathbf{s}_1 is graphical. Let G_1 be a graph of order $n - 1$ with degree sequence \mathbf{s}_1 . Then the vertices of G_1 can be labelled as v_2, \dots, v_n in such a way that $d_{G_1}(v_i) = d_i - 1$ if $i \in \{2, \dots, d_1 + 1\}$ and $d_{G_1}(v_i) = d_i$ if $i \in \{d_1 + 2, \dots, n\}$. We can now construct a new graph G with degree sequence \mathbf{s} from G_1 by adding a new vertex v_1 and joining v_1 by means of an edge to each of v_2, \dots, v_{d_1+1} . Then the degree of v_1 is d_1 , and the degree of v_i is d_i for all $i \in \{2, \dots, n\}$. Thus, we have constructed a graph with degree sequence \mathbf{s} (*i.e.* \mathbf{s} is graphical).

We show next that if \mathbf{s} is graphical, then \mathbf{s}_1 is graphical. Assume that \mathbf{s} is a graphical sequence. Therefore, there are graphs of order n with degree sequence \mathbf{s} . Among all such graphs, let G be one such that $V(G) = \{v_1, \dots, v_n\}$,

where $d_G(v_i) = d_i$ for $i \in [n]$, and the sum of the degrees of the vertices adjacent with v_1 is maximum.

We show that in the graph G , the vertex v_1 must be adjacent to vertices having degrees d_2, \dots, d_{d_1+1} . If this is not the case, then there exist two vertices v_j and v_k with $d_j > d_k$ such that v_1 is adjacent to v_k , but not to v_j . Since the degree of v_j exceeds that of v_k , there must be some vertex v_ℓ that is adjacent to v_j , but not to v_k . Removing the edges v_1v_k and v_jv_ℓ and inserting the edges v_1v_j and v_kv_ℓ produces a new graph G' that also has degree sequence s . However, in G' the sum of the degrees of the vertices adjacent to v_1 is larger than that in G , contradicting our choice of G . This contradiction shows that our initial assumption (namely, that v_1 is *not* adjacent to vertices having degrees $d_2, d_3, \dots, d_{d_1+1}$) was false.

Therefore, as claimed, v_1 must be adjacent to vertices having degrees $d_2, d_3, \dots, d_{d_1+1}$. Hence, the graph obtained from G by removing v_1 , together with all the edges incident with v_1 , produces a graph with degree sequence s_1 (*i.e.* s_1 is graphical). ■

With the aid of [Theorem 1.3](#), we may now present the following procedural steps allowing us to determine whether a finite sequence of integers is graphical and answer the question posed earlier. Given a sequence of $n \in \mathbb{N}$ nonnegative integers, we do the following:

Step 1: If some integer in the sequence exceeds $n - 1$, then the sequence is not graphical.

Step 2: If all the integers in the sequence are 0, then the sequence is graphical.

Step 3: If the sequence contains a negative integer, then the sequence is not graphical.

Step 4: Reorder the sequence (if necessary) so that it is nonincreasing.

Step 5: Delete the first number, say t , from the sequence and subtract 1 from the next t numbers in the sequence to from a new sequence. Return to [Step 2](#).

To illustrate these steps, consider the sequence

$$s : 4, 4, 4, 3, 3, 2.$$

[Step 1](#) is satisfied, and we begin the loop of [Step 2](#) to [Step 5](#). The tests in [Step 2](#) and [Step 3](#) do not immediately halt us. Proceeding to [Step 4](#) and [Step 5](#), we obtain

$$s_1 = s'_1 : 3, 3, 2, 2, 2.$$

Continuing to apply [Step 4](#) and [Step 5](#), we have

$$\begin{aligned} s'_2 &: 2, 1, 1, 2 \\ s_2 &: 2, 2, 1, 1 \quad (\text{reordering}) \\ s'_3 &: 1, 0, 1 \\ s_3 &: 1, 1, 0 \quad (\text{reordering}) \\ s_4 &= s'_4 : 0, 0. \end{aligned}$$

This therefore shows that s is graphical, since $0, 0$ is the degree sequence of the empty graph \bar{K}_2 on two vertices. If we can observe that some sequence prior to s_4 is graphical, then we may deduce that s is graphical. For example, the sequence s_2 is easily seen to be graphical since it is the degree sequence of the graph $G_{1.15}$ in Figure 1.16(a). One may verify that each of the sequences s_1 and s is, in turn, graphical. To construct a graph with degree sequence s_1 , we proceed in reverse from s'_2 to s_1 , observing that a vertex should be added to $G_{1.15}$ so that it is adjacent to one vertex of degree 2 and two vertices of degree 1 in $G_{1.15}$. We thus obtain the graph $G_{1.16}$ with degree sequence s_1 (or s'_1) in Figure 1.16(b). Proceeding from s'_1 to s , we again add a new vertex, joining it to two vertices of degree 3 and two vertices of degree 2 in $G_{1.16}$. This yields the graph $G_{1.17}$ with degree sequence s , as shown in Figure 1.16(c).

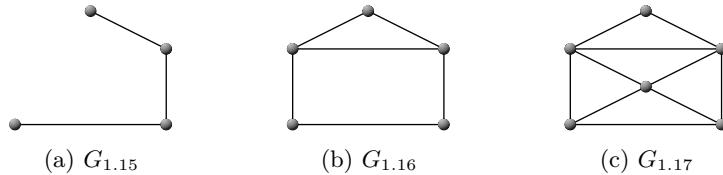


Figure 1.16: Construction of a graph $G_{1.17}$ with a given degree sequence.

It should be pointed out that the graph $G_{1.17}$ in Figure 1.16(c) is not the only graph with degree sequence $s : 4, 4, 4, 3, 3, 2$. Furthermore, the graphs $G_{1.18}$ and $G_{1.19}$ in Figure 1.17 both have the same degree sequence, namely $2, 2, 2, 2, 2, 2$. So degree sequences do not always provide enough information to describe a graph uniquely.

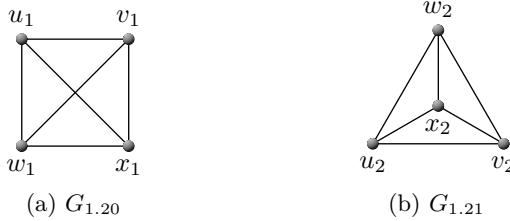


Figure 1.17: Two graphs with the same degree sequence $2, 2, 2, 2, 2, 2$.

- ❖ The reader should now be able to attempt Exercises 1.16–1.19.

1.6 Isomorphisms

In every area of mathematics, it is important to know whether two objects under investigation are the same (in some sense) or are different. For example, the numbers $\frac{2}{1}$ and $\frac{6}{3}$ are considered to be the same, or equal, but they are certainly not identical representations of the number 2. We now wish to determine what conditions must hold for two graphs to be considered the “same.” Intuitively, two graphs G and H are the same if it is possible to redraw one of them, say H , so that it appears identical to G . For example, the graphs $G_{1.20}$ and $G_{1.21}$ of Figure 1.18 have this property.

**Figure 1.18:** Isomorphic graphs.

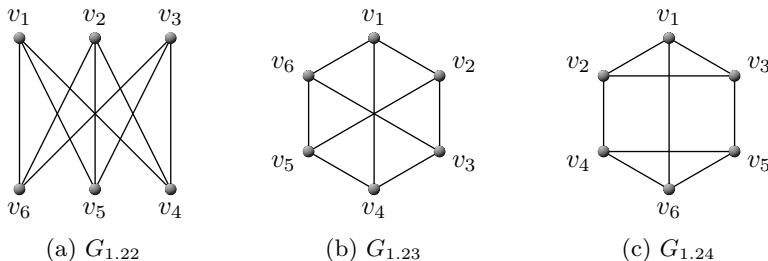
Two graphs often have the same structures, differing only in the way their vertices and edges are labelled or in the way they are drawn. To make this idea more exact, we introduce the concept of an isomorphism.

Two graphs G and H are **isomorphic** if H can be obtained from G by relabelling the vertices of G — that is, there is a one-to-one correspondence between the vertices of G and those of H such that an edge joins any pair of vertices in G if and only if an edge joins the corresponding pair of vertices in H . Hence G is isomorphic to H if there exists a one-to-one mapping ϕ , called an **isomorphism**, from $V(G)$ onto $V(H)$ such that ϕ preserves adjacency — that is, $uv \in E(G)$ if and only if $\phi(u)\phi(v) \in E(H)$. If G and H are isomorphic, then we write $G \cong H$. Two graphs are **nonisomorphic** if they are not isomorphic in which case we write $G \not\cong H$.

The graphs $G_{1.22}$ and $G_{1.23}$ in [Figure 1.19](#) are isomorphic (*i.e.* $G_{1.22} \cong G_{1.23}$). For example, the mapping $\phi : V(G_{1.22}) \mapsto V(G_{1.23})$ defined by

$$\phi(v_1) = v_1, \phi(v_2) = v_3, \phi(v_3) = v_5, \phi(v_4) = v_2, \phi(v_5) = v_4, \phi(v_6) = v_6$$

is an isomorphism. On the other hand, although each of $G_{1.22}$ and $G_{1.24}$ is a $(6, 9)$ graph, $G_{1.22} \not\cong G_{1.24}$. To see this, consider any one-to-one mapping ϕ from $V(G_{1.22})$ onto $V(G_{1.24})$. The vertices v_1, v_2 and v_3 of $G_{1.24}$ are pairwise adjacent, and ϕ must map three vertices of $G_{1.22}$ to v_1, v_2 and v_3 in some order. If ϕ is an isomorphism, then two vertices of $G_{1.22}$ are adjacent if and only if the two image vertices of $G_{1.24}$ under ϕ are adjacent. This implies that the three vertices of $G_{1.22}$ whose images are v_1, v_2 and v_3 also must be pairwise adjacent. However, $G_{1.22}$ does not contain three pairwise adjacent vertices. Hence there is no isomorphism from $V(G_{1.22})$ onto $V(G_{1.24})$.

**Figure 1.19:** Isomorphic and nonisomorphic graphs: $G_{1.22} \cong G_{1.23}$, $G_{1.22} \not\cong G_{1.24}$ and $G_{1.23} \not\cong G_{1.24}$.

Two graphs G and H are **identical**, denoted $G = H$, if $V(G) = V(H)$ and $E(G) = E(H)$. Clearly, two graphs may be isomorphic yet not identical. The graphs $G_{1.22}$ and $G_{1.23}$ of Figure 1.19 are not identical (even though $V(G_{1.22}) = V(G_{1.23})$ and $G_{1.22} \cong G_{1.23}$) since, for example, $v_1v_5 \in E(G_{1.22})$ and $v_1v_5 \notin E(G_{1.23})$.

An **automorphism** of a graph G is an isomorphism from G onto itself. Thus, an automorphism of G is a permutation of $V(G)$ that preserves adjacency (and nonadjacency). For example, the identity function on $V(G)$ is an automorphism of G . A graph G is **vertex-transitive** if, for every two vertices u and v of G , there exists an automorphism ϕ of G such that $\phi(u) = v$. Every vertex-transitive graph is therefore necessarily regular, but the converse is not true. The difference between regularity and vertex-transitivity is illustrated in Figure 1.20.

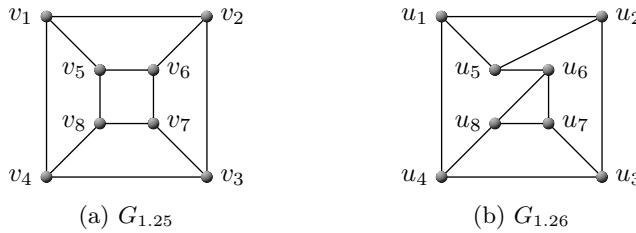


Figure 1.20: Two 3-regular graphs: (a) $G_{1.25}$ is a vertex-transitive graph, while (b) $G_{1.26}$ is not vertex-transitive, since the vertices u_3 and u_5 in $G_{1.26}$ are not indistinguishable.

❖ The reader should now be able to attempt Exercises 1.20–1.25.

1.7 Directed graphs

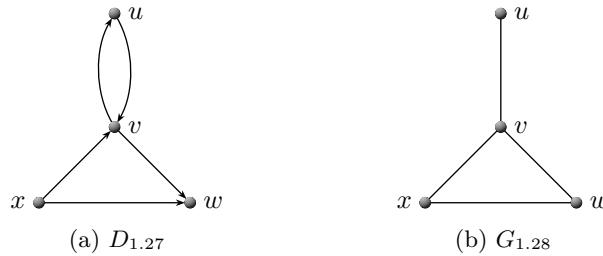
Although many problems lend themselves to a graph theoretic formulation, the concept of a graph is sometimes not quite adequate to deal with practical problems. When dealing with problems of traffic flow, for example, it is necessary to know which roads are one-way, and in which direction traffic is permitted. One may also want to deal with problems involving flow of information or water, transportation of some commodity, *etc.* To deal with such problems, one needs a graph in which every edge has been assigned a direction — a directed graph. The terminology for directed graphs is quite similar to that used for graphs.

A **digraph** (or **directed graph**) D is a finite nonempty set of objects, called **vertices**, together with a (possibly empty) set of *ordered* pairs of distinct vertices of D , called **arcs** (or **directed edges**). As with graphs, the **vertex set** of D is denoted by $V(D)$ and its **arc set** by $E(D)$. If D has vertex set V and arc set E , we write $D = (V, E)$. The cardinality of the vertex set of D is called the **order** of D and is denoted by $n(D)$, or simply by n , while the cardinality of its arc set is called its **size**, denoted by $m(D)$, or simply by m . As with graphs, digraphs may be represented by diagrams. The vertices of a digraph D are indicated by dots or small circles, and an arc (u, v) of D is represented by a curve or line segment directed by means of an arrow-head from vertex u to vertex v . Since (u, v) and (v, u) are distinct arcs, two vertices can be joined by two arcs if they have opposite direction. A

digraph D is **symmetric** if, whenever (u, v) is an arc of D , then (v, u) is also an arc of D , and D is called **asymmetric**, if whenever (u, v) is an arc of D , then (v, u) is *not* an arc of D .

With each digraph D , we associate a graph G (on the same vertex set) called the **underlying graph** of D that is obtained from D by deleting all directions from the arcs of D (equivalently, replacing each arc (u, v) by the edge uv) and deleting an edge from a pair of multiple edges if multiple edges should be produced.

A digraph $D_{1.27}$ with vertex set $V(D_{1.27}) = \{u, v, w, x\}$ and arc set $E(D_{1.27}) = \{(u, v), (v, u), (v, w), (x, v), (x, w)\}$ is shown in [Figure 1.21](#) along with $G_{1.28}$, the underlying graph of $D_{1.27}$. The digraph $D_{1.27}$ is neither symmetric nor asymmetric.



[Figure 1.21](#): A digraph and its underlying graph.

If (u, v) is an arc of a digraph D , then we say that u is **adjacent to** v , and v is **adjacent from** u . Furthermore, the arc (u, v) is **incident from** u and **incident to** v . The **outdegree** of a vertex v in D , denoted by $\text{od}_D(v)$ or merely $\text{od}(v)$, is the number of vertices adjacent from v , and the **indegree** of v , denoted by $\text{id}_D(v)$ or $\text{id}(v)$, is the number of vertices adjacent to v . The degree of v , denoted by $\text{d}_D(v)$ or $\text{d}(v)$, is defined as $\text{d}(v) = \text{od}(v) + \text{id}(v)$. The outdegrees, indegrees, and degrees of the vertices of the digraph $D_{1.27}$ in [Figure 1.21](#) are listed in [Table 1.1](#).

Vertex	Outdegree	Indegree	Degree
u	1	1	2
v	2	2	4
w	0	2	2
x	2	0	2

[Table 1.1](#): Outdegrees, indegrees and degrees of the vertices of $D_{1.27}$.

For a vertex v in a digraph D , we define the **out-neighbourhood** of v by the set $N_D^+(v) = \{u \in V(D) \mid (v, u) \in E(D)\}$. The **in-neighbourhood** of v is defined by $N_D^-(v) = \{u \in V(D) \mid (u, v) \in E(D)\}$. Hence, $\text{od}(v) = |N_D^+(v)|$, while $\text{id}(v) = |N_D^-(v)|$. The following fundamental result relates the sum of outdegrees to the sum of indegrees of a digraph.

Theorem 1.4 *If D is a digraph of size m with vertex set $V(D)$, then*

$$\sum_{v \in V(D)} \text{od}(v) = \sum_{v \in V(D)} \text{id}(v) = m.$$

Proof When the outdegrees of the vertices are summed, each arc is counted exactly once, since every arc is incident from exactly one vertex. Similarly, when the indegrees are summed, each arc is counted just once, since every arc is incident to exactly one vertex. ■

The other concepts introduced so far in this chapter for graphs generalise in a natural way for digraphs. For example, a digraph D' is called a **subdigraph** of a digraph D if $V(D') \subseteq V(D)$ and $E(D') \subseteq E(D)$. Furthermore, a digraph is **regular** if all its vertices have the same degree. In particular, if the degree of each vertex is $r \in \mathbb{N} \cup \{0\}$, then the digraph is **regular of degree r** or **r -regular**. Furthermore, two digraphs D_1 and D_2 are **isomorphic** if there exists a one-to-one mapping ϕ , called an **isomorphism**, from $V(D_1)$ onto $V(D_2)$ such that $(u, v) \in E(D_1)$ if and only if $(\phi(u), \phi(v)) \in E(D_2)$. If D_1 and D_2 are isomorphic, then we write $D_1 \cong D_2$. Two digraphs are **nonisomorphic** if they are not isomorphic.

❖ The reader should now be able to attempt Exercises 1.26–1.27.

1.8 Representing a (di)graph on a computer

A graph or digraph may be represented on a computer by means of a so-called adjacency matrix. The **adjacency matrix** of a graph G (digraph D , respectively) of order n , denoted by $A(G)$, is an $n \times n$, binary matrix whose (i, j) -th element is 1 if $v_i v_j \in E(G)$ ($(v_i, v_j) \in E(D)$, respectively), or zero otherwise. Note that the adjacency matrix of a graph is therefore symmetric, but that is not necessarily the case for a digraph. The adjacency matrix of the graph $G_{1.29}$ in Figure 1.22(a) is given in Figure 1.22(b), while that of the digraph $D_{1.30}$ in Figure 1.22(c) is given in Figure 1.22(d).

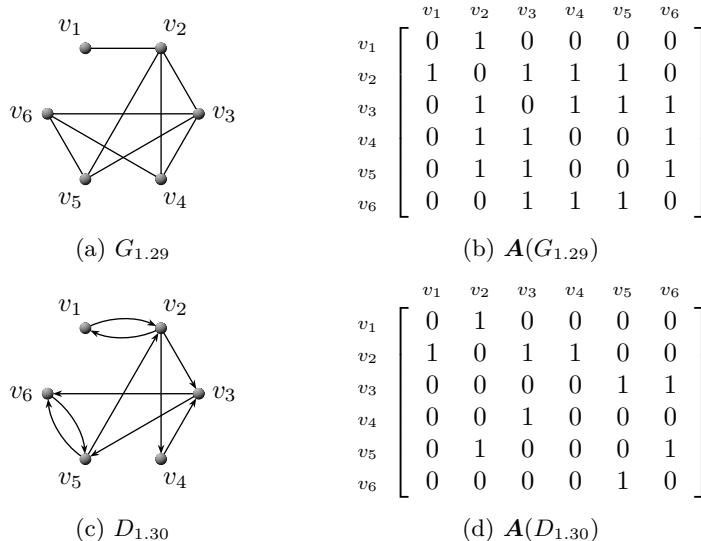


Figure 1.22: Adjacency matrix representations of the graph $G_{1.29}$ and digraph $D_{1.30}$.

An **adjacency list** (or list of lists) may also be used to represent a graph or a digraph on a computer. The adjacency list of the graph $G_{1.29}$ in Figure 1.22 is

$$\begin{aligned} L(G_{1.29}) = & \left\{ \underbrace{\{v_2\}}_{v_1}, \underbrace{\{v_1, v_3, v_4, v_5\}}_{v_2}, \underbrace{\{v_2, v_4, v_5, v_6\}}_{v_3}, \right. \\ & \left. \underbrace{\{v_2, v_3, v_6\}}_{v_4}, \underbrace{\{v_2, v_3, v_6\}}_{v_5}, \underbrace{\{v_3, v_4, v_5\}}_{v_6} \right\}, \end{aligned}$$

whilst that of the digraph $D_{1.30}$ is

$$L(D_{1.30}) = \left\{ \underbrace{\{v_2\}}_{v_1}, \underbrace{\{v_1, v_3, v_4\}}_{v_2}, \underbrace{\{v_5, v_6\}}_{v_3}, \underbrace{\{v_3\}}_{v_4}, \underbrace{\{v_2, v_6\}}_{v_5}, \underbrace{\{v_5\}}_{v_6} \right\}.$$

There are n sets in the adjacency list of an order n (di)graph. The i -th set corresponds to vertex v_i (indicated by the underbraces above) and lists the positions of nonzero entries in the adjacency matrix (*i.e.* the vertices in the open neighbourhood of v_i in the case of a graph, or the vertices in the out-neighbourhood of v_i in the case of a digraph). This method of representing a graph is efficient when there are only few edges in G , in which case the graph is called a **sparse** graph (as opposed to a **dense** graph).

The **adjacency table** of a graph (digraph, respectively) is a linked list-like data structure which is implemented in the form of a two-row array. The first row contains the labels of vertices in the graph (digraph, respectively), while the second row contains addresses in the table of adjacent vertices in the open neighbourhoods (out-neighbourhoods, respectively) of these vertices. The adjacency table of the graph $G_{1.29}$ in Figure 1.22(a) is given in Table 1.2, while that of the digraph $D_{1.30}$ is given in Table 1.3.

	v_1	v_2	v_3	v_4	v_5	v_6	7	8	9	10	11	12
Vertex label in $G_{1.29}$							v_2	v_1	v_3	v_4	v_5	v_2
Adjacency table index	7	8	12	16	19	22	0	9	10	11	0	13
	13	14	15	16	17	18	19	20	21	22	23	24
Vertex label in $G_{1.29}$	v_4	v_5	v_6	v_2	v_3	v_6	v_2	v_3	v_6	v_3	v_4	v_5
Adjacency table index	14	15	0	17	18	0	20	21	0	23	24	0

Table 1.2: Adjacency table for the graph $G_{1.29}$.

- ❖ The reader should now be able to attempt Exercise 1.28 and Project 1.3.

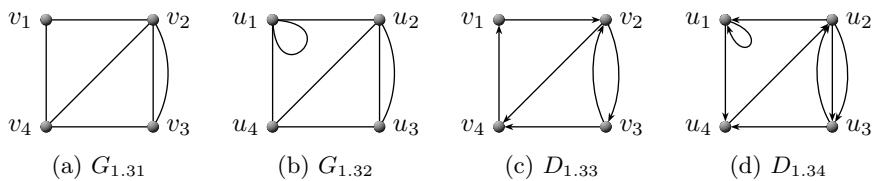
1.9 Multigraphs and pseudographs

If there is more than one edge between some pair of vertices in a graph, the graph is called a **multigraph**. Figure 1.23(a) contains an example of a multigraph, $G_{1.31}$. Two or more edges in a multigraph that join the same pair of vertices are called **parallel edges**. A **loop** is an edge that joins a vertex to itself. A graph containing loops (and/or parallel edges) is called a **pseudograph**. Therefore, multigraphs are

	v_1	v_2	v_3	v_4	v_5	v_6	7	8
Vertex label in $D_{1.30}$							v_2	v_1
Adjacency table index	7	8	11	13	14	16	0	9
	9	10	11	12	13	14	15	16
Vertex label in $D_{1.30}$	v_3	v_4	v_5	v_6	v_3	v_2	v_6	v_5
Adjacency table index	10	0	12	0	0	15	0	0

Table 1.3: Adjacency table for the digraph $D_{1.30}$.

also pseudographs, but the converse is not true. [Figure 1.23\(b\)](#) contains an example of a pseudograph, $G_{1.32}$. (We shall mostly, but not always, consider simple graphs in this book — that is, graphs without loops and with at most one edge between two vertices.)

**Figure 1.23:** Different types of graphs.

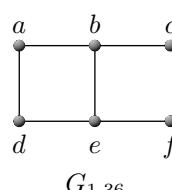
The notions of a multigraph and a pseudograph extend naturally to digraphs. A digraph may, however, have a pair of oppositely directed arcs between two vertices, yet not be a multidigraph. For example, the digraph $D_{1.33}$ in [Figure 1.23\(c\)](#) is not a multidigraph. An example of a pseudodigraph $D_{1.34}$ is shown in [Figure 1.23\(d\)](#).

- ❖ The reader should now be able to attempt [Exercise 1.29](#).

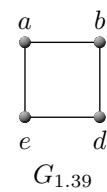
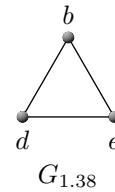
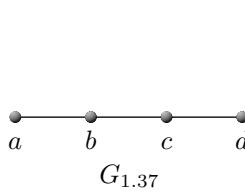
Exercises

1.1 Draw the graph $G_{1.35}$ with vertex set $V(G_{1.35}) = \{v_1, v_2, v_3, v_4, v_5\}$ and edge set $E(G_{1.35}) = \{v_1v_2, v_1v_4, v_1v_5, v_2v_3, v_3v_5, v_4v_5\}$.

1.2 Write down the vertex set $V(G_{1.36})$ and edge set $E(G_{1.36})$ of the graph $G_{1.36}$ below.



- 1.3 Which of the following graphs are subgraphs of the graph $G_{1.36}$ in Exercise 1.2?



- 1.4 Six university teams A, B, C, D, E and F belong to the same league and have played a number of matches: A has played C, D and E ; B has played C and E ; C has played A, B and D ; D has played A, C and E ; E has played A, B and D ; F has not yet played. Let $G_{1.40}$ be a graph with vertex set $V(G_{1.40}) = \{A, B, C, D, E, F\}$ and let two vertices of $G_{1.40}$ be adjacent if and only if the corresponding teams have played a match.

- (a) Write down $E(G_{1.40})$.
- (b) Draw the graph $G_{1.40}$.

- 1.5 What is the maximum possible size of a graph of

- (a) order 3;
- (b) order 4;
- (c) order 5;
- (d) order n , where n is a positive integer?

- 1.6 For each of the properties listed below, find a graph G that has the given property:

- (a) Every vertex of G is adjacent to two vertices and every edge of G is adjacent to two edges.
- (b) Every two vertices of G are adjacent and every two edges of G are adjacent.
- (c) Every vertex of G is incident with an edge, but no two edges of G are adjacent.

- 1.7 Draw the following graphs:

- (a) K_6 ;
- (b) $K_{4,4}$;
- (c) $K_{2,5}$.

- 1.8 What is the order and size of

- (a) the complete graph K_p ;
- (b) the complete bipartite graph $K_{r,s}$;
- (c) the complete k -partite graph K_{p_1, p_2, \dots, p_k} ;
- (d) the balanced complete k -partite graph $K_{k \times p}$?

- 1.9 Show that if G is an r -regular graph of odd order, then r is even.

- 1.10 Prove that if G is a regular bipartite graph with partite sets V_1 and V_2 , then $|V_1| = |V_2|$.

1.11 Draw the following circulants:

- (a) $C_8\langle 2 \rangle$;
- (b) $C_8\langle 3 \rangle$;
- (c) $C_9\langle 4 \rangle$;
- (d) $C_{12}\langle 1, 2, 3 \rangle$;
- (e) $C_{12}\langle 2, 10 \rangle$;
- (f) $C_{12}\langle 1, 2, 3, 4, 5, 6 \rangle$.

1.12 Let $G = C_n\langle i \rangle$ be an elementary circulant with $i \in \{1, 2, \dots, n-1\}$. Prove that

- (a) $G = C_n\langle n-i \rangle$;
- (b) $E(G) \cap E(C_n\langle j \rangle) = \emptyset$ for $i, j \in \{1, 2, \dots, \lfloor \frac{n}{2} \rfloor\}$ and $i \neq j$;
- (c) if G is singular, then $d(v) = 1$ for every $v \in V(G)$ and $m(G) = \frac{n}{2}$;
- (d) if G is nonsingular, then $d(v) = 2$ for every $v \in V(G)$ and $m(G) = n$.

1.13 Let $G = C_n\langle 1, 2, \dots, z \rangle$ be a composite circulant with $z \leq \lfloor \frac{n}{2} \rfloor$. Show that

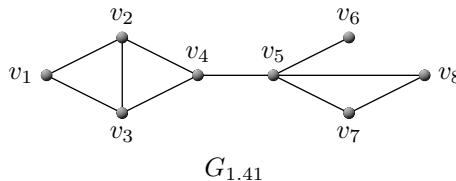
- (a) if G is singular, then $d(v) = 2z - 1$ for every $v \in V(G)$ and $m(G) = (z-1)n + \frac{n}{2}$;
- (b) if G is nonsingular, then $d(v) = 2z$ for every $v \in V(G)$ and $m(G) = zn$.

1.14 Draw the following graphs:

- (a) $3K_3$;
- (b) $K_{2,2} + \bar{K}_2$;
- (c) $K_{1,2} \square K_3$;
- (d) $\bar{K}_{2,3}$.

1.15 For $i \in \{1, 2\}$, let G_i be an (n_i, m_i) graph. Determine the order and size of $G_1 \square G_2$ in terms of n_1, n_2, m_1 and m_2 .

1.16 For the graph $G_{1.41}$ below, write down the degrees of all the vertices and its degree sequence. Verify the [Handshaking Lemma \(Theorem 1.1\)](#) for the graph.



1.17 Suppose we know the degrees of the vertices of a graph G . Is it possible to determine the order and size of G ? Explain.

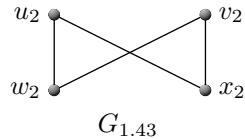
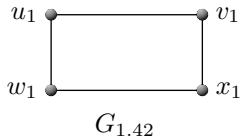
1.18 Determine whether the following sequences are graphical. If so, construct a graph with the appropriate degree sequence in each case.

- (a) 5, 5, 5, 3, 3, 2, 2, 2, 2;
- (b) 4, 4, 3, 2, 1, 0;
- (c) 3, 3, 2, 2, 2, 1, 1;

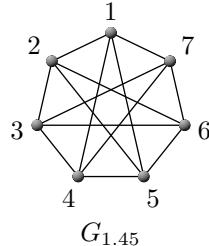
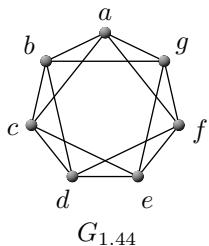
(d) 7, 4, 3, 3, 2, 2, 2, 1, 1, 1.

1.19 Show that the sequence d_1, d_2, \dots, d_n is graphical if and only if the sequence $n - d_1 - 1, n - d_2 - 1, \dots, n - d_p - 1$ is graphical. (Hint: Consider a graph and its complement.)

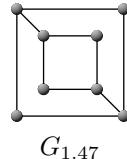
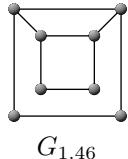
1.20 Show that the graphs $G_{1.42}$ and $G_{1.43}$ below are isomorphic by proving the existence of an isomorphism between $G_{1.42}$ and $G_{1.43}$.



1.21 Show that the graphs $G_{1.44}$ and $G_{1.45}$ below are isomorphic.



1.22 Are the graphs $G_{1.46}$ and $G_{1.47}$ below isomorphic? Motivate.



1.23 Determine all pairwise nonisomorphic graphs of

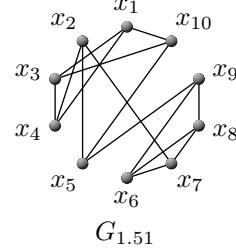
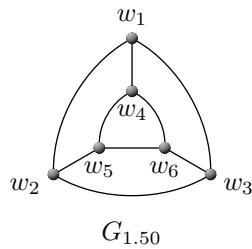
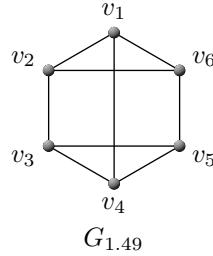
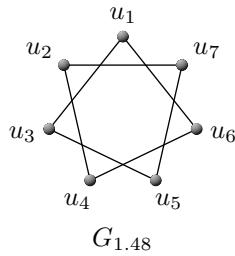
- (a) order 3;
- (b) order 4.

(A set of graphs is pairwise nonisomorphic if no two graphs in the set are isomorphic.)

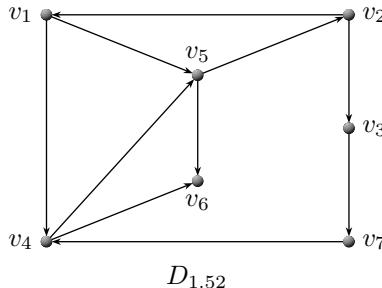
1.24 Classify each of the following statements as true or false and motivate your answer in each case:

- (a) If G and H are isomorphic graphs, then they have the same order and size.
- (b) If G and H have the same order and size, then they are isomorphic.
- (c) If G and H are isomorphic graphs, then they have the same degree sequence.
- (d) If G and H have the same degree sequence, then they are isomorphic.

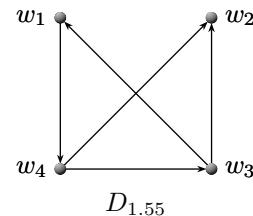
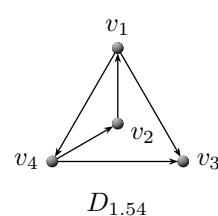
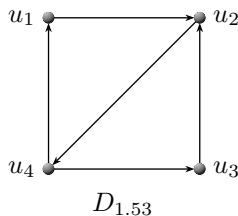
- 1.25 Which of the graphs $G_{1.48}$, $G_{1.49}$, $G_{1.50}$ and $G_{1.51}$ below are vertex-transitive?



- 1.26 Write down the in-neighbourhoods and out-neighbourhoods of the vertices v_1, \dots, v_7 of the digraph $D_{1.52}$ below. Also write down, as a consequence, the indegree and the outdegree of each vertex of the graph.



- 1.27 Consider the digraphs $D_{1.53}$, $D_{1.54}$ and $D_{1.55}$ below.



- Are $D_{1.53}$ and $D_{1.54}$ isomorphic? If so, produce an isomorphism. If not, motivate why not.
- Are $D_{1.53}$ and $D_{1.55}$ isomorphic? If so, produce an isomorphism. If not, motivate why not.
- Are $D_{1.54}$ and $D_{1.55}$ isomorphic? If so, produce an isomorphism. If not, motivate why not.

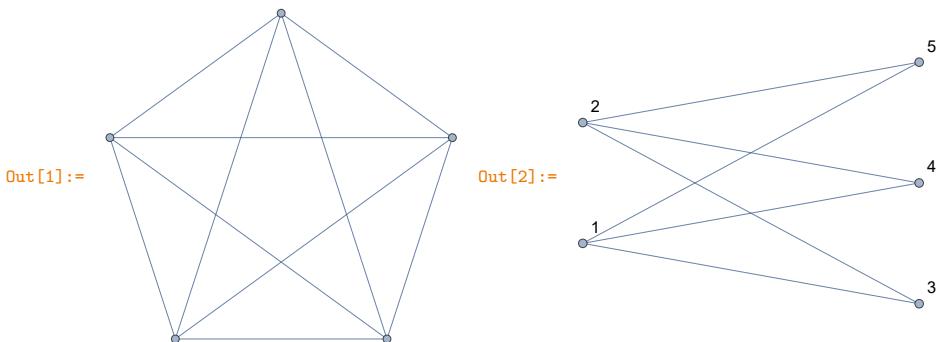
- 1.28 Find the following computer representations for the graphs $G_{1.48}$, $G_{1.49}$ and the digraph $D_{1.52}$:
- the adjacency matrix;
 - the adjacency list;
 - the adjacency table.
- 1.29 Draw the graphs whose vertex sets and edge sets are given, and decide whether each graph is a multigraph or a pseudograph:
- $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$,
 $E(G) = \{v_1v_2, v_2v_3, v_3v_4, v_3v_5, v_4v_5\}$;
 - $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$,
 $E(G) = \{v_1v_1, v_1v_2, v_1v_5, v_2v_5, v_3v_4, v_3v_4, v_4v_5\}$.

Computer exercises

The Wolfram computer package **MATHEMATICA** may be used to perform a large variety of computations involving graphs. This section contains descriptions of how to generate graphs and perform simple graph operations in **MATHEMATICA** (Version 10 or later). **MATHEMATICA** is case sensitive and commands are executed upon pressing SHIFT ENTER. A visual representation of a complete graph may be obtained by executing the command `CompleteGraph[G]`, while a visual representation of a complete k -partite graph with partite sets of cardinalities n_1, n_2, \dots, n_k may be generated by the command `CompleteGraph[{n1, n2, ..., nk}]`. The command option `VertexLabels -> "Name"` may be included in either of these commands in order to label the vertices of the graph. For example, the commands

```
In[1]:= K5 = CompleteGraph[5]
In[2]:= K23 = CompleteGraph[{2, 3}, VertexLabels -> "Name"]
```

produce the output:



A user-defined graph may be built by specifying its adjacency matrix. A matrix in **MATHEMATICA** is represented as a comma-separated list of lists. Each list represents a row in the adjacency matrix and lists are delimited by means of curly brackets. For example, the adjacency matrix of the graph in Figure 1.1 is given by $\{\{0, 1, 0, 0\}, \{1, 0, 1, 1\}, \{0, 1, 0, 1\}, \{0, 1, 1, 0\}\}$. The standard array form of a matrix may be achieved via the command `MatrixForm[•]`. Semi-colons are used to suppress output. For example, the commands

```
In[3]:= A = {{0, 1, 0, 0}, {1, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0}};
In[4]:= MatrixForm[A]
```

produce the output:

```
Out[4]:= 
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

```

The command `AdjacencyGraph[A]` may be used to obtain the graph with associated adjacency matrix A , whilst the command option `VertexLabels -> "Name"` may again be included to produce a vertex-labelled graph. For example, the command

```
In[5]:= G = AdjacencyGraph[A, VertexLabels -> "Name"]
```

produces the output:

```
Out[6]:= SparseArray[ Specified elements: 12 Dimensions: {5, 5}]
```

The command `MatrixForm[A]` may be used to convert a sparse array data structure A to the standard matrix form, while the command `Normal[A]` produces the sparse array A in standard **MATHEMATICA** matrix format. For example, the commands

```
In[7]:= MatrixForm[m]
In[8]:= Normal[m]
```

produce the output:

```
Out[7]:= 
$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

```

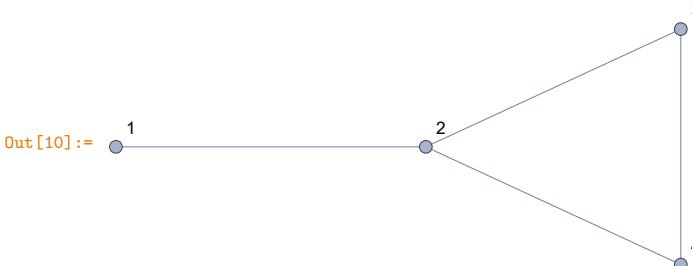
```
Out[8]:= {{0, 0, 1, 1, 1}, {0, 0, 1, 1, 1}, {1, 1, 0, 0, 0}, {1, 1, 0, 0, 0}, {1, 1, 0, 0, 0}}
```

A list of the edges in a graph G may be produced by means of the command `EdgeList[G]`. Conversely, a graph may be specified in terms of its edges in list form. For example, the commands

```
In[9]:= EdgeList[G]
In[10]:= Graph[{1 <-> 2, 2 <-> 3, 2 <-> 4, 3 <-> 4}, VertexLabels -> "Name"]
```

produce the output:

```
Out[9]:= {1 ↔ 2, 2 ↔ 3, 2 ↔ 4, 3 ↔ 4}
```



The symbol “ \leftrightarrow ” may be produced by pressing the keyboard sequence ESCAPE, U, E, ESCAPE. The abbreviation “UE” here stands for “undirected edge.” Alternatively, the undirected edge $i \leftrightarrow j$ may be produced by means of the command `UndirectedEdge[i, j]`.

For example, the command

```
In[11]:= UndirectedEdge[1, 2]
```

produces the output:

```
Out[11]:= 1 ↔ 2
```

The order and size of a graph G may be computed by means of the commands `VertexCount` [G] and `EdgeCount` [G], respectively. The degree sequence of G may furthermore be produced by executing the command `VertexDegree` [G]. For example, the commands

```
In[12]:= VertexCount[G]
In[13]:= EdgeCount[G]
In[14]:= VertexDegree[G]
```

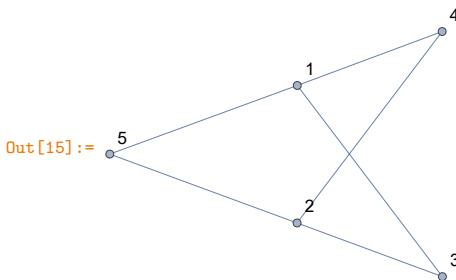
produce the output:

```
Out[12]:= 4
Out[13]:= 4
Out[14]:= {1, 3, 2, 2}
```

An arbitrary graph realising a given degree sequence d_1, d_2, \dots, d_n may be produced by embedding the command `DegreeGraphDistribution` [d_1, d_2, \dots, d_n] within the command `RandomGraph` [\bullet]. For example, the command

```
In[15]:= RandomGraph[DegreeGraphDistribution[{3, 3, 2, 2, 2}], VertexLabels -> "Name"]
```

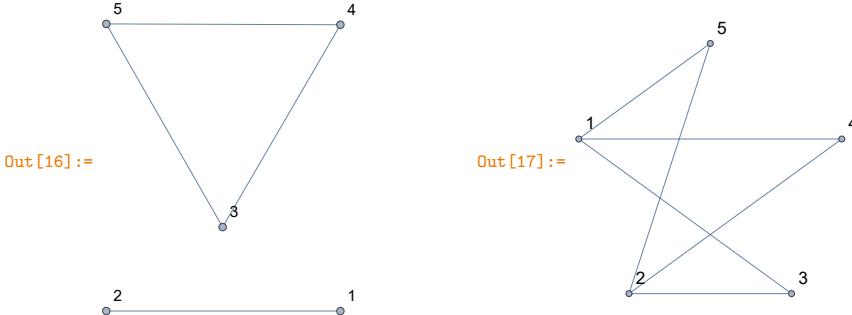
may produce the output:



The components G_1, G_2, \dots, G_k of a disconnected graph may also be specified separately within the command `GraphDisjointUnion` [G_1, G_2, \dots, G_k]. Moreover, the complement of a graph may be produced by means of the command `GraphComplement` [\bullet]. For example, the commands

```
In[16]:= J = GraphDisjointUnion[CompleteGraph[2], CompleteGraph[3], VertexLabels -> "Name"]
In[17]:= GraphComplement[J, VertexLabels -> "Name"]
```

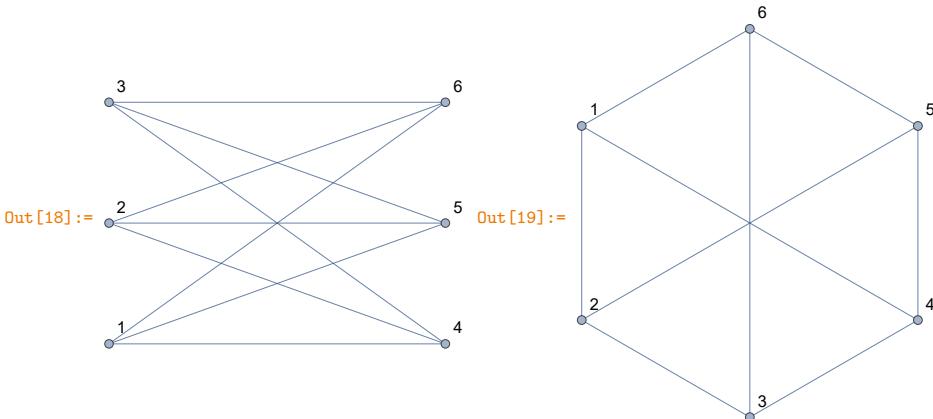
produce the output:



A circulant graph of order n with connection set $\{a_1, a_2, \dots, a_k\}$ may be generated by means of the command `CirculantGraph[n, {a1, a2, ..., ak}]`. The command `IsomorphicGraphQ[G1, G2]` produces the boolean value `True` if two specified graphs G_1 and G_2 are isomorphic, or the boolean value `False` otherwise. If G_1 and G_2 are isomorphic, the command `FindGraphIsomorphism[G1, G2]` produces an isomorphism from G_1 to G_2 . For example, the commands

```
In[18]:= K = CompleteGraph[{3, 3}, VertexLabels -> "Name"]
In[19]:= L = CirculantGraph[6, {1, 3}, VertexLabels -> "Name"]
In[20]:= IsomorphicGraphQ[K, L]
In[21]:= FindGraphIsomorphism[K, L]
```

produce the output:

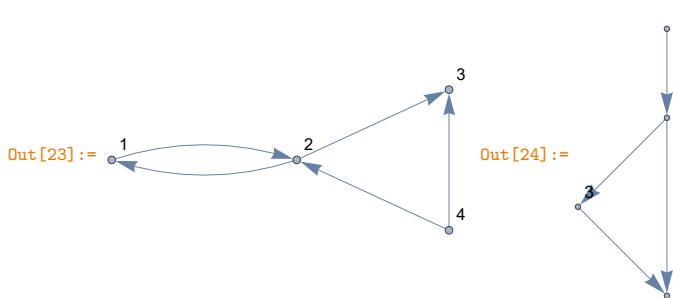


```
Out[20]:= True
Out[21]:= {<|1 -> 1, 2 -> 3, 3 -> 5, 4 -> 2, 5 -> 4, 6 -> 6|>}
```

If a nonsymmetric adjacency matrix A is specified, a directed graph will automatically be generated by the command `AdjacencyGraph[A]`. Moreover, the command `DirectedGraphQ[G]` will yield the boolean value `True` if G is interpreted by **MATHEMATICA** to be a directed graph, or the boolean value `False` otherwise. For example, the commands

```
In[22]:= m = {{0, 1, 0, 0}, {1, 0, 1, 0}, {0, 0, 0, 0}, {0, 1, 1, 0}};
In[23]:= G = AdjacencyGraph[m, VertexLabels -> "Name"]
In[24]:= Graph[{1 -> 2, 2 -> 3, 2 -> 4, 3 -> 4}, VertexLabels -> "Name"]
In[25]:= DirectedGraphQ[G]
```

produce the output:



Out[25] := True

Finally, the commands `VertexInDegree[G]` and `VertexOutDegree[G]` may be used to produce the in- and out-degree sequences of a directed graph G , respectively. For example, the commands

In[26] := `VertexInDegree[G]`
In[27] := `VertexOutDegree[G]`

produce the output:

Out[26] := {1, 2, 2, 0}
Out[27] := {1, 2, 0, 2}

- ❖ The reader should now be in a position to repeat the following exercises using **MATHEMATICA** (without a pen and paper): Exercises 1.1, 1.7, 1.11, 1.16, 1.18, 1.20, 1.21, 1.22, 1.27 and 1.28(a).

Projects

This section contains three projects. These projects generally require more thought and time, and are often of a more applied nature, than regular exercises. The first project of this section centres around the notion of an acquaintance graph and how such a graph may be used to extract powerful information about social interactions that is otherwise often cumbersome to establish. The focus in the second project falls on the activity of round-robin sports scheduling and how factorisations of complete graphs may lead to the establishment of efficient, clash-free round-robin schedules. Whereas the first two projects may be carried out by hand, the third project is essentially a computer project. In this last project, the reader is asked to set the stage for follow-up projects of later chapters by representing and storing an underlying graph model of a city street map in various forms on a computer.

Project 1.1: Acquaintances

Consider an *acquaintance graph* for a group of $n \geq 2$ people in which the vertices represent the people and in which two vertices are adjacent if the corresponding people are acquaintances.

Tasks

1. Use the notion of an acquaintance graph to show that in any group of at least two people there must be at least two who have exactly the same number of acquaintances in the group.
2. Suppose you and your partner attended a party with three other couples. Several handshakes took place at the party. No one shook hands with himself (or herself) or his (or her) partner, and no one shook hands with the same person more than once. After all the handshaking had been completed, suppose you asked each person, including your partner, how many hands he or she had shaken. Each person gave a different answer. Use the notion of an acquaintance graph to answer the following questions:
 - (a) How many hands did you shake?
 - (b) How many hands did your partner shake?

Project 1.2: Scheduling a round-robin chess tournament

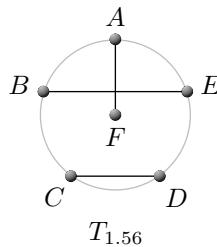
Suppose six people A, B, C, D, E and F are to play a round-robin chess tournament (*i.e.* each person has to compete *exactly once* with each other person). Suppose further that no participant may play more than one chess match per day.

Tasks

1. Determine the total number of chess matches that have to be scheduled during the tournament. Motivate your answer.
2. Determine the maximum number of chess matches that may be played on any day of the tournament. Motivate your answer.
3. Let Ξ denote the shortest possible duration of the tournament (in days). Determine Ξ and motivate your answer.
4. Spend a few minutes attempting a brute force construction of a tournament schedule of minimum duration in the following tabular form:

	Day 1	Day 2	...	Day Ξ
Match 1	A vs. F
Match 2	B vs. E
Match 3	C vs. D

Now that you are convinced that this scheduling problem is nontrivial, let us utilise the power of graph theory... The tournament may be modelled by a complete graph of order 6, called the *tournament graph*, in which the edge between two vertices denotes the chess match to be played by the corresponding two participants. The graph $T_{1.56}$ below is a 1-regular factor of the tournament graph corresponding to the three matches scheduled in the table above for Day 1 of the tournament.



5. Find a factorisation of the tournament graph comprising Ξ 1-regular factors.
6. Use your factorisation in [Task 5](#) to complete the partial tabular tournament schedule given in [Task 4](#).

Project 1.3: Graph representation of a street map

Consider the street map of the *Tshwane* (Pretoria) suburb of *Mountain View* adjacent to the Magaliesberg Nature Reserve in South Africa, shown in [Figure 1.24](#).

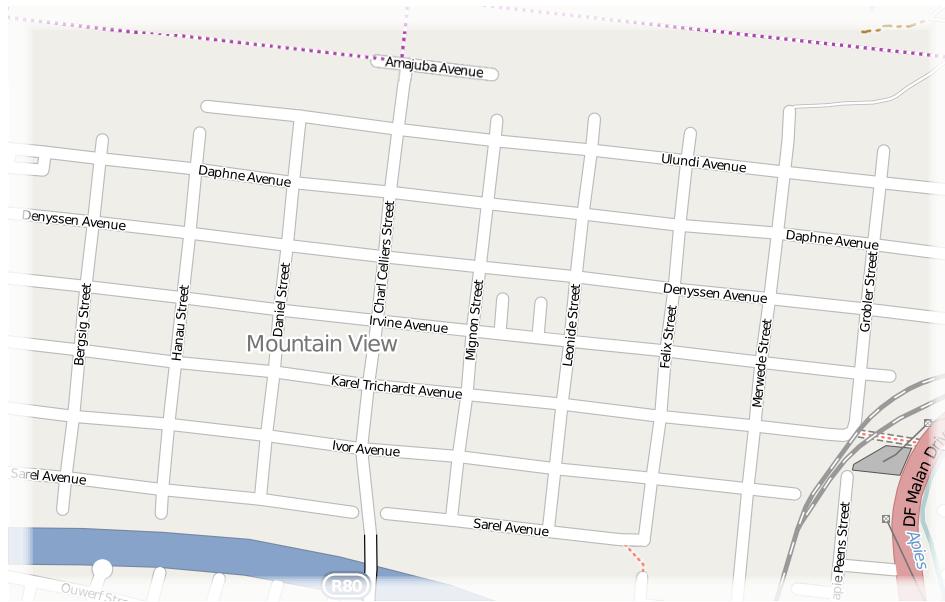


Figure 1.24: A street map of a section of the *Tshwane* suburb of *Mountain View* adjacent to the Magaliesberg Nature Reserve in South Africa.

A graph representation of this street map may be obtained by denoting each street intersection or dead-end by a vertex and by representing the interconnecting streets by means of edges. A graph $G_{1.57}$ of order 76 and size 116 results, as shown in [Figure 1.25](#).

Tasks

1. Find the *adjacency matrix* representation of $G_{1.57}$ (and store it electronically for use in projects of later chapters).

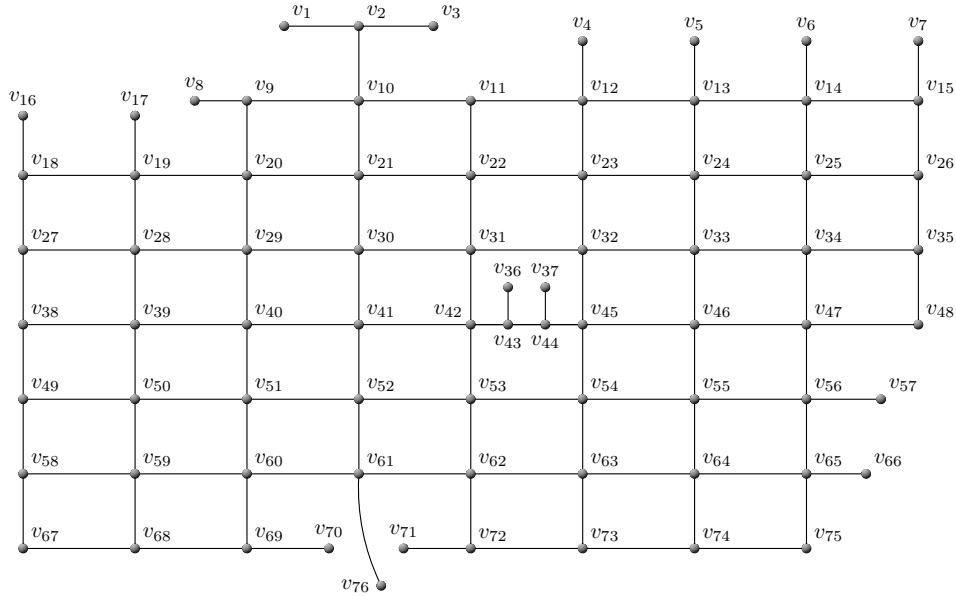


Figure 1.25: A graph $G_{1.57}$ of the *Tshwane* suburb of *Mountain View*, obtained from (a slightly modified version of) the street map contained in [Figure 1.24](#).

2. Find the *adjacency list* representation of $G_{1.57}$ (and store it electronically).
3. Find the *adjacency table* representation of $G_{1.57}$ (and store it electronically).

Further reading

- [1] B Bollobás, 1979. *Graph Theory: An Introductory Course*, Springer, New York, (NY).
- [2] JA Bondy and USR Murty, 1976. *Graph Theory with Applications*, North-Holland, New York (NY).
- [3] G Chartrand, 1985. *Introductory Graph Theory*, Dover Publications, New York (NY).
- [4] G Chartrand and L Lesniak, 1996. *Graphs & Digraphs*, Third edition, Chapman & Hall/CRC, London.
- [5] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY).
- [6] R Diestel, 1997. *Graph Theory*, Volume 173, Springer-Verlag, New York (NY).
- [7] SL Hakimi, 1962. *On the realizability of a set of integers as degrees of the vertices of a graph*, SIAM Journal on Applied Mathematics, **10(3)**, pp. 496–506.
- [8] V Havel, 1955. *A remark on the existence of finite graphs*, Časopis Pro Pěstování Matematiky, **80(4)**, pp. 477–480.

- [9] DB West, 1996. *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River (NJ).
- [10] RJ Wilson and JJ Watkins, 1990. *Graphs: An Introductory Approach*, Wiley, New York (NY).



Graph connectedness

Contents

2.1	Introduction	35
2.2	Connected graphs	36
2.3	Distance in graphs	38
2.4	Cut-vertices and bridges	40
2.5	Directed graphs	41
2.6	Further study of connectivity in graphs	43
	Exercises	43
	Computer exercises	45
	Projects	48
	Further reading	53

2.1 Introduction

It is common practice to incorporate some level of redundancy (in terms of the number of interconnecting links between points or stations in the network) as a fail-safe measure when designing networks for a variety of applications. When incorporating such redundancy into a network there are usually two conflicting objectives: (i) to build in enough redundancy so as to guarantee a certain minimum threshold of fail-safeness in the network, but (ii) to limit the level of redundancy so as to achieve cost-effectiveness. An optimum level of redundancy in network designs is therefore a trade-off between achieving these two objectives.

Consider, for instance, a road network in a large city. Roadways should be designed so that one can drive from any part of the city to any other part of the city, without too much of a detour; hence the road network should at least be connected. Furthermore, if the natural or most desirable route to one's destination has been damaged or rendered inaccessible due to traffic congestion, one would expect that there should be at least one other route to the same destination — hence a level of road redundancy should be present in the network, to be used as alternatives, when required. Another important aspect of designing road networks is that they should be constructed cost-efficiently; hence there should not be too much redundancy in terms of alternative routes, so as to avoid an excessive cost associated with building the network.

Another example of this bi-objective design phenomenon may be found in the design of a national electricity grid. In such a network, cities and power stations should be interconnected by means of high voltage power lines. In this application the aim of the national power utility is typically to interconnect the cities and power stations in such a manner that, should some power line or power station fail, all cities can still receive power from other power stations or via an alternative route (hence some power line redundancy is required). At the same time, however, the national power utility typically aims to minimise the total length of power lines that have to be built, so as to reduce cost (hence not too much power line redundancy).

In the above examples the networks may be modelled as graphs in which the street intersections/dead ends or the cities/power stations are represented by vertices and in which two vertices are adjacent if there is a street section or a power line between the two network infrastructure components corresponding to the vertices. In this chapter, we develop some very basic graph theoretic terminology and a number of results in terms of which the two aforementioned design objectives may be made precise and addressed.

2.2 Connected graphs

We start by defining a walk in a graph. Let u and v be two (not necessarily distinct) vertices of a graph G . A u - v **walk** in G is a finite, alternating sequence of vertices and edges that begins with the vertex u and ends with the vertex v in which each edge of the sequence joins the vertex that precedes it in the sequence to the vertex that follows it in the sequence. The number of edges in the walk is called the **length** of the walk.

Note that we do not require all the edges or vertices in a walk to be different. Often only the vertices of a walk are listed, for the edges present are then obvious. For example, $v_3, v_3v_2, v_2, v_2v_6, v_6, v_6v_3, v_3, v_3v_4, v_4, v_4v_5, v_5, v_5v_4, v_4$ is a v_3 - v_4 walk in the graph $G_{2.1}$ shown in Figure 2.1. This walk may be expressed more compactly, by omitting the edges (and commas), as $v_3\ v_2\ v_6\ v_3\ v_4\ v_5\ v_4$. This walk has length 6.

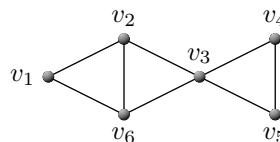


Figure 2.1: The graph $G_{2.1}$.

If all the edges (but not necessarily all the vertices) of a walk are different, then the walk is called a **trail**. If, in addition, all the vertices are different, then the trail is called a **path**. We consider a single vertex as a **trivial path** (walk or trail).

The v_3 - v_4 walk described above is not a trail of $G_{2.1}$ (since the edge v_4v_5 occurs twice). However, $v_3\ v_2\ v_6\ v_3\ v_4$ is a v_3 - v_4 trail in the graph $G_{2.1}$. This trail is not a path (since the vertex v_3 is repeated). The trail $v_3\ v_5\ v_4$ is a v_3 - v_4 path in $G_{2.1}$.

It is also useful to have special terms for those walks or trails which start and end at the same vertex. A u - v walk is **closed** if $u = v$, or **open** otherwise. A closed walk in which all the edges are different is a **closed trail**. A closed trail which

contains at least three vertices is called a **circuit**. A circuit in which no vertices are repeated (except the first and last) is called a **cycle**. The **length** of a cycle (or circuit) is the number of edges in the cycle (or circuit). A cycle of length n is sometimes called an **n -cycle**. A 3-cycle is also called a **triangle**. A cycle is **even** if its length is even; otherwise it is **odd**. An **acyclic graph** has no cycles.

In the graph $G_{2.2}$ shown in Figure 2.2, $v_1 v_2 v_3 v_4 v_5 v_2 v_6 v_1$ is a circuit (of length 7) that is not a cycle, while $v_2 v_4 v_3 v_5 v_2$ is a cycle (as well as a circuit) of length 4. The graph $G_{2.2}$ is therefore not acyclic.

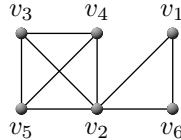


Figure 2.2: The graph $G_{2.2}$.

By definition, every path is a trail and every trail is a walk. Although the converse of each of these statements fails to hold, we do have a result that relates walks and paths.

Theorem 2.1 *Every u - v walk in a graph contains a u - v path.*

Proof Let W be a u - v walk in a graph G . If W is closed, then it contains the trivial path u . Assume, therefore, that $W : u = u_0 u_1 u_2 \cdots u_k$ is an open walk in G , with $u = u_0$ and $v = u_k$. Note that a vertex will have received more than one label if it occurs more than once in W . If no vertex is repeated, then W is already a path. Otherwise, there are vertices of G that occur twice or more in W . Let i and j be distinct integers with $i < j$ such that $u_i = u_j$. That is, the vertex u_i is repeated as u_j . If we now delete the vertices $u_i, u_{i+1}, \dots, u_{j-1}$ from W , we obtain a u - v walk W_1 which is shorter than W and has fewer repeated vertices. If W_1 is a path, we are done. If not, we continue this deletion process until finally we reach a stage where no vertices are repeated and a u - v path is obtained. ■

Before proceeding further, we mention two special classes of graphs. A **path graph** is a graph consisting of a single path. The path graph of order n is denoted by P_n . Graphical representations of the path graphs P_1, \dots, P_5 may be found in Figure 2.3(a)–(e). A **cycle graph** is a graph consisting of a single cycle. The cycle graph of order n is denoted by C_n ($n \geq 3$). Graphical representations of the cycle graphs C_3, \dots, C_7 may be found in Figure 2.3(f)–(j).

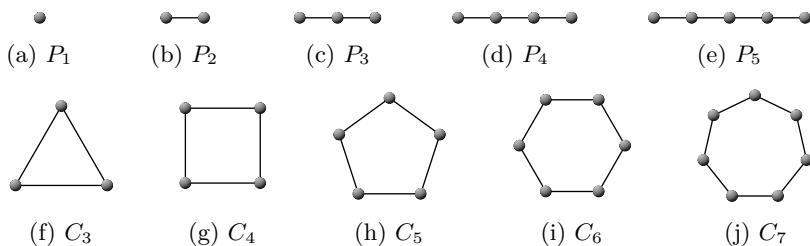


Figure 2.3: Path and cycle graphs.

A graph G is **connected** if there exists a path in G between any two of its vertices, or is **disconnected** otherwise. Every disconnected graph can be partitioned into a number of connected subgraphs, called components. A **component** of a graph G is a maximal connected subgraph of G . Two vertices u and v in a graph G are **connected** if $u = v$, or if $u \neq v$ and a u - v path exists in G . The number of components of G is denoted by $k(G)$. Of course, $k(G) = 1$ if and only if G is connected. For the graph $G_{2,3}$ in Figure 2.4, $k(G_{2,3}) = 6$.

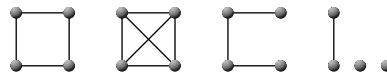


Figure 2.4: The graph $G_{2,3}$ with six components

- ❖ The reader should now be able to attempt Exercises 2.1–2.12 and Projects 2.1–2.2.

2.3 Distance in graphs

For a connected graph G , we define the **distance** $d_G(u, v)$ between two vertices u and v as the minimum of the lengths of the u - v paths of G . If G is clear from the context, we merely write $d(u, v)$ instead of $d_G(u, v)$. If G is a disconnected graph, then the distance between two vertices in the same component of G is defined as above. If, however, u and v belong to different components of G , then $d(u, v)$ is undefined (we write $d(u, v) = \infty$). For the graph $G_{2,4}$ in Figure 2.5, $d(x, u) = 2$, $d(x, w) = 3$ and $d(x, v) = 5$.

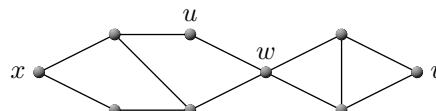


Figure 2.5: The graph $G_{2,4}$

The distance function $d(u, v)$ on pairs of vertices of a graph is a **metric**. That is, it satisfies the following fundamental properties:

Theorem 2.2 Let G be a graph and let u , v and w be any three vertices of G . Then

- (i) $d(u, v) \geq 0$ and $d(u, v) = 0$ if and only if $u = v$;
 - (ii) $d(u, v) = d(v, u)$ (symmetry property);
 - (iii) $d(u, v) \leq d(u, w) + d(w, v)$ (triangle inequality).

Proof of Theorem 2.2 is left as an exercise. We now present a useful characterisation of bipartite graphs, which is attributed by Asratian *et al.* [1] to a 1916 paper by Dénis König.

Theorem 2.3 A nontrivial graph is bipartite if and only if it contains no odd cycles.

Dénes König was a Jewish Hungarian mathematician, born in Budapest. He obtained his doctorate in mathematics from the Technische Hochschule in Budapest in 1907 and joined the faculty there, first as an assistant in problem sessions. In 1910 he was promoted to *Oberassistent* and then to *Privatdocent* in 1911, teaching topology, set theory, real analysis and graph theory (although the name graph theory did not appear in the university yearbook until 1927). Throughout his life, König devoted much of his time to school contests in mathematics, collecting problems for these contests, and organizing them. His academic activity played a vital role in promoting the field of graph theory. In fact, he wrote the first textbook in graph theory entitled *Theorie der endlichen und unendlichen Graphen* in 1936. This marked the beginning of graph theory as a discipline of mathematics. The discipline was later brought onto a firm foundation by [Claude Berge](#) who, in 1958, wrote the second book on graph theory, entitled *Théorie des Graphes et ses applications*, and by [Oystein Ore](#) who wrote the third graph theory text in 1962. König committed suicide in October 1944 in order to evade persecution by the Hungarian National Socialist (Nazi) Party, which had just won power in Hungary.



Biographic note 4: Dénes Kónig (1884–1944)

Proof Let G be a bipartite graph with partite sets V_1 and V_2 . Suppose $C : v_1 v_2 \cdots v_k v_1$ is a cycle of G . Without loss of generality, we may assume that $v_1 \in V_1$. However, then $v_2 \in V_2$, $v_3 \in V_1$, $v_4 \in V_2$, and so on. In general, $v_{2i} \in V_2$ and $v_{2i-1} \in V_1$. This implies that, since $v_k \in V_2$, $k = 2s$ for some positive integer s . Hence, C has even length.

For the converse, it suffices to prove that every nontrivial *connected* graph G without odd cycles is bipartite, since a nontrivial graph is bipartite if and only if each of its components is bipartite. Let $v \in V(G)$ and denote by V_1 the subset of $V(G)$ consisting of v and all vertices u of G with the property that any shortest $u-v$ path of G has *even* length. Let $V_2 = V(G) - V_1$. Thus,

$$V_1 = \{u \in V(G) \mid d_G(u, v) \text{ is even}\}$$

and

$$V_2 = \{u \in V(G) \mid d_G(u, v) \text{ is odd}\}.$$

We now prove that the partition V_1 , V_2 of $V(G)$ has the appropriate properties to show that G is bipartite. Let u and w be elements of V_1 , and suppose, to the contrary, that $uw \in E(G)$. Necessarily, then, neither u nor w is the vertex v . Let $P : v = u_1 u_2 \cdots u_{2r+1} = u$ ($r \geq 1$) and $Q : v = w_1 w_2 \cdots w_{2s+1} = w$ ($s \geq 1$) be a *shortest* $v-u$ path and a *shortest* $v-w$ path in G , respectively. Now let w' denote the last vertex that the two paths P and Q have in common (possibly, $w' = v$). This means that the $w'-u$ subpath of P and the $w'-w$ subpath of Q have only the vertex w' in common. The two $v-w'$ subpaths so determined are *shortest* $v-w'$ paths (for otherwise, we may find a shorter $v-u$ path or $v-w$ path than P or Q , respectively). Hence, there exists an index i such that $w' = u_i = w_i$.

However, proceeding from u_i to u along the path P , and then from u to w along the edge uw , and following the path Q from w back to w_i ($= u_i$) produces a cycle in G , namely

$$u_i \ u_{i+1} \ \cdots \ u_{2r+1} \ w_{2s+1} \ w_{2s} \ \cdots \ w_i,$$

where $w_i = u_i$. This cycle has length $(2r+1-i)+1+(2s+1-i) = 2(r+s+1-i)+1$, which is odd. We have shown, therefore, that G contains an odd cycle, which is a contradiction to our hypothesis. We deduce, therefore, that no two vertices of V_1 are adjacent. Similarly, no two vertices of V_2 are adjacent. Hence G is bipartite with partite sets V_1 and V_2 . ■

❖ The reader should now be able to attempt Exercises 2.13–2.15.

2.4 Cut-vertices and bridges

Some graphs are connected so slightly that they can be disconnected by deleting a single vertex or a single edge. Such vertices and edges play a special role in graph theory, and we discuss these next.

If e is an edge of a graph G , then $G - e$ is the subgraph of G possessing the same vertex set as G and having all the edges of G , except e . If v is a vertex of a graph G containing at least two vertices, then $G - v$ is the subgraph of G whose vertex set consists of all vertices of G , except v , and whose edge set consists of all edges of G except those incident with v . Figure 2.6 illustrates these concepts.

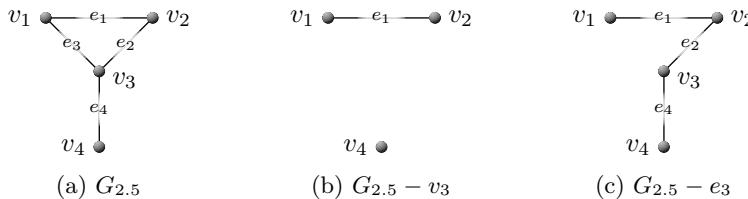


Figure 2.6: Cut-vertices and bridges.

A vertex v of a graph G is called a **cut-vertex** of G if $k(G - v) > k(G)$. So, if G is a connected graph, then v is a cut-vertex if $G - v$ is disconnected. Moreover, an edge e in a graph G is called a **bridge** if $k(G - e) > k(G)$. Therefore, if G is a connected graph, then e is a bridge if $G - e$ is disconnected. If v is a cut-vertex of a connected graph G , then $G - v$ contains two or more components. If e is a bridge of a connected graph G , however, then $G - e$ has exactly two components.

The vertex v_3 of the graph $G_{2.5}$ in Figure 2.6(a) is a cut-vertex. No other vertex of that graph is a cut-vertex. The edge e_4 of the graph $G_{2.5}$ in Figure 2.6(a) is a bridge, but no other edge of that graph is a bridge. The following theorem characterises cut-vertices.

Theorem 2.4 *A vertex v of a connected graph G is a cut-vertex of G if and only if there exist vertices u and w ($u, w \neq v$) such that v is on every $u-w$ path.*

Proof Let v be a cut-vertex of G . Then $G - v$ is disconnected. If u and w are vertices in different components of $G - v$, then there are no $u-w$ paths in $G - v$. Since G is connected, however, there are $u-w$ paths in G . Thus, v must lie on every $u-w$ path. Conversely, suppose that there exist vertices u and w in G such that v lies on every $u-w$ path in G . Then, there are no $u-w$ paths in $G - v$, and so $G - v$ is disconnected. Thus, v is a cut-vertex of G . ■

Bridges are characterised in a manner similar to that of cut-vertices. Proof of the following theorem is left as an exercise.

Theorem 2.5 *An edge e of a connected graph G is a bridge of G if and only if there exist vertices u and w such that e is on every $u-w$ path of G .*

For bridges, there is another useful characterisation.

Theorem 2.6 *Let G be a connected graph. An edge e of G is a bridge if and only if e does not lie on any cycle of G .*

Proof Let $e = uv$ be a bridge of G . Suppose, to the contrary, that e does lie on a cycle C . Then, $C - e$ is a path. Renaming u and v if necessary, we may assume that $C - e$ is a $u-v$ path. We show that every two vertices are connected in $G - e$. Let x and y be any two vertices of $G - e$. Since G is connected, there is an $x-y$ path P in G . On the one hand, suppose $e \notin E(P)$. Then, P is also a path in $G - e$, implying that x is connected to y in $G - e$. On the other hand, suppose $e \in E(P)$. Renaming u and v if necessary, we may assume that u precedes v on P . Replacing e on P by the $u-v$ path $C - e$ produces an $x-y$ walk in $G - e$. By Theorem 2.1, this walk contains an $x-y$ path (in $G - e$). Hence every two vertices are connected in $G - e$ and so $G - e$ is connected, contradicting the fact that e is a bridge of G . Therefore, e does not lie on any cycle of G .

Conversely, suppose $e = uv$ is an edge which lies on no cycle of G . Again, we present a proof by contradiction. Suppose, to the contrary, that e is not a bridge. Then $G - e$ is connected and hence there is a $u-v$ path P in $G - e$. However, P together with e produces a cycle in G containing e , a contradiction. Therefore, e is a bridge of G . ■

A **cycle edge** of a graph G is an edge that belongs to a cycle in G . By Theorem 2.6, every edge of a graph G is either a bridge or a cycle edge.

❖ The reader should now be able to attempt Exercises 2.16–2.24 and Project 2.3.

2.5 Directed graphs

The notions concerning connectedness introduced so far in this chapter for graphs, generalise naturally to digraphs as well. Let u and v be two vertices of a digraph D . Then a (directed) $u-v$ **walk** in D is a finite, alternating sequence

$$u = u_0, a_1, u_1, a_2, \dots, u_{k-1}, a_{k-1}, u_k = v$$

of vertices and arcs that begins with the vertex u and ends with the vertex v , such that $a_i = (u_{i-1}, u_i)$ for all $i \in [k]$. This walk is again abbreviated as $u_0 u_1 u_2 \dots u_k$. The number of arcs in the walk (*i.e.* k) is called the **length** of the walk. The concepts of **trails**, **paths**, **cycles** and **circuits** for digraphs are defined analogously to those for graphs, except that in digraphs, we always proceed in the direction of the arcs. Consequently, whenever we refer to a walk, trail, path, cycle or circuit in a digraph, we mean a *directed* walk, a *directed* trail, a *directed* path, a *directed* cycle or a *directed* circuit, respectively. Note that cycles of length 2 are possible in digraphs, whereas this is, of course, not possible in graphs.

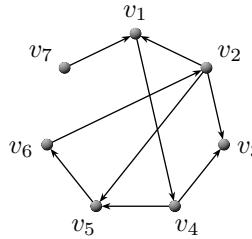


Figure 2.7: The digraph $D_{2,6}$.

To illustrate these concepts, consider the digraph $D_{2,6}$ shown in Figure 2.7. Here, $W : v_2 v_5 v_6 v_2 v_5$ is a v_2 - v_5 walk of length 4 in $D_{2,6}$. The arc (v_2, v_5) occurs twice in W , and so W is not a trail. However, $T : v_2 v_5 v_6 v_2 v_1 v_4$ is a v_2 - v_4 trail in $D_{2,6}$. This trail T is not a path, since the vertex v_2 is repeated in T . Moreover, $P : v_1 v_4 v_5 v_6 v_2 v_3$ is a v_1 - v_3 path in $D_{2,6}$. Also, $S : v_2 v_5 v_6 v_2 v_1 v_4 v_5 v_6 v_2$ is a circuit that is not a cycle, while $C : v_2 v_5 v_6 v_2$ is a cycle of length 3 in $D_{2,6}$.

A term that is unique to digraph theory is that of a semiwalk. Let u and v be two vertices of a digraph D . A **u - v semiwalk** in D is a finite, alternating sequence

$$u = u_0, a_1, u_1, a_2, \dots, u_{k-1}, a_{k-1}, u_k = v$$

of vertices and arcs that begins with the vertex u and ends with the vertex v , such that either $a_i = (u_{i-1}, u_i)$ or $a_i = (u_i, u_{i-1})$ for all $i \in [k]$, which is abbreviated as $u_0 u_1 u_2 \dots u_k$. If $a_i = (u_{i-1}, u_i)$, then we call a_i a **forward arc**; otherwise, we call a_i a **backward arc**. The number of arcs in a semiwalk is called the **length** of the semiwalk. If the vertices u_0, \dots, u_k are distinct, then the u - v semiwalk is called a **u - v semipath**.

A digraph D is **weakly connected** if the underlying graph of D is connected, while D is **strongly connected** if, for every pair u, v of distinct vertices, D contains both a u - v path and a v - u path. A strongly connected digraph is also sometimes called a **strong digraph**.

The concept of distance in digraphs may be defined as in the case of graphs. For vertices u and v in a digraph D , we define the (directed) distance $d_D(u, v)$ or merely $d(u, v)$, from u to v as the minimum of the lengths of all (directed) u - v paths in D . The distances $d(u, v)$ and $d(v, u)$ are therefore defined for all pairs u, v of vertices in a strong digraph.

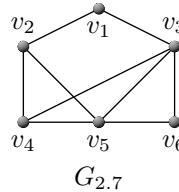
❖ The reader should now be able to attempt Exercises 2.25–2.26.

2.6 Further study of connectivity in graphs

In a later chapter we continue our study of the degree of robustness of a graph G in terms of its ability to withstand multiple edge or vertex removals before the number of components of G increases (see Chapter 12). More specifically, in that chapter we consider questions such as: *What is the smallest number of vertices or edges whose deletion from a connected graph produces a disconnected graph or trivial graph?* Before we can develop a theory in this respect, however, it is necessary to consider a number of more fundamental concepts and prerequisite notation.

Exercises

- 2.1 Give an example of a disconnected graph with four components in which each component is complete.
- 2.2 Give an example of a disconnected graph with three components in which every two components are isomorphic.
- 2.3 In the accompanying graph $G_{2.7}$, give an example of:
 - (a) a circuit that is not a cycle;
 - (b) a trail that is not a path;
 - (c) a path;
 - (d) a cycle.

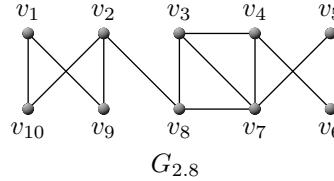


$G_{2.7}$

- 2.4 Show that if G is a graph with minimum degree $\delta \geq 2$, then G contains a cycle of length at least $\delta + 1$. (Hint: Consider a longest path P in G . Let u be an end-vertex of P and consider the vertices adjacent to u in G .)
- 2.5 Show that if G is a bipartite graph with minimum degree $\delta \geq 1$, then G contains a path of order at least 2δ .
- 2.6 Prove that a graph and its complement cannot both be disconnected.
- 2.7 A graph is **self-complementary** if it is isomorphic to its complement.
 - (a) Show that C_5 is self-complementary.
 - (b) Prove that a self-complementary graph has $4k$ or $4k + 1$ vertices, for some integer k (that is, if G is self-complementary of order n , then $n \equiv 0 \pmod{4}$ or $n \equiv 1 \pmod{4}$).
- 2.8 Let G be a graph of even order n (*i.e.* $n = 2s$ for some positive integer s) such that G has two complete components. Prove that the minimum size possible for G is $m = (n^2 - 2n)/4$. (Hint: Try a calculus argument.) If G has this size, what does G look like?

- 2.9 Let G be a graph of order n such that $d(v) \geq (n-1)/2$ for every $v \in V(G)$. Prove that G is connected. (Hint: Try a proof by contradiction: Assume, to the contrary, that G is disconnected. This means that G has two or more components. What can be said about the number of vertices in each component?)
- 2.10 Let G be a graph of order $n \geq 2$ such that $d(v) \geq (n-2)/2$ for every $v \in V(G)$. Show that G need not be connected if n is even.
- 2.11 Suppose that G is a graph with no vertex of degree 0 and no induced subgraph with exactly two edges. Prove that G is connected. Hence prove that G is a complete graph.
- 2.12 Suppose that G is a connected graph with no vertex of degree 0 and no induced subgraph with exactly three edges. Prove that G has maximum degree 2. Hence prove that G has at most four vertices.
- 2.13 Show that if G is a graph of order n and size $n^2/4$, then either G contains an odd cycle or $G \cong K_{n/2,n/2}$. (Hint: Suppose G contains no odd cycle. Then, by [Theorem 2.3](#), G is bipartite.)
- 2.14 Prove [Theorem 2.2\(i\)-\(iii\)](#).

- 2.15 Find the distances between each pair of vertices in the graph $G_{2.8}$ shown below.



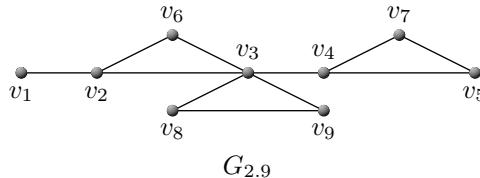
- 2.16 Identify all

- (a) cut-vertices;
- (b) bridges

in the graph $G_{2.8}$ of [Exercise 2.15](#).

- 2.17 In the graph $G_{2.9}$ shown below, identify all

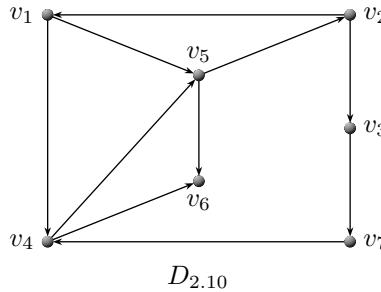
- (a) cut-vertices;
- (b) bridges.



- 2.18 Prove [Theorem 2.5](#).

- 2.19 Give an example of a connected graph containing more cut-vertices than bridges.
- 2.20 Give an example of a connected graph containing more bridges than cut-vertices.

- 2.21 Prove that every graph containing only even vertices cannot contain a bridge.
- 2.22 Let G be a connected graph containing only even vertices. Show that it is possible for G to contain cut-vertices.
- 2.23 Prove that if v is a cut-vertex of a connected graph G , then v is not a cut-vertex of \bar{G} .
- 2.24 Give an example of a connected, 3-regular graph containing a bridge.
- 2.25 Find, in the digraph $D_{2.10}$ below:
- a v_1 - v_6 path of length 6;
 - a v_1 - v_6 trail that is not a path;
 - a v_1 - v_6 walk that is not a trail;
 - v_1 - v_6 semiwalks of lengths 3, 4 and 5.



- 2.26 Is the digraph $D_{2.10}$ in [Exercise 2.25](#):

- weakly connected;
- strongly connected?

Motivate in each case.

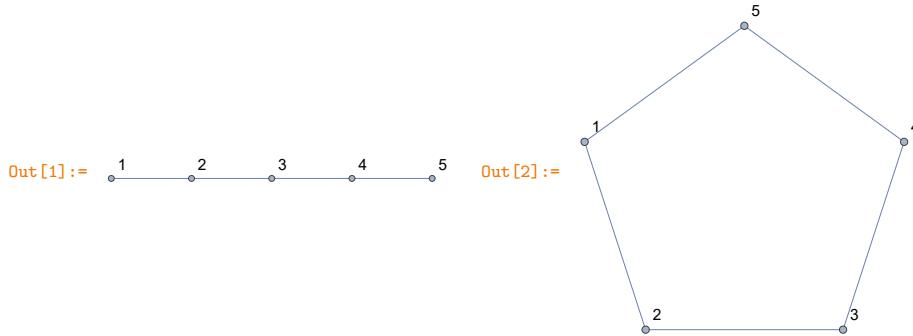
Computer exercises

The **MATHEMATICA** commands `PathGraph[Range[n]]` and `CycleGraph[n]` may be used to generate respectively a path and a cycle of order n . Here the function `Range[n]` returns the list $\{1, \dots, n\}$, which is taken as the vertex set of the path. As mentioned in the previous chapter, the optional attribute `VertexLabels -> "Name"` may be included in each of these commands to produce vertex-labelled graphs. Furthermore, the function `FindMinimumCut[G]` returns the smallest number of edges of a connected graph G whose removal will disconnect G , together with a partition of the vertex set for the cut, while the functions `FindEdgeCut[G]` and `FindVertexCut[G]` return smallest lists of edges and vertices, respectively, that will disconnect G . For example, the commands

```
In[1]:= Pa = PathGraph[Range[5], VertexLabels -> "Name"]
In[2]:= Cy = CycleGraph[5, VertexLabels -> "Name"]
In[3]:= FindMinimumCut[Pa]
In[4]:= FindMinimumCut[Cy]
In[5]:= FindEdgeCut[Pa]
In[6]:= FindEdgeCut[Cy]
```

```
In[7]:= FindVertexCut[Pa]
In[8]:= FindVertexCut[Py]
```

produce the output:

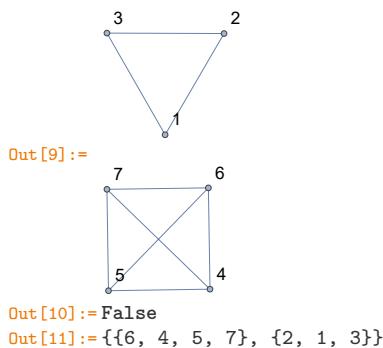


```
Out[3]:= {1, {{5}, {1, 2, 3, 4}}}
Out[4]:= {2, {{5}, {1, 2, 3, 4}}}
Out[5]:= {4 ↔ 5}
Out[6]:= {1 ↔ 5, 4 ↔ 5}
Out[7]:= {2}
Out[8]:= {2, 5}
```

The command `ConnectedGraph[G]` returns the boolean value **True** if the graph G is connected, or the value **False** otherwise. The components of G may be produced by means of the command `ConnectedComponents[G]`, while the command `AcyclicGraphQ[G]` returns the boolean value **True** if G contains no cycles, or the value **False** otherwise. For example, the commands

```
In[9]:= G = GraphDisjointUnion[CompleteGraph[3], CompleteGraph[4], VertexLabels ->
"Name"]
In[10]:= ConnectedGraphQ[G]
In[11]:= ConnectedComponents[G]
```

produce the output:



```
Out[10]:= False
Out[11]:= {{6, 4, 5, 7}, {2, 1, 3}}
```

A function has the syntax `Func[x1_, x2_, ...]:= (body)` in **MATHEMATICA**, where $x1_$, $x2_$, ... are local variables defined within the scope of the function, and where `body` contains a number of operations to be performed on these variables. Such a function may be called from command line by executing the command `Func[x1, x2, ...]`. To see how this is accomplished, consider the command `Select[list, condition]` which picks out each element a from `list` for which

condition is true when applied to a . Also, the command `EdgeDelete[G, e]` returns that graph obtained by removing the edge e from a graph G . Using these notions, a new function `Bridges[G]` may be defined as follows for determining all the bridges of G :

```
In[13]:= Bridges[G_] := Select[EdgeList[G], Not[ConnectedGraphQ[EdgeDelete[G, #]]] &]
```

Executing the above command does not produce any output of its own; it merely provides **MATHEMATICA** with a definition of how to carry out the function `Bridges`, if it were to be called from the command line. In the above definition, the hash `#` represents a variable which should take as values the items in a list (in this case the edge list of G), one at a time in turn, while the ampersand `&` denotes the end of the scope of this iterative substitution of the variable `#`. Execution of the commands

```
In[14]:= Bridges[Pa]
In[15]:= Bridges[Cy]
```

now produce the output:

```
Out[14]:= {1 ↔ 2, 2 ↔ 3, 3 ↔ 4, 4 ↔ 5}
Out[15]:= {}
```

The command `GraphDistance[G, v]` may be used to compute a list of distances from a specified vertex v of a graph G to all other vertices of G . For example, the commands

```
In[16]:= H = GraphDisjointUnion[Pa, Cy];
In[17]:= GraphDistance[Pa, 1]
In[18]:= GraphDistance[Cy, 1]
In[19]:= GraphDistance[H, 1]
```

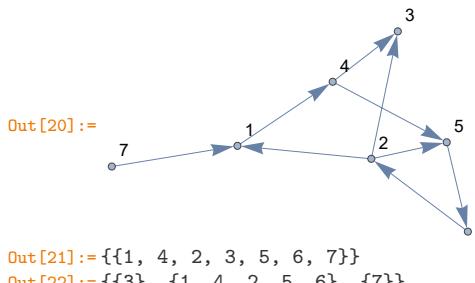
produce the output:

```
Out[17]:= {0, 1, 2, 3, 4}
Out[18]:= {0, 1, 2, 2, 1}
Out[19]:= {0, 1, 2, 3, 4, ∞, ∞, ∞, ∞, ∞}
```

Finally, `WeaklyConnectedComponents[G]` and `ConnectedComponents[G]` return the weakly connected components and the strongly connected components of a graph G , respectively. For example, the commands

```
In[20]:= G = Graph[{1->4, 2->1, 2->3, 2->5, 4->3, 4->5, 5->6, 6->2, 7->1},
  VertexLabels -> "Name"]
In[21]:= WeaklyConnectedComponents[G]
In[22]:= ConnectedComponents[G]
```

produce the output:



- ❖ The reader should now be able to repeat Exercises 2.7(a), 2.16 and 2.17 using MATHEMATICA.

Projects

This section contains three projects. The first two projects demonstrate the power of graph theory in a structured approach toward solving recreational puzzles that are challenging to solve by brute force, while the third project is more practical in nature. More specifically, the first project is dedicated to the development of a structured solution approach for a puzzle within the class of crossing recreational puzzles in which three missionaries and three cannibals have to cross a river in a boat that can hold at most two people, in such a way that the cannibals never outnumber the missionaries on any bank of the river. The second project is concerned with a harder puzzle in which four carefully coloured cubes have to be stacked on top of one another so that no stack face contains the same colour twice. In the third project, it is demonstrated how the identification of cut-vertices and bridges in a graph may emerge naturally in a practical problem setting.

Project 2.1: Cannibals and missionaries

Consider the following puzzle, known as the *cannibals and missionaries puzzle* or the *puzzle of the jealous husbands*, which is a special instance of a well-known class of “difficult crossing problems” [4]. Suppose there are three missionaries and three cannibals on the west bank of a river. There is also a boat on the west bank that can hold no more than two people. The missionaries and the cannibals wish to cross to the east bank. But they have a problem: If, on either bank, the cannibals ever outnumber the missionaries, the outnumbered missionaries will be in trouble.

Is there a way for the missionaries to get their wish (*i.e.* to travel to the east bank without losing anyone?)

Tasks

1. Solve the puzzle by brute force (*i.e.* by a process of elimination).

Let us now consider a structured, graph theoretic approach towards solving the puzzle. Because knowledge of the number of missionaries and cannibals on the west bank before and after each boat crossing completely dictates the corresponding number of missionaries and cannibals on the east bank, we shall consider all possible states of the west bank only. Let C and M denote respectively the number of cannibals and missionaries on the west bank of the river.

2. It follows that there are sixteen possible states on the west bank, since $C, M \in \{0, 1, 2, 3\}$. Six of these states are, however, not allowed due to the puzzle restriction that the cannibals should never outnumber the missionaries on any bank of the river. Identify the six coordinates (C, M) that are not allowed and mention on which river bank the puzzle requirement would have been violated had each of these six coordinate pairs been allowed.

3. Now arrange the remaining ten states as points dictated by their coordinates in the first quadrant of \mathbb{R}^2 . Consider these ten points the vertices of a *state graph*, in which two vertices are adjacent if the corresponding states can be reached from one another by means of a single boat crossing.

Observe that a passage of the boat from the west bank to the east bank reduces the values of C and M , since it removes people from the west bank. The total decrease in the value of $C + M$ is either one or two, since the capacity of the boat is two. Thus, the only vertices of the state graph that are reachable from a given vertex lie in a right triangle with the given vertex at the upper right vertex, as shown in [Figure 2.8\(a\)](#). (Of course, some of these reachable vertices may not be allowable and hence be one of the six vertices excluded in [Task 2](#).) Similarly, a passage of the boat from the east bank to the west bank of the river corresponds to an increase in the values of C and M , since it brings people to the west bank.



(a) West to east

(b) East to west

Figure 2.8: Reachable states from • .

The puzzle clearly has a solution if and only if the states $(3, 3)$ and $(0, 0)$ are in the same component of the state graph. At first glance one might think that each walk from $(3, 3)$ to $(0, 0)$ in the state graph would correspond to a solution to the puzzle. However, some walks, such as $(3, 3), (2, 2), (1, 1), (0, 0)$, are not permissible. This is because boat passages from the west bank to the east bank must alternate passages from the east bank to the west bank. This means that edges traversed downwards or to the left in a walk from $(3, 3)$ to $(0, 0)$ in the state graph should alternate edges traversed upwards or to the right — such a walk is called a *permissible walk*.

4. Find a permissible walk from $(3, 3)$ to $(0, 0)$ in the state graph, and interpret the corresponding solution to the puzzle.

It is natural to ask whether the solution that you obtained in [Task 4](#) is unique, and whether a solution exists if the numbers of cannibals and missionaries are varied, or if the boat capacity is varied. Answers to these interesting questions are provided by [Fraley et al.](#) [4].

Project 2.2: Instant insanity

Consider the following puzzle, known by many names, including *The Great Tantalizer* and *Instant Insanity*: Four cubes, whose six faces are coloured red (R), blue (B), green (G) and yellow (Y) according to the patterns shown in [Figure 2.9\(a\)–\(d\)](#) are to be stacked (on top of each other) to form a vertical tower whose four long (shaded) faces should each contain all four colours, as shown in [Figure 2.9\(e\)](#). The stacking order of the cubes is not important.

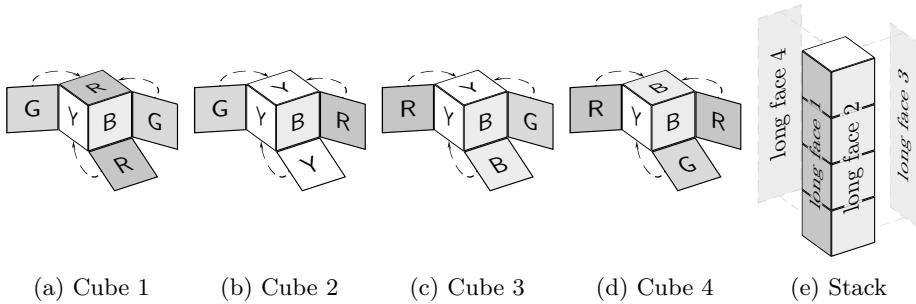


Figure 2.9: Cubes used in the puzzle *Instant Insanity*.

Tasks

1. Construct four cardboard cubes that are coloured according to the patterns shown in Figure 2.9(a)–(d).
2. Spend 10 minutes attempting to solve the puzzle by brute force (*i.e.* by a process of trial and error).

Now that you are convinced of the nontriviality of the puzzle — there are exactly 41 472 different ways to order and rotate the cubes in a stack — let us utilise the power of graph theory...

3. Draw four pseudographs, G_1 , G_2 , G_3 and G_4 , each of order 4, whose vertices represent the four colours red (R), blue (B), green (G) and yellow (Y), in such a way that two vertices v_i and v_j are adjacent in G_k if the colours i and j are on directly opposite faces of cube $k \in [4]$.
4. Now superimpose the drawings of your four pseudographs on top of one another so that all four vertices representing a single colour merge to form one vertex representing that colour, for all colours, so as to form a graphical representation of the pseudograph $G = G_1 \oplus G_2 \oplus G_3 \oplus G_4$ which we call the *stacking graph* (*i.e.* your four pseudographs in Task 3 are factors of the stacking graph G). Take care, however, as multi-edges will result — keep track of the original pseudograph producing each edge by labelling all edges in G originating in G_k with the symbol $k \in [4]$.
5. Find two disjoint cycles in the stacking graph G , each with the properties that it (i) contains all four vertices of G and (ii) contains all four edge labels 1, 2, 3 and 4. (By disjoint cycles we mean that the two cycles share no edges.) Let us call the two resulting cycles the *white cycle* and the *black cycle*.

The white cycle found in Task 5 may be used to determine a stacking order of the four cubes. Suppose the white cycle is given by the sequence of alternating vertex and edge labels $v_{i_1}, e_{j_1}, v_{i_2}, e_{j_2}, v_{i_3}, e_{j_3}, v_{i_4}, e_{j_4}, v_{i_1}$, where i_1, i_2, i_3, i_4 represent, in order, the colours encountered while traversing the white cycle (in some direction) and where j_1, j_2, j_3, j_4 represent, in order, the edge labels (cube numbers) encountered while traversing the white cycle (in the same direction).

6. Stack the cubes in the order j_1, j_2, j_3, j_4 , with cube j_1 at the bottom and cube j_4 at the top, in such a way that the colours i_1 and i_2 are opposite each other on cube j_1 , that the colours i_2 and i_3 are opposite each other on cube j_2 , that the colours i_3 and i_4 are opposite each other on cube j_3 and that the colours i_4 and i_1 are opposite each other on cube j_4 , as dictated by the white cycle, and illustrated in [Figure 2.10\(a\)](#). The resulting stack has the required colour distribution property along two of its four long (vertical) faces, marked “resolved faces” in [Figure 2.10\(b\)](#).

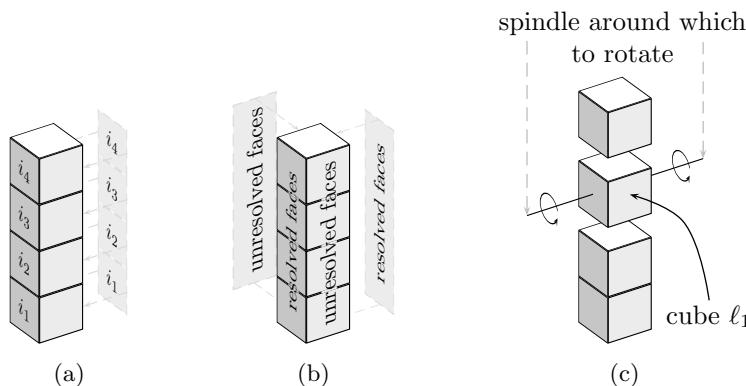


Figure 2.10: Illustration of the tasks for the puzzle *Instant Insanity*.

The black cycle found in [Task 5](#) may be used to determine the orientation of each cube around horizontal spindles, as shown in [Figure 2.10\(c\)](#). Suppose the black cycle is given by the sequence of alternating vertex and edge labels $v_{k_1}, e_{\ell_1}, v_{k_2}, e_{\ell_2}, v_{k_3}, e_{\ell_3}, v_{k_4}, e_{\ell_4}, v_{k_1}$, where k_1, k_2, k_3, k_4 represent, in order, the colours encountered while traversing the black cycle (in some direction) and where $\ell_1, \ell_2, \ell_3, \ell_4$ represent, in order, the edge labels (cube numbers) encountered while traversing the black cycle (in the same direction).

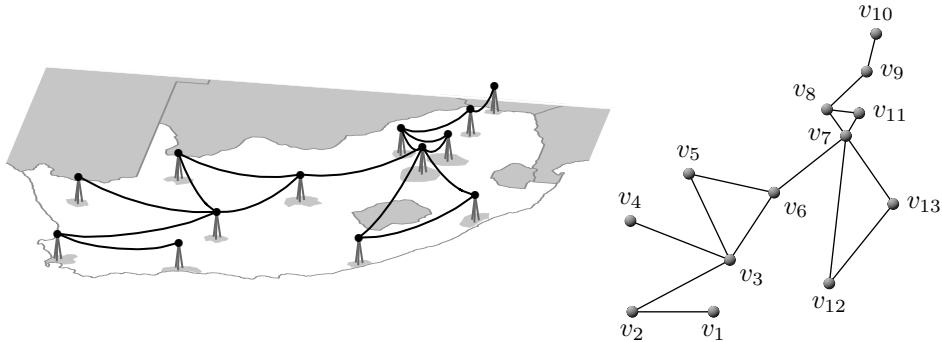
7. Rotate cube ℓ_1 around its horizontal spindle shown in [Figure 2.10\(c\)](#) until the colours k_1 and k_2 appear opposite each other on the two stack faces marked “unresolved faces” [Figure 2.10\(b\)](#). Similarly, rotate cube ℓ_2 around its horizontal spindle until the colours k_2 and k_3 appear opposite each other on the two stack faces marked “unresolved faces” in [Figure 2.10\(b\)](#), and so on.

The resulting stack has the required colour distribution property along all four of its long (vertical) faces! All this may look somewhat cumbersome, but the graph theoretic approach is surely more efficient than trying thousands of combinations, assuming that one has a method of ensuring that no previous combinations are repeated. Those familiar with graph theory can typically solve the puzzle in minutes...

Project 2.3: Fail-safe design of an electricity network

Consider the simple electricity network involving thirteen South African cities shown in [Figure 2.11\(a\)](#). This electricity network may be modelled by the graph

$G_{2.11}$, shown in Figure 2.11(b), in which the cities are represented by vertices, and in which high-voltage power lines interconnecting the cities are represented by edges.



(a) Isometric projection of a simple South African electricity network (b) Graph model $G_{2.11}$

City		City	
v_1	George	v_8	Rustenburg
v_2	Cape Town	v_9	Polokwane ^a
v_3	De Aar	v_{10}	Messina
v_4	Springbok	v_{11}	Tshwane ^b
v_5	Upington	v_{12}	Gqeberha ^c
v_6	Bloemfontein	v_{13}	Durban
v_7	Johannesburg		

^aFormerly known as Pietersburg.

^bFormerly known as Pretoria.

^cFormerly known as Port Elizabeth and then as Nelson Mandela Bay.

Figure 2.11: Simple electricity network of South Africa.

The national power utility is particularly interested in those critical city power distribution points and power lines that should especially be protected by rigorous maintenance programmes (high cost maintenance programmes for all power lines and city distribution points is not an affordable option). Here the adjective *critical* should be understood to apply to a power line which, when it fails, leaves the network disconnected, causing some cities to experience a power loss, or to a city power distribution point which, when it fails, leaves some *other* cities disconnected from each other in the network.

Tasks

1. The critical city power distribution points are clearly those that correspond to cut-vertices of the graph $G_{2.11}$. Find all cut-vertices of $G_{2.11}$.
2. The critical high voltage power lines are clearly those that correspond to bridges of the graph $G_{2.11}$. Find all bridges of $G_{2.11}$.

3. What is the minimum number of edges that can be added to $G_{2,11}$ so as to eliminate all cut-vertices (*i.e.* to render the graph cut-vertex-free), and where should this minimum number of edges be inserted into $G_{2,11}$?
4. What is the minimum number of edges that can be added to $G_{2,11}$ so as to eliminate all bridges (*i.e.* to render the graph bridge-free), and where should this minimum number of edges be inserted into $G_{2,11}$?
5. Where should the power utility invest in building additional high-voltage power lines so as to eliminate all critical infrastructure components from the national electricity grid?

Further reading

- [1] AS Asratian, TMJ Denley and R Häggkvist, 1998. *Bipartite Graphs and their Applications*, Cambridge University Press, Cambridge.
- [2] F Buckley and F Harary, 1990. *Distance in Graphs*, Addison-Wesley, Reading (MA).
- [3] G Chartrand and L Lesniak, 1996. *Graphs & Digraphs*, Third edition, Chapman & Hall/CRC, London.
- [4] R Fraley, KL Cooke and P Detrick, 1966. *Graphical solution of difficult crossing puzzles*, Mathematics Magazine, **39**(3), pp. 151–157.
- [5] DB West, 1996. *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River (NJ).



Algorithmic complexity

Contents

3.1	Introduction	55
3.2	“Big \mathcal{O} ” notation	60
3.3	A classification scheme for decision problems	63
3.4	Polynomial-time reducibility and NP-completeness	64
3.5	Decision problems vs. computation problems	70
	Exercises	73
	Computer exercises	76
	Projects	78
	Further reading	85

3.1 Introduction

An **algorithm** may be defined as an ordered sequence of procedural operations for solving a problem within a finite number of steps. In this book we shall encounter a variety of algorithms that may be used to solve a number of well-known computational problems in graph theory, and in each case we would like to be able to evaluate the efficiency of the algorithm in terms of its speed and the amount of computer memory required to execute the algorithm. These considerations are important in order to be sure that an algorithm may be expected to yield results within a realistic time on a computer with a realistic memory capacity for a computational problem instance of a given size.

Algorithmic complexity is the term given to the process of quantification of the amount of time and memory expended by a computer when performing the steps of an algorithm, and is usually measured by two variables: the **time complexity** T and the **space complexity** S of the algorithm. If n is the size of the input to the algorithm, then T and S are typically expressed as functions of n . In the chapters that follow, the input size of an algorithm will often be taken as the size or order of a graph instance for which a particular computational problem has to be solved. Consider, for example, [Algorithm 1](#) which computes an unordered version of the degree sequence of a graph.

In [Algorithm 1](#), the size of the input to the algorithm is the order of the input graph G , that is n . The amount of memory required to implement [Algorithm 1](#) is $S(n) = n^2 + n$ space units, where a space unit is the amount of memory necessary

Algorithm 1: Degree sequence (Iterative)

Input : The $n \times n$ adjacency matrix $\mathbf{A} = [A_{i,j}]$ of a graph G of order n .

Output : An n -vector $\mathbf{d} = [d_1, \dots, d_n]$ containing the (unordered) degree sequence of G .

```

1 for i = 1 to n do
2   |   di ← 0
3   |   for j = 1 to n do
4   |     |   di ← di + Aij
5 print d

```

to store the value of an integer. Here n^2 space units are required to store the adjacency matrix \mathbf{A} on input, and an additional n space units are required to store the output vector \mathbf{d} . The time complexity of [Algorithm 1](#) may be measured in terms of the number, $T(n)$, of **basic operations** required to execute the algorithm. In this case it would be appropriate to define both the addition of two integers and variable assignment as basic operations. With this definition of a basic operation, $T(n) = 2n^2 + n$. One variable assignment occurs in each of Steps 2 and 4 of the algorithm, and since Step 2 is repeated n times and Step 4 is repeated n^2 times, the total number of variable assignments is $n^2 + n$. Furthermore, one integer addition occurs in Step 4 of the algorithm, and since Step 4 is repeated n^2 times, this adds another n^2 basic operations to the tally.

[Algorithm 1](#) is an example of a so-called **iterative algorithm**, because execution of the algorithm involves iteration through a number of computational steps. The algorithm is, however, called only once during a single execution thereof — at the very start. In contrast to this type of algorithm stands the class of so-called **recursive algorithms**. Algorithms in this class also iterate through a number of computational steps, but the difference is that such algorithms may perform calls to themselves during execution; these calls are called **recursive calls**. A smaller problem that is similar in structure to the original problem being solved by the first call to a recursive algorithm is typically solved during each recursive call. Complex computation procedures may often be solved elegantly by means of recursive algorithms, but analyses of the space and time complexities of such algorithms may be quite difficult and rather tedious.

In order to illustrate the notion of a recursive algorithm, let us consider a mathematical puzzle called the *Towers of Hanoi*, which was invented by the French mathematician [Edouard Lucas](#) in 1883. The puzzle is played on three pegs mounted vertically on a board, and n discs of different sizes which can slide onto any peg,

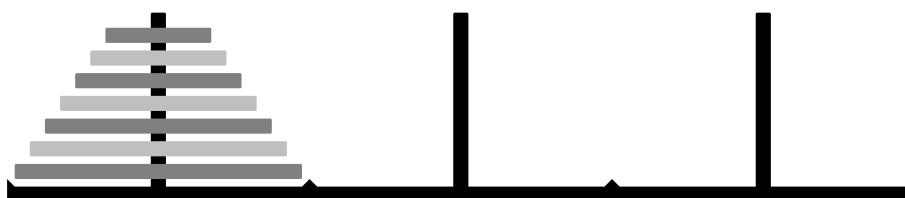


Figure 3.1: The mathematical puzzle called the *Towers of Hanoi*, with $n = 7$.

as shown in [Figure 3.1](#). At the start all the discs are neatly stacked in order of decreasing size on one peg (smallest at the top, thus making a conical shape). The objective is to move the entire stack of discs to another peg, obeying the following rules: (i) only one disc may be moved at a time, and (ii) no disc may be placed on top of a smaller disc.

The French mathematician **François Édouard Anatole Lucas** was born in Amiens. He received his education at the École Normale Supérieure and then worked at The Paris Observatory. During the Franco-Prussian War of 1870–1871, he served as an artillery officer, after which he became professor of mathematics at the Lycée Saint Louis in Paris. He was later appointed professor of mathematics at the Lycée Charlemagne, also in Paris. Lucas is best known for his work in number theory. He studied the famous Fibonacci sequence, and the associated Lucas sequence is named after him. The well-known Lucas-Lehmer primality test was originally due to Lucas and was later refined by Derrick Henry Lehmer in 1930. Lucas invented the puzzle *The Towers of Hanoi* in 1883, which he marketed under the nickname *N. Claus de Siam*, an anagram of *Lucas d'Amiens*.



Biographic note 5: Édouard Lucas (1842–1891)

The question is, “What is the smallest number of moves in which the puzzle can be completed?” Suppose the answer to this question, when n discs are to be moved, is H_n . If some *optimal* $(n - 1)$ -disc moving strategy is known, then an optimal n -disc moving strategy may be derived as follows:

Step 1: Transfer the top $n - 1$ discs from the initial peg to an intermediate peg according to the optimal $(n - 1)$ -disc moving strategy — this requires H_{n-1} moves.

Step 2: Transfer the largest disc (the one remaining on the initial peg) to the final peg — this requires one move.

Step 3: Transfer the $n - 1$ discs on the intermediate peg to the final peg, which already contains the largest disc, according to the optimal $(n - 1)$ -disc moving strategy — this requires another H_{n-1} moves.

This n -disc moving strategy requires $2H_{n-1} + 1$ moves and is clearly optimal, since before being able to move the largest disc away from the initial peg, all the other discs have to be removed from that peg, which is achieved in the minimum number of moves in [Step 1](#). Thereafter the largest disc may be moved, and only to the final peg, because the intermediate peg already contains smaller discs. For an optimal n -disc moving strategy, the stack of discs has to be rebuilt around the final peg in order to avoid moving the large disc again, and is achieved in the minimum number of moves in [Step 3](#). Hence we have the recurrence relation

$$H_n = 2H_{n-1} + 1, \quad n \in \mathbb{N}, \quad H_0 = 0, \tag{3.1}$$

for which the seed (that is, the base case when $n = 0$) is trivial. It is left as an exercise to show, by induction, that the solution to (3.1) is $H_n = 2^n - 1$.

Numbering the discs $1, \dots, n$ in order of increasing size, the pegs $0, 1, 2$ and assuming that discs have to move from peg i to peg j , the call `Hanoi([n], i, j)` of [Algorithm 2](#) will result in an optimal n -disc moving strategy between the desired two pegs. The working of [Algorithm 2](#) according to the three steps described above with the input `Hanoi({1, 2, 3}, 0, 2)` and its recursive calls to itself are illustrated in [Figure 3.2](#). The $H_3 = 2^3 - 1 = 7$ moves required for an optimal 3-disc moving strategy from peg 0 to peg 2 are clearly visible in [Figure 3.2](#).

Algorithm 2: Hanoi (Recursive)

Input : A set $\mathcal{S} = [n]$ of disc indices and peg indices $i, j \in \{0, 1, 2\}$.
Output : An optimal moving strategy involving the discs in \mathcal{S} from peg i to peg j (possibly via further recursive calls).

```

1 if |S| = 1 then
2   print "Move disc ", S, " from peg ", i, " to peg ", j
3 else
4   Hanoi([n - 1], i, 3 - i - j)
5   Hanoi({n}, i, j)
6   Hanoi([n - 1], 3 - i - j, j)

```

If we take a move of a single disc from any peg to any other peg as a basic operation, then the time complexity of [Algorithm 2](#) is $T(n) = H_n = 2^n - 1$, which is an exponential function of the number of discs to be moved. In order to appreciate the dramatic growth rate of this time complexity as n grows, it is worthwhile pondering a legend about an Indian temple in which three time-worn posts are surrounded by 64 golden discs. The priests of Brahma, reportedly acting out the command of an ancient prophecy, have been moving these discs in accordance with the rules of the puzzle. According to the legend, when the last move of the puzzle is completed, the world will end. The puzzle is therefore also known as the *Towers of Brahma* puzzle. It is not clear whether Lucas invented this legend or was inspired by it. If the legend were true, however, and if the priests were able to move discs at the awe-inspiring rate of 1 disc per second, performing the smallest number of moves, it would take them $2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615$ seconds or roughly 585 billion years to complete their puzzle...

To provide the reader with a further indication of the practical relevance and significance of performing complexity measure estimates for algorithms (such as those performed above), consider, for example, the temporal impact of growth in the number of basic operations required by algorithms as their input size n increases. Suppose the input size is $n = 10^6$ and that we have at our disposal a computer capable of processing one million basic operations per second, *i.e.* the operating speed (per basic operation) is one microsecond. The run times of algorithms of different complexity classes on this hypothetical computer are displayed in [Table 3.1](#) [4].

The advantage of measuring the time complexity of an algorithm by counting the number of basic operations, $T(n)$, that it has to perform, rather than measuring

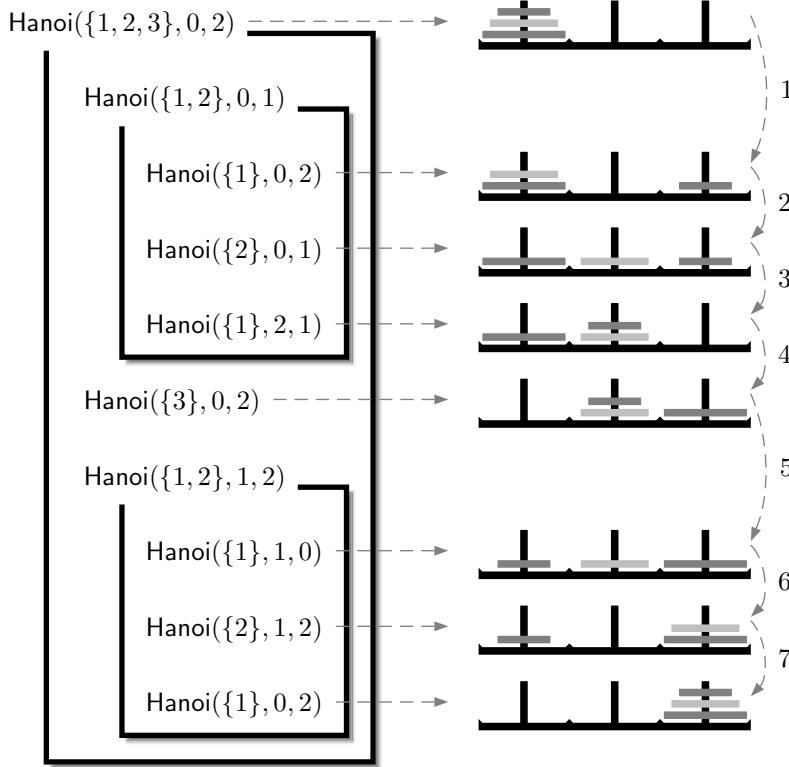


Figure 3.2: Recursive calls in $\text{Hanoi}(\{1, 2, 3\}, 0, 2)$ from [Algorithm 2](#).

Complexity class	$T(n)$	Number of operations	Approximate execution time
Logarithmic	$\ln n$	10	10 microseconds
Linear	n	10^6	1 second
$n \log n$	$n \ln n$	10^7	10 seconds
Quadratic	n^2	10^{12}	11.6 days
Cubic	n^3	10^{18}	32 000 years
Exponential	2^n	10^{100}	10^{86} millennia

Table 3.1: Run times of algorithms of different complexity classes.

the time it actually takes to execute the algorithm on a computer, is that the measure $T(n)$ is system-independent (*i.e.* independent of the speed of the computer on which the algorithm is implemented). In other words, the time complexity $T(n)$ of an algorithm is fixed, regardless of whether one computer has greater processing capabilities than another machine.

Usually a trade-off exists between the time and space complexities of an algorithm, in the sense that attempts at reducing the time complexity causes an increase in the memory required to execute the algorithm (and hence the space complexity), and *vice-versa*. In this book we shall focus (although not exclusively) on issues surrounding only the time complexity of an algorithm for the sake of brevity. The reader should not, however, infer from this convention that space

complexity estimates are unimportant — indeed, the space complexity of an extremely time efficient algorithm often renders the algorithm impractical for large instances of the problem at hand.

- ❖ The reader should now be able to attempt Exercises 3.1–3.2.

3.2 “Big \mathcal{O} ” notation

For many algorithms it is extremely difficult to quantify the exact amount of memory or the exact number of basic operations required during execution — often these numbers are input-specific, governed by recursive calls and/or by conditional structures such as **if...then** statements or **while** loops. Rather than carrying out exhaustive exact counts, it is often sufficient to have a worst-case estimate of these numbers. When evaluating the so-called **worst-case complexity** of an algorithm, this is exactly the approach adopted and one is interested in the largest value that $S(n)$ and/or $T(n)$ can attain for any specific value of n . Moreover, exact upper bounds on $S(n)$ and $T(n)$ are often abandoned in favour of so-called asymptotic upper bounds — bounds that describe the worst-case growth behaviour of the functions $S(n)$ and $T(n)$ as $n \rightarrow \infty$.

The German mathematician **Paul Gustav Heinrich Bachmann** was born in Berlin. From a young age he showed great interest in music but, following a discovery of his mathematical talent by one of his gymnasium teachers, he went on to obtain his doctorate in mathematics (group theory) in 1862. Two years later he completed his *habilitation* (a post-doctoral qualification required in German universities) in Breslau with a paper on complex units. After some years as extraordinary professor at the University of Breslau, he was appointed as professor of mathematics in Münster. In approximately 1890, however, Bachmann divorced from his wife and resigned his professorship. With his second wife he settled in Weimar, where he combined his mathematical writing with composing, playing the piano, and serving as music critic for various newspapers. His main project, however, was a complete survey of the state of number theory which resulted in the authoritative *Zahlentheorie. Versuch einer Gesamtdarstellung dieser Wissenschaft in ihren Hauptteilen* (1892–1923).



Biographic note 6: Paul Bachmann (1837–1920)

We write $f(n) = \mathcal{O}(g(n))$ if there exist constants $c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$ such that

$$0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0, \quad (3.2)$$

in which case we say that the function $g(n)$ is an **asymptotic upper bound** for the function $f(n)$ as $n \rightarrow \infty$. This “big \mathcal{O} ” notation is sometimes referred to as Bachmann-Landau notation after the mathematicians **Paul Bachmann** and **Edmund Landau** who introduced it. Consider, for example, the time complexity

function $T(n) = 2n^2 + n$ associated with [Algorithm 1](#), and let $g_1(n) = n^2$ and $g_2(n) = n^3$. Then it is clear that $T(n)$ and $3 \cdot g_1(n)$ intersect where $n^2 - n = 0$, i.e. at the points $n = 0$ and $n = 1$, and that $0 \leq T(n) \leq 3 \cdot g_1(n)$ for all $n \geq 1$. Hence, by taking $c = 3$ and $n_0 = 1$ in [\(3.2\)](#), we may write $2n^2 + n = \mathcal{O}(n^2)$. Similarly, $T(n)$ and $g_2(n)$ intersect where $n^3 - 2n^2 - n = 0$, i.e. at the points $n = 0$ and $n = 1 \pm \sqrt{2}$, and $0 \leq T(n) \leq 1 \cdot g_2(n)$ for all $n \geq 1 + \sqrt{2} \approx 2.414$. Hence, by taking $c = 1$ and $n_0 = 3$ in [\(3.2\)](#), we also have that $2n^2 + n = \mathcal{O}(n^3)$. In fact, $2n^2 + n = \mathcal{O}(n^k)$ for all $k \geq 2$.

Sometimes the notation $f(n) \prec g(n)$ is used in the literature instead to denote that $f(n) = \mathcal{O}(g(n))$. Perhaps this notation more clearly conveys the *asymptotic* upper bound nature of the function $g(n)$ in the relationship $f(n) = \mathcal{O}(g(n))$. It is left as an exercise to prove the validity of the asymptotic hierarchy

$$1 \prec \ln n \prec n \prec n \ln n \prec n^2 \prec 2^n \prec n! \quad (3.3)$$

which is depicted in [Figure 3.3](#).

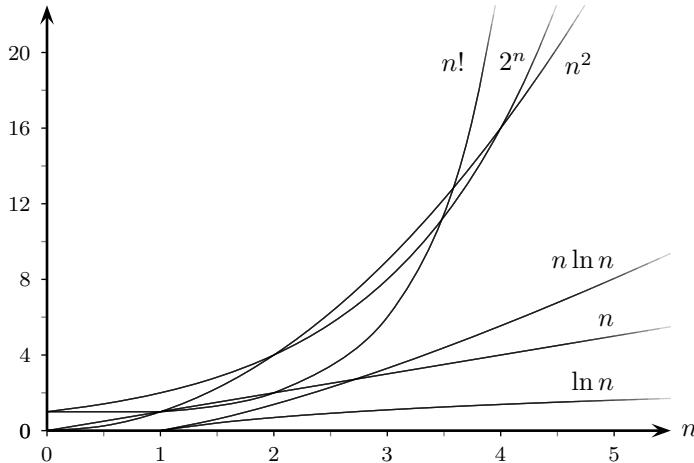


Figure 3.3: Graphical representation of the asymptotic hierarchy [\(3.3\)](#).

Some of the properties of the order notation introduced above are summarised in the following theorem.

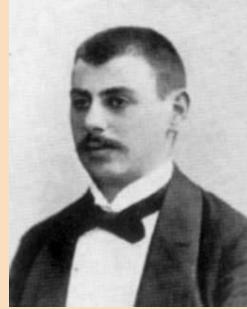
Theorem 3.1 (Properties of “big \mathcal{O} ” order notation) *Suppose f , g , h and ℓ are nonnegative functions on the set of real numbers.*

- (i) *If $f = \mathcal{O}(h)$ and $g = \mathcal{O}(h)$, then $(f + g) = \mathcal{O}(h)$;*
- (ii) *If $f = \mathcal{O}(h)$ and $g = \mathcal{O}(\ell)$, then $(fg) = \mathcal{O}(h\ell)$;*
- (iii) *$f = \mathcal{O}(f)$ (reflexivity);*
- (iv) *If $f = \mathcal{O}(g)$ and $g = \mathcal{O}(h)$, then $f = \mathcal{O}(h)$ (transitivity).*

Proof (i) Let $f(n)$, $g(n)$ and $h(n)$ be functions such that $f(n) = \mathcal{O}(h(n))$ and $g = \mathcal{O}(h(n))$. Then there exist constants $c_f, c_g \in \mathbb{R}^+$ and $n_f, n_g \in \mathbb{N}$ such that $0 \leq f(n) \leq c_f \cdot h(n)$ for all $n \geq n_f$ and such that $0 \leq g(n) \leq c_g \cdot h(n)$ for all $n \geq n_g$. Let $n_0 = \max\{n_f, n_g\}$. Then it follows, by addition of the two inequality chains, that $0 \leq f(n) + g(n) \leq (c_f + c_g) \cdot h(n)$ for all $n \geq n_0$. Hence $0 \leq (f+g)(n) \leq c \cdot h(n)$ for all $n \geq n_0$, where $c = c_f + c_g$, and we conclude that $(f+g)(n) = \mathcal{O}(h(n))$.

(ii)–(iv) Proof of these results are left as exercises. ■

The German mathematician **Edmund Georg Hermann Landau** was born in Berlin, where he obtained his PhD in mathematics (on a 14-page thesis!) from the University of Berlin in 1899 and his habilitation in 1901. He taught at the University of Berlin from 1899 until 1909, when he was appointed as professor of mathematics at the University of Göttingen. Himself Jewish, he was instrumental during the 1920s in establishing the Mathematics Institute at the Hebrew University of Jerusalem. Landau taught himself Hebrew, intending eventually to settle in Jerusalem. In 1927 Landau and his family emigrated to Palestine, where he began to teach mathematics at the Hebrew University. As a result of the Landau family's difficulties adjusting to the primitive living standards then available in Jerusalem and because of a controversy surrounding his nomination for the position of rector of the University, he returned to Göttingen. He remained there until he was ostracized by the Nazi regime in 1933. Thereafter he lectured only outside of Germany. In 1934 he moved to Berlin, where he died in 1938. In 1903 Landau gave a much simpler proof than was then known of the prime number theorem and later produced the first systematic treatment of analytic number theory.



Biographic note 7: Edmund Landau (1877–1938)

A function $f(n)$ is **nondecreasing** if $f(n_1) \geq f(n_2)$ whenever $n_1 \geq n_2$. The functions in the asymptotic hierarchy (3.3) are therefore all examples of nondecreasing functions. We accept, without proof, the following useful result on the asymptotic growth rates of nondecreasing functions.

Theorem 3.2 *For any nondecreasing function $f(n)$,*

$$(f(n))^a = \mathcal{O}(b^{f(n)}),$$

where $a > 0$ and $b > 1$ are real constants.

Theorem 3.2 states that any exponential function grows asymptotically faster than any polynomial function. The significance of **Theorem 3.2** is that it may be used to establish asymptotic upper bounds for a given function with relative ease. For example, if we substitute n for $f(n)$ in **Theorem 3.2**, we obtain, for all constants $a > 0$ and $b > 1$, that

$$n^a = \mathcal{O}(b^n).$$

- ❖ The reader should now be able to attempt Exercises 3.3–3.11, as well as Projects 3.1–3.3.

3.3 A classification scheme for decision problems

An algorithm for which the time complexity is $\mathcal{O}(n^a)$, for some constant $a \in \mathbb{R}^+$, where n denotes the input size to the algorithm as before, is referred to as a **polynomial-time algorithm**, and similarly for the space complexity of the algorithm. An algorithm that is not a polynomial-time algorithm, is called an **exponential-time algorithm**, and similarly for the space complexity.

Decision theory is that branch of complexity theory in which the problem to be solved is interpreted as a binary question that may be answered either **true** or **false**. The class consisting of all decision problems that may be solved by a polynomial-time algorithm is usually denoted P. The class NP (an abbreviation for “nondeterministic polynomial-time”) comprises all decision problems that may be answered **true** by a polynomial-time algorithm, given additional information (known as a **certificate** with respect to the specific instance of the decision problem). The class co-NP is the set of all decision problems which, given additional information (again called a certificate), may be answered **false** by a polynomial-time algorithm. Although a certificate with respect to a decision problem instance may exist, finding this certificate may be difficult. The various classes mentioned above into which a decision problem may be classified, are illustrated schematically in Figure 3.4.

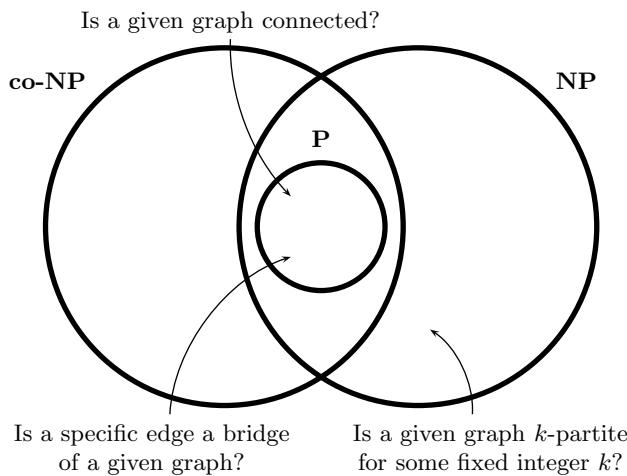


Figure 3.4: The complexity classes P, NP and co-NP, and some decision problem examples.

Consider the following decision problem: “Given a connected graph G of order n , decide whether or not an edge e is a bridge of G .” It is clear that this problem is in co-NP, since it is certainly possible to verify, by means of an algorithm whose worst-case time complexity is a polynomial function of n , that e is not a bridge when alerted to a cycle in G containing the edge e (this cycle is then a co-NP certificate to the decision problem at hand). It is left as an exercise to show that it is also possible to verify, by means of an algorithm whose worst-case time complexity is a polynomial function of n , that e is indeed a bridge of G if given a pair of vertices $u, v \in V(G)$ such that no $u-v$ path exists in $G - e$ (this

pair of vertices then constitutes an NP certificate to the decision problem). Therefore the decision problem is also in the class NP. This decision problem may, in fact, be answered by means of an algorithm whose worst-case time complexity is a polynomial function of n , without additional information (certificate) at all, by evaluating directly whether or not $k(G - e) = 1$. Hence the decision problem is also in the class P.

From the example above, it follows that the classes NP and co-NP intersect and that some subset of the complexity class P falls within this region of intersection. The following theorem confirms this observation and indeed gives a slightly stronger result, stating that no subset of P falls outside the region of intersection between the classes NP and co-NP.

Theorem 3.3 $P \subseteq NP$ and $P \subseteq \text{co-NP}$.

Proof of Theorem 3.3 is left as an exercise. We may at this point be asking ourselves whether perhaps $P = NP$, $NP = \text{co-NP}$ or $P = NP \cap \text{co-NP}$? These are difficult meta-decision problems originally posed by the renowned computer scientist [Stephen Cook](#) during the early 1970s and which remain unanswered to this day. In fact, the problem of whether $P = NP$ is one of the so-called infamous millennium problems. The *Clay Institute* has offered prize money of one million US dollars for the first successful resolution of this problem.

If a decision problem can be solved by a polynomial-time algorithm (*i.e.* if a polynomial-time algorithm is *known* by which the problem may be solved), then we say the problem is **tractable**. Conversely, if no such polynomial-time algorithm is known, then the decision problem is called an **intractable** or hard problem, since it would seem impractical to compute its solution within a reasonable amount of time as the problem input size increases, in view of [Table 3.1](#).

- ❖ The reader should now be able to attempt Exercises [3.12–3.14](#).

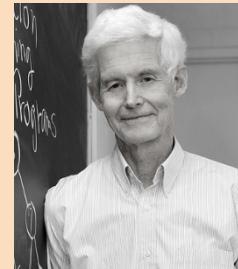
3.4 Polynomial-time reducibility and NP-completeness

Let D_1 and D_2 be two decision problems. Then we say that D_1 is **polynomial-time reducible** to D_2 if there exists an algorithm A_1 capable of solving any instance of D_1 , and which contains as subroutine an algorithm A_2 capable of solving all instances of D_2 , so that A_1 is a polynomial-time algorithm if A_2 is a polynomial-time algorithm. We write $D_1 \rightsquigarrow D_2$ if D_1 is polynomial-time reducible to D_2 . This definition of polynomial-time reducibility therefore implies that $D_1 \in P$ if $D_1 \rightsquigarrow D_2$ and $D_2 \in P$. If $D_1 \rightsquigarrow D_2$ and $D_2 \rightsquigarrow D_1$, then we say that D_1 and D_2 are **polynomial-time equivalent**, and write $D_1 \rightsquigarrow\rightsquigarrow D_2$. The following theorem, whose proof is left as an exercise, establishes a transitivity property for polynomial-time reducibility.

Theorem 3.4 Let D_1 , D_2 and D_3 be three decision problems. If $D_1 \rightsquigarrow D_2$ and $D_2 \rightsquigarrow D_3$, then $D_1 \rightsquigarrow D_3$.

We say that a decision problem D is **NP-hard** if $D_1 \rightsquigarrow D$ for all $D_1 \in NP$. Furthermore, a decision problem D is **NP-complete** if $D \in NP$ and if D is NP-hard. NP-completeness is therefore a more restrictive property than NP-hardness, in the sense that the class of NP-complete decision problems is a subset of the

The renowned American-Canadian computer scientist and mathematician **Stephen Arthur Cook** has made important contributions to the field complexity theory. He obtained his PhD from the University of Michigan in 1966, after which he joined the Department of Mathematics at the University of California, Berkeley as assistant professor. In 1970 he was appointed as associate professor of mathematics and computer science at the University of Toronto, where he was promoted to full professor in 1975 and to distinguished professor in 1985. He formalised the notions of polynomial-time reduction and NP-completeness, proved [Theorem 3.7](#), and was the first to raise the question of whether perhaps $P = NP$? Amongst many other awards, Cook won the prestigious Turing award in 1982. His citation for this award included the following statement: *For his advancement of our understanding of the complexity of computation in a significant and profound way. His seminal paper, The Complexity of Theorem Proving Procedures, ... laid the foundations for the theory of NP-completeness. The ensuing exploration of the boundaries and nature of the NP-complete class of problems has been one of the most active and important research activities in computer science for the last decade.*



Biographic note 8: Stephen Cook (1939–present)

class of NP decision problems. NP-complete problems may be thought of as the most difficult NP decision problems in our classification scheme, in the sense that they are computationally at least as hard to solve as any other problem in NP. The notion of NP-completeness may also be used to shed more light on the relation between the classes P, NP and co-NP of decision problems, as described in the following theorem.

Theorem 3.5 *Let D be a decision problem.*

- (i) *If D is NP-complete and $D \in P$, then $P = NP$;*
- (ii) *If D is NP-complete and $D \in \text{co-NP}$, then $P = \text{co-NP}$.*

Proof (i) Note that

$$P \subseteq NP \tag{3.4}$$

by [Theorem 3.3](#). Now suppose D is an NP-complete decision problem which is also in P. Then $D \in NP$ and $D_1 \rightsquigarrow D$ for all $D_1 \in NP$. But this means that

$$NP \subseteq P \tag{3.5}$$

by definition of polynomial-time reducibility. By combining (3.4) and (3.5) it follows that $NP = P$.

(ii) Proof of this result is similar to that of (i), and is left as an exercise. ■

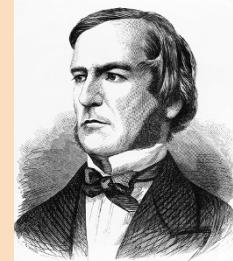
The search for a decision problem D satisfying any one of the requirements of [Theorem 3.5\(i\)](#) or [\(ii\)](#) has been a subject of intense research interest for a considerable number of years. Many complexity theorists believe that no such decision problem exists — if such a decision problem can be found or if it can be established that no such decision problem exists, it would indeed be a deep and significant result!

The next theorem, whose proof is again left as an exercise, provides a method for establishing the NP-completeness of a decision problem.

Theorem 3.6 *Let D_1 and D_2 be two decision problems. If $D_1 \in \text{NP}$, D_2 is NP-complete and $D_2 \rightsquigarrow D_1$, then D_1 is NP-complete.*

If we have a starting point for [Theorem 3.6](#) (a prototype decision problem D_2 as described in the theorem), then it is possible to build up a collection of NP-complete problems by invoking the theorem. Before such a prototype NP-complete problem may, however, be introduced, we need some preliminary definitions from the field of boolean algebra, named after its inventor, [George Boole](#).

The mathematician, logician and philosopher **George Boole** was born in Lincolnshire, England. At age 16 he became the breadwinner for his father and three siblings by taking up a position as a school teacher in Doncaster. At age 19 he established his own school in Lincoln and held various school teaching positions thereafter. He was essentially a self-taught mathematician. From 1838 onwards, Boole started making contact with various sympathetic British mathematicians. He studied algebra in the form of symbolic methods and began to publish research papers, but it was not until 1849 that his status as mathematician was recognised by his appointment as the first professor of mathematics at Queen's College, Cork (now University College Cork) in Ireland. His mathematical contributions are many and varied, including notable work in the fields of differential equations, real analysis, symbolic logic, and probability theory. He is today perhaps best known for the algebra he developed over binary variables.



Biographic note 9: George Boole (1815–1865)

A **boolean variable** is a variable that can take on one of two values; we shall assume the two values to be **true** and **false**. If x_1, \dots, x_r are r boolean variables, then the **negation** of x_i , denoted by \bar{x}_i , is another boolean variable which has the value **false** if and only if x_i has the value **true**. A **literal** formed from the above set of r boolean variables is either one of the boolean variables x_i , or its negation \bar{x}_i , whilst an **s -clause** formed from the same set is a conjunction of s literals, conjoined by the binary operation **or**, denoted by \vee and defined in [Table 3.2](#) (typically $s \leq r$). An example of a 2-clause is $x_1 \vee x_3$, while $x_1 \vee \bar{x}_1 \vee x_2$ is an example of a 3-clause.

A boolean function of the variables x_1, \dots, x_r , denoted by $f(x_1, \dots, x_r)$, is said to be in **s -conjunctive normal form** if the function consists of a number of s -clauses formed from the r variables, conjoined by the binary operation **and**, denoted by \wedge

a	b	$a \vee b$	$a \wedge b$
false	false	false	false
false	true	true	false
true	false	true	false
true	true	true	true

Table 3.2: Truth table for the binary operations **and** (\wedge) and **or** (\vee).

and also defined in [Table 3.2](#). A boolean function $f(x_1, \dots, x_r)$ in s -conjunctive normal form is **satisfiable** if there exists an assignment of values to the boolean variables x_1, \dots, x_r for which the function f evaluates to **true**.

The function $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$ is an example of a boolean function in 2-conjunctive normal form of the variables x_1 and x_2 . This function is clearly satisfiable, since the function evaluates to **true** for the values $x_1 = \text{true}$ and $x_2 = \text{false}$ of the boolean variables, according to [Table 3.2](#). On the other hand, the function $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$ is an example of a boolean function in 2-conjunctive normal form that is *not* satisfiable, because for a boolean function in conjunctive normal form to evaluate to **true** by some assignment of values to the boolean variables, each of its clauses must evaluate to **true** for the specific assignment — but the four clauses of the last example clearly cannot *all* evaluate to **true** simultaneously, for any assignment of values to the variables.

We are now in a position to introduce our prototype NP-complete decision problem, called **s -SAT**. This problem may be formulated as follows: “Given a boolean function f of r (boolean) variables in s -conjunctive normal form, does there exist an assignment of values to these variables for which f evaluates to **true**?” For example, an instance of 3-SAT would be to ask whether an assignment of values to the four boolean variables x_1 , x_2 , x_3 and x_4 exists for which the boolean function

$$f(x_1, x_2, x_3, x_4) = (\overbrace{x_1 \vee x_2 \vee \bar{x}_4}^{X_1}) \wedge (\overbrace{x_2 \vee \bar{x}_3 \vee x_4}^{X_2}) \wedge (\overbrace{\bar{x}_1 \vee x_2 \vee \bar{x}_3}^{X_3}) \wedge (\overbrace{x_1 \vee x_3 \vee x_4}^{X_4})$$

in 3-conjunctive normal form (comprising four clauses) evaluates to **true**? We may certainly try all $2^4 = 16$ possible assignments of values to the four variables until an assignment is found for which f evaluates to **true** in an attempt at solving this instance of 3-SAT as shown in [Table 3.3](#), but there seems to be no better, structured approach toward solving this instance of the problem and, indeed, s -SAT in general. We state, without proof, the next important result, which was established independently by [Cook \[2\]](#) in 1971 and [Levin \[5\]](#) in 1973.

Theorem 3.7 (Cook-Levin Theorem [2, 5]) *s -SAT is NP-complete.*

The significance of s -SAT is that it serves as a starting point (the decision problem D_2) for proving the NP-completeness of many hard decision problems (via [Theorem 3.6](#)), as mentioned above. Let us demonstrate this by an example. Let $D_{\text{clique}}(G, k)$ denote the decision problem of, given a graph G of order $n \geq k$, deciding whether or not G contains a k -clique. It is not always easy to spot with the human eye whether a given graph contains a k -clique. Consider, for example, the graph $G_{3,1}$ in [Figure 3.5\(a\)](#), without looking at [Figure 3.5\(b\)](#). Does this graph contain a 5-clique? The answer is yes, as demonstrated in [Figure 3.5\(b\)](#) — this result may be achieved by a process of elimination. Of course $D_{\text{clique}}(G, k)$ is

x_1	x_2	x_3	x_4	X_1	X_2	X_3	X_4	f
false	false	false	false	true	true	true	false	false
false	false	false	true	false	true	true	true	false
false	false	true	false	true	false	true	true	false
false	false	true	true	false	true	true	true	false
false	true	false	false	true	true	true	false	false
false	true	false	true	true	true	true	true	true
false	true	true	false	true	true	true	true	true
false	true							
true	false	false	false	true	true	true	true	true
true	false	false	true	true	true	true	true	true
true	false	true	false	true	false	false	true	false
true	false	true	true	true	true	false	true	false
true	true	false	false	true	true	true	true	true
true	true	false	true	true	true	true	true	true
true	true	true	false	true	true	true	true	true
true								

Table 3.3: Truth table for the 3-SAT example boolean function $f(x_1, x_2, x_3, x_4) = \underbrace{(x_1 \vee x_2 \vee \bar{x}_4)}_{X_1} \wedge \underbrace{(x_2 \vee \bar{x}_3 \vee x_4)}_{X_2} \wedge \underbrace{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)}_{X_3} \wedge \underbrace{(x_1 \vee x_3 \vee x_4)}_{X_4}$.

in NP, because a certificate is the k -clique itself. Observe that $D_{\text{clique}}(G, k)$ is not obviously a member of co-NP. Verifying that some substructure is *not* present in a mathematical structure seems to be more difficult than verifying that it *is* present.

Proving that $D_{\text{clique}}(G, k)$ is NP-complete is achieved by polynomial-time reducing an instance of 3-SAT to an instance of $D_{\text{clique}}(G, k)$. In order to do so, let f be a boolean function comprising k clauses in 3-conjunctive normal form. We map the function f onto a k -partite graph G as follows: The partite sets of G have three vertices each, which we shall call *triples*. Each triple corresponds to one of the clauses in f , and each vertex in a triple corresponds to a literal in the associated clause. Label each vertex of G with its corresponding literal in f . All inter-partite set edges are present in G , except that no edge is present be-

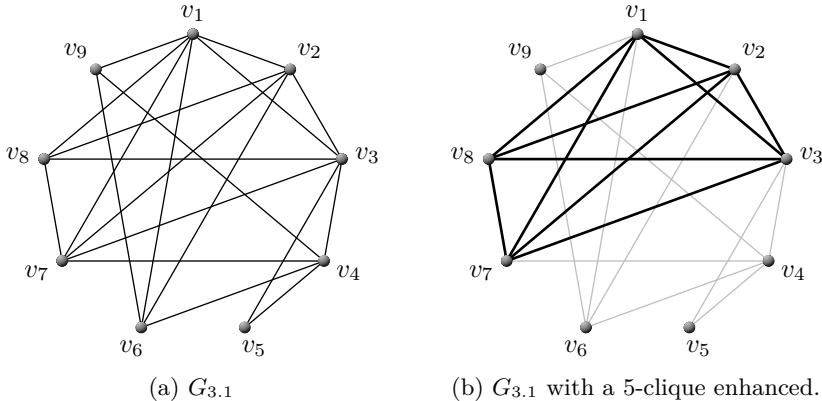


Figure 3.5: A 5-clique in the graph $G_{3.1}$.

tween two vertices corresponding to literals which are negations of each other. **Figure 3.6** illustrates the graph construction of $G_{3,2}$ for the boolean function $f(x_1, x_2) = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$.

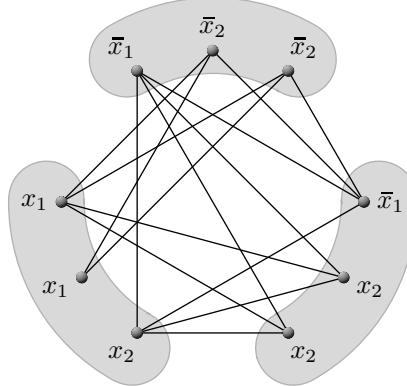


Figure 3.6: Construction of the multipartite graph $G_{3,2}$ from the boolean function $f(x_1, x_2) = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$.

We now show that f is satisfiable if and only if G has a k -clique. Suppose that f has a satisfying assignment. In that satisfying assignment, at least one literal has the value **true** in every clause. Arbitrarily select any literal in each triple of G that corresponds to a literal with value **true** in the satisfying assignment. These selected vertices form a k -clique. Each pair of selected vertices is adjacent, because no pair are the negations of each other. Furthermore, they cannot be from the same triple, because only one vertex per triple was selected. Therefore, G contains a k -clique.

Conversely, suppose that G has a k -clique. No two vertices in the clique occur in the same triple, because vertices in the same triple are not adjacent. Therefore, each of the k triples contains exactly one of the k -clique vertices. We assign values of **true** to the variables in f so that each literal labelling a clique vertex evaluates to **true**. Doing so is always possible, because two vertices corresponding to literals that negate each other are not adjacent and hence both cannot be in a clique. This assignment of values to the variables satisfies f , because each triple contains a clique vertex and hence each clause contains a literal that is assigned the value **true**. Therefore, f is satisfiable. We conclude that $3\text{-SAT} \rightsquigarrow D_{\text{clique}}(G, k)$ and we have the following result by [Theorem 3.6](#).

Theorem 3.8 $D_{\text{clique}}(G, k)$ is NP-complete.

It is possible to refine the graphical representation of the classification scheme in [Figure 3.4](#) to include the notion of NP-completeness, as shown in [Figure 3.7](#).

For more information on the notion of NP-completeness, the reader is referred to the landmark book by [Garey and Johnson \[3\]](#).

❖ The reader should now be able to attempt Exercises [3.15–3.19](#).

Leonid Anatolievich Levin was born in Dnipropetrovsk, Ukrainian Socialist Soviet Republic. He studied towards his master's degree at the University of Moscow under the supervision of Andrey Kolmogorov during the period 1970–1972. After researching algorithmic problems of information theory at the Moscow Institute of Information Transmission of the National Academy of Sciences during the period 1972–1973, and occupying a position at the Moscow National Research Institute of Integrated Automation for the Oil and Gas Industry during the period 1973–1977, he emigrated to the United States of America in 1978. He obtained his PhD at the Massachusetts Institute of Technology in 1979 and is currently professor of computer science at Boston University, a position he has held since 1980. He is well known for his work in algorithmic complexity. In 1973, he proved [Theorem 3.7](#) independently of [Stephen Cook](#). Levin was awarded the prestigious Knuth Prize in 2012 for his formalisation of the notion of NP-completeness and the development of the idea of average-case algorithmic complexity.



Biographic note 10: Leonid Levin (1948–present)

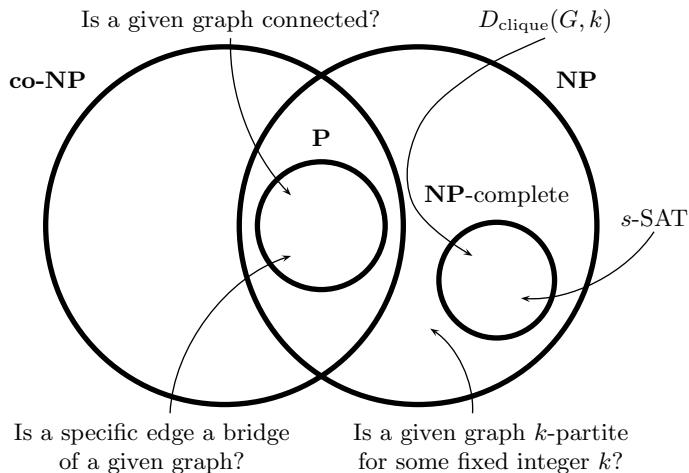


Figure 3.7: The complexity classes P, NP, co-NP and NP-complete, and some decision problem examples.

3.5 Decision problems vs. computation problems

In the remainder of this book we shall not so much be interested in solving *decision* problems. Our main concern will rather be to solve a variety of graph theoretic *computation* problems. A **computation problem** is one which has a real number (or a collection of real numbers) as solution instead of a mere binary value — in fact, we shall most often restrict ourselves to solving computation problems that have a single natural number as solution. Decision problems may therefore be seen

as special cases of computation problems. Computation problems may, however, often be solved efficiently in terms of algorithmic procedures for solving related decision problems, as we now demonstrate by an example.

The **clique number** of a graph G , denoted by $\omega(G)$, is the largest natural number k for which G contains a k -clique as subgraph. Since the graph $G_{3.1}$ in [Figure 3.5](#) contains a 5-clique, it follows, for example, that $\omega(G_{3.1}) \geq 5$. But since there are exactly six vertices of degree at least 5 in $G_{3.1}$ (the vertices v_1, v_2, v_3, v_4, v_7 and v_8) and these vertices do not form a clique in $G_{3.1}$, it also follows that $\omega(G_{3.1}) < 6$, and so $\omega(G_{3.1}) = 5$.

Let $C_{\text{clique}}(G)$ denote the computation problem of finding the clique number $\omega(G)$ of a given graph G . If the graph has order n , then it is clear that $\omega(G) \in [n]$. The value of $\omega(G)$ may be determined by a so-called *interval halving scheme* in terms of the NP-complete decision problem $D_{\text{clique}}(G, k)$ described in [Section 3.4](#). This decision problem is called the **underlying decision problem** to the computation problem $C_{\text{clique}}(G)$. Consider [Algorithm 3](#) for solving $C_{\text{clique}}(G)$ in terms of a procedure for solving $D_{\text{clique}}(G, k)$.

Algorithm 3: Clique number (Iterative)

Input : A graph G of order n .
Output: The clique number $\omega(G)$ of G .

```

1 if ( $D_{\text{clique}}(G, n) = \text{true}$ ) then
2   | print  $n$ , stop
3 else
4   |  $\ell \leftarrow 1, r \leftarrow n$ 
5   |  $m \leftarrow \lfloor (\ell + r)/2 \rfloor$ 
6   | if ( $D_{\text{clique}}(G, m) = \text{true}$ ) then
7     |   |  $\ell \leftarrow m$ 
8   | else
9     |   |  $r \leftarrow m$ 
10  | if ( $r - \ell = 1$ ) then
11    |   | print  $\ell$ , stop
12  | else
13    |   | go to Step 5

```

The rationale behind [Algorithm 3](#) is that it maintains at each iteration an interval on the real line with left and right endpoints the integers ℓ and r respectively, such that $D_{\text{clique}}(G, \ell) = \text{true}$ and $D_{\text{clique}}(G, r) = \text{false}$, implying that $\ell \leq \omega(G) < r$. This interval is halved during each iteration (hence the term interval halving scheme) by determining an integer m , which is the largest integer not exceeding the midpoint $(\ell + r)/2$ of the interval $[\ell, r]$. If $r - \ell = 1$, then the clique number of G has been found, since $\ell \leq \omega(G) < \ell + 1$ implies that $\omega(G) = \ell$, at which point the algorithm terminates. Otherwise the algorithmic search continues in the left half interval if $D_{\text{clique}}(G, m) = \text{false}$ or in the right half interval if $D_{\text{clique}}(G, \ell) = \text{true}$, so as to ensure that $D_{\text{clique}}(G, \ell) = \text{true}$ and $D_{\text{clique}}(G, r) = \text{false}$ in the new half interval.

Let $\ell_0 = 1$ and $r_0 = n$. For $i \geq 1$, if ℓ_i and r_i denote those values of respectively ℓ and r during iteration i of [Algorithm 3](#), then it follows that

$$\begin{aligned} r_i - \ell_i &= r_{i-1} - \overbrace{\left\lceil \frac{\ell_{i-1} - r_{i-1}}{2} \right\rceil}^{m_i} \\ &\leq r_{i-1} - \frac{\ell_{i-1} - r_{i-1} - 1}{2} = \frac{r_{i-1} + \ell_{i-1} + 1}{2} \end{aligned} \quad (3.6)$$

if $D_{\text{clique}}(G, m_i) = \text{true}$, or that

$$\begin{aligned} r_i - \ell_i &= \overbrace{\left\lceil \frac{\ell_{i-1} + r_{i-1}}{2} \right\rceil}^{m_i} - \ell_{i-1} \\ &\leq \frac{r_{i-1} - \ell_{i-1}}{2} \end{aligned} \quad (3.7)$$

if $D_{\text{clique}}(G, m_i) = \text{false}$. By combining (3.6) and (3.7) it therefore follows that

$$r_i - \ell_i \leq \frac{r_{i-1} - \ell_{i-1} + 1}{2}$$

in both cases. By iteratively applying this inequality j times, it follows that

$$r_i - \ell_i \leq \frac{r_{i-j} - \ell_{i-j} + 2^j - 1}{2^j}.$$

Hence, for $j = i$, we obtain the bound

$$r_i - \ell_i \leq \frac{r_0 - \ell_0 + 2^i - 1}{2^i} = \frac{n - 2 + 2^i}{2^i}.$$

Now the process of interval halving will continue as long as

$$2 \leq r_i - \ell_i \leq \frac{n - 2 + 2^i}{2^i},$$

or, equivalently, as long as $2^i \leq n - 2$, that is, as long as $i \leq \log_2(n - 2)$, which means that [Algorithm 3](#) has a worst-case time complexity of $\mathcal{O}(\log n)$ if execution of the procedure for solving the decision problem $D_{\text{clique}}(G, \ell)$, as well as integer addition, integer division, variable assignment and **if** comparisons are taken as basic operations. Therefore,

$$C_{\text{clique}}(G) \rightsquigarrow D_{\text{clique}}(G, k) \quad (3.8)$$

if we generalise the notion of polynomial-time reducibility to encompass not only decision problems, but also computation problems in the obvious way. Under the convention that a computation problem is called polynomial-time solvable (NP-complete, respectively) if the underlying decision problem is in P (NP-complete, respectively), we have therefore shown that $C_{\text{clique}}(G)$ is NP-complete in view of (3.8) and [Theorem 3.8](#). Note that the computation problem $C_{\text{clique}}(G)$ would have been polynomial-time solvable had we instead shown $D_{\text{clique}}(G, k)$ to be a member of P, in view of (3.8).

The above example serves to demonstrate that our classification scheme of [Section 3.3](#) and the notions of polynomial-time reducibility and NP-completeness of [Section 3.4](#) are not only applicable to decision problems, as was the case in [Sections 3.3](#) and [3.4](#), but may be generalised to accommodate computation problems.

- ❖ The reader should now be able to attempt [Project 3.5](#).

The mathematician **Leonardo Fibonacci**, also known as Leonardo of Pisa, was born in Pisa in what is today Italy. He was educated in North Africa where his father held a diplomatic post and he travelled widely with his father. His travels ended around the year 1200 when he returned to Pisa. There he wrote a number of important texts, including a commentary on Euclid's *Elements*, which played an important role in reviving ancient mathematical skills. He also made significant contributions of his own, such as his famous search technique. One of his texts, *Liber abaci*, was published in 1202 and was based on the arithmetic and algebra that Fibonacci had encountered during his travels. In the first part of this book, which was later widely copied and imitated, he introduced the Hindu-Arabic place-valued decimal system and the use of Arabic numerals into Europe. The second part contained a large collection of problems aimed at merchants, relating to the price of goods, how to calculate profit on transactions and how to convert between the various currencies in use in the Mediterranean countries of the time. The following problem in the third part of the book led to the introduction of the sequence for which he is best remembered today. *A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive?* The resulting sequence, to which an entire quarterly journal is today devoted, is 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...



Biographic note 11: Leonardo Fibonacci (c 1170–c 1250)

Exercises

- 3.1 The Fibonacci numbers are the sequence of positive integers defined by the recursion formula $f_i = f_{i-1} + f_{i-2}$ ($i = 2, 3, 4, \dots$), stating that each entry in the sequence is the sum of the previous two sequence entries, and taking the seeds $f_0 = 0$ and $f_1 = 1$ as starting point. The beginning of the sequence of numbers is therefore 0, 1, 1, 2, 3, 5, 8, 13, 21, ... This sequence is attributed to the twelfth-century Italian mathematician **Leonardo Fibonacci** and has a surprising number of applications in nature. [Algorithm 4](#) may be used to determine the i -th Fibonacci number. What, in terms of the parameter i , is the worst-case space and time complexities of this algorithm?

Algorithm 4: Fibonacci numbers (Iterative)

Input : An integer $i \geq 0$.
Output : The i -th Fibonacci number f_i .

```

1 if  $i = 0$  then
2   fib  $\leftarrow 0$ 
3 else if  $i = 1$  then
4   fib  $\leftarrow 1$ 
5 else
6   fib  $\leftarrow 1$ , prev_fib  $\leftarrow 0$ 
7   for  $j = 1$  to  $i$  do
8     temp  $\leftarrow$  fib
9     fib  $\leftarrow$  fib + prev_fib
10    prev_fib  $\leftarrow$  temp
11 print fib

```

3.2 Prove, by induction, that the sequence $H_n = 2^n - 1$ ($n \in \mathbb{N}_0$) satisfies the *Towers of Hanoi* recurrence relation (3.1).

3.3 Suppose that

- (a) i basic operations $[T(n) = \mathcal{O}(n^2)]$;
- (b) i^2 basic operations $[T(n) = \mathcal{O}(n^3)]$;
- (c) $\binom{n}{i}$ basic operations $[T(n) = \mathcal{O}(2^n)]$;
- (d) 2^i basic operations $[T(n) = \mathcal{O}(2^{n+1})]$;
- (e) $i 2^i$ basic operations $[T(n) = \mathcal{O}(n 2^{n+1})]$;
- (f) $\lfloor i^k \log_2 i \rfloor$ basic operations $[T(n) = \mathcal{O}(n^{k+1} \log_2 n)]$;
- (g) $\lfloor \log_2 i \rfloor$ basic operations $[T(n) = \mathcal{O}(n \log_2 n)]$

are performed during iteration i of a **for** loop that runs from $i = 1$ to n in an algorithm. Show that the time complexity $T(n)$ of the entire **for** loop is as stated in the square brackets above.

3.4 Which of the following functions grows asymptotically faster as $n \rightarrow \infty$:

- (a) $\ln n$ or $\ln(n^2)$;
- (b) \sqrt{n} or $(\ln n)^4$;
- (c) $n 2^n$ or 3^n ?

Motivate carefully in each case.

3.5 Use the definition of “big \mathcal{O} ” notation directly to prove the validity of (3.3).

3.6 Suppose $f(n) = \mathcal{O}(h(n))$ and $g(n) = \mathcal{O}(i(n))$. Prove or disprove the following statements:

- (a) $f(n) - g(n) = \mathcal{O}(h(n) - i(n))$;
- (b) $f(n)/g(n) = \mathcal{O}(h(n)/i(n))$.

3.7 Prove Theorem 3.1(ii)–(iv).

3.8 Prove, by invoking Theorem 3.2, that $(\log_A n)^B = \mathcal{O}(n)$ for any real constants $A > 1$ and $B > 0$.

- 3.9 Generalise the result of [Exercise 3.8](#) by using [Theorem 3.2](#) to prove that $(\log_A n)^B = \mathcal{O}(n^C)$ for any real constants $A, B, C > 0$.
- 3.10 Show that $\log_a n = \mathcal{O}(\log_b n)$ for any real constants $a, b > 0$, and hence that logarithm bases are of no consequence in the “big \mathcal{O} ” notation.
- 3.11 Prove that if $p(n)$ is a polynomial in n , then $\log_a p(n) = \mathcal{O}(\log n)$ for any real constant $a > 0$.
- 3.12 Prove [Theorem 3.3](#). (Hint: Consider the possibility of an empty certificate in the case of the class P.)
- 3.13 Consider the decision problem $D_{\text{multipartite}}(G, k)$ of deciding whether or not a given *labelled* graph G of order n is k -partite ($k \leq n$), together with the following (brute force) procedure for solving it: All partitions of $V(G)$ into k parts are first generated. Then each partition is tested to determine whether the parts thereof are valid partite sets for G — if so, the partition is called *successful*; if not, the partition is called *unsuccessful*. If a successful partition is encountered, then the answer to the decision problem $D_{\text{multipartite}}(G, k)$ is **true**; otherwise it is **false**. Is this procedure a polynomial-time algorithm for $D_{\text{multipartite}}(G, k)$? Explain.
- 3.14 Consider the computation problem $C_{\text{multipartite}}(G)$ of determining the smallest value $k \in [n]$ for which a given *labelled* graph G of order n is k -partite. This computation problem has the associated decision problem $D_{\text{multipartite}}(G, k)$ of [Exercise 3.13](#).
- Can the time complexity of the procedure in [Exercise 3.13](#) alone be used to determine whether or not $C_{\text{multipartite}}(G) \in \text{P}$? Explain.
 - Is $C_{\text{multipartite}}(G) \in \text{NP}$? Motivate.
- 3.15 Prove [Theorem 3.4](#).
- 3.16 Prove [Theorem 3.5\(ii\)](#).
- 3.17 Which of the following boolean functions of the variables x_1, x_2, x_3, x_4 are in k -conjunctive normal form, for some $k \in \mathbb{N}$? If a function is in k -conjunctive normal form, specify the value of k and determine whether the function is satisfiable. If not, explain why not:
- $x_1 \wedge x_3 \wedge \bar{x}_4$;
 - $x_1 \wedge x_3 \vee \bar{x}_4$;
 - $(x_1 \vee \bar{x}_2) \wedge (x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_4)$;
 - $(x_1 \vee \bar{x}_1) \wedge (x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_4)$;
 - $(x_1 \vee \bar{x}_2) \wedge (x_2 \wedge x_3) \wedge (\bar{x}_1 \vee \bar{x}_4)$;
 - $(x_1 \vee \bar{x}_2) \wedge (x_2 \vee x_3) \vee (\bar{x}_1 \vee \bar{x}_4)$;
 - $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4)$;
 - $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_4)$.
- 3.18 Produce a graphical representation of the 4-partite graph G that is associated with the boolean function

$$f(x_1, x_2, x_3) = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

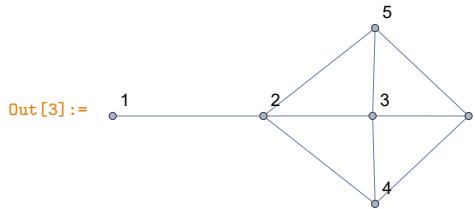
in 3-conjunctive normal form in our polynomial-time reduction of 3-SAT to the decision problem $D_{\text{clique}}(G, 4)$, as described in [Section 3.4](#).

- 3.19 Consider the following (brute force) procedure for solving the decision problem $D_{\text{clique}}(G, k)$ of [Section 3.4](#) for a graph G of order n : All subsets of $V(G)$ with cardinality not exceeding k are first generated — there are $\mathcal{O}(n^k)$ such subsets altogether (why?). Then the subgraphs of G induced by each of these subsets are tested for completeness. If a complete (sub)graph is encountered, then the answer to the decision problem $D_{\text{clique}}(G, k)$ is **true**; otherwise it is **false**. Why is this procedure not a polynomial-time algorithm for $D_{\text{clique}}(G, k)$?

Computer exercises

Iterative algorithms are easily implemented in [MATHEMATICA](#). The syntax of a **for** loop is **For[start, test, incr, body]** in [MATHEMATICA](#). The field **start** is typically used for counter initialisation purposes, whilst the field **test** usually contains a stopping criterion for the loop. The field **incr** may be used to specify the manner in which the **for** loop counter is incremented, whilst the field **body** contains one or more commands to be executed repeatedly. In general, the command executes **start**, then repeatedly evaluates **body** and **incr** until **test** fails to yield a value **True**. Double **for** loops may be implemented by placing the whole inner loop in the field **body** of the outer loop. For example, [Algorithm 1](#) may be implemented for the graph $G_{1,29}$ by the command sequence

```
In[1]:= A = {{0, 1, 0, 0, 0, 0}, {1, 0, 1, 1, 1, 0}, {0, 1, 0, 1, 1, 1}, {0, 1, 1, 0, 0, 1}, {0, 1, 0, 0, 1, 0}};
In[2]:= d = {};
In[3]:= G = AdjacencyGraph[A, VertexLabels -> "Name"]
```



```
In[4]:= MatrixForm[A]
Out[4]:= 

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

In[5]:= For[i = 1, i <= 6, i++,
d = Insert[d, 0, i];
For[j = 1, j <= 6, j++,
d[[i]] = d[[i]] + A[[i, j]]
]
]
In[6]:= Print["d = ", d]
Out[6]:= d = {1,4,4,3,3,3}
In[7]:= VertexDegree[G]
Out[7]:= {1, 4, 4, 3, 3, 3}
```

where the last built-in **MATHEMATICA** command `VertexDegree[•]` is merely included for validation purposes. The command `Insert[list, elem, n]` is used to insert the element `elem` at position `n` of the list `list`. Also note that double square brackets are used to reference list entries. For example, `d[[i]]` denotes the *i*-th member of the list `d`. A command similar to the **for** loop is the **while** loop. The syntax of the **while** loop is `While[test, body]`; here the commands in the field `body` are repeated until the test in the field `test` yields the boolean value **False**. An **if** statement in **MATHEMATICA** has the syntax `If[cond, t, f]`, performing the commands in the field `t` if the condition in the field `cond` evaluates to **True**, or the commands in the field `f` if the condition evaluates to **False**.

Recursive algorithms are best implemented as functions in **MATHEMATICA**. Recall that the command `Range[n]` creates the list $\{1, \dots, n\}$, while the command `Length[S]` returns the length of a list S , *i.e.* the number of elements in S . Furthermore, the command `Last[S]` returns the last element of the list S , while the command `Print[str]` may be used to print the string `str` to screen. [Algorithm 2](#), the procedure for solving the Towers of Hanoi puzzle, may be loaded into the memory of **MATHEMATICA** by executing the command

```
In [8]:= Hanoi[S_, i_, j_] :=
  If[Length[S] == 1,
    Print["Move disc ", S, " from peg ", i, " to peg ", j],
    Hanoi[Range[Length[S] - 1], i, 3 - i - j];
    Hanoi[{Last[S]}, i, j];
    Hanoi[Range[Length[S] - 1], 3 - i - j, j]
  ]
```

which generates no output by itself. Execution of the newly defined command

```
In [9]:= Hanoi[Range[3], 0, 2]
```

however, then generates the following output:

```
Out[9]:= Move disc {1} from peg 0 to peg 2
          Move disc {2} from peg 0 to peg 1
          Move disc {1} from peg 2 to peg 1
          Move disc {3} from peg 0 to peg 2
          Move disc {1} from peg 1 to peg 0
          Move disc {2} from peg 1 to peg 2
          Move disc {1} from peg 0 to peg 2
```

as verification of [Figure 3.2](#).

The commands `And[x1, x2, ...]` and `Or[x1, x2, ...]` may be used to evaluate the conjointment of boolean variables x_1, x_2, \dots by **and** (\wedge) and **or** (\vee) operations, respectively. Furthermore, the command `Not[x]` yields the negation of the boolean variable x , whilst the command `FindInstance[expr, {x1, x2, ...}, Booleans, n]` may be used to find n assignments of the boolean variables x_1, x_2, \dots for which the expression `expr` evaluates to **True**; if no such assignment exists (or if fewer than the requested number of instances exist), then the empty assignment `{}` is returned. For example, the command

```
In [10]:= FindInstance[And[Or[x1, x2, Not[x4]], Or[x2, Not[x3], x4], Or[Not[x1], x2, Not[x3]], Or[x1, x3, x4]], {x1, x2, x3, x4}, Booleans, 4]
```

yields the output:

```
Out[10]:= {{x1 -> True, x2 -> True, x3 -> True, x4 -> True}, {x1 -> True, x2 -> True, x3 -> True, x4 -> False}, {x1 -> True, x2 -> True, x3 -> False, x4 -> True}, {x1 -> True, x2 -> True, x3 -> False, x4 -> False}}
```

which may be verified by means of [Table 3.3](#).

Finally, the command `MemberQ[list, elem]` yields the boolean value `True` if `elem` is an element of the list `list` or `False` otherwise, whilst the command `Sort[list]` sorts the list `list` in canonical order (in nondecreasing order, if the list contains numerical values). A list of numerical values may also be sorted in nonincreasing order by the command `Sort[list, Greater]`. For example, returning to the list of degree values obtained at the start of this section, the commands

```
In[11]:= Sort[d]
In[12]:= Sort[d, Greater]
In[13]:= MemberQ[d, 0]
In[14]:= MemberQ[d, 1]
```

yield the output:

```
Out[11]:={1, 3, 3, 3, 4, 4}
Out[12]:={4, 4, 3, 3, 3, 1}
Out[13]:=False
Out[14]:=True
```

- ❖ The reader should now be able to repeat [Exercise 3.17](#) using [MATHEMATICA](#). The reader should also be able to implement all algorithms in this chapter in [MATHEMATICA](#).

Projects

This section contains five projects. The focus in the first three of these projects falls on algorithmic design and the associated analysis of algorithmic complexity. Three fundamental problems are considered in these first three projects, namely to test for list membership of a given element, sorting a list in nondecreasing order, and computing the greatest common divisor of two given natural numbers. The fourth project is concerned with establishing the tractability of computing the (ordered) degree sequence of a graph, while the reader is guided in the last project toward proving that the problem of computing the cardinality of a smallest vertex cover of a general graph is NP-complete.

Project 3.1: List membership

In many graph theoretic algorithms the procedures of

- finding the minimum value from amongst a list of integers, and
- determining whether a given number is an element of a list of integers

are important components or sub-procedures. In this project we explore these procedures and their algorithmic complexities.

Tasks

1. Write down an algorithm with worst-case time complexity $T(n) = \mathcal{O}(n)$ for finding the minimum value in an unsorted list of n integers without sorting the list, and apply your algorithm to the lists $L_1 : 3, 8, 5, 1, 4, 6, 1, 7$ and $L_2 : 3, 8, 5, 2, 4, 6, 1, 7$ in order to convince yourself of the correct working of the algorithm. Let us call this algorithm the **smallest member algorithm**. Verify that your smallest member algorithm has the desired time complexity — be sure to specify which operations you consider basic operations. What is the space complexity $S(n)$ of the smallest member algorithm?
2. (a) Write an algorithm with worst-case time complexity $T(n) = \mathcal{O}(n)$ for determining whether a given number is an element of a list of n integers, by sequentially searching through the list — let us call this algorithm the **sequential membership algorithm**. Test your sequential membership algorithm by using the algorithm to decide whether or not the numbers 2 and 3 (respectively) are members of the list L_1 above. Verify that your algorithm has the desired time complexity — be sure to specify which operations you consider basic operations. What is the space complexity of the sequential membership algorithm?
(b) The time complexity of the sequential membership algorithm is not best possible when determining whether a given number is an element of a *sorted* list of integers. Consider, for example, [Algorithm 5](#) (which we call the **binary search membership algorithm**) for performing this task when the list of n (distinct) integers is sorted in increasing order. Apply [Algorithm 5](#) to the list $L_3 : 1, 2, 3, 5, 6, 7, 8$ in order to decide whether the numbers 4 and 5 (respectively) are members of the list. What is the worst-case time complexity $T(n)$ of [Algorithm 5](#) (if integer addition, integer subtraction, integer division, variable assignment and **if** comparisons are all taken as basic operations)? Also, what is the worst-case space complexity $S(n)$ of [Algorithm 5](#)?

Algorithm 5: Binary search membership (Iterative)

Input : A list $L : \ell_1, \dots, \ell_n$ of n distinct integers sorted in increasing order, an integer z .

Output : **true** if $z \in L$; **false** otherwise.

```

1  $a \leftarrow 1, b \leftarrow n$ 
2 while  $a \leq b$  do
3    $c \leftarrow \lfloor (a + b)/2 \rfloor$ 
4   if  $\ell_c = z$  then print true
5   if  $\ell_c > z$  then  $b \leftarrow c - 1$ 
6   else  $a \leftarrow c + 1$ 
7 print false
```

- (c) In [Algorithm 5](#) it was assumed that the list members are distinct. Modify the algorithm for the case where list member repetition is allowed

and where the list is sorted in nondecreasing order. What are the new worst-case time and space complexities of the modified algorithm?

Neumann János Lajos, better known as **John von Neumann**, was born in Budapest, Hungary in 1903. He obtained his PhD (mathematics) from the Pázmány Péter University in Budapest in 1925 and simultaneously earned a diploma in chemical engineering at ETH Zürich in Switzerland. During the period 1926–1930, he taught as a *Privatdozent* (junior lecturer) at the University of Berlin, the youngest in its history. His alleged powers of speedy and massive memorization and recall allowed him to recite entire directories with ease. In 1930, von Neumann was invited to Princeton University in New Jersey where he was offered a position on the faculty of the prestigious Institute of Advanced Study in 1933 and where he remained a professor of mathematics until his death. In 1937, he became an American citizen and in 1938 was awarded the Bôchner Memorial Prize for his work in analysis. During the late 1930s, he became an expert in modelling explosions. This led to his eventual involvement as one of the principal scientists in the Manhattan Project, culminating in the two nuclear weapons that were used by the United States of America against Japan to end the Second World War in 1945. He published more than 150 research papers, contributing to areas of mathematics and physics as diverse as set theory, geometry, measure theory, operator theory, lattice theory, quantum mechanics and -logic, game theory, linear programming, the theory of computing, mathematical statistics, fluid dynamics and weather forecasting. He died of cancer in Washington DC in 1957.



Biographic note 12: John von Neumann (1903–1957)

Project 3.2: Sorting

In many graph theoretic algorithms the procedure of sorting a list of integers in nondecreasing order is an important component or sub-procedure. In this project we explore this procedure and its algorithmic complexity.

In [Task 2 of Project 3.1](#) we saw that the time complexity of determining membership of an integer with respect to a given list of integers is lower when the list is sorted than when it is not sorted. It is therefore natural to enquire about the time and space complexities of the sorting procedure itself for a list of integers.

Tasks

1. Write down an algorithm with worst-case time complexity $T(n) = \mathcal{O}(n^2)$ for sorting a list of n integers in nondecreasing order by repeatedly applying the smallest member algorithm (designed in [Task 1 of Project 3.1](#)) to the remainder of the unsorted list. Apply your algorithm to the list $L_4 : 5, 8, 2, 4, 1, 4, 9, 3$ in order to convince yourself of the correct working of the

algorithm. Let us call this algorithm the **sequential sort algorithm**. Verify that your sequential sort algorithm has the desired time complexity — be sure to specify which operations you consider basic operations. What is the worst-case space complexity of the sequential sort algorithm?

2. **Algorithm 6** is a popular sorting procedure known as the **bubble sort algorithm**, because it repeatedly sorts adjacent pairs of list members in the correct order (by swapping them, if necessary), resulting in a fully ordered list that appears to “bubble up” from the original list.

Algorithm 6: Bubble sort (Iterative)

Input : A list $L : \ell_1, \dots, \ell_n$ of n integers.
Output : The list L sorted in nondecreasing order.

```

1 for  $j = 1$  to  $n - 1$  do
2   for  $i = 1$  to  $n - j$  do
3     if  $\ell_{n+1-i} < \ell_{n-i}$  then
4       temp  $\leftarrow \ell_{n-i}$ 
5        $\ell_{n-i} \leftarrow \ell_{n+1-i}$ 
6        $\ell_{n+1-i} \leftarrow \text{temp}$ 
7 print  $L$ 
```

Apply **Algorithm 6** to the unordered list L_4 in order to gain a feel for the working of the algorithm. What is the worst-case time complexity of **Algorithm 6** (if only variable assignment and **if** comparisons are taken as basic operations)? What is the worst-case space complexity of **Algorithm 6**?

3. The time complexities of the sequential sort and bubble sort algorithms are not best possible. Consider, for example, **Algorithm 7** (known as the **merge sort algorithm**) for sorting a list of n integers in decreasing order. This recursive algorithm was designed by the legendary computer scientist **John von Neumann** in 1945.

Algorithm 7: Merge sort (Recursive)

Input : A list $L : \ell_1, \dots, \ell_n$ of n integers.
Output : The list L sorted in nondecreasing order (possibly via further recursive calls).

```

1 if  $n = 1$  then
2   print  $L$ 
3 else
4    $m \leftarrow \lfloor n/2 \rfloor$ 
5    $L_\ell \leftarrow \ell_1, \ell_2, \dots, \ell_m$ 
6    $L_r \leftarrow \ell_{m+1}, \ell_{m+2}, \dots, \ell_n$ 
7   print Merge(MergeSort( $L_\ell$ ), MergeSort( $L_r$ ))
```

A call appears in [Step 7](#) of [Algorithm 7](#) to an external procedure called Merge. A pseudocode description of this procedure is provided in [Algorithm 8](#).

Algorithm 8: Merge (Iterative)

Input : Two lists $L^{(\ell)}, L^{(r)}$ each sorted in nondecreasing order.

Output : A combined list L sorted in nondecreasing order.

```

1  $L \leftarrow \emptyset$ 
2 while  $|L^{(\ell)}| > 0$  and  $|L^{(r)}| > 0$  do
3   if  $L_1^{(\ell)} \leq L_1^{(r)}$  then
4      $L \leftarrow L | L_1^{(\ell)}, L^{(\ell)} \leftarrow L^{(\ell)} \setminus L_1^{(\ell)}$ 
5   else
6      $L \leftarrow L | L_1^{(r)}, L^{(r)} \leftarrow L^{(r)} \setminus L_1^{(r)}$ 
7 if  $|L^{(\ell)}| > 0$  then  $L \leftarrow L | L^{(\ell)}$ 
8 if  $|L^{(r)}| > 0$  then  $L \leftarrow L | L^{(r)}$ 
9 return  $L$ 
```

Apply [Algorithm 7](#) to the unordered list L_4 in order to gain a feel for the working of the algorithm. What is the worst-case time complexity of [Algorithm 7](#) (if variable assignment and **if** comparisons are taken as basic operations)? What is the worst-case space complexity of [Algorithm 7](#)? The algorithm is remarkable in that its time complexity is provably best possible.

Project 3.3: Fibonacci numbers

Recall, from [Exercise 3.1](#) that the Fibonacci numbers are the sequence of positive integers defined by the recursion formula $f_i = f_{i-1} + f_{i-2}$, $i \in \{2, 3, 4, \dots\}$, together with the seeds $f_0 = 0$ and $f_1 = 1$.

Tasks

1. It is very tempting to use a recursive algorithm to determine the i -th Fibonacci number directly from the recursion definition of the Fibonacci sequence (see [Algorithm 9](#)).
 - (a) Prove by the strong form of induction that the i -th Fibonacci number is given by $f_i = (\Phi^i - \phi^i)/\sqrt{5}$ for all $i \in \mathbb{N}$, where

$$\Phi = \frac{1 + \sqrt{5}}{2} \quad \text{and} \quad \phi = \frac{1 - \sqrt{5}}{2} \quad (3.9)$$

are the two solutions to the quadratic equation $x^2 - x - 1 = 0$.

- (b) Use your result in (a) to prove that the recursive Fibonacci algorithm, [Algorithm 9](#), has worst-case time complexity $\mathcal{O}(a^{i+1})$ for some $a \in \mathbb{R}$, i.e. that the algorithm has an exponential worst-case time complexity, and hence that this approach is very inefficient in comparison with the approach of [Exercise 3.1](#).

Algorithm 9: Fibonacci numbers (Recursive)

Input : An integer $i \geq 0$.
Output : The i -th Fibonacci number f_i (possibly via further recursive calls).

```

1 if  $i = 0$  then
2   |   fib  $\leftarrow 0$ 
3 else if  $i = 1$  then
4   |   fib  $\leftarrow 1$ 
5 else
6   |   fib  $\leftarrow$  Recursive Fibonacci( $i - 1$ ) + Recursive Fibonacci( $i - 2$ )
7 print fib
```

2. The *greatest common divisor* of two integers m, n ($m \leq n$) is the largest integer d that divides both m and n without remainder. For example, the greatest common divisor of $m = 45$ and $n = 75$ is $d = 15$, since the only integers dividing $m = 45$ without remainder are 1, 3, 5, 9, 15 and 45, whilst the only integers dividing $n = 75$ without remainder are 1, 3, 5, 15, 25 and 75. The largest entry common to these lists of divisors is clearly $d = 15$. **Algorithm 10** is known as the *Euclidean algorithm* and has been named after its inventor, the ancient Greek mathematician **Euclid of Alexandria**. The algorithm may be used to compute the greatest common divisor of two integers $m \leq n$ efficiently.

Algorithm 10: Euclidean algorithm (Iterative)

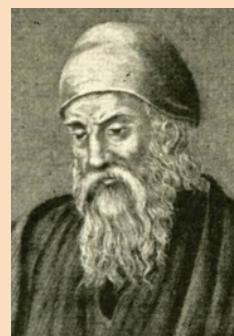
Input : Two integers $n \geq m > 0$.
Output : The greatest common divisor of m and n .

```

1  $i \leftarrow 0, n_0 \leftarrow n, m_0 \leftarrow m$ 
2 repeat
3   |    $s_i \leftarrow \lfloor n_i/m_i \rfloor, r_i \leftarrow n_i - s_i m_i$ 
4   |    $n_{i+1} \leftarrow m_i, m_{i+1} \leftarrow r_i$ 
5   |    $i \leftarrow i + 1$ 
6 until  $m_i = 0$ 
7 print  $n_i$ 
```

- (a) Show that the value $m_0 = m$ in **Algorithm 10** is bounded from below by the $(N + 1)$ -st Fibonacci number, where N is the number of repetitions of the loop spanning Steps 2–6 of **Algorithm 10**.
- (b) Use the result in (a) to show that $f_{i+1} > \Phi^{i-1}$ for all $i \in \{2, 3, 4, \dots\}$, where Φ is the famous *golden ratio* in (3.9).
- (c) Use the results in (a) and (b) to prove that the worst-case time complexity of **Algorithm 10** is $\mathcal{O}(\log m)$.

The Greek mathematician **Euclid of Alexandria**, whose name is an anglicised version of a Greek name which means *renowned* or *celebrated*, was active at the famous library in Alexandria during the reign of Ptolemy I (323–283 BC). He is often referred to as the *father of geometry* as a result of his venerated thirteen-part work *Elements*, which is one of the most influential works in all of the history of mathematics, serving as the main textbook for teaching mathematics for well over twenty centuries. In this work, Euclid deduced the principles of what is now called *Euclidean geometry* from a small set of axioms. Although lesser known, he also wrote works on perspective, conic sections, spherical geometry, number theory and optics. Very few original references to Euclid survive and hence little is known about his life. The dates, places and circumstances of his life are unknown. In fact, he is rarely mentioned by name by other Greek mathematicians from Archimedes onwards, instead only being referred to as *the author of Elements*. This has led to speculations about whether Euclid was, in fact, a person or perhaps a group of people.



Biographic note 13: Euclid of Alexandria (third to fourth century BC)

Project 3.4: Degree sequences

The problem of finding an unsorted version of the degree sequence of a graph may be solved by [Algorithm 1](#). This solution is, however, not wholly satisfactory, because a degree sequence was expressly defined as a nonincreasing sequence in [Chapter 1](#). Let us therefore consider the problem of finding the (true, sorted) degree sequence of a graph.

Tasks

1. Use the results of [Task 1](#) of [Project 3.1](#) to show that the problems of finding the minimum degree and the maximum degree of a graph are polynomial-time reducible to the problem of finding an unordered version of the degree sequence of a graph, as achieved by [Algorithm 1](#), and therefore that the problems of finding the minimum degree and the maximum degree of a graph are tractable.
2. Use the results of [Task 1](#) of [Project 3.2](#) to show that the problem of finding the ordered degree sequence of a graph is polynomial-time reducible to the problem of finding an unordered version of the degree sequence of a graph, and therefore that the problem of finding the true degree sequence of a graph is tractable.
3. The pseudocode in [Algorithm 1](#) may be used to compute an unordered version of the degree sequence of a graph. Adapt the algorithm so as to compute the *ordered* degree sequence of a graph, as described in [Section 1.5](#). What are the worst-case space and time complexities of the adapted algorithm?

Project 3.5: NP-completeness of the vertex cover problem

A **vertex cover** of a graph G is a subset $\mathcal{C} \subseteq V(G)$ with the property that each edge in $E(G)$ is incident to at least one vertex in \mathcal{C} . Define the decision problem $D_{\text{cover}}(G, k)$ as the problem of deciding whether or not a given graph G has a vertex cover of cardinality at most k , for some given $k \in \mathbb{N}$. Let $C_{\text{cover}}(G)$ be the associated computation problem of computing the cardinality of a smallest vertex cover for a given graph G .

Tasks

1. Using [Theorem 3.6](#), prove that $D_{\text{cover}}(G, k)$ is NP-complete, by polynomial-time reducing the decision problem $D_{\text{clique}}(G, k)$ to it (that is, by proving that $D_{\text{clique}}(G, k) \rightsquigarrow D_{\text{cover}}(G, k)$; recall that $D_{\text{clique}}(G, k)$ was shown to be NP-complete in [Section 3.4](#)).
2. Deduce that $C_{\text{cover}}(G)$ is also NP-complete.

Further reading

- [1] TA Budd, 1994. *Classical Data Structures in C++*, Addison-Wesley, Reading (MA), Chapters 6, 11 and 12.
- [2] SA Cook, 1971. *The complexity of theorem proving procedures*, Proceedings of the Third Annual ACM Symposium on the Theory of Computing, pp. 151–158.
- [3] MR Garey and DS Johnson, 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, First edition, WH Freeman & Co., New York (NY).
- [4] RP Grimaldi, 1994. *Discrete and Combinatorial Mathematics: An Applied Introduction*, Third edition, Addison-Wesley, New York (NY), pp. 657–665, 671–693.
- [5] LA Levin, 1973. *Universal sequential search problems*, Problemy Peredachi Informatsii, **9(3)**, pp. 115–116.
- [6] U Manber, 1989. *Introduction to Algorithms: A Creative Approach*, Addison-Wesley, Reading (MA), Chapters 3 and 11.
- [7] BA Trakhtenbrot, 1984. *A survey of Russian approaches to perebor (brute-force searches) algorithms*, Annals of the History of Computing, **6(4)**, pp. 384–400.



Optimal paths

Contents

4.1	Introduction	87
4.2	Distance in weighted graphs	90
4.3	Shortest paths in weighted graphs	91
4.4	Longest paths in weighted digraphs	99
	Exercises	103
	Computer exercises	105
	Projects	107
	Further reading	113

4.1 Introduction

When an ambulance is called to the scene of an accident, it is critical that the ambulance travels along a shortest route. Consider, for example, the scenario where an ambulance is called out to an accident scene at the corner of IVOR and DANIEL streets of the *Tshwane* suburb of *Mountain View*, a map of which may be seen in [Figure 4.1](#).

Suppose further that the ambulance is to depart from the hospital at the corner of DAPHNE and FELIX streets. The street network of *Mountain View* may be modelled by the graph $G_{4.1}$ in [Figure 4.2](#), as we saw in [Project 1.3](#), where the accident scene (denoted by “ a ”) and ambulance departure point (denoted by “ h ”), as well as all street intersections and dead-ends, are represented by vertices, while the edges represent street sections and are labelled. These labels are known as **weights** and the resulting graph is called a **weighted graph**. In this case the weights may be interpreted as the time it takes to travel down the corresponding streets, measured in seconds. These travel times typically vary according to traffic density and may be estimated by satellite.

A shortest path from the vertex h to the vertex a in $G_{4.1}$ represents an optimal route for the ambulance, enabling it to reach the accident scene in the shortest possible time. There are, of course, many other graph applications in which shortest routes are required, such as finding a set of interconnecting flights between some pair of origin and destination airports corresponding to the shortest possible travel time or least possible travel cost. This situation may be modelled by a shortest path in a graph whose vertices represent airports serviced by a group of airline alliance

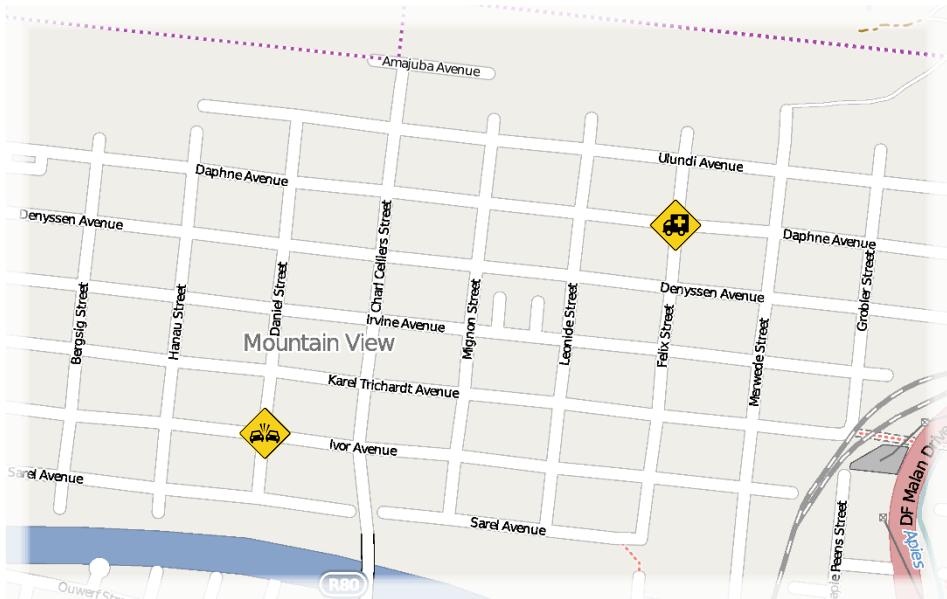


Figure 4.1: A street map of a section of the *Tshwane* suburb of *Mountain View* adjacent to the Magaliesberg Nature Reserve in South Africa, showing an accident scene (at the corner of IVOR and DANIEL streets) and a hospital (at the corner of DAPHNE and FELIX streets).

partners, whose edges represent direct flights between airports and whose edge weights represent either travel time or travel cost. Applications in which shortest paths in graphs are relevant modelling tools, are both abundant and intuitive.

It is perhaps less intuitive, however, why one would seek longest paths. Longest paths in digraphs may be used as an aid to schedule large projects consisting of a number of smaller activities. If the duration of each activity is known, the length of time required to complete the project may be determined as well as how long each activity may be delayed without delaying the overall completion date of the project.

Activities forming part of a project and which have to be completed before a certain activity may be started, are known as **predecessors** of the activity in question. The precedence relations of activities that make up the project may be summarised by a weighted digraph in which the vertices represent the activities, and in which an arc of the form (a, b) is present if activity a must be completed before activity b may be commenced (as shown in Figure 4.3(a)). In such a digraph, often called a **scheduling digraph** or an **activity digraph**, arc weights denote the time durations required to complete each activity (the value w_a in Figure 4.3(a) is an arc weight representing the completion time of activity a).

Figure 4.3(b) represents the situation where activities a and b both have to be completed, requiring completion times of w_a and w_b temporal units, respectively, before activity c may be commenced, while in Figure 4.3(c) activity a must be completed before any of activities b or c may be commenced.

A longest (directed) path in an activity digraph is called a **critical path**. If an activity is contained in any critical path of an activity graph, a delay in the

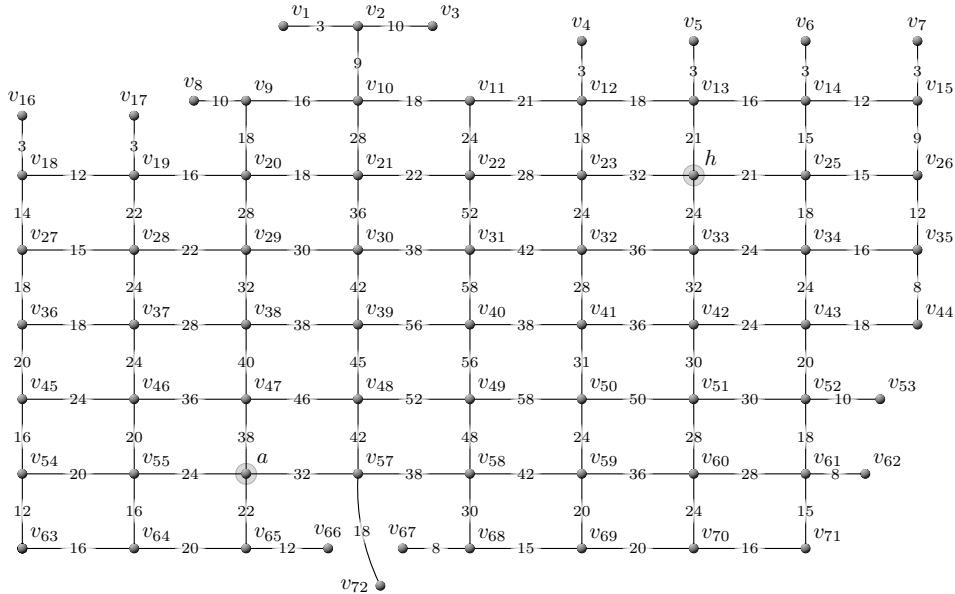


Figure 4.2: A graph model, $G_{4.1}$, of (a slightly modified version of) the street map in [Figure 4.1](#), showing an accident scene (denoted by “ a ”) and the point of departure (denoted by “ h ”) of an ambulance called out to the accident scene.

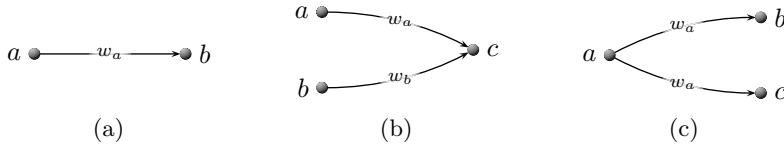


Figure 4.3: Vertices and arcs in an activity digraph.

completion of that activity will result in a delay of the overall completion time of the project, whereas if a delay occurs in the completion of an activity that is not contained in any critical path, it may have no effect on the overall completion time of the project, depending on the magnitude of the delay.

Let us illustrate the use of critical paths in activity digraphs by an example. Before a music concert may be held in Cape Town, South Africa, the activities shown in [Table 4.1](#) have to be completed. A digraph $D_{4.2}$, summarising the precedence relations and durations of the activities involved in scheduling the music concert, is shown in [Figure 4.4](#). Because activity a is the only activity without predecessors, it represents the first activity in the project, known as the **source**, and critical paths are sought from this source to activity g , which is clearly the final activity of the project, known as the **sink**. (If there were more than one activity without predecessors, one could have included a dummy vertex s — representing the virtual source or start of the project — in the activity graph, joining s to all vertices representing activities without predecessors by means of arcs with zero weights. A similar remark holds for the sink.)

There are only four directed paths from activity a (the source) to activity g (the sink) in the activity digraph $D_{4.2}$. These are the path $P^{(1)} : abdfg$ of length 11,

Activity	Description	Duration [days]	Predecessors
<i>a</i>	Find site	3	—
<i>b</i>	Hire engineers	1	<i>a</i>
<i>c</i>	Hire opening speaker	4	<i>a</i>
<i>d</i>	Prepare electronics	5	<i>b</i>
<i>e</i>	Advertising	7	<i>a, c</i>
<i>f</i>	Perform rehearsals	2	<i>c, d</i>
<i>g</i>	Concert performance	—	<i>e, f</i>

Table 4.1: Activities involved in scheduling a concert.

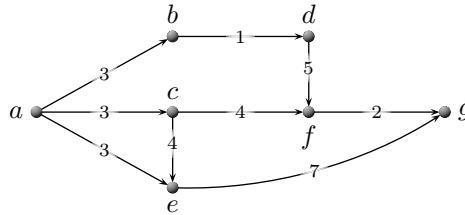


Figure 4.4: Concert activity digraph, $D_{4.2}$.

the path $P^{(2)} : a c f g$ of length 9, the path $P^{(3)} : a c e g$ of length 14 and the path $P^{(4)} : a e g$ of length 10. Of these paths $P^{(3)}$ is the (unique) critical or longest path in $D_{4.2}$ and hence any delay in finding a concert site, hiring an opening speaker or advertising the concert will delay the performance of the concert. Delays in the activities of hiring engineers, preparing the electronics or performing rehearsals may or may not affect the performance date, depending on the magnitudes of the delays. Since activity *b* (hiring engineers) is only on the path $P^{(1)}$, the maximum allowable delay for this activity before the actual concert performance is delayed, is the difference between the lengths of the path $P^{(1)}$ and that of the critical path $P^{(3)}$, i.e. three days. By the same argument, the maximal allowable delay for activity *d* (preparing electronics) is three days, and that for activity *f* (performing rehearsals) is also three days.

Other applications requiring longest paths in digraphs include optimal investment portfolio selection, as we shall show in one of the projects at the end of the chapter. In the remainder of this chapter we shall consider the computational problems of actually finding shortest paths in graphs, or longest paths in digraphs. Algorithms will be considered for solving these problems, and the computational complexities of these algorithms will be analysed. Let us begin, however, by formalising the notion of distance in a weighted graph.

4.2 Distance in weighted graphs

A **weighted (di)graph** is a (di)graph G in which each edge (arc) e is assigned a positive real number, called the **weight** of e and denoted by $w_G(e)$ or by $w(e)$ if the (di)graph G is clear from the context. The **length of a path** P in a weighted (di)graph G is the sum of the weights of the edges (arcs) of P . If a vertex u is connected to a vertex v in a weighted (di)graph G , the **distance** $d_G(u, v)$ or merely $d(u, v)$ from u to v is the minimum of the lengths of the (directed) u - v

paths of G . A $u-v$ path of minimum weight in G is called a **shortest** $u-v$ path in G (so that, for example, a path containing three edges (arcs), each of weight one, is “shorter” than a path with two edges (arcs), each of weight two).

Consider the weighted graph $G_{4.3}$ in Figure 4.5(a). The path $P : v_2 v_1 v_3 v_4$ is a shortest v_2 - v_4 path (of minimum weight 4). Notice that the path $Q : v_2 v_3 v_4$ (of weight 5) is not a shortest v_2 - v_4 path in $G_{4.3}$, even though Q contains fewer edges than P .

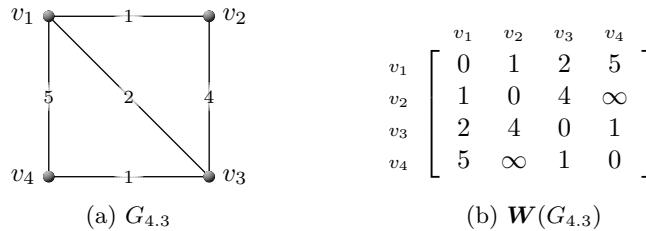


Figure 4.5: A weighted graph $G_{4.3}$ and its weight matrix $\mathbf{W}(G_{4.3})$.

Let G be a weighted (di)graph of order n with vertex set $V(G) = \{v_1, \dots, v_n\}$. It is convenient to represent G on a computer by an $n \times n$ **weight matrix** $\mathbf{W}(G) = [w_{ij}]$, where

$$w_{ij} = \begin{cases} w(v_i v_j) & \text{if } v_i v_j \in E(G) \\ \infty & \text{if } v_i v_j \notin E(G). \end{cases}$$

(By “ ∞ ” we mean a number considerably larger than any weight actually occurring in our calculations.) An example of a graph and its weight matrix are shown in Figure 4.5, where the weights of the edges are as indicated in the diagram.

Every graph may, in fact, be seen as a weighted graph in the sense that in cases where weights are omitted, each edge may be assumed to have a weight of 1. In such cases the notion of a weight matrix reduces to that of an adjacency matrix.

❖ The reader should now be able to attempt Exercises 4.1–4.3.

4.3 Shortest paths in weighted graphs

Let us explore two well-known approaches towards finding shortest paths in weighted graphs. The first of these approaches is **Dijkstra’s algorithm** (Section 4.3.1), which is a method whereby shortest paths may be found from any *specific* vertex to all other vertices of a weighted graph. Secondly, **Floyd’s algorithm** (Section 4.3.2) generalises this approach to establish the lengths of shortest paths from *all* vertices to all other vertices of a weighted graph.

4.3.1 Dijkstra’s algorithm

In 1959, **Dijkstra** [4] proposed a method for finding shortest paths between a specific vertex x and all other vertices of a weighted graph G in which all weights are positive. The idea behind **Dijkstra’s method** is to modify a label $\ell(v)$ attached to each vertex v of G in an iterative fashion, where $\ell(v)$ represents the length of a shortest x - v path presently known. As the algorithm proceeds, these labels are

The Dutch computer scientist **Edsger Wybe Dijkstra** was born in Rotterdam and originally studied theoretical physics at Leiden University in the Netherlands, but soon realised that he had a passion for computer science. He started his career at the Mathematisch Centrum in Amsterdam, where he was later professor at the Eindhoven University of Technology. Thereafter, he was appointed as research fellow at the Burroughs Corporation and in 1984 moved to the University of Texas at Austin in the United States of America, where he was professor of computer science and where he retired in 2000. He was the designer of many algorithms, including the shortest path algorithm which today carries his name. Dijkstra was one of the early pioneers of distributed computing and he later also made important contributions to the field of formal verification. He was the father of the multiprogramming system *THE*, one of the first compilers supporting recursion, and the notion of self-stabilisation in the field of distributed computing. Dijkstra was strongly against the use of the **go to** statement in computing, arguing that departures from linear control flow were clearer if allowed only in disciplined higher-level structures, such as the **if then else** environment and the **while** loop. He was the recipient of the prestigious Turing award in 1972 and after his death the PODC Influential Paper Award in distributed computing of the Association of Computing Machinery was renamed the Dijkstra Prize in his honour.



Biographic note 14: Edsger Dijkstra (1930–2002)

reduced as shorter paths are found from x to v . Initially, x is labelled $\ell(x) = 0$ and all other vertices are labelled ∞ . At any stage during execution of the algorithm, let the variable $\text{parent}(v)$ denote the vertex that precedes v on a shortest x - v detected thus far, for each vertex $v \neq x$. Furthermore, at each stage of the algorithm, we consider those vertices with temporary labels that are adjacent to the current vertex. To each such vertex v , we assign a new temporary label $\ell(v)$ representing the length of a shortest x - v path found up to that point. Among all vertices with temporary labels, we select one whose label is a minimum (such a vertex may not be unique). The selected vertex with minimum label then becomes the current vertex and its label $\ell(v)$ is now permanent. Whenever a shorter x - v path is found, the label $\ell(v)$ is updated and at this point $\text{parent}(v)$ is also updated to indicate the vertex that now precedes v on this shorter x - v path. Eventually each vertex acquires a permanent label which represents the distance from x to that vertex. For v ($\neq x$), the label $\ell(v)$ changes (perhaps several times) from ∞ to $d(x, v)$ as this distance is determined. When v acquires its permanent label $\ell(v)$ we produce a shortest x - v path $P : w_0 w_1 w_2 \dots w_t$ where $w_0 = x$, $w_t = v$ and $w_{i-1} = \text{parent}(w_i)$ for all $i \in [t]$. [Dijkstra's algorithm](#) is given in pseudocode as [Algorithm 11](#).

Let us illustrate the working of [Algorithm 11](#) by an example. Suppose we seek shortest paths from v_1 to every other vertex of the weighted graph $G_{4,4}$, shown in

Algorithm 11: Dijkstra's algorithm for computing shortest paths in a weighted graph

Input : A weighted graph G of order n with positive weights; a vertex $x \in V(G)$.

Output : Shortest paths from x to every other vertex v in G .

```

1  $S \leftarrow V(G)$ 
2 foreach  $v \in V(G) \setminus \{x\}$  do  $\ell(v) \leftarrow \infty$ 
3  $\ell(x) \leftarrow 0$ , parent( $x$ )  $\leftarrow x$ 
4 while  $|S| \neq 1$  do
5    $u \leftarrow v$  where  $\ell(v) = \min\{\ell(w) \mid w \in S\}$ 
6   foreach  $v \in S$  do
7     if  $uv \in E(G)$  and  $\ell(v) > \ell(u) + w(uv)$  then
8        $\ell(v) \leftarrow \ell(u) + w(uv)$ 
9       parent( $v$ )  $\leftarrow u$ 
10   $S \leftarrow S \setminus \{u\}$ 
```

Figure 4.6. We start with the assignments $x \leftarrow v_1$, $\ell(x) \leftarrow 0$, $\ell(v_i) \leftarrow \infty$ for all $i = 2, \dots, 7$ and $S \leftarrow V(G_{4.4}) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ in Steps 2–3.

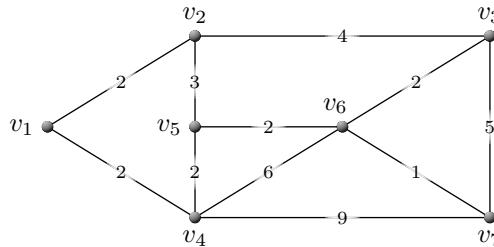


Figure 4.6: Shortest paths from v_1 in $G_{4.4}$.

Since $|S| = 7 \neq 1$ in Step 4 we continue to Step 5, where we select the vertex v_1 , since it has the smallest label among the vertices in S . The label $\ell(v_1) = 0$ is now the permanent label of v_1 . Next we examine the vertices adjacent to v_1 in Steps 6–9, namely v_2 and v_4 . Since $\ell(v_2) = \infty > 2 = \ell(v_1) + w(v_1v_2)$, we make the assignments $\ell(v_2) \leftarrow 2$ and $\text{parent}(v_2) \leftarrow v_1$. Similarly, we make the assignments $\ell(v_4) \leftarrow 2$ and $\text{parent}(v_4) \leftarrow v_1$. At this point we remove v_1 from S by the assignment $S \leftarrow S \setminus \{v_1\}$ and return to Step 4 of the algorithm.

Since $|S| = 6 \neq 1$ in Step 4 we continue to Step 5, where we select the vertex v_2 arbitrarily, since it is a vertex of minimum label in S ; the label $\ell(v_2) = 2$ is now the permanent label of v_2 . Next we examine the vertices in S adjacent to v_2 in Steps 6–9, namely v_3 and v_5 . Since $\ell(v_3) = \infty > 6 = \ell(v_2) + w(v_2v_3)$, we make the assignments $\ell(v_3) \leftarrow 6$ and $\text{parent}(v_3) \leftarrow v_2$. Similarly, we make the assignments $\ell(v_5) \leftarrow 5$ and $\text{parent}(v_5) \leftarrow v_2$. We then remove v_2 from S by the assignment $S \leftarrow S \setminus \{v_2\}$ and return to Step 4 of the algorithm.

Since $|S| = 5 \neq 1$ in Step 4 we continue to Step 5, where we select the vertex v_4 , since it is a vertex of minimum label in S ; the label $\ell(v_4) = 2$ is now the permanent label of v_4 . Next we examine the vertices in S adjacent to v_4 in

Steps 6–9, namely v_5 , v_6 and v_7 . Since $\ell(v_5) = 5 > 4 = \ell(v_4) + w(v_4v_5)$, we make the assignments $\ell(v_5) \leftarrow 4$ and $\text{parent}(v_5) \leftarrow v_4$. Also, since $\ell(v_6) = \infty > 8 = \ell(v_4) + w(v_4v_6)$, we make the assignments $\ell(v_6) \leftarrow 8$ and $\text{parent}(v_6) \leftarrow v_4$. Similarly, we make the assignments $\ell(v_7) \leftarrow 11$ and $\text{parent}(v_7) \leftarrow v_4$. At this point we remove v_4 from S by the assignment $S \leftarrow S \setminus \{v_4\}$ and return to Step 4 of the algorithm.

Since $|S| = 4 \neq 1$ in Step 4 we continue to Step 5, where we select the vertex v_5 , since it is a vertex of minimum label in S ; the label $\ell(v_5) = 4$ is now the permanent label of v_5 . Next we examine the vertices in S adjacent to v_5 in Steps 6–9, namely v_6 . Since $\ell(v_6) = 8 > 6 = \ell(v_5) + w(v_5v_6)$, we make the assignments $\ell(v_6) \leftarrow 6$ and $\text{parent}(v_6) \leftarrow v_5$. We then remove v_5 from S by the assignment $S \leftarrow S \setminus \{v_5\}$ and return to Step 4 of the algorithm.

Since $|S| = 3 \neq 1$ in Step 4 we continue to Step 5, where we select the vertex v_3 arbitrarily, since it is a vertex of minimum label in S ; the label $\ell(v_3) = 6$ is now the permanent label of v_3 . Next we examine the vertices in S adjacent to v_3 in Steps 6–9, namely v_6 and v_7 . Since $\ell(v_6) = 6 \not> 8 = \ell(v_3) + w(v_3v_6)$ and $\ell(v_7) = 11 \not> 11 = \ell(v_3) + w(v_3v_7)$, no label-altering assignments are made. At this point we remove v_3 from S by the assignment $S \leftarrow S \setminus \{v_3\}$ and return to Step 4 of the algorithm.

Since $|S| = 2 \neq 1$ in Step 4 we continue to Step 5, where we select the vertex v_6 , since it is a vertex of minimum label in S ; the label $\ell(v_6) = 6$ is now the permanent label of v_6 . Next we examine the vertices in S adjacent to v_6 in Steps 6–9, namely v_7 . Since $\ell(v_7) = 11 > 7 = \ell(v_6) + w(v_6v_7)$, we make the assignments $\ell(v_7) \leftarrow 7$ and $\text{parent}(v_7) \leftarrow v_6$. We then remove v_6 from S by the assignment $S \leftarrow S \setminus \{v_6\}$ and return to Step 4 of the algorithm.

Since $|S| = 1$ in Step 4 we stop, and the label $\ell(v_7) = 7$ now becomes the permanent label of the remaining vertex v_7 in S . The results thus obtained are summarised in Table 4.2. A shortest path from v_1 to v_2 (v_3, v_4, v_5, v_6, v_7 , respectively) is $P^{(2)} : v_1 v_2$, ($P^{(3)} : v_1 v_2 v_3$, $P^{(4)} : v_1 v_4$, $P^{(5)} : v_1 v_4 v_5$, $P^{(6)} : v_1 v_4 v_5 v_6$, $P^{(7)} : v_1 v_4 v_5 v_6 v_7$, respectively) of length 2 (6, 2, 4, 6, 7, respectively).

v_1	v_2	v_3	v_4	v_5	v_6	v_7	S
(0, v_1)	(∞ , –)	{ $v_1, v_2, v_3, v_4, v_5, v_6, v_7$ }					
—	(2, v_1)	(∞ , –)	(2, v_1)	(∞ , –)	(∞ , –)	(∞ , –)	{ $v_2, v_3, v_4, v_5, v_6, v_7$ }
—	—	(6, v_2)	(2, v_1)	(5, v_2)	(∞ , –)	(∞ , –)	{ v_3, v_4, v_5, v_6, v_7 }
—	—	(6, v_2)	—	(4, v_4)	(8, v_4)	(11, v_4)	{ v_3, v_5, v_6, v_7 }
—	—	(6, v_2)	—	—	(6, v_5)	(11, v_4)	{ v_3, v_6, v_7 }
—	—	—	—	—	(6, v_5)	(11, v_4)	{ v_6, v_7 }
—	—	—	—	—	—	(7, v_6)	{ v_7 }

Table 4.2: Consecutive steps when Algorithm 11 is applied to $G_{4,4}$, the graph in Figure 4.6. The first entry in each ordered pair is the label ℓ of the vertex at the column head, whilst the second entry in each ordered pair is the parent of the vertex at the column head in a hitherto shortest path from v_1 to that vertex.

In order to verify that, at termination, Dijkstra's algorithm has labelled the vertices with their correct distances from x , we require the following two lemmas.

Lemma 4.1 At termination of Dijkstra's algorithm, the label $\ell(v)$ is finite for all $v \in V(G)$, if G is a connected, weighted graph.

Proof Suppose, to the contrary, that at termination of Dijkstra's algorithm, when applied to a connected graph, not all vertices have finite labels. Let y be the first vertex selected from S with an infinite label upon termination of the algorithm. Since y was chosen as a vertex of S with minimum label, all vertices of S had infinite labels when y was selected, while all vertices already deleted from S had finite labels. But then there is no edge from $V(G) \setminus S$ to S (since if such an edge $e = uv$ with $u \in V(G) \setminus S$ and $v \in S$ did exist, then when u was selected as the current vertex in Step 5 of the algorithm, the label $\ell(v)$ of v would have dropped from ∞ to $\ell(u) + w(uv)$, which is finite). Thus there is no path from a vertex of S to a vertex of $V(G) \setminus S$. Hence, G is disconnected, which is a contradiction. ■

Lemma 4.2 At termination of Dijkstra's algorithm, $d(x, v) \leq \ell(v)$ for all $v \in V(G)$, if G is a connected, weighted graph.

Proof If $v = x$, then $d(x, v) = 0 = \ell(v)$. If $v \neq x$, then by Lemma 4.1, $\ell(v)$ is finite. Consider the x - v path $P : u_0 u_1 u_2 \cdots u_k$, where $u_0 = x$, $u_k = v$ and $u_{i-1} = \text{parent}(u_i)$ for all $i \in [k]$. The path P has weight $w(P) = \sum_{i=1}^k w(u_{i-1} u_i)$. Since u_{k-1} is the vertex used to label u_k , it follows that $\ell(v) = \ell(u_k) = \ell(u_{k-1}) + w(u_{k-1} u_k)$. After this labelling, the vertex u_{k-1} is removed from S (in Step 10), and hence its label can never change again. Repeating this backtracking process, we have $\ell(u_{k-1}) = \ell(u_{k-2}) + w(u_{k-2} u_{k-1})$, so that $\ell(v) = \ell(u_{k-2}) + w(u_{k-2} u_{k-1}) + w(u_{k-1} u_k)$. Eventually we will backtrack to x . Thus, $\ell(v) = \ell(u_0) + w(u_0 u_1) + w(u_1 u_2) + \cdots + w(u_{k-1} u_k) = w(P)$, since $\ell(x) = \ell(u_0) = 0$. The path P therefore has weight $\ell(v)$. Since $d(x, v)$ is the length of a shortest x - v path, it follows that $d(x, v) \leq \ell(v)$. ■

We are now in a position to verify Dijkstra's algorithm.

Theorem 4.3 Let G be a connected, weighted graph of order n in which all edge weights are positive.

- (i) At termination of Dijkstra's algorithm, $\ell(v) = d(x, v)$ for all $v \in V(G)$.
- (ii) Dijkstra's algorithm has a worst-case time complexity of $\mathcal{O}(n^2)$.

Proof (i) We proceed by induction on the order in which we delete vertices from S . The required equality is certainly true for the first vertex, namely x , deleted from S , since $\ell(x) = 0 = d(x, x)$. Assume that $\ell(u) = d(x, u)$ for all vertices u deleted from S before v . Let $P : v_0 v_1 v_2 \cdots v_k$ with $v_0 = x$ and $v_k = v$ be a shortest x - v path of length $d(x, v)$. Then the x - v_i subsection of P must be a shortest x - v_i path for all $i \in [k]$ (for otherwise we could find a shorter x - v path than P). Suppose v_i is the vertex of highest subscript on P deleted from S before v . By the induction hypothesis, $\ell(v_i) = d(x, v_i) = w(v_0 v_1) + w(v_1 v_2) + \cdots + w(v_{i-1} v_i)$. When v_i was chosen from S as the current vertex, we compared the current label of v_{i+1} with $\ell(v_i) + w(v_i v_{i+1})$ in Step 7 of the algorithm. Hence, after v_i was deleted from S in Step 10, $\ell(v_{i+1}) \leq \ell(v_i) + w(v_i v_{i+1})$. Since Step 7 can only decrease labels, $\ell(v_{i+1})$ still satisfied this inequality when v was chosen as current

vertex. Suppose $v_{i+1} \neq v$. Then,

$$\begin{aligned}\ell(v_{i+1}) &\leq \ell(v_i) + w(v_i v_{i+1}) \\&= d(x, v_i) + w(v_i v_{i+1}) \\&= d(x, v_{i+1}) \\&< d(x, v) \quad (\text{since } v_{i+1} \text{ precedes } v \text{ on } P) \\&\leq \ell(v). \quad (\text{by Lemma 4.2})\end{aligned}$$

This, however, contradicts the fact the v was chosen before v_{i+1} . Hence, $v = v_{i+1}$, and so

$$\begin{aligned}\ell(v) &\leq \ell(v_i) + w(v_i v) \\&= d(x, v_i) + w(v_i v_{i+1}) \\&= d(x, v) \\&\leq \ell(v). \quad (\text{by Lemma 4.2})\end{aligned}$$

We conclude that $\ell(v) = d(x, v)$.

(ii) Let us take the following to be basic operations for the purposes of an analysis of the time complexity of [Algorithm 11](#): all variable assignments (including inserting or removing set members), if comparisons and addition of two real numbers. Then the time complexity of Steps 1 and 2 of [Algorithm 11](#) is $\mathcal{O}(n)$, since n assignments are made in [Step 1](#), and n label assignments and n set entries are made in [Step 2](#), and these steps are executed only once. [Step 3](#) has constant time complexity because of the two assignments, i.e. a time complexity of $\mathcal{O}(1)$. The minimum label in [Step 5](#) of the algorithm may be found in $|S| - 1 < n$ if comparisons and hence has a time complexity of $\mathcal{O}(n)$. Since v has at most $n - 1$ neighbours in [Steps 6–9](#), the time complexity of these steps is no more than $5(n - 1) = \mathcal{O}(n)$ basic operations (two if comparisons, one addition and two variable assignments). Therefore, the time complexity of one traversal of [Steps 6–9](#) of the algorithm is $\mathcal{O}(n)$. [Step 10](#) may be performed in constant time complexity, that is in $\mathcal{O}(1)$ time, since there is only one variable assignment. [Steps 6–9](#) of the algorithm are repeated $n - 1$ times, and hence the portion of the algorithm represented by [Steps 6–9](#) has an overall worst-case time complexity of $\mathcal{O}(n^2)$, dominating those of [Steps 1, 2](#) and [10](#). We conclude that the worst-case time complexity of the entire algorithm is therefore $\mathcal{O}(n^2)$. ■

A shortest path may be found from a specified vertex to every other vertex in a weighted graph G by [Algorithm 11](#). To find shortest paths between all pairs of vertices of G , one may simply repeat [Algorithm 11](#), each time with a different starting vertex in G . The worst-case time complexity of such a procedure would be $\mathcal{O}(n^3)$ if G has order n . However, we next turn our attention to an elegant, alternative procedure of the same time complexity for determining the lengths of shortest paths between all pairs of vertices in a weighted graph.

❖ The reader should now be able to attempt [Exercise 4.4](#) and [Project 4.1](#).

4.3.2 Floyd's algorithm

Floyd's algorithm [5], dating from 1962, is a procedure for finding the distances between all pairs of vertices in a weighted graph that may or may not be connected and may contain edges with negative weights. No negative cycles are, however, allowed, that is, cycles whose lengths are negative. The problem with a negative cycle is that with repeated traversal of such a cycle, there is no lower bound on the length of a shortest walk coinciding with the negative cycle.

The eminent computer scientist **Robert W. Floyd** was born in New York and completed his secondary school training at age 14. In 1953, he obtained a bachelor's degree in liberal arts and in 1959 a second bachelor's degree in physics, both from the University of Chicago. His college roommate was the legendary Carl Sagan. He started his career during the 1950s working for the then Armour Research Foundation at the Illinois Institute of Technology and became a computer operator during the 1960s. Without having a PhD, he was appointed associate professor of computer science at Carnegie Mellon University at age 27 and became full professor of computer science in the Department of Computer Science at Stanford University six years later, where he remained until his retirement during the early 1990s. During the 1960s he began publishing a series of seminal papers on compilers, operator-precedence grammars and programming language semantics. At Stanford, he taught algorithmic courses on sorting and searching, and wrote a textbook entitled *The language of machines: An introduction to computability and formal languages* together with one of his former graduate students, Richard Beigel. He was also the major reviewer for Donald Knuth's influential book *The art of computer programming* and won the prestigious Turing Award in 1978 "for having a clear influence on methodologies for the creation of efficient and reliable software, and for helping to found the following important subfields of computer science: the theory of parsing, the semantics of programming languages, automatic program verification, automatic program synthesis and the analysis of algorithms." In 1991, the Institute of Electrical and Electronics Engineers (IEEE) Computer Society awarded Floyd its Computer Pioneer Award for his work on early compilers.



Biographic note 15: Robert Floyd (1936–2001)

Consider a weighted graph G of order n with weight matrix $\mathbf{D}^{(0)} = [d_{ij}^{(0)}]$. Suppose that somehow a matrix $\mathbf{D}^{(k-1)} = [d_{ij}^{(k-1)}]$ is known, whose (i, j) -th entry $d_{ij}^{(k-1)}$ denotes the length of a shortest v_i - v_j path utilising *only* the first $k - 1$ vertices v_1, \dots, v_{k-1} of G as *internal* vertices of the path. Clearly $d_{ij}^{(k)} \leq d_{ij}^{(k-1)}$ for all $i, j, k \in [n]$, because we have more freedom (due to the additional vertex at our disposal) when considering v_i - v_j paths utilising only the first k vertices v_1, \dots, v_k of G as *internal* vertices of the path, than when considering v_i - v_j paths utilising only the first $k - 1$ vertices v_1, \dots, v_{k-1} of G . If no v_i - v_j path utilising only the first k vertices v_1, \dots, v_k of G is shorter than any v_i - v_j path utilising only

the first $k - 1$ vertices v_1, \dots, v_{k-1} of G , then certainly $d_{ij}^{(k)} = d_{ij}^{(k-1)}$. If, however, there is a v_i - v_j path P utilising only the first k vertices v_1, \dots, v_k of G that is shorter than any v_i - v_j path utilising only the first $k - 1$ vertices v_1, \dots, v_{k-1} of G , then the vertex v_k appears in the path P , in which case $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ by the hereditary property of shortest paths. Combining the two cases above, it is possible to construct a matrix $\mathbf{D}^{(k)}$, whose (i, j) -th element,

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}, \quad (4.1)$$

denotes the length of a shortest v_i - v_j path utilising only the first k vertices v_1, \dots, v_k of G , by using information contained in the matrix $\mathbf{D}^{(k-1)}$ only. Taking the matrix $\mathbf{D}^{(0)}$ as starting point, a sequence of matrices $\mathbf{D}^{(0)}, \dots, \mathbf{D}^{(n)}$ may thus be formed by means of (4.1). The last matrix $\mathbf{D}^{(n)}$ in this sequence will contain the distances between every pair of vertices v_i, v_j of G due to the fact that the entries of the matrix represent lengths of shortest v_i - v_j paths utilising *any* vertices of G .

The ideas above lead us to the procedure described in pseudocode in [Algorithm 12](#). Let us illustrate the working of [Algorithm 12](#) by an example. Again consider the directed graph $G_{4,4}$ shown in [Figure 4.6](#).

Algorithm 12: Floyd's algorithm for computing shortest distances in a weighted graph

Input : The weight matrix $\mathbf{D}^{(0)} = [d_{ij}^{(0)}]$ of a weighted graph G of order n which may contain no cycles of negative weight.

Output : A matrix $\mathbf{D}^{(n)} = [d_{ij}^{(n)}]$ of shortest distances between all pairs of vertices of G .

```

1 for  $k = 1$  to  $n$  do
2   for  $i = 1$  to  $n$  do
3     for  $j = 1$  to  $n$  do
4        $d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$ 
```

The sequence of matrices $\mathbf{D}^{(0)}, \dots, \mathbf{D}^{(7)}$ computed by [Floyd's algorithm](#) for this example is given by

$$\mathbf{D}^{(0)} = \begin{bmatrix} 0 & 2 & \infty & 2 & \infty & \infty & \infty \\ 2 & 0 & 4 & \infty & 3 & \infty & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 & 5 \\ 2 & \infty & \infty & 0 & 2 & 6 & 9 \\ \infty & 3 & \infty & 2 & 0 & 2 & \infty \\ \infty & \infty & 2 & 6 & 2 & 0 & 1 \\ \infty & \infty & 5 & 9 & \infty & 1 & 0 \end{bmatrix}, \quad \mathbf{D}^{(1)} = \begin{bmatrix} 0 & 2 & \infty & 2 & \infty & \infty & \infty \\ 2 & 0 & 4 & 4 & 3 & \infty & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 & 5 \\ 2 & 4 & \infty & 0 & 2 & 6 & 9 \\ \infty & 3 & \infty & 2 & 0 & 2 & \infty \\ \infty & \infty & 2 & 6 & 2 & 0 & 1 \\ \infty & \infty & 5 & 9 & \infty & 1 & 0 \end{bmatrix},$$

$$\mathbf{D}^{(2)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 5 & \infty & \infty \\ 2 & 0 & 4 & 4 & 3 & \infty & \infty \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 6 & 9 \\ 5 & 3 & 7 & 2 & 0 & 2 & \infty \\ \infty & \infty & 2 & 6 & 2 & 0 & 1 \\ \infty & \infty & 5 & 9 & \infty & 1 & 0 \end{bmatrix}, \quad \mathbf{D}^{(3)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 5 & 8 & 11 \\ 2 & 0 & 4 & 4 & 3 & 6 & 9 \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 6 & 9 \\ 5 & 3 & 7 & 2 & 0 & 2 & 12 \\ 8 & 6 & 2 & 6 & 2 & 0 & 1 \\ 11 & 9 & 5 & 9 & 12 & 1 & 0 \end{bmatrix},$$

$$\mathbf{D}^{(4)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 8 & 11 \\ 2 & 0 & 4 & 4 & 3 & 6 & 9 \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 6 & 9 \\ 4 & 3 & 7 & 2 & 0 & 2 & 11 \\ 8 & 6 & 2 & 6 & 2 & 0 & 1 \\ 11 & 9 & 5 & 9 & 11 & 1 & 0 \end{bmatrix}, \quad \mathbf{D}^{(5)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 6 & 11 \\ 2 & 0 & 4 & 4 & 3 & 5 & 9 \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 4 & 9 \\ 4 & 3 & 7 & 2 & 0 & 2 & 11 \\ 6 & 5 & 2 & 4 & 2 & 0 & 1 \\ 11 & 9 & 5 & 9 & 11 & 1 & 0 \end{bmatrix},$$

$$\mathbf{D}^{(6)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 6 & 7 \\ 2 & 0 & 4 & 4 & 3 & 5 & 6 \\ 6 & 4 & 0 & 6 & 4 & 2 & 3 \\ 2 & 4 & 6 & 0 & 2 & 4 & 5 \\ 4 & 3 & 4 & 2 & 0 & 2 & 3 \\ 6 & 5 & 2 & 4 & 2 & 0 & 1 \\ 7 & 6 & 3 & 5 & 3 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{D}^{(7)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 6 & 7 \\ 2 & 0 & 4 & 4 & 3 & 5 & 6 \\ 6 & 4 & 0 & 6 & 4 & 2 & 3 \\ 2 & 4 & 6 & 0 & 2 & 4 & 5 \\ 4 & 3 & 4 & 2 & 0 & 2 & 3 \\ 6 & 5 & 2 & 4 & 2 & 0 & 1 \\ 7 & 6 & 3 & 5 & 3 & 1 & 0 \end{bmatrix}.$$

Note that the first row of entries of the matrix $\mathbf{D}^{(7)}$ coincides with the permanent labels of the vertices found in [Table 4.2](#) by means of [Dijkstra's algorithm](#), as expected.

We are now in a position to verify [Floyd's algorithm](#).

Theorem 4.4 *Let G be a weighted graph of order n with vertex set $V(G) = \{v_1, \dots, v_n\}$ which contains no cycles of negative weight.*

- (i) *At termination of [Floyd's algorithm](#), the (i, j) -th element, $d_{ij}^{(n)}$, of the matrix $\mathbf{D}^{(n)}$, is the length of a shortest v_i - v_j path in G , for all $i, j \in [n]$.*
- (ii) *[Floyd's algorithm](#) has a worst-case time complexity of $\mathcal{O}(n^3)$.*

Proof (ii) Let us take the following to be basic operations for the purposes of an analysis of the time complexity of [Algorithm 12](#): variable assignments, comparisons and addition of two real numbers. Three basic operations are performed during each iteration of [Step 4](#) of the algorithm (one addition, one comparison and one assignment). Because [Step 4](#) is nested in a triple **for** loop over the range $1, \dots, n$, the step is repeated n^3 times. Therefore the time complexity of [Algorithm 12](#) is $3n^3 = \mathcal{O}(n^3)$. ■

Part (i) of [Theorem 4.4](#) may be proved by induction over the superscript k in [\(4.1\)](#) as suggested in the discussion leading up to the expression [\(4.1\)](#); the proof is left as an exercise.

- ❖ The reader should now be able to attempt Exercises [4.5–4.7](#) as well as [Project 4.2](#).

4.4 Longest paths in weighted digraphs

Determining a longest path between two specified vertices in a weighted digraph D is an important requirement in many applications, as was illustrated in the introduction to this chapter. In this section we assume that D has no directed cycles of positive length, for otherwise an infinitely long walk may be associated with repeated traversal of any cycle of positive length. Furthermore, we assume that D has exactly one source vertex x (*i.e.* $\text{id}(x) = 0$ while $\text{id}(v) > 0$ for all other vertices $v \in V(D)$), that D contains (directed) paths from x to every other vertex in D and that longest paths are sought from x to all other vertices of D .

The idea behind the algorithm presented in this section for determining such longest directed paths, and given in pseudocode as [Algorithm 13](#), is to assign a label $L(v)$ to each vertex $v \in V(D)$, where $L(v)$ represents the length of a longest (directed) x - v path in D , similar to the labelling process followed in [Dijkstra's algorithm](#) (where the labels there represented the lengths of *shortest* paths uncovered up to that point). Initially all vertices $v \in V(D)$ are labelled $L(v) = 0$, and once a parent has been assigned to a vertex v , denoted by $u = \text{parent}(v)$, the parent represents the predecessor of v on a longest (directed) x - v path in D , and the vertex v receives a new, *permanent* label $L(u) + w(uv)$. The chief difference between the Dijkstra labelling process and the longest path labelling process followed here is that a vertex label never changes after a parent has been assigned to that vertex, whereas labels may be updated (even repeatedly) in [Dijkstra's algorithm](#). Apart from the initial zero-labelling, further (permanent) labels are only assigned to vertices whose in-neighbours have themselves *all* been labelled (permanently) and assigned parents. Eventually each vertex acquires a permanent label and a parent. When a vertex v acquires its permanent label $L(v)$, a longest directed path (of length $L(v)$) from x to v in D is given by

$$P^{(t)} : u_0 u_1 u_2 \cdots u_t \quad (4.2)$$

where $u_0 = x$, $u_t = v$ and $u_{i-1} = \text{parent}(u_i)$ for all $i \in [t]$.

Algorithm 13: An algorithm for computing longest paths in a weighted digraph

Input : A connected, weighted digraph D of order n , with exactly one source vertex $x \in V(D)$ and which contains directed paths from x to all other vertices and no directed cycles of positive weight.

Output : A longest (directed) path from x to every other vertex in D .

```

1  $S \leftarrow \{x\}$ ,  $\text{parent}(x) \leftarrow x$ 
2 foreach  $v \in V(D)$  do  $L(v) \leftarrow 0$ 
3 while  $|S| < n$  do
4   foreach  $v \notin S$  whose in-neighbours are all in  $S$  do
5     Let  $u \in S$  be the in-neighbour of  $v$  maximising  $L(u) + w(uv)$ 
6      $L(v) \leftarrow L(u) + w(uv)$ 
7      $\text{parent}(v) \leftarrow u$ 
8      $S \leftarrow S \cup \{v\}$ 
```

Let us demonstrate the working of [Algorithm 13](#) by an example. Consider finding longest paths from the vertex a to every other vertex in the concert activity digraph $D_{4.2}$ in [Figure 4.4](#). We start with the assignments $S \leftarrow \{a\}$ and $L(v) \leftarrow 0$ for all $v \in \{a, b, c, d, e, f, g\}$ in Steps 1–2.

Since $|S| = 1 \neq 7$ we continue to [Step 4](#), where we identify the vertices $b, c \notin S$ as having *all* their in-neighbours in S . Since each of these vertices has a unique in-neighbour in S (namely the vertex a), we make the assignments $L(b) \leftarrow L(a) + w(ab) = 0 + 3 = 3$, $\text{parent}(b) \leftarrow a$, $S \leftarrow S \cup \{b\}$ and $L(c) \leftarrow L(a) + w(ac) = 0 + 3 = 3$, $\text{parent}(c) \leftarrow a$, $S \leftarrow S \cup \{c\}$ and return to [Step 3](#).

Since $|S| = 3 \neq 7$ we continue to [Step 4](#), where we identify the vertices $d, e \notin S$ as having *all* their in-neighbours in S . Since d has a unique in-neighbour in S (namely the vertex b), we make the assignments $L(d) \leftarrow L(b) + w(bd) = 3 + 1 = 4$, $\text{parent}(d) \leftarrow b$, $S \leftarrow S \cup \{d\}$. The vertex e , however, has two in-neighbours in S (namely a and c). Because $3 + 4 = L(c) + w(ce) > L(a) + w(ae) = 0 + 3$ we make the assignments $L(e) \leftarrow L(c) + w(ce) = 3 + 4 = 7$, $\text{parent}(e) \leftarrow c$, $S \leftarrow S \cup \{e\}$ and return to [Step 3](#).

Since $|S| = 5 \neq 7$ we continue to [Step 4](#), where we identify the vertex $f \notin S$ as having *all* its in-neighbours in S . The vertex f has two in-neighbours in S (namely c and d). Because $4 + 5 = L(d) + w(df) > L(c) + w(cf) = 3 + 4$ we make the assignments $L(f) \leftarrow L(d) + w(df) = 4 + 5 = 9$, $\text{parent}(f) \leftarrow d$, $S \leftarrow S \cup \{f\}$ and return to [Step 3](#).

Since $|S| = 6 \neq 7$ we continue to [Step 4](#), where we identify the vertex $g \notin S$ as having *all* its in-neighbours in S . The vertex g has two in-neighbours in S (namely e and f). Because $7 + 7 = L(e) + w(eg) > L(f) + w(fg) = 9 + 2$ we make the assignments $L(g) \leftarrow L(e) + w(eg) = 7 + 7 = 14$, $\text{parent}(g) \leftarrow e$, $S \leftarrow S \cup \{g\}$ and return to [Step 3](#).

Since $|S| = 7$ the algorithm terminates. The results obtained are summarised in [Table 4.3](#). A longest path from a to b (c, d, e, f or g respectively) is $P^{(b)} : ab$ ($P^{(c)} : ac$; $P^{(d)} : abd$; $P^{(e)} : ace$; $P^{(f)} : abdf$ or $P^{(g)} : aceg$ respectively) of length 3 (3, 4, 7, 9 or 14 respectively).

i	a	b	c	d	e	f	g	S
0	$(0, a)$	$(0, -)$	$(0, -)$	$(0, -)$	$(0, -)$	$(0, -)$	$(0, -)$	$\{a\}$
1	$(0, a)$	$(3, a)$	$(3, a)$	$(0, -)$	$(0, -)$	$(0, -)$	$(0, -)$	$\{a, b, c\}$
2	$(0, a)$	$(3, a)$	$(3, a)$	$(4, b)$	$(7, c)$	$(0, -)$	$(0, -)$	$\{a, b, c, d, e\}$
3	$(0, a)$	$(3, a)$	$(3, a)$	$(4, b)$	$(7, c)$	$(9, d)$	$(0, -)$	$\{a, b, c, d, e, f\}$
4	$(0, a)$	$(3, a)$	$(3, a)$	$(4, b)$	$(7, c)$	$(9, d)$	$(14, e)$	$\{a, b, c, d, e, f, g\}$

Table 4.3: Consecutive steps when [Algorithm 13](#) is applied to the digraph $D_{4.2}$ in [Figure 4.4](#). The value of i denotes the number of times that Steps 3–8 have been repeated. The first entry in each ordered pair is the label L of the vertex at the column head, whilst the second entry in each ordered pair is the parent of the vertex at the column head in a longest path from a to that vertex.

We are now in a position to verify the correct working of [Algorithm 13](#).

Theorem 4.5 *Let D be a weighted digraph of order n with exactly one source vertex x which contains a (directed) path from x to every other vertex in D and no directed cycles of positive weight.*

- (i) *At termination of [Algorithm 13](#), the label $L(v)$ is the length of a longest (directed) x - v path in D for all $v \in V(D)$.*
- (ii) *[Algorithm 13](#) has a worst-case time complexity of $\mathcal{O}(n^2)$.*

Proof (i) Denote by S_i the set of all vertices of D in the set S after the first i iterations of the **while** loop spanning Steps 3–8 of the algorithm, with the convention that $S_0 = \{x\}$. We first show that the algorithm always terminates, *i.e.* that eventually $S_\Omega = V(D)$ for some $\Omega \in \mathbb{N}$. Suppose that $S_j \setminus S_{j-1} = \emptyset$, while $V(D) \setminus S_j \neq \emptyset$ for some j (*i.e.* that no vertices enter the set S during iteration j of

the **while** loop spanning Steps 3–8, but that there are still permanently unlabelled vertices at iteration j). Let $v_j \in V(D) \setminus S_j$. Then v_j has an in-neighbour, $v_{j,1}$ (say), which is also in the set $V(D) \setminus S_j$ (otherwise v_j would already have entered the set S_j). But then $v_{j,1}$ similarly has an in-neighbour, $v_{j,2}$ (say), which is also in the set $V(D) \setminus S_j$ (otherwise $v_{j,1}$ would already have entered the set S_j). Continuing this argument, a directed walk $W : \cdots v_{j,3} v_{j,2} v_{j,1} v_j$ is formed in D . Because D is finite, some vertex will be repeated in W , resulting in a directed cycle, which is a contradiction. We conclude that $S_j \setminus S_{j-1} \neq \emptyset$ for every iteration j of Step 4 of the algorithm. Hence the algorithm will terminate, again by the finiteness of D , and each vertex $v \in V(D)$ will have a permanent label $L(v)$ at termination of the algorithm.

We now proceed, by induction on the index i , to show that if $v \in S_i$ for some i , then a directed x - v path produced by Algorithm 13 is a longest directed x - v path in D , i.e. that $L(v)$ denotes the length of a longest directed x - v path in D for all $v \in S_i$.

Consider, as base case, the directed x - v paths produced by Algorithm 13 for all $v \in S_1$. The only path thus produced (containing no arcs) is the trivial x - x path $P^{(0)} : x$, which is the only directed x - x path in D , and hence also a longest directed x - x path in D (i.e. the label $L(x) = 0$ is indeed the length of a longest directed x - x path in D). Any x - v path $P^{(1)} : xv$ produced by Algorithm 13 containing exactly one arc is also a longest directed x - v path in D , because such a path would have been generated during the first iteration of the **while** loop spanning Steps 3–8 of the algorithm, at which time the source vertex x would have been the only member of $S = S_0$, indicating that the vertex v has no in-neighbours other than x (i.e. the label $L(v) = w(xv)$ is also the length of a longest directed x - v path in D for all $v \in S_1 \setminus S_0$).

Assume next, as induction hypothesis, that directed x - v paths produced by Algorithm 13 are longest directed x - v paths in D , for all $v \in S_{t-1}$ (i.e. that the labels $L(v)$ denote the length of the longest directed x - v paths in D for all $v \in S_{t-1}$).

Now suppose, for the induction step, that a directed x - u_t path $P^{(t)} : u_0 u_1 u_2 \dots u_{t-1} u_t$ with $u_0 = x$, of length $w(P^{(t)}) = L(u_t)$ and containing exactly t arcs, is produced during iteration t of the **while** loop spanning Steps 3–8 of Algorithm 13 (i.e. $u_t \in S_t$). Then there is no directed x - v path in D containing more than t arcs, and because

$$\begin{aligned} L(u_t) &= \max_u \{L(u) + w(uu_t)\} \\ &= \max_u \{(\text{length of a longest } x\text{-}u \text{ path with at most } t-1 \text{ arcs}) + w(uu_t)\} \\ &= \max_u \{(\text{length of a longest } x\text{-}u \text{ path}) + w(uu_t)\}, \end{aligned}$$

where the maximum is taken over all in-neighbours $u \in S_{t-1}$ of u_t , it follows that $L(u_t)$ is the length of a longest directed x - u_t path in D .

(ii) Let us take the following to be basic operations for the purposes of an analysis of the time complexity of Algorithm 13: all variable assignments (including inserting set members), **if** comparisons and addition of two real numbers. Then the time complexity of Steps 1–2 of Algorithm 13 is $\mathcal{O}(n)$, since n label assignments and one set member insertion are made, and these steps are executed only once. Step 3 has constant time complexity because of the single **if** comparison, i.e. a time

complexity of $\mathcal{O}(1)$. At most $n - 1$ if comparisons involving the vertex u have to be considered (and therefore at most $n - 1$ additions have to be performed) during the maximisation process in Step 5, after which two variable assignments and one set insertion have to be performed. Therefore the time complexity of one traversal of Steps 3–8 of the algorithm is $1 + 2(n - 1) + 3 = \mathcal{O}(n)$. Steps 3–8 of the algorithm are repeated at most $n - 1$ times, and hence the portion of the algorithm represented by Steps 3–8 has an overall worst-case time complexity of $\mathcal{O}(n^2)$, dominating that of Steps 1–2. We conclude that the worst-case time complexity of the entire algorithm is therefore $\mathcal{O}(n^2)$. ■

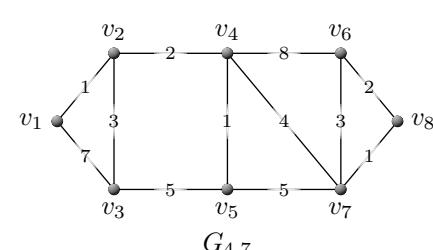
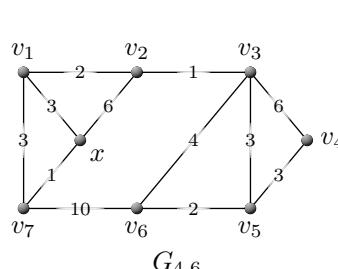
- ❖ The reader should now be able to attempt Exercise 4.8 as well as Projects 4.3–4.4.

Exercises

- 4.1 Find, by inspection, a favourable (relatively short) path from h to a in the graph model $G_{4.1}$ of the *Mountain View* street map in Figure 4.1.
- 4.2 Let $G_{4.5}$ be the weighted graph with vertex set $V(G_{4.5}) = \{v_1, \dots, v_7\}$ and weight matrix

$$\mathbf{W}(G_{4.5}) = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \\ v_1 & 0 & 8 & 1 & 4 & 2 & 7 & \infty \\ v_2 & 8 & 0 & 7 & 1 & \infty & \infty & 4 \\ v_3 & 1 & 7 & 0 & \infty & 2 & \infty & \infty \\ v_4 & 4 & 1 & \infty & 0 & \infty & 3 & 6 \\ v_5 & 2 & \infty & 2 & \infty & 0 & 5 & \infty \\ v_6 & 7 & \infty & \infty & 3 & 5 & 0 & 4 \\ v_7 & \infty & 4 & \infty & 6 & \infty & 4 & 0 \end{bmatrix}.$$

- (a) Draw the weighted graph $G_{4.5}$.
 - (b) Find, by inspection, the distance from v_3 to every other vertex of $G_{4.5}$.
 - (c) Find, by inspection, a shortest v_3 - v_2 path and a shortest v_3 - v_7 path in $G_{4.5}$.
- 4.3 Determine the weight matrix for
 - (a) the graph $G_{4.6}$ below;
 - (b) the graph $G_{4.7}$ below.



4.4 Use Algorithm 11 to compute

- (a) $d_{G_{4.6}}(x, v_i)$ and to determine a shortest $x-v_i$ path in $G_{4.6}$ above for each $v_i \in V(G_{4.6})$;
- (b) $d_{G_{4.7}}(v_1, v_i)$ and to determine a shortest v_1-v_i path in $G_{4.7}$ above for each $v_i \in V(G_{4.7})$.
- 4.5 A company has branches in each of six cities C_1, C_2, \dots, C_6 . The fare for a direct flight from C_i to C_j is given by the (i, j) -th entry in the matrix \mathbf{C} (where ∞ indicates that there is no direct flight). Use Algorithm 12 to find the cheapest fare between every pair of cities, if

$$\mathbf{C} = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\ C_1 & 0 & 500 & \infty & 400 & 250 & 100 \\ C_2 & 500 & 0 & 150 & 200 & \infty & 250 \\ C_3 & \infty & 150 & 0 & 100 & 200 & \infty \\ C_4 & 400 & 200 & 100 & 0 & 100 & 250 \\ C_5 & 250 & \infty & 200 & 100 & 0 & 550 \\ C_6 & 100 & 250 & \infty & 250 & 550 & 0 \end{bmatrix}.$$

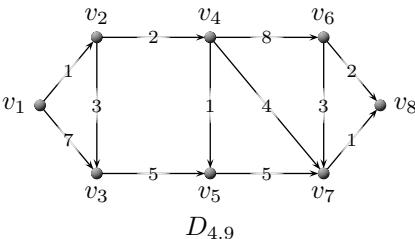
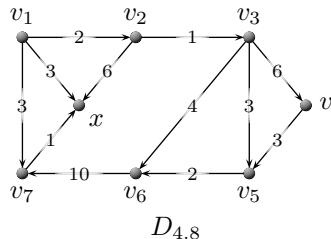
- 4.6 Consider a weighted graph G whose weights are all positive and form two new weighted graphs H and I with the same adjacency structure as that of G , but such that each edge weight in H is the reciprocal of the corresponding weight in G , and such that each edge weight in I is the negative of the corresponding weight in G . Prove or disprove each of the following statements:

- (a) Every shortest path in H is a longest path in G ;
 (b) Every shortest path in I is a longest path in G .

4.7 Prove Theorem 4.4(i).

4.8 Use Algorithm 13 to find a longest (directed) path

- (a) from v_1 to v_4 in the digraph $D_{4.8}$ below;
 (b) from v_1 to v_8 in the digraph $D_{4.9}$ below.



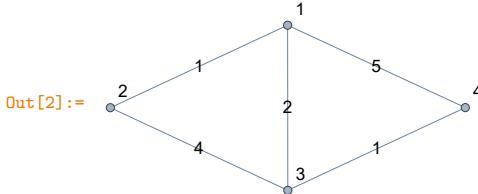
Computer exercises

A weighted graph may be created in **MATHEMATICA** by specifying its weight matrix. This may be accomplished by the command `WeightedAdjacencyGraph[A, VertexLabels -> "Name", EdgeLabels -> "EdgeWeight"]`. Here A denotes the weight matrix of the graph to be generated, with the convention that the entry **Infinity** (∞) in the (i,j) -th entry of A means that the i -th and j -th vertices of the graph are not adjacent. There is, however, an anomaly here, namely that a zero weight is not interpreted by **MATHEMATICA** as the absence of an edge — it means that there is an edge, but with weight zero. Therefore, a loop at the i -th vertex of a weighted graph may be avoided by specifying the (i,i) -th entry of the weight matrix A as **Infinity** rather than zero. The attribute specification `VertexLabels -> "Name"` causes the vertices of the weighted graph to be labelled (numbered). The attribute specification `EdgeLabels -> "EdgeWeight"` similarly causes the edges to be labelled with their weights. For example, the commands

```
In[1]:= m = {{Infinity, 1, 2, 5}, {1, Infinity, 4, Infinity}, {2, 4, Infinity, 1},
           {5, Infinity, 1, Infinity}}
In[2]:= G = WeightedAdjacencyGraph[m, VertexLabels -> "Name", EdgeLabels ->
           "EdgeWeight"]
```

produce the output:

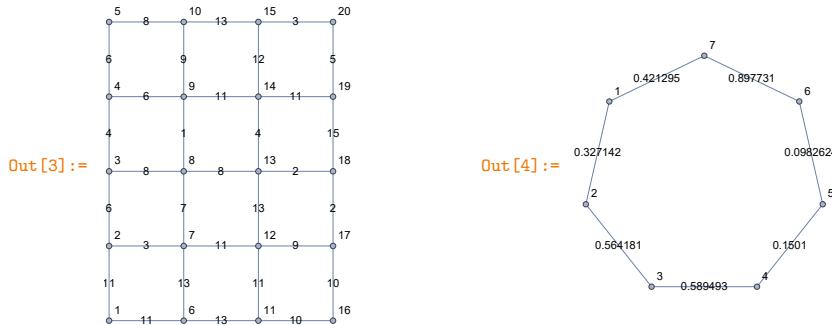
```
Out[1]:= {{∞, 1, 2, 5}, {1, ∞, 4, ∞}, {2, 4, ∞, 1}, {5, ∞, 1, ∞}}
```



An $s \times t$ grid graph (*i.e.* the cartesian product $P_s \times P_t$ of two paths P_s and P_t) is generated by means of the command `Gridgraph[{s,t}]`. The commands `RandomInteger[{a,b},m]` and `RandomReal[{a,b},m]`, furthermore, return as output sequences of m random integers in the range $a, a+1, \dots, b-1, b$ and random real numbers in the interval $[a, b]$, respectively. The edge weights of a weighted graph of size m may therefore be assigned random integer or random real values within specified ranges by including the attributes `EdgeWeight->RandomInteger[!a,b,m]!` or `RandomReal[{a,b},m]`, respectively, in the weighted graph generating command. For example, the commands

```
In[3]:= H = GridGraph[{5, 4}, EdgeWeight -> RandomInteger[{1, 15}, 5 3 + 4 4],
                     VertexLabels -> "Name", EdgeLabels -> "EdgeWeight"]
In[4]:= F = CycleGraph[7, EdgeWeight -> RandomReal[{0, 1}, 7], VertexLabels ->
           "Name", EdgeLabels -> "EdgeWeight"]
```

produce the output:

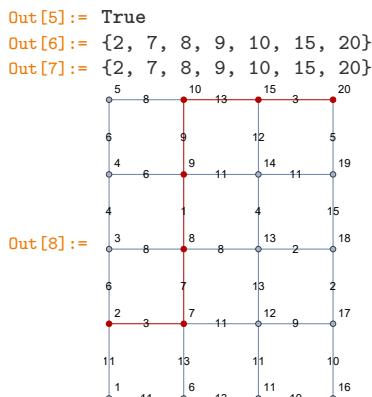


The command `WeightedGraphQ[G]` returns the boolean value `True` if **MATHEMATICA** interprets the object G as a weighted graph, or the boolean value `False` otherwise. The command `FindShortestPath[G, a, b]` may furthermore be used to determine a shortest path from a vertex a to vertex b in a (weighted and/or directed) graph G . **MATHEMATICA** uses either **Dijkstra's algorithm** (in the case of strictly positive edge weights) or the Bellman-Ford algorithm (if negative edge weights are present), but the algorithm to be used may be specified by including the attribute `Method -> "Dijkstra"` or `Method -> "BellmanFord"` in the command `FindShortestPath`. A shortest path p found between two specified vertices in G by means of the command `FindShortestPath` may be highlighted in a graphical representation of G by invoking the command `HighlightGraph[G, p]`. The command `GraphDistanceMatrix[G]`, furthermore, returns the distance matrix of the graph G (as would be the output of **Floyd's algorithm**). Alternatively, `GraphDistance[G, a]` returns a list of distances from a specified vertex a to all other vertices of G . Again, the algorithms of **Dijkstra** or Bellman-Ford may be specified as methods within the latter two commands. For example, the commands

```

In[5]:= WeightedGraphQ[G]
In[6]:= FindShortestPath[H, 2, 20]
In[7]:= p = FindShortestPath[H, 2, 20, Method -> "Dijkstra"]
In[8]:= HighlightGraph[H, PathGraph[p]]
In[9]:= MatrixForm[GraphDistanceMatrix[F, Method -> "Dijkstra"]]
In[10]:= GraphDistance[F, 2, Method -> "Dijkstra"]
  
```

produce the output:



```

Out[9]:= {{0., 0.327142, 0.891323, 1.48082, 1.41729, 1.31903, 0.421295},
          {0.327142, 0., 0.564181, 1.15367, 1.30377, 1.40204, 0.748437},
          {0.891323, 0.564181, 0., 0.589493, 0.739593, 0.837855, 1.31262},
          {1.48082, 1.15367, 0.589493, 0., 0.1501, 0.248363, 1.14609},
          {1.41729, 1.30377, 0.739593, 0.1501, 0., 0.0982624, 0.995993},
          {1.31903, 1.40204, 0.248363, 0.0982624, 0., 0.897731, 0.},
          {0.421295, 0.748437, 1.31262, 1.14609, 0.995993, 0.897731, 0.}}
Out[10]:= {0.327142, 0., 0.564181, 1.15367, 1.30377, 1.40204, 0.748437}

```

Projects

This section contains four projects. The first two of these projects highlight applications of computing shortest paths in graphs, while the focus in last two projects is on applications of computing longest (directed) paths in digraphs. The first project involves application of [Dijkstra's algorithm](#) to find a cheapest path through a maze and also a shortest path from the hospital to the accident scene along the streets of the *Tshwane* suburb of *Mountain View*, depicted in the map of [Figure 4.1](#). The objective in the second project is to compute an inter-distance table for the major towns and cities in South Africa, by utilising [Floyd's algorithm](#). The last two projects require the use of [Algorithm 13](#) to compute longest (directed) paths in a digraph with the aim of maximising the expected return on a simple financial investment and scheduling the tasks involved in the project of building a factory.

Project 4.1: Shortest paths using Dijkstra's algorithm

Recall from [Section 4.3.1](#) that [Dijkstra's algorithm](#) ([Algorithm 11](#)) may be used to compute shortest paths from a specified starting vertex to all other vertices in a connected, weighted graph.

Tasks

1. Implement [Algorithm 11](#) in [MATHEMATICA](#) as a function taking a graph G of order n with vertex set $V(G) = \{v_1, \dots, v_n\}$ and a specified vertex $v \in V(G)$ as input, and producing a list L of length n as output, whose i -th entry is the pair (d_i, p_i) , where d_i denotes the distance between v and the vertex $v_i \in V(G)$ and where p_i is the predecessor of v_i (*i.e.* $\text{parent}(v_i)$) on a shortest $v-v_i$ path in G .
2. Consider the maze in [Figure 4.7](#) and suppose that a treasure hunter has to start in the room marked “Start,” finding his way through the various rooms of the maze to the chamber marked “Treasury,” each time paying the ransom indicated in the figure when passing through a doorway, which is to be deducted from his prize when he reaches the treasury. Model the adjacency of maze chambers by a weighted graph G whose vertices represent the chambers, in which two vertices are adjacent if there is a doorway between the corresponding maze chambers and whose edge weights correspond to the amounts of ransom to be paid when passing through a doorway.
 - (a) Draw the weighted graph G .
 - (b) Determine the weight matrix $\mathbf{W}(G)$ of G .
 - (c) Use your implementation in [Task 1](#) to find a cheapest route through the maze from the starting point to the treasury.

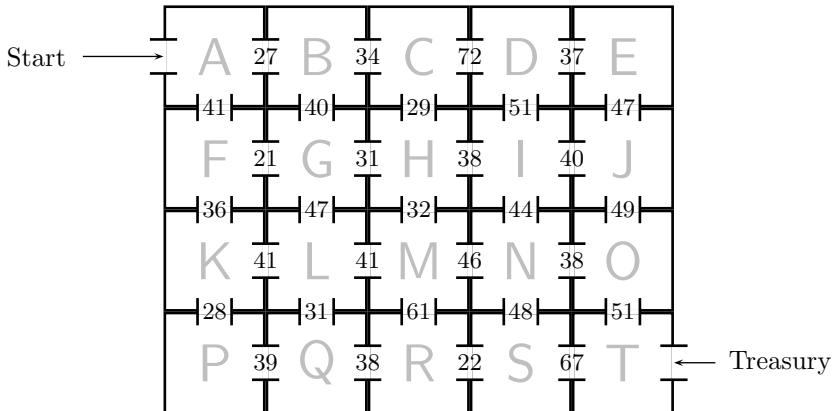


Figure 4.7: An amazing treasure hunt.

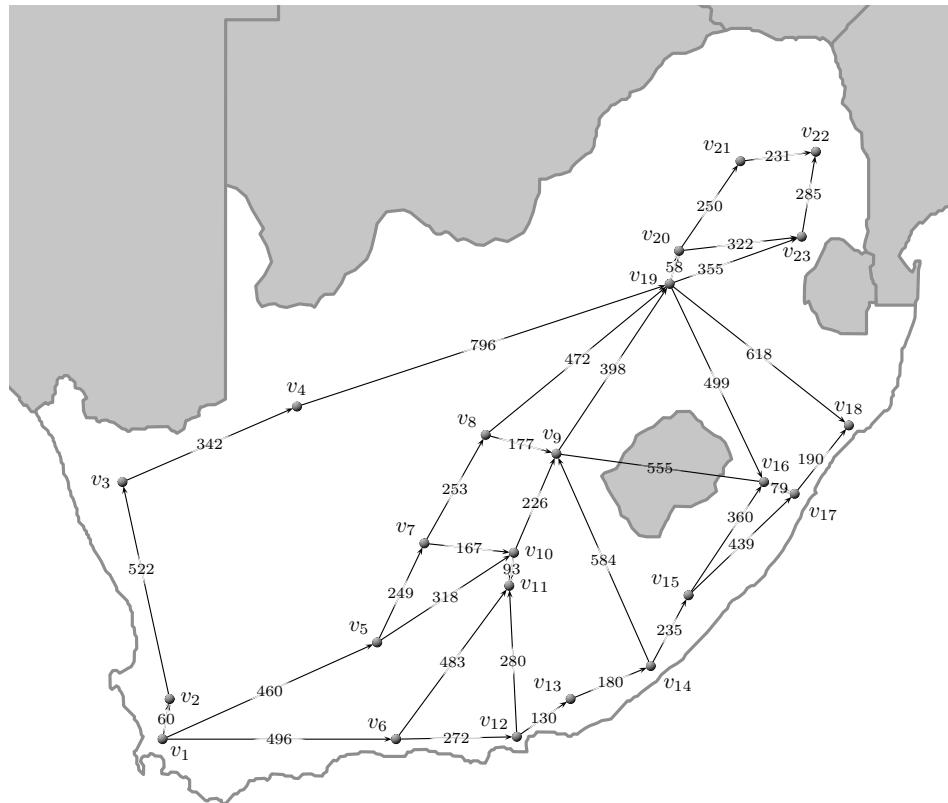
- (d) Verify the correctness of your result in [Task 2\(c\)](#) by means of the built-in [MATHEMATICA](#) command `FindShortestPath[G, Start, Treasury, Method -> "Dijkstra"]`.
3. (a) Determine the weight matrix $\mathbf{W}(G_{4.1})$ of the weighted graph $G_{4.1}$ in [Figure 4.2](#).
- (b) Use your implementation in [Task 1](#) to find a shortest path to be followed by an ambulance from the hospital (corner of DAPHNE and FELIX streets) in [Figure 4.1](#) to an accident scene at the corner of IVOR and DANIEL streets.
- (c) Verify the correctness of your result in [Task 3\(b\)](#) by means of the built-in [MATHEMATICA](#) command `FindShortestPath[G_{4.1}, h, a, Method -> "Dijkstra"]`.

Project 4.2: Shortest paths using Floyd's algorithm

Consider the map of South Africa shown in [Figure 4.8](#). A graph G has been superimposed on the map. The vertices of the graph represent major South African cities or towns, whilst edges show where there are direct road links between towns. The weights of the edges denote the distances along these road links, measured in kilometres.

Tasks

1. Determine the weight matrix $\mathbf{W}(G)$ of G .
2. Implement [Algorithm 12](#) in [MATHEMATICA](#) as a function taking a graph G of order n with vertex set $V(G) = \{v_1, \dots, v_n\}$ as input, and producing as output a distance matrix $\mathbf{D}(G)$ of G whose (i, j) -th entry represents the distance between the vertices $v_i, v_j \in V(G)$.
3. Verify the correctness of your implementation in [Task 2](#) by invoking the built-in [MATHEMATICA](#) command `GraphDistanceMatrix[G]`.



^aFormerly known as Port Elizabeth and then as Nelson Mandela Bay.

^bFormerly known as Pretoria.

^cFormerly known as Pietersburg.

Figure 4.8: Distance map of South Africa.

4. Use your results of Tasks 2 and 3 to produce a distance table of the form:

	Beaufort West	Bloemfontein	Britstown	...
Beaufort West	—	*	*	
Bloemfontein	*	—	*	
Britstown	*	*	—	
:				

Project 4.3: Investment planning via longest paths

Suppose you have x thousand euros to invest and that three investment opportunities are available. Naturally you would like to know how much money you should invest in each investment opportunity so as to maximise the expected return on your investment. Suppose it is known that if an amount of x_i (measured in thousands of euros) is invested in investment opportunity i , then a return of $r_i(x_i)$ thousands of euros is expected, where

$$\begin{aligned} r_1(x_1) &= 3x_1 + 4, & (x_1 > 0) \\ r_2(x_2) &= 7x_2 + 1, & (x_2 > 0) \\ r_3(x_3) &= 5x_3 + 4, & (x_3 > 0) \\ r_1(0) = r_2(0) = r_3(0) &= 0. \end{aligned}$$

For example, if you invested €1 000 in investment opportunity 1, you expect a return of €7 000.

Let us take $x = 6$ (*i.e.* we assume that you have a total of €6 000 to invest). Then the investment problem may be modelled by the digraph $D_{4.10}$ shown in Figure 4.9. In this digraph, the vertex (i, m) represents the situation where m thousand euros remain available for investment opportunity i , after money has been invested in lower indexed investment opportunities. The arc joining the vertices (i, m) and $(i+1, m-x_i)$ has a weight of $r_i(x_i)$ corresponding to the return expected when investing x_i thousand euros in investment opportunity i , as explained above.

For example, you may invest up to €6 000 in investment opportunity 1. This corresponds to the vertex $(1, 6)$ in $D_{4.10}$. If you were to invest €4 000 in investment opportunity 1, then €2 000 would remain to be invested in investment opportunities 2 and 3 combined. This situation corresponds to the vertex $(2, 2)$ in $D_{4.10}$. If you were further to invest €1 000 in investment opportunity 2, then €1 000 would remain to be invested in investment opportunity 3, since the whole amount has to be invested. The path corresponding to the above mentioned investment strategy is $P : (1, 6)(2, 2)(3, 1)(4, 0)$, which is expected to yield a return of €33 000 (that is, $r_1(4) + r_2(1) + r_3(1) = 16 + 8 + 9 = 33$). This is not necessarily the highest return that may be realised; an optimal return would correspond to a longest path from $(1, 6)$ to $(4, 0)$ in $D_{4.10}$.

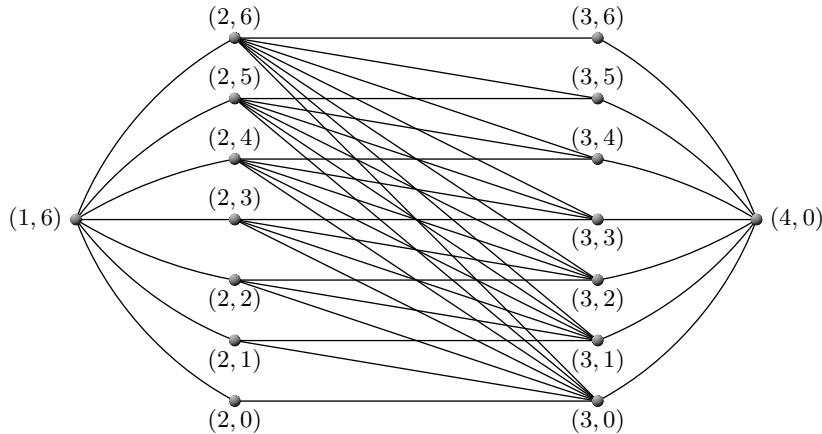


Figure 4.9: Investment strategy digraph, $D_{4,10}$. Arrow heads indicating the directions of arcs have been omitted to avoid cluttering the figure. All arcs are, however, directed towards the right.

Tasks

1. Implement [Algorithm 13](#) in **MATHEMATICA** as a function taking a digraph D and two specified vertices $v, w \in V(D)$ as input, and producing a pair L, ℓ as output, where L is a list whose i -th entry is the vertex v_i on a longest $v-w$ path $P : v_0 v_1 \cdots v_k$ where $v_0 = v$ and $v_k = w$ for some $k \in \mathbb{N}$, and where ℓ denotes the length of P in D .
2. Use your implementation in [Task 1](#) to find a longest (directed) path in the digraph $D_{4,10}$ from $(1, 6)$ to $(4, 0)$. What is the corresponding optimal investment strategy?

Project 4.4: Project scheduling via longest paths

A new factory must be built. The building process consists of thirteen different activities, as summarised in [Table 4.4](#).

Before inauguration of the factory, the safety inspection must be completed, during which all painting jobs, electrical wiring, working of machines and conveyor belts as well as the lawn and roofing qualities will be inspected. Precision Painting will be responsible for painting both the warehouse (12 working days required) and the wall panels (16 working days required) — these painting jobs may be undertaken in parallel. Electrical wiring must be completed before machines and conveyor belts may be installed. The installation of the machinery and conveyor belts may be started as soon as the roof has been completed — so as to provide protection against rainy weather. The wall panels do not have to be installed before the roof can be built. The wall pillars must, however, be sunk into the cement floor before either the wall panels may be installed or the roof may be built. The electrical wiring and setting of the cement floor must be completed before the wall pillars may be sunk. Construction of the warehouse may begin once the road and parking lot have been completed. After the plumbing has been completed, work on the cement floor, road, parking area and electrical wiring may

	Duties	Contractor	Duration
1	Plumbing	ABC Pipes & Works	10 days
2	Gardening (lawns)	Cape Envirogrow	6 days
3	Cement floor construction	PPC Cite Services	5 days
4	Roads and parking layout	CoroBrick Ltd.	15 days
5	Sinking wall panel pillars	Somerset Structures	7 days
6	Wall panel installation	Cape Panelling Co.	16 days
7	Roof construction	Peninsula Roofing	4 days
8	Electrical wiring	Delta Electricians	12 days
9	Warehouse construction	NDA Construction Eng.	28 days
10	Conveyor belt installation	United Conveyor Belts	22 days
11	Installation of machinery	Swiss Industrial Eng.	32 days
12	Painting of building	Precision Painting	16 days
13	Safety inspection	SABS Consultants	4 days

Table 4.4: Durations of activities in a factory building project.

be started. Finally, the electrical wiring must be completed before any gardening may commence.

Tasks

1. Set up an activity digraph for building the factory, as described in [Section 4.1](#).
2. Use your implementation in [Task 1 of Project 4.3](#) to find a critical path in the activity digraph.
3. Are there any *other* critical paths in the activity digraph? Motivate.
4. Use the critical paths uncovered in Tasks 2 and 3 to answer the following questions:
 - (a) What is the earliest date for the inauguration of the factory if the project is started on the first Monday of October and a Monday to Friday work week is maintained?
 - (b) If the painting of the wall panels is completed three days later than scheduled, will it have any effect on the inauguration date (if all other activities are completed on time)?
 - (c) If the painting of the warehouse is completed three days later than scheduled, will it have any effect on the inauguration date (if all other activities are completed on time)?
 - (d) What time delay is acceptable during installation of the conveyor belts without affecting the inauguration date (if all other activities are completed on time)?
 - (e) What time delay is acceptable when laying the cement floor, without affecting the inauguration date (if all other activities are completed on time)?
 - (f) What time delay is acceptable when sinking the wall pillars without affecting the inauguration date (if all other activities are completed on time)?

Further reading

- [1] F Buckley and F Harary, 1990. *Distance in Graphs*, Addison-Wesley, Reading (MA), pp. 270–274.
- [2] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY), pp. 38–43, 104–110.
- [3] JO Clark and DA Holton, 1991. *A First Look at Graph Theory*, World Scientific Publishing Company, Singapore, pp. 69–78.
- [4] EW Dijkstra, 1959. *A note on two problems in connexion with graphs*, Numerische Mathematik, **1(1)**, pp. 269–271.
- [5] RW Floyd, 1962. *Algorithm 97: Shortest path*, Communications of the ACM, **5(6)**, p. 345.
- [6] A Gibbons, 1985. *Algorithmic Graph Theory*, Cambridge University Press, Cambridge.



Trees

Contents

5.1	Introduction	115
5.2	Properties of trees	116
5.3	Constructing minimum spanning trees	119
5.4	Rooted trees	127
5.5	Depth-first tree searches	130
	Exercises	134
	Computer exercises	136
	Projects	138
	Further reading	144

5.1 Introduction

Suppose we have a collection of n cities, and that we wish to construct a railroad system connecting them in such a way that a passenger must be able to travel from any station in the system to any other station. Assume that we know the cost of building railroad tracks between every two cities. How can we build the railroad system as cheaply as possible in such a way that all railroad junctions occur at cities?

The desired railway system may be represented by a weighted graph whose vertices correspond to the cities involved and in which two vertices are adjacent if a railroad track runs between the two corresponding cities. The edges are labelled (weighted) with the proposed costs of the corresponding railroad tracks. To solve this problem, we must find a connected subgraph of this weighted graph (so that any station can be reached from any other station) with the minimum possible associated sum of edge weights. This is called the **Minimum Connector Problem**.

Note that to construct a subgraph with minimum edge weight sum, we must avoid cycles, since otherwise we could remove the most expensive edge (*i.e.* one of largest weight) on every cycle, obtaining a new connected graph (see [Theorem 2.6](#)) with smaller edge weight sum. Hence, it would be possible to leave out a set of tracks between two cities and still have all cities connected, implying that the original railway system was not the cheapest after all. Thus, what we desire is a *connected, acyclic* subgraph with minimum possible edge weight sum. Before

solving the Minimum Connector Problem, we need to investigate in more detail the type of graph we have just described.

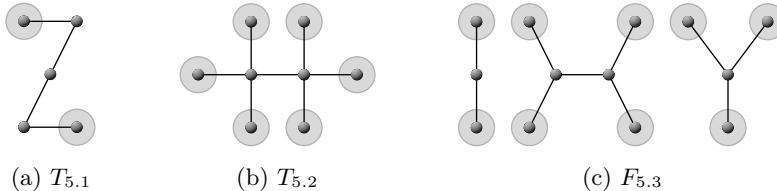


Figure 5.1: Examples of trees and forests: (a) A tree of order 5 and size 4 with two leaves; (b) A tree of order 8 and size 7 with six leaves; (c) A forest of order 13 and size 10 with nine leaves, consisting of three components (trees).

A **tree** is a connected graph which contains no cycles. A **forest** is a graph that has no cycles (and so each component of a forest is a tree). A **leaf** of a tree T is an end-vertex of T (*i.e.* a vertex of degree 1). Graphical representations of two trees and a forest are shown in [Figure 5.1](#). In each case the leaves are circled.

Trees are the simplest connected graphs. These graphs are perhaps the most useful of all special classes of graphs. Recently, there has been considerable interest in tree structures arising in the computer sciences and in artificial intelligence. We often encounter such structures when organising data in a computer memory store or when organising the flow of information through a system. Trees also have applications in chemistry (for example, in counting the alkane hydrocarbons C_nH_{2n+2}), sociology, and many other fields. The sorting of mail according to zip code is also performed according to a tree (called a decision tree). In computer programming and switching theory, we frequently deal with decision trees which contain only two alternatives at each intermediate vertex. Management and genealogical ancestry hierarchies are often also represented by trees, as illustrated in [Figure 5.2](#). Both the theory and applications of trees are considered in this chapter.

5.2 Properties of trees

In this section we present several results on the properties of trees. Since a tree contains no cycles, it follows from [Theorem 2.6](#) that every edge of a tree is a bridge. Trees have several pleasing properties — for example, any two vertices of a tree are connected by exactly one path, as we now show.

Theorem 5.1 *A graph T is a tree if and only if every two distinct vertices of T are connected by a unique path.*

Proof If T is a tree, then by definition it is connected. Hence, any two vertices are connected by at least one path. Suppose, to the contrary, however, that there are vertices u and v of T that are connected by two or more different paths. Let P and Q be two different u - v paths in T . Then there is a vertex x (possibly, $x = u$) on both paths such that the vertex immediately following x on P is different from the vertex immediately following x on Q . Let y be the first vertex of P following x that also belongs to Q (possibly, $y = v$). Then the section of P from x to y and the section of Q from x to y produce two x - y paths that have only x and y in

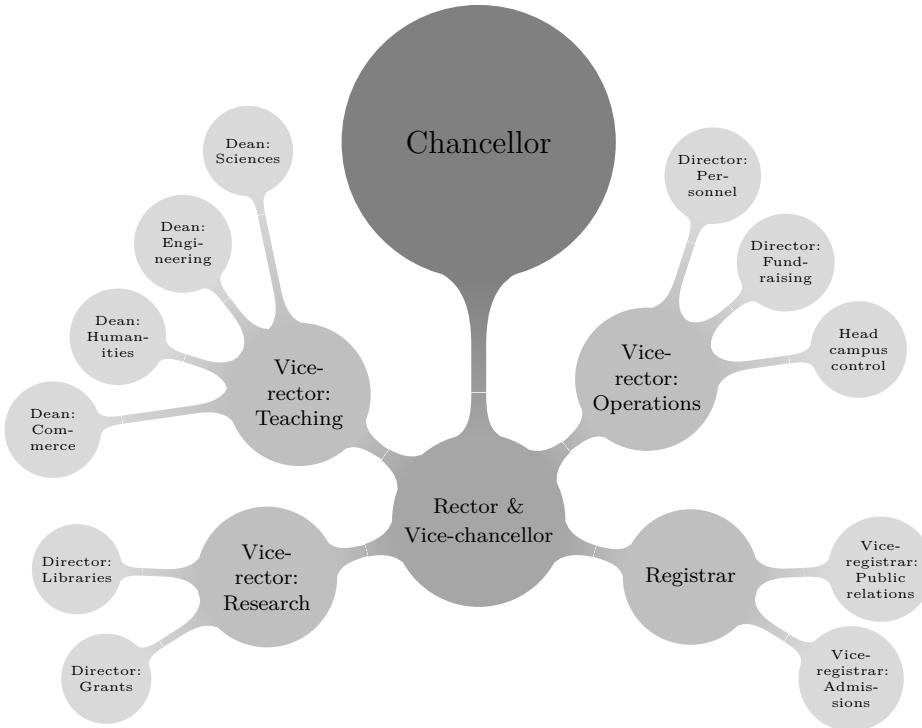


Figure 5.2: The hierarchy of a university management structure may be represented by a tree in which vertices denote managerial posts and in which the edges indicate a “line of command” of sorts.

common. These two paths produce a cycle in T , which contradicts the fact that T is a tree. Hence every two distinct vertices of T are connected by a unique path.

Conversely, suppose T is a graph in which any two distinct vertices are connected by a unique path. This implies that T is connected. If T has a cycle containing vertices u and v , then u and v are connected by at least two paths, contradicting our hypothesis. Hence T is acyclic, and therefore a tree. ■

The sizes of the trees shown in Figure 5.1 are all one less than their orders. We now verify this simple, but important, relationship between the number of vertices and the number of edges in any tree.

Theorem 5.2 *A tree T of order n has size $n - 1$.*

Proof We proceed by the strong form of induction on n . If $n = 1$, then $T \cong K_1$ and T has size 0, as required. Let $n \geq 2$ be an integer, and suppose the result is true for all trees of order less than n . Let T be a tree of order n and size m , and let $e = uv$ be an edge of T . Since every edge of a tree is a bridge, the graph $T - e$ is disconnected. In fact, $T - e$ is a forest with exactly two components (see Exercise 5.6), namely, a tree T_1 containing u , and a tree T_2 containing v . Suppose T_i has order n_i and size m_i for $i \in \{1, 2\}$. Then $n_1, n_2 < n$. Hence, by the induction hypothesis, $m_i = n_i - 1$ for $i \in \{1, 2\}$. Since $n = n_1 + n_2$ and $m = m_1 + m_2 + 1$,

we have that

$$m = (n_1 - 1) + (n_2 - 1) + 1 = n_1 + n_2 - 1 = n - 1.$$

Thus, by induction, the size of a tree is one less than its order. ■

We are now in a position to present various characterisations of trees.

Theorem 5.3 *Let T be a graph of order n . Then the following statements are equivalent:*

- (i) T is a tree;
- (ii) T is connected and has size $n - 1$;
- (iii) T has no cycles and size $n - 1$.

Proof If T is a tree of order n , then, by definition, T is connected and, by [Theorem 5.2](#), it follows that T has size $n - 1$. So (i) \implies (ii).

Suppose that T is connected and has size $n - 1$. Suppose, to the contrary, that T contains a cycle and let e be an edge of this cycle. Then, by [Theorem 2.6](#), e is not a bridge of T , and so $T - e$ is a connected graph of order n and size $n - 2$, which is a contradiction (see [Exercise 5.3](#)). Therefore, T has no cycles and so (ii) \implies (iii).

Suppose that T has no cycles and size $n - 1$. In order to prove that T is a tree, we must show that T is connected. Let T_1, \dots, T_k be the components of T ($k \geq 1$), where T_i has order n_i and size m_i for $i \in [k]$. Since each component of T is a connected graph with no cycles, each graph T_i is a tree. Thus, by [Theorem 5.2](#), $m_i = n_i - 1$ for all $i \in [k]$. Hence,

$$n - 1 = m = \sum_{i=1}^k m_i = \sum_{i=1}^k (n_i - 1) = \left(\sum_{i=1}^k n_i \right) - k = n - k,$$

and so $k = 1$, which means that T is connected and therefore a tree. We have therefore shown that (iii) \implies (i). ■

Each of the trees in [Figure 5.1](#) has at least two end-vertices. Using [Theorem 5.2](#), we show that this observation is true for trees in general.

Theorem 5.4 *Every nontrivial tree contains at least two end-vertices.*

Proof Suppose that T is a tree of order n and size m with $n \geq 2$. Let d_1, d_2, \dots, d_n denote the degrees of the vertices of T , ordered so that $d_1 \leq d_2 \leq \dots \leq d_n$. Since T is connected and $n \geq 2$, it follows that $d_1 \geq 1$. Suppose, however, that $d_2 \geq 2$. Then

$$\sum_{i=1}^n d_i = d_1 + \sum_{i=2}^n d_i \geq 1 + 2(n - 1) = 2n - 1. \quad (5.1)$$

But, by [Theorems 1.1](#) and [5.2](#),

$$\sum_{i=1}^n d_i = 2m = 2(n - 1) = 2n - 2,$$

which contradicts (5.1). Hence, $d_2 = 1$, and therefore $d_1 = 1$. The tree T therefore contains at least two end-vertices. ■

❖ The reader should now be able to attempt Exercises [5.1–5.11](#).

5.3 Constructing minimum spanning trees

In this section we return to the Minimum Connector Problem. First we shall need the concept of a spanning tree of a connected graph.

A **spanning tree** of a connected graph G is a tree that is a subgraph of G and contains all the vertices of G . A **spanning forest** of G is a forest that is a subgraph of G and contains all the vertices of G . A connected graph may actually have several nonisomorphic spanning trees. Figure 5.3 contains drawings of a graph $G_{5.4}$ and three of its spanning trees, $T_{5.5}$, $T_{5.6}$ and $T_{5.7}$.

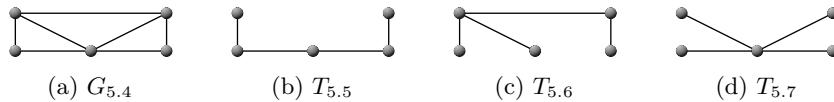


Figure 5.3: Nonisomorphic spanning trees of a graph.

Every connected graph G contains at least one spanning tree. We can find such a spanning tree as follows. If G has no cycles, then the graph G itself is a spanning tree. If there are cycles in G , then choose any cycle in G and remove one of its edges. By Theorem 2.6, we cannot disconnect G by removing one of its cycle edges, and so we still have a connected graph. We now repeat this procedure, removing cycle edges one at a time, until finally only bridges remain; this results in a spanning tree. It is therefore easy to find a spanning tree in a connected graph. However, finding a spanning tree with certain properties is often desirable from a practical point of view.

A spanning tree T of a connected graph G is **distance-preserving from a vertex** v in G if $d_T(u, v) = d_G(u, v)$ for every vertex u .

Theorem 5.5 *For every vertex v of an unweighted, connected graph G , there exists a spanning tree T of G that is distance-preserving from v .*

Proof Let k be the maximum distance from v to a vertex of G and let

$$D_i(v) = \{u \in V(G) \mid d(u, v) = i\}, \quad i \in [k].$$

Since G is connected, it follows that every vertex $u \neq v$ belongs to $D_i(v)$ for some $i \in [n]$. Furthermore, such a vertex u is adjacent to at least one vertex in $D_{i-1}(v)$ and possibly to vertices in $D_i(v)$ and $D_{i+1}(v)$ as well. Delete all but one edge that joins u to a vertex in $D_{i-1}(v)$. Also, remove every edge joining u to a vertex in $D_i(v)$. Repeat this process for each $u \neq v$ and let T denote the resulting graph.

From the manner in which T was constructed, it is clear that T is connected, since a $u-v$ path exists for each $u \neq v$. It is also clear that T is distance-preserving from v . It remains to be verified that T is a tree. We need only show that T is acyclic. If this is not the case, then there is a cycle C in T . Let w be a vertex of C whose distance from v is maximum, and let w_1 and w_2 be the vertices adjacent to w on C . Suppose $w \in D_\ell$. Then, $w_i \in D_\ell$ or $w_i \in D_{\ell-1}$ for $i = 1, 2$. If $w_1 \in D_\ell$ or $w_2 \in D_\ell$, then this contradicts the way in which T was constructed (no edge in T joins two vertices in the same set $D_i(v)$). Thus, both w_1 and w_2 belong to the set $D_{\ell-1}$. Once again, this contradicts the way in which T was constructed (exactly one edge in T joins a vertex in D_i to a vertex in D_{i-1}). We conclude, therefore, that T is acyclic and hence a tree. ■

A connected graph $G_{5.8}$ is shown in [Figure 5.4\(a\)](#) together with a spanning tree $T_{5.9}$ of $G_{5.8}$ in [Figure 5.4\(b\)](#) that is distance-preserving from the vertex v .

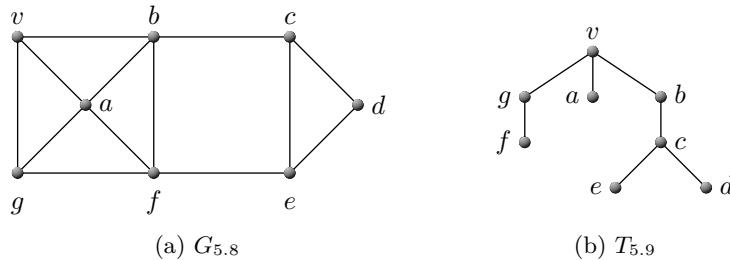


Figure 5.4: A connected graph $G_{5.8}$ (with vertex v) and a spanning tree $T_{5.9}$ of $G_{5.8}$ that is distance-preserving from v .

The **weight** $w(H)$ of a subgraph H of a weighted graph is the sum of the weights of the edges of H . A **minimum spanning tree** of a connected, weighted graph is a spanning tree of minimum weight in the graph. We may now restate the Minimum Connector Problem in graphical terms: Given a connected, weighted graph G , find a minimum spanning tree of G .

The Minimum Connector Problem is perhaps one of the best-known problems in combinatorial optimisation. Many practical problems involve connected, weighted graphs in which it is necessary to find a minimum spanning tree. In the introduction to this chapter, we highlighted one such application. Consider, as another example, a connected, weighted graph G in which each vertex represents a village, and an edge between two vertices represents a road between the corresponding villages. The length of such a road is indicated in the graph by assigning a weight to the corresponding edge. Suppose telephone lines must be installed along some existing roads that connect the villages with each other. We wish to erect these telephone lines in such a way that every pair of villages can communicate by telephone. Moreover, the total number of kilometres of telephone line is to be minimised so as to minimise cost. The problem with which we are faced is to determine along which roads telephone lines should be erected in order to produce the desired system. This amounts to finding a minimum spanning tree of the connected, weighted graph G .

A number of algorithmic solutions to the Minimum Connector Problem have been put forward. Perhaps the most famous of these algorithms is the one due to [Kruskal \[4\]](#), dating from 1956.

5.3.1 Kruskal's algorithm

The strategy of [Kruskal's algorithm](#) is simple. One begins by choosing an edge of minimum weight in the graph. One then continues by iteratively selecting from the remaining edges an edge of minimum weight that does not form a cycle with any of the edges already chosen. The end result is a minimum spanning tree of the original graph induced by the edges chosen. We present [Kruskal's algorithm](#) in pseudocode as [Algorithm 14](#).

[Kruskal's algorithm](#) is an instance of a class of algorithms known as **greedy algorithms**. This name arises from the fact that at each stage of the algorithm,

Algorithm 14: Kruskal's algorithm for computing a shortest spanning tree

Input : A connected, weighted graph G of order n .

Output : A set S of edges in G inducing a minimum spanning tree $G[S]$ of G .

```

1  $S \leftarrow \emptyset$ 
2 while  $|S| < n - 1$  do
3    $S \leftarrow S \cup \{e\}$  where  $e$  is an edge of  $G$  with minimum weight such that
     $e \notin S$  and  $G[S \cup \{e\}]$  is acyclic

```

the greediest choice available is made (*i.e.* an edge of smallest weight is chosen) with no concern for the consequences of these greedy choices during later stages of the algorithmic execution. Algorithms of this kind sometimes work very well, as we shall show in the case of [Kruskal's algorithm](#). In general, however, greedy algorithmic approaches often do not succeed in achieving global optimality in combinatorial optimisation problems, as we shall see in later chapters.

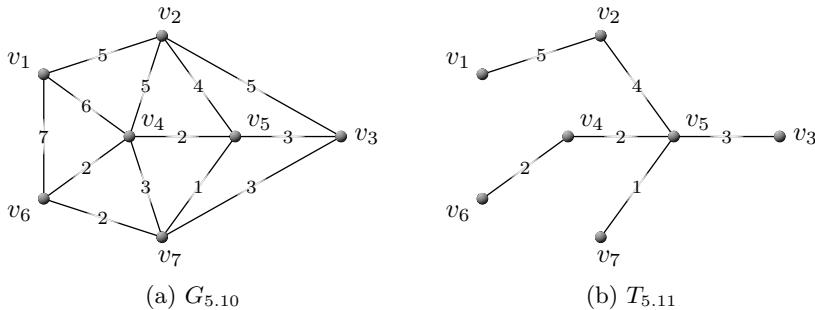


Figure 5.5: Illustrating Kruskal's algorithm (Algorithm 14).

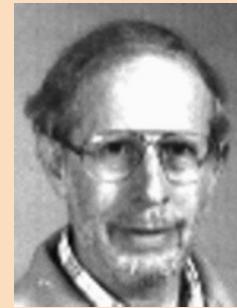
Suppose we wish to find a minimum spanning tree of the graph $G_{5,10}$ in [Figure 5.5\(a\)](#), using [Algorithm 14](#). We start with the assignment $S \leftarrow \emptyset$ in [Step 1](#) of the algorithm. The edge v_5v_7 is a (unique) edge of minimum weight in $G_{5,10}$, and cannot form a cycle when included in S , because S is currently empty. Hence the assignment $S \leftarrow S \cup \{v_5v_7\}$ is made. At this stage the weight of the graph $G_{5,10}[S]$ is 1, and because $|S| = 1 \neq 6$ in [Step 3](#), we return to [Step 2](#).

There are three edges of minimum weight (namely 2) in $E(G_{5,10}) \setminus S$; these edges are v_4v_5 , v_4v_6 and v_6v_7 . None of these edges forms a cycle with edges in S , and we arbitrarily make the assignment $S \leftarrow S \cup \{v_4v_6\}$. At this stage the weight of the graph $G_{5,10}[S]$ is 3, and because $|S| = 2 \neq 6$ in [Step 3](#), we return to [Step 2](#).

Now there are two edges of minimum weight (still 2) in $E(G_{5,10}) \setminus S$; these edges are v_4v_5 and v_6v_7 . Neither of these edges forms a cycle with edges in S , and we arbitrarily make the assignment $S \leftarrow S \cup \{v_4v_5\}$. At this stage the weight of the graph $G_{5,10}[S]$ is 5, and because $|S| = 3 \neq 6$ in [Step 3](#), we return to [Step 2](#).

Although there remains a single edge of weight 2 in $E(G_{5,10}) \setminus S$, this edge, v_6v_7 , forms the cycle $v_5v_4v_6v_7v_5$ of order 4 with edges already in S . There are, however, three edges of weight 3 in $E(G_{5,10}) \setminus S$, namely v_3v_5 , v_3v_7 and v_4v_5 , but one of these edges, v_4v_7 , forms the cycle $v_4v_5v_7v_4$ of order 3 with edges already in S .

The American mathematician, statistician and computer scientist **Joseph Bernard Kruskal** was born in New York and obtained his PhD in mathematics at Princeton University in 1954 while in the employment of the United States Office of Naval Research. During the period 1956–1959, he lectured at the University of Wisconsin at Madison and at the University of Wisconsin at Ann Arbor. In 1959, he took up an appointment at Bell Laboratories, in Murray Hill, New Jersey, where he continued to work for the rest of his career. He was, however, a visiting professor at Yale University during the period 1967–1968. He formally retired in 1993, but continued to undertake research at Bell Laboratories for several years. In statistics, his most important work was his seminal contribution to the formulation of multi-dimensional scaling, while in computer science his most notable contribution was his algorithm for finding minimum spanning trees ([Algorithm 14](#)). In combinatorics, he is known for a theorem called the *tree theorem*, which states that the set of finite trees over a well-quasi-ordered set of labels is itself well-quasi-ordered under homeomorphic embedding. Kruskal also applied his work in linguistics, in a well-known experimental lexicostatistical study of Indo-European languages. He was a fellow of the American Statistical Association, and served terms as president of both the Psychometric Society and the Classification Society of North America. He was also an active campaigner for civil rights.



Biographic note 16: Joseph Kruskal (1928–2010)

We therefore arbitrarily make the assignment $S \leftarrow S \cup \{v_3v_5\}$. At this stage the weight of the graph $G_{5.10}[S]$ is 8, and because $|S| = 4 \neq 6$ in [Step 3](#), we return to [Step 2](#) yet again.

Both of the remaining edges of weight 3 in $E(G_{5.10}) \setminus S$, namely v_3v_7 and v_4v_7 , now form 3-cycles with edges already in S . There is, however, a unique edge, v_2v_5 , of weight 4 in $E(G_{5.10}) \setminus S$ that does not form cycles with edges already in S . We therefore make the assignment $S \leftarrow S \cup \{v_2v_5\}$. At this stage the weight of the graph $G_{5.10}[S]$ is 12, and because $|S| = 5 \neq 6$ in [Step 3](#), we return to [Step 2](#) one last time.

There are three edges of weight 5 in $E(G_{5.10}) \setminus S$, namely v_1v_2 , v_2v_3 and v_2v_4 . Two of these edges (v_2v_3 and v_2v_4), however, form cycles with edges in S . We therefore make the assignment $S \leftarrow S \cup \{v_1v_2\}$. At this stage the weight of the graph $G_{5.10}[S]$ is 17, and because $|S| = 6$ in [Step 3](#), the algorithm terminates. The output of the algorithm is summarised in a step-by-step fashion in [Table 5.1](#).

The resulting minimum spanning tree, $T_{5.11}$, obtained by means of [Algorithm 14](#) is shown in [Figure 5.5\(b\)](#) and has a weight of 17. Of course, there may also be other minimum spanning trees of $G_{5.10}$.

We now establish the correct working of [Algorithm 14](#) in general.

Theorem 5.6 *Kruskal's algorithm* ([Algorithm 14](#)) produces a minimum spanning tree in a nontrivial connected, weighted graph G .

S	$ S $	$w(G_{5.10}[S])$
\emptyset	0	0
$\{v_5v_7\}$	1	1
$\{v_5v_7, v_4v_6\}$	2	3
$\{v_5v_7, v_4v_6, v_4v_5\}$	3	5
$\{v_5v_7, v_4v_6, v_4v_5, v_3v_5\}$	4	8
$\{v_5v_7, v_4v_6, v_4v_5, v_3v_5, v_2v_5\}$	5	12
$\{v_5v_7, v_4v_6, v_4v_5, v_3v_5, v_2v_5, v_1v_2\}$	6	17

Table 5.1: Output of [Algorithm 14](#) when applied to the graph $G_{5.10}$.

Proof Let G be a nontrivial connected, weighted graph of order n and let T be a subgraph of G produced by [Algorithm 14](#). By the acyclic construction of the subgraph in [Step 2](#) of the algorithm, as well as by [Theorem 5.3](#), T is a spanning tree of G . Suppose $E(T) = \{e_1, e_2, \dots, e_{n-1}\}$ where e_1, e_2, \dots, e_{n-1} is the order in which edges were included in S by the algorithm, and so $w(e_1) \leq w(e_2) \leq \dots \leq w(e_{n-1})$. The weight of T is

$$w(T) = \sum_{i=1}^{n-1} w(e_i).$$

Suppose, to the contrary, that T is not a minimum spanning tree of G . Among all the minimum spanning trees of G , let H be one with the largest number of edges in common with T . Since T and H are not equal, T must contain at least one edge that is not in H . Let e_i ($i \in [n-1]$) be the smallest indexed edge of T that is not in H and let $G_0 = H + e_i$. It is left as an exercise to prove that G_0 contains exactly one cycle C (see [Exercise 5.11](#)). Since T contains no cycles, there is a cycle edge e_0 in G_0 that is not in T . The graph $T_0 = G_0 - e_0$ is also a spanning tree of G and

$$w(T_0) = w(G_0) - w(e_0) = w(H) + w(e_i) - w(e_0),$$

so that

$$w(T_0) - w(H) = w(e_i) - w(e_0).$$

Since H is a minimum spanning tree of G , $w(H) \leq w(T_0)$ and it follows that $w(e_0) \leq w(e_i)$. From [Algorithm 14](#), e_i is an edge of minimum weight such that $\{e_1, e_2, \dots, e_{i-1}\} \cup \{e_i\}$ induces an acyclic graph. The graph induced by $\{e_1, e_2, \dots, e_{i-1}, e_0\}$ is, however, a subgraph of H and therefore also acyclic. Since T was constructed by [Algorithm 14](#), $w(e_i) \leq w(e_0)$. Consequently, $w(e_i) = w(e_0)$ and hence, $w(T_0) = w(H)$. This implies that T_0 is also a minimum spanning tree of G , but T_0 has more edges in common with T than does H , which contradicts our choice of H . We conclude that the spanning tree T produced by [Algorithm 14](#) indeed has minimum weight. ■

Let us now consider the time complexity of [Kruskal's algorithm](#) for a graph G of order n and size m . It is convenient to generate a list L of the edges of G , sorted in order of nondecreasing weight. Once this has been done, L may be used repeatedly to look up an edge $e \notin S$ of minimum weight ([Step 2](#) of the algorithm) in $\mathcal{O}(\log m)$ time (see [Task 2\(c\)](#) of [Project 3.1](#)), deleting an edge from L whenever

it forms a cycle with edges already in S (which may be tested in $\mathcal{O}(\log n)$ time), or when it is itself inserted into S . [Step 2](#) is repeated $n - 1$ times. Since $m \leq \binom{n}{2}$, it therefore follows that the worst-case time complexity of [Kruskal's algorithm](#) after formation of the list L is $\mathcal{O}(n \log m)$. Since $|L| = m$, the list L may be formed in $\mathcal{O}(m \log m)$ time by means of the merge sort procedure (see [Task 3 of Project 3.2](#)). This latter complexity dominates the former complexity, and hence the algorithm has a worst-case time complexity of $\mathcal{O}(m \log m) = \mathcal{O}(n^2 \log n)$.

5.3.2 Prim's algorithm

[Algorithm 14](#) involves computing whether $G[S \cup \{e\}]$ is acyclic. This may be cumbersome to implement on a computer and therefore we next consider an alternative algorithm for finding a minimum spanning tree of a connected, weighted graph G , known as [Prim's algorithm](#). The algorithm employs a clever technique to ensure that cycles are not formed during the iterative selection process. This technique involves successively growing the tree T (from one of order 1 initially until it is eventually a spanning tree) by iteratively selecting edges joining vertices that are already in T with vertices that are not yet in T . In this way it is clear that cycles are implicitly avoided without the need for explicitly having a check for potential cycles. We present [Prim's algorithm](#) in pseudocode as [Algorithm 15](#).

The American mathematician and computer scientist **Robert Clay Prim** was born in Sweetwater, Texas and obtained a bachelor's degree in electrical engineering from the University of Texas at Austin. During the second world war (1941–1944), he worked as an engineer at General Electric and then from 1944 to 1949 at the United States Naval Ordnance Laboratory, first as an engineer and later as a mathematician. He obtained his PhD in mathematics from Princeton University in 1949, where he was also a research associate during the period 1948–1949. Thereafter he worked at Bell Laboratories, where he served as Director of Mathematics Research during the period 1958–1961. There he developed his minimum spanning tree algorithm ([Algorithm 15](#)) in collaboration with [Joseph Kruskal](#) as a refinement to a basic stumbling block in the latter's minimum spanning tree algorithm. Prim was, in fact, not aware that his algorithm had earlier been invented by the mathematician Vojtěch Jarník (in 1930). It would later be rediscovered yet again by [Edsger Dijkstra](#) (in 1957). [Algorithm 15](#) is therefore sometimes also referred to as the *DJP algorithm* or as the *Jarník algorithm*.



Biographic note 17: Robert Prim (1921–present)

Again we consider the graph $G_{5.10}$ in [Figure 5.5\(a\)](#), but this time we use [Algorithm 15](#) to find a minimum spanning tree of $G_{5.10}$ with $x = v_1$ as the starting vertex in [Step 1](#) of the algorithm. Since v_1 is the only vertex in T , we examine the neighbours v_2 , v_4 and v_6 of v_1 . The edge v_1v_2 is clearly the (unique) edge of minimum weight joining the only vertex in T with a vertex not in T . We therefore

Algorithm 15: Prim's algorithm for computing a shortest spanning tree

Input : A connected, weighted graph G of order n .

Output : A minimum spanning tree T of G .

```

1  $T \leftarrow G[\{x\}]$  where  $x$  is an arbitrary vertex of  $G$ 
2 while  $|E(T)| < n - 1$  do
3    $T \leftarrow G[E(T) \cup \{e\}]$  where  $e$  is an edge of  $G$  with minimum weight that
      joins a vertex in  $T$  with a vertex that is not in  $T$ 
```

make the assignment $T \leftarrow G[E(T) \cup \{v_1v_2\}]$ in [Step 3](#) of the algorithm, so that the weight of T at this stage is $w(T) = 5$. Because $|E(T)| = 1 < 6$, we continue with the **while** loop in [Step 2](#).

The edges joining vertices in T (*i.e.* v_1 or v_2) with vertices not in T now are v_1v_4 , v_1v_6 , v_2v_3 , v_2v_4 and v_2v_5 . Of these edges, v_2v_5 is the one with minimum weight (namely 4) and hence we make the assignment $T \leftarrow G[E(T) \cup \{v_2v_5\}]$ in [Step 3](#) of the algorithm, so that the weight of T at this stage is $w(T) = 9$. Because $|E(T)| = 2 < 6$, we continue with the **while** loop in [Step 2](#).

Now the edges joining vertices in T (*i.e.* v_1 , v_2 or v_5) with vertices not in T are v_1v_4 , v_1v_6 , v_2v_3 , v_2v_4 , v_3v_5 , v_4v_5 and v_5v_7 . Of these edges, v_5v_7 has minimum weight (namely 1) and hence we make the assignment $T \leftarrow G[E(T) \cup \{v_5v_7\}]$ in [Step 3](#) of the algorithm, so that the weight of T at this stage is $w(T) = 10$. Because $|E(T)| = 3 < 6$, we continue with the **while** loop in [Step 2](#).

The edges joining vertices in T (*i.e.* v_1 , v_2 , v_5 or v_7) with vertices not in T are v_1v_4 , v_1v_6 , v_2v_3 , v_2v_4 , v_3v_5 , v_3v_7 , v_4v_5 , v_5v_7 and v_6v_7 . Of these edges, v_4v_5 is the one with minimum weight (namely 2) and hence we make the assignment $T \leftarrow G[E(T) \cup \{v_4v_5\}]$ in [Step 3](#), so that the weight of T at this stage is $w(T) = 12$. Because $|E(T)| = 4 < 6$, we continue with the **while** loop in [Step 2](#).

At this point, the edges joining vertices in T (*i.e.* v_1 , v_2 , v_4 , v_5 or v_7) with vertices not in T are v_1v_6 , v_2v_3 , v_3v_5 , v_3v_7 and v_4v_6 . Of these edges, v_6v_7 has minimum weight (namely 2) and hence we make the assignment $T \leftarrow G[E(T) \cup \{v_6v_7\}]$ in [Step 3](#) of the algorithm, so that the weight of T at this stage is $w(T) = 14$. Because $|E(T)| = 5 < 6$, we continue with the **while** loop in [Step 2](#) one last time.

Now the edges joining vertices in T (*i.e.* v_1 , v_2 , v_4 , v_5 , v_6 or v_7) with vertices not in T are v_2v_3 , v_3v_5 and v_3v_7 . Of these edges, v_3v_7 has minimum weight (namely 3) and hence we make the assignment $T \leftarrow G[E(T) \cup \{v_3v_7\}]$ in [Step 3](#) of the algorithm, so that the weight of T at this stage is $w(T) = 17$. Because $|E(T)| = 6$, the algorithm terminates. The output of the algorithm is summarised in the [Table 5.2](#).

A different minimum spanning tree, $T_{5.12}$, than previously found for $G_{5.10}$ is obtained using [Algorithm 15](#), as shown in [Figure 5.6\(a\)](#). [Figure 5.6\(b\)](#) shows yet another minimum spanning tree, $T_{5.13}$, of $G_{5.10}$ that may be obtained using [Algorithm 15](#). Both spanning trees in [Figure 5.6](#) have a minimum weight of 17, as does the one found in [Figure 5.5\(b\)](#) by [Kruskal's algorithm](#).

We now prove the correct working of [Algorithm 15](#).

Theorem 5.7 *Prim's algorithm (Algorithm 15) produces a minimum spanning tree in a nontrivial connected, weighted graph G .*

$E(T)$	$ E(T) $	$w(T)$
\emptyset	0	0
$\{v_1v_2\}$	1	5
$\{v_1v_2, v_2v_5\}$	2	9
$\{v_1v_2, v_2v_5, v_5v_7\}$	3	10
$\{v_1v_2, v_2v_5, v_4v_5, v_5v_7\}$	4	12
$\{v_1v_2, v_2v_5, v_4v_5, v_5v_7, v_6v_7\}$	5	14
$\{v_1v_2, v_2v_5, v_3v_7, v_4v_5, v_5v_7, v_6v_7\}$	6	17

Table 5.2: Output of Algorithm 15 when applied to the graph $G_{5.10}$.

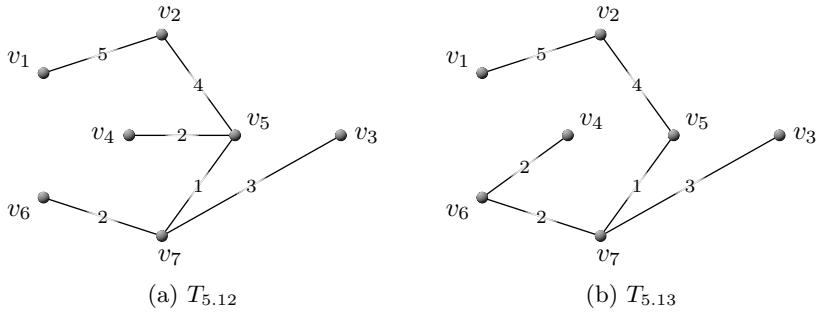


Figure 5.6: Two different minimum spanning trees of $G_{5.10}$.

Proof Let G be a nontrivial connected, weighted graph and let T be the subgraph of G produced by Algorithm 15. The subgraph T is certainly a spanning tree of G . Let $E(T) = \{e_1, e_2, \dots, e_{n-1}\}$, where e_j is the j -th edge in T produced by Algorithm 15, for all $j \in [n-1]$. Therefore, the weight of T is

$$w(T) = \sum_{i=1}^{n-1} w(e_i).$$

Suppose, to the contrary, that T is not a minimum spanning tree of G . Let H be a minimum spanning tree of G that has the maximum number of edges in common with T and let e_i be the first edge of T that is not in H . If $i = 1$, then let $U = \{u\}$; else let U be the vertex set of the subgraph induced by the edges $\{e_1, e_2, \dots, e_{i-1}\}$, i.e. $U = V(G[\{e_1, e_2, \dots, e_{i-1}\}])$.

The graph $G[E(H) \cup \{e_i\}]$ contains exactly one cycle, C say, and e_i joins a vertex of U with a vertex of $V(T) \setminus U$. C , however, contains another edge e_0 that joins a vertex in U with a vertex in $V(T) \setminus U$. Let $T' = G[(E(H) \cup \{e_i\}) \setminus \{e_0\}]$. Then T' is a spanning tree of G . Since T was constructed by Algorithm 15, $w_T(e_i) \leq w_C(e_0)$, and therefore $w(T') \leq w(H)$. This means that T' is also a minimum spanning tree of G . But T' has more edges in common with T than does H , which contradicts our choice of H . We conclude that T is indeed a minimum spanning tree of G . ■

Let us now consider the time complexity of Prim's algorithm for a graph G of order n and size m . It is again convenient to generate a list L of the edges of G , sorted in order of nondecreasing weight — a once-off task which may be achieved in $\mathcal{O}(m \log m)$ time at the start of execution of the algorithm. Since $m \leq \binom{n}{2}$, this time complexity may be rewritten as $\mathcal{O}(n^2 \log n)$.

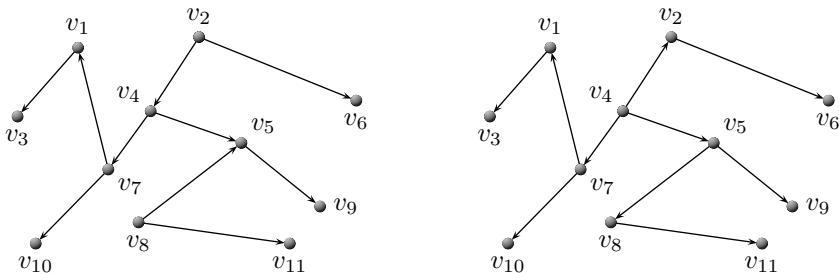
At each iteration of **Step 2** of the algorithm, those edges in L joining a vertex in T with a vertex not in T may be identified or marked in $\mathcal{O}(\log n)$ time. From the marked edges in the sorted list L , an appropriate edge e in **Step 2** of the algorithm may then be retrieved in constant time. The time complexity of the most time-consuming step in the algorithm, **Step 2**, is therefore $\mathcal{O}(\log n)$. Since this step is repeated $n - 1$ times, it follows that the worst-case time complexity of **Prim's algorithm** after formation of the list L is $\mathcal{O}(n \log n)$.

The time complexity of forming the list L in the first place, however, dominates the latter time complexity, and hence **Algorithm 15** has a worst-case time complexity of $\mathcal{O}(n^2 \log n)$. The efficiency of **Prim's algorithm** is therefore comparable to that of **Kruskal's algorithm**.

- ❖ The reader should now be able to attempt Exercises 5.12–5.15 as well as Projects 5.1–5.3.

5.4 Rooted trees

A **directed tree** is an asymmetric, directed graph for which the underlying (undirected) graph is a tree. A directed tree T containing a vertex r such that, for every vertex $v \neq r$, there exists a directed $r-v$ path in T , is called a **rooted tree**. Such a vertex r is called a **root** of the rooted tree T . **Figure 5.7(a)** contains a directed tree which is not rooted, since there are two vertices $v_2, v_8 \in V(T_{5.14})$ such that $\text{id}_{T_{5.14}}(v_2) = \text{id}_{T_{5.14}}(v_8) = 0$, while a rooted tree is shown in **Figure 5.7(b)**.



(a) A directed tree, $T_{5.14}$, that is not rooted (b) A rooted tree, $T_{5.15}$, with root v_4

Figure 5.7: Directed versus rooted trees.

Theorem 5.8 *A directed tree T is a rooted tree if and only if there exists a vertex $r \in V(T)$ with $\text{id}(r) = 0$, while $\text{id}(v) = 1$ for every other vertex $v \in V(T)$.*

Proof Suppose T is a rooted tree with root r . Thus, $\text{id}(r) = 0$. Let $v \neq r$ be a vertex of T . Then, v is adjacent from exactly one vertex of T ; hence $\text{id}(v) = 1$.

Conversely, suppose T is a directed tree that contains a vertex r with $\text{id}(r) = 0$, while $\text{id}(v) = 1$ for all vertices $v \neq r$ in $V(T)$. Let $u_1 \neq r$ be a vertex of T . Since $\text{id}(u_1) = 1$ there is exactly one vertex, u_2 (say), that is adjacent to u_1 . If $u_2 \neq r$ then there again exists a unique vertex, u_3 (say), that is adjacent to u_2 . These vertices u_1, u_2, u_3, \dots are the start of a sequence. Continue this sequence as far as possible. The sequence will consist of different (nonrepeated) vertices (for if it were possible that $u_i = u_j$ (for some $i < j$), while the vertices $u_i, u_{i+1}, \dots, u_{j-1}$

are all different, then the path $u_i, u_{i+1}, \dots, u_{j-1}, u_j$ would form a cycle, which is impossible, because T is a tree). Hence, the sequence u_1, u_2, \dots, u_k is finite. Since $\text{id}(r) = 0$ and $\text{id}(u) \neq 0$ for all $u \neq r$, it follows that $u_k = r$ and so $r = u_k, u_{k-1}, \dots, u_2, u_1$ forms an r - u_i path from r to any vertex u_i in T . Consequently T is a rooted tree. ■

The next result follows directly from [Theorem 5.8](#).

Corollary 5.9 *Every rooted tree has a unique root.*

When representing rooted trees graphically, the root is usually placed on its own at the top of the tree within a horizontal band, called level 0 of the tree. The vertices that are adjacent from the root are then placed directly below the root at level 1, and so forth. In general, any vertex at level i is adjacent from exactly one vertex at level $i - 1$. Therefore, a vertex x in a rooted tree T lies at level i if and only if there exists a directed r - x path of length i in T . The largest integer h for which there are vertices at level h , is called the **height** of the rooted tree. Since all arcs are directed downwards, it is not necessary to include the directions of the arcs explicitly. The rooted tree $T_{5.15}$ in [Figure 5.7\(b\)](#) is redrawn in this manner in [Figure 5.8\(a\)](#).

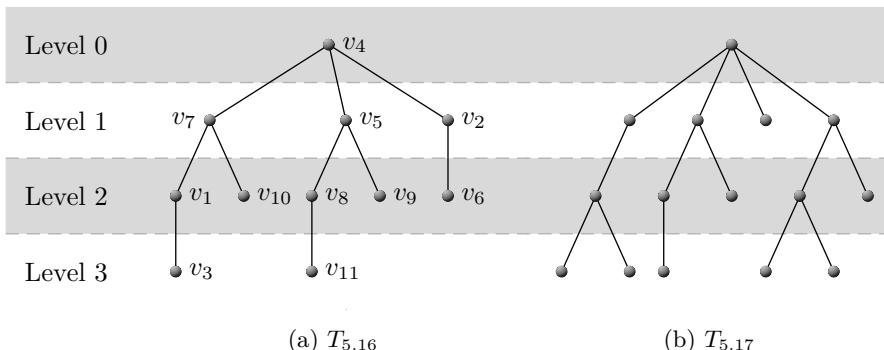


Figure 5.8: (a) A balanced and (b) an unbalanced rooted tree.

A **leaf** in a rooted tree T is a vertex $v \in V(T)$ for which $\text{od}(v) = 0$. A rooted tree T of height h is **balanced** if every leaf of T is at level h or $h - 1$. The trees $T_{5.15}$ and $T_{5.16}$ are therefore examples of balanced, rooted trees of height 3. The tree $T_{5.17}$ in [Figure 5.8\(b\)](#) is, however, not balanced, because it has a leaf on level 1 but has height 3.

If T is a rooted tree and a vertex $u \in V(T)$ is adjacent to a vertex $v \in V(T)$, then u is known as the **parent** of v and v as the **child** of u . A vertex $w \in V(T)$ is a **descendant** of v if there exists a directed v - w path in T (in which case the level of v is smaller than that of w) and v is called an **ancestor** of w . In a rooted tree, only the root has no parent (or ancestor) — every other vertex has exactly one parent. The vertices of a rooted tree T that have no children (or descendants) are exactly the leaves of T . Every vertex of T that is not a leaf is called an **internal vertex** of T , including the root. The vertices v_3, v_6, v_9, v_{10} and v_{11} are therefore the leaves of the rooted tree $T_{5.16}$, whilst the vertices v_1, v_2, v_4, v_5, v_7 and v_8 are its internal vertices.

A rooted tree T is called an **ℓ -ary tree** if every vertex of T has at most ℓ children. An ℓ -ary tree is often called a **binary tree** if $\ell = 2$, or a **ternary tree** if $\ell = 3$. A **complete ℓ -ary tree** is an ℓ -ary tree in which every vertex has either ℓ children or no children at all. The rooted tree $T_{5.18}$ in Figure 5.9(a) is an example of a 2-ary (or binary) tree that is not complete. The rooted tree $T_{5.19}$ in Figure 5.9(b), on the other hand, is a complete 2-ary tree.

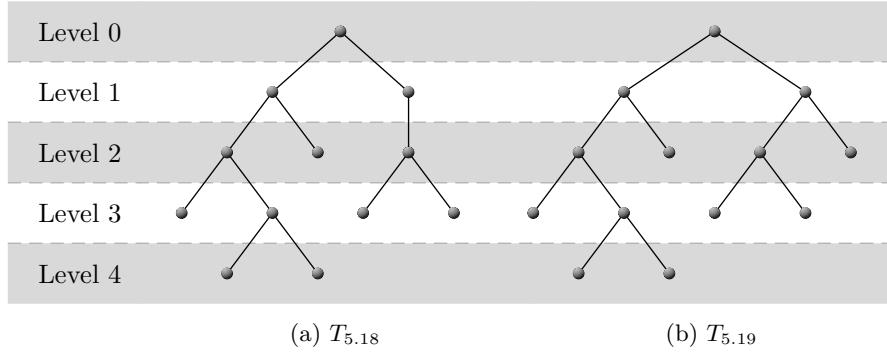


Figure 5.9: Two 2-ary trees: (a) a 2-ary tree that is not complete; (b) a complete 2-ary tree.

It is possible to enumerate the internal vertices of a complete ℓ -ary tree of given order if the number of leaves in the tree is known, and *vice versa*.

Theorem 5.10 *A complete ℓ -ary tree with i internal vertices has order $\ell i + 1$.*

Proof Let T be a complete ℓ -ary tree of order n with i internal vertices. Every internal vertex of T has exactly ℓ children and every child has a unique parent. Since the root is the only vertex that has no parent, it follows that $n = \ell i + 1$. ■

The next result follows directly from Theorem 5.10.

Corollary 5.11 *A complete ℓ -ary tree with i internal vertices has $(\ell - 1)i + 1$ leaves.*

Proof Let T be a complete ℓ -ary tree with i internal vertices. It follows from Theorem 5.10 that the order of T is $\ell i + 1$. Since T has i internal vertices, the remaining $\ell i + 1 - i = (\ell - 1)i + 1$ vertices are leaves. ■

We next establish a relationship between the height and order of an ℓ -ary tree.

Theorem 5.12 *Let T be an ℓ -ary tree of height h and order n . Then*

$$h + 1 \leq n \leq \frac{\ell^{h+1} - 1}{\ell - 1}.$$

Proof Let n_k denote the number of vertices on level k of T ($k \in \{0, \dots, h\}$). Then $\sum_{k=0}^h n_k = n$. There is at least one vertex on each level of T (*i.e.* $n_k \geq 1$ for

all $k \in \{0, \dots, h\}$). Furthermore, $n_k \leq \ell n_{k-1}$ for all $k \in [h]$, and hence it follows inductively that $n_k \leq \ell^k$ for all $k = 0, \dots, h$. Consequently,

$$\underbrace{\sum_{k=0}^h 1}_{h+1} \leq \underbrace{\sum_{k=0}^h n_k}_n \leq \underbrace{\sum_{k=0}^h \ell^k}_{\frac{\ell^{h+1}-1}{\ell-1}},$$

the latter series being a geometric series. ■

The lower bound on n in [Theorem 5.12](#) is achieved by paths, while the upper bound is achieved by complete ℓ -ary trees. The following useful result is an immediate consequence of [Theorem 5.12](#).

Corollary 5.13 *If T is a complete ℓ -ary tree of height h and order n , then*

$$h \geq \lceil \log_\ell(n(\ell - 1) + 1) \rceil - 1. \quad (5.2)$$

Furthermore, equality holds in [\(5.2\)](#) if T is balanced.

Proof It follows from [Theorem 5.12](#) that $n \leq (\ell^{h+1} - 1)/(\ell - 1)$, which may be rewritten as $n(\ell - 1) + 1 \leq \ell^{h+1}$. Therefore, $\log_\ell(n(\ell - 1) + 1) \leq h + 1$ by taking logarithms, base ℓ , on both sides of the inequality, yielding [\(5.2\)](#).

Now suppose that T is balanced. Then, $n_k \geq \ell n_{k-1}$ for all $k \in [h - 1]$, since T has no leaves on levels $0, \dots, h - 2$. Hence it follows inductively that $n_k \geq \ell^k$ for all $k \in \{0, \dots, h - 1\}$, so that $n > \sum_{k=0}^{h-1} n_k \geq \sum_{k=0}^{h-1} \ell^k = (\ell^h - 1)/(\ell - 1)$, yielding the inequality chain

$$\frac{\ell^h - 1}{\ell - 1} < n \leq \frac{\ell^{h+1} - 1}{\ell - 1} \quad (5.3)$$

in conjunction with [Theorem 5.12](#). But [\(5.3\)](#) may be rewritten as $\ell^h < n(\ell - 1) + 1 \leq \ell^{h+1}$, or equivalently as

$$h - 1 < \log_\ell(n(\ell - 1) + 1) - 1 \leq h \quad (5.4)$$

by taking logarithms, base ℓ , throughout the inequality chain. ■

❖ The reader should now be able to attempt Exercises [5.16–5.19](#) and Project [5.4](#).

5.5 Depth-first tree searches

In many applications it is necessary to be able to traverse the vertices of a labelled graph in a systematic and efficient manner. One such application is the problem of efficiently determining which edges of a graph are its bridges, and another is determining which vertices of a graph are its cut-vertices. In this section, we explore a method, called *depth-first tree search*, for traversing the vertices of a labelled graph systematically. This method is attributed to the 19th century French author Charles Pierre Trémaux (1859–1882), who invented it as a strategy for solving maze puzzles.

A unique depth-first search tree T may be associated with every connected, labelled graph G of order n . Suppose the vertex set of G is $\{v_1, \dots, v_n\}$. Such a tree T is a spanning tree of G which specifies a traversal order for the vertices of G by the assignment of labels $1, \dots, n$ to the vertices of G . A search label is assigned to vertex v_i , called the **depth-first index** of v_i , and is denoted by $\text{DFI}(v_i)$, for all $i \in [n]$.

The process of generating T proceeds as follows. First we assign the search label $\text{DFI}(v_1) = 1$ to the vertex v_1 of G , which forms the root of the depth-first search tree and the first *current vertex* in an iterated procedure for constructing a set of edges S that will eventually induce the depth-first search tree T . We initialise this set S as the empty set. During each iteration of the process we take as *current vertex* the last vertex w labelled with a DFI value. If there are as yet unlabelled neighbours of w in G , then we select amongst these unlabelled neighbours the smallest indexed vertex v , label it with the next available DFI value, insert the edge wv into the set S and take v as the new *current vertex*. If, on the other hand, all the neighbours of w have already been labelled, then we revisit the vertex labelled just before w (a process that is called **backtracking**) and take it as the new *current vertex*. This general step is repeated until every vertex of G has been labelled. The process is presented in pseudocode as [Algorithm 16](#).

Algorithm 16: A depth-first search tree construction algorithm

Input : A connected graph G with vertex set $\{v_1, \dots, v_n\}$.
Output : A set S of edges in G inducing a depth-first search tree $G[S]$ of G with v_1 as root.

```

1  $\text{DFI}(v) \leftarrow 0$  for each vertex  $v \in V(G)$ 
2  $S \leftarrow \emptyset$ ,  $i \leftarrow 1$ ,  $w \leftarrow v_1$ 
3  $\text{DFI}(w) \leftarrow i$ ,  $i \leftarrow i + 1$ 
4 if  $\text{DFI}(v) = 0$  for some  $v \in N(w)$  then
5   | let  $v$  be the smallest indexed vertex in  $N(w)$  for which  $\text{DFI}(v) = 0$ 
6 else
7   | go to Step 9
8  $S \leftarrow S \cup \{wv\}$ ,  $\text{parent}(v) \leftarrow w$ ,  $w \leftarrow v$ , go to Step 3
9 if  $w \neq v_1$  then  $w \leftarrow \text{parent}(w)$ , go to Step 4
10 else stop
```

Depth-first search trees are drawn as directed trees in the normal manner, with the root v_1 at the top and all arcs directed downwards. A **back edge** of G is an edge of G that does not appear in T , and it is customary to draw these edges as dashed arcs on T directed back towards the root v_1 .

We illustrate the working of the above [depth-first search tree construction algorithm](#) by an example, considering the graph $G_{5,20}$ in [Figure 5.10\(a\)](#). We start by initialising all depth-first index values as zero in [Step 1](#) of the algorithm. This is followed by initialising the set S as the empty set, choosing the first available depth-first index value i as 1, and selecting the vertex $w = v_1$ both as root of the depth-first search tree and as the first current vertex of the depth-first tree search process in [Step 2](#).

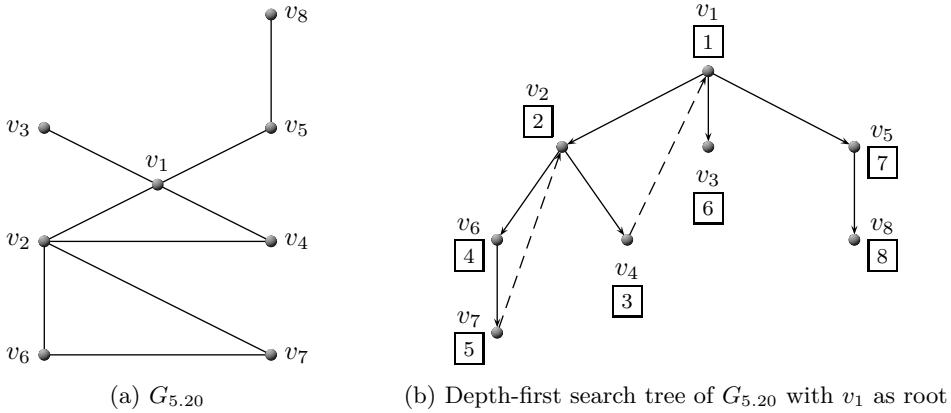


Figure 5.10: A graph $G_{5,20}$ and its depth-first search tree.

Hereafter the iterative process starts with the assignment $\text{DFI}(w) = 1$ of a depth-first traversal label to the current vertex $w (= v_1)$ and by incrementing the first available depth-first index value i to 2. Since $w = v_1$ has as zero-labelled neighbours the vertices v_2, v_3, v_4 and v_5 , the vertex $v = v_2$ is selected in Step 5 of the algorithm. The edge $wv = v_1v_2$ is inserted into the set S , the vertex $v = v_2$ is assigned $w = v_1$ as parent in the depth-first search tree, and v_2 is taken as the next current vertex w in Step 8.

Returning to Step 3, the depth-first traversal index $\text{DFI}(w) = 2$ is assigned to the current vertex $w (= v_2)$ and the first available depth-first index value i is incremented to 3. Since $w = v_2$ has as zero-labelled neighbours the vertices v_4, v_6 and v_7 , the vertex $v = v_4$ is selected in Step 5 of the algorithm. The edge $wv = v_2v_4$ is inserted into the set S so that $S = \{v_1v_2, v_2v_4\}$, the vertex v_4 is assigned $w = v_2$ as parent in the depth-first search tree, and the next current vertex w is taken as v_4 in Step 8.

Returning to Step 3, the depth-first traversal index $\text{DFI}(w) = 3$ is assigned to the current vertex $w (= v_4)$ and the first available depth-first index value i is incremented to 4. Now $w = v_4$ has no zero-labelled neighbours and so we backtrack to the parent of v_4 in the search tree, namely v_2 , which again becomes our current vertex w in Step 9. Since $w = v_2$ has as zero-labelled neighbours the vertices v_6 and v_7 , the vertex $v = v_6$ is selected in Step 5 of the algorithm. The edge $wv = v_2v_6$ is inserted into S so that $S = \{v_1v_2, v_2v_4, v_2v_6\}$, the vertex v_6 is assigned $w = v_2$ as parent in the depth-first search tree, and the next current vertex w is taken as v_6 in Step 8.

Returning to Step 3, the depth-first traversal index $\text{DFI}(w) = 4$ is assigned to the current vertex $w (= v_6)$ and the first available depth-first index value i is incremented to 5. Since $w = v_6$ has the zero-labelled neighbour v_7 , this vertex is taken as v in Step 5. The edge $wv = v_6v_7$ is inserted into S so that $S = \{v_1v_2, v_2v_4, v_2v_6, v_6v_7\}$, the vertex v_7 is assigned $w = v_6$ as parent in the depth-first search tree, and the next current vertex w is taken as v_7 in Step 8.

Returning to Step 3, the depth-first traversal index $\text{DFI}(w) = 5$ is assigned to the current vertex $w (= v_7)$ and the first available depth-first index value i is incremented to 6. Now $w = v_7$ has no zero-labelled neighbours and so we backtrack to the parent of v_7 in the search tree, namely v_6 , which becomes our current

vertex w in [Step 9](#). But $w = v_6$ also has no zero-labelled neighbours and so we backtrack again, this time to the parent of v_6 in the search tree, namely v_2 , which becomes our new current vertex w in [Step 9](#). But $w = v_2$ also has no zero-labelled neighbours and so we backtrack yet again, to v_1 , which becomes our new current vertex w in [Step 9](#). Since $w = v_1$ has as zero-labelled neighbours the vertices v_3 and v_5 , the vertex $v = v_3$ is selected in [Step 5](#). The edge $wv = v_1v_3$ is inserted into S so that $S = \{v_1v_2, v_2v_4, v_2v_6, v_6v_7, v_1v_3\}$, the vertex v_3 is assigned $w = v_1$ as parent in the depth-first search tree, and the next current vertex w is taken as v_3 in [Step 8](#).

Returning to [Step 3](#), the depth-first traversal index $\text{DFI}(w) = 6$ is assigned to the current vertex $w (= v_3)$ and the first available depth-first index value i is incremented to 7. Now $w = v_3$ has no zero-labelled neighbours and so we backtrack to v_1 , which again becomes our current vertex w in [Step 9](#). Since $w = v_1$ has the zero-labelled neighbour v_5 , this vertex is taken as v in [Step 5](#) of the algorithm. The edge $wv = v_1v_5$ is inserted into S so that $S = \{v_1v_2, v_2v_4, v_2v_6, v_6v_7, v_1v_3, v_1v_5\}$, the vertex v_5 is assigned $w = v_1$ as parent in the depth-first search tree, and the next current vertex w is taken as v_5 in [Step 8](#).

Returning to [Step 3](#), the depth-first traversal index $\text{DFI}(w) = 7$ is assigned to the current vertex $w (= v_5)$ and the first available depth-first index value i is incremented to 8. Since $w = v_5$ has the zero-labelled neighbour v_8 , this vertex is taken as v in [Step 5](#) of the algorithm. The edge $wv = v_5v_8$ is inserted into the set S so that $S = \{v_1v_2, v_2v_4, v_2v_6, v_6v_7, v_1v_3, v_1v_5, v_5v_8\}$, the vertex v_8 is assigned $w = v_5$ as parent in the depth-first search tree, and the next current vertex w is taken as v_8 in [Step 8](#).

Returning to [Step 3](#), the depth-first traversal index $\text{DFI}(w) = 8$ is assigned to the current vertex $w (= v_8)$ and the first available depth-first index value i is incremented to 9. However, now $w = v_8$ has no zero-labelled neighbours and so we backtrack to the parent of v_8 in the search tree, namely v_5 , which again becomes our current vertex w in [Step 9](#). Since $w = v_5$ also has a zero-labelled neighbours, we backtrack yet again, this time to the parent of v_5 in the search tree, namely v_1 , which has no zero-labelled neighbours and so the algorithm terminates.

The progress of the algorithm, when applied to the graph $G_{5,20}$ presented in [Figure 5.10\(a\)](#), is summarised in [Table 5.3](#), while the depth-first search tree associated with $G_{5,20}$ is shown in [Figure 5.10\(b\)](#).

Proof of the following result is left as an exercise.

Theorem 5.14 *Let G be a connected graph of order n and size m . Then [Algorithm 16](#) produces a depth-first search tree of G in $\mathcal{O}(n + m)$ time.*

As mentioned at the start of the section, one of the many applications of depth-first search trees is the problem of computing the bridges and cut-vertices of a graph efficiently. This application is explored in one of the projects at the end of the chapter.

During the late 1950s, Edward Moore invented another kind of systematic manner in which the vertices of labelled graphs may be traversed, called *breadth-first search*. In this alternative graph traversal procedure, all the neighbours of a vertex are first traversed before proceeding to any of their neighbours. This process is continued until all vertices have been visited. The notion of a breadth-first search is also considered in more detail in one of the projects at the end of the chapter.

i	w	DFI	v	S	Parent
1	v_1	$\text{DFI}(v_1) = 1$	—	\emptyset	—
2	v_1	$\text{DFI}(v_2) = 2$	v_2	$\{v_1 v_2\}$	$\text{parent}(v_2) = v_1$
3	v_2	$\text{DFI}(v_4) = 3$	v_4	$\{v_1 v_2, v_2 v_4\}$	$\text{parent}(v_4) = v_2$
4	v_4	—	—	—	—
	v_2	$\text{DFI}(v_6) = 4$	v_6	$\{v_1 v_2, v_2 v_4, v_2 v_6\}$	$\text{parent}(v_6) = v_2$
5	v_6	$\text{DFI}(v_7) = 5$	v_7	$\{v_1 v_2, v_2 v_4, v_2 v_6, v_6 v_7\}$	$\text{parent}(v_7) = v_6$
6	v_7	—	—	—	—
	v_6	—	—	—	—
	v_2	—	—	—	—
	v_1	$\text{DFI}(v_3) = 6$	v_3	$\{v_1 v_2, v_2 v_4, v_2 v_6, v_6 v_7, v_1 v_3\}$	$\text{parent}(v_3) = v_1$
7	v_3	—	—	—	—
	v_1	$\text{DFI}(v_5) = 7$	v_5	$\{v_1 v_2, v_2 v_4, v_2 v_6, v_6 v_7, v_1 v_3, v_1 v_5\}$	$\text{parent}(v_5) = v_1$
8	v_5	$\text{DFI}(v_8) = 8$	v_8	$\{v_1 v_2, v_2 v_4, v_2 v_6, v_6 v_7, v_1 v_3, v_1 v_5, v_5 v_8\}$	$\text{parent}(v_8) = v_5$
9	v_8	—	—	—	—
	v_5	—	—	—	—
	v_1	—	—	—	... stop

Table 5.3: Results obtained when applying the [depth-first search tree construction algorithm](#) to the graph $G_{5,20}$ in [Figure 5.10\(a\)](#). The resulting depth-first search tree $G_{5,20}[S]$ is shown in [Figure 5.10\(b\)](#).

- ❖ The reader should now be able to attempt Exercises 5.20–5.21 as well as Projects 5.5–5.6.

Exercises

5.1 Find graphical representations for the following:

- (a) All pairwise nonisomorphic trees of order 5 (Hint: There are 3 such trees);
- (b) All pairwise nonisomorphic trees of order 6 (Hint: There are 6 such trees);
- (c) All pairwise nonisomorphic forests of order 6 (Hint: There are 19 such forests);
- (d) All pairwise nonisomorphic trees of order 7 (Hint: There are 11 such trees).

5.2 Let G be a graph of order n and size $n - 1$. Demonstrate that G need not be a tree.

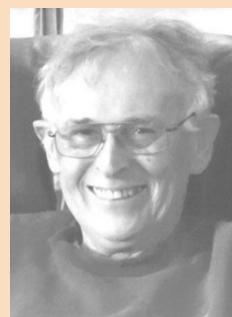
5.3 Prove that if G is a connected graph of order n and size m , then $m \geq n - 1$. (Hint: Use induction on the order n of a connected graph.)

5.4 Let T be a forest of order n , size m and comprising k components. Show that $m = n - k$.

5.5 Construct a forest with 12 vertices and 9 edges.

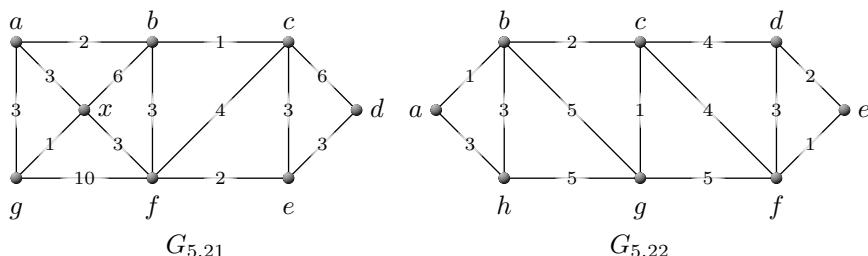
5.6 Show that if T is a tree and e is an edge of T , then $T - e$ has exactly two components.

The American mathematician and computer scientist **Edward Forrest Moore** was born in Baltimore and obtained his PhD in mathematics from Brown University in Providence, Rhode Island in 1950. During the period 1950–1952 he worked at the University of Illinois at Urbana-Champaign, and was a visiting lecturer at the Massachusetts Institute of Technology and at Harvard University simultaneously in 1952 and 1953. He then worked at Bell Laboratories for approximately ten years, after which he was appointed as professor at the University of Wisconsin at Madison in 1966 where he remained until his retirement in 1985. He was the inventor of what is today known as the Moore finite state machine and was an early pioneer of the notion of artificial life. With Claude Shannon, he did seminal work in computability theory and also spent a number of years unsuccessfully trying to prove the four colour theorem. He designed the notion of a breadth-first search tree in order to solve puzzles involving mazes.



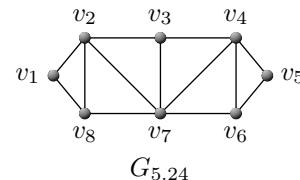
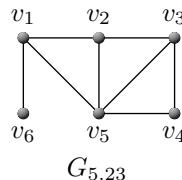
Biographic note 18: Edward Moore (1925–2003)

- 5.7 Show that the sequence d_1, d_2, \dots, d_n of positive integers is the degree sequence of a tree if and only if the graph is connected and $\sum_{i=1}^n d_i = 2(n-1)$.
- 5.8 Let G be a graph of order n and size m with $m \geq n \geq 3$. Show that G contains at least one cycle.
- 5.9 Suppose T is a tree of order n which contains only vertices of degree 1 or 3. Prove that T contains $(n-2)/2$ vertices of degree 3.
- 5.10 Let T be a tree of order 21 and suppose all the vertices of T have degrees in the set $\{1, 3, 5, 6\}$. If T has fifteen leaves and only one vertex of degree 6, how many vertices of T have degree 5?
- 5.11 Prove that if an edge is inserted between two nonadjacent vertices of a tree, then the resulting graph contains exactly one cycle.
- 5.12 Apply [Kruskal's algorithm](#) ([Algorithm 14](#)) to find minimum spanning trees for the weighted graphs (a) $G_{5.21}$ and (b) $G_{5.22}$ below.



- 5.13 Redo [Exercise 5.12](#), but instead find minimum spanning trees using [Prim's algorithm](#) ([Algorithm 15](#)).
- 5.14 Let G be a connected, weighted graph whose edges have distinct weights. Show that G has a unique minimum spanning tree.

- 5.15 Suppose all the weights in a connected, weighted graph differ.
- Will [Algorithm 14](#) necessarily include the edge of minimum weight as part of the minimum spanning tree produced? Motivate.
 - Will [Algorithm 15](#) necessarily include the edge of minimum weight as part of the minimum spanning tree produced? Motivate.
- 5.16 What is the largest number of great-grandchildren that any vertex of an ℓ -ary tree could possibly have? (A great-grandchild of a vertex v is a child of a child of a child of v .)
- 5.17 Prove that an ℓ -ary tree with i leaves has height $h \geq \lceil \log_\ell i \rceil$.
- 5.18 Draw a balanced 3-ary tree of height 3 which is not complete.
- 5.19 Prove that every tree is a bipartite graph.
- 5.20 Prove [Theorem 5.14](#).
- 5.21 Use [Algorithm 16](#) to construct depth-first trees for the graphs (a) $G_{5.23}$ and (b) $G_{5.24}$ below.



Computer exercises

Recall that a graph should satisfy two requirements in order to be a tree: It should be acyclic and connected. In Chapters 1 and 2 it was mentioned that the [MATHEMATICA](#) commands `AcyclicGraphQ[G]` and `ConnectedGraphQ[G]` may be used to test whether a graph G is acyclic and connected, respectively. These commands may be combined into the single [MATHEMATICA](#) command `TreeGraphQ[G]` which yields the boolean value `True` if the graph G is a tree (*i.e.* is both acyclic and connected), or the boolean value `False` otherwise. `BipartiteGraphQ[G]` yields the boolean value `True` if the graph G is bipartite, or the boolean value `False` otherwise. Of course, all trees are bipartite (see [Exercise 5.19](#)). Note also that stars and paths are acyclic and connected graphs, and hence also trees. A star graph of order n may be generated by the command `StarGraph[n]`. Therefore, the [MATHEMATICA](#) commands

```
In[1]:= T = StarGraph[9]
In[2]:= AcyclicGraphQ[T]
In[3]:= ConnectedGraphQ[T]
In[4]:= TreeGraphQ[T]
In[5]:= BipartiteGraphQ[T]
```

produce the output:

```
Out[1]:= 
Out[2]:= True
Out[3]:= True
Out[4]:= True
Out[5]:= True
```

Recall, from [Chapter 4](#), that the commands

```
In[6]:= m = {{Infinity, 5, Infinity, 6, Infinity, 7, Infinity}, {5, Infinity, 5, 5, 4, Infinity, Infinity}, {Infinity, 5, Infinity, Infinity, 3, Infinity, 3}, {6, 5, Infinity, Infinity, 2, 2, 3}, {Infinity, 4, 3, 2, Infinity, Infinity, 1}, {7, Infinity, Infinity, 2, Infinity, Infinity, 2}, {Infinity, Infinity, 3, 3, 1, 2, Infinity}};
In[7]:= G = WeightedAdjacencyGraph[m, VertexLabels -> "Name", EdgeLabels -> "EdgeWeight"]
```

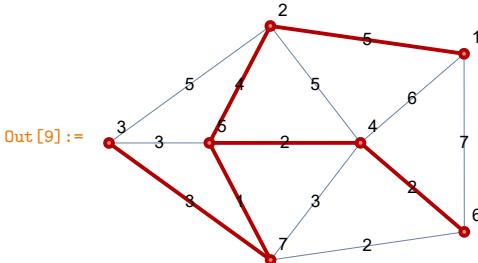
may be used to produce the weighted, connected graph $G_{5,10}$ in [Figure 5.5\(a\)](#), as shown below:

```
Out[7]:= 
```

[Kruskal's algorithm](#) ([Algorithm 14](#)) for finding a minimum spanning tree T of a graph G may be implemented in [MATHEMATICA](#) by executing the command `FindSpanningTree[G]` and its output may be visualised by means of the command `HighlightGraph[G, T]`. The command `PropertyValue` may further be used to tally the weights of the edges in T in order to find the minimum weight of a spanning tree of G . In particular, the command `PropertyValue[{obj, item}, name]` returns the value of property `name` of item `item` in object `obj`. Taking `obj = G`, `item = i` and `name = EdgeWeight`, for example, returns the weight of the i -th edge of G . Moreover, The command `Map[func, expr]` applies the function `func` to each element in `expr`. Therefore, the commands

```
In[8]:= T = FindSpanningTree[G];
In[9]:= HighlightGraph[G, T, GraphHighlightStyle -> "Thick"]
In[10]:= W = Map[PropertyValue[{G, #}, EdgeWeight] &, EdgeList[T]]
In[11]:= Total[W]
```

produce the output:

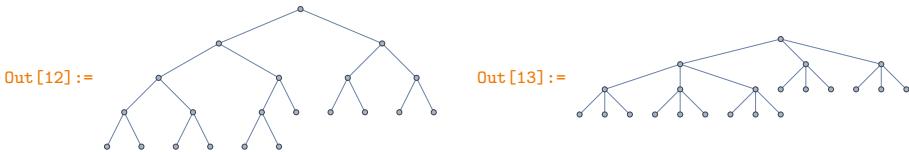


```
Out[9]:= 
Out[10]:= {5, 4, 3, 2, 2, 1}
Out[11]:= 17
```

Finally, the **MATHEMATICA** command **KaryTree**[n, ℓ] may be used to generate a complete ℓ -ary tree of order n . For example, the commands

```
In[12]:= KaryTree[21, 2]
In[13]:= KaryTree[22, 3]
```

produce the output:



Projects

This section contains six projects. Whereas the first three and last two projects have to be carried out with the help of a computer, the fourth project is small enough to be carried out by hand. More specifically, the first project requires computer implementations of [Kruskal's algorithm](#) and [Prim's algorithm](#) for computing minimum spanning trees in connected, weighted graphs. The next two projects both involve optimal network design, the former for a network of water pipes in the *Tshwane* suburb of *Mountain View* (using [Kruskal's algorithm](#)) and the latter for a network of optical fibre communication cables in Spain and Portugal (using [Prim's algorithm](#)). In the fourth project, the reader is guided towards the design of fair knock-out sports tournaments using rooted binary trees. Finally, the last two projects are concerned with search trees. In the penultimate project, the reader is guided in the design of a depth-first search tree approach toward efficiently computing the bridges and cut-vertices of an unweighted graph, whereas in the final project, the implementation of a breadth-first search tree for an unweighted graph is required.

Project 5.1: Minimum spanning trees

Two algorithms for finding minimum spanning trees in a connected, weighted graph were described in [Section 5.3](#), namely [Kruskal's algorithm](#) and [Prim's algorithm](#).

Tasks

1. Write your own **MATHEMATICA** implementation of **Kruskal's algorithm** ([Algorithm 14](#)). The input should be the weight matrix of a connected, weighted input graph G . The output should be the weight matrix of a minimum spanning tree of G produced by **Kruskal's algorithm**. Validate your implementation by applying it to the graph $G_{5.10}$ in [Figure 5.5\(a\)](#).
2. Write your own **MATHEMATICA** implementation of **Prim's algorithm** ([Algorithm 15](#)). The input should be the weight matrix of a connected, weighted input graph G , as well as a vertex $x \in V(G)$ to be used as starting point for the construction of a minimum spanning tree. The output should be the weight matrix of a minimum spanning tree of G produced by **Prim's algorithm**. Validate your implementation by applying it to the graph $G_{5.10}$ in [Figure 5.5\(a\)](#).

Project 5.2: Design of a water pipe system

Recall, from Chapters 1 and 4, that the *Tshwane* suburb of *Mountain View* ([Figure 1.24](#)), may be modelled by the connected, weighted graph shown in [Figure 5.11](#), where the edge weights denote the distances along street sections (measured in metres).

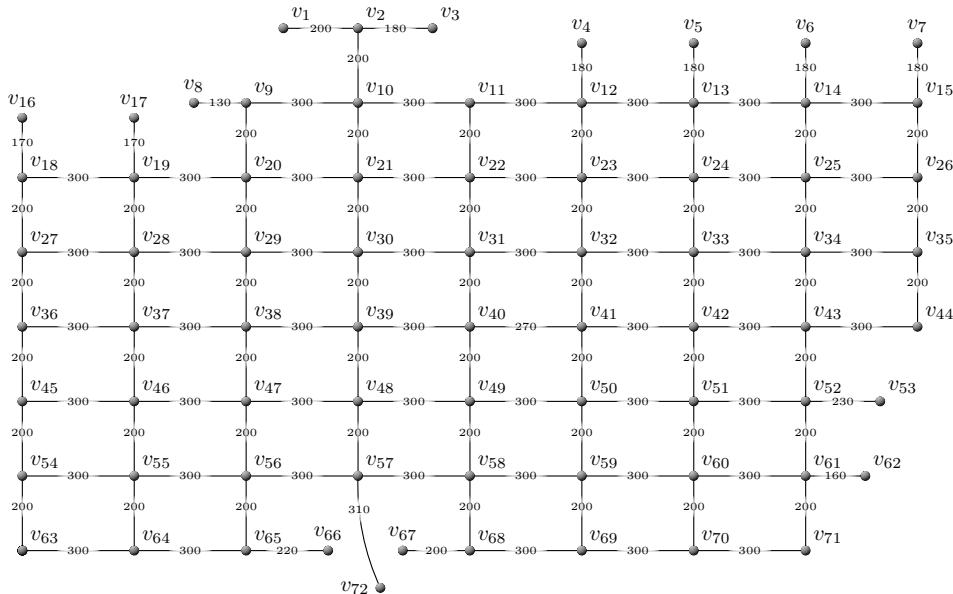


Figure 5.11: Graph model of the street layout of *Mountain View*. The edge weights represent street section lengths, measured in metres.

The suburb of *Mountain View* has a water substation at the dead-end of KAREL TRICHARDT street, modelled by the vertex v_{53} in the graph in [Figure 5.11](#). The aging, underground, concrete pipe system supplying houses in *Mountain View* with fresh water from the water substation has burst, producing leaks so often

during recent years that the municipality has decided to replace the entire pipe system with a brand new plastic pipe system, rather than to continue repairing the concrete pipe system on an almost continual basis. As a graph theorist, you have been consulted to help with the design of the new, underground plastic pipe system.

In an effort to minimise the cost of installing the new underground pipe system, the total length of plastic water pipes required to provide the suburb with water should be minimised. The municipality only has to provide each street corner and street dead-end modelled by a vertex in the graph in [Figure 5.11](#) with water from the substation — a separate, steel pipe system relays water from the nearest street corner or dead-end to every house — and this steel pipe system will not be replaced, as it is still functioning well. All plastic pipes have to be installed 1.5 metres under ground, below the streets, so that the municipality will not have to dig up private gardens during future repairs of the plastic pipe system.

Tasks

1. Use [Kruskal's algorithm](#) to find a most economical water pipe system layout providing all the street corners and street dead-ends of *Mountain View* with water, as supplied from the water substation.
2. Draw your water pipe system layout on the map in [Figure 1.24](#). What is the minimum total length of water pipes required for your layout?

Project 5.3: Designing a telecommunications network

According to recent studies by the Department of Telecommunications of the Spanish Government, the cities shown on the map in [Figure 5.12](#) will form the main communication centres in Spain and Portugal by the year 2050.

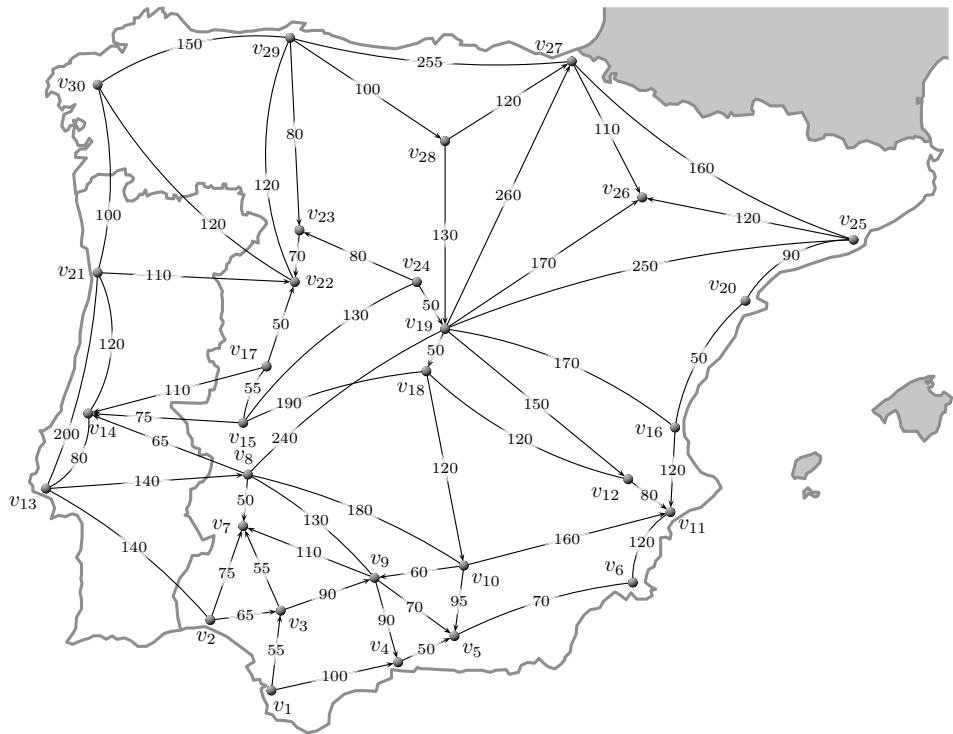
Optical fibre cables have to be installed so that all cities will be able to communicate with each other using the new, fast communication technologies of the digital age. Due to the extremely high cost of optical fibre, however, the minimum total optical cable length should be used in order to minimise installation costs. Possible locations of cable links are shown in [Figure 5.12](#). The weights associated with these links are the cable lengths that would be required for optical fibre cable installation in each case (measured in kilometres).

Tasks

1. Use [Prim's algorithm](#) to find a most economical optical fibre cable placement so as to connect all the cities in [Figure 5.12](#).
2. Draw your cable network on the map in [Figure 5.12](#). What is the minimum optical fibre cable length required?

Project 5.4: Scheduling a knock-out tennis tournament

The process of scheduling the progressive stages of a singles knock-out tennis tournament, such as the annual men's Wimbledon Championship, may be performed graphically by constructing a complete 2-ary tree (because each match involves two players), as shown in [Figure 5.13\(a\)](#). The root of the tree in [Figure 5.13\(a\)](#) represents the championship winner, the two vertices on level one of the tree represent the two players competing in the final match, the four vertices on level two



City		City		City	
v_1	Cádiz	v_{11}	Alicante	v_{21}	Porto
v_2	Huelva	v_{12}	Almansa	v_{22}	Salamanca
v_3	Sevilla	v_{13}	Lisbon	v_{23}	Zamora
v_4	Málaga	v_{14}	Tomar	v_{24}	Segovia
v_5	Granada	v_{15}	Cáceres	v_{25}	Barcelona
v_6	Cartagena	v_{16}	Valencia	v_{26}	Zaragoza
v_7	Zafra	v_{17}	Plasencia	v_{27}	San Sebastián
v_8	Mérida	v_{18}	Toledo	v_{28}	Burgos
v_9	Córdoba	v_{19}	Madrid	v_{29}	Oviedo
v_{10}	Úbeda	v_{20}	Tortosa	v_{30}	Santiago de Compostela

Figure 5.12: Map of the main communication centres in Spain and Portugal. The edge weights represent the lengths, measured in kilometres, of potential optical fibre links between pairs of cities.

of the tree represent the four players competing in the semi-finals, *etc.* Finally, the leaves of the tree represent all the participants of the knock-out tournament.

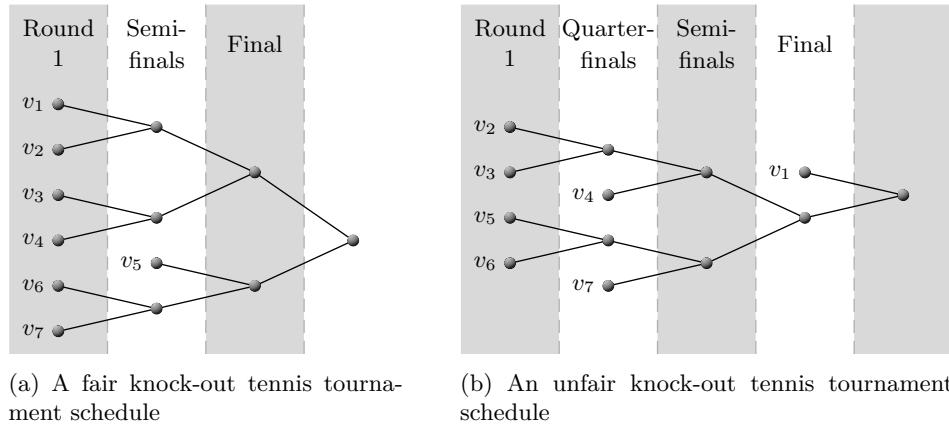


Figure 5.13: Graphical representations of a knock-out tennis tournament schedule for seven players v_1, \dots, v_7 .

To render the tournament as fair as possible, the tree must be balanced, because then the number of matches that have to be played by any participant so as to win the championship differs by at most one. For example, the schedule represented by the tree in Figure 5.13(a) is fair, because any player must play at least two and at most three matches in order to win the tournament. The schedule represented by the tree in Figure 5.13(b), on the other hand, is not fair, because player v_1 need only win one match to win the tournament, whilst players v_2, v_3, v_5 and v_6 each need to win four matches in order to win the tournament.

Tasks

- Find a graphical representation of a fair knock-out tennis tournament schedule for the twenty five players v_1, \dots, v_{25} . Indicate the final, the semi-finals, the quarter finals, *etc.* of the tournament as levels in the graphical representation.
- Repeat Task 1, but assume that fourteen of the twenty five players, v_1, \dots, v_{14} , are seeded. Ensure (a) that no seeded player competes against an unseeded player during the unseeded player's first match of the tournament, and (b) that no seeded player can win the tournament by playing fewer matches than any unseeded player has to play in order to win the tournament.

Project 5.5: Computing bridges and cut-vertices

In this project we explore how depth-first search trees may be used to test efficiently whether or not an edge of a connected graph G is a bridge of G and whether or not a given vertex of G is a cut-vertex of G . To this effect, define the **lowpoint** $\ell(v)$ of a vertex v of G as the smallest value $\text{DFI}(u)$ of a vertex u of the depth-first search tree (of G) that can be reached from v by a (directed) $v-u$ path consisting of (directed)

edges of T followed by at most one back edge. Because the $v-u$ path may be trivial, it follows that $\ell(v) \leq \text{DFI}(v)$ for all $v \in V(G)$. Strict inequality only holds if there is a back edge from v to a descendant or an ancestor of v .

The lowpoints of the vertices of the graph $G_{5,20}$ in Figure 5.10(a) are shown together with their DFI values in Table 5.4.

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$\text{DFI}(v_i)$	1	2	6	5	7	3	4	8
$\ell(v_i)$	1	2	6	1	7	2	2	8

Table 5.4: Depth-first indices and lowpoints of the vertices of the graph $G_{5,20}$ in Figure 5.10.

Tasks

- Let T be the depth-first search tree of a connected graph G . Prove that an edge uv of G with $\text{DFI}(u) < \text{DFI}(v)$ is a bridge of G if and only if uv is an edge of T and $\ell(v) > \text{DFI}(u)$.
 - Design an $\mathcal{O}(m)$ algorithm, based on the above characterisation, which takes an edge e of a connected graph G of size m as input, and yields **true** as output if e is a bridge of G , or **false** otherwise.
- Let T be the depth-first search tree of a connected graph G , let r be the root of T and let u be a vertex of G which is not the root of T . Prove that:
 - r is a cut-vertex of G if and only if it has at least two children in T , and
 - u is a cut-vertex of G if and only if it has a child in T for which $\ell(v) \geq \text{DFI}(u)$.
 - Design an $\mathcal{O}(m)$ algorithm, based on the above characterisation, which takes a vertex v of a connected graph G of size m as input, and yields **true** as output if v is a cut-vertex of G , or **false** otherwise.

Project 5.6: Breadth-first tree search

The mathematician and computer scientist Edward Moore invented an alternative systematic graph traversal protocol to the depth-first tree search protocol that we encountered in Section 5.5. This alternative is called *breadth-first tree search* and works as follows for a labelled graph G of order n with vertex set $\{v_1, \dots, v_n\}$:

- Assign a breadth-first traversal index **BFI** of zero to each vertex.
- Place the vertex v_1 at the head of a queue Q .
- Select the vertex x at the head of Q , and do the following for the neighbours of this vertex with **BFI**-values equal to zero:
 - Assign **BFI**-values to these neighbours in the order in which they have been labelled in G ,

- (b) Assign x as the parent of each of these neighbours in the breadth-first search tree,
 - (c) Insert these neighbours at the tail of Q in the order in which they have been labelled in G .
4. Delete the vertex at the head of Q .
5. If Q is nonempty, then return to [Step 3](#).

Tasks

1. Design a breadth-first tree search algorithm in pseudocode form.
2. Apply your algorithm to the graph $G_{5.20}$ in [Figure 5.10\(a\)](#), producing both a table of algorithmic progress (such as [Table 5.3](#)) and the resulting breadth-first search tree.

Further reading

- [1] G Chartrand and L Lesniak, 1996. *Graphs and Digraphs*, Third edition, Chapman & Hall/CRC, London.
- [2] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY).
- [3] RL Graham and P Hell, 1985. *On the history of the minimum spanning tree problem*, IEEE Annals of the History of Computing, **7(1)**, pp. 43–57.
- [4] JB Kruskal, 1956. *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical Society, **7(1)**, pp. 48–50.
- [5] DB West, 1996. *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River (NJ).



Location problems

Contents

6.1	Introduction	145
6.2	The centre of a graph	148
6.3	The median of a graph	152
6.4	General centres and medians	153
6.5	Absolute centres and medians	155
6.6	General absolute centres and medians	159
	Exercises	166
	Computer exercises	168
	Projects	168
	Further reading	171

6.1 Introduction

Suppose that a weighted, directed, connected graph is used to model a portion of the street layout of a city, where arcs and edges of the graph represent (possibly one-way) streets, where vertices of the graph represent street intersections and dead-ends, and where the edge weights represent distances. Consider the problem of finding a suitable intersection or dead-end for building an emergency hospital facility in this portion of the city. If the facility is to receive patients from any intersection or dead-end in the city portion, then the hospital should obviously be placed at a location for which the maximum distance that any potential patient would have to travel to the facility, is minimised. Questions that arise from this application include whether such a location is unique and how one may go about finding such an optimal location?

Another location problem is to find an optimal location for a daily newspaper distribution depot in the same city portion. Suppose that newspapers are to be distributed by a single bicycle from the distribution depot to all street intersections, where they are to be sold, but that the bicycle's capacity is such that it can only carry newspapers earmarked for a single street intersection at any one time. This means that after each delivery of newspapers to an intersection, the distributor has to return to the depot before a next delivery can be made. Where should the distribution depot be located so as to minimise the total daily distance travelled by the newspaper distributor? This location problem is fundamentally different from

the one of finding an optimal location for the emergency hospital facility, because in the former application one would like to minimise the *maximum potential distance* that would have to be travelled from an accident intersection to the facility, while in the latter application one would like to minimise the sum total of distances travelled between the facility and *all* street intersections.

In this chapter it will be demonstrated how the notion of graph centres and medians may be utilised to solve the above two location problems. We shall show that the notion of a graph centre may be used to solve the problem of optimally placing an emergency hospital facility (as we describe in [Section 6.2](#)), whilst the notion of a graph median may be used to solve the problem of optimally placing a newspaper distribution depot (as we describe in [Section 6.3](#)). Our current treatment of these two practical problems is, however, not entirely satisfactory, mainly for two reasons:

First, we assumed that both the hospital facility and the newspaper distribution depot may only be located at a street intersection or dead-end (*i.e.* at a vertex of the graph model). Of course there is no real motivation for such an absurd assumption. Why could these facilities not be located anywhere along a street (*i.e.* why should we limit the location of a facility to the vertices of the graph — why not also allow supply locations anywhere along the arcs or edges of the graph)?

Secondly, we assumed that patients would be visiting the hospital, coming only from street intersections or dead-ends. While motor vehicle accidents are possibly more likely at street intersections than elsewhere along streets, there is no reason why patients with other emergencies might not visit the hospital, starting their journeys from any point along a street (such as their homes, for example). Similarly, there is no reason why newspapers *should* be sold from street corners, although this is traditionally certainly the case. Why not allow for demand locations anywhere along the edges of the graph?

The good news is that graph location theory is able to deal with any one or both of the above objections. Location problems may, in fact, be classified according to three characteristics:

Location of the facility (supply): Whether the facility may be located at a vertex only, or at any point along an edge/arc.

Location of users of the facility (demand): Whether the facility will be used/visited from vertices only, or from arbitrary points along any edges/arcs.

Objective to be minimised: Whether to minimise the total distance from the facility to all vertices or points along edges/arcs, or to minimise the maximum distance from the facility to any vertex or any point along an edge/arc.

Each of the above characteristics may be seen as a binary question: The supply location characteristic question may be answered “vertices only” or “any point along an edge/arc”; the demand location characteristic question may be answered “vertices only” or “arbitrary points along any edges/arcs”; and the objective characteristic question may be answered “minimise total distance from a demand point to all supply points” or “minimise the maximum potential distance from a demand point to a supply point”. Since these questions may be answered independently, the above characterisation gives rise to $2^3 = 8$ fundamentally different types of location problems, as presented schematically by the decision tree in [Figure 6.1](#).

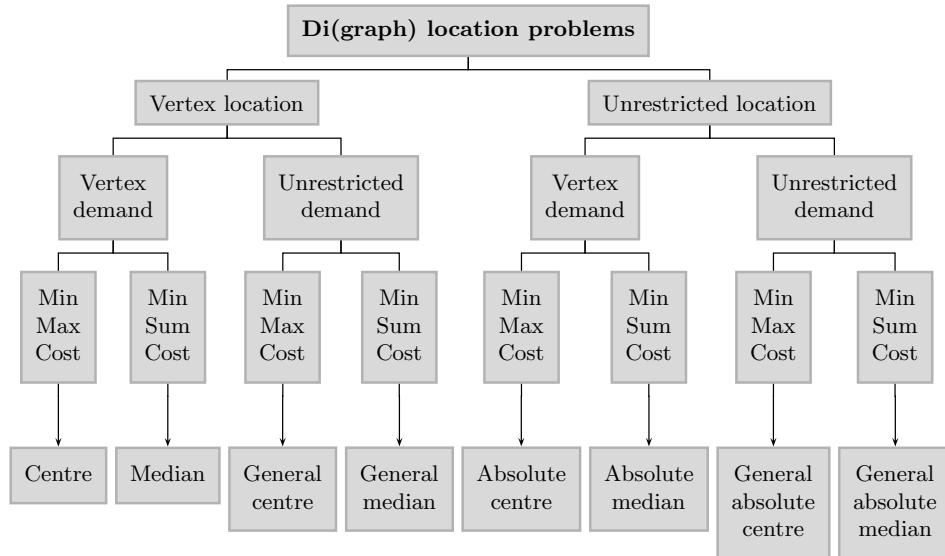


Figure 6.1: Decision tree for graph location problems.

The solutions to each of these eight location problems have different names, as also shown in the figure.

Centre (median): The subgraph induced by any vertices for which vertices farthest away in the graph are as close as possible (for which the sum total of all distances to other vertices of the graph is as small as possible, respectively). In this case both the facility (supply) and demand occur at vertices only. This notion of a centre (median, respectively) is introduced in [Section 6.2](#) ([Section 6.3](#), respectively).

General centre (general median): The subgraph induced by any vertices for which farthest points along edges/arcs in the graph are as close as possible (for which the sum total of the distances to all points along all edges/arcs of the graph is as small as possible, respectively). In this case the facility (supply) is located at a vertex (or vertices) of the graph, whilst the demand occurs at any point along any edge/arc of the graph. The notions of a general centre and of a general median are introduced in [Section 6.4](#).

Absolute centre (absolute median): The subgraph induced by any points along edges/arcs for which the vertices farthest away in the graph are as close as possible (for which the sum total of the distances to all vertices of the graph is as small as possible, respectively). In this case the facility (supply) is located at any point along any edge/arc of the graph, whilst the demand occurs at vertices of the graph. The notions of an absolute centre and of an absolute median are introduced in [Section 6.5](#).

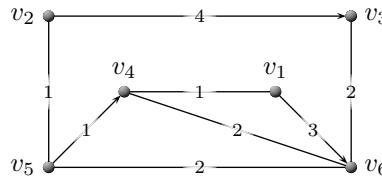
General absolute centre (general absolute median): The subgraph induced by any points along any edges/arcs for which the farthest points along edges/arcs in the graph are as close as possible (for which the sum total of the distances

to all points on all edges/arcs of the graph is as small as possible, respectively). In this case both the facility (supply) and the demand are located at any point along any edge/arc of the graph. The notions of a general absolute centre and of a general absolute median are introduced in [Section 6.6](#).

- ❖ The reader should now be able to attempt [Exercise 6.1](#).

6.2 The centre of a graph

Let v be a vertex in a weighted, (strongly) connected (di)graph G . The **eccentricity** of v , denoted by $e_G(v)$ or $e(v)$ if the graph G is clear from the context, is the distance from v to a vertex furthest from v . The eccentricity of a vertex in a weighted (di)graph G may be thought of intuitively as a measure of its centrality: the larger the eccentricity of a vertex $v \in V(G)$, the more it may be seen as an outlier in terms of the distance measure induced by the edge/arc weights (in the sense that there exist vertices in G which are far away from v). If, on the other hand, the eccentricity of a vertex in G is small, then all other vertices in G are close to v . The distance matrix, $\mathbf{D}(G_{6.1})$, of the digraph $G_{6.1}$ in [Figure 6.2](#) is given in the figure, and was determined by an application of [Floyd's algorithm](#) (see [Section 4.3.2](#)).



	v_1	v_2	v_3	v_4	v_5	v_6	Row max.	Attained by
v_1	0	6	5	1	5	3	6	v_2
v_2	3	0	4	2	1	3	4	v_3
v_3	5	5	0	4	4	2	5	v_1, v_2
v_4	1	5	4	0	4	2	5	v_2
v_5	2	1	4	1	0	2	4	v_3
v_6	3	3	2	2	2	0	3	v_1, v_2

Figure 6.2: A weighted, connected, directed graph, $G_{6.1}$.

The eccentricity of each vertex in $G_{6.1}$ is given by the (boldface) maximum in the corresponding row of the distance matrix, $\mathbf{D}(G_{6.1})$. Hence $e(v_1) = 6$, $e(v_2) = 4$, $e(v_3) = 5$, $e(v_4) = 5$, $e(v_5) = 4$ and $e(v_6) = 3$ for the digraph $G_{6.1}$.

The **radius**, denoted by $\text{rad}(G)$, of a weighted (di)graph G is the minimum eccentricity among the vertices of G , while the **diameter** of G , denoted by $\text{diam}(G)$, is the maximum eccentricity. A vertex v is a **central vertex** of G if $e(v) = \text{rad}(G)$, while v is a **peripheral vertex** of G if $e(v) = \text{diam}(G)$. Notice that the diameter of G is the maximum distance between any two vertices in G . Furthermore, if

$\text{rad}(G) = 1$, then G contains a vertex that is adjacent to every other vertex. The radius and diameter of the digraph $G_{6.1}$ in Figure 6.2 are $\text{rad}(G_{6.1}) = 3$ and $\text{diam}(G_{6.1}) = 6$, respectively.

The next result shows that in case of undirected, weighted graphs specifically, the diameter is at most twice its radius.

Theorem 6.1 *For every connected, weighted graph G satisfying the triangle inequality, $\text{rad}(G) \leq \text{diam}(G) \leq 2 \text{ rad}(G)$.*

Proof The inequality $\text{rad}(G) \leq \text{diam}(G)$ follows directly from the definitions. To verify the inequality $\text{diam}(G) \leq 2 \text{ rad}(G)$, let u and v be vertices in G satisfying $d(u, v) = \text{diam}(G)$. Furthermore, let w be a central vertex of G . Since the distance function satisfies the triangle inequality (see Theorem 2.2(iii)), $\text{diam}(G) = d(u, v) \leq d(u, w) + d(w, v) \leq 2e(w) = 2 \text{ rad}(G)$. ■

The lower bound of Theorem 6.1 is sharp since there exist connected, unweighted graphs G (of every order $n \geq 2$) for which $\text{rad}(G) = \text{diam}(G)$. In fact, the family K_n , $n \geq 2$, of complete graphs satisfies this equality, as does the family of all cycles. The upper bound is also sharp. To see this, let G belong to the family of paths of odd order. That is, $G \cong P_{2k+1}$, $k \geq 1$. Then $\text{rad}(G) = k$ and $\text{diam}(G) = 2k$. Hence for each positive integer k , there exists a connected, unweighted graph G such that $\text{diam}(G) = 2 \text{ rad}(G)$.

The **(classical) centre**, denoted by $C(G)$, of a connected, weighted (di)graph G is the sub(di)graph of G induced by its central vertices. Similarly, the **periphery** of G is the sub(di)graph of G induced by its peripheral vertices. For example, the centre of the digraph $G_{6.1}$ in Figure 6.2 is $C(G_{6.1}) = G_{6.1}[\{v_6\}]$, which is isomorphic to K_1 , whilst the periphery of the digraph $G_{6.1}$ is $G_{6.1}[\{v_1\}]$, which is also isomorphic to K_1 . Returning to our problem of finding an optimal location for an emergency hospital facility at the start of the chapter, it is clear that if the facility were to serve the portion of a city modelled by the digraph $G_{6.1}$, then an optimal placement for the hospital would be at the centre of $G_{6.1}$, i.e. at the intersection labelled v_6 . The worst possible placement for the hospital would be at the periphery of $G_{6.1}$, i.e. at the intersection labelled v_1 .

The centre of a digraph need, however, not be a single vertex, i.e. the answer to our question at the start of the chapter, whether the optimal location for the emergency hospital facility is unique, is negative. In fact, the following theorem (due to Hedetniemi; see [1]) declares that there is absolutely no restriction whatsoever on the possible structure of the centre of a graph; it need not even be connected.

Theorem 6.2 *Every unweighted (di)graph is the centre of some connected, unweighted (di)graph.*

Proof Let H be a given (di)graph. Let G be the (di)graph constructed from H by adding four new vertices v_1, v_2, w_1, w_2 and, for $i = 1, 2$, joining v_i to every vertex of H and to w_i by means of undirected edges (see Figure 6.3). For each vertex v in H , $e_G(v) = 2$, while for $i = 1, 2$, $e_G(v_i) = 3$ and $e_G(w_i) = 4$. Therefore, $\text{rad}(G) = 2$ and the centre of G is the sub(di)graph induced by the vertices of H ; that is, $C(G) = H$. ■

Stephen T Hedetniemi obtained a bachelor's degree in mathematics in 1960, a master's degree in communication science in 1962 and a doctorate in communication science in 1966, all from the University of Michigan. He was appointed as assistant professor of computer science at the University of Iowa in 1967 and, after a short sojourn as mathematician at the Naval Weapons Laboratory in Dahlgren, Virginia in 1972, moved to the Department of Applied Mathematics and Computer Science at the University of Virginia where he was appointed as associate professor. In 1977, he became professor and head in the Department of Computer and Information Science at the University of Oregon, where he remained until 1982. Thereafter he moved to Clemson University, where he is still active as professor of computer science. He also held a visiting professorship at the University of Victoria during the period 1975–76. He has published widely in the areas of domination in graphs, coverings and packings, colouring and partitions, operations on graphs, graph algorithms and chessboard problems.



Biographic note 19: Stephen Hedetniemi (1939–present)

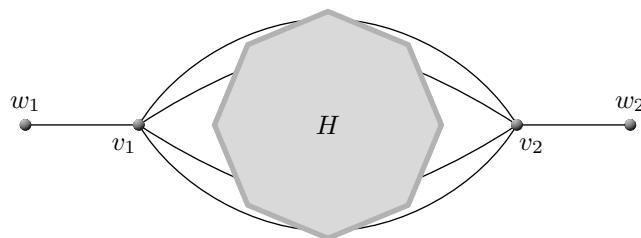


Figure 6.3: H is the centre of G .

The problem of determining the centre of a general (di)graph involves the computation of its vertex eccentricities, via an algorithm such as that of Floyd. For undirected, unweighted trees, however, there exists a much more efficient method for computing centres. This method hinges on the following theorem, a proof of which is left as an exercise.

Theorem 6.3 *Let T be an unweighted, undirected tree of order $p \geq 3$ and let T' be the tree formed by pruning away all the leaves of T . Then $C(T) = C(T')$.*

The above theorem suggests that the centre of an unweighted, undirected tree $T^{(0)}$ may be computed by generating a sequence of successive trees $T^{(0)}, T^{(1)}, T^{(2)}, \dots$, where $T^{(i)}$ is formed by pruning away all the leaves of $T^{(i-1)}$, for all $i = 1, 2, 3, \dots$. In this sequence, each tree is smaller than its predecessor, suggesting that it will be computationally more efficient to compute the centre of $T^{(i)}$, than to calculate the centre of $T^{(i-1)}$. But where does the above sequence of successive trees terminate? The answer to this question is provided by the following theorem, a proof of which is also left as an exercise.

Theorem 6.4 *The centre of every unweighted, undirected tree is isomorphic to K_1 or K_2 .*

The sequence $T^{(0)}, T^{(1)}, T^{(2)}, \dots$ is therefore terminated, by [Theorem 6.4](#), whenever $T^{(j)} \cong K_1$ or $T^{(j)} \cong K_2$ for some $j > 0$. This results in an algorithm (presented in pseudocode as [Algorithm 17](#)) for determining the centre of an unweighted, undirected tree. Notice that [Algorithm 17](#) may therefore be used to find the centre of an unweighted, undirected tree, without having to determine the eccentricities of all its vertices.

The correct working of [Algorithm 17](#) is ensured by a combination of the results of [Theorem 6.3](#) and [Theorem 6.4](#). Let us illustrate the working of [Algorithm 17](#) by means of two examples. Consider the tree $G_{6.2} = T_1^{(0)}$ of order 9 in [Figure 6.4\(a\)](#). The eccentricities of all vertices of $T_1^{(0)}$ are given in blocks within the figure.

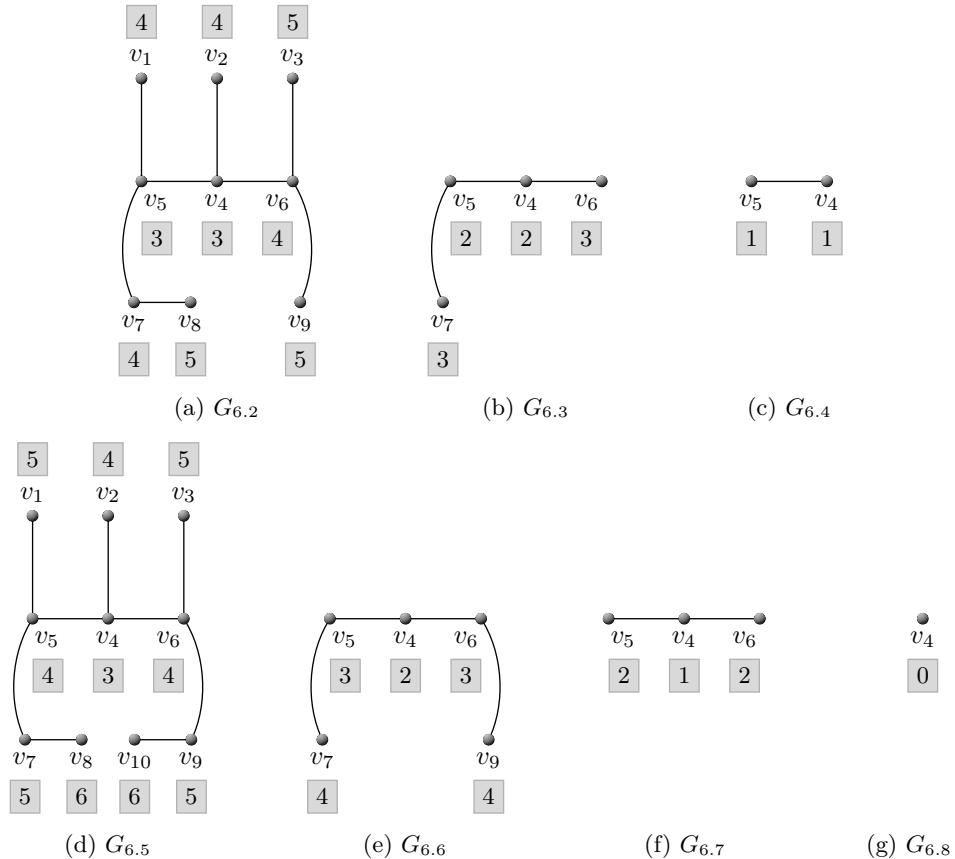


Figure 6.4: Determining the centre of an unweighted, undirected tree.

After applying [Step 6](#) of [Algorithm 17](#) once to the tree $T_1^{(0)}$, the subtree $G_{6.3} = T_1^{(1)}$ in [Figure 6.4\(b\)](#) is obtained. Applying [Step 6](#) in [Algorithm 17](#) to the tree $T_1^{(1)}$ yields the subtree $G_{6.4} = T_1^{(2)}$ in [Figure 6.4\(c\)](#), which is isomorphic to K_2 . Hence the algorithm terminates at this point, resulting in $G_{6.2}[\{v_4, v_5\}]$ as the centre of $G_{6.2}$.

Algorithm 17: An algorithm for computing the centre of an unweighted, undirected tree

Input : An unweighted, undirected tree T .
Output : The centre, $C(T)$, of T .

```

1  $T' \leftarrow T$ 
2 if  $T' \cong K_1$  or  $T' \cong K_2$  then
3    $C(T) \leftarrow T'$ 
4   print  $C(T)$ 
5 else
6   Delete each leaf of  $T'$  to obtain a tree  $T''$ 
7    $T' \leftarrow T''$ 
8   go to Step 2
```

As a further example, we consider the tree $G_{6.5} = T_2^{(0)}$ of order 10 in Figure 6.4(d). After applying Step 6 of Algorithm 17 once to the tree $T_2^{(0)}$, the subtree $G_{6.6} = T_2^{(1)}$ in Figure 6.4(e) is obtained. Another application of Step 6 in Algorithm 17 to the tree $T_2^{(1)}$ yields the subtree $G_{6.7} = T_2^{(2)}$ in Figure 6.4(f). A final application of Step 6 in Algorithm 17 to the tree $T_2^{(2)}$ yields the subtree $G_{6.8} = T_2^{(3)}$ in Figure 6.4(g), which is isomorphic to K_1 . Hence the algorithm terminates at this point, resulting in $G_{6.5}[\{v_4\}]$ as the centre of $G_{6.5}$.

- ❖ The reader should now be able to attempt Exercises 6.2–6.8.

6.3 The median of a graph

The **median**, denoted by $M(G)$, of a weighted, connected (di)graph G is the sub(di)graph of G induced by all vertices $v \in V(G)$ which achieve a minimum value for the sum total of the distances from themselves to *all* other vertices in G , i.e. the median is induced by all vertices $v \in V(G)$ for which

$$\min \left\{ \sum_{u \in V(G)} d_G(u, v) \right\} \quad (6.1)$$

is achieved. Reconsider the distance matrix for the digraph $G_{6.1}$ in Figure 6.2, which is repeated below.

	v_1	v_2	v_3	v_4	v_5	v_6	Row sum
v_1	0	6	5	1	5	3	20
v_2	3	0	4	2	1	3	13
v_3	5	5	0	4	4	2	20
v_4	1	5	4	0	4	2	16
v_5	2	1	4	1	0	2	10
v_6	3	3	2	2	2	0	12

For the computation of the median of $G_{6.1}$, we are interested in the minimum row sum of the distance matrix according to (6.1), rather than in the minimum value of the maximum element in each row, as was the case when determining the centre of $G_{6.1}$. Therefore, the median of $G_{6.1}$ is $M(G_{6.1}) = G_{6.1}[\{v_5\}]$, which

is yet again isomorphic to K_1 . Returning to our problem of finding an optimal location for a newspaper distribution depot at the start of the chapter, it is clear that if the facility were to serve all street intersections in the portion of a city modelled by the digraph $G_{6.1}$ in Figure 6.2, then an optimal placement for the depot would be at the median of $G_{6.1}$, i.e. at the intersection labelled v_5 . Note that $C(G_{6.1}) \neq M(G_{6.1})$. This example, in fact, shows that the centre and the median of a graph may be disjoint.

- ❖ The reader should now be able to attempt Exercise 6.9 and Project 6.1.

6.4 General centres and medians

Recall that the **general centre** of a (di)graph G is a vertex location in G at which a facility may be placed to supply some service to any point along an edge/arc of G if the objective is to minimise the maximal potential distance that one would have to travel from the facility in order to service a user of the facility. Similarly, the **general median** of G is a vertex location in G at which a facility may be placed to supply some service to *all* points along all the edges/arcs of G if the objective is to minimise the sum total that one has to travel in order to supply the service to all users.

Now, to solve the problem of finding a general centre or a general median for any (di)graph, we require a generalised notion of distance. In particular, we require to be able to measure the distance from a given vertex to the farthest point along a given edge/arc of a (di)graph. Let us call such a generalised distance measure the **vertex-to-edge/arc distance** measure. To achieve this generalisation, let $w(uv)$ denote the weight of an undirected edge uv in a digraph G and let $w(u, v)$ denote the weight of a (directed) arc (u, v) . Define the **f -point** of the edge uv (of the arc (u, v) , respectively) as that point along the edge uv (the arc (u, v) , respectively) at a distance $fw(uv)$ ($fw(u, v)$, respectively) from the vertex u and at a distance $(1 - f)w(uv)$ ($(1 - f)w(u, v)$, respectively) from the vertex v , for any real number $f \in [0, 1]$ (when travelling along the edge/arc), as shown in Figure 6.5. Therefore, the 0-point of the edge uv is the vertex u , the 1-point of uv is the vertex v and the $\frac{1}{2}$ -point of uv is the midpoint on the edge uv , and similarly for an arc (u, v) . Any f -point of an edge or arc for which $0 < f < 1$ is called an **internal f -point** of the edge or arc.

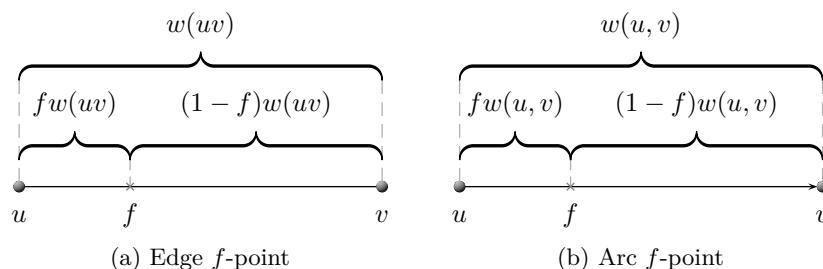


Figure 6.5: The f -point of (a) an edge uv , or (b) an arc (u, v) .

There is a certain value of $f \in [0, 1]$, namely $f = f^*$ (say), for which the vertex-to-edge/arc distance from the vertex i to the f -point of the edge uv or the arc (u, v) is maximal (of course, $f^* = 1$ in the case of a (directed) arc). Denote this maximal distance by $\tilde{d}(i, uv)$ in the case of an undirected edge uv , or by $\tilde{d}(i, (u, v))$ in the case of a (directed) arc (u, v) . Then it is clear that the vertex-to-edge distance $\tilde{d}(i, uv)$ may be written as $\tilde{d}(i, uv) = d(i, u) + f^*w(uv)$ or as $\tilde{d}(i, uv) = d(i, v) + (1 - f^*)w(uv)$. By adding these two expressions for $\tilde{d}(i, uv)$ together, we obtain $2\tilde{d}(i, uv) = d(i, u) + d(i, v) + w(uv)$, which is equivalent to

$$\tilde{d}(i, uv) = \frac{d(i, u) + d(i, v) + w(uv)}{2}. \quad (6.2)$$

On the other hand, there is of course only one way to express the vertex-to-arc distance $\tilde{d}(i, (u, v))$, namely

$$\tilde{d}(i, (u, v)) = d(i, u) + w(u, v). \quad (6.3)$$

Using (6.2) and (6.3), a **generalised vertex-to-edge/arc distance matrix** $\tilde{\mathbf{D}}(G)$ may be formed for a (di)graph G , in which the (i, j) -th entry of $\tilde{\mathbf{D}}(G)$ is the vertex-to-edge/arc distance between vertex i and edge/arc j . The general centre of G is then the sub(di)graph of G induced by the vertices achieving a minimum value of row maxima in $\tilde{\mathbf{D}}(G)$. Similarly, the general median of G is the sub(di)graph of G induced by the vertices achieving a minimum row sum in $\tilde{\mathbf{D}}(G)$.

Let us now use our generalised notion of a vertex-to-edge/arc distance to demonstrate, by means of the example digraph $G_{6.1}$ in Figure 6.2, how a general centre and a general median may be determined.

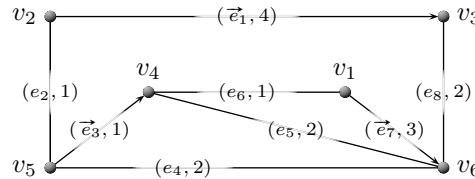


Figure 6.6: A weighted, connected digraph $G_{6.9}$, with labelled edges/arcs. The first entry of each edge/arc label is the name of the edge/arc, while the second entry is the weight associated with the edge/arc.

Figure 6.2 is reproduced as Figure 6.6, after having added edge labels to the digraph $G_{6.1}$ to form $G_{6.9}$. It follows, by (6.3), that $\tilde{d}(v_1, \vec{e}_1) = d(v_1, v_2) + w(v_2, v_3) = 6 + 4 = 10$. Similarly, by (6.2), we have $\tilde{d}(v_1, e_2) = (d(v_1, v_2) + d(v_1, v_5))/2 = (6 + 5 + 1)/2 = 6$. By continuing in this fashion, the generalised vertex-to-edge/arc distance matrix,

	\vec{e}_1	e_2	\vec{e}_3	e_4	e_5	e_6	\vec{e}_7	e_8	Row max.	Attained by	Row sum
v_1	10	6	6	5	3	1	6	5	10	\vec{e}_1	42
v_2	4	1	2	3	3.5	3	6	4.5	6	\vec{e}_7	27
v_3	9	5	5	4	4	5	5	2	9	\vec{e}_1	39
v_4	9	5	5	4	2	1	5	4	9	\vec{e}_1	35
v_5	5	1	1	2	2.5	2	5	4	5	\vec{e}_1, \vec{e}_7	22.5
v_6	7	3	3	2	2	3	3	2	7	\vec{e}_1	25

of the digraph $G_{6.9}$ is found. The subdigraph $G_{6.9}[\{v_5\}]$, which is isomorphic to K_1 , is therefore both the general centre and the general median of $G_{6.9}$.

❖ The reader should now be able to attempt [Exercise 6.10](#).

6.5 Absolute centres and medians

Recall that an **absolute centre** of a (di)graph G is an arbitrary point location along any edge or arc of G at which a facility may be placed to supply some service to any vertex of G so that the maximum distance from the supply point to a demand vertex is minimised. If the objective is, however, to minimise the sum total that one has to travel from a supply point to (demand) vertices of G , then the location is called an **absolute median** of G .

In order to solve the problem of finding an absolute centre or an absolute median of a (di)graph, we require a further generalised notion of distance. Here we need to be able to measure the distance from a given f -point of a specified edge uv (arc (u, v) , respectively) to a vertex i of the (di)graph. Let us call this generalised distance measure the **point-to-vertex distance** and denote it by $\hat{d}(f(uv), i)$ ($\hat{d}(f(u, v), i)$, respectively). If uv is undirected, then there are two possible routes for a shortest path from an f -point of uv to a vertex i , namely from the f -point via u to i (denoted by Route 1 in [Figure 6.7](#)), or from the f -point via v to i (Route 2 in [Figure 6.7](#)). Hence,

$$\hat{d}(f(uv), i) = \min\{fw(uv) + d(u, i), (1 - f)w(uv) + d(v, i)\} \quad (6.4)$$

is the length of the shorter of these two paths.

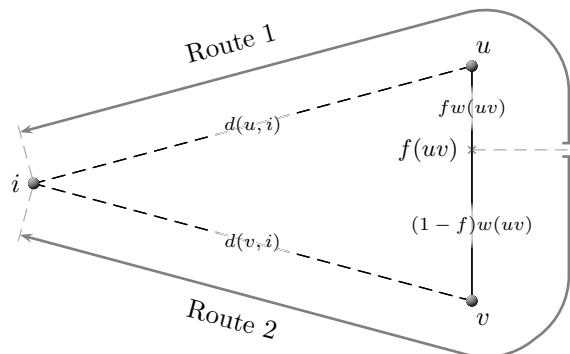


Figure 6.7: Two possible routes for a shortest path from an f -point of the edge uv to the vertex i . The dashed lines represent shortest (directed) u - i and v - i paths respectively.

The graph of $\hat{d}(f(uv), i)$ as a function of f has one of the three possible forms depicted in [Figure 6.8](#). The functional form in [Figure 6.8\(c\)](#) occurs when the absolute difference between $d(u, i)$ and $d(v, i)$ is small compared to the edge weight $w(uv)$. In this case there exists a critical value of f , f^* (say), for which a shortest path from the f -point of uv to the vertex i is either via u to i (Route 1) or via v to i (Route 2). To compute this critical value, we may solve the linear

equation

$$f^*w(uv) + d(u, i) = (1 - f^*)w(uv) + d(v, i)$$

for f^* . If, however, the difference between $d(u, i)$ and $d(v, i)$ is large compared to $w(uv)$, then one of the simpler functional forms in Figures 6.8(a) or (b) occurs, in which case a shortest path from the f -point of uv to the vertex i is fixed (via u to i (Route 1) in the case of Figure 6.8(a) or via v to i (Route 2) in the case of Figure 6.8(b)).

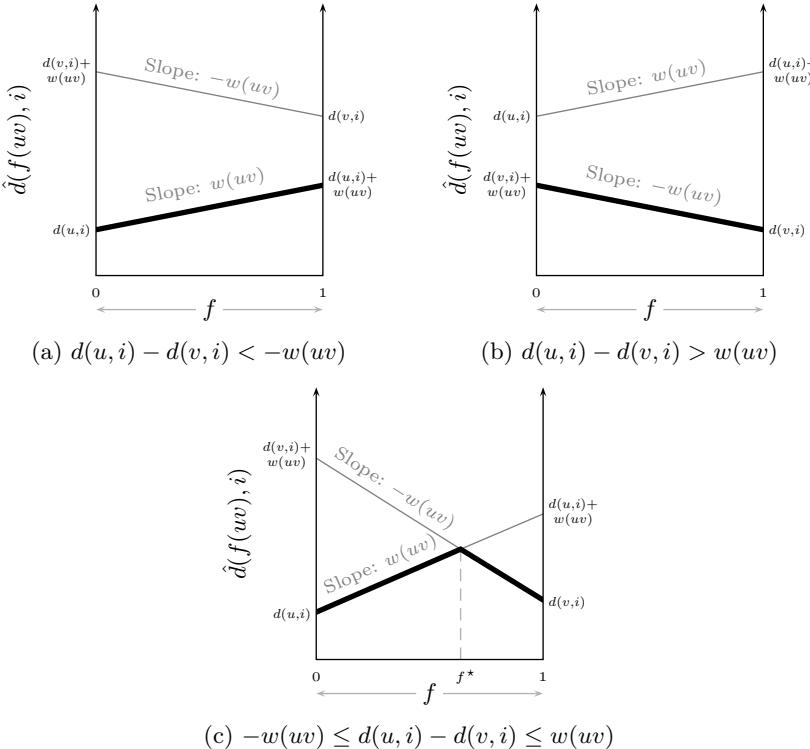


Figure 6.8: The distance $\hat{d}(f(uv), i)$ as a function of f (the piecewise linear solid curve).

In the case of a (directed) arc (u, v) , only the functional form in Figure 6.9 is possible for the shortest distance, $\hat{d}(f(u, v), i)$, from an f -point of (u, v) to the vertex i , because then only Route 2 in Figure 6.7 is possible. Hence,

$$\hat{d}(f(u, v), i) = (1 - f)w(u, v) + d(v, i). \quad (6.5)$$

Let us demonstrate, by means of an example, how the functional forms in Figures 6.8 and 6.9 of the point-to-vertex distances $\hat{d}(f(uv), i)$ and $\hat{d}(f(u, v), i)$ may be used to find an absolute centre of a (di)graph. Consider again the digraph $G_{6.9}$ in Figure 6.6. The distances from f -points of the arc \vec{e}_1 to the various vertices of

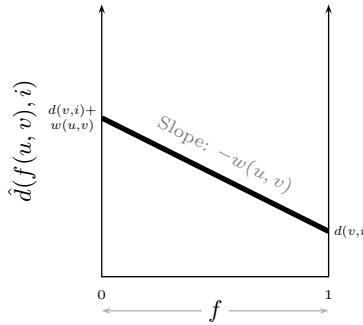


Figure 6.9: The distance $\hat{d}(f(u, v), i)$ as a function of f .

the graph are computed as

$$\left. \begin{aligned} \hat{d}(f(\vec{e}_1), v_1) &= 4(1-f) + 5 = 9 - 4f, \\ \hat{d}(f(\vec{e}_1), v_2) &= 4(1-f) + 5 = 9 - 4f, \\ \hat{d}(f(\vec{e}_1), v_3) &= 4(1-f) + 0 = 4 - 4f, \\ \hat{d}(f(\vec{e}_1), v_4) &= 4(1-f) + 4 = 8 - 4f, \\ \hat{d}(f(\vec{e}_1), v_5) &= 4(1-f) + 4 = 8 - 4f, \\ \hat{d}(f(\vec{e}_1), v_6) &= 4(1-f) + 2 = 6 - 4f. \end{aligned} \right\} \quad (6.6)$$

Similarly, the distances from f -points of the edge e_2 to the various vertices of the graph are computed as

$$\left. \begin{aligned} \hat{d}(f(e_2), v_1) &= \min\{f+3, (1-f)+2\} = 3-f, \\ \hat{d}(f(e_2), v_2) &= \min\{f+0, (1-f)+1\} = f, \\ \hat{d}(f(e_2), v_3) &= \min\{f+4, (1-f)+4\} = \min\{f+4, 5-f\}, \\ \hat{d}(f(e_2), v_4) &= \min\{f+2, (1-f)+1\} = 2-f, \\ \hat{d}(f(e_2), v_5) &= \min\{f+1, (1-f)+0\} = 1-f, \\ \hat{d}(f(e_2), v_6) &= \min\{f+3, (1-f)+2\} = 3-f, \end{aligned} \right\} \quad (6.7)$$

following the convention that u in (6.4) is v_2 (the vertex with smaller subscript incident to e_2) and that v in (6.4) is v_5 (the vertex with larger subscript incident to e_2). The distances in (6.6) and (6.7) are graphed as functions of f in Figures 6.10(a) and (b), respectively.

The absolute centre of a (di)graph is the solution to a minimax problem: we wish to minimise the maximum distance between a supply point (any f -point of an arc or edge) and any demand vertex. Therefore, we restrict our attention to the uppermost sections of the graphs in Figure 6.10. We deduce in this manner that if a supply point were to be located on the arc \vec{e}_1 (see Figure 6.10(a)), then the best location for such a supply point would be the 1-point of that arc, *i.e.* the vertex v_3 . Similarly, if a supply point were to be located on the edge e_2 (see Figure 6.10(b)), then the best location for such a supply point would be the 0-point or the 1-point of that edge, *i.e.* either of the vertices v_2 or v_5 .

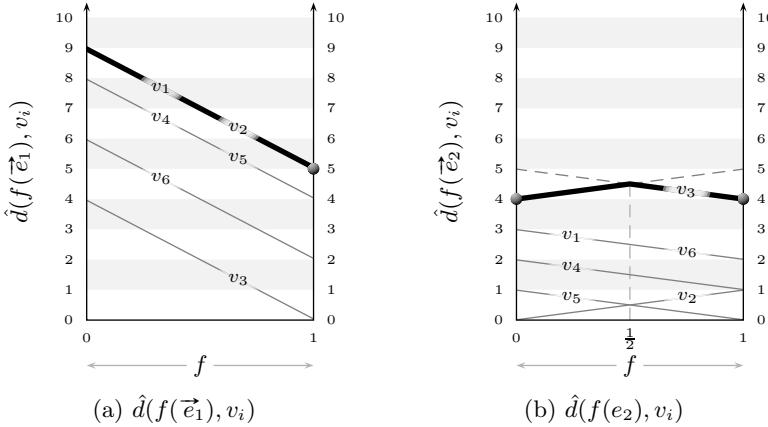


Figure 6.10: The distances (a) $\hat{d}(f(\vec{e}_1), v_i)$ and (b) $\hat{d}(f(e_2), v_i)$ for all $i \in [6]$ as functions of f .

Similar analyses may be carried out for the remaining edges and arcs \vec{e}_3 , e_4 , e_5 , e_6 , \vec{e}_7 and e_8 of $G_{6.9}$. The details of these analyses are omitted for the sake of brevity, but the end results are depicted in Figure 6.11. Upon an examination of Figures 6.10 and 6.11, the conclusions in Table 6.1 are drawn. It is clear, from Table 6.1, that the distance to the farthest demand vertex is minimised by locating a supply point either at the midpoint of the edge e_4 or at the vertex v_6 . The absolute centres of the digraph $G_{6.9}$ therefore contains the vertex v_6 and the midpoint of the edge e_4 .

If supply were to be placed on	Best possible location(s)	Distance to farthest demand vertex
\vec{e}_1	v_3	5
e_2	v_2, v_5	4, 4
\vec{e}_3	v_4	5
e_4	$\frac{1}{2}$ -point, v_6	3, 3
e_5	v_6	3
e_6	v_4	5
\vec{e}_7	v_1	6
e_8	v_6	3

Table 6.1: Supply point candidates for the digraph $G_{6.9}$ if demand occurs at vertices.

The result of the following theorem eases the process of finding an absolute median of a (di)graph considerably.

Theorem 6.5 *For any (strongly) connected, weighted (di)graph G there is always a vertex that induces an absolute median of G .*

Proof The distances $\hat{d}(f(uv), i)$ and $\hat{d}(f(u, v), i)$ are both concave functions of f . Hence the sums

$$\sum_i \hat{d}(f(uv), i) \quad \text{and} \quad \sum_i \hat{d}(f(u, v), i)$$

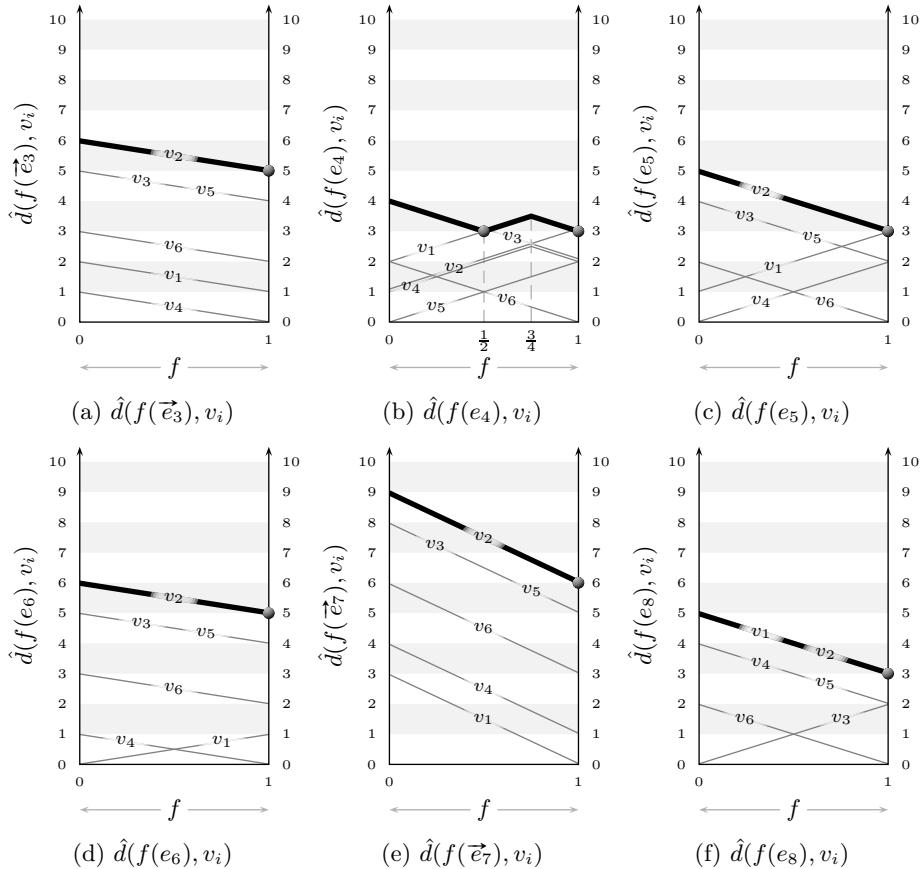


Figure 6.11: The distances (a) $\hat{d}(f(\vec{e}_3), v_i)$, (b) $\hat{d}(f(e_4), v_i)$, (c) $\hat{d}(f(e_5), v_i)$, (d) $\hat{d}(f(e_6), v_i)$, (e) $\hat{d}(f(\vec{e}_7), v_i)$ and (f) $\hat{d}(f(e_8), v_i)$ in the digraph $G_{6.9}$ for all $i \in [6]$ as functions of f .

are both sums of concave functions and therefore also themselves concave functions of f , attaining minimum values at one (or both) of the endpoints $f = 0$ or $f = 1$. Consequently, no interior f -point of an edge uv or an arc (u, v) is a better candidate for absolute median than one of the vertices u or v incident with it. ■

We deduce, from the above theorem, that any vertex in the median of a connected graph is also an absolute median of G . Hence no additional techniques are required when seeking an absolute median of a graph.

Returning to our example digraph $G_{6.9}$ in Figure 6.6, we conclude that the vertex v_5 is therefore also contained in the absolute median of $G_{6.9}$.

❖ The reader should now be able to attempt Exercise 6.11.

6.6 General absolute centres and medians

Recall that a **general absolute centre** of a (di)graph G is an arbitrary point location along any edge or arc of G at which a facility may be placed to supply some service to any point location along an edge or arc of G so that the maximum distance

from the supply point to a demand point is minimised. If, however, the objective is to minimise the sum total of the distances from the supply point to all demand points, then the location is called a **general absolute median** of G .

In order to solve the problem of finding a general absolute centre or a general absolute median of a (di)graph, we require our final generalised notion of distance. Here we need to be able to measure the distance from a given f -point of a specified edge uv (arc (u, v) , respectively) to the farthest point along a particular edge/arc e of the (di)graph. Let us call this generalised distance measure the **point-to-edge/arc distance** and denote it by $\bar{d}(f(uv), e)$ ($\bar{d}(f(u, v), e)$, respectively). In the case of an (undirected) edge $uv \neq e$, a shortest path from an f -point of uv to the farthest point along e must be either via the vertex u or via the vertex v . Therefore,

$$\bar{d}(f(uv), e) = \min\{fw(uv) + \tilde{d}(u, e), (1-f)w(uv) + \tilde{d}(v, e)\}, \quad (6.8)$$

and this function may exhibit any one of the three functional forms shown in Figure 6.12, depending on the value of f , f^* (say), at which the two function values on the right hand side of (6.8) are equal.

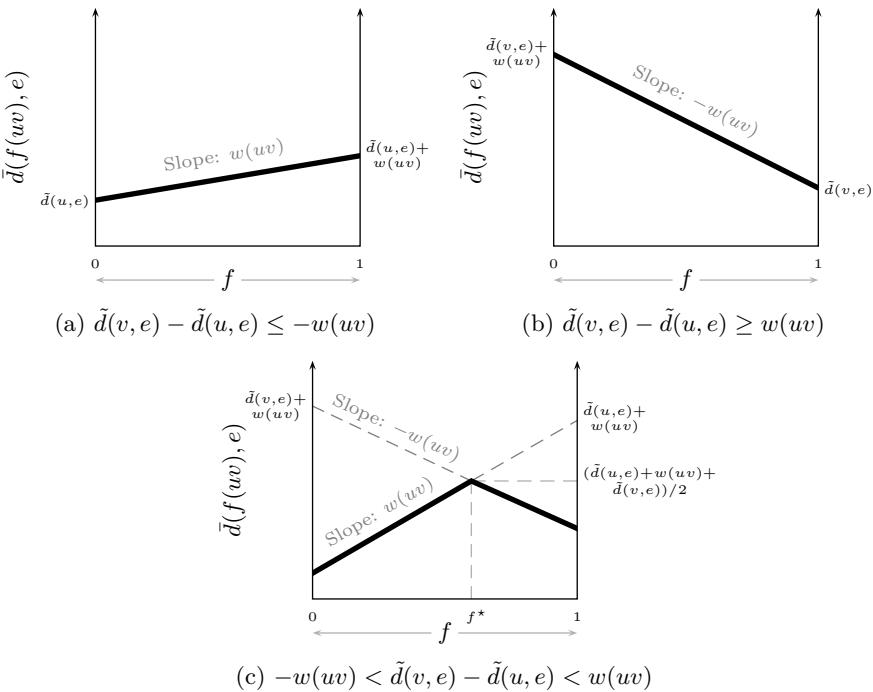


Figure 6.12: Three functional forms of $\bar{d}(f(uv), e)$ when $e \neq uv$: (a) $\tilde{d}(v, e) - \tilde{d}(u, e) \leq w(uv)$; (b) $\tilde{d}(v, e) - \tilde{d}(u, e) \geq w(uv)$; (c) $-w(uv) < \tilde{d}(v, e) - \tilde{d}(u, e) < w(uv)$.

If, however, $uv = e$, then the maximum distance from an f -point of uv to any g -point of e (where $g < f$) cannot exceed

$$A = \min\left\{fw(uv), \overbrace{\frac{w(uv) + d(v, u)}{2}}^{A'}\right\}. \quad (6.9)$$

The first entry in the above minimisation accounts for paths from the f -point to the g -point, restricted to the edge uv . The second entry accounts for routes from the f -point to the g -point, traversing the vertex v . Similarly, the maximum distance from an f -point of uv to any g -point of e (where $g > f$) cannot exceed

$$B = \min \left\{ (1-f)w(uv), \overbrace{\frac{w(uv) + d(u,v)}{2}}^{B'} \right\}. \quad (6.10)$$

The first entry in the above minimisation accounts for paths from the f -point to the g -point, restricted to the edge uv . The second entry accounts for routes from the f -point to the g -point, traversing the vertex u . Consequently,

$$\bar{d}(f(uv), uv) = \max\{A, B\}, \quad (6.11)$$

where A and B are given in (6.9) and (6.10), respectively. Suppose f_A and f_B are the values of f at which equality between the functions on the right hand sides of respectively (6.9) and (6.10) is achieved. Then $\bar{d}(f(uv), uv)$ can have one of four functional forms, depending on the values of f_A and f_B , when viewed as a function of f . These four functional forms are shown in Figure 6.13.

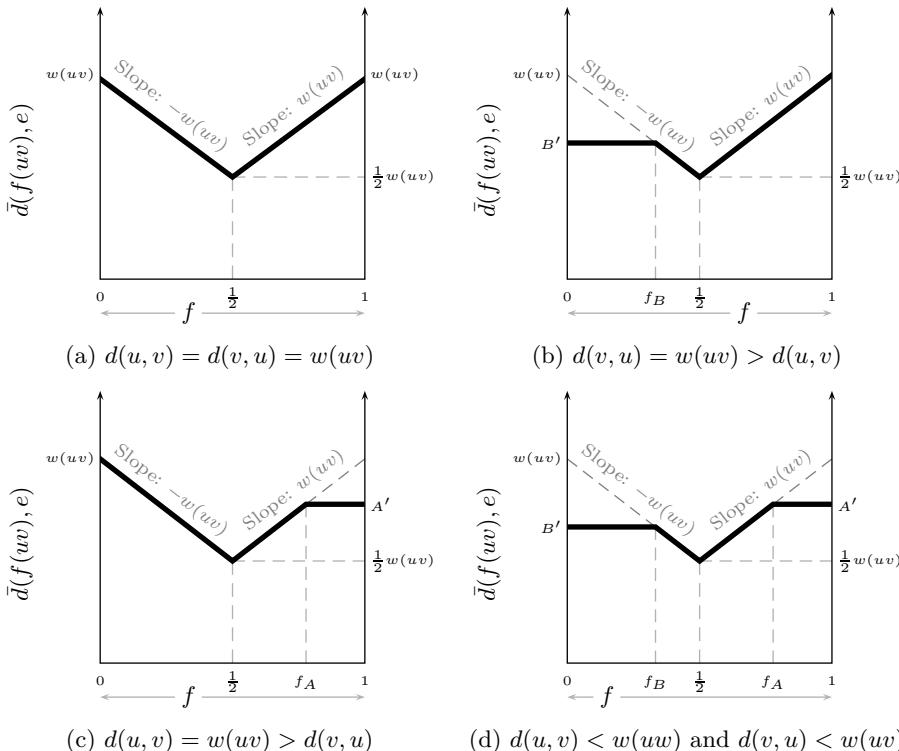


Figure 6.13: Four functional forms of $\bar{d}(f(uv), uv)$ when viewed as a function of f : (a) $d(u, v) = d(v, u) = w(uv)$; (b) $d(v, u) = w(uv) > d(u, v)$; (c) $d(u, v) = w(uv) > d(v, u)$; (d) $d(u, v) < w(uv)$ and $d(v, u) < w(uv)$.

In the case of a (directed) arc $(u, v) \neq e$, the first term in the minimisation in (6.8) may be eliminated, and hence

$$\bar{d}(f(u, v), e) = (1 - f)w(u, v) + \tilde{d}(v, e). \quad (6.12)$$

Finally, if $(u, v) = e$, then the most distant points along e from an f -point of (u, v) are g -points of e , where $g \rightarrow f^-$. In this case, therefore,

$$\bar{d}(f(u, v), (u, v)) = w(u, v) + d(v, u). \quad (6.13)$$

The following useful result, however, tells us that we may ignore interior points of (directed) arcs when seeking the general absolute centre of a (di)graph.

Theorem 6.6 *No interior point of a (directed) arc in a weighted, (strongly) connected (di)graph G can be a general absolute centre or a general absolute median of G .*

Proof Since traversal of a (directed) arc is only in one direction, the terminal vertex of the arc is a better candidate for general absolute centre and for general absolute median than is any interior point of the arc (the terminal vertex is closer to any edge/arc of the graph). ■

Let us demonstrate, by means of an example, how the point-to-edge/arc distances $\bar{d}(f(uv), e)$ and $\bar{d}(f(u, v), e)$ may be used to find the general absolute centre of a (di)graph. Consider again the (di)graph $G_{6.9}$ in Figure 6.6. It follows, by Theorem 6.6, that we may ignore all interior points along the arcs \vec{e}_1 , \vec{e}_3 and \vec{e}_7 . Using (6.8) and (6.11), the following point-to-edge/arc distances are obtained from f -points on the edge e_2 :

$$\left. \begin{aligned} \bar{d}(f(e_2), \vec{e}_1) &= \min\{f + 4, (1 - f) + 5\} = 4 + f, \\ \bar{d}(f(e_2), e_2) &= \max\{f, 1 - f\} \\ \bar{d}(f(e_2), \vec{e}_3) &= \min\{f + 2, (1 - f) + 1\} = 2 - f, \\ \bar{d}(f(e_2), e_4) &= \min\{f + 3, (1 - f) + 2\} = 3 - f, \\ \bar{d}(f(e_2), e_5) &= \min\{f + 3.5, (1 - f) + 2.5\} = 3.5 - f, \\ \bar{d}(f(e_2), e_6) &= \min\{f + 3, (1 - f) + 2\} = 3 - f, \\ \bar{d}(f(e_2), \vec{e}_7) &= \min\{f + 6, (1 - f) + 5\} = 6 - f, \\ \bar{d}(f(e_2), e_8) &= \min\{f + 4.5, (1 - f) + 4\}. \end{aligned} \right\} \quad (6.14)$$

Because the general absolute centre of a (di)graph is the solution to a minimax problem, we again restrict our attention to the uppermost sections of the graphs in Figure 6.14. We deduce in this manner that if a supply point were to be located along the edge e_2 , then the best location for such a supply point would be the 1-point of that edge, i.e. the vertex v_5 .

Similar analyses may be carried out for the remaining edges e_4 , e_5 , e_6 and e_8 of $G_{6.9}$. The details of these analyses are again omitted for the sake of brevity, but the end results are depicted in Figure 6.15. Upon an examination of Figures 6.14 and 6.15, the conclusions in Table 6.2 are drawn. It is clear, from Table 6.2, that the distance to the farthest demand point is minimised by locating a supply point at the vertex v_5 . The general absolute centre of the graph $G_{6.9}$ is therefore induced by the vertex v_5 .

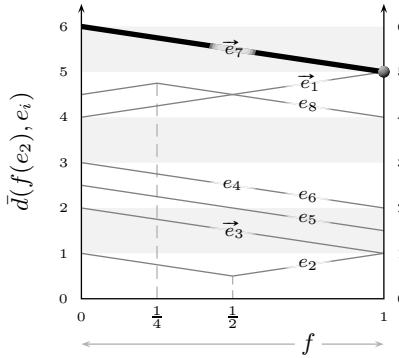


Figure 6.14: The distances $\bar{d}(f(e_2), \vec{e}_1)$, $\bar{d}(f(e_2), e_2)$, $\bar{d}(f(e_2), \vec{e}_3)$, $\bar{d}(f(e_2), e_4)$, $\bar{d}(f(e_2), e_5)$, $\bar{d}(f(e_2), e_6)$, $\bar{d}(f(e_2), \vec{e}_7)$ and $\bar{d}(f(e_2), e_8)$ in the digraph $G_{6.9}$, as functions of f .

If supply were to be placed on	Best possible location(s)	Distance to farthest demand point
\vec{e}_1	v_2	6
e_2	v_5	5
\vec{e}_3	v_5	5
e_4	v_5	5
e_5	v_6	7
e_6	v_4	9
\vec{e}_7	v_6	7
e_8	v_6	7

Table 6.2: Supply point candidates for the digraph $G_{6.9}$ if demand occurs along any point along an edge or arc.

The following useful result tells us that we may ignore interior points of (undirected) edges in certain instances when seeking the general absolute median of a (di)graph.

Theorem 6.7 *No interior point of an (undirected) edge ij in a (di)graph G can be a general absolute median of G if*

$$\left| \sum_{e \in E(G)} \tilde{d}(i, e) - \sum_{e \in E(G)} \tilde{d}(j, e) \right| > \frac{d(i, j) + d(j, i)}{2}. \quad (6.15)$$

Proof The total distance from vertex i to all edges/arcs is

$$\Xi(i) = \tilde{d}(i, ij) + \sum_{e \neq ij} \tilde{d}(i, e) \quad (6.16)$$

and similarly the total distance from vertex j to all edges/arcs is

$$\Xi(j) = \tilde{d}(j, ij) + \sum_{e \neq ij} \tilde{d}(j, e). \quad (6.17)$$

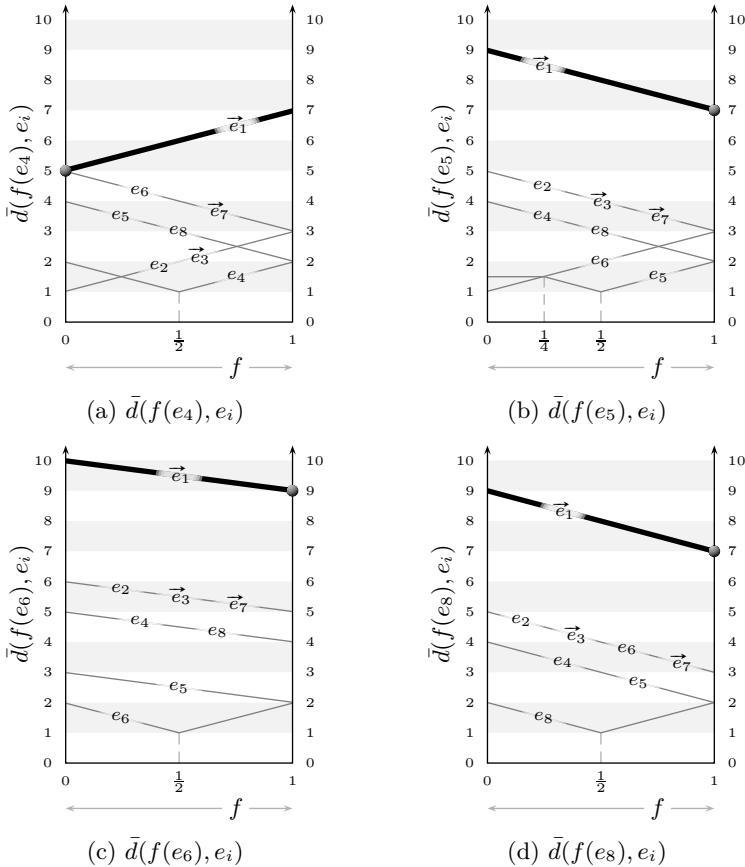


Figure 6.15: The distances $\bar{d}(f(e_i), \vec{e}_1)$, $\bar{d}(f(e_i), e_2)$, $\bar{d}(f(e_i), \vec{e}_3)$, $\bar{d}(f(e_i), e_4)$, $\bar{d}(f(e_i), e_5)$, $\bar{d}(f(e_i), e_6)$, $\bar{d}(f(e_i), \vec{e}_7)$ and $\bar{d}(f(e_i), e_8)$ for (a) $i = 4$, (b) $i = 5$, (c) $i = 6$, and (d) $i = 8$ in the digraph $G_{6,9}$, as functions of f .

Suppose, without loss of generality, that $\Xi(i) \leq \Xi(j)$. Since $\bar{d}(f(ij), e)$ is a concave function of f for all edges/arcs $e \neq ij$, it follows that the function

$$\sum_{e \neq ij} \bar{d}(f(ij), e) \quad (6.18)$$

lies above the straight line joining its endpoints, which are

$$\sum_{e \neq ij} \bar{d}(0(ij), e) = \sum_{e \neq ij} \tilde{d}(i, e) \quad \text{and} \quad \sum_{e \neq ij} \bar{d}(1(ij), e) = \sum_{e \neq ij} \tilde{d}(j, e).$$

Thus, when $f = \frac{1}{2}$, it follows that

$$\sum_{e \neq ij} \bar{d}\left(\frac{1}{2}(ij), e\right) \geq \frac{1}{2} \sum_{e \neq ij} [\tilde{d}(j, e) - \tilde{d}(i, e)].$$

Hence, if f increases from 0 to $\frac{1}{2}$, the quantity in (6.18) increases by at least

$$\frac{1}{2} \sum_{e \neq ij} [\tilde{d}(j, e) - \tilde{d}(i, e)].$$

Let us examine the function $\bar{d}(f(ij), ij)$ next. This function takes a minimum value (of $\bar{d}(\frac{1}{2}(ij), ij)$) at $f = \frac{1}{2}$. As f increases from 0 to $\frac{1}{2}$, $\bar{d}(f(ij), ij)$ experiences a decrease of magnitude $\bar{d}(i, ij) - \frac{1}{2}w(ij)$. If $\sum_e \bar{d}(f(ij), e) < \Xi(i)$ for some value $0 < f < 1$, then it is necessary that the maximum increase of $\bar{d}(f(ij), ij)$ at $f = \frac{1}{2}$ should be at least the minimum increase of $\sum_e \bar{d}(f(ij), e)$ at $f = \frac{1}{2}$. In other words, if an interior point of ij is to be a better candidate for general absolute median than the vertex i , then it is necessary that

$$\tilde{d}(i, ij) - \frac{w(ij)}{2} \geq \frac{1}{2} \sum_{e \neq ij} [\tilde{d}(j, e) - \tilde{d}(i, e)]$$

or, equivalently, that

$$\tilde{d}(j, ij) + \tilde{d}(i, ij) - w(ij) \geq \Xi(j) - \Xi(i). \quad (6.19)$$

Substitution of (6.2) into (6.19) yields

$$\frac{d(i, j) + d(j, i)}{2} \geq \Xi(j) - \Xi(i) \quad (6.20)$$

after simplification. If, however, we had originally rather assumed that $\Xi(j) \leq \Xi(i)$, then inequality (6.20) would have been

$$\frac{d(i, j) + d(j, i)}{2} \geq \Xi(i) - \Xi(j). \quad (6.21)$$

Combining inequalities (6.20) and (6.21) yields the desired result. ■

Applying Theorem 6.7 to the digraph $G_{6.9}$ in Figure 6.6, the interior points of all (undirected) edges may be eliminated as candidates for general absolute median, as may be seen in Table 6.3.

Edge ij	$\left \sum_{e \in E(G)} \tilde{d}(i, e) - \sum_{e \in E(G)} \tilde{d}(j, e) \right $	$\frac{d(i, j) + d(j, i)}{2}$	Eliminate?
$e_2 = v_2v_5$	$ 27 - 22.5 = 4.5$	$\frac{1+1}{2} = 1$	✓
$e_4 = v_5v_6$	$ 22.5 - 25 = 2.5$	$\frac{2+2}{2} = 2$	✓
$e_5 = v_4v_6$	$ 35 - 25 = 10$	$\frac{2+2}{2} = 2$	✓
$e_6 = v_1v_4$	$ 42 - 35 = 7$	$\frac{1+1}{2} = 1$	✓
$e_8 = v_3v_6$	$ 39 - 25 = 14$	$\frac{2+2}{2} = 1$	✓

Table 6.3: Elimination of interior points of (undirected) edges as candidates for general absolute median of the digraph $G_{6.9}$ in Figure 6.6.

Since no interior points of the (directed) arcs \vec{e}_1 , \vec{e}_3 or \vec{e}_7 can be a general absolute median of $G_{6.9}$ either, by Theorem 6.6, it follows that the general median vertex v_5 uncovered in Section 6.4 is also contained in the general absolute median of $G_{6.9}$. In summary, the locations of all the centres and medians of the digraph $G_{6.9}$ in Figure 6.6 are listed in Table 6.4.

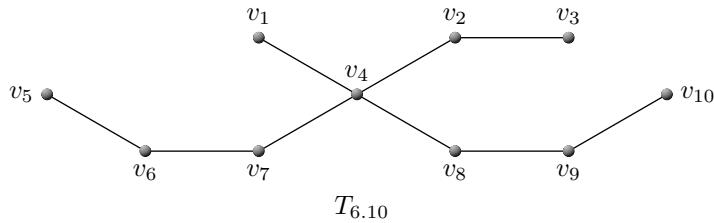
	Demand	Supply	Location	Sum or farthest distance
Classical centre	vertices only	vertices only	v_6	3
General centre	any f -point	vertices only	v_5	5
Absolute centre	vertices only	any f -point	$v_6, \frac{1}{2}e_4$	3
General absolute centre	any f -point	any f -point	v_5	5
Classical median	vertices only	vertices only	v_5	10
General median	any f -point	vertices only	v_5	22.5
Absolute median	vertices only	any f -point	v_5	22.5
General absolute median	any f -point	any f -point	v_5	22.5

Table 6.4: Centres and medians of the digraph $G_{6.9}$ in [Figure 6.6](#).

Exercises

- 6.1 Consider a (di)graph model of a portion of a city, where the edges (arcs, respectively) represent streets (one-way streets, respectively) and where the vertices represent street intersections or street dead-ends. Classify the solution to each of the following location problems either as a classical centre, a classical median, a general centre, a general median, an absolute centre, an absolute median, a general absolute centre or a general absolute median of the (di)graph model:
- (a) A location for building a depot from where newspapers have to be distributed in single batches to vendors at street corners/street dead-ends so that the total distribution distance travelled is a minimum, if the distribution vehicle has to return to the depot after every delivery of a batch of newspapers to a street corner/street dead-end, and if the depot has to be situated:
 - (i) at a street intersection/dead-end;
 - (ii) anywhere along a street.
 - (b) A location for building a pension collection venue in a portion of a city containing only two-way streets so that senior citizens may travel (from their homes) to collect their monthly pensions without having to travel an excessive distance to the venue, if the venue has to be situated:
 - (i) at a street intersection/dead-end;
 - (ii) anywhere along a street.
- 6.2 Prove that, if $uv \in E(G)$, then $|e(u) - e(v)| \leq w(uv)$, for a weighted, connected, undirected graph G , whose edge weights satisfy the triangle inequality.
- 6.3 Prove [Theorem 6.3](#).
- 6.4 Prove [Theorem 6.4](#).

6.5 Use [Algorithm 17](#) to determine the centre of the unweighted, undirected tree $T_{6.10}$:

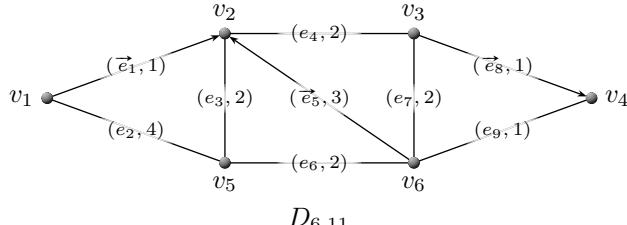
 $T_{6.10}$

6.6 A (di)graph G is called **self-centred** if $C(G) = G$.

- (a) Determine and sketch all self-centred unweighted, undirected trees.
- (b) Prove that every connected self-centred unweighted, undirected graph contains no cut-vertices.

6.7 For the labelled digraph $D_{6.11}$ below, find

- (a) the eccentricity $e(v_i)$ of each vertex $v_i \in V(D_{6.11})$;
- (b) the radius $\text{rad}(D_{6.11})$ and diameter $\text{diam}(D_{6.11})$;
- (c) the centre $C(D_{6.11})$.

 $D_{6.11}$

The first entry of each edge/arc label is the name of the edge/arc, while the second entry is the weight associated with the edge/arc.

6.8 A **universal vertex** of a graph is a vertex that is adjacent to all other vertices of the graph. Show that a graph H is the periphery of some connected graph if and only if H contains no universal vertex.

6.9 Find the median of:

- (a) the tree $T_{6.10}$ of [Exercise 6.5](#);
- (b) the digraph $D_{6.11}$ of [Exercise 6.7](#).

6.10 Find both the general centre and the general median of:

- (a) the tree $T_{6.10}$ of [Exercise 6.5](#);
- (b) the digraph $D_{6.11}$ of [Exercise 6.7](#).

6.11 Find both the absolute centre and the absolute median of:

- (a) the tree $T_{6.10}$ of [Exercise 6.5](#);
- (b) the digraph $D_{6.11}$ of [Exercise 6.7](#).

Computer exercises

The **MATHEMATICA** command **VertexEccentricity**[G, v] may be used to compute the eccentricity of a vertex v in a (di)graph G . A list of these eccentricity values for all vertices of G may be computed by the command **Table**[$\text{expr}, \{i, i_{\min}, i_{\max}\}$], which generates a list of values of the expression expr when i runs from i_{\min} to i_{\max} . **MATHEMATICA** uses the values of these eccentricities to determine the radius and diameter of G , which may be retrieved by means of the commands **GraphRadius**[G] and **GraphDiameter**[G], respectively. Furthermore, the **MATHEMATICA** command **GraphCenter**[G] returns the (classical) centre of G . Unfortunately, **MATHEMATICA** has no built-in function for computing the median of a (di)graph. For example, the commands

```
In[1]:= m = {{Infinity, Infinity, Infinity, 1, Infinity, Infinity}, {Infinity,
           Infinity, 4, Infinity, 1, Infinity}, {Infinity, Infinity, Infinity,
           Infinity, Infinity, 2}, {1, Infinity, Infinity, Infinity, Infinity,
           2}, {Infinity, 1, Infinity, 1, Infinity, 2}, {3, Infinity, 2, 2, 2,
           Infinity}};
In[2]:= MatrixForm[m]
In[3]:= G = WeightedAdjacencyGraph[m, VertexLabels -> "Name", EdgeLabels ->
          "EdgeWeight"]
In[4]:= Table[VertexEccentricity[G, v], {v, 1, 6}]
In[5]:= GraphRadius[G]
In[6]:= GraphDiameter[G]
In[7]:= GraphCenter[G]
```

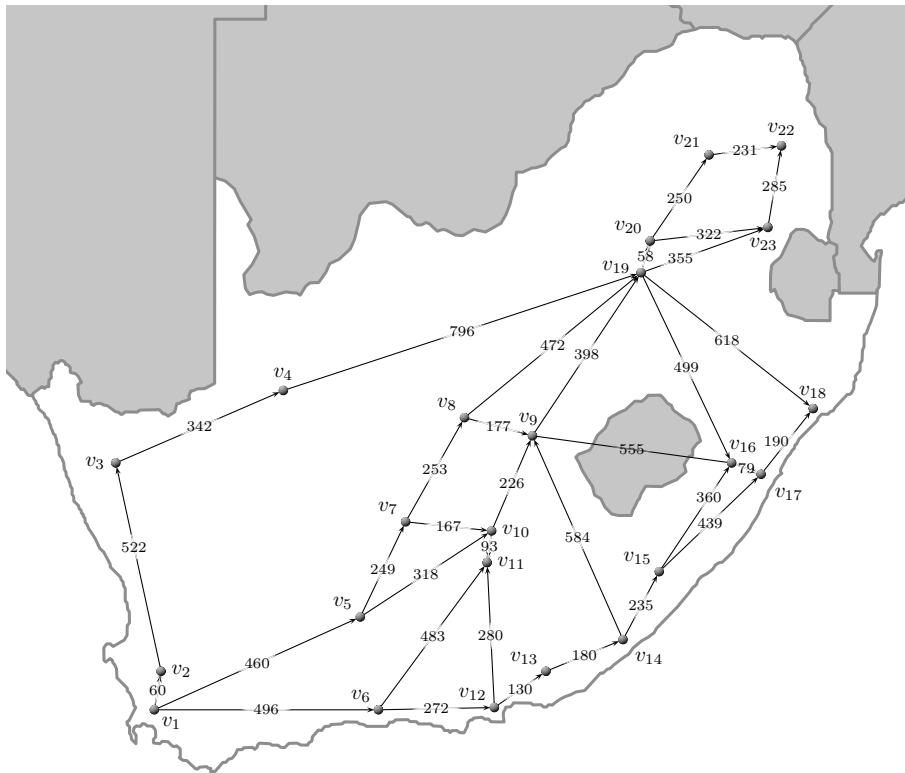
produce the output:

```
Out[2]:= {{∞, ∞, ∞, 1, ∞, ∞}, {∞, ∞, 4, ∞, 1, ∞}, {∞, ∞, ∞, ∞, ∞, 2}, {1, ∞, ∞, ∞, ∞, 2}, {∞, 1, ∞, 1, ∞, 2}, {3, ∞, 2, 2, 2, ∞}}
Out[3]:= Graph[{{1, 2, 3, 4, 5, 6}}, {DirectedEdge[1, 2, 1], DirectedEdge[1, 3, 3], DirectedEdge[2, 1, 1], DirectedEdge[2, 3, 2], DirectedEdge[2, 4, 2], DirectedEdge[3, 1, 1], DirectedEdge[3, 2, 2], DirectedEdge[3, 4, 2], DirectedEdge[4, 2, 2], DirectedEdge[4, 3, 2], DirectedEdge[4, 5, 1], DirectedEdge[5, 4, 1], DirectedEdge[5, 6, 1], DirectedEdge[6, 5, 1], DirectedEdge[6, 3, 2], DirectedEdge[6, 4, 2], DirectedEdge[3, 6, 2], DirectedEdge[4, 6, 2], DirectedEdge[5, 6, 2]}, {EdgeStyle -> {1 -> 2, 1 -> 3, 2 -> 1, 2 -> 3, 2 -> 4, 3 -> 1, 3 -> 2, 3 -> 4, 4 -> 2, 4 -> 3, 4 -> 5, 5 -> 4, 5 -> 6, 6 -> 5, 6 -> 3, 6 -> 4}], {1, 2, 3, 4, 5, 6}, {1, 2, 3, 4, 5, 6}]}
Out[4]:= {6., 4., 5., 5., 4., 3.}
Out[5]:= 3.
Out[6]:= 6.
Out[7]:= {6}
```

- ❖ The reader should now be able to reattempt Exercises 6.5 and 6.7 using only **MATHEMATICA**.

Projects

This section contains two projects. Both projects require the location of a variety of facilities, such as a legislature, a blood transfusion processing unit, an emergency power generator, a newspaper distribution depot and a fire hydrant pump station in either a graph model ([Project 6.1](#)) or a digraph model ([Project 6.2](#)) of applicable infrastructure.



City			City			City		
v_1	Cape Town		v_9	Bloemfontein		v_{17}	Durban	
v_2	Malmesbury		v_{10}	Colesberg		v_{18}	Empangeni	
v_3	Nababeep		v_{11}	Middelburg		v_{19}	Johannesburg	
v_4	Upington		v_{12}	Gqeberha ^a		v_{20}	Tshwane ^b	
v_5	Beaufort West		v_{13}	Grahamstown		v_{21}	Polokwane ^c	
v_6	Knysna		v_{14}	East London		v_{22}	Phalaborwa	
v_7	Britstown		v_{15}	Umtata		v_{23}	Nelspruit	
v_8	Kimberley		v_{16}	Pietermaritzburg				

^aFormerly known as Port Elizabeth and then as Nelson Mandela Bay.

^bFormerly known as Pretoria.

^cFormerly known as Pietersburg.

Figure 6.16: Distance map of South Africa. Graph vertices are referenced within the accompanying table. The edge weights represent distances in kilometres.

Project 6.1: National location decision support

The map in Figure 6.16 shows the main towns and cities in South Africa, together with the distances (measured in kilometres) between towns and cities that are directly linked by means of the national roads network.

Tasks

1. The South African government has, for some time, contemplated moving the legislature from Cape Town to some more central location, so as to realise the objective that no parliamentarian should have to travel too far to attend a sitting of parliament, should they travel by road. Use an appropriate centre or median for the graph model of the map in [Figure 6.16](#) to find the best town or city location(s) for new houses of parliament. Show all your working clearly.
2. The South African Blood Transfusion Service has, for some time, contemplated the establishment of a single, central blood processing unit, from where blood may be delivered by road on a weekly basis to blood banks in all the major towns and cities in South Africa. Use an appropriate centre or median for the graph model of the map in [Figure 6.16](#) to find the best town or city location(s) for such a central blood processing unit. Show all your working clearly.

Project 6.2: Local location decision support

Consider the central business district (CBD) of the suburb of *Mountain View* enclosed between the four perimeter streets: DAPHNE, FELIX, KAREL TRICHARDT and DANIEL, as shown in [Figure 6.17](#). The dimensions of a street block is 200×300 metres, as shown in the CBD graph model in [Figure 6.18](#). The two *cul-de-sacs* enclosed by MIGNON, DENYSSEN, LEONIDE and IRVINE are each 150 metres in length and open into IRVINE 100 metres apart and 100 metres from MIGNON and LEONIDE streets respectively, as shown in [Figure 6.18](#). DENYSSEN and IRVINE streets are one-way streets, as indicated.

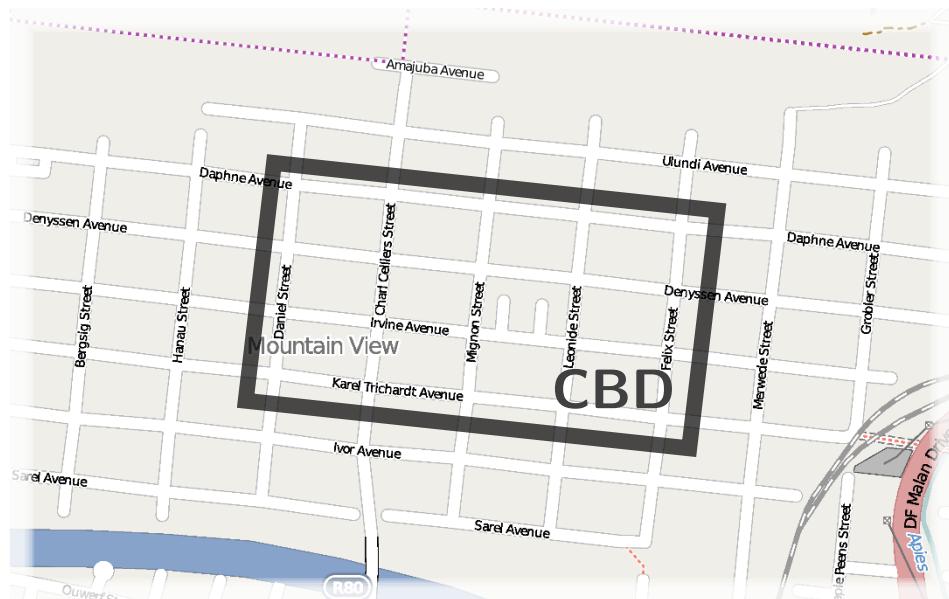


Figure 6.17: Map of the CBD of the suburb of *Mountain View*.

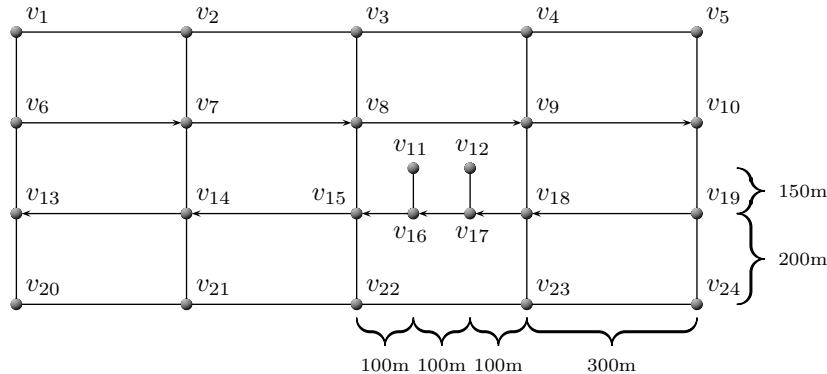


Figure 6.18: Digraph model of the *Mountain View* CBD.

Tasks

1. The Municipality of *Mountain View* has decided to install a single emergency electrical power generator somewhere at a street intersection or dead-end to serve the businesses along all streets of the CBD. Use an appropriate centre or median for the digraph model of the CBD in Figure 6.18 to find the best location(s) for such an emergency power generator in the sense that the cables that have to be installed from the generator to all the businesses in the CBD should be as short as possible in total (as part of a cost savings initiative). Show all your working clearly.
2. The local newspaper, the *Mountain View Citizen*, is interested in building a depot from where newspapers are to be delivered daily in single batches to all street corners of the CBD. Use an appropriate centre or median for the digraph model of the CBD in Figure 6.18 to find the location(s) along any street for such a depot that will render the total newspaper delivery distance covered from the depot each day, a minimum. Show all your working clearly.
3. The Municipality of *Mountain View* wants to install a pump station for the CBD fire hydrant system. This pump station is to be connected by means of a pipe system (buried under the streets) to every fire hydrant (there is a fire hydrant at every street corner and dead-end). Use an appropriate centre or median for the digraph model of the CBD in Figure 6.18 to advise the Municipality as to the location(s) for such a pump station that would render the total connecting pipe length as short as possible (as part of a cost savings initiative). Show all your working clearly.

Further reading

-
- [1] F Buckley, Z Miller and PJ Slater, 1981. *On graphs containing a given graph as center*, Journal of Graph Theory, **5**, pp. 427–434.
 - [2] JR Evans and E Minieka, 1992. *Optimization Algorithms for Networks and Graphs*, Second edition, Marcel Dekker, New York (NY), pp. 362–389, Chapter 10.



Maximum flow networks

Contents

- 7.1 Introduction 173
 - 7.2 Preliminary concepts 174
 - 7.3 The max-flow min-cut theorem 177
 - 7.4 The max-flow min-cut algorithm 179
- Exercises 183
Computer exercises 184
Projects 187
Further reading 191

7.1 Introduction

You are designing an oil pipeline. Oil is to be pumped from a distribution point s to a refinery t . To minimise the possibility that pipe failures and repairs will completely halt the flow of oil, there are typically several routes along which oil may flow from s to t . Suppose there are four pumping stations u , v , x and y along the pipelines, which are linked as shown in the digraph $N_{7.1}$ of Figure 7.1. The arrows indicate the directions in which oil can flow and the capacities of the sections of pipelines are indicated (in thousands of barrels of oil per hour). The question you are faced with is to determine the maximum volume of oil that can be pumped through the system (from s to t) per hour. In this chapter, we develop a method of solving maximum flow problems of this type.

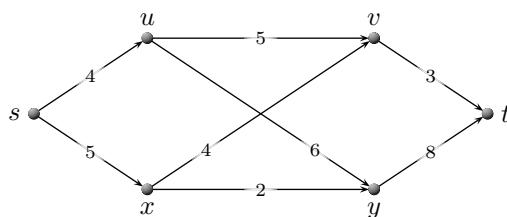


Figure 7.1: Example of an oil pipeline network, modelled as a digraph, $N_{7.1}$.

Before we attempt to solve the problem just mentioned, we adopt several fairly natural restrictions. First, no pipe can carry more oil than its capacity allows and, secondly, all intermediate stations pump out as much oil as they receive — oil therefore cannot accumulate at these stations. We formulate the aforementioned flow problem more precisely in the following section.

7.2 Preliminary concepts

A **network** N is a digraph D with a nonnegative capacity $c_N(e)$ on each arc e , collectively called the **capacity function on** N , along with two distinguished vertices s and t called the **source** and **sink**, respectively. The digraph D is called the **underlying digraph** of the network N . If the network N is clear from the context, we simply denote $c_N(e)$ by $c(e)$.

Intuitively, the capacity $c_N(x, y)$ of an arc (x, y) may be thought of as the maximum volume of some material that can be transported from x to y per unit time. For example, the capacity of the arc (x, y) may represent the number of seats available on a direct flight from city x to city y in some airline system. On the other hand, this capacity might be the capacity of a pipeline from pumping station x to pumping station y in an oil network. The problem, in general, is to maximise the “flow” of material from the source s to the sink t without exceeding the capacities of the arcs.

A network may be represented by drawing its underlying digraph D and labelling each arc of D with its capacity. For example, Figure 7.1 contains a network with $c(u, v) = 5$ and $c(x, y) = 2$.

Let N be a network with underlying digraph D , source s , sink t and capacity function c . A **flow** f in N is an integer-valued function on $E(D)$ that satisfies (i) $0 \leq f(e) \leq c(e)$ for each arc $e \in E(D)$ (the capacity constraint), and (ii) $f^+(v) = f^-(v)$ for every vertex $v \in V(D) \setminus \{s, t\}$ (a conservation constraint), where $f^+(v)$ denotes the total flow along edges exiting v and $f^-(v)$ denotes the total flow along edges entering v , i.e. $f^+(v) = \sum_{w \in N^+(v)} f(v, w)$ and $f^-(v) = \sum_{w \in N^-(v)} f(w, v)$.

For a vertex $v \in V(D)$, the **net flow out of** v is defined as $f^+(v) - f^-(v)$, while the **net flow into** v is defined as $f^-(v) - f^+(v)$. The **value** $f(N)$ of a flow in N is the net flow

$$f(N) = f^+(s) - f^-(s)$$

out of the source s . A **maximum flow** is a flow of maximum value.

A flow may therefore be seen as a mapping that describes the movement (or flow) of material along the arcs of the network, while the capacity is a mapping that describes the maximum volume of material that can move along the individual arcs of the network. The capacity constraint states that the flow in an arc cannot exceed its capacity, and the flow conservation states that all flow into a vertex, other than the source s and sink t , also flows out of that vertex. The **zero flow** assigns a flow of 0 to each arc in the network.

Figure 7.2 shows a flow f in a network $N_{7.2}$. An ordered pair is associated with each arc, where the first number denotes the capacity of the arc and the second number the actual flow along the arc. For example, in the network $N_{7.2}$, $f(u, v) = 1$ while $c(u, v) = 5$. The value of the flow is $f(N_{7.2}) = f^+(s) - f^-(s) = 8 - 0 = 8$.

Before presenting our first result on networks, we introduce some additional notation. Let $D = (V, E)$ be a digraph, and let $X, Y \subseteq V$ with $X, Y \neq \emptyset$. We

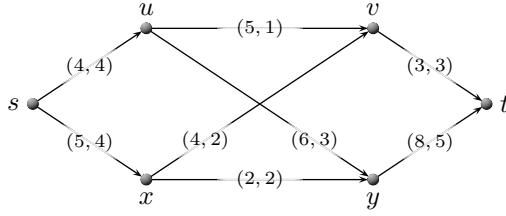


Figure 7.2: A sample network, $N_{7.2}$.

denote the set of all arcs directed from some vertex in X to some vertex in Y by

$$(X, Y) = \{(x, y) \in E \mid x \in X \text{ and } y \in Y\}.$$

For example, in Figure 7.2, if $X = \{u, v, x\}$ and $Y = \{x, y, t\}$, then $(X, Y) = \{(u, y), (v, t), (x, y)\}$. For a flow f and a capacity function c in a network N , we define

$$f(X, Y) = \sum_{e \in (X, Y)} f(e) \quad \text{and} \quad c(X, Y) = \sum_{e \in (X, Y)} c(e),$$

with the convention that $f(X, Y) = c(X, Y) = 0$ if $(X, Y) = \emptyset$. For a subset $X \subseteq V$, we denote the complement $V \setminus X$ of X by \bar{X} .

Let N be a network with underlying digraph $D = (V, E)$, source s , sink t , capacity function c and flow f . If $S \subseteq V$ is such that $s \in S$ and $t \in \bar{S}$, we call the pair (S, \bar{S}) a **cut in N** , and we call $c(S, \bar{S})$ the **capacity of the cut**. A **minimum cut** is a cut of minimum value. We call $f(S, \bar{S})$ the **flow from S to \bar{S}** and we call $f(\bar{S}, S)$ the **flow from \bar{S} to S** .

For example, suppose f is the flow in the network of Figure 7.2 and that $S = \{s, v, y\}$. Then, $\bar{S} = \{u, x, t\}$ and the cut (S, \bar{S}) is given by $(S, \bar{S}) = \{(s, u), (s, x), (v, t), (y, t)\}$. Thus, $c(S, \bar{S}) = 20$ and $f(S, \bar{S}) = 16$. Furthermore, $(\bar{S}, S) = \{(u, v), (u, y), (x, v), (x, y)\}$, and so $f(\bar{S}, S) = 8$.

We are now in a position to present our first result on network flows.

Theorem 7.1 *Let N be a network and f a flow in N . If (S, \bar{S}) is a cut of N , then*

$$f(N) = f(S, \bar{S}) - f(\bar{S}, S).$$

Proof By definition, $f(N) = f^+(s) - f^-(s)$. By the flow conservation constraint, we have $f^+(v) = f^-(v)$ for all $v \in S \setminus \{s\}$. Hence we may write

$$f(N) = \sum_{v \in S} (f^+(v) - f^-(v)).$$

Since $S \cup \bar{S} = V(D)$, where D is the underlying digraph of N ,

$$\sum_{v \in S} f^+(v) = \sum_{(v, u) \in (S, S)} f(v, u) + \sum_{(v, u) \in (S, \bar{S})} f(v, u) = f(S, S) + f(S, \bar{S}),$$

while

$$\sum_{v \in S} f^-(v) = \sum_{(u, v) \in (S, S)} f(u, v) + \sum_{(u, v) \in (\bar{S}, S)} f(u, v) = f(S, S) + f(\bar{S}, S).$$

Thus,

$$f(N) = \sum_{v \in S} f^+(v) - \sum_{v \in S} f^-(v) = f(S, \bar{S}) - f(\bar{S}, S). \quad \blacksquare$$

As an immediate consequence of [Theorem 7.1](#), we have the following corollary.

Corollary 7.2 *If f is a flow in a network N with sink t , then the value of the flow is the net flow into t , i.e.*

$$f(N) = f^-(t) - f^+(t).$$

Proof Let D be the underlying digraph of N and let $S = V(D) \setminus \{t\}$. Since $\bar{S} = \{t\}$,

$$f(S, \bar{S}) = \sum_{w \in N^-(t)} f(w, t) = f^-(t),$$

while

$$f(\bar{S}, S) = \sum_{w \in N^+(t)} f(t, w) = f^+(t).$$

Thus, by [Theorem 7.1](#), $f(N) = f(S, \bar{S}) - f(\bar{S}, S) = f^-(t) - f^+(t)$. ■

Consider, as an example, the network flow in $N_{7.3}$ shown in [Figure 7.3](#) where, as before, the first number associated with an arc e is its capacity $c(e)$ and the second number its flow $f(e)$. The value of the flow is $f(N_{7.3}) = f^+(s) - f^-(s) = (4 + 6) - 0 = 10 = (5 + 5) - 0 = f^-(t) - f^+(t)$.

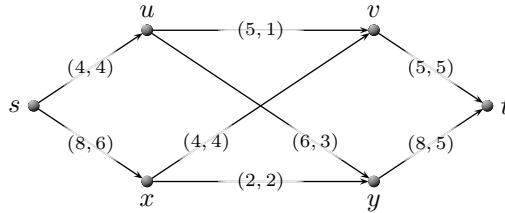


Figure 7.3: An example of a network $N_{7.3}$ with a maximum flow and a minimum cut.

Corollary 7.3 *Let f be a flow in a network N and let (S, \bar{S}) be a cut of N . Then,*

$$f(N) \leq c(S, \bar{S}).$$

Proof By [Theorem 7.1](#), the value of the flow $f(N)$ equals the net flow out of S . Thus, $f(N) = f(S, \bar{S}) - f(\bar{S}, S) \leq f(S, \bar{S})$ since $f(\bar{S}, S) \geq 0$. By the capacity constraint, however, $f(S, \bar{S}) \leq c(S, \bar{S})$, whence $f(N) \leq c(S, \bar{S})$. ■

Consider again the network flow f in [Figure 7.3](#) as an example. If $S = \{s, x\}$, then (S, \bar{S}) is a cut in $N_{7.3}$ and $10 = f(N_{7.3}) \leq c(S, \bar{S}) = c(s, u) + c(x, v) + c(x, y) = 4 + 4 + 2 = 10$.

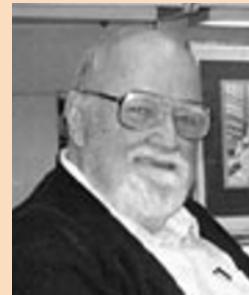
Corollary 7.4 *Let f be a flow in a network N and let (S, \bar{S}) be a cut of N . If $f(N) = c(S, \bar{S})$, then f is a maximum flow and (S, \bar{S}) is a minimum cut.*

Proof Let f^* be a maximum flow in N , and let (X, \bar{X}) be a minimum cut of N . By Corollary 7.3, $f^*(N) \leq c(X, \bar{X})$. However, f^* is a maximum flow, and so $f(N) \leq f^*(N)$, while (X, \bar{X}) is a minimum cut, and so $c(X, \bar{X}) \leq c(S, \bar{S})$. Thus, $c(S, \bar{S}) = f(N) \leq f^*(N) \leq c(X, \bar{X}) \leq c(S, \bar{S})$. We must therefore have equality throughout this inequality chain. In particular, $f(N) = f^*(N)$ and $c(X, \bar{X}) = c(S, \bar{S})$. Hence, f is a maximum flow and (S, \bar{S}) is a minimum cut. ■

Thus, by Corollary 7.4, the flow $f(N_{7.3})$ in Figure 7.3 is a maximum flow and (S, \bar{S}) is a minimum cut, where $S = \{s, x\}$.

❖ The reader should now be able to attempt Exercises 7.1–7.3.

Lester Randolph Ford Jr, an American mathematician, was born in Houston, Texas on 23 September 1927. He specialised in network flow problems and is not to be confused with his father, Lester R Ford Sr (1886–1967), who was also a well-known mathematician. Ford Junior's paper with [Delbert Ray Fulkerson](#) on the maximum flow problem contained the well-known Ford-Fulkerson algorithm for solving this problem (a fore-runner of the faster augmenting path algorithm considered in this chapter) and established what is today known as the [max-flow min-cut theorem](#) (Theorem 7.5). The algorithm was first published as a technical report in 1954 and later in a journal in 1956. Ford Jr also developed the well-known Bellman-Ford algorithm for finding shortest paths in graphs that have negatively weighted edges, before Bellman. He died on 26 February 2017.



Biographic note 20: Lester Ford Jr (1927–2017)

7.3 The max-flow min-cut theorem

It follows from Corollary 7.3 that the total value of a flow in a network is never larger than the smallest capacity of a cut. Our aim in this section is to present the so-called **max-flow min-cut** theorem due to Ford and Fulkerson in 1956 [7] which shows that this upper bound is always attained by some flow. The proof that we present is algorithmic in nature and is based on the idea of improving the current flow along some s - t semipath that is not being used “optimally.” Once this is done, we repeat the process in the network with its modified flow until we can find no such s - t semipath whose flow can be improved. This technique has come to be known as the **augmenting path technique**.

Let N be a network with source s , sink t , capacity function c and flow f . Let P be an s - v semipath. If every forward arc e on P has excess capacity (meaning $f(e) < c(e)$) and every backward arc on P has nonzero flow (meaning $f(e) > 0$), then we call P an **f -unsaturated s - v semipath**. If $v = t$, then we call P an **f -augmenting path** (even though it may not actually be a path since some arcs on P may be backward arcs). For each arc e of P , we define $\epsilon_N(e)$ (or simply $\epsilon(e)$

Delbert Ray Fulkerson, an American mathematician, was born on August 14th, 1924. He enrolled as an undergraduate at Southern Illinois University, but his study career was interrupted by military service during World War II. After the war he returned to his studies and went on to obtain a doctorate in mathematics at the University of Wisconsin at Madison in 1951. Thereafter, he joined the mathematics department at the RAND Corporation where he remained until 1971 when he moved to Cornell University to take up a position as Maxwell Upson Professor of Engineering. He remained at Cornell until his untimely death on January 10th, 1976. He is today best remembered for his co-development of the Ford-Fulkerson algorithm, one of the best-known algorithms for solving the maximum flow problem in networks — a forerunner of the augmenting path algorithm ([Algorithm 18](#)). In 1979, the prestigious Fulkerson Prize was established in his name which is awarded every three years for outstanding papers in discrete mathematics jointly by the Mathematical Programming Society and the American Mathematical Society.



Biographic note 21: Delbert Fulkerson (1924–1976)

if the network N is clear from the context) to be $c(e) - f(e)$ if e is a forward arc, or $f(e)$ if e is a backward arc. We define the **leeway** of P as $\epsilon(P) = \min\{\epsilon(e)\}$, where the minimum is taken over all arcs e on P . Intuitively, if a forward arc e on P has excess capacity, then we can increase the flow along e , while if a backward arc e on P has nonzero flow, then we can “push back” flow along the arc e .

For example, if f is the flow shown for the network $N_{7.2}$ in [Figure 7.2](#), then $P : s, (s, x), x, (x, v), v, (u, v), u, (u, y), y, (y, t), t$ is an f -augmenting path. Furthermore, $\epsilon(s, x) = 1$, $\epsilon(x, v) = 2$, $\epsilon(u, v) = 1$, $\epsilon(u, y) = 3$, and $\epsilon(y, t) = 3$. Thus, $\epsilon(P) = 1$.

We are now in a position to present the **max-flow min-cut theorem**.

Theorem 7.5 ([7]) *In every network, the value of a maximum flow equals the capacity of some minimum cut.*

Proof Let N be a network with underlying digraph D , source s , sink t , and capacity function c . We define a sequence $f^{(0)}(N), f^{(1)}(N), \dots$ of flows in N of strictly increasing value, i.e. with $f^{(0)}(N) < f^{(1)}(N) < f^{(2)}(N) < \dots$, as follows. We start with the zero flow, and so $f^{(0)}(e) = 0$ for all arcs e of D and $f^{(0)}(N) = 0$. For each flow $f^{(i)}$, $i \geq 0$, we denote by S_i the set of all vertices v such that there is an $f^{(i)}$ -unsaturated s - v semipath.

Suppose $t \in S_i$. Then there is an f -augmenting path P in N . Let $f^{(i+1)}$ be the function obtained from $f^{(i)}$ by increasing flow by $\epsilon(P)$ along forward arcs of P and decreasing flow by $\epsilon(P)$ along backward arcs of P while leaving flows on all remaining arcs unchanged. By the definition of the leeway $\epsilon(P)$, we have $0 \leq f^{(i+1)}(e) \leq c(e)$ for every arc e , and so the capacity constraint holds. To verify the conservation constraint, we need only consider vertices of P since the flow along all arcs not on P is unchanged. The net flow out of each vertex v on P different from s and t , however, remains 0 (irrespective of the direction of the two

arcs incident with v on P). Hence, $f^{(i+1)}$ is indeed a flow in N . Furthermore, the net flow out of the source s is $\epsilon(P)$ larger in $f^{(i+1)}(N)$ than in $f^{(i)}(N)$, and so $f^{(i+1)}(N) = f^{(i)}(N) + \epsilon(P)$. Thus, $f^{(i+1)}(N) > f^{(i)}(N)$, as desired.

Since a flow is an integer valued function on $E(D)$, $f^{(i)}(N) + 1 \leq f^{(i+1)}(N)$ for all i . By Corollary 7.3, the value of any flow in N is bounded from above by the capacity of any cut in N , and so our sequence $f^{(0)}(N), f^{(1)}(N), \dots$ of flows in N will terminate with some flow $f^{(n)}(N)$. Hence in $f^{(n)}(N)$, $t \notin S_n$. Let $S = S_n$. We now consider the cut (S, \bar{S}) . Suppose $(u, v) \in (S, \bar{S})$ and $f(u, v) < c(u, v)$. By taking an $f^{(n)}$ -unsaturated $s-u$ semipath, and then proceeding to v along the arc (u, v) , we produce an $f^{(n)}$ -unsaturated $s-v$ semipath, contradicting the fact that $v \in \bar{S}$. Hence if $(u, v) \in (S, \bar{S})$, then $f(u, v) = c(u, v)$. Thus, $f(S, \bar{S}) = c(S, \bar{S})$. Furthermore, if $(v, u) \in (\bar{S}, S)$, then $f(v, u) = 0$, for otherwise (if $f(v, u) > 0$) we could find an $f^{(n)}$ -unsaturated $s-v$ semipath, a contradiction. Hence, $f(\bar{S}, S) = 0$. Consequently, by Theorem 7.1,

$$f^{(n)}(N) = f^{(n)}(S, \bar{S}) - f^{(n)}(\bar{S}, S) = c(S, \bar{S}).$$

Thus, by Corollary 7.4, $f^{(n)}$ is a maximum flow and (S, \bar{S}) is a minimum cut. ■

❖ The reader should now be able to attempt Exercises 7.4–7.5.

7.4 The max-flow min-cut algorithm

As remarked earlier, the proof of Theorem 7.5 is essentially algorithmic in nature, since it involves searching for an augmenting path to increase the flow value. If such a path is not found, then the proof produces a minimum cut and maximum flow.

In this section, we present an algorithm due to Edmonds and Karp [6], given in pseudocode as Algorithm 18, which is a systematic method for finding an f -augmenting path in a network N with flow f , if such a path exists. We begin the procedure with a given flow f in N (perhaps the zero flow). As we attempt to find an f -augmenting path, we label the vertices of N as we examine them. Initially, we label the source s , and then we label every vertex v for which we can find an f -unsaturated $s-v$ semipath. The label assigned to v is an ordered pair. If u is the vertex immediately preceding v on P , then the first component of the label is u^+ or u^- depending on whether the arc preceding v on P is a forward arc (u, v) or a backward arc (v, u) . The second component of the label is a positive integer reflecting the potential change in f along P . When we finally label the sink t , we have found an f -augmenting path. This path is then used to increase the total flow in N , and the process is repeated. If at some point, we cannot find any vertex to label, then no f -augmenting path exists and, as shown in the proof of Theorem 7.5, the present value of the flow is a maximum.

Algorithm 18 terminates with a maximum flow f in N . Furthermore, if S is the set of labelled vertices upon termination, then (S, \bar{S}) is a minimum cut.

As an example to illustrate the working of Algorithm 18, consider the network $N_{7.1}$ in Figure 7.1. It has been reproduced in Figure 7.4 as $N_{7.4}$ where the labels on each arc e are the capacity $c(e)$ of e and the flow $f(e)$ in e , respectively. We begin with the zero flow shown in Figure 7.4(a).

Initially, s is labelled $(-, \infty)$, and L consists only of s . As the algorithm proceeds through Step 3 for the first time, we remove s from L . Since u is unla-

Richard Manning Karp, an American computer scientist and computational theorist, was born on January 3rd, 1935. He obtained a bachelor's degree in 1955, a master's degree in 1956, and a doctorate in applied mathematics in 1959, all from Harvard University. In 1959, he started working at IBM's Thomas J. Watson Research Center and in 1968, he was appointed as professor of computer science, mathematics, and operations research at the University of California, Berkeley. Apart from a 4-year period as a professor at the University of Washington, he remained at Berkeley, where he led the Algorithms Group within the International Computer Science Institute. In 2012, Karp became the founding director of the Simons Institute for the Theory of Computing at the University of California, Berkeley. He has made many important contributions in the fields of computer science, operations research (within the area of combinatorial algorithms) and bioinformatics. In 1971, he co-developed the [augmenting path algorithm](#) ([Algorithm 18](#)) for solving the maximum flow problem on networks with Jack Edmonds, and in 1972 he published a landmark paper in complexity theory in which he proved 21 problems to be NP-complete. For his work over many years he received the prestigious Turing Award in 1985, The Benjamin Franklin Medal in Computer and Cognitive Science in 2004, and the Kyoto Prize in 2008.



Biographic note 22: Richard Karp (1935–present)

belled, and $(s, u) \in E$ satisfies $f(s, u) < c(s, u)$, we assign the label $(s^+, 4)$ to the vertex u , and add u to the end of L . (Note that $\epsilon(u) = \min\{\epsilon(s), c(s, u) - f(s, u)\} = \min\{\infty, 4\} = 4$.) Furthermore, since x is unlabelled, and $(s, x) \in E$ satisfies $f(s, x) < c(s, x)$, we assign the label $(s^+, 5)$ to the vertex x , and add x to the end of L . At this stage, $L = \{u, x\}$. Since t has not been labelled, we return to [Step 3](#). We remove the first element of L , namely u . We then assign the label $(u^+, 4)$ to the vertex v , and add v to the end of L . (Note that $\epsilon(v) = \min\{\epsilon(u), c(u, v) - f(u, v)\} = \min\{4, 5\} = 4$.) We next assign to y the label $(u^+, 4)$, and add y to the end of L . At this stage, $L = \{x, v, y\}$. Since t has not been labelled, we return to [Step 3](#). We remove the first element of L , namely x . Since there are no unlabelled vertices that can be labelled from x , we continue through [Step 3](#) again (with $L = \{v, y\}$). We remove the first element of L , namely v , and then assign the label $(v^+, 3)$ to the vertex t , and add t to the end of L . (Note that $\epsilon(t) = \min\{\epsilon(v), c(v, t) - f(v, t)\} = \min\{4, 3\} = 3$.) We now reach [Step 8](#) and proceed to [Step 10](#) to obtain the f -augmenting path $P : s, (s, u), u, (u, v), v, (v, t), t$ as shown in [Figure 7.4\(b\)](#). We then increase the flow by $\epsilon(t) = 3$ along forward arcs of P and decrease the flow by $\epsilon(t) = 3$ along backward arcs of P while leaving flows on all remaining arcs unchanged. The resulting flow is shown in [Figure 7.4\(c\)](#).

Proceeding through [Step 2](#) of [Algorithm 18](#), we assign to the vertices the labels shown in [Figure 7.4\(c\)](#). Using the resulting f -augmenting path $s, (s, u), u, (u, y), y$,

Algorithm 18: [6] f -Augmenting path

Input : A network N with underlying graph $D = (V, E)$, source s , sink t and capacity function c .

Output : An f -augmenting path from s to t in N .

- 1 Assign values of an initial flow f to the arcs of D
- 2 Label s with $(-, \infty)$ and add s to L , the list of labelled and unscanned vertices
- 3 Select and remove the first element of L , say u , with label $(x^+, \epsilon(u))$ or $(x^-, \epsilon(u))$
- 4 **if** (L is empty) **then stop**
- 5 **else**
- 6 To all vertices v that are unlabelled and such that $(u, v) \in E$ and $f(u, v) < c(u, v)$, assign the label $(u^+, \epsilon(v))$, where $\epsilon(v) = \min\{\epsilon(u), c(u, v) - f(u, v)\}$, and add v to the end of L
- 7 To all vertices v that are unlabelled and such that $(v, u) \in E$ and $f(u, v) > 0$, assign the label $(u^-, \epsilon(v))$, where $\epsilon(v) = \min\{\epsilon(u), f(v, u)\}$ and add v to the end of L
- 8 **if** (t has been labelled) **then go to Step 10**
- 9 **else go to Step 3**
- 10 The labels describe an f -augmenting path $s = u_0, a_1, u_1, a_2, \dots, u_{n-1}, a_n, u_n = t$. For $i = 1, 2, \dots, n$, replace $f(a_i)$ by $f(a_i) + \epsilon(t)$ if a_i is a forward arc and by $f(a_i) - \epsilon(t)$ if a_i is a backward arc
- 11 Discard all labels, remove all vertices from L , and **go to Step 2**

$(y, t), t$ with leeway $\epsilon(t) = 1$, we increase the flow in each of the arcs on this path by $\epsilon(t) = 1$. The resulting flow f is shown in Figure 7.4(d).

Proceeding through Step 2 of Algorithm 18 again, we assign to the vertices the labels shown in Figure 7.4(d). Using the resulting f -augmenting path $s, (s, x), x, (x, y), y, (y, t), t$ with leeway $\epsilon(t) = 2$, we increase the flow in each of the arcs on this path by $\epsilon(t) = 2$. The resulting flow f is shown in Figure 7.4(e).

Proceeding through Step 2 of Algorithm 18 yet again, we assign to the vertices the labels shown in Figure 7.4(e). Using the resulting f -augmenting path $s, (s, x), x, (x, v), v, (u, v), u, (u, y), y, (y, t), t$ with leeway $\epsilon(t) = 3$, we increase the flow in each of the forward arcs in this path by $\epsilon(t) = 3$ and decrease the flow by $\epsilon(t) = 3$ along backward arcs in this path. The resulting flow f is shown in Figure 7.4(f).

Proceeding through Step 2 of Algorithm 18 one last time, we are only able to label the source s and no other vertex (including the sink t). Thus the given flow f is a maximum flow, and the corresponding minimum cut is (S, \bar{S}) , where $S = \{s\}$. The value of the flow f is $f(N_{7.4}) = f(S, \bar{S}) - f(\bar{S}, S) = 9$, while $c(S, \bar{S}) = 9$.

We remark that, when a vertex receives a label in Algorithm 18, it is added to the tail end of the “labelled but unscanned” list L . These vertices are scanned on a first-labelled-first-scanned basis, which insures that a shortest f -augmenting path is selected. The importance of taking a shortest f -augmenting path is illustrated in the network $N_{7.5}$ shown in Figure 7.5. If the f -augmenting path is taken alternately along $s, (s, x), x, (x, y), y, (y, t), t$ and $s, (s, y), y, (x, y), x, (x, t), t$, then we will need

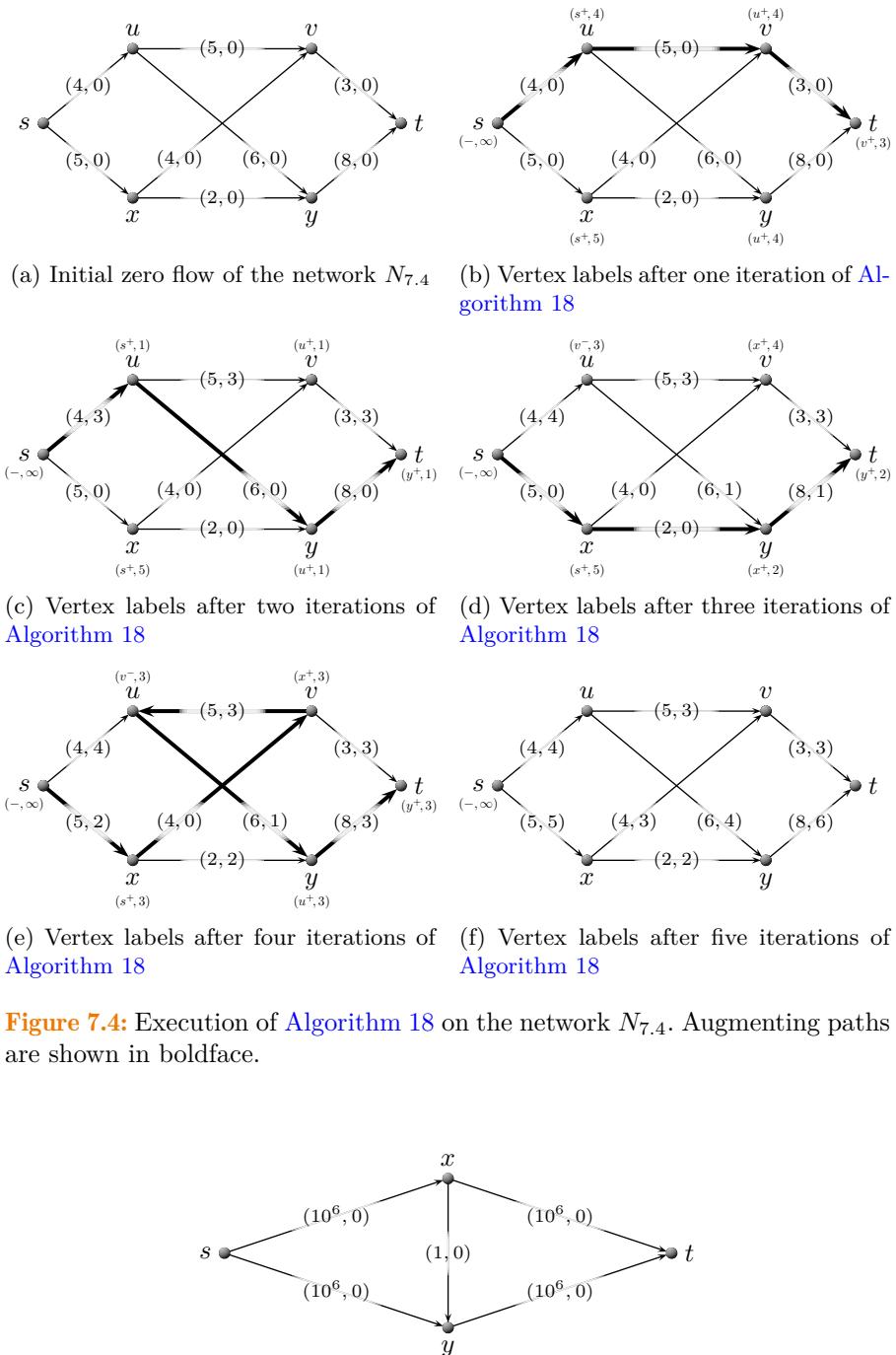


Figure 7.4: Execution of Algorithm 18 on the network $N_{7.4}$. Augmenting paths are shown in boldface.

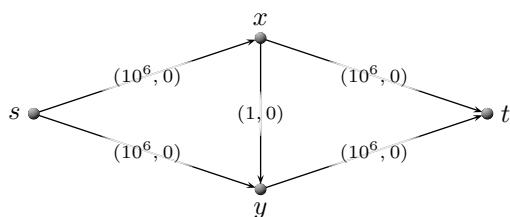


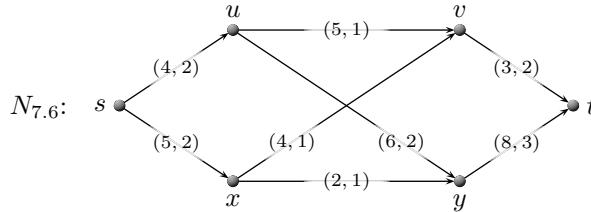
Figure 7.5: A network $N_{7.5}$ in support of the structure of Algorithm 18.

2×10^6 steps before we obtain a maximum flow of 2×10^6 , whereas [Algorithm 18](#) obtains a maximum flow in two steps.

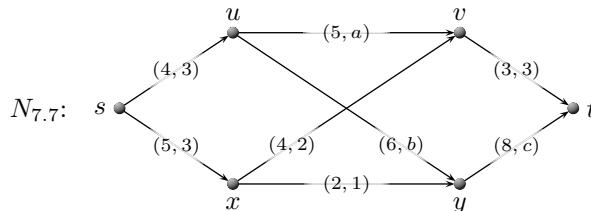
- ❖ The reader should now be able to attempt [Exercise 7.6](#).

Exercises

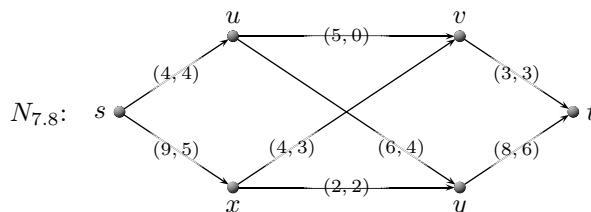
- 7.1 For the network $N_{7.6}$ shown below, an ordered pair is associated with each arc, where the first number denotes the capacity of the arc and the second number a value determined by a function f . Is f a flow? Explain.



- 7.2 For the network $N_{7.7}$ shown below, an ordered pair is associated with each arc where the first number denotes the capacity of the arc and the second number the flow in the arc. Determine the values of the flows a , b and c .



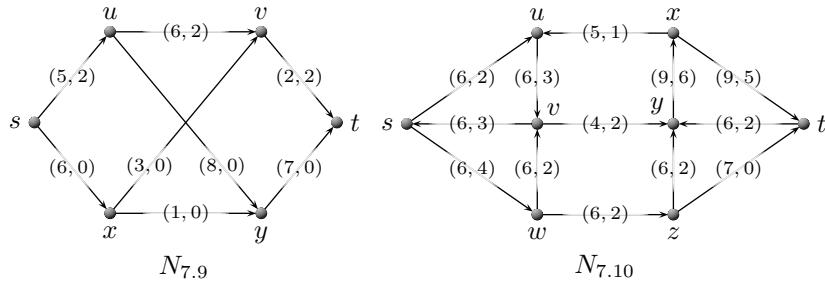
- 7.3 Consider the network $N_{7.8}$ shown below where, as before, the first number associated with an arc e is its capacity $c(e)$ and the second number its flow $f(e)$. Use [Corollary 7.4](#) to show that the given flow is a maximum flow and find the corresponding minimum cut.



- 7.4 Find an f -augmenting path P in the network flow f in [Figure 7.2](#). What is the leeway of P ?

- 7.5 Verify, by enumerating all s - t semipaths, that there is no f -augmenting path in the network flow shown in [Figure 7.3](#).

- 7.6 For the networks $N_{7.9}$ and $N_{7.10}$ shown below, an ordered pair is associated with each arc, where the first number denotes the capacity of the arc and the second number the flow in the arc. Starting with the given flow, use [Algorithm 18](#) to find a maximum flow and a minimum cut for these networks.



Computer exercises

The **MATHEMATICA** command `FindMaximumFlow[G, source, sink]` returns the value of a maximum flow through a network G from the vertex `source` to the vertex `sink`. There are also four-parameter versions of this command. The commands `FindMaximumFlow[G, source, sink, "EdgeList"]`, `FindMaximumFlow[G, source, sink, "FlowGraph"]` and `FindMaximumFlow[G, source, sink, "FlowMatrix"]` return respectively the edges of G that have positive flow in a maximum flow from the vertex `source` to the vertex `sink`, the subgraph induced by these flow edges and the adjacency matrix of the latter subgraph. For example, the commands

```
In[1]:= G = GridGraph[{2, 3, 2}, VertexLabels -> "Name"]
In[2]:= FindMaximumFlow[G, 1, 12]
In[3]:= FindMaximumFlow[G, 1, 12, "EdgeList"]
In[4]:= FindMaximumFlow[G, 1, 12, "FlowGraph"]
In[5]:= MatrixForm[FindMaximumFlow[G, 1, 12, "FlowMatrix"]]
```

produce the output:

```
Out[1]:= 6
          4
          2
          1
         12
         5
         10
         3
         8
         11
         9
         7
         1
Out[2]:= 3
Out[3]:= {1 → 2, 1 → 3, 1 → 7, 2 → 8, 3 → 5, 3 → 9, 5 → 6, 5 → 11, 6 → 12, 7 → 8, 7 → 9, 8 → 7, 8 → 10, 9 → 3, 9 → 10, 9 → 11, 10 → 9, 10 → 12, 11 → 5, 11 → 12}
Out[4]:= 6
          4
          2
          1
         12
         5
         10
         3
         8
         11
         9
         7
         1

```

The above example involved an unweighted graph (in which each edge is assumed to have a unit capacity), but the same commands also work for weighted graphs with more generally capacitated edge flows. For example, the commands

```
In[6]:= caplist = RandomInteger[{1, 8}, 31]
In[7]:= H = GridGraph[{4, 5},
    VertexLabels -> "Name",
    EdgeLabels -> "EdgeWeight",
    EdgeWeight -> caplist,
    EdgeCapacity -> caplist]
In[8]:= FindMaximumFlow[H, 1, 20]
In[9]:= FindMaximumFlow[H, 1, 20, "EdgeList"]
In[10]:= FindMaximumFlow[H, 1, 20, "FlowGraph"]
In[11]:= MatrixForm[
    FindMaximumFlow[H, 1, 20, "FlowMatrix"]]
```

may produce the output:

```

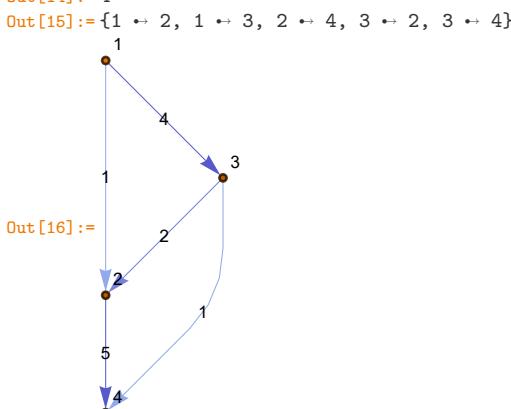
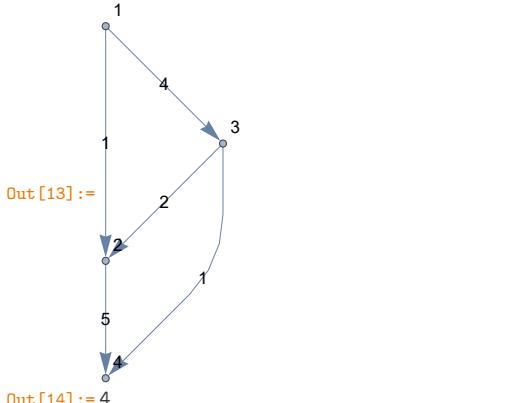
Out[6]:= {2, 1, 5, 7, 6, 4, 3, 8, 2, 7, 5, 6, 8, 4, 7, 6, 7, 6, 7, 6, 8, 8, 1, 3,
          3, 3, 1, 8, 1, 7, 3}
          4   8   12  16   20
          3   6   7   8   8
          6   3   7   11  15  19
          5   4   8   6   1   3
          2   7   6   10  14  18
          2   8   5   6   8   1
          1   5   2   9   13  17
Out[7]:= 3
Out[8]:= 3
Out[9]:= {1 ↔ 2, 1 ↔ 5, 2 ↔ 6, 5 ↔ 9, 6 ↔ 10, 9 ↔ 13, 10 ↔ 14, 13 ↔ 17, 14 ↔
           18, 17 ↔ 18, 18 ↔ 19, 19 ↔ 20}
          4   8   12  16   20
          3   6   7   11  15  19
          6   3   7   10  14  18
          5   4   8   6   8   1
          2   7   6   9   13  17
Out[10]:= 5
          4   8   12  16   20
          3   6   7   11  15  19
          6   3   7   10  14  18
          5   7   6   10  14  18
          2   8   5   6   8   1
          2   8   5   6   8   1
          1   5   2   9   13  17

```

The commands used above also work for directed graphs with generally capacitated arc flows. For example, the commands

```
In[12]:= caplist = {1, 4, 2, 1, 5};  
In[13]:= J = Graph[{1 -> 2, 1 -> 3, 3 -> 2, 3 -> 4, 2 -> 4}, VertexLabels ->  
          "Name", EdgeLabels -> "EdgeWeight", EdgeWeight -> caplist,  
          EdgeCapacity -> caplist]  
In[14]:= FindMaximumFlow[J, 1, 4]  
In[15]:= FindMaximumFlow[J, 1, 4, "EdgeList"]  
In[16]:= FindMaximumFlow[J, 1, 4, "FlowGraph"]  
In[17]:= MatrixForm[FindMaximumFlow[J, 1, 4, "FlowMatrix"]]
```

produce the output:



```
Out[17]:= 
$$\begin{pmatrix} 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

```

Projects

This section contains two projects. In the first project, the reader is guided in step-by-step fashion to apply [Algorithm 18](#) to the real-world situation of computing the maximum flow capacity of the South African fresh fruit export supply chain, while in the second project we demonstrate how the notion of maximum flow may be applied in the unlikely situation of optimally assigning dinner seating for members of a number of families.

Project 7.1: South African fruit export

Fruit is grown in 31 main regions within South Africa. These regions are listed in [Table 7.1](#) and their locations are shown in [Figure 7.6](#). Upon being picked, the fruit is transported to regional pack houses where it is washed and packed into boxes. These boxes are then loaded onto wooden pallets (which are easily moved by means of a fork lift). The fruit remains on these pallets for the remainder of the fresh fruit export process from South Africa to various foreign markets. After being packed onto pallets at pack houses, the fruit is transported to regional cold stores where it is refrigerated and stored until road transport is available to one of the four export ports (Cape Town, Gqeberha, Durban and Maputo). The capacities (in pallets of fruit processed per week) in regional pack houses and regional cold stores are also shown in [Table 7.1](#).

Fruit is transported by freight vehicles from the regional cold stores to intermediate port storage facilities just outside the ports (in the cases of Cape Town, Gqeberha and Durban), or directly into the port (in the case of Maputo). From here the fruit is loaded onto export vessels in the ports and transported to foreign markets in North America, Europe and the Far East. The capacities of the cold storage facilities in and outside the four ports are given in [Table 7.2](#).

Tasks

1. Extend the basic idea, shown only in part in [Figure 7.7](#), so as to construct a full directed graph model of the export infrastructure described above, consisting of four classes of vertices, namely regional pack houses (denoted by \circ in [Figure 7.7](#)), regional cold stores (denoted by \blacksquare), storage facilities outside the ports of Cape Town, Gqeberha and Durban (denoted by \bullet) and ports (denoted by $\bullet\circ$). Then add a generic source vertex (representing the picking process by which fruit enters the export supply chain) as well as a generic sink vertex (representing all foreign markets). Note that while the capacities of the fresh fruit export infrastructure are essentially associated with the vertices of the graph in [Figure 7.7](#), the capacity of a vertex v may be transferred to all the arcs entering v so as to transform the maximum flow problem of this project to the form considered in this chapter.
2. Use [Algorithm 18](#) to determine the maximum rate at which the export infrastructure can be utilised to export fruit from the source to the sink in your model in [Task 1](#) above.

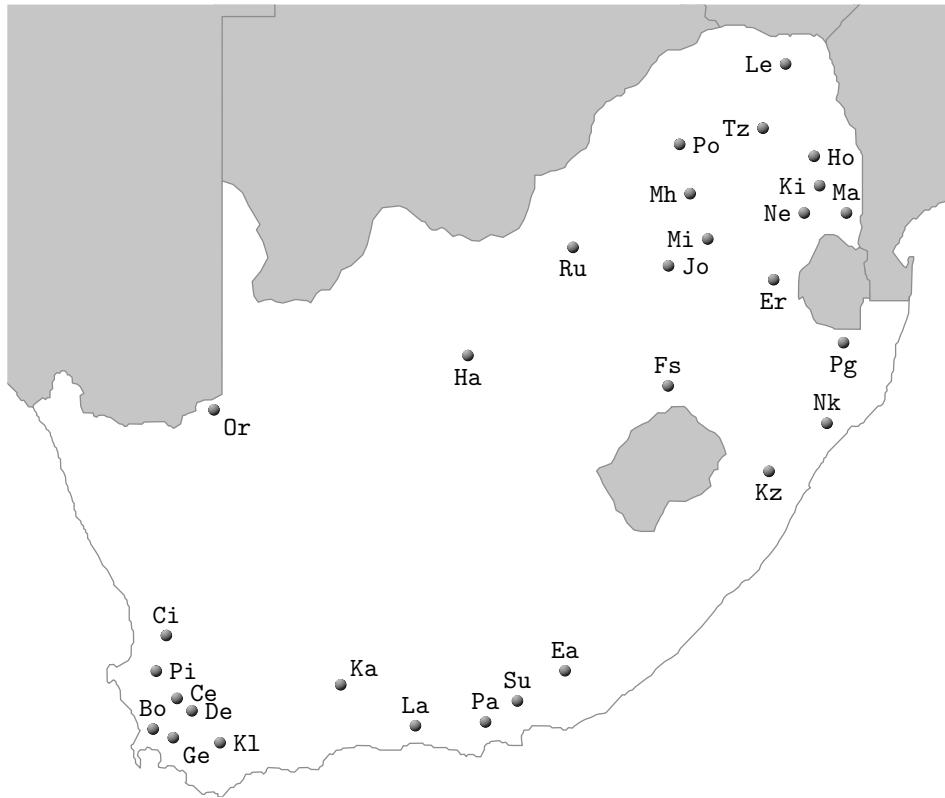


Figure 7.6: Regions in South Africa where fresh fruit is grown for export.

3. Determine the bottleneck(s) in the export infrastructure (a vertex or vertices in your graph model which, if enlarged in terms of capacity, would allow more fruit to be exported than is possible in your answer in [Task 2](#) above).

Project 7.2: Dining problem

In their interesting paper, [Bhadury et al. \[3\]](#) describe a seating arrangement problem, called the *dining problem*, which is an instance of a class of problems related to equitable partitioning. This class of problems also finds applications in areas as diverse as political districting [9], crime-based districting [10] and highway patrol districting [1].

The dining problem may be described as follows in general. Suppose there are n families attending a dinner, that there are a_i members of family $i \in [n]$, and that there are $\alpha = \sum_{i=1}^n a_i$ guests in total in these families. Suppose further that there are m dinner tables and that the seating capacity of table $j \in [m]$ is c_j . In order to maximise social interaction, the families wish to design a seating arrangement ensuring that there are at most k members of any family present at each table. Here the parameter k (≥ 1) is called the *diversity index*. Different kinds of questions may be considered in the dining problem, such as:

1. Given fixed values of n , m , a_1, \dots, a_n and c_1, \dots, c_m , and assuming that there is sufficient seating capacity in the dinner hall to accommodate all the

	Region	Code	Pack house capacity	Cold store capacity
1	Levubu	Le	11 510	1 006
2	Tzaneen	Tz	17 258	2 886
3	Hoedspruit	Ho	4 075	2 139
4	Kiepersol	Ki	1 924	2 644
5	Nelspruit	Ne	4 405	891
6	Malelane	Ma	7 151	1 776
7	Potgietersrus	Po	2 365	2 365
8	Ellisras	El	770	770
9	Marble Hall	Mh	9 365	1 670
10	Ermelo	Er	400	400
11	Pongola	Pg	1 454	1 454
12	Nkwalini	Nk	7 445	25
13	KwaZulu-Natal	Kz	1 294	1 190
14	Eastern Cape Midlands	Ea	688	460
15	Sundays River Valley	Su	10 351	3 000
16	Patensie	Pa	6 559	2 338
17	Langkloof	La	11 748	9 187
18	Karoo	Ka	366	366
19	Klein Karoo	Kl	2 571	483
20	GEVV	Ge	19 436	28 299
21	Boland	Bo	5 902	6 527
22	Ceres	Ce	15 451	9 158
23	De Doorns	De	13 904	13 904
24	Piketberg	Pi	2 428	4 550
25	Citrusdal	Ci	6 428	300
26	Orange River	Or	135	80 026
27	Hartswater	Ha	1 557	248
28	Johannesburg	Jo	310	2 296
29	Rustenburg	Ru	58	36
30	Free State	Fs	783	1 180
31	Middelburg	Mi	9 800	9 800

Table 7.1: The capacities of pack houses and cold stores (pallets per week).

guests, *i.e.* that $\alpha \leq \sum_{j=1}^m c_j$, what is the smallest value of the diversity index k for which all guests can be seated?

2. Given fixed values of $n, m, a_1, \dots, a_n, c_1, \dots, c_m$ and k , what is the largest number of guests that can be seated?
3. Given fixed values of n, a_1, \dots, a_n, k , and assuming that $c_j = c$ for all $j \in [m]$, where c is specified, what is the smallest number of tables m at which all guests can be seated?

These questions may, perhaps surprisingly, be answered by modelling the dining problem as a maximum flow problem. This may be achieved by considering the graph in Figure 7.8 in which the vertices F_1, \dots, F_n represent the families, the vertices T_1, \dots, T_m represent the tables, the vertex s represents a virtual source and the vertex t represents a virtual sink. Let us call all arcs incident from the

Port	Cold store 1	Cold store 2	Total capacity
Cape Town	56 933	77 230	134 163
Gqeberha	26 492	62 020	88 512
Durban	68 482	116 660	185 142
Maputo	—	—	26 492

Table 7.2: Storage capacities (in pallets per week) at the four ports.

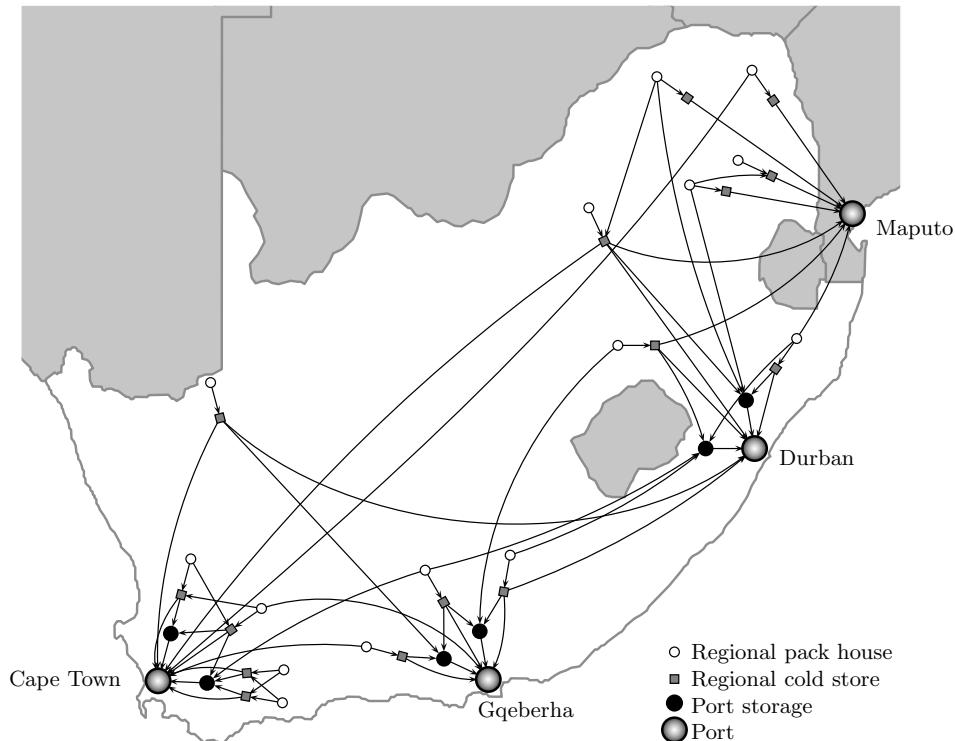


Figure 7.7: Digraph model structure for fresh fruit export from South Africa.

source *family arcs*, all arcs of the form (F_i, T_j) for some $i \in [n]$ and some $j \in [m]$ seating *arcs*, and all arcs incident to the sink *table arcs*.

If the family arc incident to F_i is assigned a capacity a_i for all $i \in [n]$, if the table arc incident to T_j is assigned a capacity c_j for all $j \in [m]$, and if all seating arcs are assigned a capacity k , then the maximum flow from s to t in the network of Figure 7.8 represents the largest number of guests that can be seated in the dining problem.

Suppose, for the remainder of this project, that each family consists of the nationals of a country, and that there are fourteen such countries. Therefore, the families may each be considered very large.

Tasks

1. Use Algorithm 18 and the modelling approach described above to show that if there are 15 dinner tables, each with a capacity of 6 guests, then it is possible

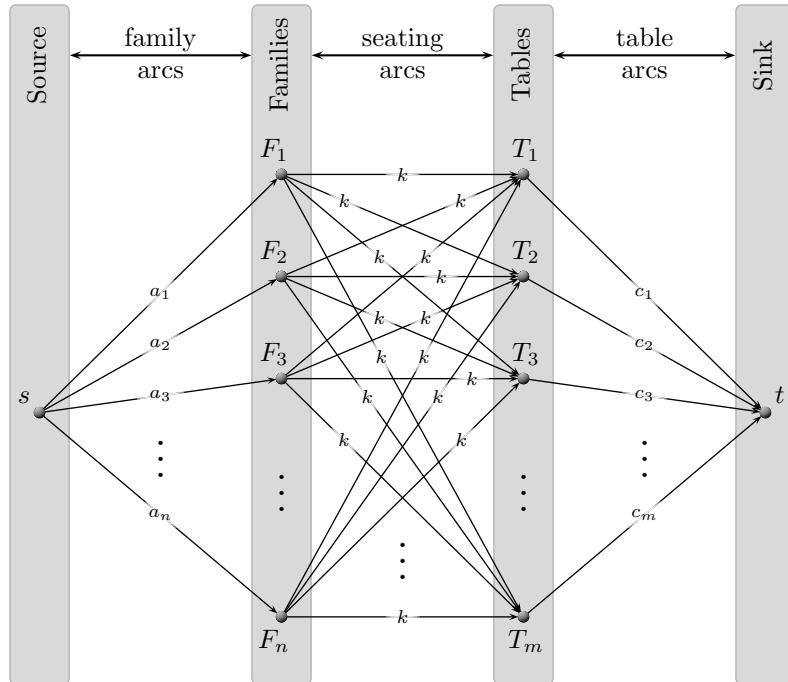


Figure 7.8: Modelling the dining problem as a maximum flow problem.

to fill all the tables to capacity subject to the condition that no table may contain more than one guest from the same country (*i.e.* the diversity index is $k = 1$).

2. Use [Algorithm 18](#) to show that if there are 10 dinner tables, each with a capacity of 9 guests, then at most 80 guests can be seated for dinner if no table may contain more than one guest from the same country.
3. Suppose again that there are 10 dinner tables, each with a capacity of 9 guests, but that no table may now contain more than *two* guests from the same country (*i.e.* the diversity index is $k = 2$). At most how many guests can be seated in total?
4. Finally, suppose that there are 6 dinner tables, each with a capacity of 15 guests. What is the smallest value of the diversity index k for which all the tables can be filled to capacity?

Further reading

- [1] A Agnetis, A Pacifici and PB Mirchandani, 2002. *Partitioning of bi-weighted trees*, Naval Research Logistics, **49**(2), pp. 143–158.
- [2] RK Ahuja, TL Magnanti and JB Orlin, 1993. *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs (NJ).
- [3] J Bhadury, EJ Mighty and H Damar, 2000. *Maximizing workforce diversity in project teams: A network flow approach*, Omega, **28**, pp. 143–153.

- [4] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY).
- [5] R Diestel, 1997. *Graph Theory*, Springer, New York (NY).
- [6] J Edmonds and RM Karp, 1972. *Theoretical improvements in algorithm efficiency for network flow problems*, Journal of the Association for Computing Machinery, **19(2)**, pp. 218–264.
- [7] LR Ford and DR Fulkerson, 1962. *Flows in Networks*, Princeton University Press, Princeton (NJ).
- [8] A Frank, 1995. *Connectivity and network flows*, pp. 111–178 in RL Graham, M Grötschel and L Lovász (Eds), *Handbook of Combinatorics (Vol 1)*, MIT Press, Cambridge (MA).
- [9] A Mehrotra, EL Johnson and GL Nemhauser, 1998. *An optimization based heuristic for political districting*, Management Science, **4(8)**, pp. 1100–1114.
- [10] A Sarac, R Batta, J Bhadury and CM Rump, 1999. *Reconfiguring the police reporting districts in the city of Buffalo*, OR Insight, **12(3)**, pp. 16–24.
- [11] DB West, 1996. *Introduction to Graph Theory*, Prentice-Hall, Upper Saddle River (NJ).



Minimum-cost network flows

Contents

8.1	Introduction	193
8.2	Network flow and linear programming theory	195
8.3	Basic feasible solutions	198
8.4	The network simplex algorithm	202
	Exercises	211
	Computer exercises	212
	Projects	214
	Further reading	219

8.1 Introduction

In Chapter 7 we introduced the notion of network flows and presented a method for finding a maximum flow in a network. In this chapter, we shift our attention to finding a minimum-cost flow in a network. We do, however, assume that the reader is familiar with fundamental concepts from linear algebra and linear programming. In order to touch up on their background in these two fields, the reader may consult any of the references [1, 2, 5, 8, 10, 12] on linear algebra and any of the references [3, 4, 6, 7, 9, 11, 13] on linear programming.

Minimum-cost network flows are distinguished by supply vertices, demand vertices, and flows on the edges of the underlying directed graph. Many practical problems may be modelled as minimum-cost flow problems. Suppose, for example, a manufacturer has a number of warehouses (supply points), each of which can supply a particular commodity to a number of retail outlets (demand points). Each warehouse has a fixed supply rate and each retail outlet has a fixed demand rate to be met. There is a transportation cost involved in sending units of the commodity from each warehouse to each retail outlet, with the cost of transporting the commodity being proportional to the distance transported and the number of units of the commodity transported. To which retail outlets should the products of each warehouse be sent so that the demand requirements of all the retail outlets are met and the total cost of transportation is a minimum? This type of problem is often referred to as the *transportation problem*. The transportation problem is an example of a minimum-cost flow problem.

As a specific example of a transportation problem, suppose a timber company has three warehouses, W_1 , W_2 and W_3 , that can supply 12, 14 and 10 tons of timber per month, respectively. The demands for the timber at four retail outlets, R_1 , R_2 , R_3 and R_4 , are 9, 8, 11 and 8 tons of timber per month, respectively. The costs (measured in thousands of dollars) of transporting one ton of timber from each warehouse to each retail outlet are given in the cost matrix

$$\begin{array}{c} R_1 \quad R_2 \quad R_3 \quad R_4 \\ W_1 \left[\begin{array}{cccc} 5 & 6 & 7 & 5 \\ 9 & 10 & 9 & 6 \\ 3 & 4 & 4 & 2 \end{array} \right] \\ W_2 \\ W_3 \end{array}$$

The problem is to determine how many tons of timber each warehouse should transport to each retail outlet so that the demand requirements of all the retail outlets are met and the total monthly cost of transportation is a minimum.

The possible flows of commodities may be modelled by means of the bipartite graph shown in [Figure 8.1](#). The warehouses are represented by the vertices labelled W_1 , W_2 and W_3 , which we call the supply vertices, while the retail outlets are represented by the vertices labelled R_1 , R_2 , R_3 and R_4 , which we call the demand vertices. The boxed numbers associated with each vertex is the available supply (in tons per month) of a warehouse (if positive) or the required demand (in tons per month) of a retail outlet (if negative).

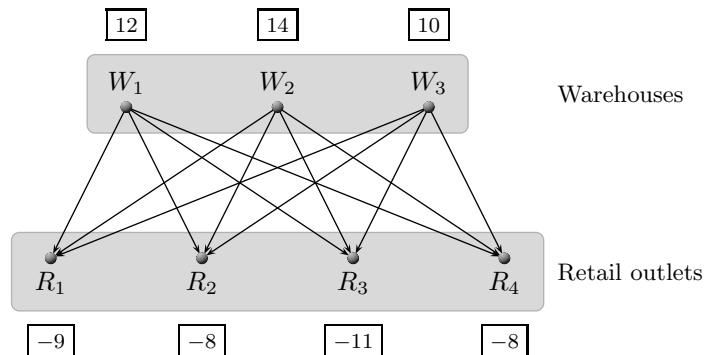


Figure 8.1: A graph representation of the timber distribution from three warehouses to four retail outlets.

We interpret the transportation of one ton of timber per month from a warehouse to a retail outlet as being a “flow” of one (commodity) unit between the corresponding vertices of the bipartite graph representation.

One transportation schedule that meets the demand requirements of all the retail outlets is to transport 9 tons of timber per month from W_1 to R_1 , 3 tons per month from W_1 to R_2 , 5 tons per month from W_2 to R_2 , 9 tons per month from W_2 to R_3 , 2 tons per month from W_3 to R_3 , and 8 tons per month from W_3 to R_4 . That is, we send a flow of 9 units along the arc (W_1, R_1) , a flow of 3 units along the arc (W_1, R_2) , a flow of 0 units along the arc (W_1, R_3) , and so on. We can calculate the total monthly transportation cost for this flow pattern by multiplying the number of units of flow along each arc by the unit cost associated with that

arc. Thus the total monthly transportation cost is

$$(9 \times 5) + (3 \times 6) + (5 \times 10) + (9 \times 9) + (2 \times 4) + (8 \times 2) = 218,$$

measured in thousands of dollars. Although this flow pattern meets the demand requirements of all the retail outlets, it is not a minimum-cost solution. Another transportation schedule that meets the demand requirements of all the retail outlets is to transport 4 tons of timber per month from W_1 to R_1 , 8 tons per month from W_1 to R_2 , 6 tons per month from W_2 to R_3 , 8 tons per month from W_2 to R_4 , 5 tons per month from W_3 to R_1 , and 5 tons per month from W_3 to R_3 . The total monthly transportation cost for this flow pattern is

$$(4 \times 5) + (8 \times 6) + (6 \times 9) + (8 \times 6) + (5 \times 3) + (5 \times 4) = 205,$$

again measured in thousands of dollars. Is this flow pattern a minimum-cost solution? Our aim in this chapter is to develop a method for solving such minimum-cost network flow problems.

❖ The reader should now be able to attempt [Exercise 8.1](#).

8.2 Network flow and linear programming theory

A minimum-cost network flow problem may be formulated as a linear programming problem. Consider a digraph $D = (V, E)$ with vertex set $V = [n]$ and arc set E of size m . With each vertex $i \in V$, we associate a number b_i representing the available supply of the vertex (if $b_i > 0$) or the required demand of the vertex (if $b_i < 0$). We call vertices for which $b_i > 0$ the **supply vertices** or **sources**, while vertices for which $b_i < 0$ we call the **demand vertices** or **sinks**. We assume that

$$\sum_{i=1}^n b_i = 0.$$

That is, we assume the total supply equals the total demand within the network. Let x_{ij} be a decision variable denoting the amount of flow on an arc (i, j) in D and let c_{ij} be the unit transportation cost along (i, j) . If there is no arc from v_i to v_j , then we define $x_{ij} = 0$ and $c_{ij} = 0$. In particular, we note that $c_{ii} = 0$ for all $i \in [n]$. We wish to solve the following linear programming problem:

$$\left. \begin{array}{ll} \text{Minimise} & \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\ \text{subject to} & \sum_{j=1}^n x_{ij} - \sum_{j=1}^n x_{ji} = b_i \text{ for all } i \in [n], \\ & x_{ij} \geq 0 \text{ for all } i, j \text{ where } 1 \leq i \leq j \leq n. \end{array} \right\} \quad (8.1)$$

Returning to our notation introduced in [Chapter 7](#) for maximum flow networks, we recall that for a vertex v and a flow f in a network, $f^+(v)$ denotes the total flow on edges exiting v and $f^-(v)$ denotes the total flow on edges entering v , while $f^+(v) - f^-(v)$ denotes the net flow out of v . Hence, for each vertex $i \in V$,

$$\sum_{j=1}^n x_{ij} = f^+(i) \quad \text{and} \quad \sum_{j=1}^n x_{ji} = f^-(i).$$

Thus, the nontrivial constraint in the linear programming problem (8.1), namely the requirement that

$$\sum_{j=1}^n x_{ij} - \sum_{j=1}^n x_{ji} = b_i,$$

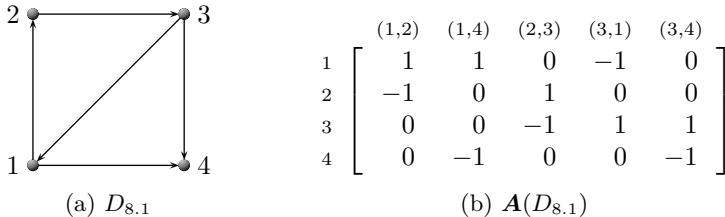
dictates that the net flow $f^+(i) - f^-(i)$ out of vertex i should equal b_i . Such a constraint is called the **flow conservation constraint** and indicates that the net flow out of each vertex is a constant. Recall that we also encountered this type of constraint in [Chapter 7](#).

The constraint in the linear program problem (8.1) that $x_{ij} \geq 0$ for all i, j where $1 \leq i \leq j \leq n$ is called the **domain constraint** or the **nonnegativity constraint** and stipulates that the flow along each arc should be nonnegative.

Some minimum-cost network flow problems also have a so-called **capacity constraint** on every arc of the form $x_{ij} \leq u_{ij}$ for all i, j where $1 \leq i \leq j \leq n$. Such a capacity constraint specifies that the flow along each arc cannot exceed a certain capacity.

We now consider the matrix form of the above linear programming problem. Let $\mathbf{A} = \mathbf{A}(D)$ be the coefficient matrix associated with the constraint set in the linear program problem (8.1). The matrix \mathbf{A} has one row for each vertex of the network and one column for each arc. Thus, \mathbf{A} is an $n \times m$ matrix. The column associated with an arc (i, j) contains a “+1” in row i , a “−1” in row j , and zeros elsewhere. The matrix \mathbf{A} is called the **vertex-arc incidence matrix** of the digraph D .

[Figure 8.2](#) shows a digraph $D_{8.1}$ of order $n = 4$ and size $m = 5$, and the vertex-arc incidence matrix $\mathbf{A}(D_{8.1})$ of the digraph $D_{8.1}$.



[Figure 8.2](#): A digraph $D_{8.1}$ and its vertex-arc incidence matrix $\mathbf{A}(D_{8.1})$.

Writing the linear program problem in (8.1) in matrix form, we are required to

$$\begin{aligned} & \text{minimise} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{Ax} = \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}, \end{aligned} \quad \left. \right\} \quad (8.2)$$

where \mathbf{A} is the vertex-arc incidence matrix described above, $\mathbf{x}, \mathbf{c} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$. The i -th entry of the vector $\mathbf{x} \in \mathbb{R}^m$ is the flow associated with the arc in the i -th column of \mathbf{A} , while the i -th entry of the vector $\mathbf{c} \in \mathbb{R}^m$ is the unit cost associated with that arc. The i -th entry of the vector $\mathbf{b} \in \mathbb{R}^n$ is the supply associated with the vertex i (in row i of the matrix \mathbf{A}). For the digraph $D_{8.1}$ and its associated

vertex-arc incidence matrix $\mathbf{A}(D_{8.1})$ in Figure 8.2, we have, for example, that

$$\mathbf{c} = \begin{bmatrix} c_{12} \\ c_{14} \\ c_{23} \\ c_{31} \\ c_{34} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_{12} \\ x_{14} \\ x_{23} \\ x_{31} \\ x_{34} \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

The **dual problem** of the linear programming problem in (8.2) is:

$$\left. \begin{array}{ll} \text{Maximise} & \mathbf{b}^T \mathbf{y} \\ \text{subject to} & \mathbf{A}^T \mathbf{y} \leq \mathbf{c}, \\ & \mathbf{y} \text{ unrestricted,} \end{array} \right\} \quad (8.3)$$

where $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{y}^T = [y_1, y_2, \dots, y_n]$, with y_i the dual variable associated with the i -th constraint in (8.2). Each dual variable y_i is unrestricted in sign.

Note that the column of the matrix \mathbf{A} corresponding to an arc $(i, j) \in E$ is given by $\mathbf{e}_i - \mathbf{e}_j$ where here \mathbf{e}_i denotes a unit vector in \mathbb{R}^n with a 1 in the i -th position. Thus associated with each arc $(i, j) \in E$ in (8.2), we have a constraint in the dual problem which is given by $(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{y} = y_i - y_j \leq c_{ij}$. In summary, the minimum-cost flow linear programming problem and its dual are given by:

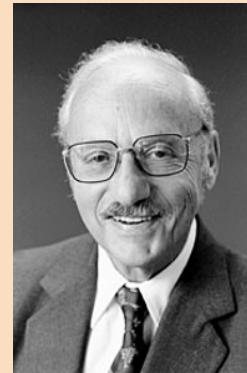
Minimum-cost flow problem	Dual problem
Minimise $\mathbf{c}^T \mathbf{x}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$,	Maximise $\mathbf{b}^T \mathbf{y}$ subject to $\mathbf{A}^T \mathbf{y} \leq \mathbf{c}$, \mathbf{y} unrestricted.

We recall the *Complementary Slackness Theorem* from linear programming theory. Let \mathbf{x}^* be a feasible solution of the primal problem (8.2) and let \mathbf{y}^* be a feasible solution of the dual (8.3). Then, \mathbf{x}^* and \mathbf{y}^* are optimal solutions to their respective problems if and only if either the k -th primal variable $x_k^* = 0$ or the slack in the k -th constraint of the dual problem is zero; that is, if $y_i^* - y_j^* = c_{ij}$ (where here the k -th constraint in the dual problem is associated with the arc $(i, j) \in E$ in the primal problem).

The linear programming problem (8.2) can be solved in any one of several ways. One way is to apply the seminal **simplex algorithm**, due to George Dantzig in 1947. Fortunately, other much more efficient algorithms exist for solving minimum-cost network flow problems. Our aim in this chapter is to present a simplification of the simplex algorithm, called the **network simplex algorithm**, which is much faster than the original, general-purpose algorithm. As we shall see, spanning trees in the network play a central role in the **network simplex algorithm** as they enable us to characterise bases of the coefficient matrix of the constraints in the linear programming problem (8.2).

- ❖ The reader should now be able to attempt Exercises 8.2–8.4.

George Bernard Dantzig was born in Portland, Oregon on 8 November 1914. He received his bachelor's degree in mathematics and physics from the University of Maryland in 1936 and his master's degree from the University of Michigan in 1938. He served as chief of the combat analysis branch of the Army Air Forces from 1941 to 1946, the year that he received his doctorate in mathematics from University of California at Berkeley. From 1946 to 1952 he was a mathematical adviser to the military and then from 1952 to 1960 held a position as research mathematician at the RAND Corporation. In 1960, he moved to the University of California at Berkeley where he was appointed professor and chair of the Operations Research Center. He moved to Stanford University in 1966, holding positions in the Departments of Operations Research and Computer Science. He was appointed the CA Criley Professor of Transportation Sciences at Stanford in 1973. His research focused on optimisation of large-scale systems and the development of energy and economic planning models, but he is today best remembered for the simplex algorithm he devised in 1947 — this algorithm has been described as one of the greatest algorithms of the twentieth century. In 1975, President Gerald Ford awarded Dantzig a National Medal of Science “for inventing Linear Programming and for discovering the Simplex Algorithm that led to wide-scale scientific and technical applications to important problems in logistics, scheduling and network optimization, and to the use of computers in making efficient use of the mathematical theory.” Many other awards were also bestowed upon him, and he received honorary doctorates from various universities. He died on 13 May 2005 at his home in Stanford.



Biographic note 23: George Dantzig (1914–2005)

8.3 Basic feasible solutions

Recall, from [Chapter 1](#), that a digraph D is *weakly connected* if the underlying graph of D is connected. We start this section by proving the following theorem.

Theorem 8.1 *If $\mathbf{A} = \mathbf{A}(D)$ is the vertex-arc incidence matrix of a weakly connected digraph D of order $n \geq 2$ and size m , then $\text{rank}(\mathbf{A}) = n - 1$.*

Proof If $\mathbf{r}_1, \dots, \mathbf{r}_n$ are the row vectors of \mathbf{A} , then $\mathbf{r}_1 + \dots + \mathbf{r}_n = \mathbf{0}$ since each column of \mathbf{A} contains exactly one “+1” and one “−1,” with zeros elsewhere. Hence the rows of \mathbf{A} are linearly dependent, and so $\text{rank}(\mathbf{A}) \leq n - 1$. To show that $\text{rank}(\mathbf{A}) \geq n - 1$, it suffices to show that there exists an $(n - 1) \times (n - 1)$ submatrix of \mathbf{A} that is invertible.

Let G be the underlying graph of D . Since the digraph D is weakly connected, the graph G is connected. Let $T = (V, E')$ be a subdigraph of D such that the underlying graph G_T of T is a spanning tree of G . Thus, T has the same vertex set as D and contains $n - 1$ arcs of D . Let $\mathbf{A}_T = \mathbf{A}(T)$ be the submatrix of \mathbf{A} associated with the vertices and arcs of T . Then, \mathbf{A}_T is the $n \times (n - 1)$ vertex-arc incidence matrix of T . Since G_T is a tree, it has at least one end-vertex (see

[Theorem 5.4](#)). Thus the digraph T has at least one vertex v_1 of degree 1; that is, at least one vertex having only one incident arc. The row of \mathbf{A}_T corresponding to such a vertex v_1 of T has only one nonzero entry (which is either “+1” or “−1”). We now permute the rows and columns of \mathbf{A}_T , if necessary, so that this nonzero entry is in the first row and first column. Thus, \mathbf{A}_T has the form

$$\mathbf{A}_T = \begin{bmatrix} \pm 1 & \mathbf{0} \\ \mathbf{p} & \mathbf{A}_{T'} \end{bmatrix},$$

where $\mathbf{p} \in \mathbb{R}^{n-1}$ and $\mathbf{A}_{T'}$ is the $(n-1) \times (n-2)$ vertex-arc incidence matrix of $T' = T - v_1$. Since the underlying graph of T' is also a tree, the digraph T' has at least one vertex v_2 having only one incident arc. Permuting the rows and columns of $\mathbf{A}_{T'}$, if necessary, so that this nonzero entry is the first row and first column of $\mathbf{A}_{T'}$, we may write \mathbf{A}_T in the form

$$\mathbf{A}_T = \begin{bmatrix} \pm 1 & 0 & \mathbf{0} \\ p & \pm 1 & \mathbf{0} \\ \mathbf{p}' & \mathbf{q}' & \mathbf{A}_{T''} \end{bmatrix},$$

where $p \in \{+1, 0, -1\}$, $\mathbf{p}', \mathbf{q}' \in \mathbb{R}^{n-2}$ and $\mathbf{A}_{T''}$ is the $(n-2) \times (n-3)$ vertex-arc incidence matrix of $T'' = T' - v_2$. Repeating this procedure $n-1$ times, and then deleting the resulting last row of \mathbf{A}_T , produces an $(n-1) \times (n-1)$ matrix $\tilde{\mathbf{B}}_T$ that is lower triangular with nonzero diagonal elements. Therefore, $\tilde{\mathbf{B}}_T$ is invertible and so $\text{rank}(\mathbf{A}) \geq n-1$. Consequently, $\text{rank}(\mathbf{A}) = n-1$. ■

To illustrate the proof of [Theorem 8.1](#), consider again the digraph $D_{8,1}$ in [Figure 8.2](#). Let $T_{8,2}$ be the spanning subdigraph of $D_{8,1}$ containing the arcs $(1,2)$, $(1,4)$ and $(2,3)$. The directed tree $T_{8,2}$ and its associated vertex-arc incidence matrix $\mathbf{A}_{T_{8,2}} = \mathbf{A}(T_{8,2})$ are shown in [Figure 8.3](#).

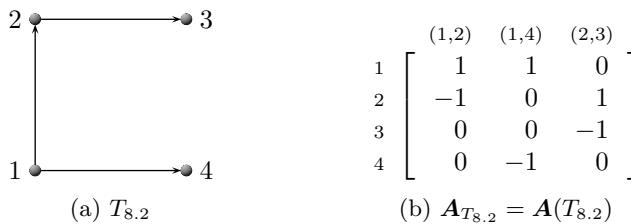


Figure 8.3: A directed tree $T_{8,2}$ and its vertex-arc incidence matrix $\mathbf{A}_{T_{8,2}}$.

Applying the procedure discussed in the proof of [Theorem 8.1](#), with end-vertices selected in the order 4, 3, and 2, and then deleting the resulting last row of $\mathbf{A}_{T_{8,2}}$, we obtain the 3×3 matrix

$$\tilde{\mathbf{B}}_{T_{8,2}} = \begin{bmatrix} (1,4) & (2,3) & (1,2) \\ 4 & -1 & 0 & 0 \\ 3 & 0 & -1 & 0 \\ 2 & 0 & 1 & -1 \end{bmatrix}.$$

Let D be a weakly connected digraph of order $n \geq 2$ and size m , and let $\mathbf{A} = \mathbf{A}(D)$ be the vertex-arc incidence matrix of D . By [Theorem 8.1](#), $\text{rank}(\mathbf{A}) = n - 1$. Let $\tilde{\mathbf{B}}_T$ be the $(n - 1) \times (n - 1)$ invertible submatrix of \mathbf{A} constructed in the proof of [Theorem 8.1](#) that is lower triangular with nonzero diagonal elements. In order to find an initial **basic feasible solution** to [\(8.2\)](#), we need a full row rank matrix (of rank n). We can transform the matrix \mathbf{A} into a full row rank matrix by adding a column consisting of all zeros and a single 1 placed in row i corresponding to the vertex i in the deleted last row of $\mathbf{A}_T = \mathbf{A}(T)$. This produces the constraint matrix $[\mathbf{A}, \mathbf{e}_i]$, where \mathbf{e}_i denotes a unit vector in \mathbb{R}^n with a 1 in the i -th position, as before. We may interpret the new added column as “half an arc” that is incident from the vertex i but is not incident to any vertex. We call this artificial arc the **root arc** and its incident vertex the **root vertex**.

By adding a root arc we have introduced an **artificial variable** corresponding to the vertex in the deleted last row of \mathbf{A}_T . Every basic feasible solution must contain n linearly independent columns, and hence this artificial variable must appear in every basic feasible solution. Furthermore, the artificial variable must be zero in every feasible solution since the original flow conservation constraints are all equality constraints.

We say that a digraph ${}^r D$ is a **rooted digraph** if it is obtained from a digraph D by adding one root arc incident from a root vertex. We call the digraph D the **underlying digraph** of ${}^r D$. Adding the root arc to a spanning subdigraph of the underlying digraph D produces a rooted subdigraph called a **rooted spanning subdigraph** of ${}^r D$. In particular, if ${}^r D$ is a weakly connected rooted digraph, then adding the root arc to a spanning directed tree of the underlying digraph D produces a rooted subdigraph called a **rooted spanning tree** of ${}^r D$.

As an example, reconsider the digraph $D_{8.1}$ of [Figure 8.2](#) and the spanning directed tree $T_{8.2}$ of $D_{8.1}$ shown in [Figure 8.3](#). Adding a root arc incident from the vertex 1 produces the rooted digraph ${}^r D_{8.1}$ and the rooted spanning tree ${}^r T_{8.2}$ shown in [Figure 8.4](#).

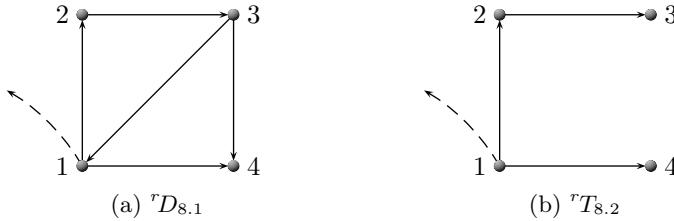


Figure 8.4: A rooted digraph and a rooted spanning tree.

The following result provides a characterisation of basis matrices for the row space of a row-arc incidence matrix of a weakly connected rooted digraph.

Theorem 8.2 *Every basis matrix for the row space of the vertex-arc incidence matrix of a weakly connected rooted digraph D is the row-arc incidence matrix of some rooted spanning tree of D . Furthermore, this basis matrix is lower triangular with nonzero diagonal elements.*

Proof Let ${}^r D$ be a weakly connected rooted digraph and let $\mathbf{A}_r = \mathbf{A}({}^r D)$ be the vertex-arc incidence matrix of ${}^r D$ with root vertex corresponding to vertex i .

Let D be the underlying digraph of rD of order $n \geq 2$ and size m , and let \mathbf{A} be the vertex-arc incidence matrix of D . Thus, $\mathbf{A}_r = [\mathbf{A}, \mathbf{e}_i]$.

We show first that the vertex-arc incidence matrix of every rooted spanning tree rT of rD is a basis for the row space of \mathbf{A} . Let rT be a rooted spanning tree of rD , and let T be the underlying directed tree of rT . Thus, T is a spanning directed tree of D . Using the notation introduced in the proof of [Theorem 8.1](#), the $n \times n$ matrix \mathbf{B}_T obtained from $\tilde{\mathbf{B}}_T$ by adding an n -th column \mathbf{e}_n corresponding to the root arc and adding an n -th row corresponding to the root vertex is lower triangular with nonzero diagonal elements. This matrix \mathbf{B}_T is therefore a basis for the row space of \mathbf{A}_r , as claimed. Furthermore, since $\tilde{\mathbf{B}}_T$ is lower triangular with nonzero diagonal elements, so too is the matrix \mathbf{B}_T .

To prove the converse, suppose that \mathbf{B} is a basis matrix for the row space of \mathbf{A}_r . Then, \mathbf{B} is an $n \times n$ matrix. We show that \mathbf{B} is the vertex-arc incidence matrix of some rooted spanning tree of rD . Since $\text{rank}(\mathbf{A}) = n - 1$, the basis matrix \mathbf{B} must contain the column corresponding to the root arc. Let ${}^rD_{\mathbf{B}}$ be the rooted spanning subdigraph whose vertex-arc incidence matrix is the matrix \mathbf{B} . Then, ${}^rD_{\mathbf{B}}$ contains the root arc and $n - 1$ arcs of D . Let $T_{\mathbf{B}}$ be the underlying digraph of ${}^rD_{\mathbf{B}}$. That is, $T_{\mathbf{B}}$ is obtained from ${}^rD_{\mathbf{B}}$ by simply removing the root arc. Then, $T_{\mathbf{B}}$ is a spanning subdigraph of D of order n and size $n - 1$. We show that the underlying graph of $T_{\mathbf{B}}$ contains no cycle.

Suppose, to the contrary, that the underlying graph of $T_{\mathbf{B}}$ contains a cycle C . We may assume C is the cycle $1, 2, \dots, k, 1$. The cycle C corresponds to a semicycle $C' : 1, a_1, 2, a_2, 3, \dots, k, a_k, 1$ in the digraph D , where for $i \in [k - 1]$, a_i is either a forward arc $(i, i + 1)$ or a backward arc $a_i = (i + 1, i)$ on C' , while a_k is either a forward arc $(k, 1)$ or a backward arc $(1, k)$ on C' . For $i \in [k]$, if a_i is a forward arc on C' we define $\alpha_i = 1$, while if a_i is a backward arc on C' we define $\alpha_i = -1$. Let $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$ denote the columns of \mathbf{B} corresponding to the arcs a_1, a_2, \dots, a_k , respectively. Then, $\alpha_1 \mathbf{c}_1 + \alpha_2 \mathbf{c}_2 + \dots + \alpha_k \mathbf{c}_k = \mathbf{0}$. This, however, contradicts the fact that the columns of \mathbf{B} are linearly independent. We deduce, therefore, that the underlying graph of $T_{\mathbf{B}}$ contains no cycles.

Since the underlying graph of $T_{\mathbf{B}}$ contains no cycles, and has order n and size $n - 1$, it is a tree by [Theorem 5.3](#). Consequently, $T_{\mathbf{B}}$ is a spanning directed tree of D . Thus, $D_{\mathbf{B}}$ (which is obtained from $T_{\mathbf{B}}$ by adding the root arc) is a rooted spanning tree of rD . Hence we have shown that \mathbf{B} is the vertex-arc incidence matrix of some rooted spanning tree of rD . ■

To illustrate the proof of [Theorem 8.2](#), consider again the digraph $D_{8,1}$ in [Figure 8.2](#). Let $T_{8,2}$ be the spanning directed tree of $D_{8,1}$ shown in [Figure 8.3](#), let ${}^rD_{8,1}$ be the rooted digraph, and let ${}^rT_{8,2}$ be the rooted spanning tree shown in [Figure 8.4](#). The matrix $\mathbf{B}_{T_{8,2}}$ is obtained from the matrix $\tilde{\mathbf{B}}_{T_{8,2}}$ by adding a fourth column \mathbf{e}_4 corresponding to the root arc and adding a fourth row corresponding to the root vertex 1. The matrix $\mathbf{B}_{T_{8,2}}$, shown in [Figure 8.5](#), is the vertex-arc incidence or basis matrix of the rooted spanning tree ${}^rT_{8,2}$ of ${}^rD_{8,1}$.

By [Theorem 8.2](#), we know that every basis matrix \mathbf{B} for the row space of the vertex-arc incidence matrix of a weakly connected rooted digraph is lower triangular with nonzero diagonal elements. Once we have such a basis matrix \mathbf{B} in our linear programming problem, we set the nonbasic variables equal to zero and solve the system of equations $\mathbf{Bx}_B = \mathbf{b}$, where \mathbf{x}_B contains the basic variables.

$$\mathbf{B}_{T_{8.2}} = \begin{array}{c} \text{root} \\ \begin{matrix} & (1,4) & (2,3) & (1,2) & \text{arc} \\ \begin{matrix} 4 & -1 & 0 & 0 & 0 \\ 3 & 0 & -1 & 0 & 0 \\ 2 & 0 & 1 & -1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{matrix} \end{matrix} \end{array}$$

Figure 8.5: A basis matrix $\mathbf{B}_{T_{8.2}}$.

Since \mathbf{B} is a lower triangular matrix, this system of equations is easily solved by forward substitution.

For example, in our digraph $D_{8.1}$ in Figure 8.2, suppose $b_1 = 7$, $b_2 = 4$, $b_3 = -6$ and $b_4 = -5$. Then the basic solution associated with the basic matrix $\mathbf{B} = \mathbf{B}_{T_{8.2}}$ in Figure 8.5 is found by solving the matrix system of equations

$$\mathbf{B}\mathbf{x}_B = \underbrace{\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} x_{14} \\ x_{23} \\ x_{12} \\ x_1 \end{bmatrix}}_{\mathbf{x}_B} = \begin{bmatrix} -5 \\ -6 \\ 4 \\ 7 \end{bmatrix} = \begin{bmatrix} b_4 \\ b_3 \\ b_2 \\ b_1 \end{bmatrix} = \mathbf{b},$$

where x_1 denotes the flow along the root arc. Taking advantage of the lower triangular structure of the basis matrix \mathbf{B} , we may readily solve for \mathbf{x}_B by forward substitution. From row 1 we have that $x_{14} = 5$. From row 2 we have that $x_{23} = 6$. From row 3 we have that $x_{12} = x_{23} - 4 = 2$. Finally substituting into row 4, we have that $x_1 = 7 - x_{14} - x_{12} = 7 - 5 - 2 = 0$. Thus, the flow along the root arc is zero (as it should be) and the basic feasible solution associated with the basis matrix \mathbf{B} is given by

$$\mathbf{x}_B^T = [x_{14}, x_{23}, x_{12}, x_1]^T = [5, 6, 2, 0]^T.$$

We observe that if the supply and demands are integers, the value of all the basic variables will be integral. Thus, every basic feasible solution of the minimum-cost network flow program will be integral and so any optimal solution will also be integral.

❖ The reader should now be able to attempt Exercises 8.5–8.7.

8.4 The network simplex algorithm

In this section we present the **network simplex algorithm** which is a simplification of the **simplex algorithm** that is very much faster than the latter general-purpose algorithm.

We begin, however, with a review of the **simplex algorithm**. A key to the **simplex algorithm** is to recognize when a given basic feasible solution is optimal. For this purpose, suppose that \mathbf{B} is an $n \times n$ basis matrix for our linear programming problem (8.1). Let \mathbf{N} be the $n \times (m - n + 1)$ matrix whose columns form the columns of \mathbf{A} associated with the nonbasic variables. Let \mathbf{x}_B and \mathbf{x}_N denote the set of basic and nonbasic variables, respectively, for the basis matrix \mathbf{B} . Thus, $\mathbf{x}_B \in \mathbb{R}^n$ and $\mathbf{x}_N \in \mathbb{R}^{m-n+1}$, and $\mathbf{A}_r \mathbf{x} = \mathbf{b}$ can be written as $\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$

(where, as before, \mathbf{A}_r is the vertex-arc incidence matrix of rD , the rooted digraph with underlying digraph D). Let \mathbf{c}_B and \mathbf{c}_N be the objective function cost vectors associated with the basic and nonbasic variables, respectively. The objective function $z = \mathbf{c}^T \mathbf{x}$ can therefore be written as $z = \mathbf{c}_B^T \mathbf{x}_B + \mathbf{c}_N^T \mathbf{x}_N$. Let

$$z_0 = \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b}$$

and

$$z_{ij} = \mathbf{c}_B^T \mathbf{B}^{-1} (\mathbf{e}_i - \mathbf{e}_j)$$

for every nonbasic variable x_{ij} in \mathbf{x}_N . Recall that $\mathbf{e}_i - \mathbf{e}_j$ is the column of the matrix \mathbf{A} that represents the arc $(i, j) \in E$ and corresponds to the variable x_{ij} . Then the linear programming problem (8.1) can be rewritten in such a way that the objective function $z = \mathbf{c}^T \mathbf{x}$ is expressed in terms of the nonbasic variables only as follows:

$$\begin{aligned} \text{Minimise} \quad & z = z_0 - \sum_{x_{ij} \in \mathbf{x}_N} \bar{c}_{ij} x_{ij} \\ \text{subject to} \quad & \left. \begin{array}{l} \mathbf{Bx}_B + \mathbf{Nx}_N = \mathbf{b} \\ \mathbf{x}_B \geq \mathbf{0}, \mathbf{x}_N \geq \mathbf{0} \end{array} \right\} \end{aligned} \quad (8.4)$$

where

$$\bar{c}_{ij} = z_{ij} - c_{ij} = \mathbf{c}_B^T \mathbf{B}^{-1} (\mathbf{e}_i - \mathbf{e}_j) - c_{ij} = \mathbf{y}^T (\mathbf{e}_i - \mathbf{e}_j) - c_{ij} = y_i - y_j - c_{ij}.$$

We remark that sometimes the variables x_{ij} are also bounded from above. In this case, $0 \leq x_{ij} \leq K_{ij}$ for each arc (i, j) and for some constant K_{ij} ; i.e. the flow along the arc (i, j) is bounded from below by zero and from above by K_{ij} .

Notice that if $\bar{c}_{ij} \leq 0$ for every nonbasic variable x_{ij} , then $z \geq z_0$ for every feasible solution. For the current basis matrix, however, we know that $z = z_0$ since $x_{ij} = 0$ for every nonbasic variable. Thus, *if $\bar{c}_{ij} \leq 0$ for every nonbasic variable x_{ij} , then the current basic feasible solution is optimal*. This observation is a key result in recognising the optimality of a given basic feasible solution. The steps of the simplex algorithm for our problem are shown in [Algorithm 19](#).

Algorithm 19: The simplex algorithm for solving linear programming problems

Input : A basis matrix \mathbf{B}

Output : An optimal solution

- 1 Find an initial basic feasible solution
- 2 Compute \bar{c}_{ij} for every nonbasic variable x_{ij}
- 3 **if** ($\bar{c}_{ij} \leq 0$ for every nonbasic variable x_{ij}) **then**
- 4 **if** ($\bar{c}_{ij} > 0$) **and** (unboundedness is not encountered) **then**
- 5 Determine a nonbasic variable to enter the basis and a basic variable to exit the basis
- 6 **stop**
- 7 After pivoting, determine the new basic feasible solution and **go to Step 2**

We are now in a position to show how the [simplex algorithm](#) can be applied to our minimum-cost network flow problem in a very specialised way, resulting in what is called the [network simplex algorithm](#).

We first illustrate the [network simplex algorithm](#) by applying the steps of the [simplex algorithm](#) summarised above to the digraph $D_{8.1}$ in [Figure 8.2](#), where $\mathbf{b}^T = [b_1, b_2, b_3, b_4]^T = [7, 4, -6, -5]^T$ and $\mathbf{c}^T = [c_{12}, c_{14}, c_{23}, c_{31}, c_{34}]^T = [2, 8, 1, 4, 2]^T$. In this example, we assume $0 \leq x_{ij} \leq 20$ for each arc (i, j) ; that is, the flow along each arc is bounded from below by zero and from above by 20.

For the moment, we assume that [Step 1](#) is completed; that is, we assume that we have an initial basic feasible solution. In our example, with the basis matrix $\mathbf{B} = \mathbf{B}_{T_{8.2}}$ in [Figure 8.5](#), suppose we already have the basic feasible solution $\mathbf{x}_\mathbf{B}^T = [x_{14}, x_{23}, x_{12}, x_1]^T = [5, 6, 2, 0]^T$. The cost vector associated with the basis vector is $\mathbf{c}_\mathbf{B}^T = [c_{14}, c_{23}, c_{12}, c_1]^T = [8, 1, 2, 0]^T$, where c_1 is the cost of the root arc from the root vertex 1.

In [Step 2](#), we compute \bar{c}_{ij} for every nonbasic variable x_{ij} . First we need to compute the dual variables. For $i \in [n]$, let y_i be the dual variable of the vertex i . Let $\mathbf{y} \in \mathbb{R}^n$ be the vector of dual variables, where the variable in the i -th position of \mathbf{y} is the dual variable of the vertex in the i -th primal constraint. For our basis matrix $\mathbf{B} = \mathbf{B}_{T_{8.2}}$ in [Figure 8.5](#), we have $\mathbf{y}^T = [y_4, y_3, y_2, y_1]^T$ since the vertex in the first primal constraint is vertex 4, the vertex in the second primal constraint is vertex 3, the vertex in the third primal constraint is vertex 2, and the vertex in the fourth primal constraint is vertex 1. To compute the dual variables associated with our current basis matrix $\mathbf{B} = \mathbf{B}_{T_{8.2}}$, we solve the system $\mathbf{y}^T = \mathbf{c}_\mathbf{B}^T \mathbf{B}^{-1}$, or, equivalently,

$$\mathbf{y}^T \mathbf{B} = \mathbf{c}_\mathbf{B}^T.$$

For our basis matrix $\mathbf{B}_{T_{8.2}}$ in [Figure 8.5](#), we have

$$\mathbf{y}^T \mathbf{B} = [y_4, y_3, y_2, y_1] \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} = [8, 1, 2, 0] = \mathbf{c}_\mathbf{B}^T,$$

which yields the four equations

$$\begin{aligned} -y_4 + y_1 &= 8, \\ -y_3 + y_2 &= 1, \\ -y_2 + y_1 &= 2 \text{ and} \\ y_1 &= 0. \end{aligned}$$

From the fourth equation, the row associated with the root vertex 1, we have that $y_1 = 0$. From the third equation, we have that $y_2 = y_1 - 2 = -2$. From the second equation, we have that $y_3 = y_2 - 1 = -3$. From the first equation, we have that $y_4 = y_1 - 8 = -8$. Thus,

$$\mathbf{y}^T = [y_4, y_3, y_2, y_1]^T = [-8, -3, -2, 0]^T,$$

which is the dual solution associated with the current basic feasible solution $\mathbf{x}_\mathbf{B}^T = [x_{14}, x_{23}, x_{12}, x_1]^T = [5, 6, 2, 0]^T$. For the nonbasic variables x_{31} and x_{34} , we have

$$\bar{c}_{31} = y_3 - y_1 - c_{31} = -3 - 0 - 4 = -7,$$

$$\bar{c}_{34} = y_3 - y_4 - c_{34} = -3 - (-8) - 2 = 3.$$

Moving on to [Step 3](#) we note that $\bar{c}_{34} > 0$ for the nonbasic variable x_{34} . As such, the current basic feasible solution

$$\mathbf{x}_B^T = [x_{14}, x_{23}, x_{12}, x_1]^T = [5, 6, 2, 0]^T$$

is not optimal.

Next we determine which current nonbasic variable enters the basis and which current basic variable exits the basis as in [Step 5](#). Let

$$\delta = \max_{x_{ij} \in \mathbf{x}_N} \{\bar{c}_{ij}\};$$

that is, δ is the maximum value of \bar{c}_{ij} taken over all nonbasic variable x_{ij} . In our example,

$$\delta = \max\{\bar{c}_{31}, \bar{c}_{34}\} = \max\{-7, 3\} = 3.$$

Since $\delta = \bar{c}_{34}$, the nonbasic variable x_{34} enters the basis (in the case of a tie, we can choose the entering variable arbitrarily). To determine which current basic variable exits the basis, we add to the rooted spanning tree $T_{8,2}$ associated with the current basis matrix $\mathbf{B}_{T_{8,2}}$, the arc $(3, 4)$. Adding this arc $(3, 4)$ forms a unique semicycle

$$C : 3, (3, 4), 4, (1, 4), 1, (1, 2), 2, (2, 3), 3$$

which contains the arc $(3, 4)$. Note that the semicycle C is obtained from the unique semipath joining the vertices 3 and 4 in $T_{8,2}$ by adding to it the arc $(3, 4)$. We now assign the semicycle C an orientation in the direction of the added arc $(3, 4)$. Thus, in this orientation of C , the arc $(3, 4)$ is a forward arc.

Suppose we wish to increase the flow x_{34} from 0 to θ . Then, by the flow conservation constraint, we need to increase the flow along every forward arc (with respect to our orientation) on the semicycle C by θ and we need to decrease the flow along every backward arc on C by θ . The flow along all arcs not in the semicycle C remains unchanged.

In our example, setting $x_{34} = \theta$ increases the flow into vertex 4 by θ and therefore, by the flow conservation constraint, the flow out of vertex 4 must decrease by θ . Thus, we must decrease x_{14} by θ . This requires that $x_{14} = 5 - \theta$. Since the flow out of vertex 1 has now decreased by θ , we must increase the flow out of vertex 1 by θ . Thus, we must increase x_{12} by θ . This requires that $x_{12} = 2 + \theta$. Since the flow into vertex 2 has now increased by θ , we must increase the flow out of vertex 2 by θ . Thus, we must increase x_{23} by θ . This requires that $x_{23} = 6 + \theta$. Hence setting $x_{34} = \theta$ decreases the flow x_{14} by θ , increases the flow x_{12} by θ , and increases the flow x_{23} by θ . In essence we have sent an additional amount of flow θ around the semicycle C created by adding the nonbasic arc to the rooted spanning tree associated with the current basis matrix $\mathbf{B}_{T_{8,2}}$. Along every backward arc in our orientation of C , we “push” back a flow of θ , while along every forward arc we increase the flow by θ . This is illustrated in [Figure 8.6](#).

We next determine the new basic feasible solution in [Step 7](#). We wish to find the maximum flow $\theta \geq 0$ that can be sent along our semicycle in the orientation of the entering arc $(3, 4)$. If $\theta \rightarrow \infty$, then the problem is unbounded and we stop. Otherwise, if $\theta < \infty$, we make θ as large as possible subject to the nonnegativity constraint, namely that every variable is nonnegative, and the capacity constraint, namely that every variable is at most 20. In our example, we have the restriction

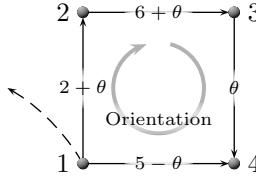


Figure 8.6: Determining a basic variable to exit the basis in $T_{8,2} + (3,4)$.

that the new value of the current basic variable $x_{14} = 5 - \theta$ is nonnegative. This requires that $\theta \leq 5$. Furthermore, the new value of the current basic variable $x_{12} = 2 + \theta \leq 20$. This requires that $\theta \leq 18$. Finally, the new value of the current basic variable $x_{23} = 6 + \theta \leq 20$. This requires that $\theta \leq 14$. Thus, the largest possible value of θ is 5. The basic variable that drops to zero when θ attains its maximum value is chosen to exit the basis (in the case of a tie, we can choose the exiting variable arbitrarily). Thus, x_{14} leaves the basis and the arc $(1,4)$ is removed from the rooted spanning tree. The flows are updated by appropriately adjusting the flows in the semicycle by θ and leaving the remaining flows unchanged. Hence the new primal vector of basic variables is

$$\mathbf{x}_B^T = [x_{34}, x_{23}, x_{12}, x_1]^T = [\theta, 6 + \theta, 2 + \theta, 0]^T = [5, 11, 7, 0]^T$$

associated with the new rooted spanning tree $T_{8,3}$ and the new basis matrix $\mathbf{B} = \mathbf{B}_{T_{8,3}}$ in Figure 8.7. The cost vector associated with the basis vector of $\mathbf{B} = \mathbf{B}_{T_{8,3}}$ is $\mathbf{c}_B^T = [c_{34}, c_{23}, c_{12}, c_1]^T = [2, 1, 2, 0]^T$.

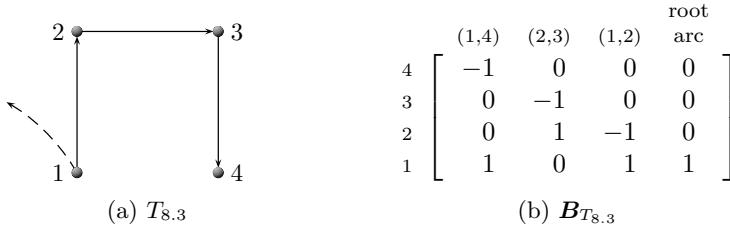


Figure 8.7: The new rooted spanning tree $T_{8,3}$ and associated basis matrix $\mathbf{B}_{T_{8,3}}$.

We now return to Step 2. For our basis matrix $\mathbf{B} = \mathbf{B}_{T_{8,3}}$ in Figure 8.7, we have $\mathbf{y}^T = [y_4, y_3, y_2, y_1]$ and

$$\mathbf{y}^T \mathbf{B} = [y_4, y_3, y_2, y_1] \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [2, 1, 2, 0] = \mathbf{c}_B^T,$$

which yields the four equations

$$\begin{aligned} -y_4 + y_3 &= 2, \\ -y_3 + y_2 &= 1, \\ -y_2 + y_1 &= 2 \text{ and} \\ y_1 &= 0. \end{aligned}$$

Solving this system yields $\mathbf{y}^T = [y_4, y_3, y_2, y_1] = [-5, -3, -2, 0]$, which is the dual solution associated with the current basic feasible solution \mathbf{x}_B^T . For the nonbasic variables x_{14} and x_{31} , we have

$$\begin{aligned}\bar{c}_{14} &= y_1 - y_4 - c_{14} = 0 - (-5) - 8 = -3, \\ \bar{c}_{31} &= y_3 - y_1 - c_{31} = -3 - 0 - 4 = -7.\end{aligned}$$

In Step 3, $\bar{c}_{ij} \leq 0$ for every nonbasic variable x_{ij} , implying that the current basic feasible solution

$$\mathbf{x}_B^T = [x_{34}, x_{23}, x_{12}, x_1] = [5, 11, 7, 0]$$

is optimal. The associated minimum cost is

$$c_{34}x_{34} + c_{23}x_{23} + c_{12}x_{12} = (2)(5) + (1)(11) + (2)(7) = 35.$$

The approach followed above may be summarised as in [Algorithm 20](#), known as the [network simplex algorithm](#).

We now return to the problem of finding an initial basic feasible solution. Suppose we are given a minimum-cost network flow problem defined on a weakly connected digraph D of order $n \geq 2$ and size m . In order to find an initial basic feasible solution for this problem, we construct a new minimum-cost network flow problem as follows. Let D' be the digraph obtained from D by adding a new vertex $n+1$, adding a root arc incident with $n+1$, and for every vertex $i \in V(D)$ with $b_i \geq 0$ adding the arc $(i, n+1)$, while for every vertex $i \in V(D)$ with $b_i < 0$ adding the arc $(n+1, i)$. With this new vertex $n+1$, we associate a number $b_{n+1} = 0$, implying that the net flow out of the root vertex $n+1$ should be zero. To every arc $(i, j) \in E(D)$ of D , we assign a cost $c'_{ij} = 0$, while to every new arc $(i, n+1)$ or $(n+1, i)$ we assign a cost of 1 and we assign a cost of 0 to the root arc. Associated with each arc incident with $n+1$, we have therefore introduced artificial variables x_1, x_2, \dots, x_{n+1} , where x_{n+1} is the flow along the root arc and for $i \in [n]$, x_i is the flow along the arc joining i and $n+1$. This creates a basic feasible solution to our new minimum-cost network flow problem consisting entirely of artificial variables. From this basic feasible solution, we solve the new minimum-cost network flow problem by invoking the [network simplex algorithm](#). This will produce an initial basic feasible solution to our original problem, assuming that the problem has a feasible solution.

To illustrate this method, consider the digraph $D_{8.1}$ in [Figure 8.8\(a\)](#), where $\mathbf{b}^T = [b_1, b_2, b_3, b_4]^T = [7, 4, -6, -5]^T$. The digraph $D'_{8.1}$ is shown in [Figure 8.8\(b\)](#), where the number associated with each arc is the cost of that arc.

The rooted spanning tree associated with the basic feasible solution consisting entirely of artificial variables $x_{15}, x_{25}, x_{53}, x_{54}, x_5$ is shown in [Figure 8.9](#), where x_5 denotes the flow along the root arc (and x_{ij} denotes the flow along the arc (i, j)).

The basic feasible solution associated with the basis matrix $\mathbf{B} = \mathbf{B}_{D'_{8.1}}$ in [Figure 8.9](#) is found by solving the matrix system of equations

$$\mathbf{Bx}_B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ -1 & -1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_{15} \\ x_{25} \\ x_{53} \\ x_{54} \\ x_5 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ -6 \\ -5 \\ 0 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}.$$

Algorithm 20: The network simplex algorithm for solving transportation problems

Input : A basis matrix \mathbf{B}

Output : An optimal solution

- 1 Find an initial basic feasible solution \mathbf{x}_B represented by a rooted spanning tree T and associated basis matrix \mathbf{B}_T
 - 2 Find the vector $\mathbf{y}^T = [y_1, y_2, \dots, y_n]$ of dual variables associated with the primal vector of basis variables $\mathbf{x}_B^T = [x_1, x_2, \dots, x_n]$
 - 3 Compute $\bar{c}_{ij} = y_i - y_j - c_{ij}$ for every nonbasic variable x_{ij} and determine $\delta = \max\{\bar{c}_{ij}\}$, where the maximum value is taken over all nonbasic variables x_{ij}
 - 4 **if** ($\delta \leq 0$) **then**
 - 5 | The current basic feasible solution is optimal, **stop**
 - 6 **if** ($\delta > 0$) **then**
 - 7 | Introduce the nonbasic variable x_{ij} into the basis for which $\bar{c}_{ij} = \delta$ (in the case of a tie, choose the entering variable arbitrarily)
 - 8 Determine the unique semicycle formed by adding the arc (i, j) to the rooted spanning tree, and find the maximum flow $\theta \geq 0$ that can be sent along the semicycle in the orientation of the entering arc (i, j)
 - 9 **if** ($\theta \rightarrow \infty$) **then**
 - 10 | The problem is unbounded; **stop**
 - 11 **if** ($\theta < \infty$) **then**
 - 12 | Make θ as large as possible subject to the nonnegativity constraint and the capacity constraint. The basic variable x_{st} that drops to zero when θ attains this maximum value exits the basis (in the case of a tie, we can choose the exiting variable arbitrarily)
 - 13 Update the flows by appropriately adjusting the flows in the semicycle by θ and leaving the remaining flows unchanged
 - 14 Remove the arc (s, t) from the rooted spanning tree and add the arc (i, j) to it so as to form a new rooted spanning tree with an associated basis matrix and associated primal vector of new basic variables. Now **go to Step 2**
-

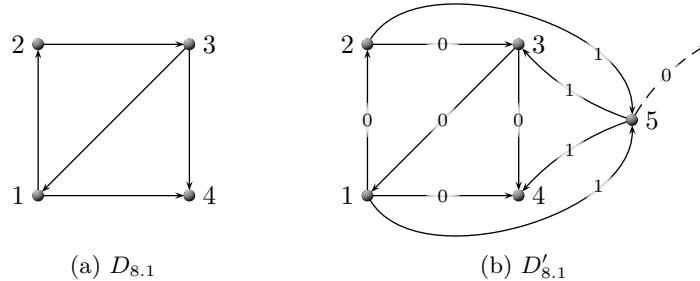


Figure 8.8: The modified digraph $D'_{8.1}$ associated with the digraph $D_{8.1}$.

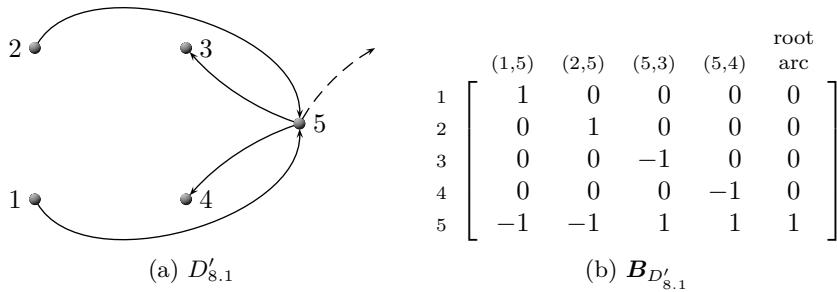


Figure 8.9: The starting rooted spanning tree $D'_{8.1}$ and associated basis matrix $\mathbf{B} = \mathbf{B}_{D'_{8.1}}$.

This yields the starting basic feasible solution

$$\mathbf{x}_{\mathbf{B}}^T = [x_{15}, x_{25}, x_{53}, x_{54}, x_5]^T = [7, 4, 6, 5, 0]^T$$

to our new minimum-cost network flow problem. The cost vector associated with this basis vector is $\mathbf{c}_{\mathbf{B}}^T = [c_{15}, c_{25}, c_{53}, c_{54}, c_5]^T = [1, 1, 1, 1, 0]^T$, where c_5 is the cost of the root arc from the root vertex 5. We now go to **Step 2** of the [network simplex algorithm](#).

For our basis matrix $\mathbf{B} = \mathbf{B}_{D'_{8.1}}$ in [Figure 8.9](#), we have $\mathbf{y}^T = [y_1, y_2, y_3, y_4, y_5]^T$ and

$$\mathbf{y}^T \mathbf{B} = [y_1, y_2, y_3, y_4, y_5] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ -1 & -1 & 1 & 1 & 1 \end{bmatrix} = [1, 1, 1, 1, 0] = \mathbf{c}_{\mathbf{B}}^T,$$

which yields the four equations

$$\begin{aligned} y_1 - y_5 &= 1, \\ y_2 - y_5 &= 1, \\ y_5 - y_3 &= -1, \\ y_5 - y_4 &= -1 \text{ and} \\ y_5 &= 0. \end{aligned}$$

Solving this system yields $\mathbf{y}^T = [y_1, y_2, y_3, y_4, y_5]^T = [1, 1, -1, -1, 0]^T$, which is the dual solution associated with the current basic feasible solution \mathbf{x}_B^T . For the nonbasic variables, we have

$$\begin{aligned}\bar{c}_{12} &= y_1 - y_2 - c_{12} = 1 - 1 - 0 &= 0, \\ \bar{c}_{14} &= y_1 - y_4 - c_{14} = 1 - (-1) - 0 &= 2, \\ \bar{c}_{23} &= y_2 - y_3 - c_{23} = 1 - (-1) - 0 &= 2, \\ \bar{c}_{31} &= y_3 - y_1 - c_{31} = 1 - 1 - 0 &= 0, \\ \bar{c}_{34} &= y_3 - y_4 - c_{34} = -1 - (-1) - 0 &= 0 \text{ and} \\ \bar{c}_{31} &= y_3 - y_1 - c_{31} = -3 - 0 - 4 &= -7.\end{aligned}$$

Since $\delta = 2 = \bar{c}_{14}$, we choose the nonbasic variable x_{14} to enter the basis. To determine which current basic variable exits the basis, we add to the rooted spanning tree $D'_{8,1}$ (see Figure 8.9(a)) associated with the current basis matrix $\mathbf{B} = \mathbf{B}_{D'_{8,1}}$, the arc $(1, 4)$ to form a unique semicycle

$$C : 1, (1, 4), 4, (5, 4), 5, (1, 2), 1$$

which contains the arc $(1, 4)$. We now orient this semicycle in the direction of the entering arc $(1, 4)$. Setting $x_{14} = \theta$ increases the flow x_{14} by θ , decreases the flow x_{54} by θ , and decreases the flow x_{15} by θ . This requires that $x_{14} = \theta$, $x_{54} = 5 - \theta$ and $x_{15} = 7 - \theta$. The maximum flow $\theta \geq 0$ that can be sent along the semicycle in the orientation of the entering arc $(1, 4)$ is therefore $\theta = 5$. Since the basic variable x_{54} drops to zero when $\theta = 5$, x_{54} leaves the basis and the arc $(5, 4)$ is removed from the rooted spanning tree. The flows are updated by $x_{14} = 5$, $x_{54} = 0$ and $x_{15} = 2$, with the remaining flows remaining unchanged. Hence the new primal vector of basic variables is

$$\mathbf{x}_B^T = [x_{14}, x_{15}, x_{25}, x_{53}, x_5]^T = [5, 2, 4, 6, 0]^T$$

associated with the new rooted spanning tree $D''_{8,1}$ and the new basis matrix $\mathbf{B} = \mathbf{B}_{D''_{8,1}}$ in Figure 8.10. The cost vector associated with the basis vector is $\mathbf{c}_B^T = [c_{14}, c_{15}, c_{25}, c_{53}, c_5]^T = [0, 1, 1, 1, 0]^T$.

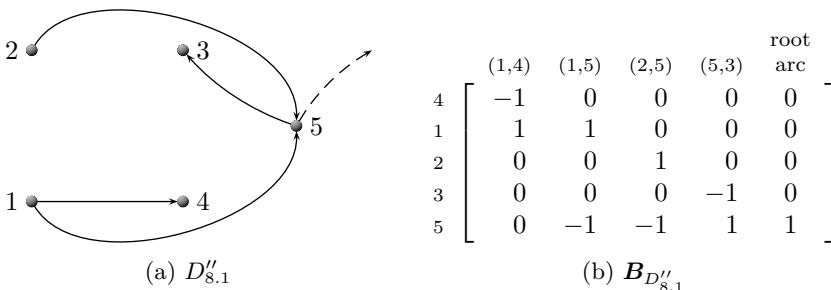


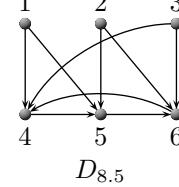
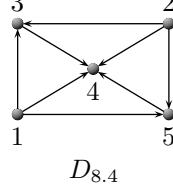
Figure 8.10: The new rooted spanning tree $D''_{8,1}$ and associated basis matrix $\mathbf{B}_{D''_{8,1}}$.

We now return to Step 2 of the [network simplex algorithm](#) and continue the process to eventually produce an initial basic feasible solution to our original problem, assuming that the problem has a feasible solution. We leave the details as an exercise.

- ❖ The reader should now be able to attempt Exercises 8.8–8.11 and Projects 8.1–8.2.

Exercises

- 8.1 Consider the transportation problem of transporting personal computers (PCs) from four assembly warehouses S_1, S_2, S_3, S_4 to seven distribution outlets D_1, \dots, D_7 . The warehouses S_1, S_2, S_3 and S_4 can assemble PCs at rates of respectively 18, 22, 15 and 20 per day. From past experience the distribution outlets expect to be able to sell respectively 8, 12, 10, 15, 6, 4 and 14 PCs per day. Draw a bipartite graph model similar to that in [Figure 8.1](#) of the possible flows in the transportation problem, showing the available supplies and required demands at each vertex.
- 8.2 Write down, for each weakly connected digraph below, its vertex-arc incidence matrix.



- 8.3 Write down the minimum-cost flow problem formulation [\(8.2\)](#) for each of the digraphs and vertex-arc incidence matrices in [Exercise 8.2](#), *not* using vector notation, if the corresponding cost vectors and supply/demand vectors are:

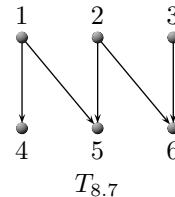
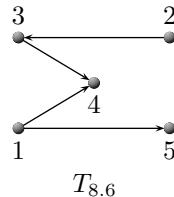
- (a) $\mathbf{c} = [c_{13}, c_{14}, c_{15}, c_{23}, c_{24}, c_{25}, c_{34}, c_{54}]^T = [10, 4, 5, 4, 3, 2, 3, 2]^T$ and
 $\mathbf{b} = [b_1, b_2, b_3, b_4, b_5]^T = [8, 6, -3, -4, -7]^T$;
- (b) $\mathbf{c} = [c_{14}, c_{15}, c_{25}, c_{26}, c_{34}, c_{36}, c_{45}, c_{56}, c_{64}]^T = [7, 6, 7, 6, 2, 8, 2, 3, 1]^T$
and $\mathbf{b} = [b_1, b_2, b_3, b_4, b_5, b_6]^T = [15, 7, 8, -10, -8, -12]^T$.

- 8.4 Write down the dual problem formulation [\(8.3\)](#) for each of the (primal) minimum-cost flow problem formulations in [Exercise 8.3](#).

- 8.5 Determine the ranks of the following matrices:

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \mathbf{W}_2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 9 & 8 \end{bmatrix} \quad \mathbf{W}_3 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

- 8.6 Determine the basis matrix \mathbf{B} for the row space of the vertex-arc incidence matrices of the weakly connected digraphs in [Exercise 8.2](#) corresponding to the respective rooted spanning trees $T_{8.6}$ and $T_{8.7}$ below.



- 8.7 Use the supply/demand vectors in [Exercise 8.3](#) and your basis matrices in [Exercise 8.6](#) to find basic feasible solutions to the minimum-cost flow problems in [Exercise 8.3](#).
- 8.8 A method was described at the end of [Section 8.4](#) for finding an initial basic feasible solution to the minimum-cost flow problem [\(8.2\)](#), in which all basic variables are artificial variables. The section closed with an incomplete illustration of applying this method to the minimum-cost flow problem for the digraph $D_{8.1}$ in [Figure 8.2](#). Complete the computation so as to find an initial basic feasible solution.
- 8.9 Apply the method described at the end of [Section 8.4](#) for finding an initial basic feasible solution to the minimum-cost flow problem [\(8.2\)](#), in which all basic variables are artificial variables, to the minimum-cost flow problems in Exercises [8.2–8.3](#).
- 8.10 Apply the [network simplex algorithm](#) ([Algorithm 20](#)) to the minimum-cost flow problems in [Exercise 8.3](#), using your basic feasible solutions in [Exercise 8.9](#) as initial solutions.
- 8.11 Verify the correctness of your answers to [Exercise 8.10](#) by solving the problem formulations in [Exercise 8.3](#) directly via the [MATHEMATICA](#) implementation [LinearProgramming\[•, •, •\]](#) of the [simplex algorithm](#) ([Algorithm 19](#)).

Computer exercises

The command [MatrixRank\[X\]](#) may be used to determine the rank of a square matrix \mathbf{X} . Recall that a matrix is stored in [MATHEMATICA](#) as a list of row entries, and that the command [MatrixForm\[X\]](#) may be used to display a matrix in normal array form. For example, the commands

```
In[1]:= X = {{1, 2, 3}, {4, 5, 6}, {7, 9, 8}};
In[2]:= MatrixForm[X]
In[3]:= MatrixRank[X]
```

produce

```
Out[2]:= 
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 9 & 8 \end{pmatrix}$$

Out[3]:= 3
```

as output. A full stop may be used to perform multiplication between matrices and vectors. More specifically, the command $\mathbf{X} \cdot \mathbf{x}$ will produce the $m \times k$ product between an $m \times n$ matrix \mathbf{X} and an $n \times k$ matrix \mathbf{x} . The command [Dot\[a, b\]](#) may be used to find the dot product or scalar product between a $1 \times r$ vector \mathbf{a} and an $r \times 1$ vector \mathbf{b} , whilst the command [Transpose\[A\]](#) may be used to find the transpose of a matrix or vector \mathbf{A} .

For example, the commands

```
In[4]:= x = {{1}, {2}, {3}};
In[5]:= w = {{1}, {1/2}, {1/3}};
In[6]:= MatrixForm[x]
In[7]:= MatrixForm[w]
In[8]:= y = X.x;
In[9]:= MatrixForm[y]
In[10]:= z = Dot[Transpose[x], w]
```

produce the output

$$\text{Out}[6]:= \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{Out}[7]:= \begin{pmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{pmatrix} \quad \text{Out}[9]:= \begin{pmatrix} 14 \\ 32 \\ 49 \end{pmatrix}$$

Out[10]:= {{3}}

The command **Inverse[X]** finds the inverse of a nonsingular matrix X over the real numbers, whilst **Det[X]** yields the determinant of X . For example, the commands

```
In[11]:= Det[X]
In[12]:= MatrixForm[Inverse[X]]
```

yield

$$\text{Out}[11]:= 9$$

$$\text{Out}[12]:= \begin{pmatrix} -\frac{14}{9} & \frac{11}{9} & -\frac{1}{3} \\ \frac{10}{9} & -\frac{13}{9} & \frac{2}{3} \\ \frac{1}{9} & \frac{5}{9} & -\frac{1}{3} \end{pmatrix}$$

as output. The command $u = \text{LinearSolve}[X, y]$ uses Gauss elimination to solve a linear system of equations of the form $X.u = y$. For example, the commands

```
In[13]:= u = LinearSolve[X, y];
In[14]:= MatrixForm[u]
```

produce

$$\text{Out}[14]:= \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

as output. Finally, **IdentityMatrix[n]** produces the $n \times n$ identity matrix. For example,

```
In[15]:= MatrixForm[IdentityMatrix[5]]
```

yields

$$\text{Out}[15]:= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

as output. Unfortunately, the network simplex algorithm for minimum-cost flow problems has not been implemented as a built-in function in **MATHEMATICA**. The command $x = \text{LinearProgramming}[c, A, b]$ may, however, be used to solve the linear program minimise $c^T x$ subject to the nontrivial constraints $Ax \geq b$ and the domain constraints $x \geq 0$ by invoking the ordinary **simplex algorithm**. Some or all of the nontrivial constraints may be made equality constraints by replacing the third argument “ b ” with $\{\{b_1, s_1\}, \{b_2, s_2\}, \dots\}$ in which case the i -th nontrivial constraint is treated as \geq if $s_i = 1$ or as $=$ if $s_i = 0$. Nonzero lower bounds of the form $x_i \geq L_i$ may also be imposed by typing $\text{LinearProgramming}[c, A, b, \{L_1, L_2, \dots\}]$, whilst lower and upper bounds of the form $L_i \leq x_i \leq U_i$ may be imposed if the format $\text{LinearProgramming}[c, A, b, \{\{L_1, U_1\}, \{L_2, U_2\}, \dots\}]$ is used. For example, the commands

```
In[16]:= c = {2, 8, 1, 4, 2};
In[17]:= A = {{1, 1, 0, -1, 0}, {-1, 0, 1, 0, 0}, {0, 0, -1, 1, 1}, {0, -1, 0, 0, -1}};
In[18]:= MatrixForm[c]
In[19]:= MatrixForm[A]
In[20]:= x = LinearProgramming[c, A, {{7, 0}, {4, 0}, {-6, 0}, {-5, 0}}]
In[21]:= Dot[c, x]
```

may be used to find the solution

$$\text{Out}[18]:= \begin{pmatrix} 2 \\ 8 \\ 1 \\ 4 \\ 2 \end{pmatrix} \quad \text{Out}[19]:= \begin{pmatrix} 1 & 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \\ 0 & -1 & 0 & 0 & -1 \end{pmatrix}$$

```
Out[20]:= {7, 0, 11, 0, 5}
Out[21]:= 35
```

to the small instance of minimum-cost network flow problem (8.2) considered throughout the chapter for the digraph $D_{8.1}$ in Figure 8.2(a).

Projects

This section contains two projects. In both projects, the reader is required to solve real-world minimum-cost network flow problems by means of the **network simplex algorithm** (Algorithm 20) — in the context of oil export from the OPEC countries in the first project and in the context of fresh fruit export from South Africa in the second project.

Project 8.1: Oil export from the Middle East

The *Organization of Petroleum Exporting Countries* (OPEC) controls the price of oil exported from the Middle East and is also responsible for deciding upon an export strategy from its ten main oil fields (the vertices numbered 1–10 in Figure 8.11 and Table 8.1) via the nine major oil exporting port cities (the vertices numbered 11–19 in Figure 8.11 and Table 8.1).

The cost of pumping oil from an origin to a destination in Figure 8.11 is directly proportional to the distance (along the pipeline) between the origin and destination. These distances are given in Table 8.2.

The following tasks provide a step-wise approach towards finding a minimum-cost oil export strategy for the OPEC countries.

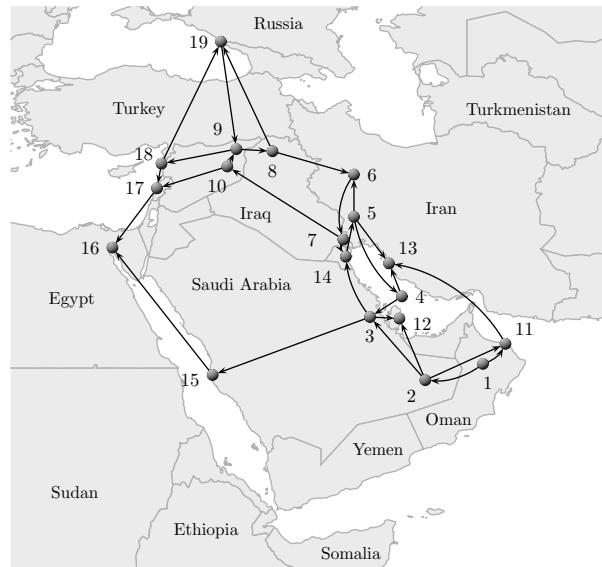


Figure 8.11: OPEC pipelines used for oil export from the Middle East oil fields numbered 1–10 (see Table 8.1 for the names and daily supply rates of these oil fields) to the major port cities of the region (see Table 8.1 for the names and daily demand rates via these ports).

Tasks

1. Spend ten minutes to produce a feasible export strategy by inspection. Evaluate the desirability of your solution in units of millions of barrel · kilometres.
 2. Construct the 19×29 vertex-arc incidence matrix for this minimum-cost network flow problem, and read this matrix into **MATHEMATICA**.
 3. Complete the minimum-cost network flow formulation by reading in a demand vector from **Table 8.1** (taking care to observe our sign convention for distinguishing between supply and demand) as well as a cost vector from **Table 8.2**.
 4. Use the **MATHEMATICA** command **LinearProgramming[•, •, •]** to solve the minimum-cost network flow problem formulated in Tasks 2 and 3 above.
 5. Evaluate the “cost” of your optimal solution in **Task 4** above in units of millions of barrel · kilometres. How good was your intuition in **Task 1** when finding a flow schedule heuristically, by inspection?
 6. Cast your optimal solution of Tasks 4 and 5 into an easily understandable form by constructing an export instruction table of the following form:

Transport ?amount? million barrels of oil from ?placename? to ?placename?

Transport ?amount? million barrels of oil from ?placename? to ?placename?

Transport ?amount? million barrels of oil from ?placename? to ?placename?

•
•

Vertex	Oil field	Supply	Vertex	Port city	Demand
1	Al Ghabah	2	11	Muscat	5
2	Ash Sharaqiyah	3	12	Doha	6
3	Al Hofuf	8	13	Būushehr	2
4	Persian Gulf	7	14	Kuwait City	7
5	Chogha Zambil	5	15	Jeddah	2
6	Hamadan	4	16	Suez	9
7	Basrah	5	17	Beirut	4
8	Mosul	3	18	Latakia	3
9	Al Hasakah	2	19	Sochi	2
10	Deir ez-Zur	1			
<i>Total</i>		40	<i>Total</i>		40

Table 8.1: Oil supply from OPEC oil fields and oil demand at port cities of the Middle East in millions of barrels per day.

Route	Distance	Route	Distance	Route	Distance
1 → 2	280 km	1 → 11	210 km	2 → 3	430 km
2 → 11	390 km	2 → 12	290 km	3 → 12	170 km
3 → 14	310 km	3 → 15	840 km	4 → 3	200 km
4 → 13	180 km	5 → 4	170 km	5 → 6	180 km
5 → 13	220 km	6 → 7	140 km	7 → 10	540 km
7 → 14	150 km	8 → 6	380 km	8 → 19	560 km
9 → 8	260 km	9 → 18	360 km	10 → 9	120 km
10 → 17	350 km	11 → 13	480 km	14 → 5	200 km
15 → 16	560 km	17 → 16	300 km	18 → 17	90 km
18 → 19	670 km	19 → 9	420 km		

Table 8.2: Distances between the oil fields and port cities in [Figure 8.11](#).

Project 8.2: Minimum export transportation cost strategy¹

The volumes of fruit grown in South Africa for fresh fruit export purposes during 2005 are shown in [Figure 8.12](#). A breakdown of these fruit volumes produced per fruit growing region in [Figure 7.6](#) during the summer peak week (week 7) and the winter peak week (week 29) are shown in [Table 8.3](#).

The costs (in Rands per pallet) of transporting fruit from the different regions in [Figure 7.6](#) by road to the four export ports are shown in [Table 8.4](#). (The regional pack houses are so close to the picking regions and the regional cold stores are so close to the corresponding regional pack houses that the transportation costs of fruit along roads linking these infrastructure segments are negligible.)

Tasks

1. Determine a minimum transportation cost export strategy for the summer fruit produced during week 7, as listed in [Table 8.3](#), taking into account the pack house and cold storage capacities in [Tables 7.1](#) and [7.2](#). What is the total minimum transportation cost? (Hint: Add two dummy variables,

¹The reader should have completed [Project 7.1](#) of [Chapter 7](#) before attempting this project.

Code	Summer peak week						Winter peak week			
	Grapes	Stone	Pome	Hard		Litchi	Pome	Hard		Soft
				Citrus	Mango			Citrus	Citrus	
Le					9	47		6 985		72
Tz					401	130		13 240		760
Ho	16				432			3 087	51	
Ki								877		487
Ne					69	37		4 545		520
Ma					209	113		7 040		
Po								625		
El										
Ma				7				2 475	322	
Er										
Po								27		
Nk								7 322	58	
Kz								981	26	150
Ea								1 494	43	
Su								6 564	406	
Pa								4 185	133	
La	198	2 011					1 422	12		
Ka										
Kl	705	588					167			
Ge	6 437	663	5 910				4 479	3 931	1 070	
Bo	8 115	1 960	930				182	3 776		
Ce	2 543	629	5 986				2 127			
De	1 272									
Pi	218	94	260				163			
Ci										
Or	52							279		
Ha		45								
Jo		79								
Ru		25						8		
Fs			33				41			
Mi				16				2 590		
Total	18 653	4 398	15 718	23	1 120	327	8 581	70 043	2 109	1 989

Table 8.3: Export fruit (in pallets) produced per region in South Africa during the summer and winter peak weeks of 2005.

Code	Region	Cape Town (R/pallet)	Durban (R/pallet)	Gqeberha (R/pallet)	Maputo (R/pallet)
Le	Levubu	1 334	782	1 137	545
Tz	Tzaneen	1 282	700	1 085	429
Ho	Hoedspruit	1 499	605	1 097	319
Ki	Kiepersol	1 489	592	1 091	262
Ne	Nelspruit	1 444	548	1 046	218
Ma	Malelane	1 504	583	1 106	176
Po	Potgietersrus	1 186	632	989	497
El	Ellisras	1 253	689	1 027	573
Mh	Marble Hall	1 404	575	1 006	430
Er	Ermelo	1 353	414	961	376
Pg	Pongola	1 403	338	981	479
Nk	Nkwalini	1 262	196	1 065	573
Kz	KwaZulu-Natal	1 146	141	659	646
Ea	EC Midlands	717	624	230	1 263
Su	Sundays River	634	753	147	1 391
Pa	Patensie	547	794	149	1 254
La	Langkloof	528	845	206	1 298
Ka	Karoo	299	1 353	376	961
Kl	Klein Karoo	197	1 135	518	1 371
Ge	GEVV	95	1 142	524	1 377
Bo	Boland	99	1 131	565	1 367
Ce	Ceres	158	1 104	584	1 339
De	De Doorns	185	1 120	571	1 348
Pi	Piketberg	175	1 170	606	1 405
Ci	Citrusdal	200	1 196	675	1 431
Or	Orange River	650	888	715	950
Ha	Hartswater	782	597	681	756
Jo	Johannesburg	1 024	484	794	458
Ru	Rustenburg	1 300	563	926	529
Fs	Free State	901	343	704	555
Mi	Middelburg	1 320	486	1 000	368

Table 8.4: Road transportation costs for containerised fruit from the fruit growing regions in South Africa to four ports in the region from which fruit export takes place.

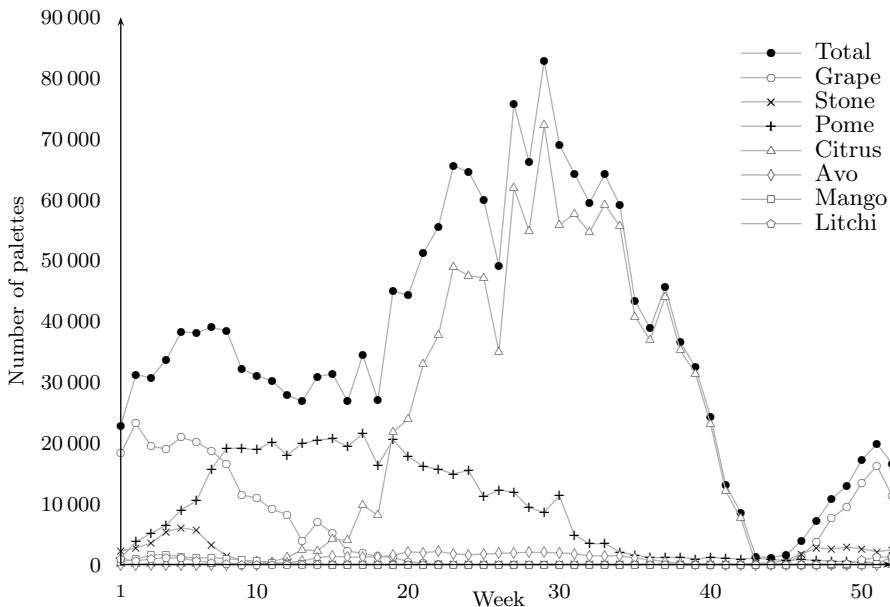


Figure 8.12: Fresh fruit volumes produced in South Africa during 2005.

namely *source*, representing all fruit growth activity and a *sink*, representing all the foreign export markets collectively. Link the source by means of zero-cost arcs to the regional packhouses and also link the four ports to the sink by means of zero-cost arcs. Now there is a single supply vertex (the source) and a single demand vertex (the sink).)

2. Repeat [Task 1](#), but this time for the winter fruit produced during week 29, as listed in [Table 8.3](#).
3. Determine the bottleneck(s) in your minimum cost strategies in Tasks 1 and 2 above (infrastructure components in your solutions which, if enlarged in terms of capacity, would allow fruit to be exported more cheaply than is possible in your answers in Tasks 1 and 2 above).

Further reading

- [1] RFV Anderson, 1986. *Introduction to Linear Algebra*, Holt, Rinehart & Winston, New York (NY).
- [2] H Anton, 1987. *Elementary Linear Algebra*, Fifth edition, John Wesley, Sons, New York (NY).
- [3] GB Dantzig and MN Thapa, 1997. *Linear Programming*, Volume I: Introduction, Springer, New York (NY).
- [4] GB Dantzig and MN Thapa, 1997. *Linear Programming*, Volume II: Theory and Extensions, Springer, New York (NY).
- [5] BN Datta, 2010. *Numerical Linear Algebra and Applications*, Second edition, SIAM, Philadelphia (PA).

- [6] SI Gass, 1969. *Linear Programming*, Third edition, McGraw-Hill, New York (NY).
- [7] G Hadley, 1962. *Linear Programming*, Addison-Wesley, London.
- [8] RC McCann, 1984. *Introduction to Linear Algebra*, Harcourt Brace Jo-vanovich Publishers, San Diego (CA).
- [9] SG Nash and A Sofer, 1996. *Linear and Nonlinear Programming*, McGraw-Hill, New York (NY), Chapters 1–9.
- [10] H Paley and PM Weichsel, 1972. *Elements of Abstract and Linear Algebra*, Holt, Rinehart & Winston, New York (NY), Chapters 7–11.
- [11] M Simonnard, 1966. *Linear Programming*, Prentice-Hall, Englewood Cliffs (NJ).
- [12] G Strang, 1993. *Introduction to Linear Algebra*, Wellesley-Cambridge Press, Wellesley (MA).
- [13] S Vajda, 1974. *Theory of Linear and Nonlinear Programming*, Longman, London, Chapters 1–3.



PART

Topics in
Classical Graph Theory



Matchings

Contents

9.1	Introduction	223
9.2	Maximum matchings	224
9.3	Vertex covers	227
9.4	Tutte's theorem	232
9.5	The Tutte-Berge formula	234
9.6	The binding number of a graph	246
9.7	Matching algorithms for bipartite graphs	250
9.8	A matching algorithm for general graphs	252
9.9	A weighted matching algorithm	262
	Exercises	281
	Computer exercises	284
	Projects	285
	Suggestions for further background reading	289
	Further reading	289

9.1 Introduction

In this chapter, we consider special subgraphs that a graph may contain. In particular, we consider 1-regular subgraphs of maximum size in a given graph. Applications of these subgraphs to the Job Assignment Problem and the Marriage Problem are described. We begin with a few definitions.

Two distinct edges in a graph G are **independent** if they are not adjacent in G (*i.e.* the two edges are not incident with a common vertex). A set of pairwise independent edges of G is called a **matching** in G , that is, a matching in G is a 1-regular subgraph of G . A matching of maximum cardinality is called a **maximum matching** in G .

In the graph $G_{9.1}$ in Figure 9.1(a), the set $M_1 = \{e_1, e_4\}$ is a matching that is not a maximum matching, while $M_2 = \{e_1, e_3, e_5\}$ and $M_3 = \{e_1, e_3, e_6\}$ are maximum matchings in $G_{9.1}$. In the graph $G_{9.2}$ in Figure 9.1(b), the set $M = \{v_1v_8, v_3v_4, v_6v_7\}$ is a maximum matching in $G_{9.2}$.

For sets M_1 and M_2 , we denote by $M_1 \Delta M_2$ the symmetric difference of M_1 and M_2 , *i.e.* $M_1 \Delta M_2 = (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$.

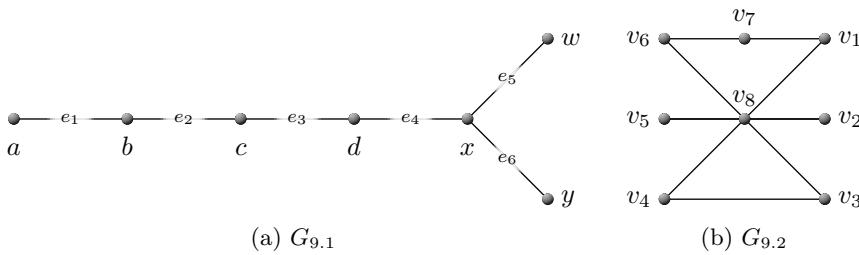


Figure 9.1: Example graphs illustrating introductory concepts.

9.2 Maximum matchings

In this section, we present a characterisation of maximum matchings. First we introduce some additional notation. Let M be a specified matching in a graph G . A vertex v of G is an **M -matched vertex** if v is incident with an edge of M ; otherwise, v is an **M -unmatched vertex**. If the matching M is clear from context, we simply call an M -matched vertex a **matched vertex** and an M -unmatched vertex an **unmatched vertex**. An **M -alternating path** of G is a path whose edges are alternately in M and not in M . An **M -augmenting path** is an M -alternating path that begins and ends with M -unmatched vertices.

Consider, for example, the matching $M = \{v_1v_8, v_3v_4, v_6v_7\}$ in the graph $G_{9.2}$ in Figure 9.1(b). The vertices v_2 and v_5 are M -unmatched vertices. The path $v_2 - v_8 - v_1 - v_7 - v_6$ is an M -alternating path that is not an M -augmenting path. For the matching $M_1 = \{e_1, e_4\}$ in the graph $G_{9.1}$ in Figure 9.1(a), $cdxy$ is an M_1 -augmenting path. We are now in a position to present a characterisation of maximum matchings due to Berge [2].

Theorem 9.1 (Berge's Theorem) *Let M be a matching in a graph G . Then M is a maximum matching if and only if there exists no M -augmenting path in G .*

Proof Suppose there exists an M -augmenting path P in G . Then P has odd length. Let M' be the set of edges of P belonging to M , and let $M'' = E(P) \setminus M'$. Since $|M''| = |M| + 1$, the set $(M \setminus M') \cup M''$ is a matching whose cardinality exceeds that of M . Hence M is not a maximum matching in G . This proves the necessity.

Next we consider the sufficiency. Suppose that M is *not* a maximum matching in G . We show that there exists an M -augmenting path in G . Let M^* be a maximum matching in G . Then $|M^*| > |M|$. Let H be the spanning subgraph of G defined by $E(H) = M^* \Delta M$. Each vertex of H has degree at most 2, since it can be incident with at most one edge of M and one edge of M^* . Thus, every component of H is a path (possibly trivial) or a cycle. Since no two edges in a matching are adjacent, the edges of each cycle and path in H are alternately in M and M^* . Thus each cycle in H is even. Since $|M^*| > |M|$, H contains more edges of M^* than of M . Hence there exists some path component P of H that starts and ends with edges of M^* . Let P be a $u-v$ path.

We show that both u and v are M -unmatched vertices. Let e be the edge of P incident with u and suppose there is an edge f in M (thus, $f \neq e$) such that f is incident with u . Since e and f are adjacent, $f \notin M^*$. Thus, $f \in M \setminus M^*$. It follows that f belongs to H , and so u is adjacent to both e and f in H . This is,

however, impossible since u has degree 1 in H . Therefore, u is an M -unmatched vertex. Similarly, v is an M -unmatched vertex, and so the path P begins and ends with M -unmatched vertices. Hence, P is an M -augmenting path in G . ■

According to [Theorem 9.1](#), given a matching M in a graph G , it is possible to decide whether M is a maximum matching by searching for an M -augmenting path in G . In applications, maximum matchings in bipartite graphs have proved to be most useful. Consider, for example, the following incarnation of the class of **Assignment Problems**: Given a set of available jobs and a set of applicants for one or more of these positions, determine an assignment of applicants to these jobs so that a maximum number of positions is filled.

This situation can be modelled by a bipartite graph G , whose partite sets correspond to the applicants and to the available jobs. A vertex corresponding to an applicant is joined by an edge to each of the vertices that correspond to the jobs for which the applicant has applied. The objective is to find a maximum matching in G .

Motivated by applications such as the one described above, we shall develop a general theory of matchings in bipartite graphs. To do this, we require some further definitions.

Let G be a graph with nonempty subsets X and Y of vertices. We say X is **matched** to Y if there exists a matching M in G such that each edge of M is incident with a vertex of X and a vertex of Y , and every vertex of $X \cup Y$ is M -matched. If $M \subseteq M^*$, where M^* is a matching in G , we also say that X is **matched under** M^* to Y . For any nonempty set of vertices of a graph G , the neighbourhood $N(S)$ of S denotes the set of all vertices of G adjacent with at least one element of S .

We now present a result, due to [König \[29\]](#) and [Hall \[18\]](#), dating back to 1935.

Theorem 9.2 ([Hall's Theorem](#)) *Let G be a bipartite graph with partite sets X and Y . Then X can be matched to a subset of Y if and only if*

$$|N(S)| \geq |S|$$

for every nonempty subset S of X .

Proof Suppose that X can be matched to a subset of Y under a matching M . Then every nonempty subset S of X can be matched under M to some subset of Y . Since these vertices of S are matched under M with distinct vertices in $N(S)$, it follows that $|N(S)| \geq |S|$.

To verify the converse, suppose $|N(S)| \geq |S|$ for every nonempty subset S of X . Assume, to the contrary, that X cannot be matched to a subset of Y , and let M be a maximum matching in G . By assumption, there is vertex v in X that is M -unmatched. Let Z be the set of all vertices of G that are connected to v by an M -alternating path. Since M is a maximum matching, it follows from [Theorem 9.1](#) that v is the only M -unmatched vertex in Z .

Let $S = Z \cap X$ and let $T = Z \cap Y$. Using the definition of the set Z , together with the fact that no vertex of $Z \setminus \{v\}$ is M -unmatched, we conclude that $S \setminus \{v\}$ is matched under M to T . Therefore,

$$|T| = |S| - 1$$

and $T \subseteq N(S)$. Furthermore, since every vertex w in $N(S)$ is connected to v by an M -alternating v - w path (for otherwise there would exist a v - w M -augmenting path), we know that $N(S) \subseteq T$. Thus, $T = N(S)$ and so

$$|N(S)| = |T| = |S| - 1 < |S|.$$

This, however, contradicts our assumption that $|N(S)| \geq |S|$ for every nonempty subset S of X . We conclude, therefore, that X can be matched to a subset of Y . ■

Philip Hall was born in Hampstead, London, on 11 April 1904. He obtained a doctorate in mathematics at the University of Cambridge under the supervision of Karl Pearson. Hall was elected to a Fellowship at King's College, Cambridge in March 1927. In 1933, he was appointed as a lecturer at Cambridge and was promoted to reader in 1949. In 1953, he was appointed Sadleirian Professor of Mathematics at Cambridge. His major work was in group theory, notably on finite groups and solvable groups. In graph theory, he is best known for what is now known as [Hall's marriage theorem](#) ([Theorem 9.4](#)), which dates back to 1935. He was elected Fellow of the Royal Society in 1951 and was awarded its Sylvester Medal in 1961. He was President of the London Mathematical Society during the period 1955–1957, and was awarded its Berwick Prize in 1958 and the De Morgan Medal in 1965. He died in Cambridge on 30 December 1982.



Biographic note 24: Philip Hall (1904–1982)

We next consider special matchings called perfect matchings. If M is a matching in a graph G with the property that every vertex of G is M -matched, then M is called a **perfect matching** in G . If G has a perfect matching M , then every vertex of G is incident with an edge of M , and so G has even order and the edges of M induce a 1-regular spanning subgraph of G . Thus, the graph $G_{9.1}$ in [Figure 9.1\(a\)](#) cannot have a perfect matching.

A matching in a graph G that matches all but one vertex of G is called an **almost perfect matching** in G . Hence if G has an almost perfect matching, then $G - v$ has a perfect matching for some vertex v in G . Note that if G has an almost perfect matching, then G has odd order.

The next result, due to [König \[28\]](#), is a consequence of [Theorem 9.2](#).

Corollary 9.3 *Every regular bipartite graph of degree $r \geq 1$ has a perfect matching.*

Proof Let G be an r -regular bipartite graph for some integer $r \geq 1$ with partite sets X and Y . Since G has $r|X| = r|Y|$ edges, we know that $|X| = |Y|$. Let S be a nonempty subset of X , and let E_1 and E_2 denote the sets of edges incident with the vertices in S and $N(S)$, respectively. By definition of $N(S)$, every edge in E_1 joins a vertex in S with a vertex in $N(S)$, and so every edge in E_1 belongs

to E_2 . It follows that, $|E_1| \leq |E_2|$. However, the number of edges incident with the vertices in S is $r|S|$ and the number of edges incident with the vertices in $N(S)$ is $r|N(S)|$. Therefore,

$$r|N(S)| = |E_2| \geq |E_1| = r|S|,$$

and so, $|N(S)| \geq |S|$ since $r \geq 1$. Since S was an arbitrary subset of X , this shows that $|N(S)| \geq |S|$ for every nonempty subset S of X . Hence, by [Theorem 9.2](#), X can be matched to a subset of Y . Since $|X| = |Y|$, a matching that matches X to a subset of Y is a perfect matching. ■

[Corollary 9.3](#) is sometimes known as the [Marriage Theorem](#), since it can be more colourfully restated as follows:

Theorem 9.4 (Hall's Marriage Theorem) *If every girl in a village knows exactly r boys, and every boy knows exactly r girls, then each girl can marry a boy she knows, and each boy can marry a girl he knows.*

❖ The reader should now be able to attempt Exercises [9.1–9.8](#).

9.3 Vertex covers

We next introduce the concept of a cover in a graph. A vertex and an edge are said to **cover** each other in a graph G if they are incident in G . A **(vertex) cover** in G is a set of vertices that covers all the edges of G . The minimum cardinality among all the vertex covers in G is called the **(vertex) covering number** of G and is denoted by $\beta(G)$.

In the graph $G_{9.3}$ in [Figure 9.2](#), the set $S = \{v_1, v_2, \dots, v_5\}$ is a minimal vertex cover in $G_{9.3}$ (*i.e.* no proper subset of S is a vertex cover) that is not a minimum vertex cover. However, $K = \{v, v_2, v_4, v_5\}$ is a minimum vertex cover, and so $\beta(G_{9.3}) = 4$.

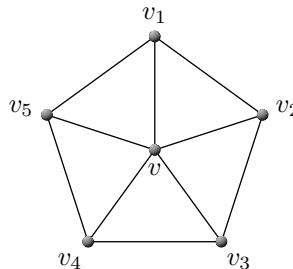


Figure 9.2: The graph $G_{9.3}$.

The number of edges in a maximum matching of a graph G is called the **matching number** (or the **edge independence number**) of G , denoted by $\alpha'(G)$. (Equivalently, $\alpha'(G)$ is the maximum cardinality among all the independent sets of edges of G .)

We show that the matching number is at most the vertex covering number.

Theorem 9.5 *For every graph G , $\alpha'(G) \leq \beta(G)$.*

Proof Let M be a maximum matching in G and let K be a minimum covering of G . Since K is a vertex cover of G , every edge of M is incident with at least one vertex of K . Hence, K contains at least one end of each of the edges in M , and so $\alpha'(G) = |M| \leq |K| = \beta(G)$. \blacksquare

A simple, but useful, observation follows from [Theorem 9.5](#).

Corollary 9.6 *Let M be a matching in a graph G and let K be a vertex covering of G such that $|M| = |K|$. Then M is a maximum matching and K is a minimum covering.*

Proof Let M^* be a maximum matching in G and let K^* be a minimum covering of G . Then, $|M| \leq |M^*|$ and $|K^*| \leq |K|$. Hence, by [Theorem 9.4](#),

$$|M| \leq |M^*| = \alpha'(G) \leq \beta(G) = |K^*| \leq |K|.$$

Since $|M| = |K|$, we must have equality throughout in the above chain. In particular, it follows that $|M| = |M^*|$ and $|K^*| = |K|$. Hence M is a maximum matching in G and K is a minimum covering of G . \blacksquare

In general, equality does not hold in the result of [Theorem 9.5](#). For example, for the graph $G_{9,3}$ in [Figure 9.2](#), $\alpha'(G_{9,3}) = 3$ ($M = \{v_1v_2, v_3v_4, vv_5\}$ is a maximum matching in G) while $\beta(G_{9,3}) = 4$. Hence, there do exist graphs G for which $\alpha'(G) < \beta(G)$. Our next result, due to [König \[29\]](#) and [Egerváry \[12\]](#), however, shows that if we restrict our attention to bipartite graphs G , then we must have equality in the result of [Theorem 9.5](#).

Theorem 9.7 (König-Egerváry Theorem) *If G is a bipartite graph, then*

$$\alpha'(G) = \beta(G).$$

Proof Let G have partite sets X and Y , and let M be a maximum matching of G , so that $|M| = \alpha'(G)$. If X is matched to Y under M , then $|M| = |X|$. Since G is a bipartite graph, however, X is a vertex cover of G . Hence, by [Corollary 9.6](#), X is a minimum vertex cover of G , and so $|X| = \beta(G)$ and $\alpha'(G) = \beta(G)$. We may therefore assume that X is not matched to Y under M .

Let U be the set of all M -unmatched vertices of X , and let Z be the set of all vertices of G that are connected to vertices of U by M -alternating paths. Let $S = Z \cap X$ and let $T = Z \cap Y$. Then, as in the proof of [Theorem 9.2](#), $S \setminus U$ is matched to T under M (so, $|S| - |U| = |T|$) and $N(S) = T$. We now let

$$K = (X \setminus S) \cup T$$

and show that K is a vertex cover in G . If this were not the case, then there would be an edge vw in G with $v \in S$ and $w \in Y \setminus T$, which contradicts our earlier observation that $N(S) = T$. Hence K is a vertex cover in G . Furthermore,

$$|K| = |X| - |S| + |T| = |X| - |U| = |M|.$$

Therefore, by [Corollary 9.6](#), K is a minimum vertex cover in G . That is, $\alpha'(G) = |M| = |K| = \beta(G)$. \blacksquare

Our next aim is to provide an upper bound on the covering number of a graph in terms of the order and size of the graph. For this purpose, we introduce some additional terminology. We call a tree that contains a perfect matching a **matching-tree**, and we call a component of a graph that is a matching-tree a **matching-tree component** of the graph. As a consequence of [Theorem 9.7](#), we have the following observation.

Observation 9.8 *If T is a matching-tree of order n , then $\beta(T) = \alpha'(T) = \frac{n}{2}$.*

For a graph G , we define $\text{doc}(G)$ to be the maximum number of vertex disjoint odd cycles in G and we define $\text{mtc}(G)$ to be the number of matching-tree components in G . Recall that $n(G)$ and $m(G)$ denote the order and size, respectively, of the graph G . We are now in a position to present an upper bound on the vertex cover number due to [Henning and Yeo \[24\]](#).

Theorem 9.9 *For every graph G ,*

$$4\beta(G) \leq n(G) + m(G) + 2\text{doc}(G) + \text{mtc}(G).$$

Proof Let $\varphi(G) = n(G) + m(G) + 2\text{doc}(G) + \text{mtc}(G)$. We wish to show that $4\beta(G) \leq \varphi(G)$. We proceed by induction on the order $n(G) \geq 1$ of G . If $n(G) = 1$, then $\alpha(G) = 0$ and $\varphi(G) = 1$, while if $n(G) = 2$, then $\alpha(G) = 1$ and $\varphi(G) = 4$. In both cases the desired result holds. This establishes the base case. Now let G be any graph of order $n(G) \geq 3$ and assume, as induction hypothesis, that the desired result holds for all graphs of order less than $n(G)$.

Suppose that $\delta(G) = 0$, let v be an isolated vertex in G and let $G_v = G - v$. Then $4\beta(G) = 4\beta(G_v)$ and by induction, $4\beta(G_v) \leq \varphi(G_v)$. Since $\varphi(G_v) = \varphi(G) - 1$, we have that $4\beta(G) \leq \varphi(G)$, as desired. Hence we may assume that $\delta(G) \geq 1$.

Suppose that $\delta(G) = 1$. Let x be a vertex of degree 1 in G and let y be the neighbour of x . Let $G' = G - \{x, y\}$. Then $n(G') = n(G) - 2$ and $m(G') = m(G) - d_G(y)$. Since G' is a subgraph of G , it follows that $\text{doc}(G') \leq \text{doc}(G)$. We show next that $\text{mtc}(G') \leq \text{mtc}(G) + d_G(y) - 2$. Any matching-tree component in G' which was not a matching-tree component in G contains a vertex which is incident with one of the edges we deleted from G . As the vertex x is not in G' , there are $d_G(y) - 1$ vertices in G' that are incident with one of the edges we deleted from G . Therefore, $\text{mtc}(G') \leq \text{mtc}(G) + (d_G(y) - 1)$. If $\text{mtc}(G') = \text{mtc}(G) + (d_G(y) - 1)$, then x and y belonged to a matching-tree component in G which now does not belong to G' . This implies that $\text{mtc}(G') \leq \text{mtc}(G) + (d_G(y) - 1) - 1 = \text{mtc}(G) + d_G(y) - 2$, as claimed. Applying the induction hypothesis to the graph G' , it follows that

$$\begin{aligned} 4\beta(G') &\leq n(G') + m(G') + 2\text{doc}(G') + \text{mtc}(G') \\ &\leq (n(G) - 2) + (m(G) - d_G(y)) + 2\text{doc}(G) + (\text{mtc}(G) + d_G(y) - 2) \\ &= n(G) + m(G) + 2\text{doc}(G) + \text{mtc}(G) - 4 = \varphi(G) - 4. \end{aligned}$$

Let X' be a minimum vertex cover in G' and let $X = X' \cup \{y\}$. Then $|X'| = \beta(G')$ and X is a vertex cover of G . Thus, $4\beta(G) \leq 4|X| = 4|X'| + 4 = 4\beta(G') + 4 \leq \varphi(G)$, as desired. We may therefore assume that $\delta(G) \geq 2$.

Suppose that $\delta(G) = 2$. Then $\text{mtc}(G) = 0$. Let x be a vertex of degree 2 in G and let $N(x) = \{y_1, y_2\}$. Furthermore, let Y be the set of vertices in G' that are adjacent to y_1 or y_2 in G . Thus, $Y = (N_G(y_1) \cup N_G(y_2)) \setminus \{x, y_1, y_2\}$.

We first consider the case where $y_1y_2 \in E(G)$. If $n(G) = 3$, then G is a 3-cycle and $\beta(G) = 2$, $m(G) = 3$ and $\text{doc}(G) = 1$, whence $4\beta(G) = \varphi(G)$, as desired. Hence, we may assume that $n(G) \geq 4$. Let $G' = G - \{x, y_1, y_2\}$ and let X' be a minimum vertex cover in G' . Then $n(G') = n(G) - 3$ and $m(G') = m(G) - d_G(y_1) - d_G(y_2) + 1$. As $\delta(G) = 2$, all vertices of degree less than 2 in G' belong to the set Y . Therefore, every matching-tree in G' contains at least two vertices from Y . Hence, $\text{mtc}(G') \leq |Y|/2 \leq (d_G(y_1) + d_G(y_2) - 4)/2$. Finally, we note that $\text{doc}(G) \geq \text{doc}(G') + 1$, as we can add the 3-cycle $x y_1 y_2 x$ to any collection of odd disjoint cycles in G' in order to obtain a collection of odd disjoint cycles in G . Applying the induction hypothesis to the graph G' , it follows that

$$\begin{aligned} 4\beta(G') &\leq n(G') + m(G') + 2\text{doc}(G') + \text{mtc}(G') \\ &\leq (n(G) - 3) + (m(G) - d_G(y_1) - d_G(y_2) + 1) \\ &\quad + 2(\text{doc}(G) - 1) + ((d_G(y_1) + d_G(y_2))/2 - 2) \\ &= n(G) + m(G) + 2\text{doc}(G) - 6 - (d_G(y_1) + d_G(y_2))/2 \\ &\leq n(G) + m(G) + 2\text{doc}(G) - 8 = \varphi(G) - 8. \end{aligned}$$

Let X' be a minimum vertex cover in G' and let $X = X' \cup \{y_1, y_2\}$. Then $|X'| = \beta(G')$ and X is a vertex cover of G . Thus, $4\beta(G) \leq 4|X| = 4|X'| + 8 = 4\beta(G') + 8 \leq \varphi(G)$. We may therefore assume that $y_1y_2 \notin E(G)$, for otherwise the desired result follows.

Let G^* be obtained from the graph $G - \{x, y_1, y_2\}$ by adding a new vertex w and joining w to every vertex in Y . Then $n(G^*) = n(G) - 2$ and $m(G^*) = m(G) - 2$. If G is a 4-cycle, then $\beta(G) = 2$, $m(G) = 4$, $m(G) = 4$ and $\text{doc}(G) = 0$, whence $4\beta(G) = \varphi(G)$, as desired. Hence, we may assume that G is not a 4-cycle. In particular, we note that if $d_{G^*}(w) = 1$, then all other vertices in G^* have degree at least two.

We show that $\text{mtc}(G^*) = 0$. Suppose, to the contrary, that $\text{mtc}(G^*) \geq 1$. By construction of the graph G^* and since $\text{mtc}(G) = 0$, we have that $\text{mtc}(G^*) = 1$ and that the vertex w belongs to the matching-tree component T_w in G^* . In particular, we note that the matching-tree T_w contains at least two vertices of degree one. These vertices of degree one in G^* must belong to the set $N_{G^*}(w)$; that is, $N_{G^*}(v) = \{y_1, y_2\}$ for each such vertex v with $d_{G^*}(v) = 1$. This implies that T_w is a star with w as the centre vertex. Since $d_{G^*}(w) \geq 2$, however, such a tree T_w cannot be a matching-tree. Hence, $\text{mtc}(G^*) = 0$.

Any odd cycle in G^* containing w can be transformed into an odd cycle in G by either substituting w by y_1 or y_2 or y_1, x, y_2 . Hence, $\text{doc}(G^*) \leq \text{doc}(G)$. Applying the induction hypothesis to the graph G^* , it follows that

$$\begin{aligned} 4\beta(G^*) &\leq n(G^*) + m(G^*) + 2\text{doc}(G^*) + \text{mtc}(G^*) \\ &\leq (n(G) - 2) + (m(G) - 2) + 2\text{doc}(G) + 0 \\ &\leq n(G) + m(G) + 2\text{doc}(G) - 4 = \varphi(G) - 4. \end{aligned}$$

Let X^* be a minimum vertex cover in G^* , and so $|X^*| = \beta(G^*)$. If $w \in X^*$, let $X = (X^* \setminus \{w\}) \cup \{y_1, y_2\}$. If, on the other hand, $w \notin X^*$, let $X = X^* \cup \{x\}$. In both cases, X is a vertex cover of G . Thus, $4\beta(G) \leq 4|X| = 4|X^*| + 4 = 4\beta(G^*) + 4 \leq \varphi(G)$, as desired. We may therefore assume that $\delta(G) \geq 3$.

Let x be an arbitrary vertex in G and let $G' = G - \{x\}$. Then $n(G') = n(G) - 1$ and $m(G') = m(G) - d_G(x) \leq m(G) - 3$. As $\delta(G) \geq 3$, we note that $\delta(G') \geq 2$, which implies that $\text{mtc}(G') = 0$. As G' is a subgraph of G , we also have $\text{doc}(G') \leq \text{doc}(G)$. Applying the induction hypothesis to the graph G' , it follows that

$$\begin{aligned} 4\beta(G') &\leq n(G') + m(G') + 2\text{doc}(G') + \text{mtc}(G') \\ &\leq (n(G) - 1) + (m(G) - 3) + 2\text{doc}(G) + 0 \\ &\leq n(G) + m(G) + 2\text{doc}(G) - 4 = \varphi(G) - 4. \end{aligned}$$

Let X' be a minimum vertex cover in G' and let $X = X' \cup \{x\}$. Then, $|X'| = \beta(G')$ and X is a vertex cover of G . Thus, $4\beta(G) \leq 4|X| = 4|X'| + 4 = 4\beta(G') + 4 \leq \varphi(G)$, as desired. ■

As an immediate consequence of Theorems 9.5 and 9.9, we have the following result.

Theorem 9.10 *If G is a connected graph that is not a matching-tree, then*

$$4\alpha'(G) \leq 4\beta(G) \leq n(G) + m(G) + 2\text{doc}(G).$$

The upper bound in Theorem 9.10 on the matching number is tight. For example, let G be the graph obtained from an even cycle C_{2k} by adding $r \geq 1$ vertex disjoint matching-trees and adding r edges so that the resulting graph becomes connected. The set of edges consisting of a perfect matching in each of the r added vertex disjoint matching-trees together with a perfect matching in the cycle is a perfect matching in G , and so $4\alpha'(G) = 4n(G)/2 = 2n(G)$. Since $n(G) = m(G)$ while $\text{doc}(G) = 0$, it follows that $n(G) + m(G) + 2\text{doc}(G) = 2n(G)$. Hence the graph G achieves equality in the bounds of Theorem 9.10.

We remark that since

$$\max_{E \subseteq E(G)} \{\text{mtc}(G - E) - |E|\} \geq \text{mtc}(G) - |\emptyset| = \text{mtc}(G),$$

we have the following immediate consequence of Theorem 9.9.

Corollary 9.11 *For every graph G ,*

$$4\beta(G) \leq n(G) + m(G) + 2\text{doc}(G) + \max_{E \subseteq E(G)} \{\text{mtc}(G - E) - |E|\}.$$

It is also possible to establish an upper bound on the vertex cover number of a graph in terms of its order and size.

Theorem 9.12 *Let G be a graph of order n and size m . Then $\beta(G) \leq \frac{3n}{4} + \frac{m}{28}$.*

Proof By induction. Let $a = \frac{3}{4}$ and $b = \frac{1}{28}$. There are two (overlapping) possibilities to be considered.

Case 1: G has maximum degree $\Delta \geq 7$. Let v be a vertex of maximum degree and let G' be the graph $G - v$. Then

$$\alpha'(G) \leq \alpha'(G') + 1 \leq a(n - 1) + b(m - \Delta) + 1 = an + bm + R(\Delta),$$

where $R(\Delta) = 1 - a - b\Delta$. Clearly, $R(\Delta)$ is a decreasing function of Δ and $R(7) = 0$.

Case 2: G has minimum degree $\delta \leq 6$. Let v be a vertex of minimum degree and let G' be the graph obtained from G by deleting v and its δ neighbours. Then

$$\alpha'(G) \leq \alpha'(G') + \delta \leq a(n - (\delta + 1)) + b(m - \binom{\delta+1}{2}) + \delta = an + bm + S(\delta),$$

where $S(\delta) = \delta - a(\delta + 1) - b\binom{\delta+1}{2}$. It may easily be verified that $S(\delta)$ is an increasing function of δ for $\delta \leq 6$, and that $S(6) = 0$. This yields the desired result. ■

Equality in [Theorem 9.12](#) occurs if G is K_7 or K_8 . (We remark that [Theorem 9.12](#) also holds for $a = d(d+3)/(d^2+5d+2)$ and $b = 2/(d^2+5d+2)$, where d is any nonnegative integer.)

- ❖ The reader should now be able to attempt Exercises [9.9–9.12](#).

9.4 Tutte's theorem

In this section we present a characterisation due to [Tutte \[45\]](#) of arbitrary graphs that have perfect matchings. In order to present this characterisation, we introduce some additional notation.

An **odd component** of a graph G is a component of odd order. The number of odd components of G is denoted by $\text{oc}(G)$.

If a graph $G = (V, E)$ has a perfect matching, then it has even order. Moreover, if G has a perfect matching and if $S \subseteq V$, then any perfect matching matches at least one vertex from each odd component of $G - S$, if any, with a vertex in S . Hence, if G has a perfect matching, then $\text{oc}(G - S) \leq |S|$ for every $S \subseteq V$. As a consequence of [Hall's Theorem \(Theorem 9.2\)](#), this necessary condition is also sufficient for a bipartite graph to have a perfect matching, as we show next.

Corollary 9.13 *A bipartite graph G has a perfect matching if and only if $\text{oc}(G - S) \leq |S|$ for every vertex subset $S \subseteq V$.*

Proof Let $G = (V, E)$. The necessary condition follows as noted above. To prove the sufficiency, suppose $\text{oc}(G - S) \leq |S|$ for every subset $S \subseteq V$. Let G have partite sets X and Y . If $S = X$, then $G - S = G[Y]$ consists of isolated vertices, and so $|X| = |S| \geq \text{oc}(G - S) = |Y|$. Therefore, $|X| \geq |Y|$. Similarly, letting $S = Y$, we can show that $|X| \leq |Y|$. Consequently, $|X| = |Y|$. Now let D be an arbitrary nonempty subset of X and consider the set $S = N(D)$. Each vertex of D is isolated in $G - S$ and is, therefore, an odd component in $G - S$, whence $|N(D)| = |S| \geq \text{oc}(G - S) \geq |D|$. By [Hall's Theorem \(Theorem 9.2\)](#) we conclude that G has a perfect matching. ■

Surprisingly, the necessary condition above is also sufficient for an arbitrary graph to have a perfect matching, as was shown by [Tutte \[45\]](#). The following proof of this result is due to [Lovász \[33\]](#).

Theorem 9.14 (Tutte's Theorem) *A graph $G = (V, E)$ has a perfect matching if and only if $\text{oc}(G - S) \leq |S|$ for every vertex subset $S \subseteq V$.*

Proof The necessary condition follows as noted above. To prove the sufficiency, suppose $\text{oc}(G - S) \leq |S|$ for every $S \subseteq V$. Suppose, however, that G does not have a perfect matching. Let G be chosen to be such a counter example with a maximum number of edges. Note that if we add any edge e to G to produce the graph $G' = G + e$, then $\text{oc}(G' - S) \leq \text{oc}(G - S)$ for every $S \subseteq V$. Hence, by the maximality of $E(G)$, G' has a perfect matching.

Let X be the set of all vertices in G that are adjacent to every other vertex in G , and let $Y = V \setminus X$. Furthermore, let $F = G[Y]$. Then each vertex of F is not adjacent to at least one other vertex of F .

We show next that F is the disjoint union of complete graphs. To prove this claim, it suffices to show that F contains no induced path P_3 on three vertices. Suppose that F contains an induced path $a b c$. Let d be a vertex in F not adjacent to b .

By the maximality of $E(G)$, the graph $G + ac$ has a perfect matching M_1 that necessarily contains the edge ac , and $G + bd$ has a perfect matching M_2 that necessarily contains the edge bd . Let H be the subgraph of G induced by the edges $M_1 \Delta M_2$. Note that $ac \in M_1 \setminus M_2$ and $bd \in M_2 \setminus M_1$. We show that H is a disjoint union of cycles. Since each of M_1 and M_2 is a perfect matching, every vertex v of G is incident with an edge of M_1 and an edge of M_2 . If these two edges are equal, then $v \notin V(H)$, while if these two edges are distinct, then v has degree 2 in H . Hence, H is a disjoint union of cycles.

Let C be the cycle in H containing the edge ac . If bd is not an edge of C , then the perfect matching obtained from M_1 by replacing the edges in $M_1 \cap E(C)$ with the edges in $M_2 \cap E(C)$ is a perfect matching of G , contradicting our choice of G . Therefore, bd is an edge of C .

Removing ac and bd from C we obtain two paths. Let P be the path having d as one end-vertex. Then the other end-vertex of P is either a or c . We may assume, without loss of generality, that P is an a - d path. Consider the cycle $C' = P + bd + ab$. This is an alternating cycle with respect to M_2 . Hence the perfect matching obtained from M_2 by replacing the edges in $M_2 \cap E(C')$ with the edges in $E(C') \setminus M_2$ is a perfect matching of G , contradicting our choice of G . We conclude, therefore, that F has no induced path on three vertices. Hence, F is the disjoint union of complete graphs, as claimed.

By considering $S = \emptyset$, we know that $\text{oc}(G) = 0$ (*i.e.* G has even order). Hence, if F has at most $|X|$ odd components, then it follows that G has a perfect matching, a contradiction. But, if F has more than $|X|$ odd components, then letting $S = X$ implies that $\text{oc}(G - S) > |S|$, again contradicting the choice of G . Our original supposition that G does not have a perfect matching is therefore false. ■

The following parity lemma will prove useful.

Lemma 9.15 *If $G = (V, E)$ is a graph with an even number of vertices and $S \subseteq V$, then $|S|$ and $\text{oc}(G - S)$ have the same parity.*

Proof Since $|V|$ is even, $|S|$ and $|V \setminus S|$ have the same parity. But the parity of $|V \setminus S|$ is equal to the parity of the number of odd components in $G - S$. ■

The following theorem is an interesting consequence of [Tutte's Theorem](#), due to [Petersen](#) [39].

Theorem 9.16 ([Petersen's Theorem](#)) *A connected cubic graph with at most two bridges has a perfect matching.*

Proof Let $G = (V, E)$ be a connected cubic graph with at most two bridges. Since every graph has an even number of odd vertices (by [Corollary 1.2](#)), $|V|$ is even. Suppose G has no perfect matching. Then, by [Tutte's Theorem](#), there is a subset S of V such that $\text{oc}(G - S) > |S|$. As $|S|$ and $\text{oc}(G - S)$ have the same parity, it follows from [Lemma 9.15](#) that $\text{oc}(G - S) \geq |S| + 2$.

Since G is connected, each component of $G - S$ is joined to S by at least one edge. Since G is cubic, no odd component of $G - S$ is joined to S by exactly two edges (for otherwise such a component would have an odd number of odd vertices, contradicting Corollary 1.2). If an odd component of $G - S$ is joined to S by exactly one edge, then this edge must be a bridge. Since G has at most two bridges, it follows that at most two odd components of $G - S$ are joined to S by exactly one edge while all the other odd components of $G - S$ are joined to S by at least three edges. Hence the odd components are joined to S by at least

$$3(\text{oc}(G - S) - 2) + 2 \geq 3|S| + 2$$

edges. But since G is cubic, the number of edges in G incident with a vertex in S is at most $3|S|$, a contradiction. We conclude that G has a perfect matching. ■

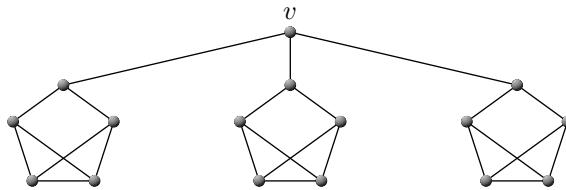


Figure 9.3: A cubic graph $G_{9.4}$ with no perfect matching.

The result of Petersen's Theorem (Theorem 9.16) is best possible since there exist connected cubic graphs containing three bridges that do not have perfect matchings. For example, the connected cubic graph $G_{9.4}$ in Figure 9.3 has three bridges but no perfect matching (letting $S = \{v\}$, we note that $\text{oc}(G_{9.4} - S) = 3 > 1 = |S|$ and so by Tutte's Theorem, $G_{9.4}$ has no perfect matching).

- ❖ The reader should now be able to attempt Exercises 9.13–9.15.

9.5 The Tutte-Berge formula

In this section we present an important max-min formula, called the **Tutte-Berge Formula**, for the matching number of a graph. We begin with the following useful result.

Lemma 9.17 *Let G be a graph and let X be an arbitrary proper subset of vertices of G . If M is a matching in G , then there are at least $\text{oc}(G - X) - |X|$ vertices of G that are M -unmatched.*

Proof Let M_X be the edges of M that belong to $G - X$. Every odd component of $G - X$ contains at least one vertex that is M_X -unmatched. Let U contain exactly one M_X -unmatched vertex from each odd component of $G - X$. Then $|U| = \text{oc}(G - X)$. Let U_X be the subset of vertices in U that are M -matched. Since each vertex in U_X is M -matched with a vertex of X , it follows that $|U_X| \leq |X|$. Thus the number of M -unmatched vertices of G is at least $|U \setminus U_X| = |U| - |U_X| \geq \text{oc}(G - X) - |X|$. ■

As an immediate consequence of Lemma 9.17, we have the following simple upper bound on the matching number of a graph.

Lemma 9.18 *For every graph $G = (V, E)$,*

$$\alpha'(G) \leq \min_{X \subset V} \frac{1}{2}(|V| + |X| - \text{oc}(G - X)).$$

Proof Let M be a maximum matching in G , and so $\alpha'(G) = |M|$. Let $X \subset V$ be an arbitrary proper subset of vertices of G . By Lemma 9.17, at least $\text{oc}(G - X) - |X|$ vertices are M -unmatched, and so $\alpha'(G) = |M| \leq \frac{1}{2}(|V| - (\text{oc}(G - X) - |X|)) = \frac{1}{2}(|V| + |X| - \text{oc}(G - X))$. ■

Berge [3] showed that the upper bound of Lemma 9.18 is, in fact, always an equality. The following max-min formula for the matching number of a graph, due to Berge, is referred to as the *Tutte-Berge Formula* for the matching number.

Theorem 9.19 (Tutte-Berge Formula) *For every graph $G = (V, E)$,*

$$\alpha'(G) = \max |M| = \min_{X \subset V} \frac{1}{2}(|V| + |X| - \text{oc}(G - X)),$$

where the maximum is taken over all maximum matchings in G .

Proof Let $\psi(G)$ denote the expression on the right hand side of the formula in the statement of the theorem. We show that $\alpha'(G) = \psi(G)$. We proceed by induction on the order $n \geq 1$ of G . When $n = 1$, $\alpha'(G) = 0$ and $\psi(G) = \frac{1}{2}(|V| - \text{oc}(G)) \leq \frac{1}{2}(1 - 1) = 0$. This establishes the base case. Assume, as induction hypothesis, that $n \geq 2$ and that $\alpha'(G') = \psi(G')$ for all graphs G' of order less than n . Now, let $G = (V, E)$ be a graph of order n .

Suppose that G is a disconnected graph with components G_1, G_2, \dots, G_k , where $k \geq 2$. Applying the induction hypothesis to each component of G , we have that $\alpha'(G_i) = \psi(G_i)$ for each i . Hence, for each $i \in [k]$, there is a proper subset $X_i \subset V(G_i)$ such that $\alpha'(G_i) = \frac{1}{2}(|V(G_i)| + |X_i| - \text{oc}(G_i - X_i))$. Letting $X^* = \bigcup_{i=1}^k X_i$, we have that $X^* \subset V$ and that

$$\alpha'(G) = \sum_{i=1}^k \alpha'(G_i) = \frac{1}{2}(|V| + |X^*| - \text{oc}(G - X^*)),$$

whence $\alpha'(G) \geq \psi(G)$. By Lemma 9.18, $\alpha'(G) \leq \psi(G)$. Consequently, $\alpha'(G) = \psi(G)$. We may therefore assume that G is a connected graph, for otherwise the desired result holds.

On the one hand, suppose that G contains a vertex v that is matched by *every* maximum matching in G . If $\alpha'(G - v) = \alpha'(G)$, then there would exist a maximum matching in G that leaves v unmatched, a contradiction. Hence, $\alpha'(G - v) = \alpha'(G) - 1$. Let $V' = V \setminus \{v\}$. Applying the induction hypothesis to $G - v$, we have that $\alpha'(G - v) = \psi(G - v)$. Hence, there exists a proper subset $X' \subset V'$ such that $\alpha'(G - v) = \frac{1}{2}(|V'| + |X'| - \text{oc}(G - v - X'))$. Let $X = X' \cup \{v\}$. Then $\alpha'(G) = \alpha'(G - v) + 1 = \frac{1}{2}((|V| - 1) + (|X| - 1) - \text{oc}(G - X)) + 1 = \frac{1}{2}(|V| + |X| - \text{oc}(G - X))$, whence $\alpha'(G) \geq \psi(G)$. Consequently, $\alpha'(G) = \psi(G)$ by Lemma 9.18.

Suppose next that, for every vertex v in G , there is a maximum matching that does not match v . We show that G has an almost perfect matching. Suppose, to the contrary, that G does not have an almost perfect matching. Then each maximum matching of G leaves at least two vertices unmatched. Among all maximum

matchings of G , let M be one for which the distance between two M -unmatched vertices is a minimum. Let u and v be two distinct M -unmatched vertices at minimum distance apart. If $uv \in E$, then the edge uv can be added to M to obtain a larger matching, contradicting the maximality of M . Hence, $d(u, v) \geq 2$. Let x be an internal vertex on a shortest u - v path.

Among all maximum matchings of G that do not match x , let M_1 be chosen so that $|M_1 \cap M|$ is a maximum. If u is an M_1 -unmatched vertex, then our choice of the matching M would be contradicted since $d(u, x) < d(u, v)$. Both u and v are therefore M_1 -matched vertices. Since $|M| = |M_1|$, there exists a second vertex y , different from x , that is M_1 -unmatched but is M -matched. By the minimality of the distance $d(u, v)$, we have that $d(x, y) \geq d(u, v) \geq 2$. Let z be a neighbour of y . Then $x \neq z$. If z is an M_1 -unmatched vertex, then the edge yz can be added to M_1 to obtain a larger matching, contradicting the maximality of M_1 . Hence, z is an M_1 -matched vertex. Let f be the edge of M_1 incident with z , and let e be the edge of M incident with y , so that $e = yz$. Furthermore, let $M_2 = (M_1 \setminus \{f\}) \cup \{e\}$. Then M_2 is a maximum matching of G that does not match x but such that $|M_2 \cap M| > |M_1 \cap M|$, contradicting our choice of the matching M_1 . We conclude, therefore, that G has an almost perfect matching. Thus, $\alpha'(G) = \frac{1}{2}(|V| - 1)$ and since G is connected, $\text{oc}(G) = 1$. Hence, setting $X = \emptyset$, it follows that $\alpha'(G) = \frac{1}{2}(|V| - 1) = \frac{1}{2}(|V| - \text{oc}(G)) = \frac{1}{2}(|V| + |X| - \text{oc}(G - X))$, whence $\alpha'(G) \geq \psi(G)$. Consequently, $\alpha'(G) = \psi(G)$ by Lemma 9.18. ■

As an immediate consequence of Lemmas 9.17 and 9.18, as well as Theorem 9.19, we have the following result.

Corollary 9.20 *Let $G = (V, E)$ be a graph and let X be a proper subset of the vertex set of G such that $(|V| + |X| - \text{oc}(G - X))/2$ is minimum. If M is a maximum matching in G , then $|M| = (|V| + |X| - \text{oc}(G - X))/2$ and there are exactly $\text{oc}(G - X) - |X|$ vertices that are M -unmatched. Furthermore, if M_X is the subset of edges of M that belong to $G - X$, then every vertex in $G - X$ is M_X -matched, except for exactly one vertex from each odd component of $G - X$. If U denotes this set of $\text{oc}(G - X)$ vertices that are M_X -unmatched, one from each odd component of $G - X$, then X is M -matched to a subset of vertices in U .*

Using the Tutte-Berge Formula, we can establish lower bounds on the size of a maximum matching in a regular graph. The matching number of a connected 2-regular graph is $\alpha'(C_n) = \lfloor \frac{n}{2} \rfloor$. Biedl *et al.* [4] established the following lower bound on the matching number of a connected cubic graph. The proof we present is due to Henning and Yeo [23]. Recall that if X and Y are vertex disjoint subsets in a graph, then $[X, Y]$ denotes the set of edges between X and Y in the graph.

Theorem 9.21 *If $G = (V, E)$ is a connected cubic graph G of order n , then*

$$\alpha'(G) \geq \frac{4n - 1}{9},$$

and this bound is tight.

Proof Let $X \subset V$ be a subset which minimises $(n + |X| - \text{oc}(G - X))/2$. By Lemma 9.17, $\alpha'(G) = (n + |X| - \text{oc}(G - X))/2$. If $X = \emptyset$, then $\alpha'(G) \geq \frac{1}{2}(n - 1) > \frac{4}{9}(n - 1)$, and the desired result holds. Hence, we may assume that $X \neq \emptyset$. Let Y

be the set of all vertices that belong to an odd component in $G - X$. If F is an odd component in $G - X$ that is joined to X by s edges, then $2|E(F)| = 3|V(F)| - s$, which implies that s is odd, since $|V(F)|$ is odd. Hence every odd component in $G - X$ is joined to X by an odd number of edges. We shall call an odd component in $G - X$ that is joined to X by exactly one edge in G a *special odd component* of $G - X$, and we call the vertex in a special odd component of $G - X$ that is joined to a vertex in X an *X -special vertex* of $G - X$. Let r be the number of special odd components of $G - X$. All odd components in $G - X$, if any, that are not special are joined to X by at least three edges in G . Hence, $3|X| \geq |[X, Y]| \geq r + 3(\text{oc}(G - X) - r)$, and so $\text{oc}(G - X) - |X| \leq 2r/3$. Thus,

$$\alpha'(G) = \frac{1}{2}(n + |X| - \text{oc}(G - X)) \geq \frac{1}{2}(n - \frac{2r}{3}) = \frac{n}{2} - \frac{r}{3}.$$

We show next that $r \leq (n + 2)/6$. Let R be a special odd component in $G - X$. Then R has exactly one vertex of degree 2, namely the X -special vertex that belongs to R , with all other vertices in R of degree 3. Both neighbours of the X -special vertex in R have degree 3, and so, since $R \neq K_4$, it follows that R has at least five vertices. Let H be the graph obtained from G by deleting from G all vertices that belong to special odd components in $G - X$, except for the r X -special vertices in $G - X$. Since G is a connected graph, the graph H is a connected graph by construction, and so $|E(H)| \geq |V(H)| - 1$. Every vertex of H has degree 3 in H , except for the r X -special vertices in $G - X$, each of which has degree 1 in H . Hence, $2|E(H)| = r + 3(|V(H)| - r)$, and so $3|V(H)| - 2r = 2|E(H)| \geq 2|V(H)| - 2$. This implies that $2r \leq |V(H)| + 2$. However, $|V(H)| \leq n - 4r$ since we deleted at least four vertices from each special odd component in $G - X$, implying that $2r \leq n - 4r + 2$, or, equivalently, $r \leq (n + 2)/6$, as claimed. Hence,

$$\alpha'(G) \geq \frac{n}{2} - \frac{r}{3} \geq \frac{n}{2} - (n + 2)/18 = (4n - 1)/9.$$

This establishes the desired lower bound. ■

Before we discuss the tightness of the bound in [Theorem 9.21](#), we turn our attention to the following question: What can be said about the matching number of connected k -regular graphs for some fixed integer $k \geq 4$? Using the [Tutte-Berge Formula](#), we are fortunately able to generalise our earlier results for 2-regular graphs and 3-regular graphs, although we need to distinguish the case where $k \geq 4$ is even from the case where $k \geq 3$ is odd.

We note the following trivial lemma which will prove to be useful.

Lemma 9.22 *Let $k > 1$ be an integer and let G be a graph of order n with maximum degree at most k . If*

$$\sum_{v \in V(G)} (k - d_G(v)) < k,$$

then $n \geq k + 1$.

Proof As the maximum degree of G is at most $n - 1$, we note that

$$\sum_{v \in V(G)} (k - d_G(v)) \geq n(k - (n - 1)).$$

By assumption, the left-hand sum is strictly less than k , whence $n(k+1-n) < k$. As this is equivalent to the inequality $0 < (n-1)(n-k)$, it follows that $n > k$, which implies the desired result. \blacksquare

For $k \geq 3$, we define a k -diamond, denoted by D_k , to be the graph obtained from K_{k+1} by deleting one edge, that is, $D_k = K_{k+1} - e$ where e is an edge of K_{k+1} . We note that a 3-diamond is the diamond $K_4 - e$. We define an even diamond as a k -diamond for some even integer $k \geq 4$. We observe that for $k \geq 4$, all vertices of the diamond D_k have degree k , except for the two nonadjacent vertices of degree $k-1$, which we call the *links* of the diamond.

For $k \geq 4$ even, Henning and Shozi [22] constructed a family \mathcal{H}_k of connected k -regular graphs as follows. For $\ell \geq 1$ an integer, let $B_{k,\ell}$ be a connected bipartite graph of order $n = (k+2)\ell$ with partite sets X and Y , where $|X| = 2\ell$ and $|Y| = k\ell$, and where every vertex in X has degree k and every vertex in Y has degree 2. Let $H_{k,\ell}$ be obtained from $B_{k,\ell}$ by replacing every vertex $v \in Y$, whose two neighbours in X we label v_1 and v_2 , with a copy of a diamond D_k , where we join the one link vertex of the added diamond to v_1 and the other link vertex to v_2 . We call the graph $B_{k,\ell}$ the *underlying bipartite graph* of $H_{k,\ell}$. Let $\mathcal{H}_k = \{H_{k,\ell} \mid \ell \geq 1\}$. An example of the graph $H_{4,1}$ that belongs to the family \mathcal{H}_4 is illustrated in Figure 9.4, where $H_{4,1}$ is constructed from the underlying bipartite graph $B_{4,1} \cong K_{2,4}$.

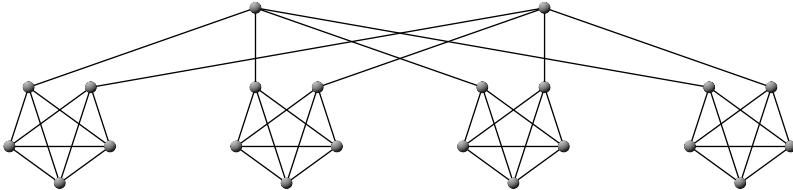


Figure 9.4: The graph $H_{4,1}$ in the family \mathcal{H}_4 .

The following lower bound on the matching number of a connected k -regular graph, where $k \geq 4$ is even. This result is due to Henning and Yeo [23]. From their proof, one can readily obtain a characterisation of the extremal graphs, as observed by Henning and Shozi [22].

Theorem 9.23 *If G is a connected k -regular graph of order n for some even integer $k \geq 4$ and $\alpha'(G) < \frac{1}{2}(n-1)$, then*

$$\alpha'(G) \geq \left(\frac{k^2 + 4}{k^2 + k + 2} \right) \times \frac{n}{2}, \quad (9.1)$$

with equality if and only if $G \in \mathcal{H}_k$.

Proof For $k \geq 4$ an even integer, let G be a connected k -regular graph of order n such that $\alpha'(G) < \frac{1}{2}(n-1)$. Let X be a set of vertices in G that yield equality in the Tutte-Berge Formula given in Theorem 9.19. Thus,

$$\alpha'(G) = \frac{1}{2}(n + |X| - \text{oc}(G - X)).$$

If $X = \emptyset$, then $\alpha'(G) \geq (n - 1)/2$, contradicting our supposition that $\alpha'(G) < \frac{1}{2}(n - 1)$. Hence, $X \neq \emptyset$. Since every graph has an even number of vertices of odd degree, there is no odd component in $G - X$ that is joined in G to X by exactly one edge, as such a component would have only one vertex of odd degree. Therefore, every odd component in $G - X$ is joined in G to X by at least two edges. Let $Y = V(G) \setminus X$. Also, let y_k denote the number of odd components in $G - X$ that are joined in G to X by at least k edges, and let y_2 denote the number of odd components in $G - X$ that are joined in G to X by fewer than k edges. Thus, $\text{oc}(G - X) = y_2 + y_k$. If $[X, Y]$ denotes the number of edges between X and Y , then by double counting we have

$$k|X| \geq |[X, Y]| \geq 2y_2 + ky_k,$$

and so

$$|X| \geq y_k + \frac{2y_2}{k} \geq \frac{2y_2}{k}. \quad (9.2)$$

We show next that every odd component of $G - X$ with fewer than k edges into X in G must contain at least $k+1$ vertices. Let G_X be such an odd component of $G - X$ with fewer than k edges into X in G . If $V_X = V(G_X)$, then

$$k|V_X| = \sum_{v \in V_X} d_G(v) < k + \sum_{v \in V_X} d_{G_X}(v)$$

and so

$$\sum_{v \in V_X} (k - d_{G_X}(v)) < k.$$

It therefore follows by Lemma 9.22 that $|V_X| \geq k+1$. Hence, every odd component of $G - X$ with fewer than k edges into X in G contains at least $k+1$ vertices. This implies that

$$n \geq |X| + y_2(k+1). \quad (9.3)$$

As observed earlier, $\text{oc}(G - X) = y_2 + y_k$. Therefore,

$$\begin{aligned} \alpha'(G) &= \frac{1}{2}(n + |X| - \text{oc}(G - X)) \\ &= \left(\frac{k^2 + 4}{k^2 + k + 2} \times \frac{n}{2} \right) + \left(\frac{k - 2}{k^2 + k + 2} \times \frac{n}{2} \right) + \left(\frac{|X| - (y_2 + y_k)}{2} \right) \\ &\geq \left(\frac{k^2 + 4}{k^2 + k + 2} \times \frac{n}{2} \right) + \left(\frac{k - 2}{k^2 + k + 2} \times \frac{|X| + y_2(k+1)}{2} \right) + \left(\frac{\frac{2y_2}{k} - y_2}{2} \right) \\ &\geq \left(\frac{k^2 + 4}{k^2 + k + 2} \times \frac{n}{2} \right) + \left(\frac{k - 2}{k^2 + k + 2} \times \frac{\frac{2y_2}{k} + y_2(k+1)}{2} \right) - y_2 \left(\frac{k - 2}{2k} \right) \\ &= \left(\frac{k^2 + 4}{k^2 + k + 2} \times \frac{n}{2} \right) + \left(\frac{y_2(k-2)}{2k(k^2 + k + 2)} \times (k^2 + k + 2 - (k^2 + k + 2)) \right) \\ &= \left(\frac{k^2 + 4}{k^2 + k + 2} \right) \times \frac{n}{2}. \end{aligned}$$

This establishes the lower bound, namely inequality (9.1), in the statement of Theorem 9.23. Suppose that G achieves equality in inequality (9.1), that is,

$$\alpha'(G) = \left(\frac{k^2 + 4}{k^2 + k + 2} \right) \times \frac{n}{2}.$$

We must have equality throughout the above inequality chain, implying that we must have equality in both inequalities (9.2) and (9.3). Equality in inequality (9.2) implies that X is an independent set and every vertex in X has degree k . Furthermore, $\text{oc}(G - X) = y_2$ and every odd component in $G - X$ is joined to X by exactly two edges. Equality in inequality (9.3) implies that every odd component in $G - X$ has order exactly $k + 1$. Moreover, every vertex belongs to X or to an odd component of G . These properties of the graph G imply that every component of $G - X$ is an odd component and is isomorphic to an even diamond D_k . By the way in which the family \mathcal{H}_k is constructed, this in turn implies that $G \in \mathcal{H}_k$. This completes the characterisation of the extremal graphs achieving equality in inequality (9.1). ■

We note that if G is a connected 2-regular graph, then $G = C_n$ for some $n \geq 3$ and $\alpha'(G) = \frac{1}{2}(n - 1)$ if n is odd, while $\alpha'(G) = \frac{1}{2}n$ if n is even. Hence, as an immediate consequence of [Theorem 9.23](#), we have the following lower bound on the matching number of a connected k -regular graph, where $k \geq 2$ is even.

Corollary 9.24 *If G is a connected k -regular graph of order n for some even integer $k \geq 2$, then*

$$\alpha'(G) \geq \min\left\{\left(\frac{k^2 + 4}{k^2 + k + 2}\right)\frac{n}{2}, \frac{n - 1}{2}\right\},$$

and this bound is tight.

For small values of $k \geq 2$ even, the lower bounds in [Corollary 9.24](#) are summarised in [Table 9.1](#).

$r \rightarrow$	1	2	3	4
$\alpha'(G) \geq$	$\frac{n - 1}{2}$	$\min\left\{\frac{5n}{11}, \frac{n - 1}{2}\right\}$	$\min\left\{\frac{5n}{11}, \frac{n - 1}{2}\right\}$	$\min\left\{\frac{17n}{37}, \frac{n - 1}{2}\right\}$

Table 9.1: Lower bounds on $\alpha'(G)$ for a connected $2r$ -regular graph G of order n .

The $(n - 1)/2$ bound in the statement of [Corollary 9.24](#) is only included as it is necessary when n is very small or if $k = 2$. As a consequence of [Corollary 9.24](#), we have the following slightly weaker result. We leave the proof of this result as an exercise (see [Exercise 9.16](#)).

Corollary 9.25 *If G is a connected k -regular graph of order n for some even integer $k \geq 4$, then*

$$\alpha'(G) \geq \left\lfloor \left(\frac{k^2 + 4}{k^2 + k + 2}\right)\frac{n}{2} \right\rfloor,$$

and this bound is tight.

We next turn our attention to tight lower bounds on the matching number of a connected k -regular graph, where $k \geq 3$ is odd. Theorem 9.21 presented earlier covers the special case where $k = 3$. The lower bound on the matching number of a connected 3-regular graph given in Theorem 9.21 can be generalised to connected k -regular graphs, where $k \geq 3$ is odd. We state this result without proof, suffice to say that the proof of [Theorem 9.26](#) presented in 2007 by [Henning and Yeo \[23\]](#) relies heavily on the [Tutte-Berge Formula](#).

Theorem 9.26 *If G is a connected k -regular graph of order n for some odd integer $k \geq 3$, then*

$$\alpha'(G) \geq \frac{n(k^3 - k^2 - 2) - 2k + 2}{2(k^3 - 3k)}.$$

If $k \geq 3$ is an odd integer, then $k = 2r+1$ for some integer $r \geq 1$. Performing this substitution into the statement of [Theorem 9.26](#) yields the equivalent statement below.

Theorem 9.27 *If G is a connected $(2r+1)$ -regular graph of order n for some natural number r , then*

$$\alpha'(G) \geq \frac{n(4r^3 + 4r^2 + r - 1) - 2r}{2(2r+1)(2r^2 + 2r - 1)}.$$

For small values of r , the lower bounds in [Theorem 9.27](#) are summarised in [Table 9.2](#).

$r \rightarrow$	1	2	3	4
$\alpha'(G) \geq$	$\frac{4n-1}{9}$	$\frac{49n-4}{110}$	$\frac{73n-3}{161}$	$\frac{323n-8}{702}$

Table 9.2: Lower bounds on $\alpha'(G)$ for a connected $(2r+1)$ -regular graph G of order n .

The (infinite) family of graphs achieving equality in the lower bound in [Theorem 9.27](#) were characterised in 2010 by [O and West \[36\]](#). For this purpose, they defined a *balloon* B_r to be the graph obtained from K_{2r+3} by deleting a matching of size $r+1$ and one more edge incident to the remaining vertex for some integer r . Thus, B_r is the graph obtained from K_{2r+3} by removing from it the edges of a spanning subgraph isomorphic to $P_3 \cup rP_2$. The balloon B_1 is illustrated in [Figure 9.5](#).

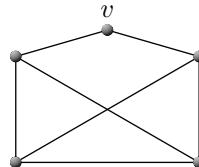


Figure 9.5: The balloon B_1 .

For $r \geq 1$, all vertices of a balloon B_r have degree $2r+1$, except for one vertex of degree $2r$, called the *neck* of the balloon. To illustrate this definition, the vertex v

in [Figure 9.5](#) is the neck of the balloon B_1 . We note that deleting the neck of a balloon B_r leaves a subgraph having a *perfect matching*. If $G = B_r$, then G has maximum degree $2r + 1$, order $n = 2r + 3$, and matching number $\alpha'(G) = r + 1$.

For $r \geq 1$, let \mathcal{T}_{2r+1} be the family of trees such that every nonleaf has degree $2r + 1$ and all leaves have the same colour in a proper 2-colouring of the vertices. For example, when $r = 1$ a red-blue coloring of the vertices of a tree in the family \mathcal{T}_{2r+1} is illustrated in [Figure 9.6](#) with all leaves of T having the same colour, namely blue. Equivalently, \mathcal{T}_{2r+1} is the family of trees with bipartite sets X and Y , where every leaf belongs to Y and every nonleaf has degree $2r + 1$. For the tree shown in [Figure 9.6](#), the bipartite set X contains the red vertices and the bipartite set Y contains the blue vertices.

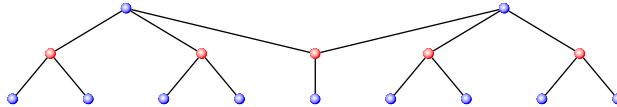


Figure 9.6: A tree in the family \mathcal{T}_3 .

For $r \geq 1$, let \mathcal{G}_{2r+1} be the family of $(2r + 1)$ -regular graphs obtained from trees in the family \mathcal{T}_{2r+1} by identifying each leaf of such a tree with the neck in a copy of the balloon B_r . Such a tree in \mathcal{T}_{2r+1} used to construct a graph $G \in \mathcal{G}_{2r+1}$, we call the *underlying tree* of G .

For example, if $r = 1$ and $T \in \mathcal{T}_{2r+1}$ is the tree illustrated in [Figure 9.6](#), then the graph $G \in \mathcal{G}_{2r+1}$ with underlying tree T is illustrated in [Figure 9.7](#). In this example, the edges in bold face form a matching, and so $\alpha'(G) \geq 23$. If X denotes the set of red vertices illustrated in [Figure 9.7](#), then $|X| = 5$ and $\text{oc}(G - X) = 11$. Thus, by [Lemma 9.18](#), $\alpha'(G) \leq \frac{1}{2}(n + |X| - \text{oc}(G - X)) = \frac{1}{2}(52 + 5 - 11) = 23$. Consequently, $\alpha'(G) = 23 = (4n - 1)/9$.

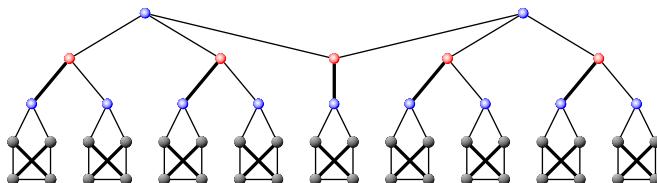


Figure 9.7: A graph in the family \mathcal{G}_3 .

More generally, we show that every graph in the family \mathcal{G}_{2r+1} achieves equality in the lower bound of [Theorem 9.27](#).

Proposition 9.28 *Let r be a natural number. If $G \in \mathcal{G}_{2r+1}$ has order n , then*

$$\alpha'(G) = \frac{n(4r^3 + 4r^2 + r - 1) - 2r}{2(2r + 1)(2r^2 + 2r - 1)}.$$

Proof Let $G \in \mathcal{G}_{2r+1}$ and let T be the underlying tree of G . Let X and Y be the bipartite sets of T , where every leaf belongs to Y and every nonleaf has degree $2r + 1$. Let Y_1 be the set of leaves of T , and let $Y_2 = Y \setminus Y_1$. Thus, every vertex in $X \cup Y_2$ has degree $2r + 1$ in T . Let $|X| = x$, $|Y_1| = y_1$ and $|Y_2| = y_2$, and so

$n(T) = x + y_1 + y_2$. Every edge of the tree T is incident with a vertex of Y , and so $x + y_1 + y_2 - 1 = n(T) - 1 = m(T) = y_1 + (2r + 1)y_2$, or, equivalently,

$$y_2 = \frac{x - 1}{2r}. \quad (9.4)$$

Double counting the edges in T , we have $(2r + 1)x = m(T) = y_1 + (2r + 1)y_2$. Substituting the value for y_2 in (9.4), we obtain

$$y_1 = \left(\frac{2r + 1}{2r} \right) ((2r - 1)x + 1). \quad (9.5)$$

Let G have order n , so that $n = x + (2r + 3)y_1 + y_2$. Moreover, let

$$\Phi(G) = \frac{(4r^3 + 4r^2 + r - 1)x + (2r^2 + 3r + 1)}{2r}.$$

The graph $G - X$ contains $y_1 + y_2$ odd components. Therefore, by Lemma 9.18 and by (9.5), we have

$$\begin{aligned} \alpha'(G) &\leq \frac{1}{2}(n + |X| - \text{oc}(G - X)) \\ &= \frac{1}{2}((x + (2r + 3)y_1 + y_2) + x - (y_1 + y_2)) \\ &= \frac{1}{2}(2x + (2r + 2)y_1) \\ &= x + (r + 1)y_1 \\ &= \Phi(G). \end{aligned}$$

By (9.4) and (9.5), Theorem 9.27 yields the lower bound

$$\begin{aligned} \alpha'(G) &\geq \frac{(4r^3 + 4r^2 + r - 1)n - 2r}{2(2r + 1)(2r^2 + 2r - 1)} \\ &= \frac{(4r^3 + 4r^2 + r - 1)(x + (2r + 3)y_1 + y_2) - 2r}{2(2r + 1)(2r^2 + 2r - 1)} \\ &= \frac{(4r^3 + 4r^2 + r - 1)[x + (2r + 3)\left(\frac{2r+1}{2r}\right)((2r - 1)x + 1) + \frac{x-1}{2r}] - 2r}{2(2r + 1)(2r^2 + 2r - 1)} \\ &= \Phi(G) \end{aligned}$$

on the matching number of G . By the above inequalities, it follows that $\Phi(G) \leq \alpha'(G) \leq \Phi(G)$. Consequently, $\alpha'(G) = \Phi(G)$, and so we must have equality throughout the above inequality chains. In particular, this implies that we must have equality in the lower bound of Theorem 9.27. ■

By Proposition 9.28, every graph G that belongs to the family \mathcal{G}_{2r+1} , for any natural number r , achieves equality in the lower bound of Theorem 9.27. O and West [36] showed that the family \mathcal{G}_{2r+1} is precisely the family of connected graphs that achieve equality in the lower bound of Theorem 9.27 when the graphs are $(2r + 1)$ -regular. We omit their proof.

Theorem 9.29 *Let r be a natural number. If G is a connected $(2r + 1)$ -regular graph of order n satisfying*

$$\alpha'(G) = \frac{(4r^3 + 4r^2 + r - 1)n - 2r}{2(2r + 1)(2r^2 + 2r - 1)},$$

then $G \in \mathcal{G}_{2r+1}$.

Combining the results of Theorems 9.27 and 9.29, and Proposition 9.28, we have the following result.

Theorem 9.30 *If G is a connected k -regular graph of order n for some odd integer $k \geq 3$, then*

$$\alpha'(G) \geq \frac{(k^3 - k^2 - 2)n - 2k + 2}{2(k^3 - 3k)},$$

with equality if and only if $G \in \mathcal{G}_k$.

As remarked earlier, we omit the proofs of Theorems 9.27 and 9.29. We do, however, prove here the following weaker result than Theorem 9.27.

Theorem 9.31 *If G is a connected k -regular graph of order n with $k \geq 1$ odd, then*

$$\alpha'(G) \geq \frac{n}{2} \left(\frac{k^2 + k + 2}{k^2 + 2k + 1} \right),$$

and this bound is tight.

Proof If $k = 1$, then n is even and $G = \frac{n}{2}K_2$. Therefore, $\alpha'(G) = \frac{n}{2}$, and the desired result holds. Hence, we may assume that $k \geq 3$. Let X be a set of vertices in G such that $(n + |X| - \text{oc}(G - X))/2$ is a minimum and let M be a maximum matching in G . By Corollary 9.20 we note that $|M| = (n + |X| - \text{oc}(G - X))/2$. If $X = \emptyset$, then since every graph has an even number of vertices of odd degree according to Corollary 1.2, n is even and $|M| = \frac{n}{2}$, and the desired result holds. Therefore, we may assume that $X \neq \emptyset$.

Let y_k denote the number of odd components in $G - X$ that have at least k edges into X in G and let y_1 denote the number of odd components in $G - X$ that have fewer than k edges into X in G . If there are $d(X, V - X)$ edges between X and $V(G) - X$, then $k|X| \geq d(X, V - X) \geq y_1 + ky_k$. Hence, $|X| \geq y_k + \frac{y_1}{k} \geq \frac{y_1}{k}$.

By Lemma 9.22, we note that every odd component in $G - X$ with fewer than k edges into X in G contains at least $k + 1$ vertices, and since $k + 1$ is even it must, in fact, contain at least $k + 2$ vertices. This implies that $n \geq |X| + y_1(k + 2)$. Therefore,

$$\begin{aligned} |M| &= \frac{1}{2}(n + |X| - \text{oc}(G - X)) \\ &= \frac{n}{2} \left(\frac{k^2 + k + 2}{k^2 + 2k + 1} \right) + \left(\frac{k-1}{k^2 + 2k + 1} \times \frac{n}{2} \right) + \left(\frac{|X| - (y_1 + y_k)}{2} \right) \\ &\geq \frac{n}{2} \left(\frac{k^2 + k + 2}{k^2 + 2k + 1} \right) + \left(\frac{k-1}{k^2 + 2k + 1} \times \frac{|X| + y_1(k+2)}{2} \right) + \left(\frac{\frac{y_1}{k} - y_1}{2} \right) \\ &\geq \frac{n}{2} \left(\frac{k^2 + k + 2}{k^2 + 2k + 1} \right) + \left(\frac{k-1}{k^2 + 2k + 1} \times \frac{y_1/k + y_1(k+2)}{2} \right) - y_1 \left(\frac{k-1}{2k} \right) \\ &= \frac{n}{2} \left(\frac{k^2 + k + 2}{k^2 + 2k + 1} \right) + \frac{y_1(k-1)}{2k(k^2 + 2k + 1)} (k^2 + 2k + 1 - (k^2 + 2k + 1)) \\ &= \frac{n}{2} \left(\frac{k^2 + k + 2}{k^2 + 2k + 1} \right). \end{aligned}$$

■

As a special case of Theorem 9.31 when $k = 3$, we have the following result.

Corollary 9.32 *If G is a cubic graph of order n , then $\alpha'(G) \geq 7n/16$.*

We remark that relaxing the regularity constraint, the lower bound in [Theorem 9.30](#) was generalised in 2018 by [Henning and Yeo \[25\]](#) who established the following tight lower bound on the matching number in a graph with given maximum odd degree. We omit the proof of this result.

Theorem 9.33 *If G is a connected k -regular graph of order n , size m , and with maximum degree $\Delta(G) \leq k$ for some odd integer $k \geq 3$, then*

$$\alpha'(G) \geq \left(\frac{k-1}{k(k^2-3)} \right) n + \left(\frac{k^2-k-2}{k(k^2-3)} \right) m - \frac{k-1}{k(k^2-3)}.$$

The extremal graphs achieving equality in the lower bound of [Theorem 9.33](#) were characterised by [Henning and Shozi \[21, 20\]](#). We remark that for each $k \geq 3$ odd, the lower bounds in [Theorem 9.33](#) are tight for essentially all densities of graphs. In particular, these lower bounds are achieved for infinitely many trees, and for infinitely many k -regular graphs. Moreover, in the special case where $k = 3$, the result of [Theorem 9.33](#) yields a lower bound on the matching number of a subcubic graph first established in 2017 by [Haxell and Scott \[19\]](#).

Lower bounds on the matching number of a graph with given maximum even degree are very different from lower bounds in the case of given maximum odd degree, as shown in 2018 by [Henning and Yeo \[25\]](#). We omit the proof of this result.

Theorem 9.34 *If G is a connected k -regular graph of order n , size m , and with maximum degree $\Delta(G) \leq k$ for some even integer $k \geq 4$, then*

$$\alpha'(G) \geq \frac{n}{k(k+1)} + \frac{m}{k+1} - \frac{1}{k(k+1)},$$

unless the following holds:

(i) G is k -regular and $n = k+1$, in which case

$$\alpha'(G) = \frac{n-1}{2} = \frac{n}{k(k+1)} + \frac{m}{k+1} - \frac{1}{k}.$$

(ii) G is k -regular and $n = k+3$, in which case

$$\alpha'(G) = \frac{n-1}{2} = \frac{n}{k(k+1)} + \frac{m}{k+1} - \frac{3}{k(k+1)}.$$

If G is k -regular and $n = k+1$, then $G = K_{k+1}$, while if G is k -regular and $n = k+3$, then the complement \bar{G} of G is the vertex disjoint union of cycles. [Henning and Shozi \[22\]](#) provided a characterisation of graphs G achieving equality in the lower bound of [Theorem 9.34](#), when G is not one of these two exceptional graphs. We remark that for each $k \geq 4$ even, the lower bounds in [Theorem 9.34](#) are tight for essentially all densities of graphs.

❖ The reader should now be able to attempt [Exercise 9.16](#).

9.6 The binding number of a graph

The concept of the binding number of a graph was introduced by [Woodall \[51\]](#) in 1973 based on an idea of [Anderson \[1\]](#). For a graph $G = (V, E)$, we let

$$\mathcal{X}(G) = \{X \mid \emptyset \neq X \subseteq V \text{ and } N(X) \neq V\}.$$

[Woodall](#) defined the **binding number** of G as

$$\text{bind}(G) = \min_{X \in \mathcal{X}(G)} \frac{|N(X)|}{|X|}.$$

We define a set $X \in \mathcal{X}(G)$ to be a **binding set** of G if $\text{bind}(G) = |N(X)| / |X|$. We begin with several properties of the binding number. The following result is due to [Kane et al. \[26\]](#).

Lemma 9.35 *If H is a spanning subgraph of a graph G , then $\text{bind}(H) \leq \text{bind}(G)$.*

Proof Let X be a binding set of G , and so $\text{bind}(G) = |N(X)| / |X|$. Therefore, $N_H(X) \subseteq N_G(X) \neq V(G)$, and so $X \in \mathcal{X}(H)$. Hence,

$$\text{bind}(H) \leq \frac{|N_H(X)|}{|X|} \leq \frac{|N_G(X)|}{|X|} = \text{bind}(G),$$

as desired. ■

Recall, from [Chapter 1](#), that the join operation of two graphs G and H is denoted by $G + H$.

Lemma 9.36 *If G and H are vertex-disjoint graphs, then*

$$\begin{aligned} \text{bind}(G \cup H) &= \min \left\{ \min_{X \in \mathcal{X}(G)} (|N_G(X)| + |V(H)|) / |X|, \right. \\ &\quad \left. \min_{X \in \mathcal{X}(H)} (|N_H(X)| + |V(G)|) / |X| \right\}. \end{aligned}$$

Proof Let $S \in \mathcal{X}(G + H)$ be a binding set of $G + H$. Then, since $N(S) \neq V(G + H)$, either $S \in \mathcal{X}(G)$ or $S \in \mathcal{X}(H)$. If $S \in \mathcal{X}(G)$, then $\text{bind}(G \cup H) = (|N_G(S)| + |V(H)|) / |S| \geq \min_{X \in \mathcal{X}(G)} (|N_G(X)| + |V(H)|) / |X|$. On the other hand, $\text{bind}(G \cup H) \leq \min_{X \in \mathcal{X}(G)} (|N_G(X)| + |V(H)|) / |X|$. Consequently, if $S \in \mathcal{X}(G)$, then $\text{bind}(G \cup H) = \min_{X \in \mathcal{X}(G)} (|N_G(X)| + |V(H)|) / |X|$. Analogously, if $S \in \mathcal{X}(H)$, then $\text{bind}(G \cup H) = \min_{X \in \mathcal{X}(H)} (|N_H(X)| + |V(G)|) / |X|$. ■

The following result is also due to [Kane et al. \[27\]](#).

Lemma 9.37 *If G and H are vertex-disjoint graphs, then*

$$\text{bind}(G \cup H) = \min\{\text{bind}(G), \text{bind}(H), 1\}.$$

Proof Let X_1 be a binding set of G , let X_2 be a binding set of H , and let $X_3 = V(G)$. Then

$$\begin{aligned} \text{bind}(G \cup H) &\leq \min \left\{ \frac{|N(X_1)|}{|X_1|}, \frac{|N(X_2)|}{|X_2|}, \frac{|N(X_3)|}{|X_3|} \right\} \\ &= \min\{\text{bind}(G), \text{bind}(H), 1\}. \end{aligned}$$

On the other hand, let X be a binding set of $G \cup H$, let $X_G = X \cap V(G)$ and let $X_H = X \cap V(H)$. If $X = X_G$ and $N(X) = V(G)$, then $\text{bind}(G \cup H) = |N(X)| / |X| \geq 1$, while if $X = X_G$ and $N(X) \neq V(G)$, then $\text{bind}(G \cup H) = |N(X)| / |X| \geq \text{bind}(G)$. Hence, if $X = X_G$, then $\text{bind}(G \cup H) \geq \min\{\text{bind}(G), 1\}$. Similarly, if $X = X_H$, then $\text{bind}(G \cup H) \geq \min\{\text{bind}(H), 1\}$. We may therefore assume that both X_G and X_H are nonempty sets. Note that if a, b, c and d are positive integers, then $(a+b)/(c+d) \geq \min\{\frac{a}{c}, \frac{b}{d}\}$. Hence,

$$\begin{aligned} \text{bind}(G \cup H) &= \frac{|N(X)|}{|X|} = \frac{|N(X_G)| + |N(X_H)|}{|X_G| + |X_H|} \\ &\geq \min \left\{ \frac{|N(X_G)|}{|X_G|}, \frac{|N(X_H)|}{|X_H|} \right\} \geq \min\{\text{bind}(G), \text{bind}(H), 1\}. \end{aligned}$$

Consequently, $\text{bind}(G \cup H) = \min\{\text{bind}(G), \text{bind}(H), 1\}$. ■

As an immediate consequence of [Lemma 9.37](#), we have the following result.

Corollary 9.38

- (i) *If G is a disconnected graph, then $\text{bind}(G) \leq 1$.*
- (ii) *If G is the disjoint union of k graphs G_1, G_2, \dots, G_k , then*

$$\text{bind}(G) = \min\{\text{bind}(G_1), \text{bind}(G_2), \dots, \text{bind}(G_k), 1\}.$$

As a consequence of [Lemma 9.37](#) and [Corollary 9.38](#), we also have the following result.

Theorem 9.39 *If a graph G has a perfect matching, then $\text{bind}(G) \geq 1$.*

Proof Let H be the spanning subgraph of G that is induced by the edges of a perfect matching in G . Then G has even order n and $H = \frac{n}{2}K_2$. By [Corollary 9.38\(ii\)](#), $\text{bind}(H) = \min\{\text{bind}(K_2), 1\} = 1$. Thus, by [Lemma 9.35](#), $\text{bind}(G) \geq \text{bind}(H) = 1$. ■

The converse of [Theorem 9.39](#) is not true, however. To see this, consider, for example, the graph $G = (t+2)K_3 + tK_1$, where $t \geq 1$, which has no perfect matching. By [Lemma 9.36](#), it follows that

$$\text{bind}(G) = \frac{3(t+1) + t}{3(t+1)} = \frac{4t+3}{3t+3} \rightarrow \frac{4}{3} \quad \text{as } t \rightarrow \infty.$$

Thus, there are graphs of even order with binding number arbitrarily close to $\frac{4}{3}$ that do not have a perfect matching. [Anderson \[1\]](#), however, established the following major result on perfect matchings.

Theorem 9.40 (Anderson's Theorem) *If G is a graph of even order with $\text{bind}(G) \geq \frac{4}{3}$, then G has a perfect matching.*

We shall prove a generalisation of [Anderson's Theorem](#) due to [Woodall \[51\]](#). Our proof is based on that of [Goddard \[17\]](#). We shall need the following lemma.

Lemma 9.41 *Let $G = (V, E)$ be a graph of order n with $\text{bind}(G) \geq b$. Let $X \subset V$, let $|X| = x$, and let $V_X = V \setminus X$. Furthermore, let Y be the set of isolated vertices*

in $G - X$ and let $|Y| = y$. Then the following four properties hold:

- (i) If $y > 0$, then $y \leq bx - (b-1)n$, which is valid for $y = 0$ if $bx - (b-1)n \geq 0$.
- (ii) If $y > 0$, then $yb \leq x$, which is also valid if $y = 0$.
- (iii) If $G - X$ is disconnected and has a component of order k , then $b \leq 1 + x/(n-x-k)$.
- (iv) $n \geq x + y + 3(\text{oc}(G - X) - y)$.

Proof If $y > 0$, then $b \leq \text{bind}(G) \leq |\text{N}(V_X)| / |V_X| \leq (n-y)/(n-x)$, and so the desired inequality in part (i) holds. If $y > 0$, then $b \leq |\text{N}(Y)| / |Y| \leq \frac{x}{y}$, and the desired inequality in part (ii) holds. Suppose, next, that $G - X$ is disconnected and has a component H of order k . Let $T = V \setminus (X \cup V(H))$. Then, $b \leq |\text{N}(T)| / |T| \leq (n-k)/(n-x-k) = 1 + x/(n-x-k)$, and part (iii) holds. Finally, since every nontrivial odd component of $G - X$ has order at least 3, we have that $n \geq |X| + |Y| + 3(\text{oc}(G - X) - y)$.

We are now in a position to prove Woodall's generalisation of Anderson's Theorem.

Theorem 9.42 (Woodall's Theorem) *Let G be a graph of order n with $\text{bind}(G) \geq b$. Then*

- (i) $\alpha'(G) \geq bn/(b+1)$ if $0 \leq b \leq \frac{1}{2}$.
- (ii) $\alpha'(G) \geq \frac{n}{3}$ if $\frac{1}{2} \leq b \leq 1$.
- (iii) $\alpha'(G) \geq (3b-2)n/3b - 2(b-1)/b$ if $1 \leq b \leq \frac{4}{3}$.

Proof Let $X \subset V$ and let $|X| = x$. Furthermore, let Y be the set of isolated vertices in $G - X$ and let $|Y| = y$. We consider the three cases in turn.

(i) Let $0 \leq b \leq \frac{1}{2}$. Then, by Lemma 9.41(i) and (ii), the two inequalities $y \leq bx - (b-1)n$ and $yb \leq x$ both hold even if $y = 0$. Multiplying the first inequality by $(1-2b)/(1-b^2)$ and the second by $(2-b)/(1-b^2)$, and adding the resultant inequalities together, we obtain

$$y \leq 2x + \left(\frac{1-2b}{1+b} \right) n.$$

Hence, by Lemma 9.41(iv),

$$3\text{oc}(G - X) \leq n - x + 2y \leq 3x + \left(\frac{1-b}{1+b} \right) 3n,$$

and so $\text{oc}(G - X) \leq x + (1-b)n/(1+b)$. Thus,

$$\frac{1}{2}(|V(G)| + |X| - \text{oc}(G - X)) \geq \frac{1}{2} \left(n - \left(\frac{1-b}{1+b} \right) n \right) = \left(\frac{b}{b+1} \right) n.$$

Hence, by the Tutte-Berge Formula (see Theorem 9.19), it follows that $\alpha'(G) \geq bn/(b+1)$. This establishes part (i).

(ii) Let $\frac{1}{2} \leq b \leq 1$. Then, $\text{bind}(G) \geq \frac{1}{2}$, and so, by part (i), $\alpha'(G) \geq \frac{1}{2}n/(\frac{1}{2}+1) = \frac{n}{3}$.

(iii) Let $1 \leq b \leq \frac{4}{3}$. We shall prove that

$$\frac{1}{2}(n + x - \text{oc}(G - X)) \geq \left(\frac{3b-2}{3b} \right) n - \frac{2(b-1)}{b},$$

or, equivalently, that

$$\text{oc}(G - X) \leq x + \left(\frac{4 - 3b}{3b} \right) n + 4 \left(\frac{b - 1}{b} \right), \quad (9.6)$$

which, by the [Tutte-Berge Formula](#), establishes the desired result. We consider two subcases.

Suppose first that $x \geq n(b - 1)/b$. Then, by [Lemma 9.41\(i\)](#), the inequality $y \leq bx - (b - 1)n$ holds even if $y = 0$. Hence,

$$\begin{aligned} 3\text{oc}(G - X) &\leq n - x + 2y && (\text{Lemma 9.41(iv)}) \\ &\leq (3 - 2b)n + x(2b - 1) && (\text{Lemma 9.41(i)}) \\ &\leq 3x + \left(\frac{4 - 3b}{b} \right) n - (4 - 2b) \left(x - \left(\frac{b - 1}{b} \right) n \right) && (\text{by rearranging}) \\ &\leq 3 \left[x + \left(\frac{4 - 3b}{3b} \right) n \right] && (\text{since } x \geq n(b - 1)/b) \\ &\leq 3 \left[x + \left(\frac{4 - 3b}{3b} \right) n \right] + 4 \left(\frac{b - 1}{b} \right) && (\text{since } b \geq 1). \end{aligned}$$

Hence we may assume that $x < n(b - 1)/b$, for otherwise the required inequality (9.6) follows. Then, $y = 0$, for otherwise [Lemma 9.41\(i\)](#) is contradicted, and so $b > 1$. We therefore have, by [Lemma 9.41\(i\)](#), that

$$\text{oc}(G - X) \leq \frac{1}{3}(n - x). \quad (9.7)$$

If $G - X$ has an odd component of order k , then since $y = 0$, it follows that $n \geq k + x + 3(\text{oc}(G - X) - 1)$, or, equivalently, $\text{oc}(G - X) \leq (n - x - k)/3 + 1$. If $\text{oc}(G - X) \geq 2$, then this implies that

$$\text{oc}(G - X) \leq \frac{x}{3(b - 1)} + 1 \quad (9.8)$$

by [Lemma 9.41\(iii\)](#), which holds even if $\text{oc}(G - X) \leq 1$. Multiplying inequality (9.7) by $(4 - 3b)/b$ and inequality (9.8) by $4(b - 1)/b$ and adding the resultant inequalities together, we obtain the required inequality (9.6). ■

We note that the bounds in [Theorem 9.42\(i\)](#) and (ii) agree if $b = \frac{1}{2}$, while the bounds in [Theorem 9.42\(ii\)](#) and (iii) agree if $b = 1$. The bounds established in [Theorem 9.42](#) are best possible, as may be seen by the following examples provided by Woodall [51].

Example 1: For $s \geq r \geq 1$, consider the complete bipartite graph $G = K_{r,s}$ of order $n = r + s$ with matching number $\alpha'(G) = r$. Let G have binding number $\text{bind}(G) = b$. It follows from [Theorem 9.30](#) that $b = \frac{r}{s} \leq 1$. Hence, $\alpha'(G) = r = bn/(b + 1)$, which shows that the bound in [Lemma 9.41\(i\)](#) is best possible.

Example 2: The bound in [Lemma 9.41\(ii\)](#) is best possible in view of the fact that the bound in [Lemma 9.41\(iii\)](#) is best possible.

Example 3: For $r > s \geq 1$, consider the graph $G = rK_3 + sK_1$ of order $n = 3r + s$ which does not have a perfect matching. Let G have binding number $\text{bind}(G) = b$. By [Theorem 9.30](#),

$$b = \frac{3(r-1)+s}{3(r-1)} = \frac{n-3}{n-s-3}.$$

Since $s \geq 1$, we note that $b > 1$. Furthermore, since $r \geq s+1$ and $n = 3r+s$, we note that $b \leq \frac{4}{3}$, with strict inequality if $r > s+1$. In particular, $1 \leq b \leq \frac{4}{3}$. The matching number of G is therefore

$$\alpha'(G) = r+s = \left(\frac{3b-2}{3b}\right)n - 2\left(\frac{b-1}{b}\right).$$

- ❖ The reader should now be able to attempt Exercises [9.17–9.19](#).

9.7 Matching algorithms for bipartite graphs

In this section, we consider the question of finding a maximum matching in a bipartite graph. For this purpose, we describe two methods. The first method is based on the network flow algorithm in [Chapter 7](#), while the second method involves an algorithm for finding augmenting paths.

9.7.1 Method I: Network flows

The min-max form (a minimum vertex cover with cardinality equal to that of a maximum matching) of the [König-Egerváry Theorem \(Theorem 9.7\)](#) is reminiscent of the [Max-flow Min-cut Theorem](#) due to Ford and Fulkerson ([Theorem 7.5](#)). We show next that flows in networks can, in fact, be used to determine maximum matchings in bipartite graphs.

Let G be a bipartite graph with partite sets X and Y . Let D be the digraph obtained from G by orienting all edges of G from X to Y , adding two new vertices s and t , and adding arcs from s to all vertices of X and arcs from all vertices of Y to t . Let N be the network with D as underlying graph, source s , sink t , and with a capacity function c of N that assigns the value 1 to every arc out of s and to every arc into t , and the value ∞ to all remaining arcs (from X to Y) of D . We call N the **network associated with** the bipartite graph G .

Theorem 9.43 *The cardinality of a maximum matching in a bipartite graph equals the maximum flow in the associated network.*

Proof Let G be a bipartite graph with partite sets X and Y , and let N be the network associated with G .

Let M be a maximum matching in G and let f be the flow in N , defined as follows. For each edge $xy \in M$, let $f(a) = 1$ for each arc a on the directed path $sxyt$. Observe that since M is a matching, all such directed paths have only the vertices s and t in common. Assign to all remaining arcs of N the flow value 0. Then f is a flow in N with $f(N) \geq w(f) = |M| = \alpha'(G)$.

On the other hand, let g be a maximum flow in N . Consider a directed path P , each arc of which has a positive flow under g , i.e. $g(a) = 1$ for each arc on P . Then, P has the form $sxyt$ where $x \in X$ and $y \in Y$. By the flow conservation equation, the flow into x , namely $g(s, x) = 1$, equals the flow out of x , and so $g(a) = 0$ for all arcs $a \neq (x, y)$ incident from x . Since the flow out of y , namely $g(y, t) = 1$, equals the flow incident to y , it follows that $g(a) = 0$ for all arcs $a \neq (y, t)$ into y . Thus the set of edges xy for which $g(x, y) = 1$ determines a matching M in G , and so $\alpha'(G) \geq |M| = w(g) = f(N)$. Consequently, $f(N) = \alpha'(G)$. ■

As an immediate consequence of the [Max-flow Min-cut Theorem](#) ([Theorem 7.5](#)), the [König-Egerváry Theorem](#) ([Theorem 9.7](#)) and [Theorem 9.43](#), we have the following result.

Corollary 9.44 *The vertex covering number of a bipartite graph equals the capacity of a minimum cut in the associated network.*

We can, in fact, use cuts to determine a vertex cover. Suppose, following our earlier notation, that (S, \bar{S}) is a minimum cut of N (with $s \in S$ and $t \in \bar{S}$). Since $f(N)$ is finite, no arc in (S, \bar{S}) has infinite capacity. Therefore, each arc in (S, \bar{S}) has capacity 1 and is of the type (s, x) where $x \in X$ or of the type (y, t) where $y \in Y$. Let $X_S = X \cap S$ and let $X_{\bar{S}} = X \cap \bar{S}$. Similarly, let $Y_S = Y \cap S$ and let $Y_{\bar{S}} = Y \cap \bar{S}$. Then there is no arc from X_S to $Y_{\bar{S}}$, and so $X_{\bar{S}} \cup Y_S$ is a vertex cover in G . Furthermore, since

$$c(S, \bar{S}) = c(s, X_{\bar{S}}) + c(Y_S, t) = |X_{\bar{S}}| + |Y_S| = |X_{\bar{S}} \cup Y_S|,$$

the vertex cover $X_{\bar{S}} \cup Y_S$ is, in fact, a minimum vertex cover by [Corollary 9.44](#).

9.7.2 Method II: Augmenting paths

Our second method for finding maximum matchings in bipartite graphs involves an algorithm for finding augmenting paths. Suppose that G is bipartite with bipartition (X, Y) and let M be a matching in G (perhaps the empty matching). Our aim is to find a maximum matching in G .

The method we use is due to the Hungarian mathematician [König](#), and is often referred to as the *Hungarian method*. The algorithm essentially follows the proof of the [König-Egerváry Theorem](#) presented earlier.

We start with an arbitrary matching M in a bipartite graph G with partite sets X and Y . If every vertex in X is M -matched, then M is a maximum matching in G and we are done. If not, we find the set U of all M -unmatched vertices of X . We then search for an M -augmenting path from a vertex in U to a vertex in Y . If we find such an augmenting path P , we obtain a matching $M \triangle E(P)$ larger than M , and we repeat the above process for the new matching. If, however, there is no such augmenting path, then we find the set S (respectively, T) of all vertices in X (respectively, in Y) that are connected to vertices of U by M -alternating paths. As in the proof of the [König-Egerváry Theorem](#), it follows that $S \setminus U$ is matched to T under M and that $N(S) = T$. Thus, $K = (X \setminus S) \cup T$ is a vertex cover in G with $|K| = |M|$. Hence, by [Corollary 9.6](#), M is a maximum matching in G . This algorithm is formally presented in pseudocode form as [Algorithm 21](#).

- ❖ The reader should now be able to attempt Exercises [9.20–9.22](#).

Algorithm 21: A maximum matching algorithm for bipartite graphs

Input : A bipartite graph $G = (X \cup Y, E)$ and a (possibly empty) matching M .

Output : A maximum matching M^* of G .

```

1 if every vertex in  $X$  is  $M$ -matched then  $M^* \leftarrow M$ , stop
2 Let  $U$  be the set of all  $M$ -unmatched vertices of  $X$ 
3  $S \leftarrow U$ ,  $T \leftarrow \emptyset$ 
4 if  $N(S) = T$  then  $M^* \leftarrow M$ , stop
5 Choose  $y \in N(S) \setminus T$ , choose  $x \in S \cap N(y)$ ,  $\text{parent}(y) \leftarrow \{x\}$ 
6 if  $y$  is  $M$ -unmatched then
7   Identify an  $M$ -augmenting path  $P$  (using the  $\text{parent}$  array) from  $y$  to
     a vertex in  $U$ 
8    $M \leftarrow M \triangle E(P)$ , go to Step 1
9 if  $y$  is  $M$ -matched then
10  Let  $yz \in M$ ,  $S \leftarrow S \cup \{z\}$ ,  $T \leftarrow T \cup \{y\}$ ,  $\text{parent}(z) \leftarrow \{y\}$ , go to Step 4

```

9.8 A matching algorithm for general graphs

In this section, we present an algorithm for finding a maximum matching in a general graph. The algorithm we present is a version of [Edmonds](#)' well-known matching algorithm in which the key idea is to “shrink” certain odd cycles.

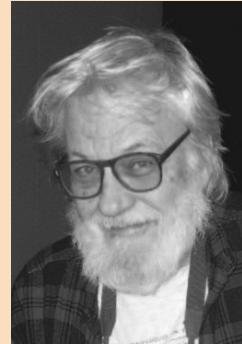
As in the case of matchings in bipartite graphs (see [Section 9.7.2](#)), we shall again search for augmenting paths in our matching algorithm for general graphs. Finding an augmenting path in a graph which is not bipartite is, however, considerably more challenging than finding an augmenting path in a bipartite graph.

We begin our exploration of a matching algorithm for general graphs by defining what [Edmonds](#) coined a flower, a blossom, and a stem. Let M be a matching in a graph G . We call an odd cycle C in G an **odd M -alternating cycle** if $E(C) \cap M$ is an almost perfect matching in the cycle C ; that is, C has length $2k + 1$ for some $k \geq 1$ and C contains exactly k edges of M . An **M -flower** consists of an M -alternating path, called an **M -stem**, of even length from an M -unmatched vertex u to a vertex b (possibly, $u = b$), and an odd M -alternating cycle, called an **M -blossom**, that contains b and is otherwise vertex-disjoint from the M -stem. If the matching M is clear from the context, we call an M -flower, M -stem and M -blossom simply a flower, stem and blossom, respectively. The vertex u is called the **root** of the flower and the vertex b the **base** of the flower. Note that the two edges of the blossom incident with b are not in M . If $u \neq b$, then the edge incident with b on the stem belongs to M .

The graph $G_{9,5}$ in [Figure 9.8](#) is itself an M -flower in which the edges in the matching $M = \{u_1u_2, u_3b, v_1v_2, v_3v_4, v_5v_6\}$ are indicated by solid lines and the remaining edges of $G_{9,5}$ by dashed lines. The stem is the path $u u_1 u_2 u_3 b$ and the blossom is the 7-cycle.

The central idea behind the matching algorithm for general graphs we present here was devised by [Edmonds](#) [11] who proposed an ingenious method, called “blossom shrinking.” Let G be a graph and let M be a matching in G . Let F be an M -flower in G with stem Q and blossom B . Let u be the root of F and let b be the base of F .

Jack R Edmonds, an American computer scientist, was born on 5 April 1934. He obtained a bachelors degree from George Washington University in 1958 and a master's degree from the University of Maryland in 1959 based on a thesis on the problem of embedding graphs on surfaces. During the period 1959–1969 he worked at the then National Bureau of Standards of the United States of America where he was a founding member of the operations research group in 1961. Thereafter, he held a faculty position in the Department of Combinatorics and Optimisation at the University of Waterloo, where he retired in 1999. Edmonds is regarded as a prominent contributor to the field of combinatorial optimisation. One of his earliest and most notable contributions is the blossom algorithm for constructing maximum matchings in graphs, designed in 1961 and published in 1965 (we consider this algorithm in detail in this chapter). His research also focused on matroid theory and polyhedral descriptions for combinatorial optimisation problems on graphs. He is today well known for his work together with [Richard Karp](#) which led to the establishment of the augmenting path algorithm for maximum flows ([Algorithm 18](#)). He was the recipient of the 1985 John von Neumann Theory Prize.



Biographic note 25: Jack Edmonds (1934–present)

Let M_Q be the matching obtained from M by replacing the edges of M that belong to the stem Q by the edges of Q not in M . Thus, $M_Q = M \Delta E(Q)$, i.e. $M_Q = (M \setminus E(Q)) \cup (E(Q) \setminus M)$. Then M_Q is a matching in G and $|M_Q| = |M|$. Note that the blossom B is an odd M_Q -alternating cycle and that the vertex b is an M_Q -unmatched vertex in G . Hence, B is an M_Q -flower with an empty stem. In particular, B is an M_Q -flower.

Let G_B denote the graph obtained from G by replacing the blossom B by a new vertex, called \tilde{B} , and joining this new vertex to all vertices in $V(G) \setminus V(B)$ that are adjacent to a vertex of the blossom B in the graph G . We say that the graph G_B is obtained from G by **contracting** or **shrinking** the blossom B into a single vertex. Let M_B be the restriction of the edges of M_Q to G_B , i.e.

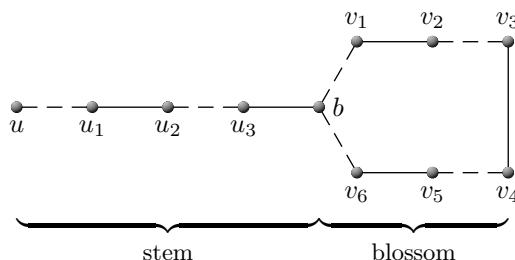


Figure 9.8: An M -flower with matching M indicated by solid edges in the graph $G_{9,5}$.

$M_B = M_Q \setminus E(B)$. Note that the vertex \tilde{B} is an M_B -unmatched vertex in G_B and that $|M| = |M_B| + \frac{1}{2}(|V(B)| - 1)$. We are now in a position to present a result due to [Edmonds](#) [11].

Lemma 9.45 ([Edmonds](#)' Blossom-Shrinking Lemma) *Let G be a graph, let M be a matching in G and let B be an M -blossom in G . Then M_B is a maximum matching in G_B if and only if M is a maximum matching in G .*

Proof We shall employ the notation introduced in the paragraphs immediately preceding the lemma. In particular, b is the base of the blossom B and \tilde{B} is the new vertex in G_B obtained by shrinking the blossom B .

We show first that if M_B is a maximum matching in G_B , then M is a maximum matching in G . Assume, to the contrary, that M is not a maximum matching in G . Since $|M| = |M_Q|$, the matching M_Q is not a maximum matching in G . By [Berge's Theorem](#) ([Theorem 9.1](#)), there exists an M_Q -augmenting path P in G . Let P be an $x-y$ path. Then both x and y are M_Q -unmatched vertices. If P contains no vertex of B , then P is an M_B -augmenting path in G_B , and so M_B is not a maximum matching in G_B . Hence, we may assume that P contains a vertex from the blossom B . Since every vertex of B , except for the vertex b , is M_Q -matched in G , at least one end of P does not belong to B . We may assume that $x \notin V(B)$. Let z be the first vertex on P that belongs to B , and let w be the vertex on P immediately preceding z on P (possibly, $x = w$). Then the $x-w$ subpath of P , followed by the edge from w to \tilde{B} in G_B , is an M_B -augmenting path P in G_B , and so M_B is not a maximum matching in G_B , a contradiction.

We show next that if M is a maximum matching in G , then M_B is a maximum matching in G_B . Assume, to the contrary that M_B is not a maximum matching in G_B . Let M_B^* be a maximum matching in G_B . On the one hand, suppose the vertex \tilde{B} is M_B^* -matched in G_B . Let $c\tilde{B} \in M_B^*$ and let d be a vertex in the blossom B that is adjacent to c in G . Let M' be a perfect matching in the path obtained from B by deleting the vertex d and let $M^* = (M_B^* \setminus \{c\tilde{B}\}) \cup (M' \cup \{cd\})$. On the other hand, suppose the vertex \tilde{B} is M_B^* -unmatched in G_B . In this case, let d be an arbitrary vertex in the blossom B . Let M' be defined as before and let $M^* = M_B^* \cup M'$. In both cases, M^* is a matching in G such that $|M^*| = |M_B^*| + \frac{1}{2}(|V(B)| - 1) > |M_B| + \frac{1}{2}(|V(B)| - 1) = |M|$. Hence, M is not a maximum matching in G , a contradiction. ■

In 1965, [Edmonds](#) [11] presented a matching algorithm for general graphs in a well-cited paper titled *Paths, trees, and flowers*. We describe here a slight modification of [Edmonds](#)' matching algorithm. At the heart of our modified version of the matching algorithm lies the [Edmonds Blossom-Shrinking Lemma](#) ([Lemma 9.45](#)).

Let $G = (V, E)$ be a graph and let M be a (possibly empty) matching in G . If M is a perfect matching, we are done. So assume that the set S of M -unmatched vertices is nonempty. We start by describing a procedure for generating a so-called alternating forest.

Our aim in this subroutine is to construct a forest F , called an **M -alternating forest**, such that each component of F contains one vertex of S , called the **root** of the component, and such that all paths that start at the root are M -alternating paths.

Initially, F is the forest with vertex set S and no edges. As we proceed, we grow the forest by adding vertices and edges to F as follows. We call the vertices at odd distance from S in the forest F the **inner vertices** and the vertices at even distance from S the **outer vertices**. We consider the outer vertices in turn, starting with the M -unmatched vertices in S . Suppose v is an outer vertex currently under consideration. If v is not a root in F , then we list the edge incident with v in F as marked (this edge belongs to the matching M and joins v to an inner vertex) and we list all other edges as unmarked. If v is a root in F , then we list all edges as unmarked. For each edge uv incident with v that has not yet been marked, we have four possibilities:

Case 1: *The vertex u is not yet labelled.*

Since $u \notin S$, the vertex u is M -matched. Let $uw \in M$ and add the vertex w and the edge uw to the forest F . Label u as an inner vertex and w as an outer vertex. In this way, we have grown the forest F we are generating.

Case 2: *The vertex u is an outer vertex, and u and v belong to different components of F .*

In this case, the unique path in $F+uv$ joining the roots of these two components is an M -augmenting path. We augment the matching M along this path to obtain a new, larger matching in place of M and we repeat the entire process with this new matching.

Case 3: *The vertex u is an outer vertex, and u and v belong to the same component of F .*

Adding the edge uv to the unique $u-v$ path in F produces an odd M -alternating cycle B . Let b be the vertex in B whose two incident edges on the cycle are not in M and let P be the unique path in F that joins b to a root of F . (Possibly, b is a root of F in which case P consists of a single vertex.) This produces an M -flower in G with stem P and blossom B . By the Lemma 9.45, we note that M_B is a maximum matching in G_B if and only if M is a maximum matching in G . We now consider the smaller graph G_B and the matching M_B , and we repeat the entire process with this new matching. In this way we reduce the problem of finding a matching larger than M in G to that of finding a matching larger than M_B in the smaller graph G_B (each shrinking of a blossom reduces the number of vertices by at least 2).

Case 4: *The vertex u is an inner vertex.*

In this case, we mark the edge uv as *not* in the forest F and consider the next unmarked edge incident with v .

When we reach the state in the Alternating-Forest subroutine of being unable to grow the forest, we say that the forest has become **Hungarian** and we call each of its components a **Hungarian tree**. Hence the Alternating-Forest subroutine terminates with Case 1 an augmenting path, Case 2 an odd cycle, or Case 3 a Hungarian forest.

Let F be an alternating forest generated by the Alternating-Forest subroutine. Furthermore, let x be a vertex in F and let r be the root in the component of F that contains x (possibly, $r = x$). If x is an inner vertex in F , then x has degree 2 in F and the distance between x and the root r in F is odd. If, however, x is an outer vertex in F , then the distance between x and the root r in F is even. An alternating forest is illustrated in Figure 9.9 where the edges in the matching are indicated by solid lines and the remaining edges by dashed lines.

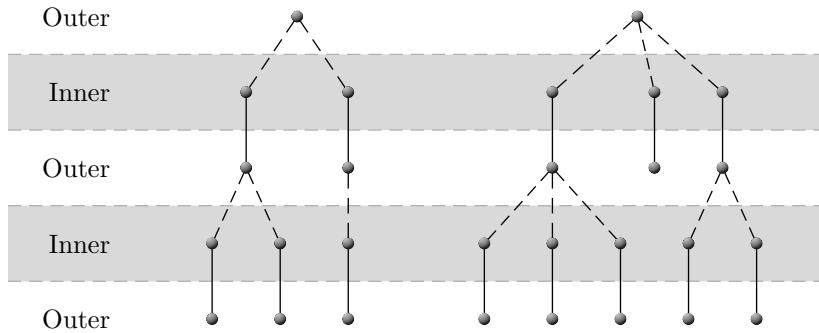


Figure 9.9: An alternating forest with a matching M indicated by solid lines.

This process must eventually terminate in a maximum matching as the following lemma guarantees.

Lemma 9.46 *Let G be a graph, let M be a matching in G and let F be an M -alternating forest produced by the Alternating-Forest subroutine. If every outer vertex in F is adjacent in G only to inner vertices in F , then M is a maximum matching in G .*

Proof Let $G = (V, E)$, let U denote the set of outer vertices in F and let X denote the set of all inner vertices in F . Since every outer vertex in F is adjacent in G only to inner vertices in F , each outer vertex is an (odd) component by itself in $G - X$, and so $\text{oc}(G - X) \geq |U|$. Every vertex in G that is neither an inner vertex nor an outer vertex of F is an unlabelled vertex, and so the number of unlabelled vertices is $|V| - |X| - |U|$. By construction, the forest F contains the set of all M -unmatched vertices in G and each component of F contains exactly one M -unmatched vertex, namely the root of that component. Hence, all unlabelled vertices are M -matched, and so the number of edges of M not incident with a vertex of F is $\frac{1}{2}(|V| - |X| - |U|)$. The number of edges of M in the forest F is precisely the number of inner vertices, namely $|X|$. Hence, $|M| = |X| + \frac{1}{2}(|V| - |X| - |U|) = \frac{1}{2}(|V| + |X| - |U|) \geq \frac{1}{2}(|V| + |X| - \text{oc}(G - X))$. By Lemma 9.17, however, at least $\text{oc}(G - X) - |X|$ vertices are M -unmatched, and so $|M| \leq \frac{1}{2}(|V| - (\text{oc}(G - X) - |X|)) = \frac{1}{2}(|V| + |X| - \text{oc}(G - X))$. Consequently, $|M| = \frac{1}{2}(|V| + |X| - \text{oc}(G - X))$, and so, by Lemma 9.18,

$$\frac{1}{2}(|V| + |X| - \text{oc}(G - X)) = |M| \leq \alpha'(G) \leq \frac{1}{2}(|V| + |X| - \text{oc}(G - X)).$$

We therefore have equality throughout the above inequality chain. In particular, $|M| = \alpha'(G)$. ■

Starting with a (possibly empty) matching M in a graph $G = (V, E)$ of order $|V| = n$ and size $|E| = m$, and applying the Alternating-Forest subroutine, we can therefore always do one of the following:

- Grow the forest F (in Case 1);
- Obtain a larger matching than M (in Case 2);
- Find an M -blossom and decrease $|V|$ (in Case 3); or

- Show that M is a maximum matching (if none of the possibilities in Cases 1, 2 or 3 apply any more for any outer vertex in F).

The Alternating-Forest subroutine will perform at most $\frac{n}{2}$ augmentations of the matching (during repetitions of Case 2), as each augmentation increases the size of the matching by one and, therefore, increases the number of matched vertices by two. Between two augmentations of the matching (during repetitions of Case 3), the Alternating-Forest subroutine will shrink a blossom at most $\frac{n}{2}$ times, as each blossom-shrinking procedure reduces the number of vertices by at least two. Hence, the Alternating-Forest subroutine will be applied at most $\frac{n^2}{4}$ times before terminating with a maximum matching. The time taken by the Alternating-Forest subroutine to construct an alternating forest is $\mathcal{O}(m)$ as each edge is marked at most once. Therefore, repeated applications of the Alternating-Forest subroutine will eventually produce a maximum matching in the resulting graph in $\mathcal{O}(n^2m)$ time.

Once we have applied the Alternating-Forest subroutine to obtain a maximum matching in the reduced graph, we must construct a maximum matching in the original graph. During the repetitions of Case 3 in the Alternating-Forest subroutine, a sequence G_1, \dots, G_k of reduced graphs is generated, resulting from the blossom-shrinking at each stage, and a sequence $\tilde{B}_1, \dots, \tilde{B}_k$ of new vertices is generated by shrinking each blossom into a single vertex. Furthermore, a sequence of matchings M_1, \dots, M_k is generated in G_1, \dots, G_k , respectively.

From the maximum matching M_k in G_k we now construct a matching in the original graph G as follows. Let $M_k^* = M_k$. For $i = k, \dots, 2$, let M_{i-1}^* be the matching in G_{i-1} obtained as follows from the matching M_i^* in G_i . If the vertex \tilde{B}_i in G_i is M_i^* -unmatched, then expand \tilde{B}_i back into the odd cycle B_i to recover the graph G_{i-1} and let M_{i-1}^* consist of the edges of M_i^* together with an almost perfect matching in the odd cycle B_i . If the vertex \tilde{B}_i in G_i is M_i^* -matched, then let $u_i\tilde{B}_i \in M_i^*$. Expand \tilde{B}_i back into the odd cycle B_i to recover the graph G_{i-1} and let v_i denote a vertex in B_i that is adjacent with u_i in G_{i-1} . Let M_{i-1}^* be obtained from $M_i^* \setminus \{u_i\tilde{B}_i\}$ by adding the edge u_iv_i and adding the edges of a perfect matching in the path B_i-v_i . In both cases, $|M_{i-1}^*| = |M_i^*| + \frac{1}{2}(|B_i| - 1)$. Continuing in this way, we construct a matching $M^* = M_1^*$ in the original graph G . Repeated applications of Lemma 9.45 show that M^* is a maximum matching in the graph G .

We present a more formal description of the working of the [maximum matching algorithm](#) in pseudocode form as [Algorithm 22](#). To illustrate the use of [Algorithm 22](#), consider the graph $G_{9.6}$ in [Figure 9.10](#), where the edges of a matching M are indicated by solid lines and the remaining edges by dashed lines.

The set of M -unmatched vertices in $G_{9.6}$ is given by the set $S = \{a, j\}$. Initially, we let F be the forest consisting of the two singleton vertices a and j (and so, $F = 2K_1$). We label $\ell(a) = 1$ and $\ell(j) = 2$. Moreover, we define the sets $\text{outer} = \{a, j\}$, $\text{inner} = \emptyset$, $\text{marked} = \emptyset$, and initialise the list $L = [a, j]$. Since L is not empty, we select and remove the first element of L , namely a .

Let $E_a = [ab, ai]$ be a list consisting of the edges incident with a in $G_{9.6}$. We select and remove the first element of E_a , namely ab . Since $b \notin V(F)$, [Step 12](#) of the algorithm applies. We add the vertices b and c , together with the edges ab and bc , to the forest F . We also add the vertex b to the set inner and the vertex c to the set outer . We then add c to the tail-end of the list L , so that $L = [j, c]$.

Algorithm 22: A maximum matching algorithm for general graphs

Input : A graph $G = (V, E)$ and a (possibly empty) matching M .
Output : A maximum matching M^* in G .

```

1 if every vertex in  $V$  is  $M$ -matched then
2    $M^* \leftarrow M$ , print  $M^*$ , stop
3  $S \leftarrow \{r_1, \dots, r_k\}$  (the set of all  $M$ -unmatched vertices),  $F \leftarrow (S, \emptyset) = kK_1$ 
4 Label each vertex  $r_i$  with the label  $\ell(r_i) = i$  (to denote that  $r_i$  belongs to
   the  $i$ -th component of  $F$ )
5  $\text{outer} \leftarrow S$ ,  $\text{inner} \leftarrow \emptyset$ ,  $\text{marked} \leftarrow \emptyset$ ,  $L \leftarrow S$ 
6 if  $L$  is empty then stop
7 else  $v \leftarrow$  first element in  $L$ ,  $L \leftarrow L \setminus \{v\}$ 
8  $E_v \leftarrow \{\text{a list consisting of the edges incident with } v \text{ in } G\}$ 
9 if  $E_v = \emptyset$  then  $L \leftarrow L \setminus \{v\}$ , go to Step 6
10 else  $uv \leftarrow$  first element in  $E_v$ ,  $E_v \leftarrow E_v \setminus \{uv\}$ 
11 if  $uv \in E(F)$  then  $\text{marked} \leftarrow \text{marked} \cup \{uv\}$ , go to Step 9
12 if  $u \notin V(F)$  then
13    $uw \in M$  for some vertex  $w \notin V(F)$ 
14    $V(F) \leftarrow V(F) \cup \{u, w\}$ ,  $E(F) \leftarrow E(F) \cup \{uv, uw\}$ 
15    $\text{inner} \leftarrow \text{inner} \cup \{u\}$ ,  $\text{outer} \leftarrow \text{outer} \cup \{w\}$ 
16   Add  $w$  to the tail-end of the list  $L$ ,  $\ell(u) = \ell(w) = \ell(v)$ 
17    $E_v \leftarrow E_v \setminus \{uv\}$ ,  $\text{marked} \leftarrow \text{marked} \cup \{uv\}$ , go to Step 9
18 if  $u \in \text{outer}$  and  $\ell(u) \neq \ell(v)$  then
19   Augment the matching  $M$  along an  $M$ -augmenting path from  $r_{\ell(u)}$ 
     to  $r_{\ell(v)}$  in  $F$ , go to Step 1
20 if  $u \in \text{outer}$  and  $\ell(u) = \ell(v)$  then
21   Let  $P$  be the unique path in  $F$  that joins  $r_{\ell(v)}$  to the vertex of  $B$  that
     is  $M$ -unmatched in the cycle  $B$ 
22   Let  $M_B$  be obtained from the matching  $M$  by augmenting  $M$  along
     the path  $P$  and then removing the edges of  $M$  that belong to the
      $M$ -blossom  $B$ 
23   Let  $G_B$  be the graph obtained from  $G$  by shrinking the  $M$ -blossom  $B$ 
     into a single vertex
24    $G \leftarrow G_B$ ,  $M \leftarrow M_B$ , go to Step 1
25 if  $u \in \text{inner}$  then
26    $\text{marked} \leftarrow \text{marked} \cup \{uv\}$ , go to Step 9

```

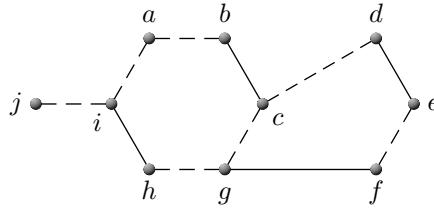


Figure 9.10: A graph $G_{9.6}$ with a given matching M in $G_{9.6}$ indicated by solid edges.

We choose the labels $\ell(b) = \ell(c) = \ell(a) = 1$ (to indicate that b and c are in the same component of F as the root vertex a), add the edge ab to the set **marked**, and return to **Step 9**. We next select and remove the first (and only) element of $E_a = [ai]$, namely ai . Since $i \notin V(F)$, we add the vertices i and h , together with the edges ai and ih , to the forest F , add the vertex i to the set **inner** and add the vertex h to the set **outer**. We also add h to the tail-end of the list L , and so $L = [j, c, h]$. We then choose the labels $\ell(i) = \ell(h) = \ell(a) = 1$, add the edge ai to the set **marked**, and return to **Step 9**. Since the list E_a is now empty, we return to **Step 6**.

Since $L = [j, c, h]$ is not empty, we select and remove the first element of L , namely j . Let $E_j = [ji]$ be a list consisting of the edges incident with j in G . We select and remove the first element of E_j , namely ji . Since $i \in \text{inner}$, we add the edge ji to the set **marked** in **Step 12** of **Algorithm 22**, and return to **Step 9**. Since the list E_j is now empty, we return to **Step 6**.

Because $L = [c, h]$ is not empty, we select and remove the first element of L , namely c . Let $E_c = [cb, cd, cg]$ be a list consisting of the edges incident with b in $G_{9.6}$. We select and remove the first element of E_c , namely bc . Since $bc \in E(F)$, we add the edge bc to the set **marked**, and return to **Step 9**. We select and remove the first element of $E_c = [cd, cg]$, namely cd . Since $d \notin V(F)$, we add the vertices d and e , together with the edges cd and de , to the forest F . We also add the vertex d to the set **inner** and the vertex e to the set **outer**. We then add e to the tail-end of the list L , and so $L = [h, e]$. We next assign the labels $\ell(d) = \ell(e) = \ell(c) = 1$, add the edge cd to the set **marked**, and return to **Step 9**. Now we select and remove the first element of $E_c = [cg]$, namely cg . Since $g \notin V(F)$, we add the vertices g and f , together with the edges hg and gf , to the forest F . We also add the vertex g to the set **inner** and the vertex f to the set **outer**. We then add f to the tail-end of the list L , so that $L = [h, e, f]$. Let $\ell(g) = \ell(f) = \ell(h) = 1$. We next add the edge cg to the set **marked**, and return to **Step 9**. Since the list E_c is now empty, we return to **Step 6**. The forest F is shown in **Figure 9.11**, where the edges of a matching M are indicated by solid lines and the remaining edges by dashed lines.

Since $L = [h, e, f]$ is not empty, we select and remove the first element of L , namely h . Let $E_h = [hi, hg]$ be a list consisting of the edges incident with h in $G_{9.6}$. We select and remove the first element of E_h , namely hi . Since $hi \in E(F)$, we add the edge hi to the set **marked**, and return to **Step 9**. We select and remove the first (and only) element of $E_h = [hg]$, namely hg . Since $g \in \text{inner}$, we add the edge hg to the set **marked**, and return to **Step 9**. The list E_h is now empty, and we return to **Step 6**.

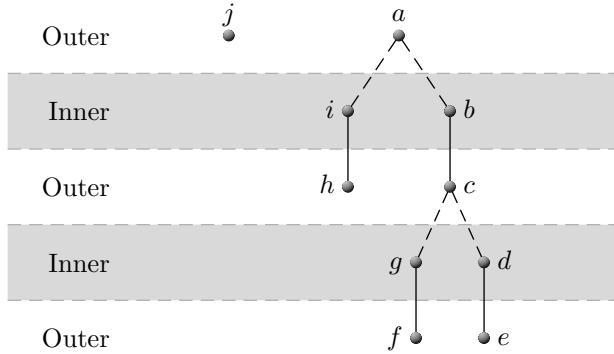


Figure 9.11: The forest F with the edges of a matching M indicated by solid lines.

Because $L = [e, f]$ is not empty, we select and remove the first element of L , namely e . Let $E_e = [ed, ef]$ be a list consisting of the edges incident with e in $G_{9.6}$. We select and remove the first element of E_e , namely ed . Since $ed \in E(F)$, we add the edge ed to the set **marked**, and return to **Step 9**. Therefore, we select and remove the first element of $E_e = [ef]$, namely ef . Since $f \in \text{outer}$ and $\ell(e) = \ell(f) = 1$, **Step 20** of [Algorithm 22](#) applies. This produces an M -flower in $G_{9.6}$ with stem $P : abc$ and blossom $B : cdefgc$, as shown in [Figure 9.12](#).

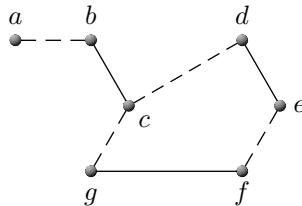


Figure 9.12: The M -flower in $G_{9.6}$ with stem $P : abc$ and blossom $B : cdefgc$.

Let M_B be obtained from the matching M by augmenting M along the path P and then removing the edges of M that belong to the M -blossom B . Thus, $M_B = (M \setminus \{bc, de, fg\}) \cup \{ab\}$. Let G_B be the graph obtained from $G_{9.6}$ by shrinking the M -blossom B into a single vertex \tilde{B} . The graph G_B is shown in [Figure 9.13](#), where the edges of the matching M_B are indicated by solid lines and the remaining edges by dashed lines.

Let $G_{9.7} = G_B$ and $M_{9.7} = M_B$, and return to **Step 1**. The set of $M_{9.7}$ -unmatched vertices in $G_{9.7}$ is $S = \{\tilde{B}, j\}$. Initially, we let $F_{9.7}$ be the forest consisting of the two singleton vertices \tilde{B} and j (and so, $F_{9.7} = 2K_1$ with \tilde{B} and j the roots of the two components of $F_{9.7}$). We label $\ell(\tilde{B}) = 1$ and $\ell(j) = 2$. Moreover, we define the sets **outer** = $\{\tilde{B}, j\}$, **inner** = \emptyset , and **marked** = \emptyset , and initialise the list $L = [\tilde{B}, j]$. Since L is not empty, we select and remove the first element of L , namely \tilde{B} .

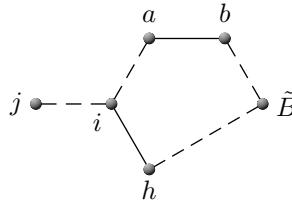


Figure 9.13: A graph G_B with a matching M_B in G_B indicated by solid lines.

Let $E_{\tilde{B}} = [\tilde{B}b, \tilde{B}h]$ be a list consisting of the edges incident with \tilde{B} in $G_{9.7}$. We select and remove the first element of $E_{\tilde{B}}$, namely $\tilde{B}b$. Since $b \notin V(F_{9.7})$, we add the vertices b and a , together with the edges $\tilde{B}b$ and ab , to the forest $F_{9.7}$. We also add the vertex b to the set **inner** and the vertex a to the set **outer**. We then add a to the tail-end of the list L , and so $L = [j, a]$. Let $\ell(b) = \ell(c) = \ell(\tilde{B}) = 1$. We add the edge ab to the set **marked**, and return to **Step 9**. We now select and remove the first (and only) element of $E_{\tilde{B}} = [\tilde{B}h]$, namely $\tilde{B}h$. Since $h \notin V(F_{9.7})$, we add the vertices h and i , together with the edges $\tilde{B}h$ and hi , to the forest $F_{9.7}$. We also add the vertex h to the set **inner** and the vertex i to the set **outer**. We next add i to the tail-end of the list L , so that $L = [j, a, i]$. Let $\ell(h) = \ell(i) = \ell(\tilde{B}) = 1$. We then add the edge $\tilde{B}h$ to the set **marked**, and return to **Step 9**. Since the list $E_{\tilde{B}}$ is now empty, we return to **Step 6**. The forest $F_{9.7}$ is shown in **Figure 9.14**, where the edges of the matching $M_{9.7}$ are indicated by solid lines and the remaining edges by dashed lines.

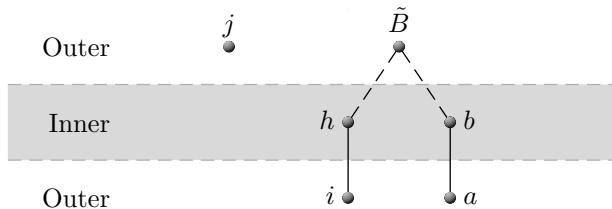


Figure 9.14: The forest $F_{9.7}$ with the edges of a matching $M_{9.7}$ in $F_{9.7}$ indicated by solid edges.

Since $L = [j, a, i]$ is not empty, we select and remove the first element of L , namely j . Let $E_j = [ji]$ be a list consisting of the edges incident with j in $G_{9.6}$. We select and remove the first (and only) element of E_j , namely ji . Since $i \in \text{outer}$ and $\ell(j) \neq \ell(i)$, there is an $M_{9.7}$ -augmenting path from \tilde{B} to j in $F_{9.7}$, namely $P_{9.7} : \tilde{B}hij$. We augment the matching $M_{9.7}$ along this path to obtain a new, larger matching $M'_{9.7} = \{\tilde{B}h, ij, ab\}$ and return to **Step 1** with the graph $G_{9.7}$ and matching $M'_{9.7}$. The graph $G_{9.7}$ is shown in **Figure 9.15**, where the edges of the matching $M'_{9.7}$ are indicated by solid lines and the remaining edges by dashed lines.

The matching $M'_{9.7}$ is a perfect matching in $G_{9.7}$. Hence we proceed to **Step 2** of **Algorithm 22**. Let $M_{9.7}^* = M'_{9.7}$. Since the vertex \tilde{B} in $G_{9.7}$ is $M_{9.7}^*$ -matched (the edge $\tilde{B}h \in M_{9.7}^*$), we expand \tilde{B} back into the odd cycle B to recover the graph $G_{9.6}$ and we note that g is a vertex in B that is adjacent with h in $G_{9.6}$. We

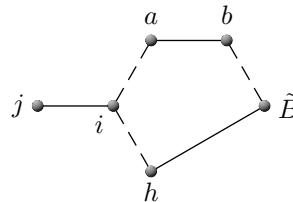


Figure 9.15: A graph $G_{9.7}$ with a matching $M'_{9.7}$ in $G_{9.7}$ indicated by solid edges.

let M^* be obtained from $M^*_{9.7} \setminus \{\tilde{B}h\}$ by adding the edge gh and adding the edges of a perfect matching in the B - g path, namely $\{cd, ef\}$. Thus, $M^* = \{ab, cd, ef, gh, ij\}$ and M^* is a maximum matching in the original graph $G_{9.6}$ (in fact, M^* is a perfect matching in $G_{9.6}$). The graph $G_{9.6}$ and the maximum matching M^* are shown in Figure 9.16, where the edges of M^* are indicated by solid lines and the remaining edges by dashed lines.

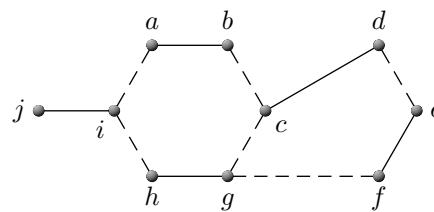


Figure 9.16: A graph $G_{9.6}$ with a maximum matching M^* indicated by solid edges.

- ❖ The reader should now be able to attempt Exercise 9.23.

9.9 A weighted matching algorithm

In this section, we present a polynomial-time algorithm for finding a matching of maximum weight in a weighted graph. Given a weighted graph $G = (V, E)$ of order n with a weight w_e assigned to each edge $e \in E$, we wish to find a matching, or a perfect matching, for which the sum of the edge weights is maximised. As an extension of a maximum matching in an unweighted graph, Edmonds and Johnson proved that a maximum-weight matching in G can be found in $\mathcal{O}(n^3)$ time by exploiting a connection with linear programming. The Edmonds-Johnson algorithm is an example of a so-called “primal-dual” algorithm which relies on complementary slackness conditions in the theory of linear programming. Primal-dual algorithms are regarded as amongst the most important and elegant algorithms in combinatorial optimisation.

9.9.1 Linear programming background

Since there is a deep-rooted connection between the weighted matching algorithm and the theory of linear programming, we first explore some foundational concepts of the latter.

Consider the *primal* linear programming problem:

$$\left. \begin{array}{ll} \text{Maximise} & \sum_{j=1}^q c_j x_j \\ \text{subject to} & \sum_{j=1}^q a_{ij} x_j \leq b_i \text{ for all } i \in [p], \\ & x_j \geq 0 \text{ for all } j \in [q]. \end{array} \right\} \quad (9.9)$$

Recall, from elementary linear programming theory, that the *dual* linear programming problem associated with the primal linear programming problem (9.9) is:

$$\left. \begin{array}{ll} \text{Minimise} & \sum_{i=1}^p b_i y_i \\ \text{subject to} & \sum_{i=1}^p a_{ij} y_i \geq c_j \text{ for all } j \in [q], \\ & y_i \geq 0 \text{ for all } i \in [p]. \end{array} \right\} \quad (9.10)$$

The following well-known result relates a primal linear programming problem with its dual (see, for example, [8, Theorem 2.12]).

Theorem 9.47 (Complementary Slackness Theorem) *If $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_q^*)$ is a feasible solution to the primal problem (9.9) and $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_p^*)$ is a feasible solution to the dual problem (9.10), then \mathbf{x}^* and \mathbf{y}^* are optimal solutions to their respective problems if they satisfy*

$$y_i^* \left(b_i - \sum_{j=1}^q a_{ij} x_j^* \right) = 0 \text{ for each } i \in [p] \text{ and} \quad (9.11)$$

$$x_j^* \left(\sum_{i=1}^p a_{ij} y_i^* - c_j \right) = 0 \text{ for each } j \in [q]. \quad (9.12)$$

The **Complementary Slackness Theorem** therefore states that either the i -th dual variable $y_i^* = 0$ or the slack in the i -th constraint of the primal problem is zero. Furthermore, either the j -th primal variable $x_j^* = 0$ or the excess in the j -th constraint of the dual problem is zero.

9.9.2 The maximum weight matching problem

We assume that all edge weights of the graph $G = (V, E)$ for which we seek a matching of maximum weight are nonnegative. Moreover, we may assume that the graph G is a complete graph, for otherwise we may simply add all missing edges and assign each added edge a weight of zero. In order to find a matching in G for which the sum of the edge weights is maximised, we associate a binary variable x_e with each edge $e \in E$ with the interpretation that $x_e = 1$ if the edge e is included in the matching, or $x_e = 0$ otherwise. For each vertex $v \in V$, we let $\partial(v)$ denote the set of all edges incident with v in G . Since the vertex v is incident with at most one edge of the matching, we note that $\sum_{e \in \partial(v)} x_e \leq 1$ for every $v \in V$.

We note that the maximum-weight matching problem may be formulated as an *integer* programming problem in which the objective is to

$$\left. \begin{array}{ll} \text{maximise} & \sum_{e \in E} w_e x_e \\ \text{subject to} & \sum_{e \in \partial(v)} x_e \leq 1 \quad \text{for all } v \in V, \\ & x_e \in \{0, 1\} \text{ for all } e \in E. \end{array} \right\} \quad (9.13)$$

Let $O = \{S \subseteq V \mid |S| \geq 3 \text{ and } |S| \text{ is odd}\}$ (here O stands for “odd” subsets of V). For each nonempty subset $S \subseteq V$, we let E_S denote the set of all edges in G with both ends in S ; that is, $E_S = E(G[S])$ is the edge set of the subgraph of G induced by the set S . We note that if $S \in O$, then at most $(|S| - 1)/2$ edges in E_S belong to the matching. Hence we can add to the constraints in (9.13) the additional constraint

$$\sum_{e \in E_S} x_e \leq \frac{1}{2}(|S| - 1)$$

for every $S \in O$. We now consider the following linear programming relaxation of the above integer programming problem, in which the objective is to

$$\left. \begin{array}{ll} \text{maximise} & \sum_{uv \in E} w_{uv} x_{uv} \\ \text{subject to} & \sum_{uv \in E} x_{uv} \leq 1 \quad \text{for all } v \in V, \\ & \sum_{uv \in E_S} x_{uv} \leq \frac{1}{2}(|S| - 1) \quad \text{for all } S \in O, \\ & x_{uv} \geq 0 \quad \text{for all } uv \in E. \end{array} \right\} \quad (9.14)$$

We shall later prove that there exists an optimal solution to the linear programming problem (9.14) that is integral. As remarked earlier, if x_{uv} is the binary variable associated with a matching M and edges $uv \in E$, where

$$x_{uv} = \begin{cases} 1 & \text{if } uv \in M \\ 0 & \text{if } uv \notin M, \end{cases}$$

then the variables x_{uv} satisfy the constraints in (9.14).

In order to formulate the dual of the linear programming problem (9.14), we introduce a variable y_v for each primal constraint corresponding to the vertices $v \in V$ and we introduce a variable z_S for each primal constraint corresponding to the subsets $S \in O$. For each edge $uv \in E$, let O_{uv} consist of all sets $S \in O$ that contain both u and v . Thus, we can write the dual of the linear programming problem (9.14) as requiring to

$$\left. \begin{array}{ll} \text{minimise} & \sum_{v \in V} y_v + \sum_{S \in O} \frac{1}{2}(|S| - 1) z_S \\ \text{subject to} & y_v \geq 0 \quad \text{for all } v \in V, \\ & z_S \geq 0 \quad \text{for all } S \in O, \\ & y_u + y_v + \sum_{S \in O_{uv}} z_S \geq w_{uv} \quad \text{for all } uv \in E, \\ & x_{uv} \geq 0 \quad \text{for all } uv \in E. \end{array} \right\} \quad (9.15)$$

For each edge $uv \in E$, we define the **reduced cost** of the edge uv , abbreviated $\pi(uv)$, by

$$\pi(uv) = y_u + y_v - w_{uv} + \sum_{S \in O_{uv}} z_S.$$

By the dual constraints, we note that $\pi(uv) \geq 0$ for all $uv \in E$. An edge $uv \in E$ is called a **tight edge** if $\pi(uv) = 0$.

The complementary slackness conditions in (9.11)–(9.12) for the above pair of primal-dual linear programming problems (9.14) and (9.15) are

$$x_{uv} > 0 \Rightarrow \pi(uv) = 0 \quad \text{for all } uv \in E, \quad (\text{CS1})$$

$$y_v > 0 \Rightarrow \sum_{uv \in E} x_{uv} = 1 \quad \text{for all } v \in V, \quad (\text{CS2})$$

$$z_S > 0 \Rightarrow \sum_{uv \in E_S} x_{uv} = \frac{1}{2}(|S| - 1) \quad \text{for all } S \in O. \quad (\text{CS3})$$

Our aim is to associate the binary variable x_{uv} , where $uv \in E$, with a matching M and construct feasible values for the dual variables y_v , where $v \in V$, and z_S , where $S \in O$, which satisfy the complementary slackness conditions (CS1), (CS2) and (CS3).

For our initial solution, we start with the empty matching $M = \emptyset$, and so $x_e = 0$ for all $uv \in E$. For all $v \in V$, we initially assign to each dual variable y_v a value equal to half of the maximum edge weight incident with the vertex v , and we let $z_S = 0$ for all $S \in O$. Thus, our initial solution is

$$\begin{cases} x_e = 0 & \text{for all } uv \in E, \\ y_v = \frac{1}{2} \max\{w_e \mid e \in \partial(v)\} & \text{for all } v \in V, \\ z_S = 0 & \text{for all } S \in O. \end{cases}$$

For this initial solution, the three primal constraints

$$\sum_{uv \in \partial(v)} x_{uv} \leq 1 \quad \text{for all } v \in V, \quad (\text{P1})$$

$$\sum_{uv \in E_S} x_{uv} \leq \frac{1}{2}(|S| - 1) \quad \text{for all } S \in O, \quad (\text{P2})$$

$$x_{uv} \geq 0 \quad \text{for all } uv \in E \quad (\text{P3})$$

are satisfied.

Hence, the variables x_{uv} are all feasible solutions to the primal problem. For each edge $uv \in E$, we note that $y_u + y_v \geq w_{uv}$. Moreover, since $z_S = 0$ for each subset $S \in O$, the three dual constraints

$$y_v \geq 0 \quad \text{for all } v \in V, \quad (\text{D1})$$

$$z_S \geq 0 \quad \text{for all } S \in O, \quad (\text{D2})$$

$$\pi(uv) \geq 0 \quad \text{for all } uv \in E \quad (\text{D3})$$

are satisfied. Therefore, the variables y_v and z_S are all feasible solutions of the dual problem.

9.9.3 Outline of the maximum weight matching algorithm

Note that the initial primal variables $x_{uv} = 0$ all vacuously satisfy the complementary slackness condition (CS1), while the initial dual variables $z_S = 0$ all vacuously satisfy the complementary slackness condition (CS3). Only the complementary slackness condition (CS2) remains unsatisfied. Let

$$f(G) = \sum_{v \in V} y_v + \sum_{S \in O} \frac{1}{2}(|S| - 1)z_S$$

denote the objective function of the dual problem. At each iteration of our algorithm, the aim is to change the values of the primal and dual variables in such a way that the following requirements are satisfied:

- (R1) The objective function $f(G)$ strictly decreases after each dual adjustment,
- (R2) the primal solution remains feasible,
- (R3) the dual solution remains feasible,
- (R4) conditions (CS1) and (CS3) remain satisfied, and
- (R5) condition (CS2) is satisfied by at least one more dual variable y_v .

Since there are $|V| = n$ dual variables y_v , $v \in V$, it follows that condition (CS2) is satisfied by all dual variables y_v after at most n iterations of the algorithm. By complementary slackness, the resulting matching must be a maximum weight matching of G .

In order to ensure that condition (CS1) remains satisfied, we only use tight edges. Initially we let E_0^* be the set of tight edges in G . Possibly, $E_0^* = \emptyset$. After each iteration of the algorithm, however, the set of tight edges grows in size. We let E_i^* denote the set of tight edges during the i -th iteration of the algorithm.

Condition (CS2) implies that if, for some vertex $v \in V$, the dual variable $y_v > 0$, then the vertex v is matched. Thus, if condition (CS2) is violated for some vertex $v \in V$, then $y_v > 0$ and the vertex v is unmatched.

At each stage of the algorithm, an unmatched vertex v with $y_v > 0$ is selected. As remarked earlier, the vertex v violates condition (CS2). We then use our Alternating-Forest subroutine described in Section 9.8 to produce an augmenting path, an odd cycle, or a Hungarian tree:

Augmenting path. If the subroutine finds an augmenting path, then we augment the matching along this path. This increases the weight of the matching and matches the vertex v which now satisfies condition (CS2).

Odd cycle. If the subroutine finds an odd cycle (a blossom), then the cycle is shrunk into a new vertex, called a **pseudovertex**, and the matching is restricted to the edges in the resulting graph.

Hungarian tree. If the subroutine finds a Hungarian tree, then we adjust the dual variables so that the requirements (R1)–(R5) are all satisfied.

In all three cases, either y_v is reduced to zero or the vertex v is matched, so that the vertex v satisfies condition (CS2). Let **outer** denote the set of vertices of G which are outer vertices, or are contained in outer pseudovertices, of an alternating tree, and let **inner** denote the set of vertices of G which are inner vertices, or are contained in inner pseudovertices, of an alternating tree. Furthermore, we let **unlabelset** = $V(G) \setminus (O \cup I)$, and so U denotes the set of vertices of G that are unlabelled (and therefore do not belong to the alternating tree).

Algorithm 23: Maximum-weight matching algorithm

Input : A weighted graph G and the associated primal-dual linear programming problems (9.14) and (9.15).

Output : A maximum-weight matching M in G .

- 1 $x_e \leftarrow 0$ for all $uv \in E$, $y_v \leftarrow \frac{1}{2} \max\{w_e \mid e \in \partial(v)\}$ for all $v \in V$, $z_S \leftarrow 0$ for all $S \in \mathcal{O}$
- 2 $F_1 \leftarrow G$, $E_1^* \leftarrow \{\text{all tight edges in } F_1\}$,
 $F_1^* \leftarrow \text{spanning subgraph of } F_1 \text{ with } E(F_1^*) = E_1^*$
- 3 $M_1 \leftarrow \text{MaximumMatching}(F_1^*)$, $x_e \leftarrow 1 [0]$ if $e \in M_1$ [otherwise], $k \leftarrow 1$
- 4 Let $F \leftarrow F_k$, $M \leftarrow M_k$, $E^* \leftarrow E_k^*$, $x_e \leftarrow 1 [0]$ if $e \in M$ [otherwise]
- 5 Select an M -unmatched vertex $v \in V(F)$ such that either v is not a pseudovertex in F and $y_v > 0$, or v is a pseudovertex in F and every vertex w in the odd cycle associated with the pseudovertex v has $y_w > 0$
- 6 **if** no such vertex v exists **then go to Step 13**
- 7 Let F^* be the spanning subgraph of F with $E(F^*) = E^*$
- 8 $\text{MaximumMatching}(F^*)$, Use the Alternating-Forest subroutine of Section 9.8 to grow an alternating tree rooted at v
- 9 **if** Alternating-Forest subroutine finds an M -augmenting path **then call Algorithm 23a**
- 10 **else if** Alternating-Forest subroutine finds an odd cycle **then call Algorithm 23b**
- 11 **else if** Alternating-Forest subroutine finds a Hungarian tree containing v **then call Algorithm 23c**
- 12 Expand all pseudovertices, if any, in the reverse order in which they were constructed
- 13 Modify the matching at each stage so that all vertices in the associated odd cycles are matched (follow Step 14 in Algorithm 23c)
- 14 Let F_{k+1} denote the resulting graph and let M_{k+1} denote the resulting matching, $k \leftarrow k + 1$, $M \leftarrow M_k$
- 15 Update the primal variables $x_e = 1 [0]$ if $e \in M$ [otherwise], **stop**

A pseudocode description of the maximum weight matching algorithm is presented as Algorithm 23. We show next that Algorithm 23 terminates with a matching of maximum weight in G .

Theorem 9.48 *Algorithm 23 terminates with a matching of maximum weight in the weighted input graph G .*

Proof We show that the terminal primal variables x_{uv}^* as well as the dual variables y_v^* and z_S^* produced by Algorithm 23 are optimal solutions to their respective problems. By the Complementary Slackness Theorem (Theorem 9.47), it suffices to show that the following holds: First, that the three primal constraints (P1), (P2) and (P3) are satisfied. Secondly, that the three dual constraints (D1), (D2) and (D3) are satisfied. Thirdly, that the complementary slackness conditions (CS1), (CS2) and (CS3) are satisfied.

For our initial solution, both the primal and dual constraints are satisfied as are both complementary slackness conditions (CS1) and (CS3). Only the complementary slackness condition (CS2) remains unsatisfied. We show that during

Algorithm 23a: Auxiliary function used in Step 9 of the Maximum-weight matching algorithm (Algorithm 23)

- 1 Augment the matching along the M -augmenting path, Let M_{k+1} denote the resulting matching and let $E_{k+1}^* = E^*$
 - 2 $F_{k+1} \leftarrow F_k$, $k \leftarrow k + 1$
 - 3 go to Step 4 of Algorithm 23
-

Algorithm 23b: Auxiliary function used in Step 10 of the Maximum-weight matching algorithm (Algorithm 23)

- 1 Shrink the odd cycle B_{k+1} into a pseudovertex b_{k+1} , Let F_{k+1} denote the resulting graph
 - 2 Let M_{k+1} consist of all edges of M with no end in B_{k+1}
 - 3 if $uv \in M$ is such an edge where u is outside B_{k+1} and v is inside B_{k+1} then $M_{k+1} \leftarrow M_{k+1} \cup \{ub_{k+1}\}$
 - 4 if pseudovertex b_{k+1} is an outer [inner] vertex in a future alternating tree then
 - 5 Assign all original vertices inside odd cycle B_{k+1} as outer [inner] vertices in such an alternating tree
 - 6 $E_{k+1}^* \leftarrow$ all edges in E^* with no end in B_{k+1}
 - 7 $E_{k+1}^* \leftarrow E_{k+1}^* \cup \{ub_{k+1} \in F_{k+1} \mid uw \in E^* \text{ for some } w \in V(B_{k+1}) \text{ in } F_k\}$
 - 8 $k \leftarrow k + 1$, go to Step 4 of Algorithm 23
-

each iteration, Algorithm 23 changes the values of the primal and dual variables in such a way that the requirements (R1)–(R5) are all satisfied. Let T denote the Hungarian tree during the current iteration.

Claim (R1) Requirement (R1) is satisfied at all times by the algorithm.

Proof We show that the objective function

$$f(G) = \sum_{v \in V} y_v + \sum_{S \in O} \frac{1}{2}(|S| - 1)z_S$$

of the dual problem strictly decreases after each dual adjustment of the algorithm. Let $f'(G)$ be the new value of the objective function after an iteration of the algorithm, and let $\Delta f = f'(G) - f(G)$ denote the change in the dual objective function value after the dual adjustment. For each vertex $v \in V$, let $\Delta f_v = y'_v - y_v$, while for each $S \in O$, let

$$f(S) = \sum_{v \in S} y_v + \frac{1}{2}(|S| - 1)z_S$$

and let $\Delta f_S = f'(S) - f(S)$. The dual interchange implies that

$$\Delta f_v = \begin{cases} -\delta & \text{if } v \in \text{outer}, \\ \delta & \text{if } v \in \text{inner}, \\ 0 & \text{if } v \in \text{unlabelset}. \end{cases}$$

Algorithm 23c: Auxiliary function used in Step 11 of the Maximum-weight matching algorithm (Algorithm 23)

- 1 Let T denote the Hungarian tree
 - 2 Modify the dual solution as follows: $\delta \leftarrow \min\{\delta_1, \delta_2, \delta_3, \delta_4\}$ where

$$\delta_1 = \min\{y_w \mid w \in \text{outer}\}$$

$$\delta_2 = \min\{\pi(uw) \mid u \in \text{outer} \text{ and } w \in \text{unlabelset}\}$$

$$\delta_3 = \frac{1}{2} \min\{\pi(uw) \mid u, w \in \text{outer} \text{ and } u, w \text{ do not belong to the same pseudovertex of } T\}$$

$$\delta_4 = \frac{1}{2} \min\{z_S \mid S \text{ is the set of vertices belonging to an inner pseudovertex of } T\}$$
 - 3 **if** any of these sets is empty **then** let corresponding $\delta_i = \infty$
 - 4 **if** $\delta = \infty$ **then** G has no perfect matching, **stop**
 - 5 Let

$$y'_w \leftarrow \begin{cases} y_w - \delta & \text{for } w \in \text{outer} \\ y_w + \delta & \text{for } w \in \text{inner} \\ y_w & \text{for } w \in \text{unlabelset} \end{cases}$$
 - and let

$$z'_S \leftarrow \begin{cases} z_S + 2\delta & \text{if } S \text{ is the set of vertices contained in an outer pseudovertex of } T \\ z_S - 2\delta & \text{if } S \text{ is the set of vertices contained in an inner pseudovertex of } T \\ z_S & \text{otherwise} \end{cases}$$
 - 6 **if** $\delta = \delta_1$ **then**
 - 7 The dual variable $y'_w = 0$ for some outer vertex w
 - 8 Augment the matching along the M -alternating path in T from v to w
 - 9 $K_{k+1} \leftarrow F_k$, Let M_{k+1} denote the resulting matching, $E_{k+1}^* \leftarrow E^*$
 - 10 $k \leftarrow k + 1$, **go to** Step 4 of Algorithm 23
 - 11 **if** $\delta = \delta_2$ **or** δ_3 **then**
 - 12 $K_{k+1} \leftarrow F_k$, $M_{k+1} \leftarrow M$, Let E_{k+1}^* denote the new set of tight edges after the dual adjustment
 - 13 $k \leftarrow k + 1$, **go to** Step 4 of Algorithm 23
 - 14 **if** $\delta = \delta_4$ **then**
 - 15 $z'_S \leftarrow 0$ where z_S is the dual variable that determines δ_4 and S is the associated set of vertices belonging to an inner pseudovertex b^* of T
 - 16 Let b^*w be the edge of M incident with the pseudovertex b^* corresponding to the dual variable z_S
 - 17 Expand the pseudovertex b^* back to its original cycle B^*
 - 18 Let z be the vertex of B^* incident with w that gave rise to the matched edge b^*w
 - 19 Let F_{k+1} denote the resulting expanded graph; Let M_{k+1} consist of all edges of $(M_k \setminus \{b^*w\}) \cup \{zw\}$ together with the edges of B^* that match all vertices of $B^* \setminus \{z\}$
 - 20 $E_{k+1}^* \leftarrow E^*$, $k \leftarrow k + 1$, **go to** Step 4 of Algorithm 23
-

In particular, for each vertex $v \in V(T)$ that is not a pseudovertex of T , we have $\Delta f_v = -\delta$ if $v \in \text{outer}$ or $\Delta f_v = \delta$ if $v \in \text{inner}$. For each vertex $v \in V(T)$ that is an outer pseudovertex of T corresponding to a set $S \in O$, we have that $z'_S - z_S = 2\delta$ and that $\Delta f_w = -\delta$ for each vertex w contained in the set S (recall that if a vertex w is contained in an outer pseudovertex of a Hungarian tree, then $w \in \text{outer}$), implying that

$$\delta f_S = |S|(-\delta) + \frac{1}{2}(|S|-1)(2\delta) = -\delta.$$

For each vertex $v \in V(T)$ that is an inner pseudovertex of T corresponding to a set $S \in O$, we have that $z'_S - z_S = -2\delta$ and that $\Delta f_w = \delta$ for each vertex w contained in the set S (recall that if a vertex w is contained in an inner pseudovertex of a Hungarian tree, then $w \in \text{inner}$), implying that

$$\delta f_S = |S|(\delta) + \frac{1}{2}(|S|-1)(-2\delta) = \delta.$$

Thus, every outer vertex or outer pseudovertex of T contributes $-\delta$ to the change in the dual objective function, while every inner vertex or inner pseudovertex of T contributes δ to the change in the dual objective function. The values of all other dual variables remains unchanged. Since the total number of outer vertices and outer pseudovertices of T is exactly one more than the total number of inner vertices and inner pseudovertices of T , we have that $\Delta f = -\delta$. Since $\delta > 0$, the objective function $f(G)$ of the dual problem strictly decreases after each iteration of the algorithm. ■

Claim (R2) *Requirement (R2) is satisfied at all times by the algorithm.*

Proof At each stage of the algorithm, the primal variable $x_e = 1$ if the edge e belongs to the matching M , or $x_e = 0$ otherwise. Thus, x_e satisfies the three primal constraints (P1), (P2) and (P3). ■

Claim (R3) *Requirement (R3) is satisfied at all times by the algorithm.*

Proof For the initial solution it holds that $y_v \geq 0$ for all $v \in V$. At each stage of the algorithm, the dual interchange implies that $y'_v \geq y_v$ if $v \notin \text{outer}$, while $y'_v = y_v - \delta$ if $v \in \text{outer}$. If, however, $v \in \text{outer}$, then $\delta \leq \delta_1 \leq y_v$. In both cases, $y'_v \geq 0$. Hence, the dual constraint (D1) is satisfied.

It furthermore holds for the initial solution that $z_S = 0$ for all $S \in O$. At each stage of the algorithm, the dual interchange implies that $z'_S = z_S - 2\delta$ if S is the set of vertices belonging to an inner pseudovertex of T , or $z'_S \geq z_S$ otherwise. If, however, S is the set of vertices belonging to an outer pseudovertex of T , then $\delta \leq \delta_4 \leq z_S/2$. In both cases, $z'_S \geq 0$. Hence, the dual constraint (D2) is satisfied.

To show that the dual constraint (D3) is satisfied, we show that $\pi(uv) \geq 0$ for all $uv \in E$. Recall that

$$\pi(uv) = y_u + y_v - w_{uv} + \sum_{S \in O_{uv}} z_S.$$

For the initial solution we have that $\pi(uv) = y_u + y_v - w_{uv} \geq \frac{1}{2}w_{uv} + \frac{1}{2}w_{uv} - w_{uv} \geq 0$ for all $uv \in E$. Let $uv \in E$ and let

$$\pi(uv)' = y'_u + y'_v - w_{uv} + \sum_{S \in O_{uv}} z'_S$$

denote the new value of $\pi(uv)$ after the dual adjustment. Suppose first that the edge uv is contained in some pseudovertex w of T . Let S be the set of vertices belonging to this pseudovertex of T . If w is an outer pseudovertex of T , then $u \in \text{outer}$ and $v \in \text{outer}$. Thus, $y'_u = y_u - \delta$, $y'_v = y_v - \delta$ and $z'_S = z_S + 2\delta$, implying that $\pi(uv)' = \pi(uv)$. If w is an inner pseudovertex of T , then $u \in \text{inner}$ and $v \in \text{inner}$. Thus, $y'_u = y_u + \delta$, $y'_v = y_v + \delta$ and $z'_S = z_S - 2\delta$, implying that $\pi(uv)' = \pi(uv)$. In both cases, $\pi(uv)' \geq 0$.

Suppose secondly that the edge uv is not contained in any pseudovertex of T . Then $\pi(uv) = y_u + y_v - w_{uv}$. The dual interchange implies the following. If $u \in \text{outer}$ and $v \in \text{inner}$, then $y'_u = y_u - \delta$ and $y'_v = y_v + \delta$, and so $\pi(uv)' = \pi(uv)$. If $u \in \text{outer}$ and $v \in \text{outer}$, then $y'_u = y_u - \delta$ and $y'_v = y_v - \delta$, and so $\pi(uv)' = \pi(uv) - 2\delta$. In this case, however, $\delta \leq \delta_3 \leq \frac{1}{2}\pi(uv)$, and so $\pi(uv)' = \pi(uv) - 2\delta \geq 0$. If $u \in \text{outer}$ and $v \in \text{unlabelset}$, then $y'_u = y_u - \delta$ and $y'_v = y_v$, and so $\pi(uv)' = \pi(uv) - \delta$. In this case, $\delta \leq \delta_2 \leq \pi(uv)$, and so $\pi(uv)' = \pi(uv) - \delta \geq 0$. If $u \in \text{inner}$ and $v \in \text{inner}$, then $y'_u = y_u + \delta$ and $y'_v = y_v + \delta$, and so $\pi(uv)' = \pi(uv) + 2\delta$. If $u \in \text{inner}$ and $v \in \text{unlabelset}$, then $y'_u = y_u + \delta$ and $y'_v = y_v$, and so $\pi(uv)' = \pi(uv) + \delta$. In all cases, we have that $\pi(uv)' \geq 0$. Hence the dual constraint (D3) is satisfied. ■

Claim (R4) Requirement (R4) is satisfied at all times by the algorithm.

Proof Suppose that $x_{uv} > 0$ for some $uv \in E$. Then the edge $uv \in M$. Every edge in the matching is, however, a tight edge, and so $\pi(uv) = 0$. Thus, the complementary slackness condition (CS1) is satisfied at all times by the algorithm.

Suppose that $z_S > 0$ for some set $S \in O$. The only way in which the dual variable z_S can become positive is if S is the set of vertices contained in a pseudovertex of T . By Step 10 of Algorithm 23, the set S contains an odd cycle consisting of edges in the alternating tree. Such an odd cycle therefore contains $(|S|-1)/2$ edges in the matching M , and so $\sum_{uv \in E_S} x_{uv} = (|S|-1)/2$. Hence, condition (CS3) is satisfied at all times by the algorithm. ■

Claim (R5) Requirement (R5) is satisfied at all times by the algorithm.

Proof At each stage of the algorithm, an unmatched vertex v with $y_v > 0$ is selected. We then use the Alternating-Forest subroutine described in Section 9.8 to produce an augmenting path, an odd cycle, or a Hungarian forest. In all three cases, however, either y_v is reduced to zero or the vertex v is matched, so that the vertex v satisfies Condition (CS2). ■

By Claims (R1)–(R5), Algorithm 23 changes the values of the primal and dual variables during each iteration of the algorithm in such a way that requirements (R1)–(R5) are all satisfied. Since there are $|V| = n$ dual variables y_v , $v \in V$, Claim (R5) implies that after at most n iterations of the algorithm, Condition (CS2) is satisfied for all dual variables y_v . By complementary slackness, the resulting matching is a maximum-weight matching. This completes the proof of Theorem 9.48. ■

9.9.4 A worked example

To illustrate the working of [Algorithm 23](#), consider the weighted graph $G_{9.8}$ shown in [Figure 9.17](#). For our initial solution in [Step 1](#), we start with the empty matching and let $x_e = 0$ for all $uv \in E$. Furthermore, we let $y_v = \frac{1}{2} \max\{w_e \mid e \in \partial(v)\}$ for all $v \in V$ and we set $z_S = 0$ for all $S \in O$. Thus, $y_a = 10$, $y_b = y_c = 11$, $y_d = y_e = 9$, $y_f = 8$, $y_g = y_i = y_j = 7$, and $y_h = 5$.

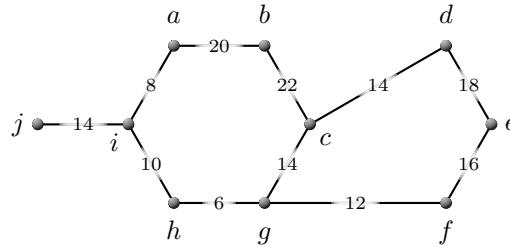


Figure 9.17: A weighted graph $G_{9.8}$.

In [Step 2](#), the set of tight edges is given by $E^* = \{bc, de, ij\}$. Let F_1 denote the original graph $G_{9.8}$ and let F_1^* be the spanning subgraph of F_1 with $E(F_1^*) = E^*$. A maximum matching in F_1^* is given by $M_1 = E_1^* = \{bc, de, ij\}$. Updating the primal variables, we have that $x_{bc} = x_{de} = x_{ij} = 1$ and $x_e = 0$ for all other edges $e \in E \setminus M_1$. Set $k = 1$.

Let $F = F_1$, $M = M_k$ and $E^* = E_k^*$ in [Steps 3](#) and [4](#). Thus,

$$E^* = \{bc, de, ij\} \quad \text{and} \quad M = \{bc, de, ij\}.$$

The primal variables x_e remain unchanged. We write the value of the dual variable y_v inside the vertex v , as illustrated in [Figure 9.18](#). Furthermore, we denote the edges in E^* by double lines, the edges of the matching M by thick lines, and the remaining edges in $E(F) \setminus E^*$ by normal lines. Note that $M \subseteq E^*$ will always hold.

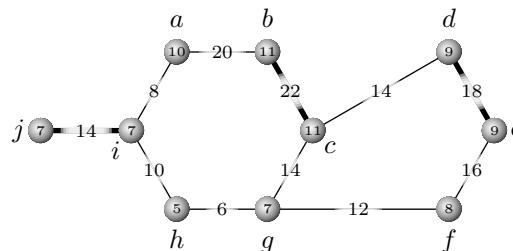


Figure 9.18: The weighted graph F_1 produced during the first iteration of [Algorithm 23](#).

In [Step 5](#), we select the M -unmatched vertex a with $y_a > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in [Step 7](#). The Alternating-Forest subroutine for growing an alternating tree rooted at the vertex a produces a tree consisting entirely of the vertex a (note that the vertex a is incident with no edge in E^*). Label the vertex a as “outer” and label all other vertices as “unlabelled.” Thus, $\text{outer} = \{a\}$, $\text{inner} = \emptyset$ and $\text{unlabelset} = V \setminus \{a\}$.

Since the Alternating-Forest subroutine finds a Hungarian tree T (containing a) in [Step 11](#), we continue on to [Algorithm 23c](#). We now perform a dual adjustment as follows. Since

$$\begin{aligned}\delta_1 &= y_a = 10, \quad \pi(ab) \\ \delta_2 &= \min\left\{\overbrace{y_a + y_b - w(ab)}^{\pi(ab)}, \overbrace{y_a + y_i - w(ai)}^{\pi(ai)}\right\} \\ &= \min\{10 + 11 - 20, 10 + 7 - 8\} = \min\{1, 9\} = 1, \text{ and} \\ \delta_3 &= \delta_4 = \infty,\end{aligned}$$

we let $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{10, 1, \infty, \infty\} = 1$. Moreover, since $\text{outer} = \{a\}$ and $\text{unlabelset} = V \setminus \{a\}$, we let $y'_a = y_a - \delta = 10 - 1 = 9$ and let $y'_v = y_v$ for all other vertices $v \in V$. There are no pseudovertices of T , and so $z'_S = z_S = 0$ for all $S \in O$. Since $\delta = \delta_2$, we let $M_2 = M$ and let $E_2^* = E_1^* \cup \{ab\}$ denote the new set of tight edges after the dual adjustment. Let $F_2 = F$, let $k = 2$ and return to [Step 4](#) of [Algorithm 23](#).

Let $F = F_2$, $M = M_2$ and $E^* = E_2^*$. Thus,

$$E^* = \{ab, bc, de, ij\} \quad \text{and} \quad M = \{bc, de, ij\}.$$

Since the matching is unchanged, the values of the primal variables are unchanged. The values of the dual variables are illustrated in [Figure 9.19](#).

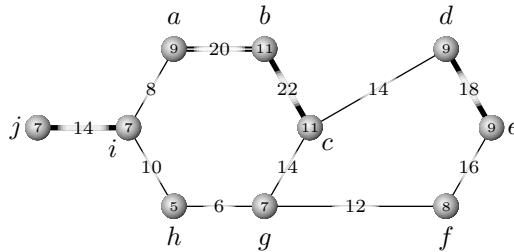


Figure 9.19: The weighted graph F_2 produced during the second iteration of [Algorithm 23](#).

In [Step 5](#), we select the M -unmatched vertex a with $y_a > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in [Step 7](#). The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex a produces the Hungarian tree T with $V(T) = \{a, b, c\}$ and with $E(T) = \{ab, bc\}$ (where $bc \in M$). We label the vertices a and c as “outer,” the vertex b as “inner” and all other vertices as “unlabelled.” Thus, $\text{outer} = \{a, c\}$, $\text{inner} = \{b\}$ and $\text{unlabelset} = V \setminus \{a, b, c\}$.

Since the Alternating-Forest subroutine finds a Hungarian tree in [Step 11](#), we continue on to [Algorithm 23c](#). Let T again denote our Hungarian tree. We perform a dual adjustment as follows. Since

$$\begin{aligned}\delta_1 &= \min\{y_a, y_c\} = \min\{9, 11\} = 9, \\ \delta_2 &= \min\left\{\overbrace{9 + 7 - 8}^{\pi(ai)}, \overbrace{11 + 9 - 14}^{\pi(cd)}, \overbrace{11 + 7 - 14}^{\pi(cg)}\right\} = \min\{8, 6, 4\} = 4, \text{ and} \\ \delta_3 &= \delta_4 = \infty,\end{aligned}$$

we let $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{9, 4, \infty, \infty\} = 4$. We furthermore let

$$y'_w = \begin{cases} y_w - \delta & \text{for } w \in \{a, c\}, \\ y_w + \delta & \text{for } w = b, \\ y_w & \text{for } w \in V \setminus \{a, b, c\}. \end{cases}$$

Thus, $y'_a = y_a - 4 = 5$, $y'_b = y_b + 4 = 15$ and $y'_c = y_c - 4 = 7$, while $y'_v = y_v$ for all other vertices $v \in V$. There are no pseudovertices of T , and so $z'_S = z_S = 0$ for all $S \in O$. Since $\delta = \delta_2$, we let $M_3 = M$ and let $E_3^* = E_3^* \cup \{cg\}$ denote the new set of tight edges after the dual adjustment. Let $F_3 = F$, let $k = 3$ and again return to Step 4 of Algorithm 23.

Let $F = F_3$, $M = M_3$ and $E^* = E_3^*$. Thus,

$$E^* = \{ab, bc, cg, de, ij\} \quad \text{and} \quad M = \{bc, de, ij\}.$$

Since the matching is unchanged, the values of the primal variables are again unchanged. The values of the dual variables are illustrated in Figure 9.20.

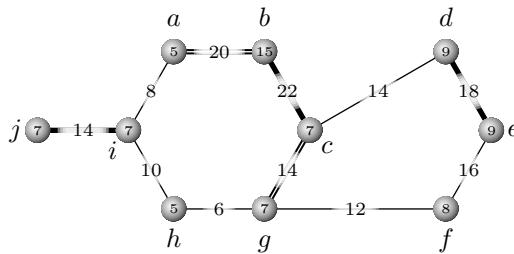


Figure 9.20: The weighted graph F_3 produced during the third iteration of Algorithm 23.

In Step 5, we select the M -unmatched vertex a with $y_a > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex a produces the Hungarian tree T with $V(T) = \{a, b, c, g\}$ and with $E(T) = \{ab, bc, cg\}$.

Since the Alternating-Forest subroutine finds an M -augmenting path $abcg$ in Step 9 (where $bc \in M$ and both a and g are M -unmatched), we continue on to Algorithm 23a. We augment the matching along the M -augmenting path. Let $M_4 = \{ab, cg, de, ij\}$ denote the resulting matching and let $E_4^* = E^* = \{ab, bc, cg, de, ij\}$. The vertex a is now M_4 -matched. Let $F_4 = F$, let $k = 4$ and return to Step 4 of Algorithm 23.

Let $F = F_4$, $M = M_4$ and $E^* = E_4^*$. Thus,

$$E^* = \{ab, bc, cg, de, ij\} \quad \text{and} \quad M = \{ab, cg, de, ij\}.$$

Updating the primal variables, we have that $x_{ab} = x_{cg} = x_{de} = x_{ij} = 1$ and $x_e = 0$ for all other edges $e \in E \setminus M_0$. The new matching M is illustrated in Figure 9.21.

In Step 5, we select the M -unmatched vertex f with $y_f > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine for growing an alternating tree rooted at the vertex f consists entirely of the vertex f (note that the vertex f is incident with no edge in F^*). Label the vertex f as “outer” and all other vertices as “unlabelled.” Thus, $\text{outer} = \{f\}$, $\text{inner} = \emptyset$ and $\text{unlabelset} = V \setminus \{f\}$.

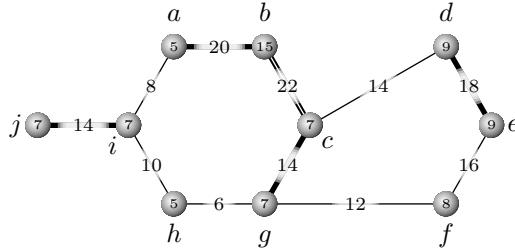


Figure 9.21: The weighted graph F_4 produced during the fourth iteration of Algorithm 23.

Since the Alternating-Forest subroutine finds a Hungarian tree in Step 11, we continue on to Algorithm 23c. Let T denote the Hungarian tree. We now perform a dual adjustment by letting

$$\begin{aligned}\delta_1 &= y_f = 8, \\ \delta_2 &= \min\left\{\overbrace{9+8-16}^{\pi(ef)}, \overbrace{8+7-12}^{\pi(fg)}\right\} = \min\{1, 3\} = 1, \text{ and} \\ \delta_3 &= \delta_4 = \infty,\end{aligned}$$

and $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{8, 1, \infty, \infty\} = 1$. Since $\text{outer} = \{f\}$ and $\text{unlabelset} = V \setminus \{f\}$, we let $y'_f = y_f - \delta = 8 - 1 = 7$ and let $y'_v = y_v$ for all other vertices $v \in V$. There are no pseudovertices of T , and so $z'_S = z_S = 0$ for all $S \in O$. Since $\delta = \delta_2$, we let $M_5 = M$ and let $E_5^* = E_4^* \cup \{ef\}$ denote the new set of tight edges after the dual adjustment. Let $F_5 = F$, let $k = 5$ and return to Step 4 of Algorithm 23 yet again.

Let $F = F_5$, $M = M_5$ and $E^* = E_5^*$. Thus,

$$E^* = \{ab, bc, cg, de, ef, ij\} \quad \text{and} \quad M = \{ab, cg, de, ij\}.$$

Since the matching is unchanged, the values of the primal variables are again unchanged. The values of the dual variables are illustrated in Figure 9.22.

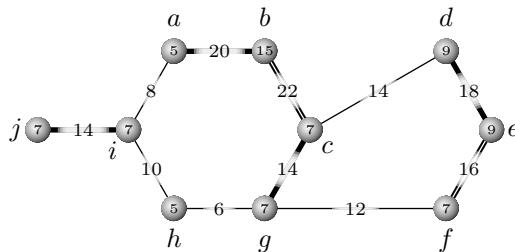


Figure 9.22: The weighted graph F_5 produced during the fifth iteration of Algorithm 23.

In Step 5, we select the M -unmatched vertex f with $y_f > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex f produces the Hungarian tree T with $V(T) = \{d, e, f\}$ and with $E(T) = \{de, ef\}$, where

$de \in M$. Label the vertices d and f as “outer,” the vertex e as “inner,” and all other vertices as “unlabelled.” Thus, $\text{outer} = \{d, f\}$, $\text{inner} = \{e\}$ and $\text{unlabelset} = V \setminus \{d, e, f\}$.

Since the Alternating-Forest subroutine finds a Hungarian tree T in [Step 11](#), we continue on to [Algorithm 23c](#). We perform a dual adjustment by letting

$$\begin{aligned}\delta_1 &= \min\{y_d, y_f\} = \min\{\overbrace{7, 9}^{\pi(cd)}, \overbrace{7, 9}^{\pi(fg)}\} = 7, \\ \delta_2 &= \min\{\underbrace{9 + 7 - 14}_{\pi(cd)}, \underbrace{7 + 7 - 12}_{\pi(fg)}\} = \min\{2, 2\} = 2, \text{ and} \\ \delta_3 &= \delta_4 = \infty,\end{aligned}$$

and $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{7, 2, \infty, \infty\} = 2$. We also let

$$y'_w = \begin{cases} y_w - \delta & \text{for } w \in \{d, f\}, \\ y_w + \delta & \text{for } w = e, \\ y_w & \text{for } w \in V \setminus \{d, e, f\}. \end{cases}$$

Thus, $y'_d = y_d - 2 = 7$, $y'_e = y_e + 2 = 11$ and $y'_f = y_f - 2 = 5$, while $y'_v = y_v$ for all other vertices $v \in V$. There are no pseudovertices of T , and so $z'_S = z_S = 0$ for all $S \in O$. Since $\delta = \delta_2$, we let $M_6 = M$ and let $E_6^* = E_6^* \cup \{cd, fg\}$ denote the new set of tight edges after the dual adjustment. Let $k = 6$ and again return to [Step 4](#) of [Algorithm 23](#).

Let $M = M_6$ and let $E^* = E_6^*$. Thus,

$$E^* = E(F) \setminus \{gh, hi\} \quad \text{and} \quad M = \{ab, cg, de, ij\}.$$

Since the matching is unchanged, the values of the primal variables remain unchanged. The values of the dual variables are illustrated in [Figure 9.23](#).

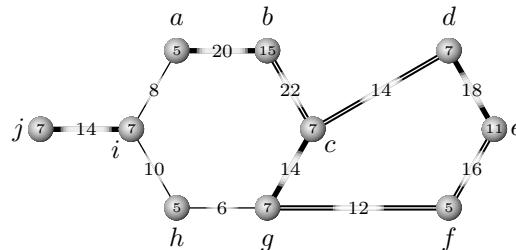


Figure 9.23: The weighted graph F_6 produced during the sixth iteration of [Algorithm 23](#).

In [Step 5](#), we select the M -unmatched vertex f with $y_f > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in [Step 7](#). The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex f produces an odd M -alternating cycle $B_7 : f \rightarrow e \rightarrow c \rightarrow g \rightarrow f$, where $\{de, cf\} \subset M$.

Since the Alternating-Forest subroutine finds an odd cycle in [Step 10](#), we continue on to [Algorithm 23b](#). We shrink the odd cycle $B_{k+1} = B_7$ into a pseudovertex B_7 . Let F_7 denote the resulting graph, let $M_7 = \{ab, ij\}$ and let $E_7^* = E_6^*$. Furthermore, let $k = 7$ and return to [Step 4](#) of [Algorithm 23](#).

Now let $F = F_7$, $M = M_7$ and $E^* = E_7^*$. Thus,

$$E^* = \{ab, b_7, ij\} \quad \text{and} \quad M = \{ab, ij\}.$$

We update the primal variables x_e so that $x_e = 1$ if $e \in M$, or $x_e = 0$ otherwise. The values of the dual variables are illustrated in Figure 9.24.

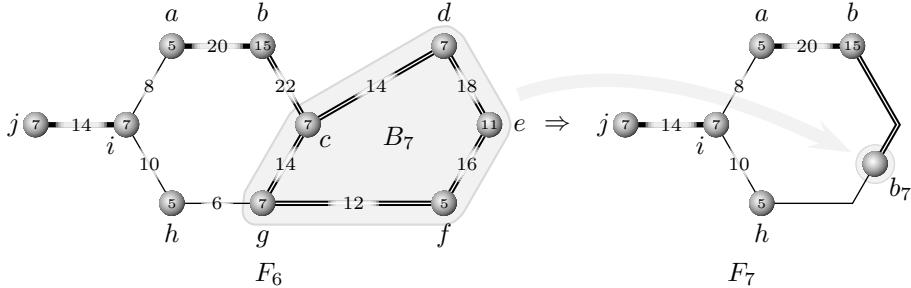


Figure 9.24: The weighted graph F_7 produced during the seventh iteration of Algorithm 23.

In Step 5, we select the pseudovertex b_7 . Note that every vertex w in the odd cycle B_7 associated with the pseudovertex b_7 has $y_w > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex b_7 produces the Hungarian tree T with $V(T) = \{a, b, b_7\}$ and with $E(T) = \{ab, bb_7\}$ (where $ab \in M$). We label the vertices a and b_7 as “outer,” the vertex b as “inner” and all other vertices as “unlabelled.” Since the pseudovertex b_7 is labelled “outer,” so too are all vertices inside the odd cycle B_7 . Thus, $\text{outer} = \{a, b_7, c, d, e, f, g\}$, $\text{inner} = \{b\}$ and $\text{unlabelset} = \{h, i, j\}$.

Since the Alternating-Forest subroutine finds a Hungarian tree in Step 11, we continue on to Algorithm 23c. Let T denote the Hungarian tree. We perform a dual adjustment as follows. Note that there is no inner pseudovertex, and so $\delta_4 = \infty$. Furthermore, since

$$\delta_1 = \min\{y_a, y_c, y_d, y_e, y_f, y_g\} = \min\{5, 7, 7, 11, 5, 7\} = 5,$$

$$\delta_2 = \min\{\overbrace{5+7-8}^{\pi(ai)}, \overbrace{7+5-6}^{\pi(gh)}\} = \min\{4, 6\} = 4, \text{ and}$$

$$\delta_3 = \delta_4 = \infty,$$

we let $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{5, 4, \infty, \infty\} = 4$. We also let

$$y'_w = \begin{cases} y_w - \delta & \text{for } w \in \{a, c, d, e, f, g\}, \\ y_w + \delta & \text{for } w = b, \\ y_w & \text{for } w \in V \setminus \{h, i, j\}. \end{cases}$$

Thus, $y'_a = 1$, $y'_b = 19$, $y'_c = 3$, $y'_d = 3$, $y'_e = 7$, $y'_f = 1$, $y'_g = 3$, $y'_h = 5$, $y'_i = 7$ and $y'_j = 7$. Since b_7 is an outer pseudovertex of T , we let $z'_S = z_S + 2\delta = 0 + 8 = 8$ for $S = V(B_7)$ and we let $z'_S = z_S = 0$ for all other sets $S \in O$. Since $\delta = \delta_2$, we furthermore let $F_8 = F$, $M_8 = M$ and let $E_8^* = E_7^* \cup \{ai\}$ denote the new set of tight edges in F_8 after the dual adjustment. Let $k = 8$ and return to Step 4 of Algorithm 23 yet again.

Let $F = F_8$, $M = M_8$ and let $E^* = E_8^*$, so that

$$E^* = \{ab, ai, bb_7, ij\} \quad \text{and} \quad M = \{ab, ij\}.$$

Since the matching M is unchanged, the values of the primal variables are unchanged. The values of the dual variables are illustrated in Figure 9.25.

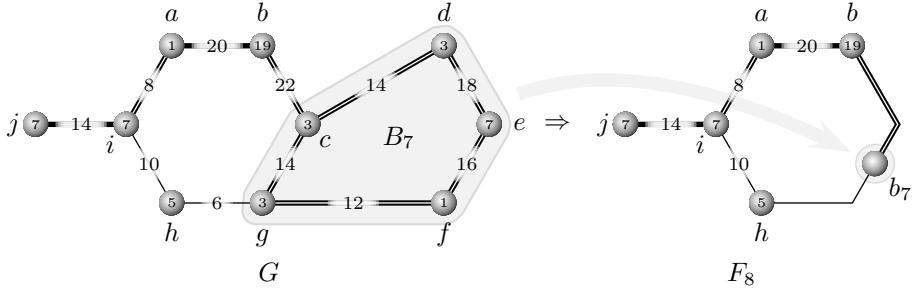


Figure 9.25: The weighted graph F_8 produced during the eighth iteration of Algorithm 23.

In Step 5 we select the pseudovertex b_7 . Note that every vertex w in the odd cycle B_7 associated with the pseudovertex b_7 has $y_w > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex b_7 produces the Hungarian tree T with $V(T) = \{a, b, b_7, i, j\}$ and with $E(T) = \{ab, bb_7, ai, ij\}$ (where $ab, ij \in M$). We label the vertices a, b_7, j as “outer,” the vertices b, i as “inner” and all other vertices as “unlabelled.” Since the pseudovertex b_7 is labelled “outer,” so too are all vertices inside the odd cycle B_7 . Thus, $\text{outer} = \{a, b_7, c, d, e, f, g, j\}$, $\text{inner} = \{b, i\}$ and $\text{unlabelset} = \{h\}$.

Since the Alternating-Forest subroutine finds a Hungarian tree T in Step 11, we continue on to Algorithm 23c. We now perform a dual adjustment as follows. Note again that there is no inner pseudovertex, and so $\delta_4 = \infty$. Furthermore, since

$$\delta_1 = \min\{y_a, y_c, y_d, y_e, y_f, y_g, y_j\} = \min\{1, 3, 3, 7, 1, 3, 7\} = 1,$$

$$\delta_2 = \min\{\overbrace{3+5-6}^{\pi(gh)}, \overbrace{5+7-10}^{\pi(hi)}\} = \min\{2, 2\} = 2, \text{ and}$$

$$\delta_3 = \delta_4 = \infty,$$

we let $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{1, 2, \infty, \infty\} = 1$. We also let

$$y'_w = \begin{cases} y_w - \delta & \text{for } w \in \{a, c, d, e, f, g, j\}, \\ y_w + \delta & \text{for } w \in \{b, i\}, \\ y_w & \text{for } w = h. \end{cases}$$

Thus, $y'_a = 0$, $y'_b = 20$, $y'_c = 2$, $y'_d = 2$, $y'_e = 6$, $y'_f = 0$, $y'_g = 2$, $y'_h = 5$, $y'_i = 8$ and $y'_j = 6$. Since b_7 is an outer pseudovertex of T , we let $z'_S = z_S + 2\delta = 8 + 2 = 10$ for $S = V(B_7)$ and we let $z'_S = z_S = 0$ for all other sets $S \in O$. Since $\delta = \delta_1$, we augment the matching M along the M -alternating path b_7ba in T . Let $M_9 = \{bb_7, ij\}$ denote the resulting matching. Furthermore, let $F_9 = F$, let $E_9^* = E^*$, let $k = 9$ and return to Step 4 of Algorithm 23.

Now let $F = F_9$, $M = M_9$ and $E^* = E_9^*$. Thus,

$$E^* = \{ab, ai, bb_7, ij\} \quad \text{and} \quad M = \{bb_7, ij\}.$$

We update the primal variables x_e so that $x_e = 1$ if $e \in M$, or $x_e = 0$ otherwise. The values of the dual variables are illustrated in Figure 9.26.

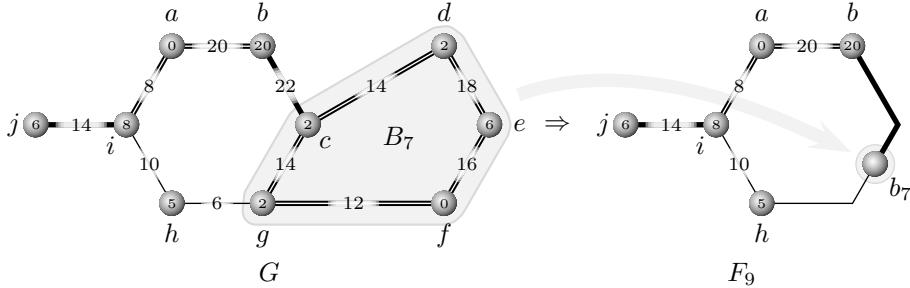


Figure 9.26: The weighted graph F_9 produced during the ninth iteration of Algorithm 23.

In Step 5, we select the M -unmatched vertex h with $y_h > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine for growing an alternating tree rooted at the vertex h consists entirely of the vertex h (note that the vertex h is incident with no edge in F^*). Label the vertex h as “outer” and all other vertices as “unlabelled.” Thus, $\text{outer} = \{h\}$, $\text{inner} = \emptyset$ and $\text{unlabelset} = V \setminus \{h\}$.

Since the Alternating-Forest subroutine finds a Hungarian tree T in Step 11, we continue on to Algorithm 23c. We again perform a dual adjustment as follows. Since

$$\delta_1 = y_h = 5,$$

$$\delta_2 = \min\left\{\overbrace{5+2-6}^{\pi(hg)}, \overbrace{8+5-10}^{\pi(hi)}\right\} = \min\{1, 3\} = 1, \text{ and}$$

$$\delta_3 = \delta_4 = \infty,$$

we let $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4\} = \min\{5, 1, \infty, \infty\} = 1$. Furthermore, since $\text{outer} = \{h\}$ and $\text{unlabelset} = V \setminus \{h\}$, we let $y'_h = y_a - \delta = 4$ and let $y'_v = y_v$ for all other vertices $v \in V$. Since there are no pseudovertices in T , we let $z'_S = z_S$ for all $S \in O$, and since $\delta = \delta_2$, we let $F_{10} = F$, $M_{10} = M$ and we let $E_{10}^* = E_9^* \cup \{b_7h\}$ denote the new set of tight edges after the dual adjustment. Let $k = 10$ and return to Step 4 of Algorithm 23.

Let $F = F_{10}$, $M = M_{10}$ and $E^* = E_{10}^*$. Thus,

$$E^* = \{ab, ai, bb_7, b_7h, ij\} \quad \text{and} \quad M = \{bb_7, ij\}.$$

Since the matching is unchanged, the values of the primal variables are unchanged. The values of the dual variables are illustrated in Figure 9.27.

In Step 5, we select the M -unmatched vertex h with $y_h > 0$. Let F^* be the spanning subgraph of F with $E(F^*) = E^*$ in Step 7. The Alternating-Forest subroutine in F^* for growing an alternating tree rooted at the vertex h produces

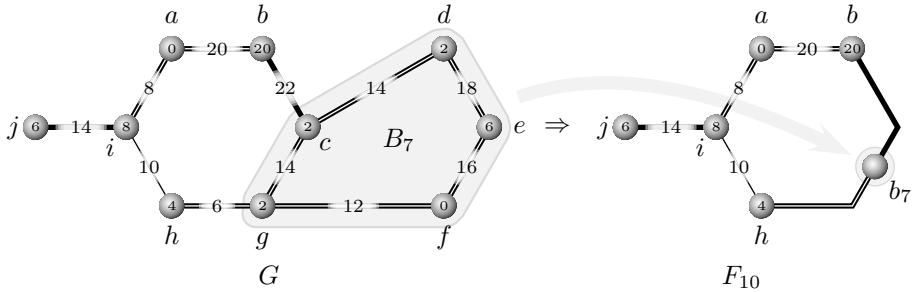


Figure 9.27: The weighted graph F_{10} produced during the tenth iteration of Algorithm 23.

an M -augmenting path h, b_7, b, a in Step 9 (where $b_7b \in M$ and both h and a are M -unmatched), and so we continue on to Algorithm 23a.

We augment the matching along the M -augmenting path. Let $M_{11} = \{ab, b_7h, ij\}$ denote the resulting matching and let $E_{11}^* = E^*$. The vertex h is now M_{11} -matched. Let $F_{11} = F$, let $k = 4$ and return to Step 4 of Algorithm 23.

Now $F = F_{11}$, $M = M_{11}$ and $E^* = E_{11}^*$, so that

$$E^* = \{ab, ai, bb_7, b_7h, ij\} \quad \text{and} \quad M = \{ab, b_7h, ij\}.$$

We update the primal variables x_e so that $x_e = 1$ if $e \in M$, or $x_e = 0$ otherwise. The values of the dual variables are illustrated in Figure 9.28.

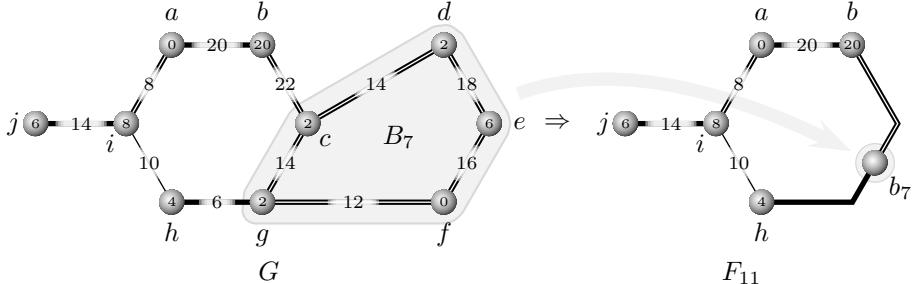


Figure 9.28: The weighted graph F_{11} produced during the eleventh iteration of Algorithm 23.

Since there is no M -unmatched vertex in F , we go to Step 13. We now expand the pseudovertex b_7 back to the original odd cycle B_7 and let F_{12} denote the expanded graph. Let $M_{12} = \{ab, cd, ef, gh, ij\}$ denote the modified matching in F_{12} . Furthermore, let $E_{12}^* = E(F_{12}) \setminus \{hi\}$, let $F = F_{12}$, $M = M_{12}$ and let $E^* = E_{12}^*$. We note that M is a perfect matching in F . Updating the primal variables x_e so that $x_e = 1$ if e is in the matching M , or $x_e = 0$ otherwise, we have that

$$x_e = \begin{cases} 1 & \text{for } e \in \{ab, cd, ef, gh, ij\}, \\ 0 & \text{otherwise.} \end{cases}$$

The values of the dual variables y_v and z_S remain unchanged. Thus, $y_a = 0$, $y_b = 20$, $y_c = 2$, $y_d = 2$, $y_e = 6$, $y_f = 0$, $y_g = 2$, $y_h = 4$, $y_i = 8$ and $y_j = 6$.

Moreover, letting $S^* = \{c, d, e, f, g\}$, we have that $z_{S^*} = 10$ and $z_S = 0$ for all other sets $S \in O \setminus \{S^*\}$. The values of the dual variables are illustrated in Figure 9.29, as is the matching M .

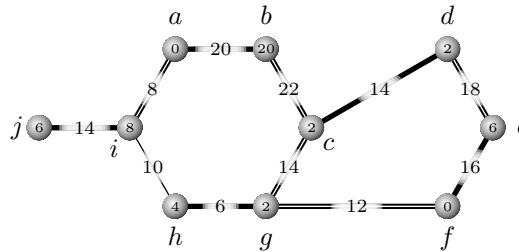


Figure 9.29: The weighted graph F_{12} produced during the twelfth iteration of Algorithm 23.

We note that the primal variables satisfy all three primal constraints, namely (P1), (P2) and (P3), since they are associated with a matching in the graph. Furthermore the three dual constraints, namely (D1), (D2) and (D3), are all satisfied, as are the three complementary slackness conditions (CS1), (CS2) and (CS3). By the Complementary Slackness Theorem (Theorem 9.47), the final primal and dual variables are optimal solutions to their respective problems. The value of the primal objective function is

$$\sum_{e \in E} w_e x_e = w(ab) + w(cd) + w(ef) + w(gh) + w(ij) = 20 + 14 + 16 + 6 + 14 = 70,$$

while that of the dual objective function is

$$\sum_{v \in V} y_v + \sum_{S \in O} \frac{1}{2}(|S| - 1)z_S = 50 + \frac{1}{2}(|S^*| - 1)z_{S^*} = 50 + \frac{1}{2}(5 - 1)10 = 70,$$

where recall that $S^* = \{c, d, e, f, g\}$. A maximum weight matching is therefore given by $M = \{ab, cd, ef, gh, ij\}$ with weight $w(M) = 70$.

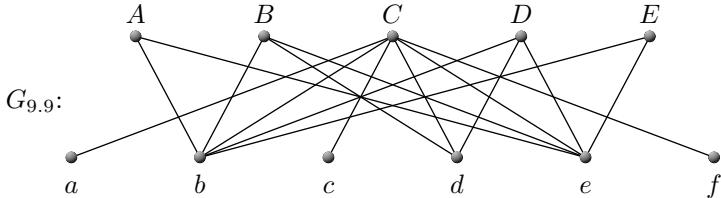
❖ The reader should now be able to attempt Exercise 9.24.

Exercises

- 9.1 In a company there are four employees x_1, x_2, x_3 and x_4 who have to perform tasks y_1, y_2, y_3 and y_4 . Suppose that x_1 can perform jobs y_1, y_2, y_3 , that x_2 can perform job y_1 , that x_3 can perform jobs y_1, y_4 , and that x_4 can perform jobs y_1, y_4 . Practical considerations dictate that no employee may perform more than one job and that jobs may not be shared. Using Theorem 9.2, show that the jobs cannot be allocated in such a way that
- (i) every job is performed;
 - (ii) every employee is assigned to perform a job.

- 9.2 Let M be a maximum matching in a graph G and let v be an M -unmatched vertex of G . Prove or disprove the following statement: The vertex v is an M^* -unmatched vertex of G for every maximum matching M^* in G .

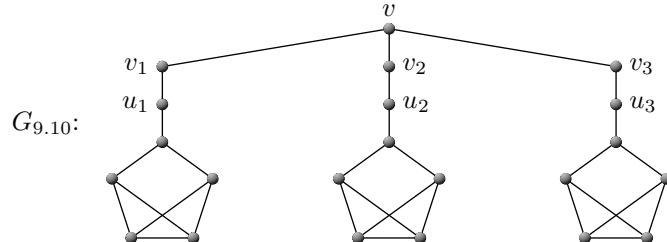
- 9.3 There are five women [Alice (A), Barbara (B), Clair (C), Debbie (D) and Edith (E)] and six men [Albert (a), Ben (b), Charles (c), Derek (d), Ed (e) and Frank (f)] at a party. The compatible dancing partnerships are modelled by the graph $G_{9.9}$ shown below. Is it possible to have all women dancing so that each dancing partnership is a compatible one?



- 9.4 Six professors A, B, C, D, E and F are members of four subcommittees of the Board of the Faculty. The membership of the committees is $\{B, C, D\}$, $\{A, E, F\}$, $\{A, B, E, F\}$, and $\{A, B, C\}$. In a wave of bureaucracy, it is decided to increase the membership of each committee by one, in such a way that no professor will join more than one additional committee. How can this be done?
- 9.5 Show that it is impossible, using 1×2 rectangles, to cover exactly an 8×8 square grid from which two diagonally opposite 1×1 corner squares have been removed.
- 9.6 (a) Show that a bipartite graph G has a perfect matching if and only if $|N(S)| \geq |S|$ for every nonempty subset S of $V(G)$.
(b) Give an example illustrating that the above statement does not remain valid if the condition that G be bipartite is relaxed.
- 9.7 Show that a tree has at most one perfect matching.
- 9.8 Two people play a game on a graph G by alternately selecting distinct vertices v_0, v_1, v_2, \dots such that, for $i > 0$, v_i is adjacent to v_{i-1} . The last player able to select a vertex wins. Show that the first player has a winning strategy if and only if G has no perfect matching.
- 9.9 A **line** of a matrix is a row or a column of the matrix. Show that the minimum number of lines containing all the 1's of a binary matrix is equal to the maximum number of 1's, no two of which are in the same line.
- 9.10 Deduce **Hall's Theorem** ([Theorem 9.2](#)) from the **König-Egerváry Theorem** ([Theorem 9.7](#)).
- 9.11 If T is a matching-tree of order n , then, by [Observation 9.8](#), $\alpha(T) = \frac{n}{2}$. Prove the slightly stronger result that if x is an arbitrary vertex of T , then there exists a vertex cover X in T such that $|X| = \frac{n}{2}$ and $x \in X$.
- 9.12 Construct a family of connected graphs G that are not matching-trees and such that
- $$4\alpha'(G) < 4\beta(G) = n(G) + m(G) + 2 \text{ doc}(G).$$
- 9.13 For each integer $r \geq 2$, construct a connected r -regular graph that has no perfect matching.

9.14 Use [Tutte's Theorem \(Theorem 9.14\)](#) to show that if G is a graph of order $2n$ with minimum degree n , then G contains a perfect matching.

9.15 Use [Tutte's Theorem \(Theorem 9.14\)](#) to show that the graph $G_{9.10}$ below has no perfect matching.



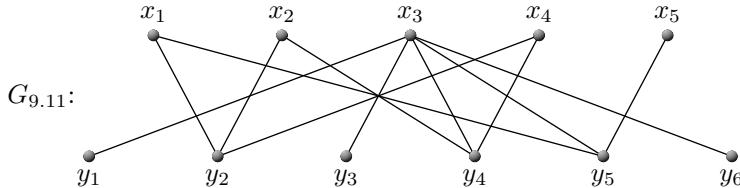
9.16 Prove [Corollary 9.24](#).

9.17 Prove that $0 \leq \text{bind}(G) \leq \delta(G)$ for every graph G .

9.18 Let G be a graph of order n . Prove that $\text{bind}(G) = n - 1$ if and only if $G = K_n$.

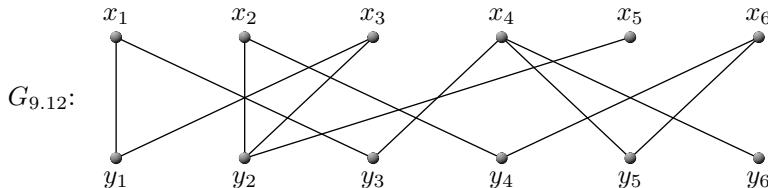
9.19 Let $G = K_{n_1, \dots, n_k}$ be a complete k -partite multigraph of order n where $1 \leq n_1 \leq \dots \leq n_k$ and where $k \geq 2$. Prove that $\text{bind}(G) = (n - n_k)/n_k$.

9.20 Let $G_{9.11}$ be the bipartite graph shown below and let $M = \{x_1y_5, x_3y_4\}$. Starting with the matching M , use [Algorithm 22](#) to find a maximum matching and a minimum cover in $G_{9.11}$.



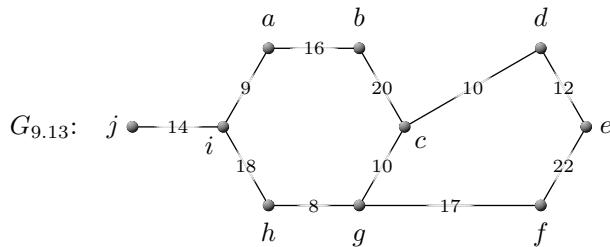
9.21 Using the network flow method, find a maximum matching and a minimum cover in the bipartite graph $G_{9.11}$ in [Exercise 9.21](#).

9.22 Let $G_{9.12}$ be the bipartite graph shown below and let $M = \{x_2y_4, x_3y_1, x_4y_3, x_6y_5\}$. Starting with the matching M , use [Algorithm 22](#) to find a maximum matching and a minimum cover in $G_{9.12}$.



9.23 Using the network flow method, find a maximum matching and a minimum cover in the bipartite graph $G_{9.12}$ in [Exercise 9.23](#).

- 9.24 Use [Algorithm 23](#) to find a maximum weight matching in the graph $G_{9.13}$ below.

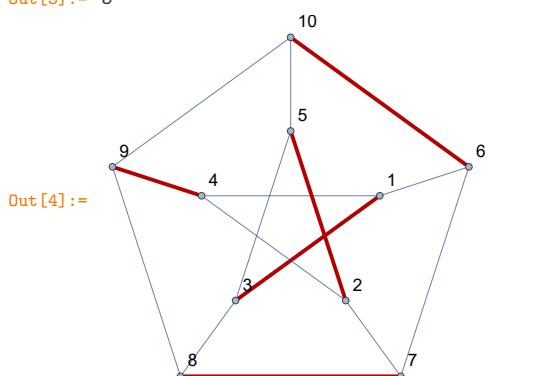
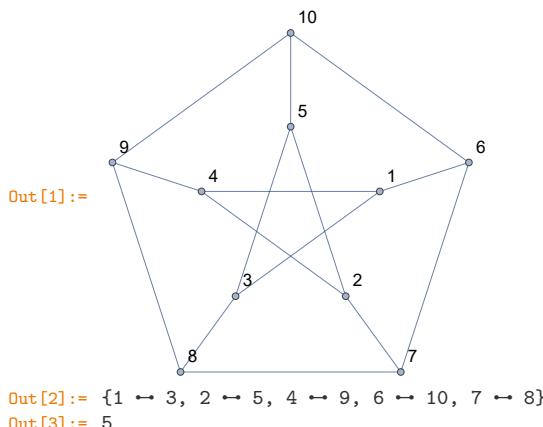


Computer exercises

The command `PetersenGraph` produces the well-known Petersen graph, while the command `FindIndependentEdgeSet[G]` returns a maximum matching in an unweighted graph. The matching number $\alpha'(G)$ of G may further be found by means of the command `Length[FindIndependentEdgeSet[G]]`. For example, the commands

```
In[1]:= g = PetersenGraph[VertexLabels -> "Name"]
In[2]:= e = FindIndependentEdgeSet[g]
In[3]:= Length[e]
In[4]:= HighlightGraph[g, e, GraphHighlightStyle -> "Thick"]
```

produce the output:



Unfortunately there is no **MATHEMATICA** command for computing a maximum weight matching in a weighted graph directly. This may, of course, be done by invoking the command **LinearProgramming** in **MATHEMATICA** in order to solve the primal problem (9.14).

Projects

This section contains two projects, both dedicated to the celebrated assignment problem from the operations research literature. In graph theoretic terms this problem asks for a minimum weight matching in a complete bipartite graph with partite sets of equal size. The aim of the first project is to familiarise the reader with the assignment problem and its relation to the bipartite minimum weight matching problem, while that of the second project is to outline how the well-known *Hungarian method* may be used to solve the assignment problem efficiently.

Project 9.1: The assignment problem

The (classical) *assignment problem* dates back to the celebrated 1952 paper by [Votaw and Orden](#) [46]. In this problem, a one-to-one matching between n tasks and n agents is sought, the objective being to minimise the total cost of all the assignments. As [Pentico](#) [38] points out, “classic examples involve such situations as assigning jobs to machines, jobs to workers, or workers to machines.”

Let c_{ij} denote the cost of assigning agent i to perform task j and let \mathbf{C} be an $n \times n$ cost matrix containing as entry in row i and column j the assignment cost value c_{ij} . Then an optimal solution to the assignment problem corresponds to a smallest sum of the costs c_{ij} of agents actually assigned to perform tasks, *i.e.* a selection of binary x_{ij} values that minimises

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

subject to the constraints

$$\begin{aligned} \sum_{i=1}^n x_{ij} &= 1, \quad j \in [n] && \text{(each task performed by one agent)} \\ \sum_{j=1}^n x_{ij} &= 1, \quad i \in [n] && \text{(each agent performs one task),} \end{aligned}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to perform task } j \\ 0 & \text{otherwise} \end{cases}$$

for all $i, j \in [n]$.

The above problem formulation seems restrictive in that the number of agents and the number of tasks are required to be the same (*i.e.* the cost matrix \mathbf{C} has to be an $n \times n$ matrix). This deficiency may, however, be overcome by adding additional dummy rows to the cost matrix of an assignment problem in which there are fewer agents than tasks. These dummy rows should be populated with very large costs. Additional dummy columns may similarly be added to the cost matrix of an assignment problem in which there are fewer tasks than agents.

Tasks

- Suppose c_{ij}^* is the smallest cost in row i and in column j of the cost matrix \mathbf{C} of an assignment problem. Will agent i necessarily be assigned to perform task j in an optimal solution to the problem? Motivate your answer.
- Suppose five auditors are available to perform four auditing tasks. The estimated time it will take each auditor to perform each task is given in [Table 9.3](#). A ‘—’ in the table indicates that an auditor is not qualified to perform a particular auditing task. Since the auditors are paid per hour, the objective is to assign exactly one auditor to each task in such a manner that the total expected time required to complete all four tasks is minimised. Model this assignment problem as a minimum weight matching problem in a bipartite graph.

Auditor	Time (hours)			
	Task 1	Task 2	Task 3	Task 4
1	44	36	60	36
2	36	—	54	44
3	52	40	56	56
4	32	44	—	28
5	42	—	50	56

Table 9.3: The estimated times it will take auditors to perform auditing tasks.

- Use [Algorithm 23](#) to solve the problem modelled in [Task 2](#) above. Make a recommendation, based on your solution, as to the assignment of auditors to tasks. (Hint: Recall that [Algorithm 23](#) finds a *maximum* weight matching while a *minimum* time assignment is sought here. A minimum weight matching may, of course, be found (using [Algorithm 23](#)) by taking the negations of the times in [Table 9.3](#) as edge weights.)

Project 9.2: The Hungarian method

An interesting, efficient alternative solution method for the assignment problem was put forward by [Kuhn](#) [31] in 1955. This method is today widely known as the *Hungarian method* because of the relationship between [Kuhn](#)'s proposed technique and its antecedents in papers by two Hungarian mathematicians. The method works as follows:

Step 1: Find the smallest element in each row of the cost matrix \mathbf{C} and construct a new matrix \mathbf{C}' by subtracting from each element of \mathbf{C} the minimum cost in its row. For this new matrix \mathbf{C}' , find the smallest element in each column and construct another matrix \mathbf{C}'' by subtracting from each element of \mathbf{C}' the minimum cost in its column. The matrix \mathbf{C}'' is called the **reduced cost matrix** of \mathbf{C} .

Step 2: Draw the smallest number of lines (horizontal, vertical, or a combination of these) required to cover all the zeros in the current reduced cost matrix \mathbf{C}'' .

If n such lines are required, then an optimal solution to the assignment problem is available among the zeros in the matrix. If, however, fewer than n lines suffice to cover all the zeros, proceed to [Step 3](#).

Step 3: Find the smallest nonzero element in the current reduced cost matrix that is not covered by any of the lines drawn in [Step 2](#). Suppose the value of this element is k . Subtract k from each uncovered element in the reduced cost matrix and add k to each element of the matrix that is covered by *two* lines to form a new reduced cost matrix \mathbf{C}'' . Return to [Step 2](#).

The rationale behind the Hungarian method, as we shall show in this project, is that any optimal solution to the assignment problem with any of the reduced cost matrices \mathbf{C}'' produced by repeated application of the loop spanning Steps 2–3 as cost matrix is also an optimal solution to the assignment problem with the original matrix \mathbf{C} as cost matrix. Furthermore, the number of zeros in the reduced cost matrix strictly increases with each application of Steps 2–3 of the method, as will also become clear during the course of this project. As a result of the stopping criterion in [Step 2](#), a situation will therefore be encountered after a finite number of applications of the loop spanning Steps 2–3 where at least n lines are required to cover all the zeros in the reduced cost matrix. At that point there will be a selection of n zeros from the final reduced cost matrix with the property that there is exactly one of these zeros in each row and in each column of the matrix. A one-to-one assignment of agents to tasks associated with these zero costs will therefore represent an assignment of total cost zero of agents to tasks. Since all the entries in any of the reduced cost matrices are nonnegative (why is this the case?), this assignment will be an optimal solution with the last reduced cost matrix as cost matrix, and hence also an optimal solution of the original assignment problem.

Consider, for example, the cost matrix

$$\mathbf{C} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{matrix} 14 & \underline{5^*} & 8 & 7 \\ 2^* & 12 & 6 & \underline{5} \\ 7 & 8 & \underline{3^*} & 9 \\ \underline{2^*} & 4 & 6 & 10 \end{matrix} \right] \end{matrix}$$

in which the rows represent four agents, the columns represent four tasks to be performed by these agents (exactly one agent per task and exactly one task per agent), and the smallest element of each row is denoted by an asterisk. Then the matrix

$$\mathbf{C}' = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{matrix} 9 & 0^* & 3 & 2^* \\ 0^* & 10 & 4 & 3 \\ 4 & 5 & 0^* & 6 \\ 0^* & 2 & 4 & 8 \end{matrix} \right] \end{matrix}$$

is obtained in which the smallest element of each column is now denoted by an asterisk. [Step 1](#) of the Hungarian method is completed by computing the reduced

cost matrix

$$\mathbf{C}'' = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{matrix} 9 & 0 & 3 & 0 \\ 0 & 10 & 4 & 1 \\ 4 & 5 & 0 & 4 \\ 0 & 2 & 4 & 6 \end{matrix} \right] \end{matrix}$$

In [Step 2](#), the five zeros in the \mathbf{C}'' above can be covered by three lines: horizontal lines through the first and third rows as well as a vertical line through the first column. We therefore continue to [Step 3](#), where $k = 1$ and the new reduced cost matrix

$$\mathbf{C}'' = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{matrix} 10 & \underline{0} & 3 & 0 \\ 0 & 9 & 3 & \underline{0} \\ 5 & 5 & \underline{0} & 4 \\ \underline{0} & 1 & 3 & 5 \end{matrix} \right] \end{matrix}$$

is computed. Note that this matrix has one zero more than the previous reduced cost matrix. Returning to [Step 2](#), the six zeros in the last reduced cost matrix \mathbf{C}'' above can no longer be covered by three lines or fewer. Hence we have a (unique) optimal solution when agent 1 is assigned to task 2, agent 2 is assigned to task 4, agent 3 is assigned to task 3, and agent 4 is assigned to task 1. The (reduced) cost of this assignment is zero (as may be found by tallying the underlined zeros) if the last reduced cost matrix is taken as the cost matrix. The cost of this assignment in the original problem may, however, be found by tallying the costs in the same underlined positions within the original matrix, *i.e.* $5 + 5 + 3 + 2 = 15$.

The aim of this project is to guide the reader towards answering questions such as: Will the Hungarian method always terminate within a finite number of steps? And will the Hungarian method always produce an optimal solution?

Tasks

1. Familiarise yourself with the Hungarian method by applying it to an assignment problem with cost matrix

$$\mathbf{C} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{matrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{matrix} \right]. \end{matrix}$$

2. Convince yourself of the correctness of your solution in [Task 1](#) above by solving the same assignment problem by means of [Algorithm 23](#). (Hint: Consider a bipartite graph in which the agents form one partite set and the tasks form the other partite set. Also remember that [Algorithm 23](#) finds a *maximum* weight matching while the Hungarian method finds a *minimum* cost assignment. A minimum weight matching may, of course, be found using [Algorithm 23](#) by taking the negations of the costs in the matrix \mathbf{C} in [Task 1](#) above.)
3. Prove that the entries in all the reduced cost matrices produced by repeated application of the loop spanning Steps 2–3 of the Hungarian method are nonnegative, even if there were negative costs in the original cost matrix \mathbf{C} .

4. Prove that if a constant is added to (or subtracted from) each cost in a row (or column) of the cost matrix of an assignment problem, then the status of any optimal solution to the problem remains unaffected.
5. Prove that Step 3 of the Hungarian method is equivalent to performing the following operations:
 - (a) Add k to each cost that lies in a covered row (of the reduced cost matrix).
 - (b) Subtract k from each cost that lies in a covered column (of the reduced cost matrix).
6. Deduce from Tasks 4–5 above that any optimal solution to an assignment problem with any of the reduced cost matrices C'' produced by repeated application of the loop spanning Steps 2–3 as cost matrix is also an optimal solution to the assignment problem with the original matrix C as cost matrix.
7. Prove that the number of zeros strictly increases with each application of Steps 2–3 of the method.
8. Deduce from Tasks 3 and 6 above that the Hungarian method will terminate within a finite number of steps.
9. Why will an *optimal* assignment of agents to tasks be available among the zeros of the final reduced cost matrix generated by the method?

A difficulty with the Hungarian method lies in the fact that finding the *smallest* number of horizontal and/or vertical lines with which to cover all the zeros in a reduced cost matrix of a very large assignment problem instance may be rather cumbersome. For a structured method to find this minimum number of lines, the reader is referred to the discussion by Gillet [16].

Suggestions for further background reading

A more extensive discussion of matchings in graphs is given by Chartrand and Lesniak [7] and by West [48]. Lovász and Plummer [34] have written a book devoted entirely to the theory of matchings in graphs. Two excellent survey articles on matchings in graphs were written by Plummer [40] and Pulleyblank [41]. The books by Korte and Vygen [30], Schrijver [42], and Evans and Minieka [15] provide excellent coverage of the **maximum matching algorithm** due to Edmonds and the **maximum-weight matching algorithm** due to Edmonds and Johnson.

Further reading

- [1] I Anderson, 1971. *Perfect matchings of a graph*, Journal of Combinatorial Theory, **10**, pp. 183–186.
- [2] C Berge, 1957. *Two theorems on graph theory*, Proceedings of the National Academy of Sciences of the United States of America, **43**, pp. 842–844.
- [3] C Berge, 1958. *Sur le couplage maximum d'un graphe*, Comptes Rendus de l'Académie des Sciences – Series I – Mathematics, **247**, pp. 258–259.

- [4] T Biedl, ED Demaine, CA Duncan, R Fleischer and SG Kobourov, 2004. *Tight bounds on maximal and maximum matchings*, Discrete Mathematics, **285**, pp. 7–15.
- [5] F Buckley and F Harary, 1990. *Distance in Graphs*, Addison-Wesley, Reading (MA).
- [6] G Chartrand and F Harary, 1968. *Graphs with prescribed connectivities*, pp. 61–63 in *Theory of Graphs: Proceedings of the Colloquium Held at Tihany, Hungary*, Budapest.
- [7] G Chartrand and L Lesniak, 1996. *Graphs & Digraphs*, Third edition, Chapman & Hall, London.
- [8] GB Dantzig and MN Thapa, 2003. *Linear Programming — 2 Theory and Extensions*, Springer, New York (NY).
- [9] R Diestel, 1997. *Graph Theory*, Springer, New York (NY).
- [10] GA Dirac, 1960. *In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre Unterteilungen*, Mathematische Nachrichten, **22**, pp. 61–85.
- [11] J Edmonds, 1965. *Paths, trees, and flowers*, Canadian Journal of Mathematics, **17**, pp. 449–467.
- [12] E Egerváry, 1931. *On combinatorial properties of matrices*, Matematikai Lapok, **38**, pp. 16–28.
- [13] P Erdős, A Rubin and H Taylor, 1980. *Choosability in graphs*, pp. 125–157 in *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing; Congressus Numerantium*, Utilitas Mathematica Publishing Inc.
- [14] L Euler, 1758. *Demonstratio nunnularum insignium proprietatum quibus solida hedris planis inclusa sunt praedita*, Novi commentarii academiae scientiarum Petropolitanae, **4**, pp. 140–160.
- [15] JR Evans and E Minieka, 1992. *Optimization Algorithms for Networks and Graphs*, Second edition, Marcel Dekker, New York (NY).
- [16] B Gillet, 1976. *Introduction to Operations Research: A Computer-oriented Algorithmic Approach*, McGraw-Hill, New York (NY).
- [17] WD Goddard, 1998. *On the vulnerability of graphs*, PhD thesis, University of Natal, Durban.
- [18] P Hall, 1935. *On representation of subsets*, Journal of the London Mathematical Society, **10**, pp. 26–30.
- [19] PE Haxell and AD Scott, 2017. *On lower bounds for the matching number of subcubic graphs*, Journal of Graph Theory, **85(2)**, pp. 336–348.
- [20] MA Henning and ZB Shozi, 2021. *A characterization of graphs with given maximum degree and smallest possible matching number*, Discrete Mathematics, Paper no. 112426.
- [21] MA Henning and ZB Shozi, 2021. *A characterization of the subcubic graphs achieving equality in the Haxell-Scott lower bound for the matching number*, Journal of Graph Theory, **96**, pp. 455–471.

- [22] MA Henning and ZB Shozi, 2022. *A characterization of graphs with given maximum degree and smallest possible matching number: II*, Discrete Mathematics, Paper no. 112731.
- [23] MA Henning and A Yeo, 2007. *Tight lower bounds on the size of a matching in a regular graph*, Graphs and Combinatorics, **23**, pp. 647–657.
- [24] MA Henning and A Yeo, 2008. *Total domination in graphs with given girth*, Graphs and Combinatorics, **24**, pp. 333–348.
- [25] MA Henning and A Yeo, 2018. *Tight lower bounds on the matching number in a graph with given maximum degree*, Journal of Graph Theory, **89(2)**, pp. 115–149.
- [26] VG Kane, SP Mohanty and RS Hales, 1981. *Product graphs and binding numbers*, Ars Combinatoria, **11**, pp. 201–296.
- [27] VG Kane, SP Mohanty and EG Strauss, 1981. *Which rational numbers are binding numbers?*, Journal of Graph Theory, **5**, pp. 379–384.
- [28] D König, 1916. *Über Graphen und ihre Anwendung auf Determinantheorie und Mengenlehre*, Mathematische Annalen, **77**, pp. 453–465.
- [29] D König, 1931. *Graphen und Matrizen*, Math. Riz. Lapok, **38**, pp. 116–119.
- [30] B Korte and J Vygen, 2008. *Combinatorial Optimization: Theory and Algorithms*, Fourth edition, Springer-Verlag, Berlin.
- [31] HW Kuhn, 1955. *The Hungarian method for the assignment problem*, Naval Research Logistics Quarterly, **2(1-2)**, pp. 83–97.
- [32] K Kuratowski, 1930. *Sur le problème des courbes gauches en topologie*, Fundamenta Mathematicae, **15**, pp. 271–283.
- [33] L Lovász, 1975. *Three short proofs in graph theory*, Journal of Combinatorial Theory Series B, **19**, pp. 269–271.
- [34] L Lovász and MD Plummer, 1986. *Matching Theory*, North-Holland, Amsterdam.
- [35] K Menger, 1927. *Zur allgemeinen Kurventheorie*, Fundamenta Mathematicae, **10**, pp. 96–115.
- [36] S O and DB West, 2010. *Balloons, cut-edges, matchings, and total domination in regular graphs of odd degree*, Journal of Graph Theory, **64(2)**, pp. 116–131.
- [37] S O and DB West, 2011. *Matching and edge-connectivity in regular graphs*, European Journal of Combinatorics, **32(2)**, pp. 324–329.
- [38] DW Pentico, 2007. *Assignment problems: A golden anniversary survey*, European Journal of Operational Research, **176(2)**, pp. 774–793.
- [39] P Petersen, 1891. *Die Theorie der regulären Graphen*, Acta Mathematica, **15**, pp. 193–220.
- [40] M Plummer, 2004. *Factors and Factorization*, pp. 403–430 in JL Gross and J Yellen (Eds), *Handbook of Graph Theory*, Chapman & Hall/CRC, New York (NY).

- [41] WR Pulleyblank, 1995. *Matchings and extension*, pp. 179–232 in RL Graham, M Grötschel and L Lovász (Eds), *Handbook of Combinatorics*, Elsevier Science, New York (NY).
- [42] A Schrijver, 2003. *Combinatorial Optimization: Polyhedra and Efficiency*, Springer-Verlag, Berlin.
- [43] C Thomassen, 1980. *Planarity and duality of finite and infinite graphs*, Journal of Combinatorial Theory Series B, **29**, pp. 244–271.
- [44] C Thomassen, 1994. *Every planar graph is 5-choosable*, Journal of Combinatorial Theory Series B, **62**, pp. 180–181.
- [45] WT Tutte, 1954. *A short proof of the factor theorem for finite graphs*, Canadian Journal of Mathematics, **6**, pp. 347–352.
- [46] DF Votaw and A Orden, 1952. *The personnel assignment problem*, pp. 155–163 in *SCOOP*.
- [47] K Wagner, 1937. *Über eine Eigenschaft der ebene Komplexe*, Mathematische Annalen, **114**, pp. 570–590.
- [48] DB West, 1996. *Introduction to Graph Theory*, Prentice-Hall, Upper Saddle River (NJ).
- [49] H Whitney, 1932. *Congruent graphs and the connectivity of graphs*, American Journal of Mathematics, **54**, pp. 150–168.
- [50] WL Winston, 2004. *Operations Research — Applications and Algorithms*, Brooks/Cole Cengage Learning, Belmont (CA), §7.5.
- [51] DR Woodall, 1973. *The binding number of a graph and its Anderson number*, Journal of Combinatorial Theory Series B, **15**, pp. 225–255.



Eulerian graphs

Contents

10.1	Introduction	293
10.2	Finding eulerian circuits and trails	294
10.3	The Chinese postman problem	300
10.4	Other postman problems	306
10.5	Eulerian digraphs	308
10.6	Fleury's algorithm for digraphs	309
	Exercises	311
	Computer exercises	314
	Projects	317
	Further reading	320

10.1 Introduction

During the eighteenth century, the medieval city of Königsberg in Eastern Prussia (now Kaliningrad in modern Russia) contained a central island, called the *Kneiphof*, around which the river Pregolya flowed before dividing into two streams. The four land masses of the city were interconnected by seven bridges, as shown in Figure 10.1(a). According to legend the citizens of Königsberg amused themselves by attempting to find a route crossing each bridge exactly once, and returning to the starting point (blessed were the days before television!). Try as they might, the citizens of Königsberg could find no such route, and they eventually came to believe that such a route was not possible. It was not until the great [Leonard Euler](#) (1707–1783) investigated the problem in 1736, however, that the existence of such a route was *proved* to be impossible.

Although [Euler](#)'s proof was not in the language of graph theory, the ideas in it are essentially graph theoretic in nature. Translated into the language of graph theory, the proof went something like this... Model the four land masses of Königsberg as vertices A , B , C and D , and the seven bridges as edges of a multigraph $G_{10.1}$, as shown in Figure 10.1(b). Suppose, to the contrary, that there exists a circuit T traversing each edge of $G_{10.1}$ exactly once. Then T may be represented by eight letters from the set $\{A, B, C, D\}$ in which each pair of consecutive letters represents an edge being traversed. Since five bridges lead to and from the region A , the letter A must appear at least three times in T — twice to indicate

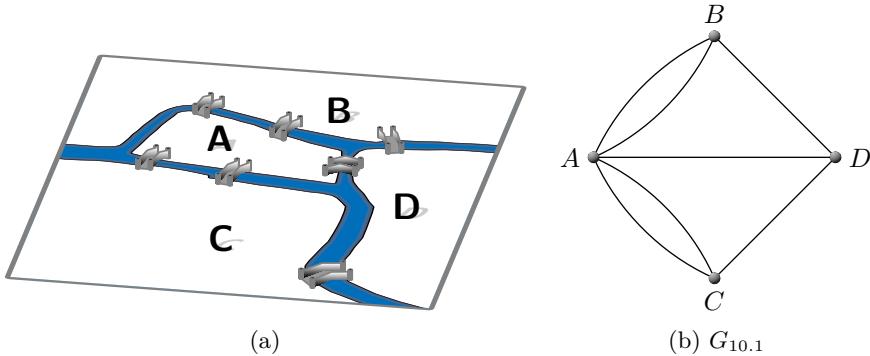


Figure 10.1: The seven bridges of Königsberg.

entries to and exits from the region A via four of the bridges leading to and from it, and once to indicate either an entry to or an exit from the region A via the fifth bridge leading to and from it. By similar arguments, the letters B , C and D must each appear at least twice in T . But this means that T contains at least $3 + 2 + 2 + 2 = 9$ letters — a contradiction. The townsfolk’s suspicion that there was no route through Königsberg crossing each bridge exactly once and returning to the starting point was therefore well-founded.

This legend gives rise to several interesting graph theoretic questions: Given a graph or multigraph G , does there exist a circuit traversing all the edges of G exactly once? If such a circuit exists, how should one go about finding the circuit in a systematic manner? And if no such circuit exists, how can one traverse all the edges of G with the minimum amount of repeated traversals or backtracking? These questions are addressed in this chapter.

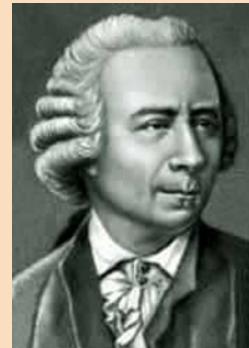
10.2 Finding eulerian circuits and trails

The problem of the bridges of Königsberg is perhaps the oldest and best known of all problems in graph theory. As a result of Euler’s involvement in the problem, graphs which admit circuits traversing all edges exactly once carry his name. In particular, an **eulerian circuit** of a connected (multi)graph G is a closed trail of G that contains all the edges of G (each edge exactly once), whilst an **eulerian trail** of G is an open trail (its start and end vertices are not the same) that contains all the edges of G (again, each edge exactly once). A (multi)graph that contains an eulerian circuit is called an **eulerian (multi)graph**. The graph $G_{10.2}$ in Figure 10.2(a) is an example of an eulerian graph — it contains the eulerian circuit $C : a e b c d e f g h i g j k e g k f j e l a$ — but $G_{10.2}$ contains no eulerian trail. The graph $G_{10.3}$ in Figure 10.2(b), on the other hand, is not an eulerian graph — it contains no eulerian circuit, yet it contains the eulerian trail $T : u v w x v z x y z$.

The following necessary condition for a (multi)graph to be eulerian is due to Euler [9]. Although Euler was aware of the fact that the condition is also sufficient, he did not prove it — the proof of sufficiency is due to Hierholzer [19] and dates back to 1873.

Theorem 10.1 (Euler and Hierholzer) *A connected (multi)graph G is eulerian if and only if the degree of each vertex in G is even.*

Leonhard Euler was born on 15 April 1707 in Basel, Switzerland. He was one of the most eminent mathematicians of the 18th century and is widely regarded to be one of the greatest in history. He was, in fact, considered a mathematician, physicist, astronomer, logician and engineer who made many important contributions in a wide variety of areas, such as infinitesimal calculus, topology, analytic number theory, graph theory, mechanics, fluid dynamics, optics, astronomy and even the theory of music. To Euler we owe much of our modern mathematical terminology and notational conventions, such as the notion of a mathematical function, for example. In 1723, he received a master's degree from the University of Basel based on a thesis in which he compared the philosophies of Descartes and Newton. He followed this up with a doctoral dissertation on the propagation of sound in 1726, also at Basel. After applying unsuccessfully for a professorship at his alma mater, he moved to St Petersburg in 1727. There he collaborated closely with Daniel Bernoulli and took on an additional job as a medic in the Russian Navy. His concerns about the continuing turmoil in Russia prompted Euler to leave St Petersburg in 1741 and take up a position at the Berlin Academy. He lived for twenty five years in Berlin, where he wrote more than 380 scientific papers. He also published the two works for which he is most renowned: A text on functions entitled *Introductio in analysin infinitorum* (1748) and a text on differential calculus entitled *Institutiones calculi differentialis* (1755). Euler accepted an invitation in 1766 to return to the St Petersburg Academy. Toward the end of his life, he suffered from a serious deterioration of his eyesight, but managed to remain scientifically productive as a result of his remarkable memory. He died on 18 September 1783 in St Petersburg, Russia.



Biographic note 26: Leonard Euler (1707–1783)

Proof Suppose G is a connected (multi)graph with an eulerian circuit C . Each time a vertex is encountered as C is traversed there is a contribution of 2 toward the degree of that vertex — once as the vertex is approached and once as one leaves the vertex — even for the initial vertex v (because the initial and final occurrences of v in C each contribute 1 to the degree of v). Since each edge of G is traversed exactly once, the degree of each vertex of G is the sum of a number of twos, and hence even.

For the converse we proceed by (the strong form of) induction over the (multi)graph size m . If $m = 2$, then the (multi)graph consists of two vertices joined by two edges, which clearly has an eulerian circuit. Assume, as induction hypothesis, that every (nontrivial) connected (multi)graph of size less than $m \geq 3$ which has only vertices of even degree has an eulerian circuit, and let G be a connected (multi)graph of size m with vertices of even degree only.

If G has order 2, then the two vertices are joined by an even number of edges, and hence G is eulerian. Assume, therefore, that G has order at least 3. Then G contains a vertex v adjacent to distinct vertices u and w . Let H be the (multi)graph

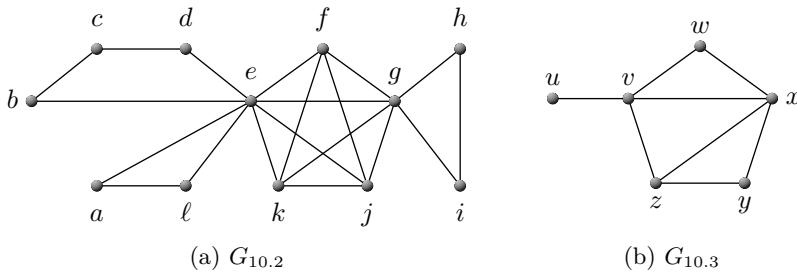
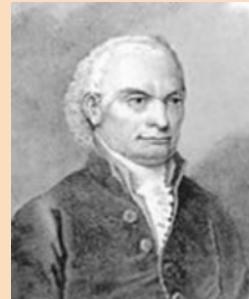


Figure 10.2: Eulerian and noneulerian graphs.

obtained from G by deleting one of each of the edges uv and vw and adding an edge uw . Then H has fewer than m edges and every vertex of H has even degree. If H is connected, then H contains an eulerian circuit C by the induction hypothesis. An eulerian circuit of G may now be formed by replacing the added edge uw in C by the deleted edges uv and vw .

If, however, H is disconnected, then H contains two components, namely a component H_u containing u and a (possibly trivial) component H_v containing v . The component H_u contains an eulerian circuit C_u by the induction hypothesis and, if H_v is nontrivial, it also contains an eulerian circuit C_v . An eulerian circuit of G may now be formed by replacing an edge uw in C_u by the deleted edge uv , then following the eulerian circuit C_v (if it exists) at v , and returning to w along the deleted edge vw . ■

The German mathematician **Carl Hierholzer** was born in Freiburg am Breisgau on 2 October 1840. He studied mathematics in Karlsruhe and obtained his doctorate in mathematics from Ruprecht-Karls-Universität in Heidelberg in 1865. In 1870, Hierholzer completed his *habilitation* on conic sections entitled *Über Kegelschnitte im Raum* in Karlsruhe, where he took up a teaching position. Although the original result that a graph is eulerian if and only if it is connected and every vertex has an even degree (**Theorem 10.1**) is attributed to **Euler**, he merely stated the result without providing a complete proof. Hierholzer is reported to have articulated a full proof of this result just before his premature death in 1871 to a colleague who subsequently arranged for its posthumous publication, which appeared in 1873. Hierholzer died in Karlsruhe on 13 September 1871.



Biographic note 27: Carl Hierholzer (1840–1871)

The following result is a characterisation of those (multi)graphs containing eulerian trails.

Theorem 10.2 *A connected (multi)graph G possesses an eulerian trail if and only if G has exactly two vertices of odd degree. Moreover, the trail begins at one of the vertices of odd degree and terminates at the other.*

Proof Suppose G is a (nontrivial) connected (multi)graph that contains an eulerian trail T . Let u and v be the first and last vertices of T . Form a new (multi)graph H from G by adding an edge e joining the vertices u and v (possibly u and v are joined in G by several edges). Then H is eulerian (it contains the eulerian circuit $T + e$) and hence all vertices of H have even degree by [Theorem 10.1](#). If we recover the (multi)graph G by deleting the edge e from H , then u and v are the only vertices of odd degree.

Conversely, suppose G is a connected (multi)graph with two vertices u and v of odd degree. Form a new (multi)graph H from G by adding an edge e joining the vertices u and v . Then all vertices of H have even degree and hence H contains an eulerian circuit C by [Theorem 10.1](#). Removal of the edge e from C produces an open trail which includes every edge of G . Thus G contains an eulerian trail. Furthermore, this trail begins at u or v and terminates at the other. ■

A collection of trails is said to be **pairwise edge-disjoint** if every two trails in the collection are edge-disjoint, *i.e.* have no edge in common. As a consequence of [Theorem 10.1](#), we have the following useful property of a (multi)graph that will prove useful later.

Lemma 10.3 *If G is a (multi)graph with $2k$ vertices of odd degree for some integer $k \geq 1$, then G contains k pairwise edge-disjoint trails whose $2k$ ends consist of the $2k$ vertices of odd degree.*

Proof Let G_1, \dots, G_r , $r \geq 1$, be the components of G that collectively contain the $2k$ vertices of odd degree of G . Since every such component of G contains an even number of vertices of odd degree by [Theorem 1.1](#), we note that $r \leq k$. Hence it is possible to construct a connected (multi)graph H from the disjoint union of these r components of G by adding k new edges, each edge joining a pair of vertices of odd degree, in such a way that every odd vertex is incident with an added edge. Since every vertex of H has even degree, H has an eulerian circuit by [Theorem 10.1](#). If we write out this circuit, and then omit the k added edges, we obtain the required k pairwise edge-disjoint trails of G whose $2k$ ends consist of the $2k$ vertices of odd degree. ■

❖ The reader should now be able to attempt Exercises [10.1–10.5](#).

Now that we are able to determine exactly which (multi)graphs contain eulerian circuits and which contain eulerian trails via [Theorems 10.1](#) and [10.2](#), respectively, the next question is: “How does one find such a trail or circuit?” [Algorithm 24](#), due to [Fleury](#) [11] and dating back to 1883, may be used to compute eulerian circuits.

We illustrate the working of [Fleury’s algorithm](#) by means of an example, considering the graph $G_{10.2}$ in [Figure 10.2\(a\)](#). Since $G_{10.2}$ is eulerian by [Theorem 10.1](#), we may choose any vertex of $G_{10.2}$ as `current_vertex` for eulerian circuit initialisation purposes in [Step 4](#) of the algorithm. Suppose we choose the vertex a in [Step 4](#) of the algorithm and initialise `current_trail` as the empty trail in [Step 5](#). Since neither of the two edges ae nor al incident with `current_vertex = a` is a bridge of $G_{10.2}$, any of these edges may be selected for insertion into `current_trail`. Suppose we select `current_vertex = e` in [Step 6](#). Then `current_trail = ae` and `other_vertex = e` in [Step 7](#). We consequently delete the edge ae from $G_{10.2}$ in [Step 8](#) to obtain the graph $G'_{10.2}$ in [Figure 10.3\(a\)](#). Because there are still edges in the graph $G'_{10.2}$ we return to [Step 6](#) of the algorithm.

Algorithm 24: Fleury [11]

Input : A (nontrivial) connected (multi)graph G with at most two vertices of odd degree.

Output : An eulerian trail or circuit of G .

```

1 if  $G$  has exactly two vertices of odd degree then
2   | current_vertex  $\leftarrow$  any vertex of odd degree
3 else
4   | current_vertex  $\leftarrow$  any vertex of  $G$ 
5 current_trail  $\leftarrow$  empty sequence of edges
6 Select any edge  $e$  of the form (current_vertex, other_vertex), but
    choosing a bridge only if there is no alternative
7 current_trail  $\leftarrow$  (current_trail,  $e$ ),
    current_vertex  $\leftarrow$  other_vertex
8  $G \leftarrow G$  with  $e$  and all isolated vertices (if there are any) removed
9 if  $E(G) \neq \emptyset$  then go to Step 6
10 print current_trail

```

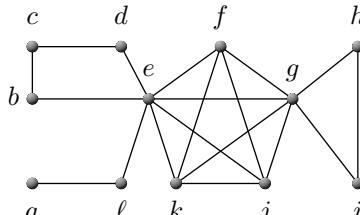
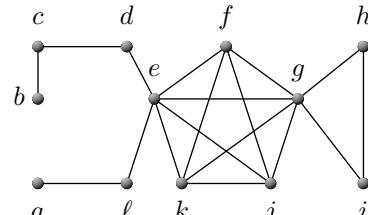
(a) $G'_{10.2}$ (b) $G''_{10.2}$

Figure 10.3: The first two subgraphs formed during the application of Fleury's algorithm to the graph $G_{10.2}$ in Figure 10.2(a).

Now the edge $e\ell$ incident with $\text{current_vertex} = e$ is a bridge of $G'_{10.2}$ and may hence not be selected in Step 6 for insertion into current_trail . Of the other two edges be and de incident with $\text{current_vertex} = e$, we arbitrarily select $\text{other_vertex} = b$ in Step 6. Then $\text{current_trail} = aeb$ and $\text{current_vertex} = b$ in Step 7. We delete the edge be from $G'_{10.2}$ in Step 8 to obtain the graph $G''_{10.2}$ in Figure 10.3(b). Because there are still edges in the graph $G''_{10.2}$ we return yet again to Step 6 of the algorithm. This process continues, and the results obtained at each iteration of the loop spanning Steps 6–9 in Fleury's algorithm are shown in Table 10.1. The algorithm yields the eulerian circuit $C : aebcdedefghigjkækfjelaa$ as encountered at the start of this section.

Let us now prove the correct working of Fleury's algorithm in general, and estimate its worst-case time complexity.

Theorem 10.4 Suppose G is a (nontrivial) connected (multi)graph of size m with at most two vertices of odd degree. Then

- (i) the trail produced by Fleury's algorithm is an eulerian trail of G . Moreover, if G is eulerian, then this trail is an eulerian circuit.
- (ii) the worst-case time complexity of Fleury's algorithm is $\mathcal{O}(m^2)$.

i	Current vertex	Incident bridges	Other vertex	Current trail
0	a	—	—	—
1	a	none	e	$a e$
2	e	el	b	$a e b$
3	b	bc	c	$a e b c$
4	c	cd	d	$a e b c d$
5	d	de	e	$a e b c d e$
6	e	el	f	$a e b c d e f$
7	f	none	g	$a e b c d e f g$
8	g	none	h	$a e b c d e f g h$
9	h	hi	i	$a e b c d e f g h i$
10	i	ig	g	$a e b c d e f g h i g$
11	g	none	j	$a e b c d e f g h i g j$
12	j	none	k	$a e b c d e f g h i g j k$
13	k	none	e	$a e b c d e f g h i g j k e$
14	e	el	g	$a e b c d e f g h i g j k e g$
15	g	gk	k	$a e b c d e f g h i g j k e g k$
16	k	kf	f	$a e b c d e f g h i g j k e g k f$
17	f	fj	j	$a e b c d e f g h i g j k e g k f j$
18	j	je	e	$a e b c d e f g h i g j k e g k f j e$
19	e	el	ℓ	$a e b c d e f g h i g j k e g k f j e l$
20	ℓ	al	a	$a e b c d e f g h i g j k e g k f j e l a$

Table 10.1: Progression of Fleury's algorithm when applied to the graph $G_{10.2}$ in Figure 10.2(a). Here i is a counter measuring the number of iterations of the loop spanning Steps 6–9 in Algorithm 24, while columns 2, 4 and 5 respectively indicate the values of the variables `current_vertex`, `other_vertex` and `current_trail`.

Proof (i) By induction over the size m of the (multi)graph. If $m = 1$, then G consists of two vertices joined by a single edge, in which case Fleury's algorithm clearly produces an eulerian trail of G within a single step. Now suppose that G is a (multi)graph of size $m > 1$ with at most two vertices of odd degree, and assume as induction hypothesis that Fleury's algorithm produces an eulerian trail for every (multi)graph of size $m - 1$ with at most two vertices of odd degree.

If G has only vertices of even degree, then G contains no bridge (otherwise, deleting a bridge from G would leave two subgraph components, each with one vertex of odd degree, which is a contradiction, because a graph cannot have an odd number of vertices of odd degree — see Corollary 1.2). So, after deleting the first edge in Step 8 of Fleury's algorithm, $e = uv$ (say), a connected (multi)graph $G - uv$ of size $m - 1$ with exactly two vertices of odd degree results, *i.e.* the vertices u and v . It therefore follows by the induction hypothesis that Fleury's algorithm produces an eulerian trail of $G - uv$, starting at u and ending at v . By prepending the edge e to this trail, as Fleury's algorithm would have proceeded to do for G , an eulerian circuit of G is returned by the algorithm.

Therefore, suppose G has two vertices, u and v , of odd degree. Then Fleury's algorithm selects either u or v as the current vertex (during Step 2). We may assume that u is chosen. Let $e = uw$ be the edge incident with u selected for inclu-

sion in the trail (during Step 6) of [Fleury's algorithm](#). If $d(u) = 1$, then when the algorithm selects the edge e for inclusion in the trail (during Step 6), the resulting (multi)graph (after the deletions in Step 8) has one (nontrivial) component. If $d(u) > 1$, then let P be a u - v path in G , and let ux be an edge incident with u , but not on P . Since u and v are connected in $G - ux$, the edge ux is not a bridge of G (for otherwise x would be the only vertex of odd degree in its component of $G - ux$, again contradicting [Corollary 1.2](#)). Hence, since at least one edge incident with u is not a bridge, the edge e is not a bridge of G . Thus, the selection of e in Step 6 of [Fleury's algorithm](#) results in a (multi)graph with one (nontrivial) component (after the deletions in Step 8 of the algorithm).

If $w = v$, then all vertices of $G - e$ have even degree. If $w \neq v$, then all vertices of $G - e$, except w and v , have even degree. Since $G - e$ has size $m - 1$, it follows by the induction hypothesis that [Fleury's algorithm](#) produces an eulerian trail T' of $G - e$, starting at w and ending at v . By prepending the edge ux to this trail, as [Fleury's algorithm](#) would have proceeded to do for G , an eulerian trail of G is returned by the algorithm.

(ii) Suppose G is a (multi)graph of size m . During each repetition of the loop spanned by Steps 6–9 of the algorithm it is tested whether an arbitrarily chosen edge incident with `current_vertex` is a bridge; this procedure has a worst-case time complexity of $\mathcal{O}(m)$ — see [Project 5.5](#). If the edge is not a bridge, it may be selected for insertion into `current_trail`. If it is a bridge, then another edge incident with `current_vertex` is inserted into `current_trail`, if such an edge exists. (Note that there cannot be two bridges incident with `current_vertex` — why?). The worst-case time complexity of a single loop spanned by Steps 6–9 of the algorithm is therefore $\mathcal{O}(m)$. Since this loop is repeatedly executed as long as edges remain — *i.e.* a total of m times — the worst-case time complexity of [Fleury's algorithm](#) is $\mathcal{O}(m^2)$. ■

Although [Fleury's algorithm](#) is efficient according to [Theorem 10.4\(ii\)](#) and is a fairly intuitive procedure in view of the characterisations of Theorems 10.1–10.2, it is not the fastest algorithm for computing eulerian trails and circuits. For example, it is possible to devise a depth-first search algorithm for computing eulerian trails and circuits which runs in near linear time (in terms of the number of *vertices* of the (multi)graph!), based on a different characterisation of eulerian (multi)graphs (namely that they are precisely the (multi)graphs whose edge sets may be partitioned into edge-disjoint cycles — see [Exercise 10.5](#)).

❖ The reader should now be able to attempt Exercises [10.6–10.7](#).

10.3 The Chinese postman problem

It is a natural progression from the notion of eulerian graphs to wonder how one may traverse all the edges of a (multi)graph that is not eulerian, incurring the minimum amount of repeated edge traversals or backtracking along edges already traversed. Consider, for example, the application where a postman delivers mail to the homes in the *Tshwane* suburb of *Mountain View*, a map of which is shown in [Figure 10.4](#). Suppose the postman is responsible for delivering mail to the north-western portion of the suburb highlighted in [Figure 10.4](#). A graph modelling this portion of the suburb, in which vertices represent street intersections or dead-ends

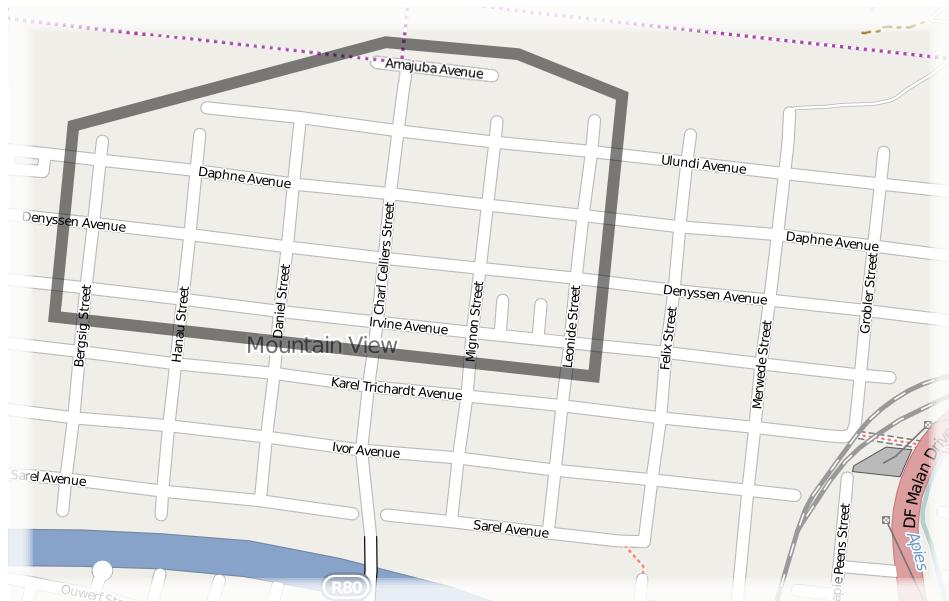


Figure 10.4: A map of the Tshwane suburb of Mountain View.

and in which edges represent the streets themselves, is shown in Figure 10.5. The post office is located at the corner of IRVINE and CHARL CILLIERS streets, *i.e.* at vertex v_{27} in Figure 10.5. The postman has to start out from the post office on bicycle every weekday morning and deliver mail to all the houses in his area of responsibility, returning to the post office thereafter. Naturally he would want to accomplish this task by cycling the shortest distance possible.

The graph in Figure 10.5 is obviously not eulerian — there are many vertices of degree 1 (at street dead-ends) and of degree 3 (at T-junctions). This means that the postman’s delivery route can certainly not be modelled as an eulerian circuit of the graph in Figure 10.5. A certain amount of repeated edge traversals (as a result of backtracking along edges already traversed) is unavoidable, but the postman would like to limit such “needless” traversal distances to a minimum.

The application described above is an instance of the following famous optimisation problem for graphs, called the Chinese¹ postman problem:

Chinese postman problem. Given a weighted connected graph G , find a closed circuit of minimum weight containing every edge of G .

A closed circuit containing every vertex of a weighted connected graph G is called a **Chinese postman tour** of G . A Chinese postman tour is **optimal** if it has minimum weight (*i.e.* if it is a solution to the Chinese postman problem). If the graph G is eulerian, then an optimal Chinese postman tour can, of course, be constructed by means of Fleury’s algorithm (Algorithm 24). If G is not eulerian, however, as is often the case in applications of the Chinese postman problem, such as in the aforementioned mail delivery application related to Figure 10.5, then

¹The prefix “Chinese” in the name of the problem is due to the fact that the formulation of this problem was published in the journal *Chinese Mathematics* by Guan [17] in 1962, who was a Chinese national. Guan was a mathematician at the Shantun Normal College who spent some time working as a post office worker during the Chinese cultural revolution.

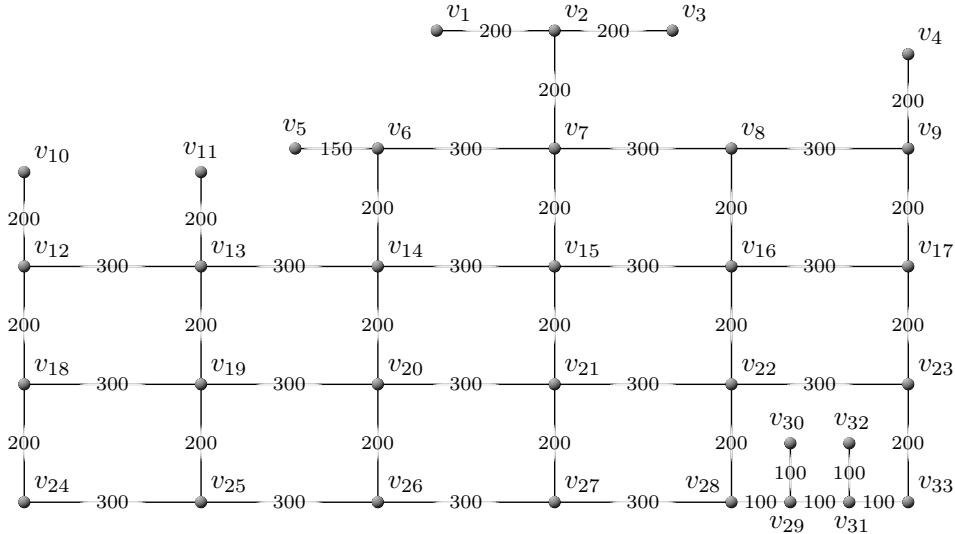


Figure 10.5: A graph model of the north western portion of the street network in the *Tshwane* suburb of *Mountain View*, a map of which may be found in Figure 10.4. The edge weights represent distances (in metres).

a slightly more sophisticated solution procedure is required to find an optimal Chinese postman tour.

In order to describe such a procedure, we require a few notions related to the vertices of odd degree in a (multi)graph. Let G be a connected weighted (multi)graph with $2k$ vertices of odd degree and let V_{odd} denote the set of vertices of odd degree in G . An **odd pair partition** of G is a partition of V_{odd} into k two-element subsets. The **weight** of an odd pair partition $\mathcal{P} = \{\{u_1, v_1\}, \dots, \{u_k, v_k\}\}$ of G is

$$w(\mathcal{P}) = \sum_{i=1}^k d(u_i, v_i),$$

where $d(u_i, v_i)$ denotes the distance between the vertices u_i and v_i in G . The **odd pair partition number** of G , denoted by $\omega_{\text{odd}}(G)$, is the minimum weight of an odd pair partition of G , that is $\omega_{\text{odd}}(G) = \min_{\mathcal{P}} \{w(\mathcal{P})\}$, where the minimum is taken over all odd pair partitions \mathcal{P} of G . Denote the sum of the weights of the edges of G by $w(G)$. Then the following result shows that the weight of an optimal Chinese postman tour in G is intricately linked to the parameter $w(G)$ and the odd pair partition number $\omega_{\text{odd}}(G)$ of G .

Theorem 10.5 *An optimal tour in a weighted connected (multi)graph G has weight $w(G) + \omega_{\text{odd}}(G)$.*

Proof Let G be a weighted connected (multi)graph with $2k$ vertices of odd degree and let V_{odd} denote the set of vertices with odd degree in G .

We first prove that there exists a Chinese postman tour of G of weight exactly $w(G) + \omega_{\text{odd}}(G)$. Let $\mathcal{P} = \{\{u_1, v_1\}, \dots, \{u_k, v_k\}\}$ be an odd pair partition of G of minimum weight, and so $w(\mathcal{P}) = \omega_{\text{odd}}(G)$. Let G^* be obtained from G by duplicating the edges on a shortest u_i - v_i path in G for each $i \in [k]$. Then, G^*

is an eulerian multigraph. Let T^* be an eulerian circuit of G^* . Then, T^* may be transformed into a Chinese postman tour T of G as follows: If there are n copies of an edge in G^* , then replace these n copies of the edge with the original edge (and so, this edge would be traversed n times in T). The resulting Chinese postman tour T of G has weight

$$w(T) = w(T^*) = w(G) + \sum_{i=1}^k d_G(u_i, v_i) = w(G) + w(\mathcal{P}) = w(G) + \omega_{\text{odd}}(G).$$

Hence, there exists a Chinese postman tour of G of weight exactly $w(G) + \omega_{\text{odd}}(G)$, as desired.

We show next that every Chinese postman tour of G has weight at least $w(G) + \omega_{\text{odd}}(G)$. Let T be a Chinese postman tour of G . For each edge e of G , let $\mu(e) + 1$ be the number of times e is traversed in T (where we may have $\mu(e) = 0$).

We now form a new multigraph G^* from G as follows: Let G^* be obtained from G by joining the ends of the edge e by $\mu(e)$ new edges of weight $w(e)$ for each edge e of G for which $\mu(e) > 0$. Now T may be transformed into an eulerian circuit T^* of G^* as follows: If an edge e of G needs to be traversed n times in T , then there are n copies of this edge in G^* , and so we may replace each occurrence of e in T by one of these n edges in such a way that none of them is repeated in T^* . Thus, G^* is an eulerian multigraph.

We next form a multigraph H^* from G^* as follows. By [Theorem 10.1](#), every vertex of G^* has even degree. Therefore, at least one new edge incident with each vertex of odd degree in G must have been added to G in order to produce G^* . Let E^* be the set of edges of G^* that do not belong to G , and so $E^* = E(G^*) \setminus E(G)$. Let $H = G^*[E^*]$ be the multigraph induced by the edges in E^* . (So the edge set of H is E^* , and the vertex set of H consists of those vertices that are incident with edges in E^* .) Necessarily, every vertex of odd degree in G is in H . Observe that for each vertex w of H , the degree of w in H is $d_H(w) = d_{G^*}(w) - d_G(w)$. As observed earlier, $d_{G^*}(w)$ is even. By definition, if w is a vertex of even degree (of odd degree, respectively) in G , then $d_G(w)$ is even (odd, respectively). Hence, $d_H(w)$ is even if w is a vertex of even degree in G , while $d_H(w)$ is odd if w is a vertex of odd degree in G . Thus, the $2k$ odd vertices of G are the only vertices in H of odd degree.

By [Lemma 10.3](#), the multigraph H contains k pairwise edge-disjoint trails whose $2k$ ends consist of the $2k$ odd vertices of H . Let T_1^*, \dots, T_k^* denote these k pairwise edge-disjoint trials in H . By definition of H , these k trails consist solely of new (duplicated) edges of G . For $i \in [k]$, let P_i^* be a shortest path in G that joins the ends of the trail T_i^* . Then, $w(P_i^*) \leq w(T_i^*)$ for all i . The $2k$ ends of the paths P_1^*, \dots, P_k^* , form an odd pair partition of G into k two-element subsets, where the 2-element subsets consist of the ends of the paths P_1^*, \dots, P_k^* , and so $\omega_{\text{odd}}(G) \leq \sum_{i=1}^k w(P_i^*)$. Hence,

$$\begin{aligned} w(T) &= \sum_{e \in E(G)} w(e) + \sum_{e \in E^*} w(e) = w(G) + w(H) \\ &\geq w(G) + \sum_{i=1}^k w(T_i^*) \geq w(G) + \sum_{i=1}^k w(P_i^*) \geq w(G) + \omega_{\text{odd}}(G). \end{aligned}$$

Therefore, $w(T) \geq w(G) + \omega_{\text{odd}}(G)$, and so every Chinese postman tour of G has weight at least $w(G) + \omega_{\text{odd}}(G)$. As observed earlier, there exists a tour of G of weight exactly $w(G) + \omega_{\text{odd}}(G)$. Consequently an optimal tour in a weighted connected multigraph G has weight exactly $w(G) + \omega_{\text{odd}}(G)$. ■

The proof of [Theorem 10.5](#), in fact, suggests the following solution procedure for the Chinese postman problem. Start by duplicating strategically chosen edges in G until an eulerian multigraph H of weight $w(G) + \omega_{\text{odd}}(G)$ is obtained which has G as underlying graph, and then use [Fleury's algorithm \(Algorithm 24\)](#) to find an eulerian circuit of H . It follows from [Theorem 10.5](#) that this circuit of H is an optimal Chinese postman tour of G . We achieve this solution procedure in four simple steps for graphs:

Step 1: Construct a complete, weighted graph $G^* \cong K_{2k}$ of order $2k$ whose vertices correspond to the vertices of odd degree in G and are labelled the same as their counterparts in G . The weight of an edge uv of G^* is taken as the weight of a shortest $u-v$ path in G . These weights may be computed by means of [Floyd's algorithm \(Algorithm 12\)](#) in $\mathcal{O}(n^3)$ time, where n denotes the order of G .

Step 2: Construct a minimum-weight (perfect) matching of G^* . This may be achieved by constructing an intermediate complete graph \widehat{G}^* of order $2k$ whose vertices correspond to and are labelled the same as those of G^* , but where the weight of an edge e of \widehat{G}^* is $m^* - w(e)$, where $w(e)$ is the weight of the corresponding edge e in G^* and $m^* = 1 + \max_e\{w(e)\}$. The computation of m^* and the subsequent construction of \widehat{G}^* may be achieved in $\mathcal{O}(k)$ time. Now any maximum-weight (perfect) matching of \widehat{G}^* is a minimum-weight (perfect) matching of G^* . Therefore, the [maximum-weight matching algorithm](#) of [Chapter 9 \(Algorithm 23\)](#) may be used to compute a maximum-weight (perfect) matching $\{u_1^*U_1^*, \dots, u_k^*U_k^*\}$ of \widehat{G}^* whose weight is as large as possible in $\mathcal{O}(k^4)$ time. Then $\mathcal{P}^* = \{\{u_1^*, U_1^*\}, \dots, \{u_k^*, U_k^*\}\}$ is an odd pair partition of G with associated weight $\omega_{\text{odd}}(G)$.

Step 3: Find a $u_i^*-U_i^*$ path P_i in G of smallest weight by means of [Dijkstra's algorithm \(Algorithm 11\)](#) and duplicate the edges of G that appear on P_i for all $i \in [n]$ to yield an eulerian multigraph H of weight $w(G) + \omega_{\text{odd}}(G)$ with G as underlying graph. Since the worst-case time complexity of [Dijkstra's algorithm](#) is $\mathcal{O}(n^2)$, the multigraph H may be constructed in $\mathcal{O}(n^2k)$ time. Of course this time complexity may be offset by storing in memory the paths of smallest weight computed in **Step 1**.

Step 4: Use [Fleury's algorithm \(Algorithm 24\)](#) to find an eulerian circuit of weight $w(G) + \omega_{\text{odd}}(G)$ for H in $\mathcal{O}(m^2)$ time, where m denotes the size of G . This circuit is an optimal Chinese postman tour by [Theorem 10.5](#).

The most complicated and time-consuming parts of the above procedure are Steps 2 and 4, which have worst-case time complexities of $\mathcal{O}(k^4) = \mathcal{O}(n^4)$ and $\mathcal{O}(m^2) = \mathcal{O}(n^4)$, respectively. Hence a solution to the Chinese postman problem may be computed in $\mathcal{O}(n^4)$ time.

Let us demonstrate the working of the above scheme by means of an example, for the graph $G_{10.4}$ in [Figure 10.6\(a\)](#). The graph is clearly not eulerian, because it has four vertices of odd degree, namely b, c, d and e . We therefore construct a complete, weighted graph $G_{10.4}^* \cong K_4$ of order 4 as shown in [Figure 10.6\(b\)](#). The

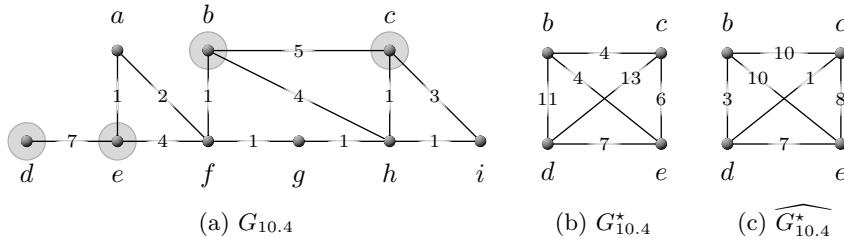


Figure 10.6: The weighted graphs $G_{10.4}$, $G_{10.4}^*$ and $\widehat{G_{10.4}^*}$.

weights of the edges of $G_{10,4}^*$ are those of shortest paths between the odd degree vertices of $G_{10,4}$, determined by means of [Floyd's algorithm](#). These shortest paths are:

- Shortest b - c path in $G_{10:4}$: $b f g h c$ (weight 4)
 - Shortest b - d path in $G_{10:4}$: $b f a e d$ (weight 11)
 - Shortest b - e path in $G_{10:4}$: $b f a e$ (weight 4)
 - Shortest c - d path in $G_{10:4}$: $c h g f a e d$ (weight 13)
 - Shortest c - e path in $G_{10:4}$: $c h g f a e$ (weight 6)
 - Shortest d - e path in $G_{10:4}$: $d e$ (weight 7)

We see from Figure 10.6(b) that $m^* = 1 + 13 = 14$, since the largest edge weight in $\widehat{G_{10.4}^*}$ is 13. This value gives rise to the intermediate complete, weighted graph $\widehat{G_{10.4}^*}$ of order 4 shown in Figure 10.6(c). At this point we would normally have used the maximum-weight matching algorithm of Chapter 9 to compute a (perfect) matching of $\widehat{G_{10.4}^*}$ whose weight is as large as possible, but since the graph $\widehat{G_{10.4}^*}$ is so small, it is easier and faster just to enumerate all the perfect matchings of $\widehat{G_{10.4}^*}$, as shown in Figure 10.7.

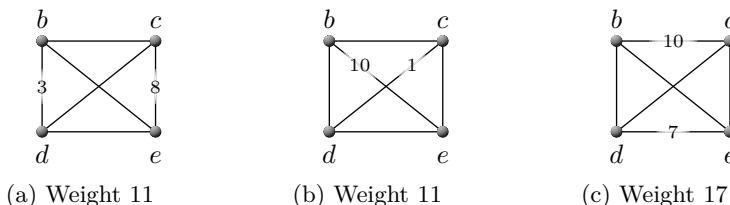


Figure 10.7: The three perfect matchings of $\widehat{G_{10,4}^*}$

The matching $\{bc, de\}$ in Figure 10.7(c) clearly has the largest weight and hence the shortest b - c path $b f g h c$ and the shortest d - e path $d e$ are duplicated in $G_{10.4}$ to form the multigraph $G_{10.5}$ shown in Figure 10.8.

The multigraph $G_{10.5}$ is clearly eulerian. If we choose a as initial vertex, then Fleury's algorithm yields the eulerian circuit $a e d e f b c i h c h b f g h g f a$ of weight 42 for $G_{10.5}$. This circuit is our solution to the Chinese postman problem for the graph $G_{10.4}$ and involves double traversals of the “dead-end” de (this will always be the case, for any graph with “dead-ends”) as well as of the edges bf , fg , qh and hc .

❖ The reader should now be able to attempt Exercise 10.8 and Project 10.1.

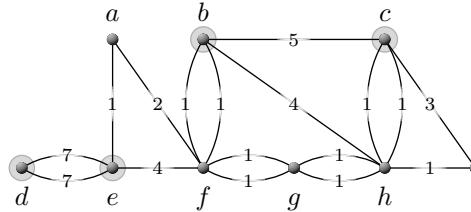


Figure 10.8: A multigraph $G_{10.5}$ which has $G_{10.4}$ as underlying graph.

10.4 Other postman problems

In this section, we present a brief overview of a number of postman problem variants that have been proposed and studied since Guan introduced the original Chinese postman problem in 1962. In each case we suggest suitable references for more details on the problems and how to solve them.

Chinese postman problem: Introduced by Guan [17] in 1962. *Given a weighted connected graph G of order n , find a circuit of minimum weight traversing each edge of G at least once.* May be solved in $\mathcal{O}(n^4)$ time, as explained in Section 10.3. Slightly faster algorithms exist, however. Applications: Urban mail delivery, snow ploughing, street sweeping and street gritting.

Oriented postman problem: Introduced by Edmonds and Johnson [8] in 1973. *Given a weighted connected oriented graph G of order n , find a directed circuit of minimum weight traversing each arc of G at least once.* May be solved as a minimum cost flow problem in polynomial time if both the indegree and outdegree of every vertex in G is at least 1. Applications: Similar to those of the Chinese postman problem, but in a one-way street network. A variation of this problem is the *Mixed postman problem*, where only some edges of G are directed, and others are not.

Rural postman problem: Introduced by Orloff [25] in 1974. *Given a weighted connected graph $G = (V, E)$ of order n , find a circuit of minimum weight traversing each edge in a subset $R \subseteq E$ at least once.* The problem is more complex than the Chinese postman problem due to the possible disconnectedness of $G[R]$; the solution complexity increases exponentially with the number of components of $G[R]$ [12]. Shown to be NP-complete if $R \neq E$ by Lenstra and Rinnooy Kan [21] in 1976. There are heuristics for the problem which run in $\mathcal{O}(n^4)$ time [2]. Applications: Similar to those of the Chinese postman problem, but where not every street necessarily requires mail delivery, snow ploughing, etc.

Windy postman problem: Introduced by Minieka [24] in 1979. *Given a doubly weighted connected graph G of order n , find a circuit of minimum weight traversing each edge of G at least once, where the weight of traversing an edge in one direction may differ from traversing it in the opposite direction.* Shown to be NP-hard by Guan [18] in 1984, who polynomial-time reduced the mixed postman problem (the Chinese postman problem, but where some edges of G are directed) to it. May be solved in polynomial time (with respect to n) in the special cases where all cycles of G are symmetric (*i.e.* if the weight of traversing any cycle in one direction is the same as traversing it in the other) or where G is eulerian (see Win [32]). Applications: Similar to those of the Chi-

nese postman problem, but where the street network is subjected to conditions of wind or steep gradients.

Capacitated postman problem: Introduced by [Golden and Wong \[15\]](#) in 1981.

Given a weighted connected graph $G = (V, E)$ of order n and size m , a collection of N postmen and a real number Q , find N circuits for the N postmen, each visiting a specified vertex (called the depot), such that no circuit has weight more than Q , such that each edge in a specified subset $R \subseteq E$ is traversed at least once by some postman and such that the combined weight of the N circuits is a minimum. If $N = 1$, then the problem reduces to the rural postman problem, and is therefore NP-hard (if $R \neq E$). Various heuristics [3, 4, 6, 14, 15, 27] with worst-case time complexities ranging between $\mathcal{O}(n^3)$ and $\mathcal{O}(mn^4)$ exist for solving the problem approximately. A number of exponential-time exact algorithms also exist [28, 29]. Applications: Similar to those of the Chinese postman problem, but with a fleet of vehicles starting out from (and returning to) a depot, where time and fuel constraints play a role for each vehicle.

Hierarchical postman problem: Probably introduced by [Lemieux and Champagna \[20\]](#) in 1984.

Given a weighted connected graph $G = (V, E)$ of order n and a k -partition E_1, \dots, E_k of the edge set E , find a circuit of minimum weight traversing each edge of G at least once subject to a set of precedence relations specifying that each edge in E_i must be traversed before any of the edges in E_{i+1}, \dots, E_k may be traversed, for all $i \in [k - 1]$. Shown to be NP-hard by [Dror et al. \[7\]](#) in 1987. An $\mathcal{O}(n^5)$ heuristic was also suggested by [Dror et al. \[7\]](#). Applications: Similar to those of the Chinese postman problem, but where certain streets have service priority above others, such as in snow ploughing where main streets have to be cleared first, before secondary roads may be ploughed.

Maximum benefit postman problem: Introduced by [Malandraki and Daskin \[22\]](#) in 1993.

Given a connected graph G where a benefit function is associated with each edge (typically decreasing with multiple traversals of the edge), find a circuit of maximum benefit starting out from and returning to a depot, possibly traversing edges of G more than once, and not necessarily traversing all edges of G . May be solved exactly as a network flow problem in conjunction with a branch-and-bound scheme. Applications: Similar to those of the Chinese postman problem, but where it is beneficial to traverse certain streets more than once (and perhaps others not at all), such as in snow ploughing, where main roads should be ploughed regularly during snow falls, and where it may be beneficial to ignore small secondary roads so as to ensure that larger roads remain clear.

Multiple traversal postman problem: Introduced by [Groves et al. \[16\]](#) in 2005.

Given a connected graph G of order n and size m where the i -th edge is weighted by a triple (a_i, b_i, c_i) , find a circuit of minimum weight traversing the i -th edge at least c_i times, where the weight of traversing it actively is b_i and the weight of traversing it passively (i.e. unnecessarily traversing the edge so as to reach an edge to be traversed actively) is a_i , in such that a manner that multiple active traversals of the same edge is spread out as much as possible in the tour, for all edges. If $a_i = b_i$ for all edges, and if $c_i = 0$ for some edges and $c_i = 1$ for the remaining edges, the problem reduces to the rural postman problem, and is therefore NP-hard. An $\mathcal{O}(n^3)$ heuristic and an $\mathcal{O}(m^3)$ improvement

metaheuristic exists for solving the problem approximately [16]. Applications: Similar to those of the Chinese postman problem, but where the number traversals of each edge during some time window is governed by law, such as with the servicing of a railway grid by a single high-tech service engine, where it may, for example, be required that main lines should be serviced four times annually, secondary lines three times annually and metro lines twice annually.

10.5 Eulerian digraphs

There are many applications in which the methods of the previous sections require slight modifications so as to apply to digraphs, since some streets in a town or city may be one-way, for example. The notions of an eulerian trail or circuit introduced for (multi)graphs in [Section 10.1](#) may be defined analogously for (multi)digraphs. In particular, an **eulerian circuit** of a strongly connected (multi)digraph D is a (directed) circuit of D that contains all arcs of D (each exactly once), whilst an **eulerian trail** of D is an open (directed) trail (its start and end vertices are not the same) that contains all arcs of D (each exactly once). A (multi)digraph that contains an eulerian circuit is called an **eulerian (multi)digraph**. The digraph $D_{10.6}$ in [Figure 10.9\(a\)](#) is an example of an eulerian digraph — it admits the eulerian circuit $C : abchbdahdefgdca$ — but $D_{10.6}$ contains no eulerian trail. The digraph $D_{10.7}$ in [Figure 10.9\(b\)](#), on the other hand, is not an eulerian digraph — it contains no eulerian circuit, yet it contains the eulerian trail $T : bacbdcefgdheghfde$.

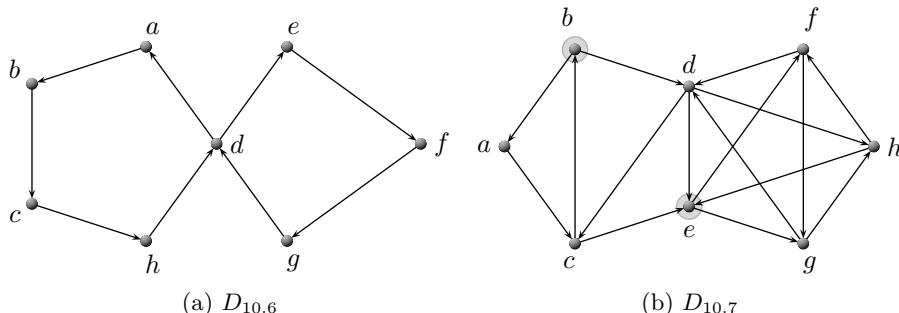


Figure 10.9: Eulerian and noneulerian digraphs.

The following necessary condition for a (multi)digraph to be eulerian is an obvious generalisation of [Theorem 10.1](#).

Theorem 10.6 *A strongly connected (multi)digraph D is eulerian if and only if the indegree and the outdegree are equal for each vertex of D .*

It is also possible to generalise the result of [Theorem 10.2](#) so as to obtain a characterisation of those (multi)digraphs that contain eulerian trails.

Theorem 10.7 *A strongly connected, nontrivial (multi)digraph D possesses an eulerian trail if and only if D has two vertices u and v such that*

$$\text{od}(u) = \text{id}(u) + 1 \quad \text{and} \quad \text{id}(v) = \text{od}(v) + 1$$

and $\text{od}(w) = \text{id}(w)$ for all other vertices w of D . Moreover, the trail begins at u and terminates at v .

The proofs of Theorems 10.6 and 10.7 are similar to those of Theorems 10.1 and 10.2, respectively, and are left as exercises.

- ❖ The reader should now be able to attempt Exercises 10.9–10.12.

10.6 Fleury's algorithm for digraphs

Upon application of Theorem 10.7 to the digraph $D_{10.7}$ in Figure 10.9(b) it becomes clear why $D_{10.7}$ contains an eulerian trail, and why it starts at b and ends at e , when one notes the in- and outdegrees of the vertices of $D_{10.6}$, as listed in Table 10.2.

Vertex →	a	b	c	d	e	f	g	h
Indegree	1	1	2	3	3	2	2	2
Outdegree	1	2	2	3	2	2	2	2

Table 10.2: Indegrees and outdegrees of the vertices of the digraph $D_{10.7}$ in Figure 10.9(b).

The notion of not choosing a bridge in Step 6 of Fleury's algorithm until absolutely necessary does not naturally carry over to a digraph setting. Hence we present a modified version of Fleury's algorithm for finding eulerian circuits in eulerian digraphs as in Algorithm 25.

We illustrate the working of the above algorithm by means of an example, namely for the eulerian digraph $D_{10.6}$ in Figure 10.9(a). Suppose we choose the vertex $v = a$ in Step 1 of the algorithm. The reversed digraph $D'_{10.6}$ obtained in Step 2 is shown in Figure 10.10(a). The depth-first search tree $T'_{10.8}$ of $D'_{10.6}$ found in Step 3 is shown in Figure 10.10(b) and illustrated in bold in Figure 10.10(a) — see Project 5.4 for a description of how a depth-first search tree is constructed. The reversed, directed tree $T_{10.8}$ obtained in Step 4 of the algorithm is shown in bold in Figure 10.10(c). The exit order decided upon in Step 5 of the algorithm is shown in Figure 10.10(d).

Steps 6–11 of Algorithm 25 are similar to Fleury's original algorithm, and are straight forward. The algorithm produces the eulerian circuit $abda\overline{h}defgdchbca$ for the digraph $D_{10.6}$ in Figure 10.9(a).

Let us now prove the correct working of Algorithm 25 for digraphs in general, and estimate its worst-case time complexity.

Theorem 10.8 *Suppose D is a digraph which has a (nontrivial) connected graph G of size m as underlying graph and suppose that the indegree of each vertex of D equals its outdegree. Then*

- (i) *the trail produced by Algorithm 25 is an eulerian trail of D , and*
- (ii) *the worst-case time complexity of Algorithm 25 is $\mathcal{O}(m^2)$.*

Proof (i) We first prove that the graph underlying the directed depth-first search tree produced in Step 3 of Algorithm 25 is, in fact, a *spanning* tree of G . Suppose

Algorithm 25: Finding an eulerian circuit in a digraph

Input : A (nontrivial) strongly connected digraph D for which the indegree and outdegree of every vertex are equal.

Output : An eulerian circuit of D .

- 1 Choose any vertex v of D
- 2 Let D' be the digraph obtained by reversing each arc of D
- 3 Apply a depth-first search to D' in order to construct a directed tree T' consisting of paths from v to all other vertices
- 4 Let T be the directed tree obtained by reversing each arc of T' (then T contains a $u-v$ path in D for every vertex u of D)
- 5 Specify, for each vertex u of D , an arbitrary ordering of the arcs exiting u , except that for $u \neq v$ the arc exiting u in T must be ordered last
- 6 Let `current_vertex` be v and set `current_trail` equal to the empty sequence of arcs
- 7 Select the next remaining arc a of the form
`{current_vertex, other_vertex}` in the ordering of arcs exiting `current_vertex`
- 8 Append a to `current_trail` and set `current_vertex` equal to `other_vertex`
- 9 Delete a and all isolated vertices (if there are any) from D and call the resulting digraph D again
- 10 if arcs remain in D then go to Step 7
- 11 print `current_trail`

this is not the case. Then the depth-first search procedure does not reach a new vertex at each of its iterations, which means that at a certain iteration all arcs between the vertex set R reached at that point and the vertices not in R are directed towards vertices in R . Since each edge of $D[R]$ contributes once to the indegree and once to the outdegree of vertices in R , we therefore have that the sum of the indegrees of vertices in R exceeds the sum of the outdegrees of vertices in R . But this contradicts our supposition that the indegree of each vertex of D equals its outdegree.

It therefore follows that the trail produced by the loop spanning Steps 7–10 of [Algorithm 25](#) terminates at v , because when the `current_trail` reaches a vertex $u \neq v$, the arc leaving u in T remains outside `current_trail`. Hence the trail produced by [Algorithm 25](#) is closed.

Furthermore, this trail can only terminate at v once all the arcs leaving v are contained within `current_trail`. Since an arc in T cannot be included in `current_trail` until it is the only arc outside `current_trail` leaving some vertex of D , it follows that all arcs entering v cannot be included in `current_trail` until all arcs entering and leaving other vertices of D have been included in `current_trail`, because T contains a directed path from each vertex of D to v (recall that the graph underlying T is a spanning tree of G). We conclude that `current_trail` contains all arcs of D (and hence is eulerian) at termination of [Algorithm 25](#).

(ii) This part of the proof is left as an exercise. ■

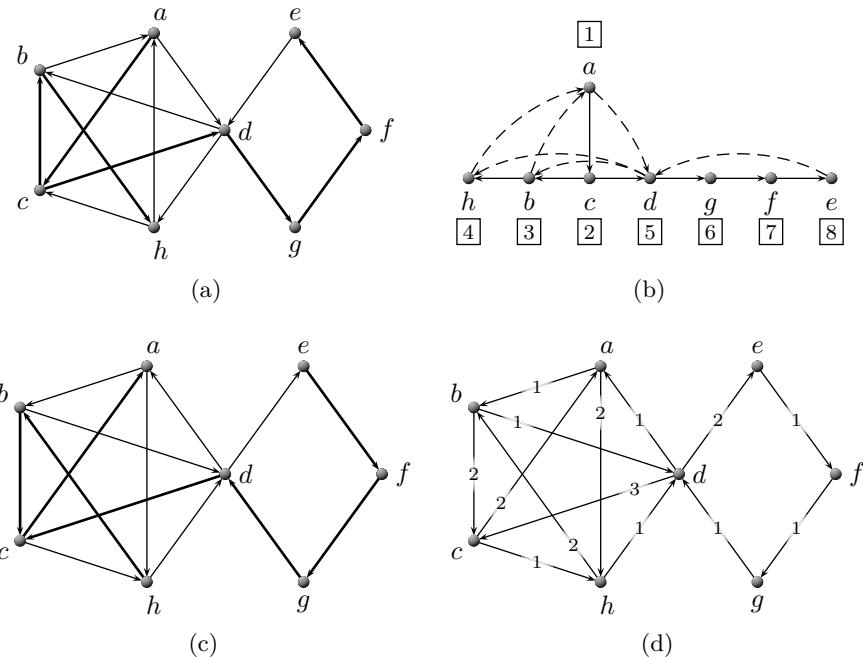


Figure 10.10: Working of the adaptation of Fleury's algorithm for digraphs (Algorithm 25), when applied to the digraph $D_{10.6}$ in Figure 10.9(a).

- The reader should now be able to attempt Exercises 10.13–10.14 and Projects 10.2–10.3.

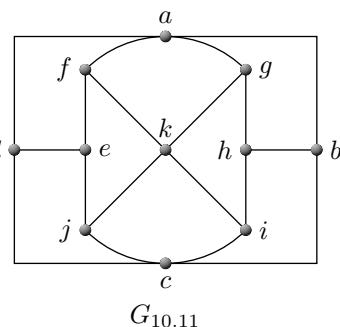
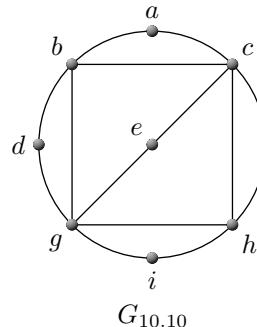
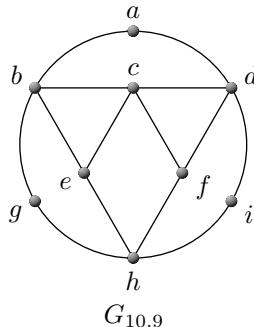
Exercises

- 10.1 In present day Königsberg (now called Kaliningrad) there are two additional bridges, one between regions B and C , and one between regions B and D . Is it now possible to devise a route traversing all the bridges (not necessarily ending where one started out)?

10.2 (a) Show where the inhabitants of Königsberg could have built two additional bridges, so that a route traversing all the bridges (and returning to the starting point) exists.
(b) Could the same effect have been achieved by building only one additional bridge? If so, where? If not, why not?

10.3 For which values of n are the following graphs eulerian?
(a) C_n ;
(b) K_n ; and
(c) $K_{n,n}$.

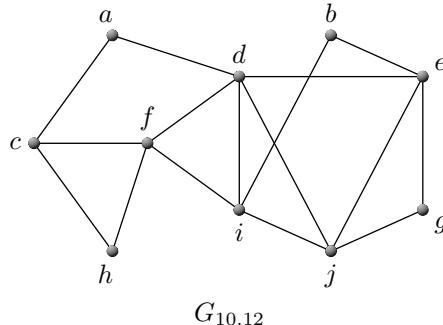
10.4 Decide which of the following graphs are eulerian:



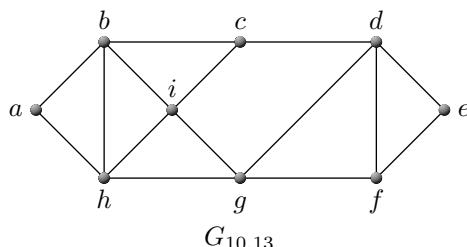
Motivate your answer.

10.5 Prove that a connected (multi)graph is eulerian if and only if its edge set may be partitioned into edge-disjoint cycles, where 2-cycles are allowed. (Hint: Use induction over the number of cycles.)

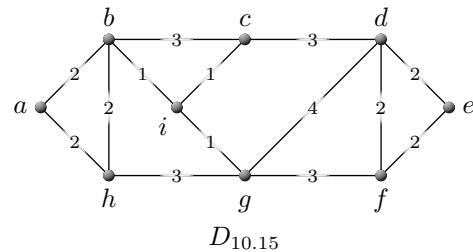
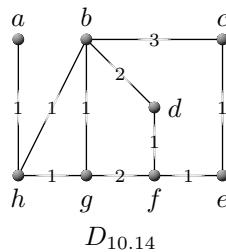
10.6 Use [Fleury's algorithm](#) to find an eulerian circuit for the graph $G_{10.12}$ below. Produce a table such as [Table 10.1](#) to summarise the output of the algorithm during each step of its progress.



10.7 Use [Fleury's algorithm](#) to find an eulerian trail for the graph $G_{10.13}$ below. Produce a table such as [Table 10.1](#) to summarise the output of the algorithm during each step of its progress.



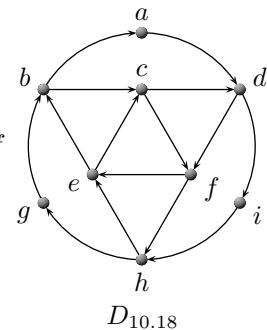
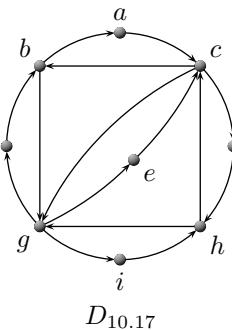
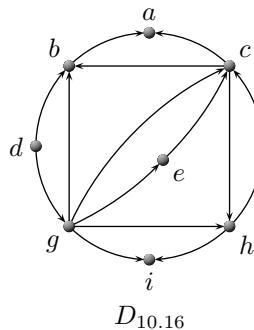
10.8 Solve the Chinese postman problem for each of the following weighted connected graphs:



10.9 Prove [Theorem 10.6](#).

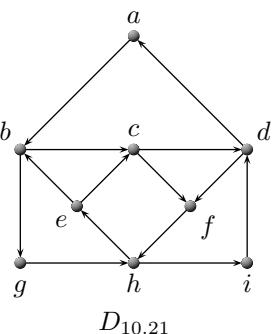
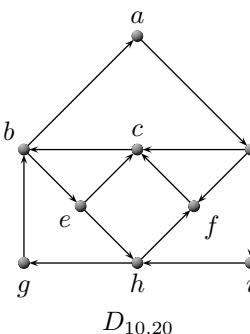
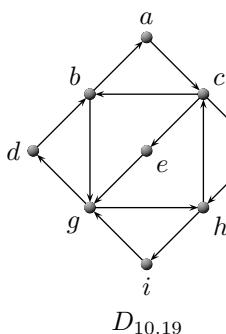
10.10 Prove [Theorem 10.7](#).

10.11 Decide which of the following digraphs are eulerian:



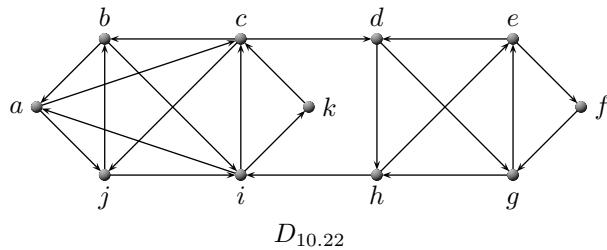
Motivate your answer.

10.12 Decide which of the following digraphs admit an eulerian trail:



Motivate your answer.

- 10.13 Use the modification of Fleury's algorithm (Algorithm 25) to find an eulerian circuit for the digraph $D_{10.22}$ below.



- 10.14 Prove Theorem 10.8(ii).

Computer exercises

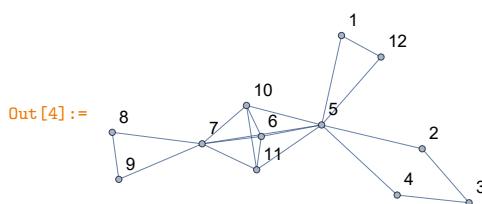
Recall, from Chapter 1, that a graph may be represented on a computer by its adjacency lists. Up to this point we have not demonstrated how this may be done in MATHEMATICA. While MATHEMATICA has no built-in command for converting specified adjacency lists into graphs, the following function, called `AdjList2SymMat[L]`, takes a set L of adjacency lists and produces a (symmetric) adjacency matrix of the corresponding graph as output. It is then easy to generate the graph itself from this adjacency matrix. The code

```
In[1]:= (* function to convert adjacency list to symmetric adj matrix: *)
AdjList2SymMat[adj_] := Module[{i, j, m, n},
  n = Length[adj];
  m = Table[0, {n}, {n}];
  For[i = 1, i <= n, i++,
    For[j = 1, j <= Length[adj[[i]]], j++,
      m[[i, adj[[i, j]]]] = 1;
      m[[adj[[i, j]], i]] = 1
    ]
  ];
  m
]
```

produces no output on its own; it merely loads this newly defined command into memory. Execution of the subsequent commands

```
In[2]:= adjlist = {{5, 12}, {3, 5}, {2, 4}, {3, 5}, {1, 6, 7, 10, 11, 12}, {5, 7, 10, 11}, {5, 6, 8, 9, 10, 11}, {7, 9}, {7, 8}, {5, 6, 7, 11}, {5, 6, 7, 10}, {1, 5}};
In[3]:= m = AdjList2SymMat[adjlist];
In[4]:= G = AdjacencyGraph[m, VertexLabels -> "Name"]
```

produces the following graph as output:

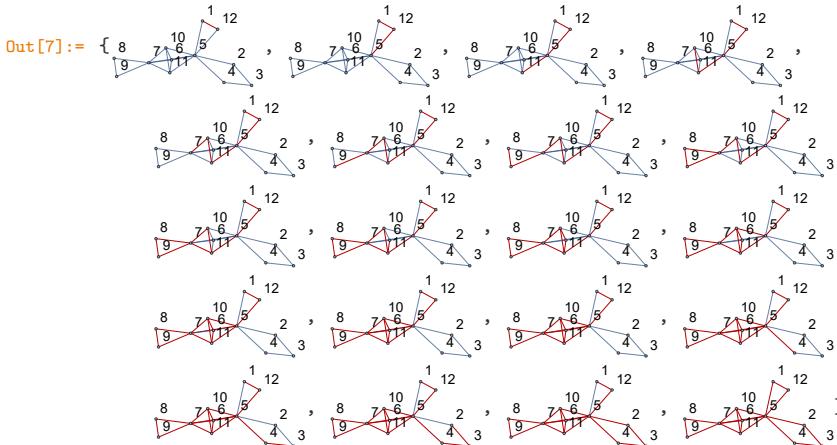


The **MATHEMATICA** command **EulerianGraphQ**[G] returns the boolean value **True** if a specified graph G is eulerian, or the boolean value **False** otherwise. The command **FindEulerianCycle**[G] returns an eulerian circuit of G if G is indeed eulerian, or otherwise it returns an empty circuit. An eulerian circuit may be highlighted in G in a step-by-step fashion by combining the commands **Table** and **HighlightGraph**. Bearing in mind that a percentage sign in **MATHEMATICA** represents the output of the last command executed, the commands

```
In[5]:= EulerianGraphQ[G]
In[6]:= FindEulerianCycle[G]
In[7]:= Table[HighlightGraph[G, Part[First[%], 1; ; i]], {i, Length[First[%]]}]
```

produce the output:

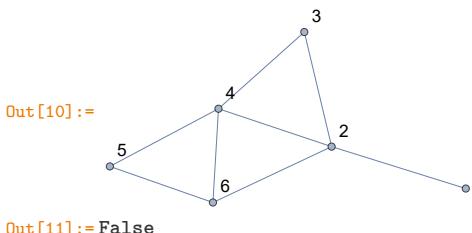
```
Out[5]:= True
Out[6]:= {{1 → 12, 12 → 5, 5 → 11, 11 → 10, 10 → 7, 7 → 9, 9 → 8, 8 → 7, 7 → 11, 11 → 6, 6 → 10, 10 → 5, 5 → 7, 7 → 6, 6 → 5, 5 → 4, 4 → 3, 3 → 2, 2 → 5, 5 → 1}, }
```



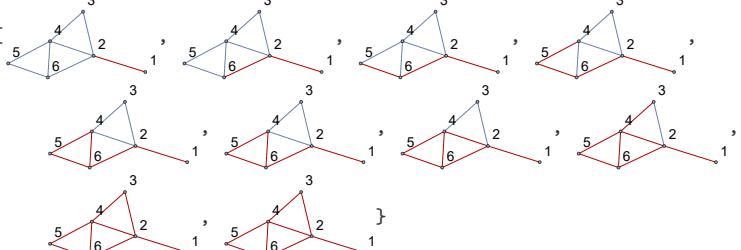
The command **FindPostmanTour**[G] returns an optimal solution to the Chinese postman problem for a specified connected, weighted graph G that need not be eulerian. For example, the commands

```
In[8]:= adjlist = {{2}, {3, 4, 6}, {2, 4}, {2, 3, 5, 6}, {4, 6}, {2, 4, 5}};
In[9]:= m = AdjList2SymMat[adjlist];
In[10]:= H = AdjacencyGraph[m, VertexLabels -> "Name"];
In[11]:= EulerianGraphQ[H]
In[12]:= FindEulerianCycle[H]
In[13]:= FindPostmanTour[H]
In[14]:= Table[HighlightGraph[H, Part[First[%], 1; ; i]], {i, Length[First[%]]}]
```

produce the output:



```

Out[12]:= {}
Out[13]:= {{1 → 2, 2 → 6, 6 → 5, 5 → 4, 4 → 6, 6 → 2, 2 → 4, 4 → 3, 3 → 2, 2 → 1}}
Out[14]:= {


```

The following new command, called `AdjList2Mat`, is the equivalent of the command `AdjList2SymMat` defined above, but for directed graphs (hence not necessarily producing a *symmetric* adjacency matrix):

```

In[15]:= AdjList2Mat[adj_] := Module[{i, j, m, n},
  n = Length[adj];
  m = Table[0, {n}, {n}];
  For[i = 1, i <= n, i++,
    For[j = 1, j <= Length[adj[[i]]], j++,
      m[[i, adj[[i, j]]]] = 1;
    ]
  ];
  m
]

```

All the commands illustrated above for eulerian graphs may also be applied to digraphs, as illustrated by the code

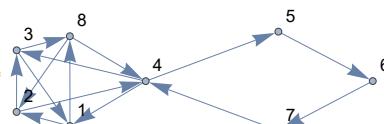
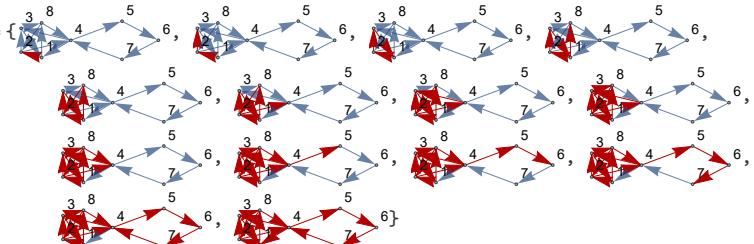
```

In[16]:= adjlist = {{2, 8}, {3, 4}, {1, 8}, {1, 3, 5}, {6}, {7}, {4}, {2, 4}};
In[17]:= m = AdjList2Mat[adjlist];
In[18]:= J = AdjacencyGraph[m, VertexLabels -> "Name"]
In[19]:= EulerianGraphQ[J]
In[20]:= FindEulerianCycle[J]
In[21]:= Table[HighlightGraph[J, Part[First[%], 1; ; i]], {i, Length[First[%]]}]

```

which produces the output:

```

Out[18]:= 
Out[19]:= True
Out[20]:= {{1 → 2, 2 → 3, 3 → 4, 4 → 5, 5 → 6, 6 → 7, 7 → 4, 4 → 1}, {1 → 8, 8 → 2, 2 → 6, 6 → 5, 5 → 4, 4 → 3, 3 → 8, 8 → 4, 4 → 5, 5 → 1}}
Out[21]:= {


```

❖ The reader should now be able to attempt Exercises 10.4, 10.6, 10.7, 10.9–10.11.

Projects

This section contains three projects. In the first project, the reader is required to solve the Chinese postman problem instance corresponding to the mail delivery application of [Figure 10.5](#). The second project is devoted to the problem of counting the number of distinct eulerian circuits in an eulerian digraph, while the last project is also related to the enumeration of eulerian circuits, but this time in pseudo-digraphs that are closely related to an application in coding theory, generated by the well-known notion of a De Bruijn sequence.

Project 10.1: Chinese postman problem

Consider the street map of the *Tshwane* suburb of *Mountain View* adjacent to the *Magaliesberg Nature Reserve*, shown in [Figure 10.4](#) with accompanying graph representation of its north western portion (the area of responsibility of one of the suburb's postmen) shown in [Figure 10.5](#).

Task

Solve the Chinese postman problem for the graph in [Figure 10.5](#) so as to provide the postman with an optimal tour for mail delivery in his area of responsibility. Assume that all streets allow for two-way traffic and start construction of the tour (eulerian circuit) at the post office, which is vertex v_{27} (corner of IRVINE and CHARL CILLIERS streets). Show each step of your working and reasoning clearly.

Project 10.2: Counting eulerian circuits in digraphs

In this project we are interested in counting the number of distinct eulerian circuits in an eulerian digraph. In order to solve this enumeration problem we return to the modification of [Fleury's algorithm](#) for digraphs introduced in [Section 10.5](#) ([Algorithm 25](#)).

Tasks

1. Prove that for every tree T in [Step 4](#) of the modification of [Fleury's algorithm](#) for digraphs, any one of

$$\prod_{i=1}^n (\text{id}(v_i) - 1)!$$

distinct eulerian circuits may be generated by the algorithm for a digraph of order n with vertex set $\{v_1, \dots, v_n\}$.

Suppose \mathbf{A} is the adjacency matrix of an eulerian digraph D with vertex set $\{v_1, \dots, v_n\}$ and let \mathbf{A}^* be the matrix obtained by replacing the entry on the i -th diagonal of $-\mathbf{A}$ by $\text{od}(v_i)$. For example, for the eulerian digraph $D_{10.23}$ in [Figure 10.11\(a\)](#) these matrices are respectively

$$\mathbf{A}(D_{10.23}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{A}^*(D_{10.23}) = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

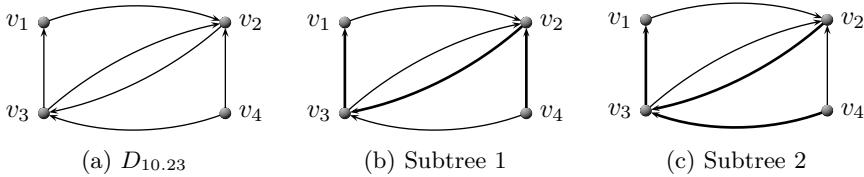


Figure 10.11: The digraph $D_{10.23}$ and its $\Xi(D_{10.23}) = 2$ subtrees whose branches are paths in $D_{10.23}$ to the vertex v_1 .

In 1948, Tutte [30] proved a result in a slightly more general setting from which we may deduce that the number $\Xi(D)$ of distinct subtrees of a digraph D whose branches are paths in D to some vertex of D is any cofactor of the matrix \mathbf{A}^* . Recall that the (i, j) -th cofactor of \mathbf{A}^* is $(-1)^{i+j}|\mathbf{A}_{ij}^*|$, where \mathbf{A}_{ij}^* is the matrix obtained by deleting row i and column j from \mathbf{A}^* . For example, the $(1, 1)$ -th cofactor of $\mathbf{A}^*(D_{10.23})$ is

$$(-1)^2 \begin{vmatrix} 2 & -1 & -1 \\ -1 & 2 & 0 \\ 0 & -1 & 1 \end{vmatrix} = 2.$$

The other fifteen cofactors of $\mathbf{A}^*(D_{10.23})$ also all have a value of 2. The $\Xi(D_{10.23}) = 2$ distinct subtrees of the digraph $D_{10.23}$ whose branches are paths in $D_{10.23}$ to the vertex v_1 are shown in Figure 10.11(b)–(c).

2. Prove that the number of eulerian circuits in a digraph D of order n with vertex set $\{v_1, \dots, v_n\}$ is

$$\Xi(D) \prod_{i=1}^n (\text{id}(v_i) - 1)!$$

(Hint: Show that the set of all eulerian circuits that may be generated by the modification of [Fleury's algorithm](#) for a digraph D consists of *all* the eulerian circuits of D .)

Project 10.3: De Bruijn sequences

Consider an alphabet $\mathcal{A} = \{0, 1, \dots, b-1\}$ with $b \geq 2$ and define a **word** of length n over \mathcal{A} to be an ordered n -tuple of letters from \mathcal{A} . Then a **De Bruijn sequence** $d_0 d_1 \dots d_{N-1}$ is a sequence of N elements from \mathcal{A} such that, for every word \mathbf{w} of length n over \mathcal{A} , there exists a *unique* index i such that $\mathbf{w} = d_i d_{i+1} \dots d_{i+n-1}$, where the subscripts are taken modulo N .

Suppose, for example, $\mathcal{A}_2 = \{0, 1, 2\}$ and $n = 2$. Then 001122021 is a De Bruijn sequence producing all 3^2 words of length $n = 2$ over \mathcal{A}_2 as one slides a moving window of length $n = 2$ digits across the sequence (modulo $N = 9$). The words thus produced are 00, 01, 11, 12, 22, 20, 02, 21 and 10 (in that order).

Tasks

1. Prove that b^n is an upper bound on the length N of a De Bruijn sequence for words of length n over an alphabet $\mathcal{A} = \{0, 1, \dots, b-1\}$ with $b \geq 2$.

2. Prove that b^n is also a lower bound on the length N of a De Bruijn sequence for words of length n over an alphabet $\mathcal{A} = \{0, 1, \dots, b-1\}$ with $b \geq 2$.

From the above results we conclude that $N = b^n$. Let us now consider how eulerian pseudodigraphs may be used to construct De Bruijn sequences. Consider a **De Bruijn pseudodigraph** $D_{b,n}$ of order b^{n-1} whose vertex set corresponds to the set of all b^{n-1} words of length $n-1$ over \mathcal{A} and whose arc set corresponds to the set of all b^n words of length n over \mathcal{A} . In particular, the arc corresponding to the word $d_1d_2 \cdots d_n$ is the ordered pair $(d_1d_2 \cdots d_{n-1}, d_2d_3 \cdots d_n)$. The De Bruijn pseudodigraphs $D_{2,3}$, $D_{2,4}$ and $D_{3,2}$ are shown in Figure 10.12.

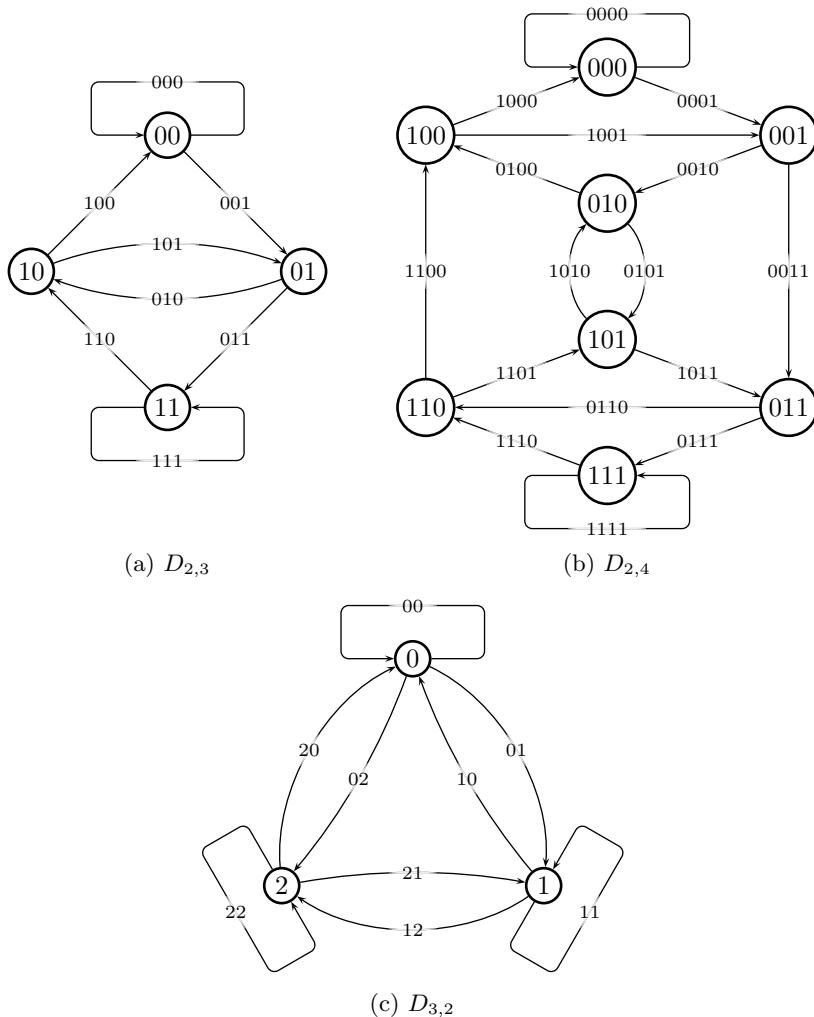


Figure 10.12: The De Bruijn pseudodigraphs $D_{2,3}$, $D_{2,4}$ and $D_{3,2}$.

3. Draw the De Bruijn pseudodigraph $D_{4,2}$.
4. Prove that every De Bruijn pseudodigraph is eulerian.

5. Show how an eulerian circuit of the De Bruijn pseudodigraph $D_{b,n}$ may be used to construct a De Bruijn sequence for words of length n over an alphabet $\mathcal{A} = \{0, 1, \dots, b - 1\}$ with $b \geq 2$.

Note that if $b \geq 2$ and $n = 1$, then there are $(b - 1)!$ possible De Bruijn sequences, because there are $(b - 1)!$ distinct circular arrangements of b distinct objects. Therefore a De Bruijn sequence for words of length 1 over an alphabet $\mathcal{A} = \{0, 1, \dots, b - 1\}$ is not unique, unless $b = 2$. The question arises as to how many De Bruijn sequences exist for general values of $b \geq 2$ and $n \in \mathbb{N}$. In order to answer this question, recall from [Project 10.2](#) that the number of eulerian circuits in a digraph D of order n with vertex set $\{v_1, \dots, v_n\}$ is

$$\Xi(D) \prod_{i=1}^n (\text{id}(v_i) - 1)!$$

6. Prove that the number of eulerian circuits in a pseudodigraph D' obtained by inserting loops at the vertices v_{j_1}, \dots, v_{j_k} of an eulerian digraph D of order n with vertex set $\{v_1, \dots, v_n\}$ is

$$\zeta(D') = \Xi(D) \prod_{\ell=1}^k \text{od}_D(v_{j_\ell}) \prod_{i=1}^n (\text{id}_D(v_i) - 1)!$$

We conclude that the number of distinct De Bruijn sequences for words of length n over an alphabet $\mathcal{A} = \{0, 1, \dots, b - 1\}$ with $b \geq 2$ is given by the quantity $\zeta(D_{b,n})$.

7. How many De Bruijn sequences for words of length 2 over the alphabet $\mathcal{A} = \{0, 1, 2\}$ are there? Find them all.

Further reading

- [1] R Balakrishnan and K Ranganathan, 2000. *A Textbook of Graph Theory*, Second edition, Springer, New York (NY), pp. 117–142.
- [2] MO Ball, TL Magnanti, CL Monma and GL Nemhauser, 1995. *Network Routing*, Volume 8, Elsevier, Amsterdam.
- [3] E Benavent, V Campos, A Corberan and E Mota, 1990. *The capacitated arc routing problem — A heuristic algorithm*, Quaderns d’Estadística i Investigació Operativa (Quèstiió), **14(1–3)**, pp. 107–122.
- [4] L Chapleau, JA Ferland, G Lapalme and J-M Rousseau, 1984. *A parallel insert method for the capacitated arc routing problem*, Operations Research Letters, **3(2)**, pp. 95–99.
- [5] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY), Chapter 7.
- [6] N Christofides, 1973. *The optimum traversal of a graph*, Omega, **1(6)**, pp. 719–732.

- [7] M Dror, H Stern and P Trudeau, 1987. *Postman tour on a graph with precedence relation on arcs*, Networks, **17(3)**, pp. 283–294.
- [8] J Edmonds and E Johnson, 1973. *Matching, Euler tours and the Chinese postman problem*, Mathematical Programming, **5(1)**, pp. 88–124.
- [9] L Euler, 1741. *Solutio problematis ad geometriam situs pertinentis*, Commentarii Academiae Scientiarum Petropolitanae, **8**, pp. 128–140.
- [10] JR Evans and E Minieka, 1992. *Optimization for Networks and Graphs*, Marcel Dekker, New York (NY), Chapter 8.
- [11] M Fleury, 1883. *Deux problèmes de géométrie de situation*, Journal de Mathématiques Élémentaires, pp. 257–261.
- [12] GN Frederickson, 1979. *Approximation algorithms for some postman problems*, Journal of the Association of Computing Machinery, **26(3)**, pp. 538–554.
- [13] M Gendreau, G Laporte and Y Zhao, *The windy postman problem on general graphs*, Technical Report, Montreal, 1990.
- [14] BL Golden, JS DeArmon and EK Baker, 1983. *Computational experiments with algorithms for a class of routing problems*, Computers and Operations Research and their Application to Problems of World Concern, **10(1)**, pp. 47–59.
- [15] BL Golden and RT Wong, 1981. *Capacitated arc routing problems*, Networks, **11(3)**, pp. 305–315.
- [16] GW Groves, J le Roux and JH van Vuuren, 2005. *On a routing and scheduling problem concerning multiple edge traversals in graphs*, Networks, **46(2)**, pp. 69–81.
- [17] M Guan, 1962. *Graphic programming using odd and even points*, Chinese Mathematics, **1**, pp. 273–277.
- [18] M Guan, 1984. *On the windy postman problem*, Discrete Applied Mathematics, **9**, pp. 41–46.
- [19] C Hierholzer, 1873. *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechnung zu umfahren*, Mathematische Annalen, **6**, pp. 30–32.
- [20] PF Lemieux and L Champagna, 1984. *The snow ploughing problem solved by a graph theory algorithm*, Civil Engineering Systems, **1(6)**, pp. 337–341.
- [21] JK Lenstra and AHG Rinnooy Kan, 1976. *On general routing problems*, Networks, **6(3)**, pp. 273–280.
- [22] C Malandraki and MS Daskin, 1993. *The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem*, European Journal of Operational Research, **65(2)**, pp. 218–234.
- [23] J Matoušek and J Nešetřil, 2003. *Invitation to Discrete Mathematics*, Clarendon Press, Oxford.
- [24] E Minieka, 1979. *The Chinese postman problem for mixed networks*, Management Science, **25(7)**, pp. 643–648.

- [25] CS Orloff, 1974. *A fundamental problem in vehicle routing*, Networks, **4**(1), pp. 35–64.
- [26] CH Papadimitriou, 1976. *On the complexity of edge traversing*, Journal of the Association of Computing Machinery, **23**(3), pp. 544–554.
- [27] WL Pearn, 1991. *Augment-insert algorithms for the capacitated arc routing problem*, Computers and Operations Research, **18**(2), pp. 189–198.
- [28] Y Saruwatari, R Hirabayashi and N Nishida, 1992. *Node duplication lower bounds for the capacitated arc routing problem*, Journal of the Operations Research Society of Japan, **35**(2), pp. 119–133.
- [29] Y Saruwatari, R Hirabayashi and N Nishida, 1992. *Subtour elimination algorithm for the capacitated arc routing problem*, pp. 334–341 in W Bühler, G Feichtinger, R Hartl, F Radermacher and P Stähly (Eds), *Papers of the 19th Annual Meeting / Vorträge der 19. Jahrestagung*, Operations Research Proceedings, Springer, Berlin.
- [30] WT Tutte, 1948. *The dissection of equilateral triangles into equilateral triangles*, Mathematical Proceedings of the Cambridge Philosophical Society, **44**(4), pp. 463–482.
- [31] DB West, 1996. *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River (NJ).
- [32] Z Win, 1989. *On the windy postman problem on eulerian graphs*, Mathematical Programming, **44**, pp. 97–112.



Hamiltonian graphs

Contents

11.1	Introduction	323
11.2	Which graphs are hamiltonian?	325
11.3	The closure function	329
11.4	The travelling salesman problem	334
11.5	Almost traceability and almost hamiltonicity	344
	Exercises	348
	Computer exercises	350
	Projects	352
	Further reading	357

11.1 Introduction

In this chapter, we turn our attention to the notion of a *hamiltonian graph* — a graph in which there is a cycle passing through each of its vertices. The name “hamiltonian” is derived from a game invented by the Irish mathematician, Sir William Rowan Hamilton. In 1857, Hamilton introduced a game consisting of a solid regular dodecahedron (see Figure 11.1(a)) made of wood, twenty pegs (one inserted at each vertex), and a supply of string. Every vertex was given the name of an important city of the time — Brussels, Canton, Delhi, . . . , Zanzibar. The aim of the game, called *A Voyage Round the World*¹, was to find a route along the edges of the dodecahedron that visits each city exactly once and that ended at the city where it began. In order for the player to keep track of the cities visited, the player could use the string to join the pegs in the order in which the route was traversed. Hamilton sold the idea of his game to a wholesale dealer of games and puzzles for 25 pounds. The game did not prove to be very popular, however, possibly because it is easy to solve the puzzle.

The aforementioned game suggests the graphical concept called *hamiltonicity*. A cycle of a graph G containing every vertex of G is called a **hamiltonian cycle** of G . A **hamiltonian graph** is a graph possessing a hamiltonian cycle. A path of G containing every vertex of G is called a **hamiltonian path** and a graph containing a hamiltonian path is called **traceable**. Every hamiltonian graph is therefore trace-

¹Also known as the *Traveller’s Dodecahedron*.

Label	City	Label	City	Label	City	Label	City
<i>B</i>	Brussels	<i>H</i>	Hanover	<i>N</i>	Naples	<i>T</i>	Tobolsk
<i>C</i>	Canton	<i>J</i>	Jeddo	<i>P</i>	Paris	<i>V</i>	Vienna
<i>D</i>	Delhi	<i>K</i>	Kashmere	<i>Q</i>	Quebec	<i>W</i>	Washington
<i>F</i>	Frankfurt	<i>L</i>	London	<i>R</i>	Rome	<i>X</i>	Xeres
<i>G</i>	Geneva	<i>M</i>	Moscow	<i>S</i>	Stockholm	<i>Z</i>	Zanzibar

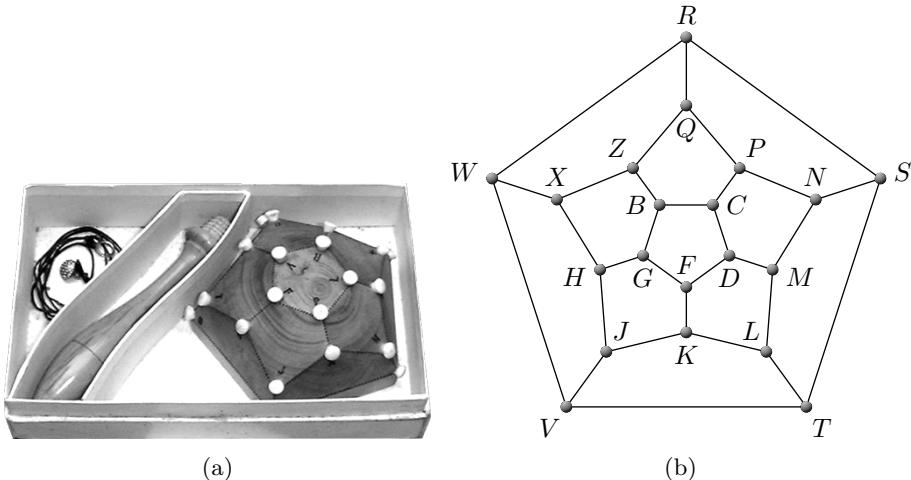


Figure 11.1: (a) The handle and dodecahedron in Hamilton's game *A Voyage Round the World* that a player would assemble. The supply of string would then be wrapped around the dodecahedron pegs with labels corresponding to the twenty cities shown in the table. (b) The graph of the dodecahedron; the vertices of this graph represent the corner points of the dodecahedron (the cities in Hamilton's game), and two vertices are adjacent in the graph if there exists an edge (the intersection between two faces) between the corresponding corner points of the dodecahedron.

able, but the converse is not true, as exemplified by the path graph P_n of order n . Hamilton's game *A Voyage Round the World* therefore requires the player to find a hamiltonian cycle in the graph of the dodecahedron, shown in Figure 11.1(b). Solving the puzzle is left as an exercise. The graph in Figure 11.1(b) is therefore an example of a hamiltonian graph.

An earlier example of a problem that can be expressed in terms of hamiltonian cycles is the *Knight's Tour Problem*: “Can a knight visit each square of a standard 8×8 chessboard exactly once by a sequence of knight’s moves, and finish on the same square where it began?” In order to see the connection between this problem and that of finding hamiltonian cycles in graphs, note that the knight’s tour problem can be represented by a graph in which each vertex corresponds to a square of the chessboard, and edges correspond to those pairs of squares connected by a knight’s move. The resulting graph, which has order 64 and size 168, actually contains several hamiltonian cycles, one of which is shown in Figure 11.2.

- ❖ The reader should now be able to attempt Exercises 11.1–11.3.

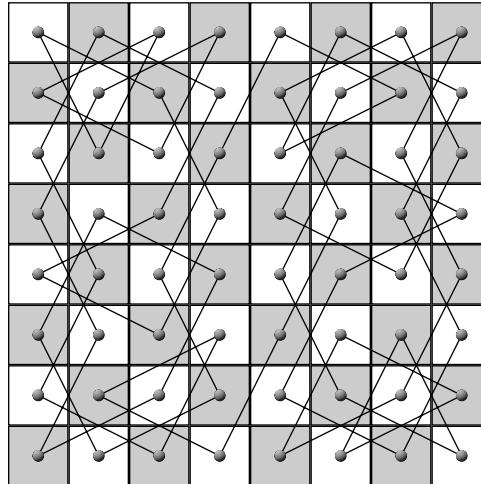


Figure 11.2: A solution to the *Knight's Tour Problem*.

11.2 Which graphs are hamiltonian?

The problem of deciding whether or not a given graph is hamiltonian seems similar to the problem of deciding whether or not a graph is eulerian. Since there is a simple characterisation for eulerian graphs, one might expect there to be a simple characterisation for hamiltonian graphs. Unfortunately, no such characterisation is known. To date, there is no known condition in closed form that is both necessary and sufficient for a graph to be hamiltonian. Indeed, the problem of finding a simple characterisation of hamiltonian graphs is considered one of the major unsolved problems in graph theory.

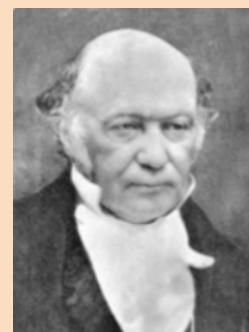
If G is a hamiltonian graph, then G contains a hamiltonian cycle. Hence a necessary condition for G to be hamiltonian is that G is connected, has no cut-vertices and, of course, has order at least 3. Recall that the number of components of a graph G is denoted by $k(G)$. A forest in which every component is a path is called a **linear forest**. Another necessary condition for a graph to be hamiltonian is the following.

Theorem 11.1 *If G is a hamiltonian graph, then for every nonempty proper subset S of vertices of G , $k(G - S) \leq |S|$.*

Proof Let C be a hamiltonian cycle of G . Let S be a proper nonempty subset of vertices of G , and let H be obtained from C by deleting the vertices of S , i.e. $H = C - S$. Then, H is a linear forest with at most $|S|$ components, and so $k(H) \leq |S|$. Since H is a spanning subgraph of $G - S$, we have that $k(G - S) \leq k(H) \leq |S|$. ■

We remark that [Theorem 11.1](#) may be restated in its contrapositive form as follows: If G is a graph and $k(G - S) > |S|$ for some nonempty proper subset S of vertices of G , then G is not hamiltonian. As an illustration of [Theorem 11.1](#), consider the graph $G_{11.1}$ of order 9 in [Figure 11.3\(a\)](#) where $S = \{u, v, w, x\}$. The graph $G_{11.1} - S$ contains five components (each of which consists of a singleton vertex). Thus, $k(G_{11.1} - S) = 5 > 4 = |S|$. Hence, by the contrapositive of [Theorem 11.1](#), the graph $G_{11.1}$ is not hamiltonian.

Sir **William Rowan Hamilton** was born in Dublin, Ireland on 4 August 1805. He was appointed a Professor of Astronomy at Trinity College in Dublin where he obtained his master's degree in 1837. He then took up residence at Dunsink Observatory where he spent most of his working life. He made important contributions to classical mechanics, optics and algebra. He is best known in mathematical physics for his contribution to the reformulation of Newtonian mechanics, now called hamiltonian mechanics. In pure mathematics, he is best known as the inventor of quaternions. In graph theory, hamiltonian paths and cycles are named after him. He invented the icosian game, now also known as *Hamilton's Puzzle* or *A Voyage Round the World*, which involves finding a hamiltonian cycle in the edge graph of the dodecahedron. He was elected to the President's Chair in the Royal Irish Academy in 1837. There is also a commemorative coin in his honour, which was issued by the Central Bank of Ireland. He died in Dublin on 2 September 1865.



Biographic note 28: Sir William Hamilton (1805–1865)

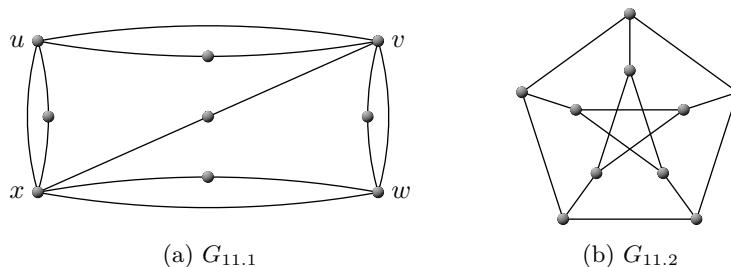


Figure 11.3: (a) The graph $G_{11.1}$, and (b) the Petersen graph $G_{11.2}$.

The necessary condition for a graph to be hamiltonian given in [Theorem 11.1](#) is not sufficient, however. For example, the Petersen graph $G_{11.2}$, shown in [Figure 11.3\(b\)](#), satisfies $k(G_{11.2} - S) \leq |S|$ for every nonempty proper subset S of vertices of $G_{11.2}$. Yet, the Petersen graph is not a hamiltonian graph (see [Exercise 11.7](#)).

Several sufficient conditions have also been established for a graph to be hamiltonian. For a graph G of order at least 3, it seems logical that the more edges G has, the more likely it is to be hamiltonian. The simplest sufficient condition for hamiltonicity utilising this observation is due to [Dirac \[9\]](#). The proof technique employed to establish this condition is typical of the type of argument used to prove that a graph is hamiltonian.

Theorem 11.2 ([Dirac's Theorem](#)) *Let G be a graph of order $n \geq 3$ with minimum degree δ . If $\delta \geq \frac{n}{2}$, then G is hamiltonian.*

Proof If $n = 3$, then $\delta \geq 2$. This implies that $G \cong K_3$ and, hence, G is hamiltonian. So we may assume that $n \geq 4$.

Let $P : v_1 \dots v_k$ be a longest path in G . Then every vertex adjacent to v_1 lies on P and every vertex adjacent to v_k lies on P , for otherwise there would be a longer path in G than P . Hence,

$$k = |V(P)| \geq d(v_1) + 1 \geq \delta + 1 \geq \frac{n}{2} + 1.$$

We show that there exists some vertex v_i , where $i \in \{2, \dots, k\}$, such that v_1 is adjacent to v_i , and v_k is adjacent to v_{i-1} . If this were not the case, then, whenever v_1 is adjacent to a vertex v_i , the vertex v_k would *not* be adjacent to v_{i-1} . This implies that at least $d(v_1) \geq \frac{n}{2}$ vertices on P different from v_k are not adjacent to v_k . Hence,

$$d(v_k) \leq |V(P) - \{v_k\}| - d(v_1) \leq (n-1) - \frac{n}{2} < \frac{n}{2},$$

which contradicts the fact that $d(v_k) \geq \delta \geq \frac{n}{2}$. Hence, as claimed, there is some vertex v_i ($i \in \{2, \dots, k\}$) adjacent to v_1 with v_{i-1} adjacent to v_k . Thus, G has a cycle

$$C : v_1 v_i v_{i+1} \dots v_k v_{i-1} v_{i-2} \dots v_2 v_1$$

that contains all the vertices of P . We show that C is a hamiltonian cycle of G . If this were not the case, then there would be some vertex u of G that does not belong to C . By hypothesis, $d(u) \geq \frac{n}{2}$. Since P contains $k \geq \frac{n}{2} + 1$ vertices, there are fewer than $\frac{n}{2}$ vertices not on C . So u must be adjacent to some vertex v that lies on C . The edge uv together with the cycle C , however, contains a path whose length exceeds that of P , which contradicts our choice of P . Hence C contains all the vertices of G and so G is hamiltonian. ■

The sufficient condition in [Theorem 11.2](#) is not a necessary condition for a graph to be hamiltonian. For example, the cycle C_n ($n \geq 5$) is hamiltonian but has minimum degree 2, which is less than $\frac{n}{2}$. Many other hamiltonian graphs also cannot be shown to be hamiltonian using [Theorem 11.2](#).

A corresponding degree condition to that presented in [Theorem 11.2](#) also exists for the traceability of a graph.

Corollary 11.3 *Let G be a graph of order n with minimum degree δ . If $\delta \geq (n-1)/2$, then G is traceable.*

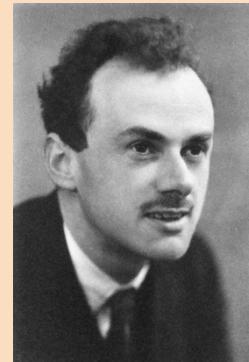
Proof If $n = 1$, then $G \cong K_1$, and G contains a (trivial) hamiltonian path. So we may assume that $n \geq 2$. Let H be the graph obtained from G by adding a new vertex v and joining v to every vertex of G . Then H has order $n+1$, and so v has degree n in H . Moreover, for every vertex u of G ,

$$d_H(u) = d_G(u) + 1 \geq \frac{n-1}{2} + 1 = \frac{n+1}{2} = \frac{|V(H)|}{2}.$$

Hence H is a graph of order $n+1$ with minimum degree $\delta(H) \geq (n+1)/2$. It therefore follows from [Theorem 11.2](#) that H contains a hamiltonian cycle C . By removing the vertex v from C , we obtain a hamiltonian path in G . ■

It is interesting to note that if G is a bipartite hamiltonian graph with partite sets V_1 and V_2 , then it follows that $|V_1| = |V_2|$ (see [Exercise 11.6](#)). Our final result of this section extends [Dirac's Theorem](#) ([Theorem 11.2](#)) to this class of graphs.

Paul Adrien Maurice Dirac was born in Bristol, England on 8 August 1902. He obtained a degree in electrical engineering at the University of Bristol in 1921, and obtained his doctorate at St John's College, Cambridge in 1926 with the first thesis on quantum mechanics to be submitted at any university. He is regarded as one of the most significant physicists of the 20th century. Among other discoveries, he formulated the Dirac equation which describes the behaviour of fermions and predicted the existence of antimatter. He also made significant contributions to the reconciliation of general relativity with quantum mechanics. He was the Lucasian Professor of Mathematics at the University of Cambridge, a member of the Center for Theoretical Studies at the University of Miami, and spent the last decade of his life at Florida State University. He shared the 1933 Nobel Prize for physics with Erwin Schrodinger "for the discovery of new productive forms of atomic theory." Dirac was also awarded the Royal Medal in 1939 and both the Copley Medal and the Max Planck Medal in 1952. He was elected a Fellow of the Royal Society in 1930, an Honorary Fellow of the American Physical Society in 1948, and an Honorary Fellow of the Institute of Physics, London in 1971. He died in Tallahassee, Florida on 20 October 1984.



Biographic note 29: Paul Dirac (1902–1984)

Theorem 11.4 Let G be a bipartite graph with partite sets V_1 and V_2 such that $|V_1| = |V_2| = n \geq 2$ and with minimum degree δ . If $\delta > \frac{n}{2}$, then G is hamiltonian.

Proof By contradiction. Suppose, to the contrary, that there exist nonhamiltonian bipartite graphs satisfying the hypothesis of the theorem and let G be such a maximal graph. Since $K_{n,n}$ is hamiltonian, the graph G must contain a pair u, v of nonadjacent vertices with $u \in V_1$ and $v \in V_2$.

The maximality of G implies that $G + uv$ is hamiltonian. Thus, $G + uv$ has a hamiltonian cycle, which contains the edge uv . Hence, G contains a hamiltonian $u-v$ path

$$P : u = v_1 v_2 \cdots v_{2n} = v,$$

where $v_i \in V_1$ if i is odd and $v_j \in V_2$ if j is even (since G is bipartite). If $v_1 v_i \in E(G)$, where $i \in \{2, \dots, 2n\}$ is even, then $v_{i-1} v_{2n} \notin E(G)$, for otherwise

$$v_1 v_i v_{i+1} \cdots v_{2n-1} v_{2n} v_{i-1} v_{i-2} \cdots v_2 v_1$$

is a hamiltonian cycle of G (which would contradict the fact that G is nonhamiltonian). Hence, for each vertex in $\{v_2, v_4, \dots, v_{2n}\}$ adjacent to v_1 there is a vertex in $\{v_1, v_3, \dots, v_{2n-1}\}$ not adjacent to v_{2n} . It therefore follows that

$$d(v_{2n}) \leq n - d(v_1) < n - \frac{n}{2} = \frac{n}{2},$$

which contradicts the hypothesis. ■

❖ The reader should now be able to attempt Exercises 11.4–11.10.

11.3 The closure function

We begin this section by presenting a result first observed by [Bondy and Chvátal](#) [2]. This technique can be proved using the same technique employed in the proofs of Theorems 11.2 and 11.4.

Theorem 11.5 (Bondy and Chvátal's Theorem) *Let u and v be distinct nonadjacent vertices of a graph G of order $n \geq 3$ such that $d(u) + d(v) \geq n$. Then G is hamiltonian if and only if $G + uv$ is hamiltonian.*

Proof If G is hamiltonian, then certainly so too is $G + uv$. Conversely, suppose $G + uv$ is hamiltonian and let C be a hamiltonian cycle of $G + uv$. If C does not contain the edge uv , then C is a hamiltonian cycle of G and so G is hamiltonian. We may therefore assume that C contains the edge uv . Now $P = C - uv$ is a hamiltonian u - v path of G , that is $P : u = v_1 \dots v_n = v$ (say) contains every vertex of G .

We show that there exists some vertex v_i , with $i \in \{2, \dots, n\}$, such that $v_1v_i \in E(G)$ and $v_{i-1}v_n \in E(G)$. If this were not the case, then, whenever $v_1v_i \in E(G)$ for some $i \in \{2, \dots, n\}$, we would have $v_{i-1}v_n \notin E(G)$. This would mean that for each vertex of $\{v_2, \dots, v_n\}$ adjacent to v_1 there is a vertex in $\{v_1, \dots, v_{n-1}\}$ not adjacent to v_n . Thus, $d(v_n) \leq (n-1) - d(v_1)$ so that

$$d(u) + d(v) \leq n-1,$$

a contradiction. Hence there exists some vertex v_i with $i \in \{2, \dots, n\}$ such that $v_1v_i \in E(G)$ and $v_{i-1}v_n \in E(G)$. But then

$$v_1, v_i, v_{i+1}, \dots, v_n, v_{i-1}, v_{i-2}, \dots, v_2, v_1$$

is a hamiltonian cycle of G and so G is hamiltonian. ■

[Theorem 11.5](#) suggests the following definition. The **closure** of a graph G of order n , denoted by $\text{cl}(G)$, is the graph obtained from G by recursively joining pairs of nonadjacent vertices whose degree sum is at least n (in the resulting graph at each stage) until no such pair remains. The closure function is illustrated in [Figure 11.4](#).

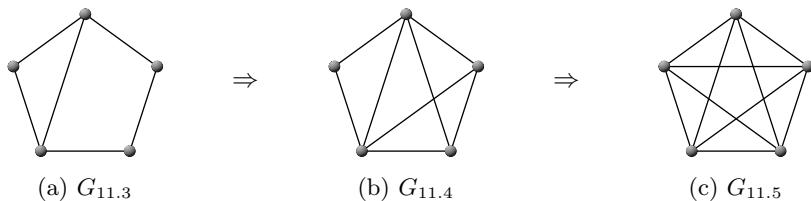


Figure 11.4: The closure $G_{11.5}$ of a graph $G_{11.3}$.

We show next that the closure $\text{cl}(G)$ of a graph G is a well-defined notion.

Theorem 11.6 *If G_1 and G_2 are two graphs obtained from a graph G of order $n \geq 3$ by iteratively joining pairs of nonadjacent vertices whose degree sum is at least n , then $G_1 = G_2$.*

John Adrian Bondy was born a dual British and Canadian citizen in 1944. He obtained a doctorate in mathematics from Oxford University in 1969 under the supervision of Dominic Welsh based on a thesis entitled *Some uniqueness theorems in graph theory*. He was appointed as professor of graph theory at the University of Waterloo between 1969 and 1994, before holding the same position at Université Lyon 1 until his retirement in 2011. He supervised twelve PhD students. Together with [USR Murty](#) he wrote *Graph Theory with Applications*, a textbook that was published in 1976 (followed by a new graduate text version in 2008), and which was instantly recognised as a reference text for graph theory courses all around the world. He also served (with [Murty](#)) as a managing editor and co-editor-in-chief of the Journal of Combinatorial Theory, Series B from the late 1970s until 2004. He is well known for his (nonclosed form) vertex degree characterisation of hamiltonian graphs in 1972, now known as the Bondy-Chvátal theorem.



Biographic note 30: John Bondy (1944–present)

Proof Let e_1, \dots, e_k and f_1, \dots, f_ℓ be the sequences of edges added to G to obtain G_1 and G_2 , respectively. We show that each edge in the set $\{e_1, \dots, e_k\}$ is an edge of G_2 and that each edge in the set $\{f_1, \dots, f_\ell\}$ is an edge of G_1 . Suppose, to the contrary, that some edge $e_i \in \{e_1, \dots, e_k\}$ does not belong to G_2 . Let t be the smallest integer such that e_{t+1} is not an edge of G_2 . Suppose $e_{t+1} = uv$ and let $H = G + \{e_1, \dots, e_t\}$. Then H is a subgraph of both G_1 and G_2 . It furthermore follows from the definition of G_1 , that

$$d_H(u) + d_H(v) \geq n.$$

Therefore,

$$d_{G_2}(u) + d_{G_2}(v) \geq d_H(u) + d_H(v) \geq n.$$

This is a contradiction, however, since u and v are nonadjacent vertices of G_2 . Hence each edge e_i belongs to G_2 . It can be shown in a similar fashion that each edge in the set $\{f_1, \dots, f_\ell\}$ belongs to G_1 . This shows that $G_1 = G_2$. ■

Our next result is a simple consequence of the definition of closure and [Theorem 11.5](#).

Theorem 11.7 *A graph is hamiltonian if and only if its closure is hamiltonian.*

Proof If we apply [Theorem 11.5](#) every time an edge is added in the formation of the closure, then we see that a graph G is hamiltonian if and only if $\text{cl}(G)$ is hamiltonian. ■

Since each complete graph with at least three vertices is hamiltonian, we obtain the following sufficient condition, due to [Bondy and Chvátal \[2\]](#), for a graph to be hamiltonian.

Corollary 11.8 *Let G be graph of order $n \geq 3$. If the closure $\text{cl}(G)$ is complete, then G is hamiltonian.*

Another consequence of [Theorem 11.7](#) is due to [Ore \[19\]](#).

Theorem 11.9 ([Ore's Theorem](#)) *If $d(u) + d(v) \geq n$ for all pairs u, v of nonadjacent vertices of a graph G of order $n \geq 3$, then G is hamiltonian.*

Proof Since the closure of G is complete, the result follows immediately from [Corollary 11.8](#). ■

Václav (Vašek) Chvátal was born in Prague in 1946. He studied at Charles University in Prague under the supervision of Zdeněk Hedrlín. In 1968, he fled from Czechoslovakia, three days after the Soviet invasion, and completed his doctorate in mathematics at the University of Waterloo under the supervision of Crispin Nash-Williams, in 1970. He subsequently held positions at McGill University (1971 and 1978–1986), Stanford University (1972 and 1974–1977), the Université de Montréal (1972–1974 and 1977–1978), and Rutgers University (1986–2004) before returning to Montreal to occupy the Canada Research Chair in Combinatorial Optimisation at Concordia (2004–2011) and the Canada Research Chair in Discrete Mathematics (2011–2014). He retired in 2014 after having made significant contributions in many areas of graph theory and combinatorics. In a 1973 paper he introduced the concept of graph toughness as a measure of graph connectivity. He is known for proving many important results, such as the art gallery theorem. He wrote a popular textbook entitled *Linear Programming*, which was published in 1983 and introduced the notion of a cutting-plane proof in linear programming. In 2015, he was awarded the [John von Neumann Theory Prize](#) and in 2017 the Frederick W Lanchester Prize.



Biographic note 31: Václav Chvátal (1946–present)

We observed earlier that if a graph G of order at least 3 has sufficiently many edges, then it is hamiltonian. The next consequence of [Theorem 11.7](#) is a result in the spirit of this observation.

Corollary 11.10 *Let G be a graph of order $n \geq 3$ and size m . If*

$$m \geq \binom{n-1}{2} + 2,$$

then G is hamiltonian.

Proof If G is complete, then G is hamiltonian. Suppose, therefore, that G is not complete, but satisfies the hypothesis of the corollary. Let u and v be two distinct nonadjacent vertices of G , and define $H = G - \{u, v\}$. Then, $m(G) = m(H) + d(u) + d(v)$. Since

$$m(H) \leq m(K_{n-2}) = \binom{n-2}{2},$$

it therefore follows that

$$d(u) + d(v) = m(G) - m(H) \geq \binom{n-1}{2} + 2 - \binom{n-2}{2} = n.$$

Hence, by [Theorem 11.9](#), G is hamiltonian. ■

We close this section with the following result, due to [Chvátal](#) [5], which may be deduced from [Corollary 11.8](#).

Theorem 11.11 ([Chvátal's Theorem](#)) *Let G be a graph of order $n \geq 3$ with degree sequence d_1, \dots, d_n where $d_1 \leq \dots \leq d_n$. If, for every integer $i < \frac{n}{2}$ we have that*

$$d_i \leq i \text{ implies that } d_{n-i} \geq n - i,$$

then G is hamiltonian.

Proof We show that the closure $\text{cl}(G)$ of G is complete. Assume, to the contrary, that $\text{cl}(G)$ is not complete, let $H = \text{cl}(G)$ and let $V = V(G) = V(H)$. Among all nonadjacent vertices in H , let x and y be those for which $d_H(x) + d_H(y)$ is as large as possible. Without loss of generality, we may assume that $d_H(x) \leq d_H(y)$. By definition of the closure of a graph, $d_H(x) + d_H(y) < n$. Letting $i = d_H(x)$, it follows that $i < \frac{n}{2}$ and that $d_H(y) < n - d_H(x) = n - i$. Let X be the set of vertices in $V \setminus \{y\}$ that are not adjacent to y in H , and let Y be the set of vertices in $V \setminus \{x\}$ that are not adjacent to x in H . Note that $x \in X$ and $y \in Y$. Therefore,

$$|X| = n - 1 - d_H(y) > (n - 1) - (n - i) = i - 1,$$

and

$$|Y| = n - 1 - d_H(x) = n - 1 - i.$$

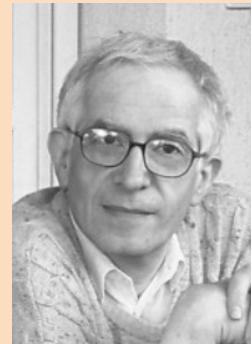
By our choice of x and y , each vertex in X has degree at most $d_H(x)$ and each vertex in Y has degree at most $d_H(y)$. Hence, H has at least $|X| \geq i$ vertices of degree at most i , and at least $|Y \cup \{x\}| \geq n - i$ vertices of degree strictly less than $n - i$. Therefore, $d_i \leq i$ and $d_{n-i} < n - i$. This, however, contradicts the hypothesis of the theorem. We conclude that the closure $\text{cl}(G)$ of G is complete, from which the desired result follows by [Corollary 11.8](#). ■

The sufficient condition for a graph to be hamiltonian in [Chvátal's Theorem](#) ([Theorem 11.11](#)) is not a necessary condition for hamiltonicity. Consider, for example, the hamiltonian graph $G_{11.6}$ of order $n = 6$ in [Figure 11.5\(a\)](#) with degree sequence $d_1, d_2, d_3, d_4, d_5, d_6 = 2, 2, 2, 3, 3, 4$. Note that $2 < 3 = \frac{n}{2}$ and $d_2 \leq 2$, but $d_4 = 3 < 4$. That is, when $i = 2$, we have that $i < \frac{n}{2}$ and $d_i \leq i$, but $d_{n-i} < n - i$, which violates the degree constraint condition in [Chvátal's Theorem](#).

The following (weaker) sufficient condition for a graph to be hamiltonian, due to [Pósa](#) [20], is an immediate consequence of the proof of [Chvátal's Theorem](#).

Theorem 11.12 ([Pósa's Theorem](#)) *Let G be a graph of order $n \geq 3$ and let $n_{\leq i}$ denote the number of vertices of degree at most i in G . If $n_{\leq i} < i$ for every integer $i < \frac{n}{2}$, then G is hamiltonian.*

Lajos Pósa was born in Budapest, Hungary on 9 December 1947. He obtained a doctorate in mathematics in 1983 from ELTE University based on a dissertation entitled *Hamiltonian circuits of random graphs*. From 1971 to 1982 he worked in the Department of Mathematical Analysis at ELTE University. From 1984 to 2002 he held a position in the Department of Computer Science at ELTE University, and since 2002 he has been a researcher at the Hungarian Academy of Sciences within the Rényi Mathematical Institute. He was a child prodigy, and his first paper was written when he was only 13, co-authored with [Paul Erdős](#). Pósa won second prize (silver medal) in 1965 (Germany) and first prize (gold medal) in the International Mathematical Olympiad in 1966 (Bulgaria). In graph theory he is well known for what is now known as the Erdős-Pósa theorem. Despite his significant results in mathematical research, he is best known for finding and teaching gifted students. In 2011, he was awarded the Széchenyi Prize in recognition of those who have made an outstanding contribution to academic life in Hungary.



Biographic note 32: Lajos Pósa (1947–present)

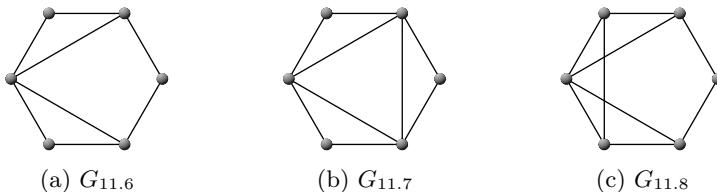


Figure 11.5: Graphs used as examples in the theorems of this section.

Proof We show by contradiction that the closure $\text{cl}(G)$ of G is complete. Assume, to the contrary, that $\text{cl}(G)$ is not complete and let $H = \text{cl}(G)$. Following the proof of [Theorem 11.11](#) exactly, we have that H has at least $|X| \geq i$ vertices of degree at most i where $i = d_H(x) < \frac{n}{2}$. Hence, the proof of [Theorem 11.11](#) shows that there exists an integer $i < \frac{n}{2}$ such that $n_{\leq i} \geq i$, contradicting the hypothesis of the theorem. Hence, $\text{cl}(G)$ of G is complete and the desired result follows from [Corollary 11.8](#). ■

We remark that if G is a graph of order $n \geq 3$ satisfying the hypothesis of [Pósa's Theorem](#) ([Theorem 11.12](#)), then G has no vertex of degree 1. Furthermore, if $n \geq 5$, then G has at most one vertex of degree 2. In particular, the graph $G_{11.7}$ in [Figure 11.5\(b\)](#) of order $n = 6$ with degree sequence $d_1, d_2, d_3, d_4, d_5, d_6 = 2, 2, 2, 4, 4, 4$ does not satisfy the hypothesis of [Pósa's Theorem](#) (since it has three vertices of degree 2). Note, however, that the graph $G_{11.7}$ does, in fact, satisfy the hypothesis of [Chvátal's Theorem](#). Hence we can deduce that the graph $G_{11.7}$ is hamiltonian by [Chvátal's Theorem](#), but not by [Pósa's Theorem](#). This illustrates that [Chvátal's Theorem](#) is stronger than [Pósa's Theorem](#).

We close with the remark that Pósa's Theorem is stronger than Ore's Theorem (Theorem 11.9). For example, the graph $G_{11.8}$ in Figure 11.5(c) of order $n = 6$ with degree sequence $d_1, d_2, d_3, d_4, d_5, d_6 = 2, 3, 3, 3, 3, 4$ satisfies the hypothesis of Pósa's Theorem, but does not satisfy the hypothesis of Ore's Theorem (since G contains two nonadjacent vertices whose degree sum is less than n). Hence we can deduce that the graph $G_{11.8}$ is hamiltonian by Pósa's Theorem, but not by Ore's Theorem.

- ❖ The reader should now be able to attempt Exercises 11.11–11.16.

11.4 The travelling salesman problem

Suppose a travelling salesman plans to visit a number of cities and then return to his starting point. Given the travel distance between each pair of cities, what route should the salesman take to minimise his total distance travelled subject to the constraint that he visits each city exactly once. This is the well-known **Travelling Salesman Problem** (TSP).

An instance of the TSP can be modelled by a weighted complete graph whose vertices correspond to the cities that the travelling salesman plans to visit and where the weight of an edge joining two vertices denotes the travel distance between the corresponding cities. Hence, in graph theoretic terms, the TSP can be formulated as follows: *Find a hamiltonian cycle of minimum weight in a weighted complete graph.*

A natural (albeit rather crude) brute-force approach toward solving a TSP instance on n cities would be to consider all $(n - 1)!/2$ distinct hamiltonian cycles of the weighted complete input graph and to select one achieving the smallest weight. This approach is expected to be very time-consuming, however, in view of the super-exponential growth in the amount of work that has to be done as n increases, as illustrated in Table 11.1. As is clear from the table, the aforementioned brute-force TSP solution approach is not practical as many real-world applications of the TSP involve hundreds or even thousands of cities. Fortunately, there are better ways to solve TSP instances than following this brute-force approach.

n	$(n - 1)!/2$	Time
5	12	—
10	181 440	~ 0.2 seconds
15	43 589 145 600	~ 12 hours
20	60 822 550 204 416 000	~ 10^3 years
25	310 224 200 866 619 719 680 000	~ 10^9 years

Table 11.1: The time-complexity of considering each of the $(n - 1)!/2$ distinct solutions to a TSP instance on n cities. The column labelled *Time* contains an indication of the duration of such a brute-force TSP solution procedure if a million hamiltonian cycles were to be considered per second.

Recall that we defined an efficient algorithm in Chapter 3 as one with a worst-case time complexity bounded from above by a polynomial function of the input size. Unfortunately, no efficient algorithm for solving the TSP has yet been found.

In fact, the following underlying decision problem, denoted by $D_{\text{TSP}}(G, k)$, is NP-complete: Given a weighted complete graph G and a positive integer k , does there exist a hamiltonian cycle in G of weight at most k ?

A considerable amount of research has gone into exact TSP solution techniques within the realm of (mixed) integer programming. The reader is referred to §9.6 of the operations research text by Winston [23] for a very basic integer programming model formulation of the TSP. Impressive results have, in fact, been achieved solving large TSP instances *via* standard integer programming solution techniques such as the celebrated branch-and-cut method. The reader is referred to the authoritative work on the TSP by Applegate *et al.* [1] for an excellent account of the most important findings related to investigations centred on the TSP. The algorithm that holds the record in terms of TSP solution speed is the state-of-the-art Concorde algorithm [17], which is a combination of an impressive array of heuristic and cutting plane algorithms. TSPLIB is an online library of TSP test instances that may be used by researchers in order to test the efficiency of their TSP algorithms. The Concorde TSP solver has been used to obtain optimal solutions to all 110 TSP instances in this library; the largest involving 85 900 cities.

Since $D_{\text{TSP}}(G, k)$ is NP-complete and since we do not wish to diverge into the realm of integer programming, we turn our attention to efficient algorithms that can determine a hamiltonian cycle in a complete weighted graph whose weight is “close” to the weight of a hamiltonian cycle of minimum weight. In particular, we seek an *approximation algorithm*² which can quickly find a hamiltonian cycle in a complete weighted graph whose weight is bounded from above by a constant multiple of the weight of a hamiltonian cycle of minimum weight.

11.4.1 The nearest-neighbour heuristic

Perhaps the best-known heuristic³ for solving instances of the TSP is the so-called **nearest neighbour heuristic**. According to this heuristic, the salesman starts his tour from any city and iteratively constructs a longer and longer path of cities visited, each time visiting the closest unvisited city from the city last visited, until all cities have been visited. Thereafter, he returns to the original city. The heuristic is given in pseudocode form as [Algorithm 26](#).

Let us illustrate the working of [Algorithm 26](#) by means of a worked example. Consider the graph $G_{11.9}$ in [Figure 11.6](#) and suppose the initial vertex v is selected as v_3 . Therefore, **current vertex** is initially v_3 and so C is initialised as $C : v_3$ (weight 0) in [Step 2](#) of the algorithm. The closest unvisited vertex u in the set of unvisited city vertices $\{v_1, v_2, v_4, v_5\}$ from v_3 is either v_1 or v_2 (at a distance 2 from v_3). Suppose u is selected as v_1 . At this point the visitation path is updated to $C : v_3 v_1$ (weight 2), and **current vertex** is updated to the vertex last selected for visitation, v_1 .

²An *approximation algorithm* is an efficient algorithm for finding an approximate solution to an optimisation problem (most often, an NP-hard problem) with a provable guarantee on the quality of the approximate solution returned (in terms of its objective function value relative to the objective function value of an optimal solution).

³A *heuristic* is an efficient algorithm for finding an approximate solution to an optimisation problem for which there is no provable guarantee on the quality of the approximate solution returned. The word heuristic is derived from the Greek word *heuristikein*, which means *to find*. While a heuristic may return an optimal solution, this is usually not the case. In fact, heuristics often return relatively poor results (especially for NP-hard optimisation problems).

Algorithm 26: The Nearest neighbour heuristic

Input : A weighted complete graph G of order at least three and a vertex v of G .

Output : A hamiltonian cycle C of G .

```

1 current vertex  $\leftarrow v$ 
2  $C \leftarrow$  trivial path containing current vertex only
3 repeat
4   Let  $u$  be a vertex of  $G$  not already in  $C$  that is closest to current vertex
5   Append  $u$  to the end of  $C$ 
6   current vertex  $\leftarrow u$ 
7 until  $C$  contains all vertices of  $G$ 
8 Append  $v$  to the end of  $C$ 
9 print  $C$ 
```

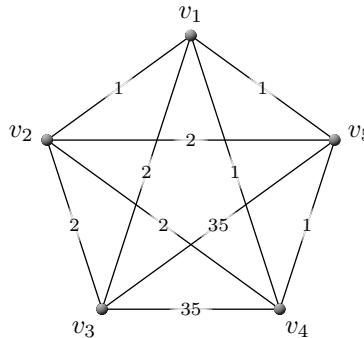


Figure 11.6: A graph model $G_{11.9}$ of a small TSP instance.

Now the closest unvisited vertex u in the set $\{v_2, v_4, v_5\}$ from v_1 can be chosen arbitrarily (they are each at a distance 1 from v_1). Suppose u is selected as v_5 . At this point the visitation path is updated to $C : v_3 v_1 v_5$ (weight 3), and **current vertex** is updated to v_5 .

The closest unvisited vertex u in the set $\{v_2, v_4\}$ from v_5 is v_4 (at a distance 1 from v_5). At this point the visitation path is updated to $C : v_3 v_1 v_5 v_4$ (weight 4), and **current vertex** is updated to v_4 .

Finally, u is taken as the only remaining unvisited vertex v_2 (at a distance 2 from v_4). At this point the visitation path is updated to $C : v_3 v_1 v_5 v_4 v_2$ (weight 6), and **current vertex** is updated to v_2 . Since C now contains all the vertices of G , the starting vertex v_3 is appended to the end of C to obtain the hamiltonian cycle $C : v_3 v_1 v_5 v_4 v_2 v_3$ of weight 8 in $G_{11.9}$. This is, in fact, a hamiltonian cycle of minimum weight in $G_{11.9}$ (although the heuristic usually does not produce a hamiltonian cycle of minimum weight).

A major weakness of the **nearest neighbour heuristic** is that its output is sensitive to both the vertex chosen to initialise the hamiltonian cycle, and the manner in which ties are broken during algorithmic execution. Reconsider, for example, the partial visitation tour $C : v_3 v_1$ above just before we selected v_5 as the next vertex to be visited. If, instead, we had chosen v_2 as the next vertex, the partial

tour would have been updated to $C : v_3 v_1 v_2$ (weight 3) and **current vertex** would have been updated to v_2 .

The closest unvisited vertex u in the set $\{v_4, v_5\}$ from v_2 could then have been chosen arbitrarily (each is at a distance 2 from v_s). Suppose u were to be selected as v_4 . Then the visitation path would have been updated to $C : v_3 v_1 v_2 v_4$ (weight 5), and **current vertex** would have been updated to v_4 .

Finally, u would then have been taken as the only remaining unvisited vertex v_5 (at a distance 1 from v_4). At this point the visitation path would have been updated to $C : v_3 v_1 v_2 v_4 v_5$ (weight 6). Since C now contains all the vertices of G , the starting vertex v_3 would have been appended to the end of C to obtain the hamiltonian cycle $C : v_3 v_1 v_5 v_4 v_2 v_3$ of weight 41 in $G_{11.9}$. The weight of this hamiltonian cycle is more than five times that of a minimum-weight hamiltonian cycle in $G_{11.9}$!

The aforementioned potentially poor performance of the **nearest neighbour heuristic** is not merely a result of a single exceptional TSP instance. The following result is a dire warning that the **nearest neighbour heuristic** cannot, in general, be trusted to deliver high-quality approximate solutions to instances of the TSP.

Theorem 11.13 *Let G be a weighted complete graph. The weight of a hamiltonian cycle of G returned by the **nearest neighbour heuristic** cannot necessarily be bounded from above by a constant multiple of the minimum weight of a hamiltonian cycle of G .*

Proof Let G be a weighted complete graph of order $n \geq 4$ with vertex set $V(G) = \{v_1, \dots, v_n\}$. Suppose $w(v_1 v_2) = 1$ and $w(v_1 v_3) = w(v_2 v_3) = 2$. Suppose, furthermore, that the weights of all the other edges incident with v_1 are 1, those incident with v_2 are 2, and those incident with v_3 are $n^2 + 2n$. Suppose, finally, that all the remaining edges have weight 1 and let C be a hamiltonian cycle of G returned by the **nearest neighbour heuristic**, starting from any vertex other than v_3 .

If $v_1 v_3 v_2$ were consecutive vertices in C , then when v_1 was the current vertex, both v_2 and v_3 would have been unvisited neighbours of v_1 . But since $w(v_1 v_2) = 1 < 2 = w(v_1 v_3)$, the vertex v_2 would then have been visited before v_3 according to the **nearest neighbour heuristic**, a contradiction. Similarly, if $v_2 v_3 v_1$ were consecutive vertices in C , then when v_2 was the current vertex, both v_1 and v_3 would have been unvisited neighbours of v_2 . But since $w(v_1 v_2) = 1 < 2 = w(v_1 v_3)$, the vertex v_1 would then have been visited before v_3 according to the **nearest neighbour heuristic**, again a contradiction. Hence it follows that neither $v_1 v_3 v_2$ nor $v_2 v_3 v_1$ appear as consecutive vertices along C .

C therefore contains at least one edge incident with v_3 of weight $n^2 + 2n$. The sum of the weights of the two edges incident with v_3 in C is consequently at least $n^2 + 2n + 2$, while the remaining $n - 2$ edges of C all have weight 1, so that $w(C) \geq n(n + 3)$. For a hamiltonian cycle C^* of minimum weight in G , however, it holds that $w(C^*) = n + 3$ (one such hamiltonian cycle of minimum weight is $v_1 v_3 v_2 v_4 v_5 \dots v_{n-1} v_n v_1$). Therefore, $w(C) \geq nw(C^*)$. ■

The edge weights of the graph $G_{11.9}$ in Figure 11.6 conform to the general edge weight construction in the proof of Theorem 11.13 in the special case where $n = 5$.

❖ The reader should now be able to attempt Exercise 11.17.

11.4.2 A lower bound on TSP solutions

Recall from [Chapter 5](#) that a minimum spanning tree of a connected weighted graph is a spanning tree of minimum weight in the graph. In [Section 5.3](#), we considered efficient algorithms ([Kruskal's algorithm](#) and [Prim's algorithm](#)) for finding a minimum spanning tree in a connected weighted graph. In this section, we show that we can use the weight of such a spanning tree to obtain a lower bound on the minimum weight of a hamiltonian cycle in a weighted complete graph.

In particular, the following result shows that the weight of a shortest hamiltonian cycle in a weighted complete graph G is bounded from below by the weight of a minimum spanning tree in G added to the minimum weight of an edge not in the same minimum spanning tree in G .

Theorem 11.14 *Let G be a weighted complete graph of order $n \geq 3$, let C^* be a hamiltonian cycle of minimum weight in G and let T^* be a minimum spanning tree of G . If e^* is a minimum-weight edge of G not in T^* , then $w(C^*) \geq w(T^*) + w(e^*)$.*

Proof Let e be an edge of C^* not in T^* , that is, $e \in E(C^*) \setminus E(T^*)$. Removing the edge e from C^* produces a spanning path $C^* - e$ of G . Hence the weight of $C^* - e$ is at least that of the minimum spanning tree T^* of G , that is, $w(C^* - e) \geq w(T^*)$. Since $e \in E(G) \setminus E(T^*)$, it follows from the definition of the edge e^* that $w(e^*) \leq w(e)$. Therefore, $w(C^*) = w(C^* - e) + w(e) \geq w(T^*) + w(e^*)$. ■

We can apply [Kruskal's algorithm](#) ([Algorithm 14](#)) to find a minimum spanning tree in a weighted complete graph and readily compute the lower bound of [Theorem 11.14](#). To illustrate this, consider the weighted complete graph $G_{11.10}$ in [Figure 11.7\(a\)](#). One minimum spanning tree of $G_{11.10}$ is the tree T^* in [Figure 11.7\(b\)](#) whose edges are v_1v_4 , v_1v_5 , v_2v_3 and v_2v_4 , with weight 17. The edge v_4v_5 of weight 6 is an edge of $G_{11.10}$ of minimum weight not in T^* . Thus by [Theorem 11.14](#), an optimal hamiltonian cycle in $G_{11.10}$ has weight at least $17 + 6 = 23$.

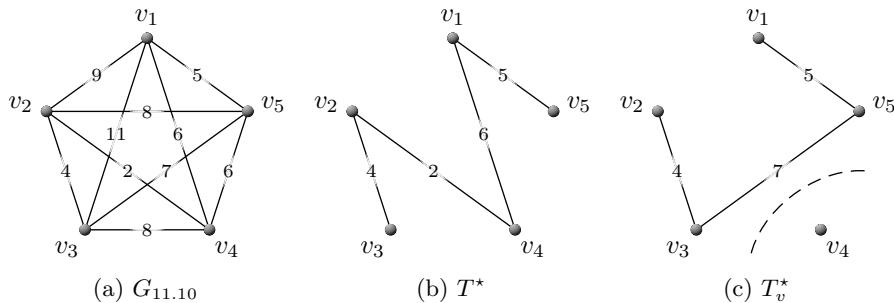


Figure 11.7: Illustration of the lower bound of [Theorem 11.14](#).

An alternative lower bound on the smallest weight of a hamiltonian cycle of a weighted complete graph G in terms of the weight of a minimum spanning tree of G is as follows.

Theorem 11.15 *Let G be a weighted complete graph of order $n \geq 3$ and let C^* be a hamiltonian cycle of minimum weight in G . Furthermore, let v be any vertex of G and let T_v^* be a minimum spanning tree of $G - v$. Then, $w(C^*) \geq w(T_v^*) +$*

$w(e) + w(f)$, where e and f are two distinct edges of smallest weight incident with v in G .

Proof Removing the vertex v from C^* produces a hamiltonian path $C^* - v$ in $G - v$. The path $C^* - v$ is therefore a spanning tree of $G - v$. Hence, the weight of $C^* - v$ is at least that of a minimum spanning tree of $G - v$, and so $w(C^* - v) \geq w(T_v^*)$. Let e_1 and e_2 be the two edges of C^* incident with v . Then, by definition of the edges e and f , $w(e_1) + w(e_2) \geq w(e) + w(f)$. Hence, $w(C^*) = w(C^* - v) + w(e_1) + w(e_2) \geq w(T_v^*) + w(e) + w(f)$. ■

To illustrate [Theorem 11.15](#), consider again the weighted complete graph $G_{11.10}$ shown in [Figure 11.7\(a\)](#). If we take $v = v_4$, then a minimum spanning tree in $G_{11.10} - v$ is the tree T_v^* (in [Figure 11.7\(c\)](#)) whose edges are v_1v_5 , v_2v_3 and v_3v_5 , with weight $w(T_v^*) = 16$. Two edges of smallest weight incident with v are $e = v_2v_4$ and $f = v_1v_4$ (or $f = v_4v_5$). By [Theorem 11.15](#), a minimum-weight hamiltonian cycle in $G_{11.10}$ has weight at least $w(T_v^*) + w(e) + w(f) = 16 + 2 + 6 = 24$. This lower bound of 24 is, in fact, the smallest weight of a hamiltonian cycle in $G_{11.10}$, realised by the cycle $v_1 v_5 v_3 v_2 v_4 v_1$. We remark that if we were to take $v = v_1$ in [Theorem 11.15](#), then we would obtain the same lower bound of 23 that we obtained by [Theorem 11.14](#).

❖ The reader should now be able to attempt Exercises [11.18–11.19](#).

11.4.3 Instances of the TSP satisfying the triangle inequality

In many applications of the TSP, a natural restriction on the edge weights of the TSP input graph, called the **triangle inequality**, is satisfied. This inequality is satisfied by the edge weights of a graph G if $w(uv) + w(vw) \geq w(uw)$ for any triple u, v, w of vertices in G . For a graph model G of a TSP instance satisfying the triangle inequality, it is possible to establish an approximation algorithm that is able to determine an approximate TSP solution (very quickly) which has weight no more than double that of a hamiltonian cycle of minimum weight in G , as we demonstrate in [Algorithm 27](#).

In [Step 1](#) of the algorithm, we can use [Kruskal's algorithm](#) to find a minimum spanning tree T of G in $\mathcal{O}(n^2 \log n)$ time. Note that the resulting tree T has size $n - 1$. In [Step 2](#), G^* has $2(n - 1)$ edges and n vertices. [Step 2](#) can therefore be completed in $\mathcal{O}(n)$ time. In [Step 3](#), we can use [Fleury's algorithm](#) to find an eulerian circuit T^* of G^* in $\mathcal{O}(n^2)$ time. In [Step 4](#), note that by the triangle equality we have $w(v_iv_k) \leq w(v_iv_j) + w(v_jv_k)$, implying that the new eulerian circuit we obtain has weight at most that of the old eulerian circuit. [Step 4](#) can be completed in $\mathcal{O}(n^2)$ time. In [Step 8](#), note that C has n edges and n vertices, and is a hamiltonian cycle of G . The entire worst-case time complexity of [Algorithm 27](#) is therefore dominated by that of [Step 1](#), which is $\mathcal{O}(n^2 \log n)$.

To illustrate the working of [Algorithm 27](#), consider the weighted complete graph $G_{11.11}$ of order $n = 5$ shown in [Figure 11.8](#). Since $G_{11.11}$ satisfies the triangle inequality, we can apply [Algorithm 27](#). In [Step 1](#), we use [Kruskal's algorithm](#) to find the minimum spanning tree T of $G_{11.11}$ with edges v_1v_5 , v_2v_3 , v_2v_4 and v_4v_5 , which has weight 17 (see [Figure 11.9\(a\)](#)). In [Step 2](#), we duplicate the edges of T to produce the eulerian multigraph $G_{11.11}^*$ shown in [Figure 11.9\(b\)](#), where $w(e_1) = w(e_2) = 5$, $w(e_3) = w(e_4) = 6$, $w(e_5) = w(e_6) = 2$, and $w(e_7) = w(e_8) = 4$.

Algorithm 27: Hamilton cycle

Input : A weighted complete graph G on $n \geq 3$ vertices whose edge weights satisfy the triangle inequality.

Output : A hamiltonian cycle in G whose weight is “close” to that of a hamiltonian cycle of minimum weight.

- 1 Find a minimum spanning tree T of G
- 2 Duplicate the edges of T to produce an eulerian multigraph G^*
- 3 Find an eulerian circuit T_0^* in G^*
- 4 Successively reduce the number of edges in T^* until only n edges remain in the following way:
 - 5 Let v_i, v_j, v_k be three consecutive vertices in T^* where v_j is the first repeated vertex in T^*
 - 6 **if** $v_i = v_k$ **then** replace v_i, v_j, v_k in T^* by v_i
 - 7 **if** $v_i \neq v_k$ **then** replace v_i, v_j, v_k in T^* by v_i, v_k
- 8 **print** T^*

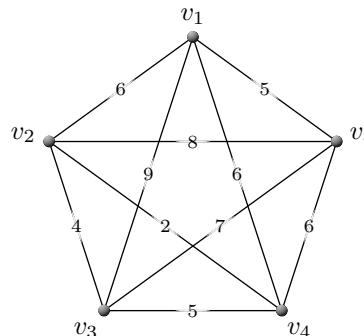


Figure 11.8: A graph $G_{11.11}$ used to illustrate the working of Algorithm 27.

In Step 3 of Algorithm 27, we use Fleury’s algorithm (Algorithm 24) to find an eulerian circuit T^* in $G_{11.11}^*$. There are many such eulerian circuits. Let us, for example, take T^* to be $v_3 v_2 v_4 v_2 v_1 v_5 v_1 v_2 v_3$.

Since T^* has $8 > 5 = n$ edges, we enter the loop spanning Steps 4–7 of Algorithm 27. The vertex v_2 is the first repeated vertex in T^* . Applying Step 7 of the algorithm to the underlined portion of T^* above, we obtain the updated tour $v_3 v_2 v_4 v_1 \underline{v_5 v_1 v_2} v_3$.

T^* now has $7 > 5 = n$ edges, and so we enter the loop spanning Steps 4–7 of Algorithm 27 again. The vertex v_1 is now the first repeated vertex in T^* . Applying Step 7 of the algorithm to the underlined portion of T^* above, we obtain the updated tour $v_3 v_2 v_4 v_1 \underline{v_5 v_2} v_3$.

Now T^* has $6 > 5 = n$ edges, and so we enter the loop spanning Steps 4–7 of Algorithm 27 yet again. The vertex v_2 is now again the first repeated vertex in T^* . Applying Step 7 of the algorithm to the underlined portion of T^* above, we obtain the updated tour $T^* : v_3 v_2 v_4 v_1 v_5 v_3$.

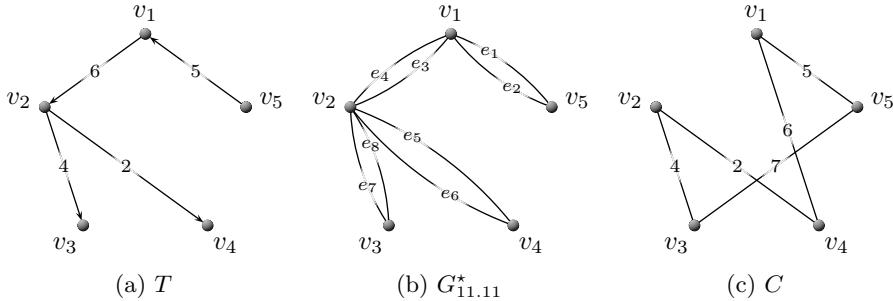


Figure 11.9: Finding a hamiltonian cycle C of “small” weight in the graph $G_{11.11}$ of Figure 11.8.

Since T^* now has $5 = n$ edges, the algorithm terminates after returning T^* as hamiltonian cycle of $G_{11.11}$. This hamiltonian cycle has weight 24 (and is, by accident, an optimal solution to the related TSP instance).

We close this section by showing that Algorithm 27 produces a hamiltonian cycle of G of weight at most twice that of a hamiltonian cycle of minimum weight.

Theorem 11.16 *The subgraph produced by Algorithm 27 is a hamiltonian cycle of G of weight at most twice that of a hamiltonian cycle of minimum weight in G .*

Proof By construction, the subgraph, C (say), produced by Algorithm 27 has n edges and n vertices, and has no repeated vertex, except for the initial and final vertex of C . Thus, C is a hamiltonian cycle of G . It remains to be shown that $w(C) \leq 2w(C^*)$, where C^* is a hamiltonian cycle of minimum weight in G . By the triangle equality, the weight of the eulerian circuit C produced in Step 8 is at most the weight of the initial eulerian circuit T_0^* found in Step 3. By construction, $w(T_0^*) = 2w(T)$, where T is the minimum spanning tree of G found in Step 1. Hence, $w(C) \leq 2w(T)$. By Theorem 11.14, a hamiltonian cycle of minimum weight in G is bounded from below by the weight of a minimum spanning tree of G . Hence, $w(T) \leq w(C^*)$. Consequently, $w(C) \leq 2w(C^*)$, as claimed. ■

❖ The reader should now be able to attempt Exercise 11.20.

11.4.4 Christofides’ algorithm

Algorithm 27, which employs a minimum spanning tree to determine a hamiltonian cycle of “small” weight in a weighted complete graph whose weights satisfy the triangle inequality, was discovered independently by many authors. In this case, where the triangle inequality is satisfied, however, there is a constant-factor approximation algorithm due to Christofides [4] which always finds a hamiltonian cycle of weight at most $\frac{3}{2}$ times the weight of a shortest hamiltonian cycle.

Christofides’ algorithm follows a similar outline to that presented in Algorithm 27. The only difference is that in Step 2 of Algorithm 27, rather than duplicating every edge of the minimum spanning tree T found in Step 1 in order to obtain an eulerian multigraph, it suffices to duplicate edges to pair up vertices that have odd degree in T (observe that T has an even number of odd vertices by Corollary 1.2). The resulting multigraph G^* obtained in this way is again eu-

Nicos Christofides was born in Cyprus in 1942. In 1966, he obtained a doctorate in electrical engineering at Imperial College, London based on a dissertation entitled *The origin of load losses in induction motors with cast aluminum rotors*. In 1984, he became professor of operations research and later of financial mathematics at Imperial College. His mathematical research focused on combinatorial optimisation, including scheduling and the analysis of traffic flows, as well as algorithms in graph theory. In graph theory, he is best known for his algorithm for finding approximate solutions to TSP instances where the distances form a metric space (they are symmetric and obey the triangle inequality). This approximation algorithm, which guarantees that its solutions will be within a factor of $3/2$ of the optimal solution weight, is to date the best approximation ratio that has been established for the TSP on general metric spaces. In 1990, he was co-founder and director of the Centre for Quantitative Finance (now the Institute for Financial Engineering), which is fully funded by major international banks and finance departments of large companies. He focused in particular on fund management, risk management and real options analysis of companies. He also led the company Financial Engineering Ltd in London, which deals with automated trading and risk management for hedge funds. He died in October 2019.



Biographic note 33: Nicos Christofides (1942–2019)

lerian. Thereafter, the remaining steps, namely Steps 3, 4 and 8 of [Algorithm 27](#), are applied as before.

Note that if M denotes a matching on the vertices that have odd degree in T , then the resulting hamiltonian cycle C in G obtained by means of [Christofides' algorithm](#) has weight at most the weight of the minimum spanning tree T added to the sum of the weights of edges in M ; that is, $w(C) \leq w(T) + w(M)$. It suffices to show that there is such a matching M with $w(M) \leq \frac{1}{2}w(C^*)$, where C^* is a hamiltonian cycle of minimum weight in G . This would imply, by [Theorem 11.14](#), that $w(C) \leq w(T) + w(M) \leq w(C^*) + \frac{1}{2}w(C^*) = \frac{3}{2}w(C^*)$. That this is indeed the case, follows from the next result.

Theorem 11.17 *If G is a weighted complete graph whose edge weights satisfy the triangle inequality and if V_G is a subset of vertices of G of even cardinality, then there exists a matching of the vertices in V_G such that the sum of the weights of the edges in this matching is at most half the weight of a shortest hamiltonian cycle in G .*

Proof Let C^* be a hamiltonian cycle of minimum weight in G . Suppose that v_1, \dots, v_{2k} are the $2k$ vertices in V_G . Renaming vertices if necessary, we may assume that the vertices v_1, \dots, v_{2k} appear in this order along the cycle C^* . Thus, our cycle C^* is labelled so that C^* starts at v_1 , and for $i \in [2k - 1]$, the cycle proceeds from v_i to v_{i+1} along a subpath that contains no vertices in V_G as internal vertices. For $i \in [k]$, let P_i be the corresponding $v_{2i-1}-v_{2i}$ subpath of C^* that contains no vertices in V_G as internal vertices and let E_P be the set of all edges of C^*

that belong to one of these paths P_i for some i . For $i \in [k-1]$, let Q_i be the corresponding $v_{2i}-v_{2i+1}$ subpath that contains no vertices in V_G as internal vertices and let Q_k be the $v_{2k}-v_1$ subpath that contains no vertices in V_G as internal vertices. Let E_Q be the set of all edges of C^* that belong to one of the paths Q_i for some i . Then, (E_P, E_Q) is a partition of the edge set $E(C^*)$ of C^* . Without loss of generality, we may assume that the sum of the weights of all edges in E_P is at most the sum of the weights of all edges in E_Q , that is, $w(E_P) \leq w(E_Q)$. Hence, $w(E_P) \leq \frac{1}{2}w(C^*)$. From the triangle inequality, it follows that the weight of the edge $v_{2i-1}v_{2i}$ is at most the sum of the weights of the edges on the subpath P_i that joins v_{2i-1} and v_{2i} . Hence, if we pair the vertices v_{2i-1} and v_{2i} for each $i \in [k]$, the sum of the weights of the edges in this matching is at most the sum of the weights of all edges in E_1 . Consequently, the sum of the weights of the edges in this matching is at most half the weight of an optimal hamiltonian cycle in G . ■

To illustrate Christofides' algorithm, consider again the weighted complete graph $G_{11.11}$ of order $n = 5$ shown in Figure 11.8. As before, let T be the minimum spanning tree of $G_{11.11}$ whose edges are v_1v_2 , v_1v_5 , v_2v_3 and v_2v_4 , with weight 17 (see Figure 11.9(a)). Since T contains four vertices of odd degree, namely v_2 , v_3 , v_4 and v_5 , we consider the three possible matchings M_1 , M_2 and M_3 on these vertices shown in Figure 11.10(a)–(c). Since $w(M_1) = 9$, $w(M_2) = 10$ and $w(M_3) = 13$, we duplicate the edges v_2v_4 and v_3v_5 in T to form the eulerian multigraph $G_{11.12}$ shown in Figure 11.10(d). Fleury's algorithm (Algorithm 24) may be used to find the eulerian circuit $v_1 v_2 \underline{v_4} v_2 v_3 v_5 v_1$ of $G_{11.12}$. Applying Step 7 of Algorithm 27 to the this circuit, the underlined portion of the circuit is replaced by $v_4 v_3$, resulting in the hamiltonian cycle $C : v_1 v_2 v_4 v_3 v_5 v_1$ of $G_{11.11}$. This hamiltonian cycle has weight 28. As remarked previously, a shortest hamiltonian cycle C^* of $G_{11.11}$ has weight 24. Note, therefore, that the performance bound $28 = w(C) \leq \frac{3}{2}w(C^*) = 36$ of Theorem 11.17 is indeed satisfied in this special case.

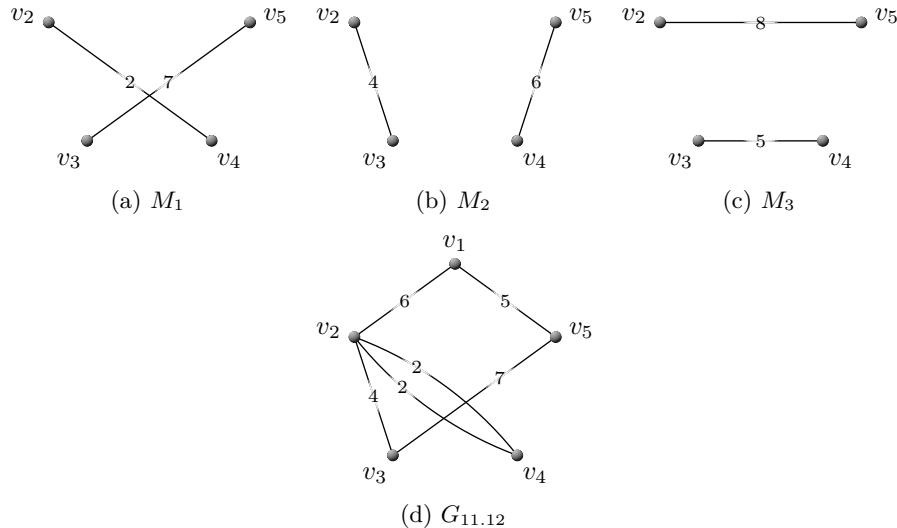


Figure 11.10: Matchings associated with an application of Theorem 11.17.

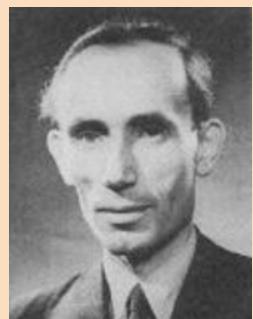
We close with the remark that a minimum-weight perfect matching of a complete weighted (sub)graph of even order can be found in polynomial time (see [Algorithm 23](#)). By [Theorem 11.17](#), such a matching has weight at most half the weight of a hamiltonian cycle of minimum weight in G . Thus [Christofides' algorithm](#) is an efficient algorithm that can quickly find a hamiltonian cycle whose weight is “close” to the weight of a shortest hamiltonian cycle. [Christofides' algorithm](#) was one of the first approximation algorithms, and drew much attention to approximation schemes as a practical approach towards attacking intractable problems. It remains an open problem, however, to improve the factor $\frac{3}{2}$ in the approximation guarantee of [Christofides' algorithm](#) to a smaller constant.

- ❖ The reader should now be able to attempt [Exercise 11.21](#).

11.5 Almost traceability and almost hamiltonicity

Thus far in this chapter we have considered only graphs that are traceable or hamiltonian. Graphs that are *almost* traceable or *almost* hamiltonian have, however, also been the subject of considerable study. In this section, we briefly outline the history of some intriguing questions about the existence of graphs with the property that removal of a single vertex renders them traceable or hamiltonian.

Tibor Gallai was born in Budapest, Hungary on 15 July 1912. He obtained his doctorate in mathematics from the Technical University of Budapest under the supervision of [Dénes König](#). Gallai held an academic position at the prestigious Eötvös Loránd University in Hungary. He was doctoral supervisor to the famous Hungarian mathematician, László Lovász. Gallai worked in combinatorics, especially in graph theory, and was a lifelong friend and collaborator of [Paul Erdős](#). Gallai played a key role in creating the Hungarian school in combinatorics and worked at the Institute of Applied Mathematics at the Hungarian Academy of Sciences between 1958 and 1968. He is known for, among others, the Sylvester-Gallai theorem in geometry and for the Edmonds-Gallai decomposition theorem, which was proved independently by Gallai and [Jack Edmonds](#), and describes finite graphs from the point of view of matchings. In 1991, he became a corresponding member of the Hungarian Academy of Sciences. He died in Budapest on 2 January 1992.



Biographic note 34: Tibor Gallai (1912–1992)

11.5.1 Hypotraceability of graphs

A graph G is **hypotraceable** if G does not possess a hamiltonian path, but deleting any vertex from G results in a graph that does contain a hamiltonian path. The existence of hypotraceable graphs is not an obvious question at all. In fact, the Hungarian mathematician [Tibor Gallai](#) conjectured in 1966 that hypotraceable

graphs do not exist [11]. This led to the publication of the following question in a 1969 edition of the American Mathematical Monthly by Hudson Kronk: *Does there exist a hypotraceable graph?* His own opinion on the answer to this question was as follows: “I feel very strongly that they do not exist and I hope some interested reader will provide a proof” [18].

In 1973, however, the Canadian mathematician Joseph Horton surprisingly answered this existence question in the affirmative by demonstrating that the graph of order 40 in Figure 11.11(a) is hypotraceable [16].

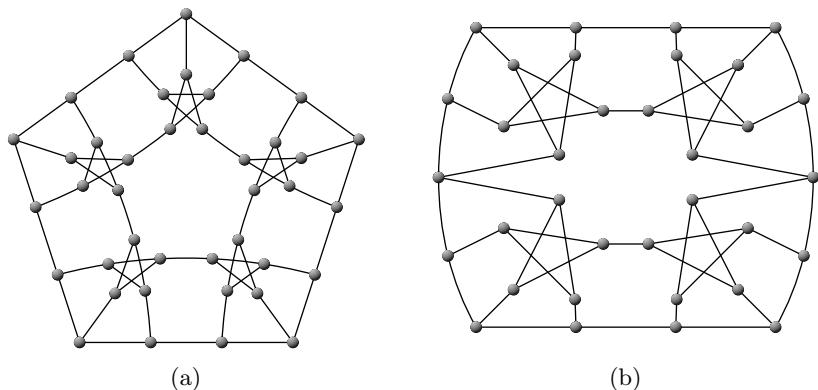
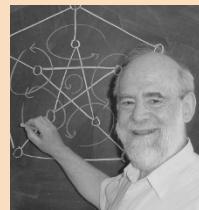


Figure 11.11: (a) A hypotraceable graph of order 40. (b) A hypotraceable graph of order 34.

Joseph Douglas Horton is professor of computer science at the University of New Brunswick, Canada. His original field of research was combinatorial designs. He found the first known nonhamiltonian bipartite cubic graph, which is now called the *Horton graph*, and he also found the first hypotraceable graph. His research interest has since shifted to algorithm analysis. In this area, he is best known for the discovery of a polynomial-time algorithm for finding the shortest cycle basis of a graph.



Biographic note 35: Joseph Horton (dates unknown)

This naturally led to the following follow-up question: *What is the smallest order for which there exists a hypotraceable graph?* This question has not yet been settled, but the smallest known hypotraceable graph has order 34 and is shown in Figure 11.11(b). This graph was constructed by the Danish mathematician Carsten Thomassen [21] who, generalising the construction, also established the following result.

Theorem 11.18 ([21]) *There exist infinitely many hypotraceable graphs.*

11.5.2 Hypohamiltonicity of graphs

A graph G is **hypohamiltonian** if G is nonhamiltonian, but deleting any vertex from G results in a hamiltonian graph. Existence questions similar to those raised about hypotraceable graphs have also been asked about hypohamiltonian graphs. The Czechoslovakian/Canadian mathematician Václav Chvátal, for instance, asked the following question in a 1973 issue of the Canadian Mathematical Bulletin [6]: *Does there exist a planar hypohamiltonian graph?* A graph is **planar** if it can be drawn in the plane without any edge crossings — we devote an entire chapter later in this book to the notion of graph planarity (see Chapter 13).

The Croatian/Brazilian mathematician Branko Grünbaum had reason to believe that the answer to the aforementioned question is negative, publishing the following conjecture one year later.

Conjecture 11.19 ([15]) *There are no planar hypohamiltonian graphs.*

This conjecture was, however, also proven to be incorrect by Thomassen who showed in 1974 that the planar graph of order 105 in Figure 11.12 is hypohamiltonian [21]. Thomassen, in fact, established the following more general existence result.

Theorem 11.20 ([21]) *There are infinitely many planar hypohamiltonian graphs.*

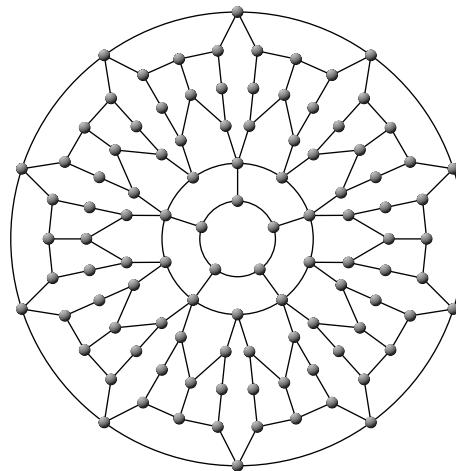


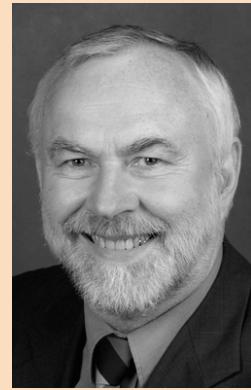
Figure 11.12: A planar hypohamiltonian graph of order 105.

The German mathematician Martin Grötschel [12] claimed that the properties of hypotraceable and hypohamiltonian graphs are partly to blame for the difficulty associated with solving instances of the TSP.

11.5.3 Hypotraceability and hypohamiltonicity of digraphs

The existence of hypotraceable and hypohamiltonian digraphs has also been considered. The following theorem was established independently by Thomassen, Grötschel, the Brazilian mathematician Yoshiko Wakabayashi and French mathematicians JL Fouquet and JL Jolivet.

Martin Grötschel was born in Schwelm, Germany on 10 September 1948. He obtained his doctorate in 1977 from the University of Bonn under the supervision of Bernhard Korte, and completed his habilitation at Bonn in operations research in 1981. One year later he was appointed professor of applied mathematics at the University of Augsburg. From 1991 until his retirement in 2015 he was professor of information technology at the Technical University, Berlin. From 1991 to 2012 he was Vice President of the *Zuse Institute Berlin* (ZIB) and served from 2012 to 2015 as ZIB's President. He is an internationally renowned expert in the field of combinatorial optimisation with a special emphasis on graph theory, linear and mixed-integer optimisation, and operations research. His publications together with Lovász and Schrijver on the ellipsoid method and its application in combinatorial and convex optimisation gained worldwide recognition, which earned them the Fulkerson Prize from the American Mathematical Society in 1982. In 2006, the same trio won the **John von Neumann Theory Prize** from the *Institute for Operations Research and the Management Sciences* (INFORMS). In 1991, Grötschel was the recipient of the **George B Dantzig Prize**, awarded by the *Society for Industrial and Applied Mathematics* (SIAM) and the Mathematical Optimisation Society.



Biographic note 36: Martin Grötschel (1948–present)

Theorem 11.21 ([10, 13, 21]) *There exists a hypohamiltonian digraph of order n if and only if $n \geq 6$.*



Figure 11.13: (a) A hypohamiltonian digraph of order 6. (b) A hypotraceable digraph of order 7 obtained from the digraph in (a) by applying Theorem 11.22.

The smallest hypohamiltonian digraph is shown in Figure 11.13(a). The following result relating hypotraceability and hypohamiltonicity of digraphs provides a useful construction mechanism.

Theorem 11.22 *A hypotraceable digraph of order $n+1$ may be obtained from any hypohamiltonian digraph G of order n by splitting a vertex of G into a source and a sink.*

Yoshiko Wakabayashi was born in Lavínia, Brazil on 21 May 1950. Her parents had immigrated from Japan. She obtained her master's degree in applied mathematics from the *University of São Paulo* (USP) under the guidance of Hungarian-born Imre Simon. In 1977, her master's thesis, titled *On hamiltonian graphs*, attracted the attention of [Martin Grötschel](#), who was visiting USP, and motivated their joint work on hypohamiltonian/hypotraceable digraphs and the TSP polytope. Later, in 1986, she obtained a doctorate from the University of Augsburg, Germany under the supervision of [Martin Grötschel](#), working in polyhedral combinatorics. She has been at USP since 1975, where she is currently professor of computer science. Her research interests include combinatorial optimisation and graph theory. More recently she has focused her research on approximation algorithms, and has made a number of contributions in the field of packing problems. In 2018, she was elected a member of the Brazilian Academy of Sciences.



Biographic note 37: Yoshiko Wakabayashi (1950–present)

The hypotraceable digraph of order 7 in [Figure 11.13\(b\)](#) may, for example, be obtained from the hypohamiltonian digraph of order 6 in [Figure 11.13\(a\)](#) by splitting the vertex y of the latter into a source x and a sink z in the former. We therefore have the following corollary of [Theorem 11.22](#).

Corollary 11.23 *There exist hypotraceable digraphs of order n for all $n \geq 7$.*

[Grötschel and Wakabayashi](#) [14] described how the properties of hypotraceable and hypohamiltonian digraphs contribute to the complexity of the asymmetric TSP⁴.

❖ The reader should now be able to attempt [Exercise 11.22](#).

Exercises

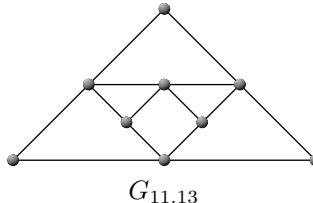
- 11.1 Solve Hamilton's puzzle *A Voyage Round the World* by finding a hamiltonian cycle in the dodecahedron graph of [Figure 11.1\(b\)](#).
- 11.2 Prove that it is not possible for a knight to tour a 4×4 chessboard, visit each square exactly once, and return to the initial square.
- 11.3 Prove that if n is odd, then it is not possible for a knight to tour an $n \times n$ chessboard, visit each square exactly once, and return to the initial square.
- 11.4 Prove that the complete bipartite graph $K_{n,n+1}$ is nonhamiltonian for every positive integer n .

⁴The problem of finding a (directed) hamiltonian cycle of minimum weight in a weighted, complete digraph. In contrast, the symmetric travelling salesman problem is the problem, considered in [Section 11.4](#), of finding an undirected hamiltonian cycle of minimum weight in a weighted, complete graph which is not directed.

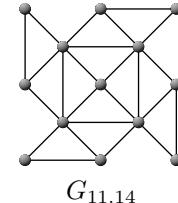
11.5 Use Exercise 11.3 to show that lowering the bound in Theorem 11.2 from $\frac{n}{2}$ to $\frac{n}{2} - 1$ will not guarantee that the graph G is hamiltonian (and thus, that the bound of $\frac{n}{2}$ in Theorem 11.2 is best possible.)

11.6 Show that if G is a hamiltonian bipartite graph with partite sets V_1 and V_2 , then $|V_1| = |V_2|$.

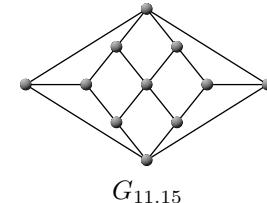
11.7 Decide whether each of the following graphs has a hamiltonian cycle, a hamiltonian path, or neither:



$G_{11.13}$



$G_{11.14}$



$G_{11.15}$

11.8 Prove that the Petersen graph $G_{11.2}$ in Figure 11.3(b) is nonhamiltonian.

11.9 Show that the bound presented in Theorem 11.4 is sharp by finding a class of nonhamiltonian bipartite graphs G with partite sets V_1 and V_2 such that $|V_1| = |V_2| = n \geq 2$ and $\delta(G) \geq \frac{n}{2}$.

11.10 Suppose a mouse eats its way through a $3 \times 3 \times 3$ cube of cheese, tunnelling through all 27 of the $1 \times 1 \times 1$ cubes. If the mouse starts at a corner, can it finish in the centre?

11.11 Let u and v be distinct nonadjacent vertices of a graph G of order $n \geq 3$ such that $d(u) + d(v) \geq n - 1$. Prove or disprove the following statement: The graph G is traceable if and only if $G + uv$ is traceable.

11.12 Show that if a graph of order at least 3 has an isolated vertex or an end-vertex, then its closure is not complete.

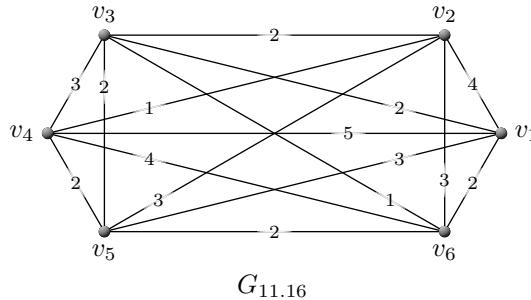
11.13 Show that the bound presented in Theorem 11.9 is best possible, by finding a class of nonhamiltonian graphs of order $n \geq 3$ such that $d(u) + d(v) \geq n - 1$ for all pairs u, v of nonadjacent vertices of G .

11.14 Show that the bound presented in Corollary 11.10 is best possible, by finding a nonhamiltonian graph of order $n \geq 3$ with $\binom{n-1}{2} + 1$ edges.

11.15 Let G be a bipartite graph with partite sets V_1 and V_2 such that $|V_1| = |V_2| = n \geq 2$. Let u and v be nonadjacent vertices of G with $u \in V_1$ and $v \in V_2$ such that $d(u) + d(v) > n$. Show that G is hamiltonian if and only if $G + uv$ is hamiltonian.

11.16 Define the notion of the **bipartite closure** of a bipartite graph with partite sets V_1 and V_2 such that $|V_1| = |V_2| = n \geq 2$. Show that the bipartite closure of a given bipartite graph is unique.

- 11.17 Use the [nearest-neighbour heuristic](#) to find an approximate solution to the TSP for the graph $G_{11.16}$ below.



- 11.18 Use [Theorem 11.14](#) to find a lower bound on the weight of an optimal hamiltonian cycle for the graph $G_{11.16}$ in [Exercise 11.17](#).
- 11.19 Use [Theorem 11.15](#) to find a lower bound on the weight of an optimal hamiltonian cycle for the graph $G_{11.16}$ in [Exercise 11.17](#).
- 11.20 Use [Algorithm 27](#) to find an approximate solution to the TSP for the graph $G_{11.16}$ in [Exercise 11.17](#).
- 11.21 Use [Christofides' algorithm](#) to find an approximate solution to the TSP for the graph $G_{11.16}$ in [Exercise 11.17](#).
- 11.22 A hypohamiltonian digraph of order 7 is shown in [Figure 11.14](#). Consider this digraph as inspiration and produce hypohamiltonian digraphs of orders 7 and 11. (Hint: The number 3 seems to be an important quantity in the digraph of [Figure 11.14](#) in the sense that there is a triangle surrounding the central vertex and, apart from this central vertex, there are three vertices not on the aforementioned triangle. Try constructions in which this quantity is increased to 4 and 5.)

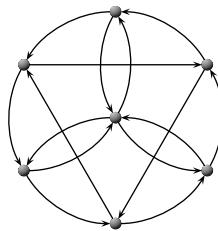


Figure 11.14: A hypohamiltonian digraph of order 7.

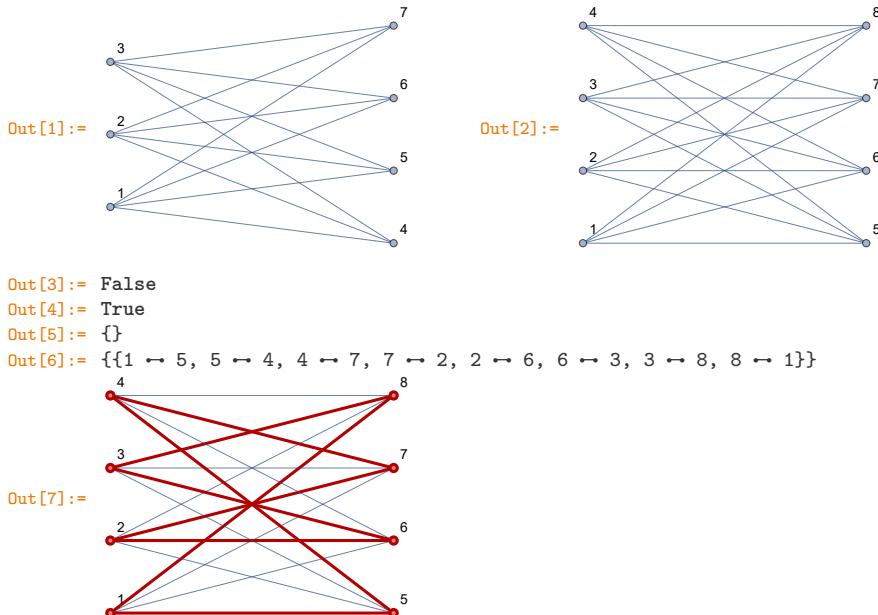
Computer exercises

The [MATHEMATICA](#) command `HamiltonianGraphQ[G]` yields the boolean value **True** if the graph G is hamiltonian, or the boolean value **False** otherwise. Furthermore, the command `FindHamiltonianCycle[G]` returns a hamiltonian cycle

of G if G is a hamiltonian graph, or the empty cycle ($\{\}$) otherwise. For example, the commands

```
In[1]:= G1 = CompleteGraph[{3, 4}, VertexLabels -> "Name"]
In[2]:= G2 = CompleteGraph[{4, 4}, VertexLabels -> "Name"]
In[3]:= HamiltonianGraphQ[G1]
In[4]:= HamiltonianGraphQ[G2]
In[5]:= FindHamiltonianCycle[G1]
In[6]:= Cy = FindHamiltonianCycle[G2]
In[7]:= HighlightGraph[G2, GraphHighlight[cy[[1]]], GraphHighlightStyle -> "Thick"]
```

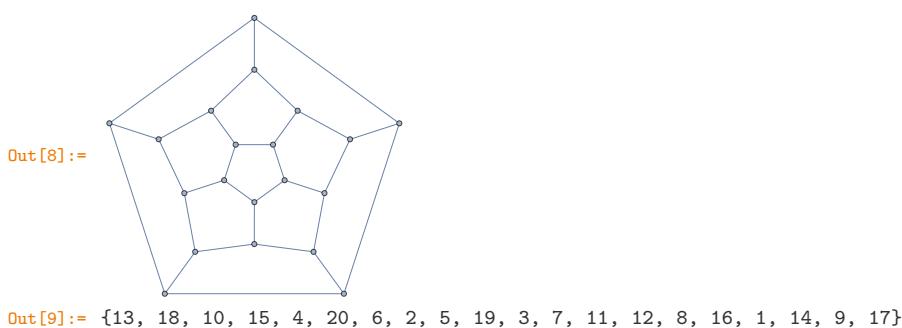
produce the output:

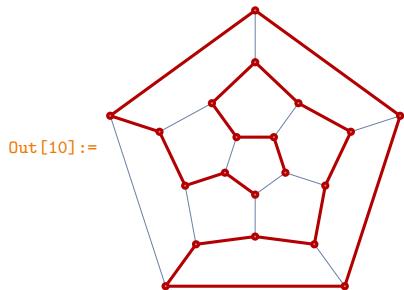


The command `FindHamiltonianPath[G]` similarly returns a hamiltonian path in G if G is a traceable graph, or the empty path ($\{\}$) otherwise. For example, the commands

```
In[8]:= G3 = PolyhedronData["Dodecahedron", "SkeletonGraph"]
In[9]:= P = FindHamiltonianPath[G3]
In[10]:= HighlightGraph[G3, PathGraph[P], GraphHighlightStyle -> "Thick"]
```

produce the output:

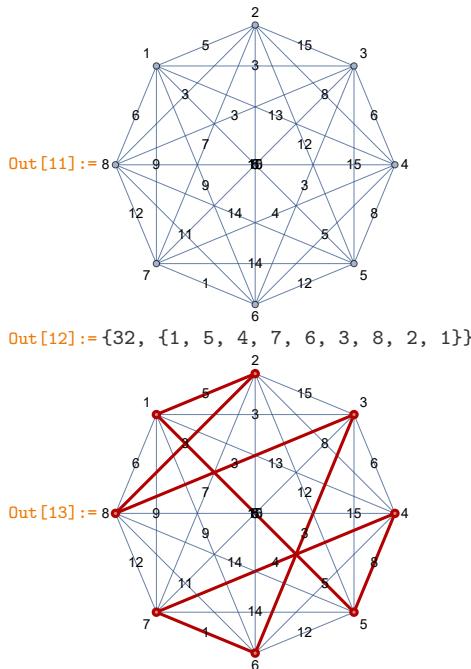




Finally, the **MATHEMATICA** command `FindShortestTour[G]` may be used to find the weight of a minimum-weight hamiltonian cycle together with such an optimal cycle itself in a hamiltonian graph G . For example, the commands

```
In[11]:= G4 = CompleteGraph[8, EdgeWeight -> RandomInteger[{1, 15}, Binomial[8, 2]], EdgeLabels -> "EdgeWeight", VertexLabels -> "Name"]
In[12]:= Tour = FindShortestTour[G4]
In[13]:= HighlightGraph[G4, PathGraph[Tour[[2]]], GraphHighlightStyle -> "Thick"]
```

may produce the output:



- ❖ The reader should now be able to re-attempt Exercises 11.7 and 11.8.

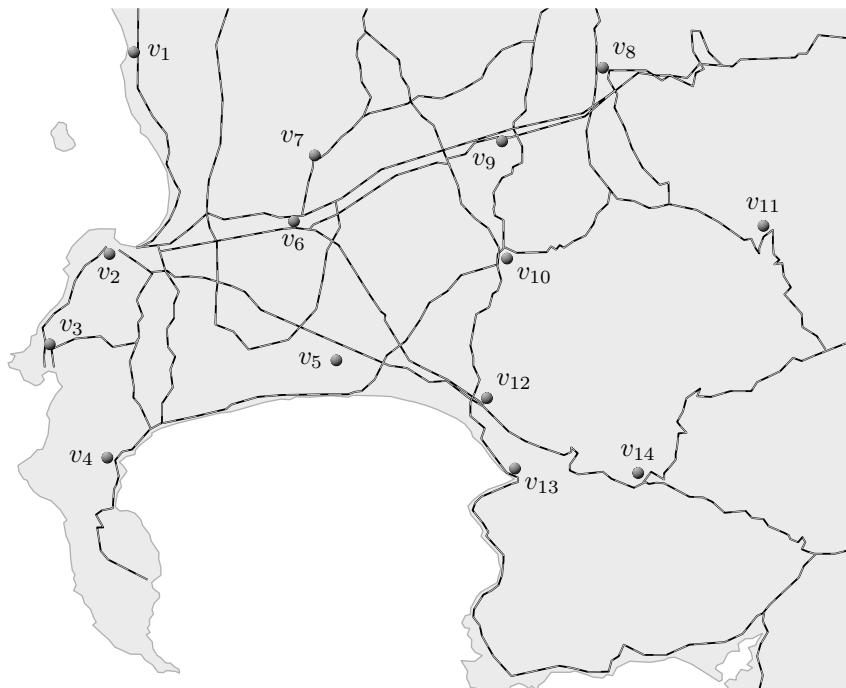
Projects

This section contains two projects. The first project is devoted to an extension of the TSP called the *capacitated vehicle routing problem*. In particular, it is shown how many logistics transportation problems resemble the TSP, but involve multiple delivery vehicles instead of a single salesman vehicle as is the case in the TSP. The

reader is then led in a step-by-step fashion to solve an instance of the CVRP approximately by applying a well-known heuristic called the savings method. The purpose of the second project is to showcase perhaps an unexpected application of the TSP within the realm of machine scheduling. More specifically, the reader is led in a step-by-step fashion to model (and solve approximately) the sequencing of a collection of jobs to be performed on a machine as an instance of the TSP.

Project 11.1: The capacitated vehicle routing problem

A certain retailer has outlets at thirteen locations (locations 1–5 and 7–14) in an area around the Cape Metropole in South Africa, as shown in [Figure 11.15](#). The retailer's depot is located in Bellville (location 6). Stock has to be delivered from the depot to all the outlets by a single vehicle.



City	City		
v_1	Melkbosstrand	v_8	Paarl
v_2	Cape Town	v_9	Klapmuts
v_3	Hout Bay	v_{10}	Stellenbosch
v_4	Fish Hoek	v_{11}	Franschhoek
v_5	Khayelitsha	v_{12}	Somerset West
v_6	Bellville	v_{13}	Gordons Bay
v_7	Durbanville	v_{14}	Grabouw

Figure 11.15: A small retail outlet network in an area around the Cape Metropole in South Africa.

The distances between the fourteen locations in Figure 11.15 are given (in kilometres) in the following distance matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
v_1	0	28	44	57	45	31	22	57	43	48	78	64	73	84
v_2	28	0	14	27	28	26	31	71	54	49	80	52	60	73
v_3	44	14	0	18	29	37	46	84	77	58	88	56	62	77
v_4	57	27	18	0	30	41	53	87	69	78	85	48	55	60
v_5	45	28	29	30	0	15	29	56	40	27	57	24	32	45
v_6	31	26	37	41	15	0	14	47	30	23	54	33	41	52
v_7	22	31	46	53	29	14	0	41	24	27	56	44	53	62
v_8	57	71	84	87	56	47	41	0	17	30	32	53	59	60
v_9	43	54	77	69	40	30	24	17	0	13	34	42	48	52
v_{10}	48	49	58	78	27	23	27	30	13	0	31	24	31	46
v_{11}	78	80	88	85	57	54	56	32	34	31	0	40	46	33
v_{12}	64	52	56	48	24	33	44	53	42	24	40	0	6	21
v_{13}	73	60	62	55	32	41	53	59	48	31	46	6	0	15
v_{14}	84	73	77	60	45	52	62	60	52	46	33	21	15	0

Tasks

1. Use the nearest neighbour heuristic (Algorithm 26) to determine a route for the stock delivery vehicle, starting out from the depot in Bellville, visiting the outlets at all other locations, and finally returning to Bellville.
2. Repeat Task 1, but use Algorithm 27 instead of the nearest neighbour heuristic.
3. Repeat Tasks 1 and 2, but use Christofides' algorithm instead of the nearest neighbour heuristic or Algorithm 27.

Consider now the more general situation where one vehicle can no longer visit all the outlets to deliver stock in a single trip as a result of limited loading space onboard for the transportation of stock. In particular, suppose a fleet of identical delivery vehicles is instead available for stock delivery, that each delivery vehicle has a transportation capacity of 120 cubic metres of retail stock, and that the delivery demand of stock at each outlet is as listed in Table 11.2.

Outlet	v_1	v_2	v_3	v_4	v_5	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
Demand	66	90	55	60	50	45	35	30	40	38	56	38	25

Table 11.2: Delivery demand of stock at each of the retail outlets in Figure 11.15.

The practical situation described above is an instance of a generalisation of the TSP, called the *capacitated vehicle routing problem* (CVRP). This problem was first introduced in a 1959 landmark paper by Dantzig and Ramser [8]. The CVRP may be stated as follows in general: A set of customers with known locations and demands for commodity delivery are to be supplied from a depot by a fleet of identical delivery vehicles, each with a known commodity loading capacity,

subject to the requirements that all customer demand is met, no vehicle capacity is exceeded, as few vehicles as possible are used (so as to minimise fixed vehicle maintenance costs), and the total trip distance of all vehicles is minimised (so as to minimise variable fuel costs) [22].

In 1964, Clarke and Wright [7] proposed a heuristic, called the *savings method*, for finding approximate solutions to the CVRP. According to this method, a savings value is calculated for every pair of customers. If there are two customers i and j at distances c_{id} and c_{dj} from the depot d , respectively, and at a distance c_{ij} from one another, then if commodity deliveries were to be made to these customers in separate trips from the depot, the distance travelled would be $2c_{id} + 2c_{dj}$ (assuming a symmetric distance matrix so that $c_{id} = c_{di}$ and $c_{jd} = c_{dj}$), as illustrated in Figure 11.16(a), while if the deliveries were to be combined into a single vehicle route, then the total distance would be $c_{id} + c_{ij} + c_{dj}$, as illustrated in Figure 11.16(b). Hence there would be a saving in distance of $s_{ij} = c_{id} + c_{dj} - c_{ij}$ if the deliveries were to be combined rather than carried out separately.

The savings thus calculated are ranked and customers are iteratively placed on routes into which they are linked in single vehicle routes in this order (otherwise a new route is started), until the vehicle capacity is reached. The process of adding a new route to an existing route is illustrated in Figure 11.16(c).

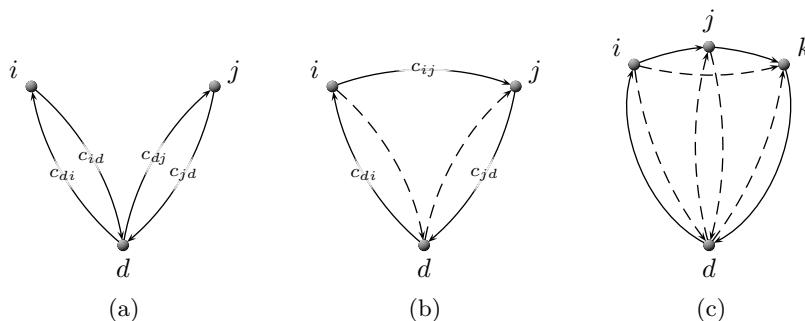


Figure 11.16: Working of the savings method. (a) The cost of visiting customers i and j separately. (b) The saving achieved when merging deliveries to customers i and j into one route. (c) Adding another customer, k , to a combined route for customers i and j .

4. Use the savings method to find an approximate solution to the retail stock delivery problem instance described above. How many vehicles are required to implement this delivery vehicle fleet routing? Do think your solution is optimal? Motivate.

Project 11.2: Print job scheduling

Consider the problem of sequencing n print jobs with variable setup times on a single industrial colour printer, called the *colour print scheduling problem* (CPSP) [3]. The setup time incurred between two consecutive print jobs consists of washing out the ink cartridges in the printer's magazine corresponding to colours not required for the later of the two jobs in the printing sequence so as to free up space

in the magazine for new colours that are indeed required for that job. If, however, some of the colours of two consecutive jobs in the printing sequence match, then the ink cartridges containing the matching colours need not be washed out between processing of the two jobs, thus reducing the setup time associated with the printing sequence. The duration of the setup time between two jobs is therefore job sequence-dependent.

Since it typically takes a constant time to wash out a single ink cartridge and the objective is to sequence the jobs in such a manner that the total wash-out time is minimised, it suffices to sequence the jobs so that the total number of ink cartridges washed out, called the *number of washes*, is minimised over the entire scheduling period.

Consider, for example, the $n = 12$ jobs in [Table 11.3](#) involving colours from the following colour set: 1 (black), 2 (blue), 3 (brown), 4 (cyan), 5 (gold), 6 (green), 7 (grey), 8 (magenta), 9 (orange), a (pink), b (red), c (silver), d (white) and e (yellow). In particular, job 1 requires colours 2, 3, 9 and d, while job 2 requires colours 2, 5, 7 and c. These two jobs therefore have the colour 2 in common.

Job	1	2	3	4	5	6
Colours	239d	257c	36ab	36bc	46ad	4cde
Job	7	8	9	10	11	12
Colours	124	16d	235	238	28e	5be

Table 11.3: Twelve jobs in an instance of the CPSP.

Since the printer starts out with all its ink cartridges empty and has to be returned to this state after all jobs have been printed, we include a dummy job, denoted by 0, corresponding to the all-cartridges-empty printer state. One possible printing sequence would then be

$$0 \ 1 \ 10 \ 9 \ 11 \ 12 \ 2 \ 7 \ 6 \ 8 \ 5 \ 3 \ 4 \ 0. \quad (11.1)$$

This sequence would correspond to starting out with an empty printer, processing job 1 first, followed by job 10, followed by job 9, and so on, until job 4 is eventually processed last, after which all the printer's cartridges are washed out to obtain an empty printer again upon conclusion of the printing schedule. If the printing schedule in (11.1) is carried out on a printer with $m = 4$ cartridges, a total of 20 washes will have to be carried out, as detailed in [Table 11.4](#).

Tasks

1. Describe in detail how an instance of the CPSP described above can be transformed into an instance of the TSP by defining the distance metric

$$d_{ij} = m - \text{number of colours common to jobs } i \text{ and } j$$

between two jobs i and j , where m denotes the number of cartridges in the colour printer.

Job	Cartridges				Washes afterwards
	1	2	3	4	
0	—	—	—	—	0
1	2	3	9	d	1
10	2	3	8	d	1
9	2	<u>3</u>	8	5	1
11	2	e	<u>8</u>	5	1
12	2	<u>e</u>	b	5	2
2	2	<u>7</u>	c	5	2
7	2	<u>1</u>	c	4	2
6	d	<u>e</u>	<u>c</u>	4	2
8	d	<u>1</u>	6	4	1
5	<u>d</u>	a	6	<u>4</u>	2
3	3	<u>a</u>	6	b	1
4	3	<u>c</u>	<u>6</u>	b	4
0	—	—	—	—	—
Total no of washes				20	

Table 11.4: A total of 20 washes are incurred on a 4-cartridge printer by the CPSP sequence in (11.1). Colours typeset in boldface are actively required for the corresponding jobs, while colours in normal font remain inactive within cartridges. Underlined colours have to be washed out of cartridges *after* having completed the corresponding jobs.

2. Populate a distance matrix between the jobs in Table 11.3 based on the metric in Task 1 above and then use Algorithm 27 to find a sequence in which to print the jobs on a printer with $m = 4$ ink cartridges.
3. How many washes are required in the sequence you determined in Task 2 above? Do you think this job sequence is optimal? Motivate.
4. Repeat Task 2 again, but this time for a printer with $m = 6$ ink cartridges. Comment on the quality of the sequence you obtain in view of the fact that

0 10 11 9 2 12 6 4 3 5 1 8 7 0

is an optimal sequence (involving 15 washes) in this case.

Further reading

- [1] DL Applegate, RE Bixby, V Chvátal and WJ Cook, 2006. *The Traveling Salesman Problem: A Computational Study*, Second edition, Princeton University Press, Princeton (NJ).
- [2] JA Bondy and V Chvátal, 1976. *A method in graph theory*, Discrete Mathematics, **15**, pp. 111–136.
- [3] AP Burger, CG Jacobs, JH van Vuuren and SE Visagie, 2015. *Scheduling multi-colour print jobs with sequence-dependent setup times*, Journal of Scheduling, **18**, pp. 131–145.

- [4] N Christofides, 1976. *Worst-case analysis of a new heuristic for the travelling salesman problem*, Technical Report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburg (PA).
- [5] V Chvátal, 1972. *On Hamilton's ideals*, Journal of Combinatorial Theory, **12**, pp. 163–168.
- [6] V Chvátal, 1973. *Flips-flops in hypohamiltonian graphs*, Canadian Mathematical Bulletin, **16**(1), pp. 33–41.
- [7] G Clarke and JW Wright, 1964. *Scheduling of vehicles from a central depot to a number of delivery points*, Operations Research, **12**(4), pp. 568–581.
- [8] GB Dantzig and JH Ramser, 1959. *The truck dispatching problem*, Management Science, **6**(1), pp. 80–91.
- [9] GA Dirac, 1952. *Some theorems on abstract graphs*, Proceedings of the London Mathematical Society, **2**, pp. 69–81.
- [10] JL Fouquet and JL Jolivet, 1976. *Problèmes combinatoires et théorie des graphes*, Orsay, p. 438.
- [11] T Gallai, 1966. *Theory of graphs*, Proc. Tihany, p. 362.
- [12] M Grötschel, 1980. *On the monotone symmetric travelling salesman problem — Hypohamiltonian/hypotraceable graphs and facets*, Mathematics of Operations Research, **5**(2), pp. 285–292.
- [13] M Grötschel, C Thomassen and Y Wakabayashi, 1980. *Hypotraceable digraphs*, Journal of Graph Theory, **4**(4), pp. 377–381.
- [14] M Grötschel and Y Wakabayashi, 1981. *On the structure of the monotone asymmetric travelling salesman polytope I: Hypohamiltonian facets*, Discrete Mathematics, **34**(1), pp. 43–59.
- [15] B Grünbaum, 1974. *Vertices missed by longest paths or circuits*, Journal of Combinatorial Theory (Series A), **17**(1), pp. 31–38.
- [16] RK Guy, 1973. *Monthly research problems (1969–1973)*, American Mathematical Monthly, **80**(10), pp. 1120–1128.
- [17] M Hahsler and K Hornik, 2007. *TSP — Infrastructure for the Traveling Salesperson Problem*, Journal of Statistical Software, **23**(2), pp. 1–21.
- [18] HV Kron, 1969. *Does there exist a hypotraceable graph?*, American Mathematical Monthly, **76**(7), pp. 809–810.
- [19] Ø Ore, 1960. *Note on Hamilton circuits*, American Mathematical Monthly, **67**, p. 55.
- [20] L Pósa, 1962. *A theorem concerning Hamilton lines*, Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közlemény, **7**, pp. 225–226.
- [21] C Thomassen, 1974. *Hypohamiltonian and hypotraceable graphs*, Discrete Mathematics, **9**(1), pp. 91–96.
- [22] P Toth and D Vigo, 2014. *Vehicle Routing — Problems, Methods and Applications*, Second edition, Society for Industrial and Applied Mathematics, Philadelphia (PA).
- [23] WL Winston, 2004. *Operations Research — Applications and Algorithms*, Fourth edition, Brooks/Cole Cengage Learning, Belmont (CA).



Graph connectivity

Contents

12.1	Introduction	359
12.2	Cuts and separating sets	360
12.3	Blocks	361
12.4	Connectivity and edge connectivity	368
12.5	Menger's Theorem	370
12.6	Computing the connectivity of a graph	377
	Exercises	380
	Computer exercises	384
	Projects	386
	Further reading	392

12.1 Introduction

A communication network may be modelled by means of a graph in which the vertices represent entities that either communicate directly with other entities in the network or else merely serve as possible intermediate message relaying infrastructure (such as hand sets and routers in a telephone network or personal computers and modems in a local area computer network), and in which the edges represent data transportation infrastructure (such as telephone lines or fibre-optic cables) joining these entities, along which messages may be relayed.

Clearly, a minimum requirement of a functional communication network is that it should be connected, for otherwise there would be pairs of entities in the network that cannot communicate with each other. But typically much more than connectedness is desirable in communication networks. A characteristic of a good communication network is that it should be robust in terms of limited infrastructure failure or unavailability in the sense that it should be impossible to disconnect the graph into more than one component by the removal of a small number of vertices or edges.

Consider the three cubic graphs in Figure 12.1. Each of these graphs has order 10 and size 15, but clearly $G_{12.2}$ is more robust as a model of a communication network topology than $G_{12.1}$, because $G_{12.1}$ may be disconnected into a 4-cycle with a chord and a 5-cycle with two chords by the removal of the single vertex v_{10} , as illustrated in Figure 12.1(a), whereas $G_{12.2}$ cannot be disconnected by removing

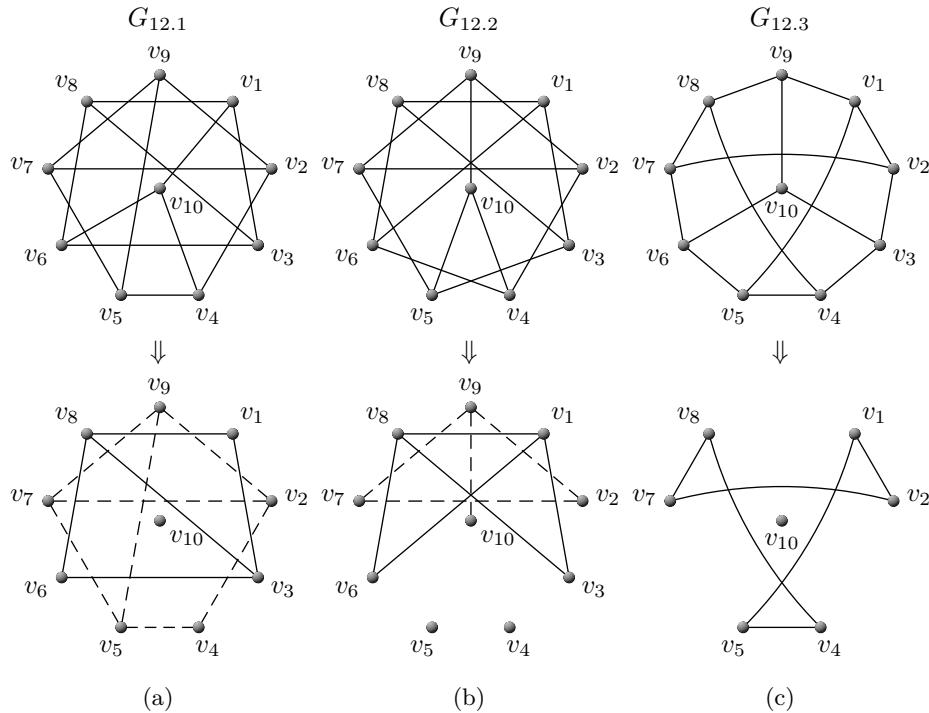


Figure 12.1: Three communication network topologies.

any single vertex (it can, however, be disconnected into a triangle with a pendent edge and a 4-cycle with a chord by the removal of the two vertices v_4 and v_5 , as illustrated in Figure 12.1(b)). The graph $G_{12.3}$, in turn, is even more robust as a model of a communication network topology than $G_{12.2}$, because $G_{12.3}$ cannot be disconnected by the removal of any two vertices (it is, however, possible to disconnect it into a singleton and a 6-cycle by removing the three vertices v_3 , v_6 and v_9 , as illustrated in Figure 12.1(c)). Similar observations may be made when focussing on the robustness of $G_{12.1}$, $G_{12.2}$ and $G_{12.3}$ in terms of edge removals rather than vertex removals.

In this chapter, we consider a number of aspects of the degree of robustness of a graph in terms of its ability to withstand edge or vertex removal before becoming disconnected.

12.2 Cuts and separating sets

A **vertex cut** of a connected graph G of order n is a subset X of the vertex set of G with the property that $G - X$ is disconnected (has more than one component). A vertex cut X is called a **k -vertex cut** if $|X| = k$ for some integer $k \leq n$. A vertex cut is sometimes also called a **vertex separating set** of G , because its removal separates or breaks G up into two or more components. Recall, from Chapter 2, that a vertex v of G is called a **cut-vertex** of G if $G - v$ is disconnected; v is therefore a cut-vertex of G if $\{v\}$ is a vertex cut of G . The sets $\{v_2\}$ and $\{v_3, v_5\}$ are examples of vertex cuts or vertex separating sets of the graph $G_{12.4}$ in Figure 12.2.

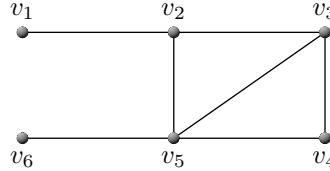


Figure 12.2: A connected graph $G_{12.4}$.

Let S be a nonempty subset of the vertex set $V(G)$ of a nontrivial, connected graph G of size m . Then the set $X'(S)$ of all edges of G which have one end vertex in S and the other end vertex in $V(G) \setminus S$ is called an **edge cut** of G . An edge cut $X'(S)$ is called a **k -edge cut** if $|X'(S)| = k$ for some integer $k \leq m$. Recall, again from [Chapter 2](#), that an edge e of G is called a **bridge** of G (also sometimes called a **cut-edge** of G) if $G - e$ is disconnected; e is therefore a bridge or cut-edge of G if $\{e\}$ is an edge cut of G . The sets $\{v_1v_2\}$, $\{v_2v_3, v_2v_5\}$ and $\{v_2v_3, v_3v_5, v_4v_5\}$ are examples of edge cuts of the graph $G_{12.4}$ in [Figure 12.2](#). For the edge cut $\{v_2v_3, v_2v_5\}$, for example, the set S may be taken as $\{v_1, v_2\}$.

A subset of the edge set of a connected graph G whose deletion from G results in a disconnected graph is called an **edge separating set** of G . Every edge cut is therefore an edge separating set, but note that the converse is not true — the entire edge set of a 3-cycle is an example of an edge separating set which is not an edge cut (why?). It is left as an exercise to show, however, that every *minimal* edge separating set of G is an edge cut of G if G is a graph of order $n > 1$.

Whereas there exist graphs in which every edge is a bridge (namely trees), our next result shows that there is no similar result for vertices.

Theorem 12.1 *Let G be a connected graph of order $n \geq 2$. Then G contains at least two vertices that are not cut-vertices.*

Proof If $n = 2$, then $G \cong K_2$ and so G has no cut-vertices. Suppose, therefore, that $n \geq 3$. Let u and v be two vertices furthest apart in G (*i.e.* for which $d_G(u, v)$ is a maximum). We show by contradiction that neither u nor v is a cut-vertex of G . Suppose, to the contrary, that u is a cut-vertex of G . Then $G - u$ contains at least two components. Let w be any vertex belonging to a component of $G - u$ not containing v . Then every $v-w$ path in G contains u , and so $d_G(v, w) > d_G(v, u)$, contradicting the choice of u and v . ■

❖ The reader should now be able to attempt Exercises [12.1–12.12](#).

12.3 Blocks

A **block** of a graph G is a maximal connected subgraph of G which has no cut vertex of its own. Note that a block of G may, however, contain cut vertices of G . If a connected graph contains a single block, we call the graph itself a **block**. The graph K_1 is called the **trivial block**. The graph $G_{12.5}$ in [Figure 12.3](#), for example, has the four blocks B_1 , B_2 , B_3 and B_4 , as indicated. The vertices v_2 , v_4 and v_6 are cut vertices of $G_{12.5}$, while v_2v_4 and v_6v_8 are bridges of $G_{12.5}$.

The next result shows that two blocks of a graph cannot overlap too much.

Theorem 12.2 *Any two blocks of a graph have at most one vertex in common, namely a cut-vertex.*

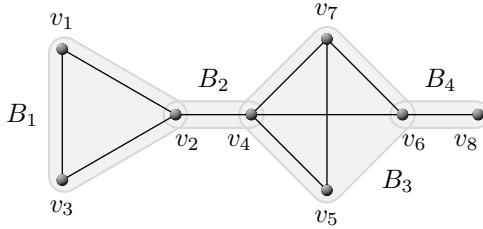


Figure 12.3: A graph $G_{12.5}$ containing four blocks B_1 , B_2 , B_3 and B_4 .

Proof By contradiction. Suppose B_1 and B_2 are two blocks of G which have two common vertices, u and v (say). Since deleting a single vertex from a block cannot disconnect it, there is a path in $B_i - u$ from every vertex in $B_i - u$ to v , for $i = 1, 2$. Therefore $(B_1 \cup B_2) - u$ is connected. But then $B_1 \cup B_2$ is a subgraph of G with no cut-vertex, contradicting the maximality of B_1 and B_2 . This contradiction shows that B_1 and B_2 have at most one vertex in common. If this vertex is not a cut-vertex, then $B_1 \cup B_2$ is again a subgraph of G with no cut-vertex, contradicting the maximality of B_1 and B_2 . ■

An immediate consequence of the above result is that every edge of a graph G lies in a unique block of G (*i.e.* the blocks of G partition its edge set) and that a graph may hence be viewed as the union of its blocks. Furthermore, the blocks of G also partition the set of all its vertices that are not cut-vertices.

A block containing exactly one cut-vertex is called an **end-block**. The blocks B_1 and B_4 are end-blocks of the graph $G_{12.5}$ in Figure 12.3.

We present two characterisations of blocks. The first characterisation is in terms of its vertices.

Theorem 12.3 *A connected graph G of order at least 3 is a block if and only if every two vertices of G lie on a common cycle of G .*

Proof Suppose every two vertices of G lie on a common cycle. Then G is connected. Let u , v and w be any three vertices of G and let C be a cycle containing u and w . The cycle C determines two distinct $u-w$ paths which have only the vertices u and w in common. Since v is distinct from u and w , it follows that v can lie on at most one of these paths. Hence, v does not belong to every $u-w$ path. Since u and w were chosen arbitrarily, it follows that v is not a cut-vertex of G . Hence, G cannot have a cut-vertex, and so G is a block. This proves the sufficiency.

Next we consider the necessity. Suppose $G = (V, E)$ is a block of order $n \geq 3$ and let u and v be arbitrary vertices of G . We show that u and v lie on a common cycle of G . Set $U_0 = \{u\}$ and for $i \in [e(u)]$, let $U_i = \{x \mid d_G(u, x) = i\}$; since G is connected, these sets partition V .

Suppose there exists a vertex that does not lie on a cycle containing u . Let k be the smallest subscript for which there is a vertex $y \in U_k$ that is not in a cycle with u . Since G is a block, no edge of G is a bridge. Thus, by Theorem 2.6, every edge of G lies on a cycle of G . In particular, every vertex in U_1 lies on a cycle of G containing u , and so $k \geq 2$.

By the definition of U_k there exists a $u-y$ path of length k . Let x be the vertex on this path that immediately precedes y . Then, $d_G(u, x) = k-1$, and so $x \in U_{k-1}$. By our choice of k , there exists a cycle C containing u and x . Since x is not a

cut-vertex of G , there exists a y - u path, Q say, in $G - x$. Let t (possibly $t = u$) be the first vertex in Q which is also in C . A cycle containing u and y can now be constructed by starting with the y - t subpath of Q from y to t , proceeding along C from t to x along the path that contains u , and finally taking the edge xy back to y . This produces a contradiction, implying that u is in a cycle with every other vertex in G . In particular, u and v lie on a common cycle. Since u and v were chosen arbitrarily, it follows that every two vertices of G lie on a common cycle. ■

For our next characterisation of blocks, we define, for any two edges e and f of a graph G , a binary relation, denoted by $e \sim f$, if either $e = f$ or if e and f lie on a common cycle of G . It may be shown that \sim is an equivalence relation on the edge set of G (see [Exercise 12.14](#)).

Theorem 12.4 *Let G be a connected graph of order at least 3. Then G is a block if and only if every two edges of G lie on a common cycle of G .*

Proof Suppose $G = (V, E)$ is a block. Let e and f be arbitrary distinct edges of G . Since G is connected there is a path P whose first edge is e and whose last edge is f . Suppose P is given by $u_1 u_2 \dots u_k$ for some $k \geq 3$. For $i \in [k-1]$, let $e_i = u_i u_{i+1}$. Then, $e = e_1$ and $f = e_{k-1}$. For $i = 2, \dots, k-1$, $G - u_i$ is connected since G is a block. Hence there is a $u_{i-1}u_{i+1}$ path in $G - u_i$. Adding the vertex u_i and the edges e_{i-1} and e_i to this path produces a cycle containing e_{i-1} and e_i . Hence, using the relation \sim defined above, $e_{i-1} \sim e_i$ for all $i = 2, \dots, k-1$. Since \sim is an equivalence relation on E , the edges e_1, e_2, \dots, e_{k-1} all belong to the same equivalence class and therefore any edges e_i and e_j lie on a common cycle of G for $1 \leq i < j \leq k-1$. In particular, e and f lie on a common cycle. This establishes the necessity.

To prove the sufficiency, suppose that every two edges of G lie on a common cycle of G . Let u and v be arbitrary distinct vertices of G . Let e and f be distinct edges incident with u and v , respectively. By assumption, e and f lie on a common cycle of G . Since this cycle contains u and v , it follows that every two vertices of G lie on a common cycle of G . Thus, by [Theorem 12.3](#), G is a block. ■

Recall, from [Project 5.5](#), that the cut-vertices and bridges of a connected graph of order m may be computed in $\mathcal{O}(m)$ time by means of a depth-first search. We close this section by showing that it is also possible to compute the blocks of a connected graph efficiently, by adapting the depth-first search algorithm in [Task 2\(ii\) of Project 5.5](#) slightly to obtain [Algorithm 28](#).

Steps 1–3 of [Algorithm 28](#) serve the purpose of parameter initialisation. The progress of the algorithm is measured by means of an iteration counter, denoted by i , which is initialised to take the value zero in [Step 1](#). The algorithm maintains a stack S of vertices, which is initialised as the empty stack in [Step 1](#). Each vertex is traversed at least once and each edge is traversed exactly once during execution of the algorithm. Two boolean arrays are maintained in order to keep track of which vertices and edges have been traversed. These boolean arrays are called `Stacked(•)` and `Scanned(•)`, respectively. These arrays are initialised in [Steps 2](#) and [3](#) to contain the value `false` for each vertex v and each edge e of G , respectively.

The input graph G with vertex set $\{v_1, \dots, v_n\}$ and edge set $\{e_1, \dots, e_m\}$ is traversed in a depth-first manner, keeping track of the depth-first index and the parent of each vertex v in the usual manner, denoted by $\text{DFI}(v)$ and $\text{Parent}(v)$,

Algorithm 28: The blocks of a connected graph

Input : A graph G of order n and size m with vertex set $V(G) = \{v_1, \dots, v_n\}$ and edge set $E(G)$.

Output : The blocks of G .

- 1 $i \leftarrow 0, S \leftarrow \emptyset$
- 2 **for all** $v \in V(G)$ **do** $\text{DFI}(v) \leftarrow 0, \text{Stacked}(v) \leftarrow \text{false}, \text{Parent}(v) \leftarrow \emptyset$
- 3 **for all** $e \in E(G)$ **do** $\text{Scanned}(e) \leftarrow \text{false}$
- 4 **if** vertices with zero depth-first index values remain **then**
- 5 **let** v be the smallest indexed vertex for which $\text{DFI}(v) = 0$
- 6 **else stop**
- 7 $i \leftarrow i + 1, \text{DFI}(v) \leftarrow i, \ell(v) \leftarrow i, \text{Stacked}(v) \leftarrow \text{true}$, add v to the top of S
- 8 **if** there is no unscanned edge incident with v **then go to Step 16**
- 9 **else**
- 10 **let** $u \leftarrow$ smallest indexed neighbour of v such that $\text{Scanned}(uv) = \text{false}$
- 11 $\text{Scanned}(uv) \leftarrow \text{true}$
- 12 **if** $\text{DFI}(u) > 0$ **then**
- 13 **go to Step 15**
- 14 $\text{Parent}(u) \leftarrow v, v \leftarrow u$, **go to Step 7**
- 15 $\ell(v) \leftarrow \min\{\ell(v), \text{DFI}(u)\}$, **go to Step 8**
- 16 **if** $\text{Parent}(v) = \text{Root}$ **then go to Step 21**
- 17 **if** $\ell(v) < \text{DFI}(\text{Parent}(v))$ **then**
- 18 $\ell(\text{Parent}(v)) \leftarrow \min\{\ell(v), \ell(\text{Parent}(v))\}$, **go to Step 20**
- 19 Remove all vertices from S down to, and including v ; together with $\text{Parent}(v)$ they induce a block of G
- 20 $v \leftarrow \text{Parent}(v)$, **go to Step 8**
- 21 Remove all vertices from S down to, and including v ; together with Root they induce a block of G
- 22 **if** there are no unscanned edges incident with Root **then go to Step 4**
- 23 **else go to Step 8**

respectively. These variables are initialised to respectively zero and the empty set in **Step 2** for every vertex v of G . During each iteration of the algorithm, a current vertex $v \in S$ is maintained and a next vertex u is sought such that u is the smallest indexed vertex of G for which vu is unscanned at that point. As soon as a vertex v is placed on the stack S , the value of $\text{Stacked}(v)$ is updated to **true**; this happens in **Step 7** of the algorithm. As soon as an edge e has been considered, this so-called scan of the edge is reflected by updating the value of $\text{Scanned}(e)$ to **true**; this happens in **Step 11**.

The traversal of G continues as long as there remain vertices with zero depth-first indices (**Step 5**) or unscanned edges (**Step 10**). As described in Project 5.5, the value of the lowpoint $\ell(v)$ of each vertex v is also maintained during execution of the algorithm. This variable is initialised as the iteration counter i in **Step 7**, is updated as necessary in Steps **15** and **18**, and facilitates the discovery of the blocks of G . As soon as a cut-vertex of G is discovered (and hence a block of G is determined), the stack S is cleared of the noncut vertices in the block; this happens in Steps **19** and **21**.

Let us illustrate the working of [Algorithm 28](#) by applying it to the graph $G_{12.5}$ in [Figure 12.3](#). The depth-first search tree associated with the order in which the algorithm traverses the vertices of $G_{12.5}$ is shown in [Figure 12.4](#). After the initialisation process in Steps 1–3, the vertex v_1 is selected in [Step 5](#), the iteration counter is incremented and the assignments in Steps 7–14 are made, as shown in the row of [Table 12.1](#) corresponding to the value $i = 1$, by scanning along the edge v_1v_2 of $G_{12.5}$.

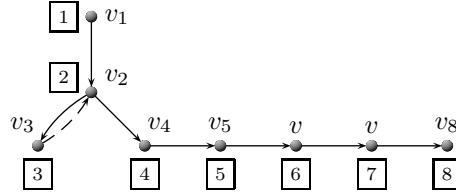


Figure 12.4: The depth-first search tree associated with [Table 12.1](#), as produced by [Algorithm 28](#) when applied to the graph $G_{12.5}$ in [Figure 12.3](#).

i	v	$\text{DFI}(v)$	$\ell(v)$	S	u	$\text{Parent}(u)/\text{Output}$
Root	v_1	—	—	—	—	—
1	v_1	1	1	$\{v_1\}$	v_2	v_1
2	v_2	2	2	$\{v_2, v_1\}$	v_3	v_2
3	v_3	3	3	$\{v_3, v_2, v_1\}$	v_1	
			1			
	v_2		1		v_4	v_2
4	v_4	4	4	$\{v_4, v_3, v_2, v_1\}$	v_5	v_4
5	v_5	5	5	$\{v_5, v_4, v_3, v_2, v_1\}$	v_7	v_5
6	v_7	6	6	$\{v_7, v_5, v_4, v_3, v_2, v_1\}$	v_4	
			4		v_6	v_7
7	v_6	7	7	$\{v_6, v_7, v_5, v_4, v_3, v_2, v_1\}$	v_4	
			4		v_8	v_6
8	v_8	8	8	$\{v_8, v_6, v_7, v_5, v_4, v_3, v_2, v_1\}$		$G_{12.5}[\{v_6, v_8\}]$
	v_6		4			
	v_7		4			
	v_5			$\{v_4, v_3, v_2, v_1\}$		$G_{12.5}[\{v_4, v_5, v_6, v_7\}]$
	v_4			$\{v_3, v_2, v_1\}$		$G_{12.5}[\{v_2, v_4\}]$
	v_2			$\{v_1\}$		$G_{12.5}[\{v_1, v_2, v_3\}]$

Table 12.1: The four blocks $B_1 = G_{12.5}[\{v_1, v_2, v_3\}]$, $B_2 = G_{12.5}[\{v_2, v_4\}]$, $B_3 = G_{12.5}[\{v_4, v_5, v_6, v_7\}]$ and $B_4 = G_{12.5}[\{v_6, v_8\}]$ produced (in reversed order) as output by [Algorithm 28](#) when applied to the graph $G_{12.5}$ in [Figure 12.3](#).

Upon returning to [Step 7](#), the iteration counter is incremented again and the assignments in Steps 7–14 are made again, as shown in the row of [Table 12.1](#) corresponding to the value $i = 2$, by scanning along the edge v_2v_3 of $G_{12.5}$.

Returning to [Step 7](#) again, the iteration counter is incremented yet again and the assignments in Steps [7–11](#) are made, as shown in the first row of [Table 12.1](#) corresponding to the value $i = 3$, by scanning along the edge v_3v_1 of $G_{12.5}$. Since $\text{DFI}(u) = 1 > 0$ for $u = v_1$, however, the algorithm continues at [Step 15](#), where the label of $v = v_3$ is lowered from 3 to $\ell(v_3) = 1$, after which the algorithm returns to [Step 8](#). Since no unscanned edges of $G_{12.5}$ remain at $v = v_3$, the algorithm proceeds at [Step 16](#). Since $v_2 = \text{Parent}(v_3) \neq \text{Root} = v_1$ and $1 = \ell(v_3) < \text{DFI}(\text{Parent}(v_3)) = \text{DFI}(v_2) = 2$, the label of v_2 is updated from 2 to $\ell(v_2) = 1$ in [Step 18](#), as shown in the first row of [Table 12.1](#) corresponding to the value $i = 3$. Thereafter, the current vertex v becomes v_2 in [Step 20](#) and the algorithm returns to [Step 8](#). Since unscanned edges incident with v_2 remain, the assignments in Steps [7–14](#) are made, as shown in the last row of [Table 12.1](#) corresponding to the value $i = 3$, by scanning along the edge v_2v_4 of $G_{12.5}$.

Upon returning to [Step 7](#), the iteration counter is incremented and the assignments in Steps [7–14](#) are made, as shown in the row of [Table 12.1](#) corresponding to the value $i = 4$, by scanning along the edge v_4v_5 of $G_{12.5}$.

Returning to [Step 7](#) again, the iteration counter is incremented again and another set of assignments are made in Steps [7–14](#), as shown in the row of [Table 12.1](#) corresponding to the value $i = 5$, by scanning along the edge v_5v_7 of $G_{12.5}$.

When the algorithm returns to [Step 7](#) again, the iteration counter is incremented yet again and the assignments in Steps [7–11](#) are made, as shown in the first row of [Table 12.1](#) corresponding to the value $i = 6$, by scanning along the edge v_7v_4 of $G_{12.5}$. Since $\text{DFI}(u) = 4 > 0$ for $u = v_4$, however, the algorithm continues to [Step 15](#), where the label of $v = v_7$ is lowered from 6 to $\ell(v_7) = 4$, after which the algorithm returns to [Step 8](#). Since an unscanned edge of $G_{12.5}$ incident with v_7 remains, the vertex u becomes v_6 in [Step 10](#) and the assignments in Steps [11–14](#) are made, as shown in the second row of [Table 12.1](#) corresponding to the value $i = 6$, by scanning along the edge v_7v_6 of $G_{12.5}$.

Returning to [Step 7](#), the iteration counter is incremented again and another set of assignments are made in Steps [7–11](#), as shown in the first row of [Table 12.1](#) corresponding to the value $i = 7$, by scanning along the edge v_6v_4 of $G_{12.5}$. Since $\text{DFI}(u) = 4 > 0$ for $u = v_4$, however, the algorithm continues to [Step 15](#), where the label of $v = v_6$ is lowered from 7 to $\ell(v_6) = 4$, after which the algorithm returns to [Step 8](#). Since an unscanned edge of $G_{12.5}$ incident with v_6 remains, the vertex u becomes v_8 in [Step 10](#) and the assignments in Steps [11–14](#) are made, as shown in the second row of [Table 12.1](#) corresponding to the value $i = 7$, by scanning along the edge v_6v_8 of $G_{12.5}$.

Returning to [Step 7](#), the iteration counter is incremented again and another set of assignments are made in [Step 7](#), as shown in the first row of [Table 12.1](#) corresponding to the value $i = 8$. Now no unscanned edges of $G_{12.5}$ remain in [Step 8](#) and the algorithm continues to [Step 16](#). Since $v_6 = \text{Parent}(v_8) \neq \text{Root} = v_1$ and $8 = \ell(v_8) \not\prec \text{DFI}(\text{Parent}(v_8)) = \text{DFI}(v_6) = 7$, the vertex v_8 is removed from the stack S and the block $G_{12.5}[\text{Parent}(v_8), v_8] = G_{12.5}[v_6, v_8]$ is produced as output in [Step 19](#), as shown in the second row of [Table 12.1](#) corresponding to the value $i = 8$.

The vertex v becomes v_6 in [Step 20](#) and the algorithm returns to [Step 8](#). Since no unscanned edges of $G_{12.5}$ remain, the algorithm continues to [Step 16](#) where $v_7 = \text{Parent}(v_6) \neq \text{Root} = v_1$ (in [Step 16](#)) and $4 = \ell(v_6) < \text{DFI}(\text{Parent}(v_6)) =$

$\text{DFI}(v_7) = 6$ (in Step 18), and so the label of $\text{Parent}(v_6) = v_7$ is confirmed as $\ell(v_7) = 4$ in Step 18, as shown in the third row of Table 12.1 corresponding to the value $i = 8$. The vertex v now becomes $\text{Parent}(v_6) = v_7$ in Step 20 and the algorithm returns to Step 8 again. Since no unscanned edges of $G_{12.5}$ remain, the algorithm continues at Step 16 where $v_5 = \text{Parent}(v_7) \neq \text{Root} = v_1$ (in Step 16) and $4 = \ell(v_7) < \text{DFI}(\text{Parent}(v_7)) = \text{DFI}(v_5) = 5$ (in Step 18), and so the label of $\text{Parent}(v_7) = v_5$ is lowered from 5 to $\ell(v_5) = 4$ in Step 18, as shown in the fourth row of Table 12.1 corresponding to the value $i = 8$. The vertex v now becomes $\text{Parent}(v_7) = v_5$ in Step 20 and the algorithm returns to Step 8 again. Since no unscanned edges of $G_{12.5}$ remain, the algorithm continues to Step 16 where $v_4 = \text{Parent}(v_5) \neq \text{Root} = v_1$ (in Step 16) and $4 = \ell(v_5) < \text{DFI}(\text{Parent}(v_5)) = \text{DFI}(v_4) = 4$ (in Step 18). The vertices v_5, v_6 and v_7 are therefore removed from the stack S and the block $G_{12.5}[\text{Parent}(v_5), v_5, v_6, v_7] = G_{12.5}[v_4, v_5, v_6, v_7]$ is produced as output in Step 19, as shown in the fifth row of Table 12.1 corresponding to the value $i = 8$.

The vertex v now becomes $\text{Parent}(v_5) = v_4$ in Step 20 and the algorithm returns to Step 8 again. Since no unscanned edges of $G_{12.5}$ remain, the algorithm continues to Step 16 where $v_2 = \text{Parent}(v_4) \neq \text{Root} = v_1$ (in Step 16) and $4 = \ell(v_4) < \text{DFI}(\text{Parent}(v_4)) = \text{DFI}(v_2) = 2$ (in Step 18). The vertex v_4 is therefore removed from the stack S and the block $G_{12.5}[\text{Parent}(v_4), v_4] = G_{12.5}[v_2, v_4]$ is produced as output in Step 19, as shown in the penultimate row of Table 12.1.

The vertex v becomes $\text{Parent}(v_4) = v_2$ in Step 20 and the algorithm returns to Step 8. Since no unscanned edges of $G_{12.5}$ remain, the algorithm continues to Step 16 where indeed $v_1 = \text{Parent}(v_2) = \text{Root}$. The vertices v_2 and v_3 are removed from the stack S and the block $G_{12.5}[\text{Parent}(v_2), v_2, v_3] = G_{12.5}[v_1, v_2, v_3]$ is produced as output in Step 21, as shown in the last row of Table 12.1.

Since no unscanned edges of $G_{12.5}$ remain in Step 22, the algorithm returns to Step 5 and since all vertices of $G_{12.5}$ now have positive DFI-values, the algorithm finally terminates at Step 6, having produced exactly the four blocks shown in Figure 12.3.

The worst-case time complexity of Algorithm 28 is estimated in the final result of this section.

Theorem 12.5 *The worst-case time complexity of Algorithm 28 is $\mathcal{O}(m)$, where m is the size of the input graph.*

Proof The number of variable assignments in Steps 1, 2 and 3 are 2, 4 and 1, respectively. Since Step 1 is performed once, Step 2 is performed n times and Step 3 is performed m times, it follows that the time complexity of the initialisation Steps 1–3 is $\max\{\mathcal{O}(n), \mathcal{O}(m)\}$.

The if-statement in Step 5 requires at most n comparisons, while five basic operations are performed in Step 7 (four variable assignments and one stack insertion). The worst-case time complexity of Steps 5–7 is therefore $\mathcal{O}(n)$. Every edge incident with each vertex is considered exactly once in Steps 8–10. If there are no edges incident with a vertex, then the single operation is performed of determining that the adjacency list of that vertex is empty. The worst-case time complexity of Steps 8–10 is therefore $\max\{\mathcal{O}(n), \mathcal{O}(m)\}$. Furthermore, since the four basic operations (three variable assignments and one if-comparison) in Steps 11–14 are performed at most once for each edge of the input graph, the worst-case time complexity of Steps 11–14 is $\mathcal{O}(m)$.

The value of a lowpoint of a vertex is updated at most once in **Step 15** for each edge incident with that vertex, *i.e.* at most $\sum_{i=1}^n d_G(v_i) = 2m$ times by **Theorem 1.1**. **Step 15** therefore has a worst-case time complexity of $\mathcal{O}(m)$. Since only one **if**-test is performed in **Step 16**, that step contributes a time complexity of $\mathcal{O}(n)$ to the algorithm. A backtrack is performed at most once from each vertex in Steps **18–21**. These steps therefore also contribute a time complexity of $\mathcal{O}(n)$ to the algorithm. Each edge incident with the root is scanned at most once in Steps **22–23**; if there remain no unscanned edges, then a single basic operation (variable assignment) is made. The worst-case time complexity of Steps **22–23** is therefore $\mathcal{O}(m)$.

We conclude that the overall worst-case time complexity of **Algorithm 28** is $\max\{\mathcal{O}(n), \mathcal{O}(m)\}$, but since the input graph is connected, it holds that $n \leq m+1$, which means that $n = \mathcal{O}(m)$. ■

- ❖ The reader should now be able to attempt Exercises [12.13–12.16](#).

12.4 Connectivity and edge connectivity

The **connectivity**, denoted $\kappa(G)$, of a graph G is the minimum cardinality of a vertex cut of G if G is not a complete graph and $\kappa(G) = n - 1$ if G is the complete graph on $n \geq 2$ vertices. A graph G is **k -connected**, for some $k \in \mathbb{N}$, if $\kappa(G) \geq k$. Thus, $\kappa(G)$ is the smallest number of vertices whose deletion from G produces a disconnected graph or trivial graph.

The graph $G_{12.6}$ in [Figure 12.1\(a\)](#) is 1-connected, but not 2-connected, because it is connected and has the cut vertex v_{10} . This shows that $\kappa(G_{12.6}) = 1$. The graph $G_{12.7}$ in [Figure 12.1\(b\)](#), on the other hand, is both 1-connected and 2-connected, but not 3-connected. In fact, $\kappa(G_{12.7}) = 2$, because $G_{12.7}$ is connected and has no cut-vertices, but removal of the vertices v_4 and v_5 disconnects the graph. Finally, the graph $G_{12.8}$ in [Figure 12.1\(c\)](#) is 1-connected, 2-connected and 3-connected, but not 4-connected, because $G_{12.8}$ can be disconnected by the removal of the three vertices v_3 , v_6 and v_9 (but such a disconnection cannot be achieved via the removal of any pair of vertices from $G_{12.8}$). Therefore, $\kappa(G_{12.8}) = 3$.

A nontrivial graph has connectivity 0 if and only if it is disconnected. A graph is 1-connected if and only if it is nontrivial and connected. A graph is 2-connected if and only if it is a block of order at least 3. In general, a graph is k -connected if and only if the removal of fewer than k vertices produces neither a disconnected graph nor the trivial graph.

The notion of connectivity also has an edge counterpart. The **edge connectivity** of a graph G , denoted by $\lambda(G)$, is the minimum cardinality of an edge cut of G if G is nontrivial, while $\lambda(K_1) = 0$. A graph G is said to be **k -edge connected** for some $k \in \mathbb{N}$ if $\lambda(G) \geq k$. Thus, $\lambda(G)$ is the smallest number of edges whose deletion from G produces a disconnected graph or trivial graph. Hence, $\lambda(G) = 0$ if and only if G is disconnected or trivial. A graph is 1-edge connected if and only if it is connected and contains a bridge. Examples of 1-edge connected, 2-edge connected and 3-edge connected graphs are shown in [Figure 12.5](#).

The connectivity, edge connectivity and minimum degree of a graph are related by the following result due to [Whitney](#) [4].

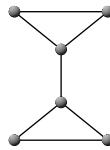
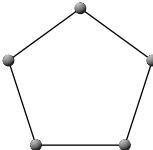
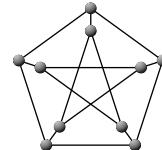
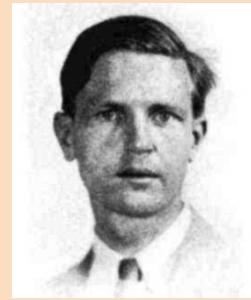
(a) $G_{12.6}$ (b) $G_{12.7}$ (c) $G_{12.8}$

Figure 12.5: (a) A graph $G_{12.6}$ for which $\lambda(G_{12.6}) = 1$. (b) A graph $G_{12.7}$ for which $\lambda(G_{12.7}) = 2$. (c) A graph $G_{12.8}$ for which $\lambda(G_{12.8}) = 3$.

Hassler Whitney was born in New York on 23 March 1907. He was one of the founders of singularity theory, and although he is best known for his work in algebraic and differential topology, Whitney's earliest work (roughly during the period 1930 to 1933) was in the field of graph theory. He made many contributions to the theory of graph colouring, and the eventual 1976 computer-assisted proof of the Four-colour Theorem by Haken, Appel and Koch relied on some of his results. His work in graph theory culminated in an important paper in 1935 in which he laid the foundations for matroids, a fundamental notion in modern combinatorics and representation theory. He also worked in cohomology theory and wrote a book titled *Geometric Integration Theory*. He died on 10 May 1989 in Mount Dents Blanches, Switzerland.



Biographic note 38: Hassler Whitney (1907–1989)

Theorem 12.6 (Whitney's Theorem) *Let G be a graph with connectivity $\kappa(G)$, edge connectivity $\lambda(G)$ and minimum degree $\delta(G)$. Then $\kappa(G) \leq \lambda(G) \leq \delta(G)$.*

Proof If G is a disconnected graph or the trivial graph, then $\kappa(G) = \lambda(G) = 0 \leq \delta(G)$. So we may assume that G is neither disconnected nor trivial. The edges incident with a vertex of minimum degree form an edge cut, and so $\lambda(G) \leq \delta(G)$. It remains only to verify that $\kappa(G) \leq \lambda(G)$.

If $\lambda(G) = 1$, then G is a connected graph containing a bridge. Therefore, either $G \cong K_2$ or G is a connected graph that has a cut-vertex. In either case, $\kappa(G) = 1$. Hence, we may assume $\lambda(G) \geq 2$.

Let X be an edge cut of G with $|X| = \lambda(G)$ and let $e = uv \in X$. Then, $G - (X \setminus \{e\})$ is a connected graph containing e as a bridge. For each edge in $X \setminus \{e\}$, select an incident vertex different from u and v . Let S denote the resulting set of vertices. Then, $|S| \leq \lambda(G) - 1$. If S is a vertex cut, then $\kappa(G) \leq |S| < \lambda(G)$. On the other hand, if S is not a vertex cut, then $G - S$ is a connected graph. Since $G - S$ is a subgraph of $G - (X \setminus \{e\})$, it follows that e is also a bridge of $G - S$. Therefore, either $G - S \cong K_2$ or $G - S$ is a connected graph that has a cut-vertex. In either case, there is a vertex $u \in V(G) \setminus S$ such that $G - (S \cup \{u\})$ is a disconnected graph or the trivial graph. Hence, $\kappa(G) \leq |S| + 1 = \lambda(G)$. ■

We remark that the inequalities in the inequality chain of Theorem 12.6 may be strict. For example, for the graph $G_{12.9}$ in Figure 12.6 it holds that $\kappa(G_{12.9}) = 1$,

$\lambda(G_{12.9}) = 2$ and $\delta(G_{12.9}) = 3$. In fact, for any choice of natural numbers a , b and c satisfying $1 \leq a \leq b \leq c$, there exists graph G for which $\kappa(G) = a$, $\lambda(G) = b$ and $\delta(G) = c$ (see [Exercise 12.30](#)).

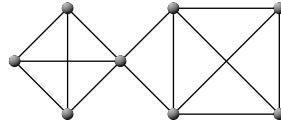


Figure 12.6: A graph $G_{12.9}$ for which $\kappa(G_{12.9}) = 1$, $\lambda(G_{12.9}) = 2$ and $\delta(G_{12.9}) = 3$.

We close this section with a sufficient condition for a graph to be k -connected. It seems logical that the larger the degrees of the vertices of a graph, the more likely it is to have large connectivity. The simplest such condition is due to [Chartrand and Harary \[1\]](#). The value of this result lies in the realisation that by computing a very basic basic graph parameter we can obtain information about the global structure of the graph.

Theorem 12.7 *Let G be a graph of order $n \geq 2$, and let k be a natural number not exceeding $n - 1$. If $\delta(G) \leq (n + k - 2)/2$, then G is k -connected.*

Proof Suppose, to the contrary, that G is not k -connected. Then G is not a complete graph. Hence there exists a vertex cut S of G such that $|S| < k$. Let H be a component of minimum order in $G - S$. Since $|V(G - S)| = n - |S|$ and $G - S$ has at least two components, the order of H is at most $(n - |S|)/2$. Let v be a vertex of H . Then v is adjacent in G only to vertices which are in $S \cup V(H)$. Thus,

$$\begin{aligned} d_G(v) &\leq |S| + |V(H)| - 1 \leq |S| + (n - |S|)/2 - 1 \\ &= (n + |S| - 2)/2 < (n + k - 2)/2, \end{aligned}$$

contradicting the fact that $\delta(G) \geq (n + k - 2)/2$. We conclude that G must be k -connected. ■

The condition of [Theorem 12.7](#) is not a necessary condition for k -connectedness. That is, there exist choices for n and k , such that $1 \leq k \leq n - 1$, and k -connected graphs G of order n for which $\delta(G) < (n + k - 2)/2$. Thus, while [Theorem 12.7](#) may be helpful in identifying some k -connected graphs, it is not a characterisation of k -connected graphs.

❖ The reader should now be able to attempt Exercises [12.17–12.32](#).

12.5 Menger's Theorem

By definition there exists a path between each pair of nonadjacent vertices in a connected graph. Since there may be more than just one path connecting such a pair of vertices, however, it is natural to ask how one may measure the degree of connectedness between a pair of vertices in a connected graph. One way of doing this is to determine the largest number of such paths that are pairwise “independent” of one another in the sense that they share no internal vertices (*i.e.* do not intersect at any vertices other than the origin-destination pair). Another

way of measuring the degree of connectedness between a pair of vertices in a connected graph is to determine the smallest number of vertices (different from the given pair of vertices) whose deletion from the graph destroys every path between the pair. A classic result of Menger [3], dating back to 1927, states that for nonadjacent vertices these two measures are equivalent. We prove two versions of this theorem — a vertex form and an edge form.

12.5.1 The vertex form of Menger's Theorem

Let u and v be distinct vertices of a graph G . Then a vertex separating set X is called a **(u, v) -vertex separating set** if u and v are in different components of $G - X$. For nonadjacent vertices u and v of G , let $\kappa_G(u, v)$ denote the minimum cardinality of a (u, v) -vertex separating set in G . Two u - v paths are **internally disjoint** if u and v are the only common vertices on these paths. Let $\kappa'_G(u, v)$ denote the largest number of pairwise internally disjoint u - v paths in G . We state the following observation whose proof is left as an exercise.

Observation 12.8 $\kappa(G) = \min\{\kappa_G(u, v) \mid u, v \in V \text{ and } uv \in E\}$ for any incomplete graph $G = (V, E)$.

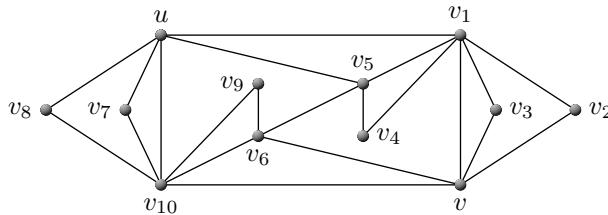


Figure 12.7: A graph $G_{12.10}$ for which $\kappa_{G_{12.10}}(u, v) = \kappa'_{G_{12.10}}(u, v) = 3$.

Note, as an example, that $\kappa_{G_{12.10}}(u, v) = \kappa'_{G_{12.10}}(u, v) = 3$ for the graph $G_{12.10}$ in Figure 12.7. The fact that $\kappa_{G_{12.10}}(u, v) = \kappa'_{G_{12.10}}(u, v)$ is not a mere coincidence, as we show next.

Theorem 12.9 (Vertex form of Menger's Theorem) *For any two nonadjacent vertices u and v of a graph G , it holds that $\kappa_G(u, v) = \kappa'_G(u, v)$.*

Proof Let G be a graph and let u, v be a pair of nonadjacent vertices in G . Clearly, $\kappa_G(u, v) \geq \kappa'_G(u, v)$ since a (u, v) -vertex separating set must contain at least one vertex from each one of the paths in any collection of internally disjoint u - v paths of G . We now show that $\kappa_G(u, v) \leq \kappa'_G(u, v)$. Let $k = \kappa_G(u, v)$. Then no set of fewer than k vertices separates u and v .

We proceed by induction on k . If $k = 1$, then there is a u - v path in G , and so $\kappa'_G(u, v) \geq 1 = \kappa_G(u, v)$. This establishes the base case. Assume, therefore, as induction hypothesis that $k \geq 2$ and that if no set of fewer than k vertices separates a pair u, v of nonadjacent vertices in G , then there are k internally disjoint u - v paths in G . Suppose there is a pair u, v of nonadjacent vertices in G such that no set of fewer than $k + 1$ vertices separate u and v .

By the induction hypothesis there are k internally disjoint u - v paths P_1, \dots, P_k in G . Since the collection of vertices that follow u on these paths (there are k of these) does not separate u and v , there is a u - v path P whose initial edge is not on P_i for any $i \in [n]$. Let x be the first vertex on P after u that belongs to P_i for some $i \in [k]$ and let P_{k+1} be the u - x subpath of P . Suppose that P_1, \dots, P_{k+1} have been chosen in such a way that the distance from x to v in $G - u$ is a minimum. If $x = v$, then we have the desired collection of $k + 1$ internally disjoint paths. Suppose, therefore, that $x \neq v$.

Again, by the induction hypothesis, there are k internally disjoint u - v paths Q_1, \dots, Q_k in $G - x$. Assume that Q_1, \dots, Q_k have been chosen so that a minimum number of edges in $B = E(G) \setminus \bigcup_{i=1}^{k+1} E(P_i)$ are used. Let H be the graph consisting of the vertices and edges of Q_1, \dots, Q_k together with the vertex x . Choose P_j for some $j \in [k+1]$ whose initial edge is not in $E(H)$. Let y be the first vertex on P_j after u which is in $V(H)$. If y is v , then we have the desired collection of $k + 1$ internally disjoint paths. So assume $y \neq v$.

If $y = x$, then let R be a shortest x - v path in $G - u$. Moreover, let z be the first vertex on R that belongs to Q_s for some $s \in [k]$. Then the distance in $G - u$ from z to v is less than the distance from x to v . This contradicts our choice of P_1, \dots, P_{k+1} , and so $y \neq x$.

If y is on Q_t for some $t \in [k]$, then the u - y subpath of Q_t has an edge in B . Otherwise, two paths in $\{P_1, \dots, P_{k+1}\}$ intersect at a vertex other than u, v or x . If we replace the u - y subpath of Q_t by the u - y subpath of P_j we obtain a collection of k internally disjoint u - v paths in $G - x$ which contains fewer edges from B than Q_1, \dots, Q_k , a contradiction. ■

Karl Menger was born in Vienna on 13 January 1902. He entered the University of Vienna in 1920 to study physics. In 1921, however, Menger attended a course by Hans Hahn on *What's new concerning the concept of a curve*. He became interested in the topic and was encouraged by Hahn to work in the field of curves. Menger's work led him to a definition of dimension. He wrote a number of important papers on dimension while in a sanatorium, recovering from a severe lung disease, and completed his doctorate in mathematics in 1924. In 1925, Menger was invited by Luitzen Egbertus Jan Brouwer to take up a post at the University of Amsterdam. Menger was subsequently invited by Hahn in 1927 to accept the chair of geometry at the University of Vienna, where he remained until 1938. With war looming large in Austria he resigned his chair and accepted a post in the United States at the University of Notre Dame. In 1948, Menger moved to the Illinois Institute of Technology where he was to remain for the rest of his life. He died on 5 October 1985 in Chicago, Illinois.



Biographic note 39: Karl Menger (1902–1985)

[Theorem 12.9](#) provides us with an easy way to verify that there is no set of $k - 1$ or fewer vertices separating two nonadjacent vertices u and v in a graph G . We simply have to produce k pairwise internally disjoint u - v paths in G . Although it might be difficult to find such a collection of paths, it is easy to verify that they are indeed internally disjoint.

There is a beautiful characterisation of k -connected graphs, due to [Whitney \[4\]](#), which is based on the result of [Theorem 12.9](#). Before we can state and prove this characterisation, however, we need another definition and intermediate result. Let $G = (V, E)$ be a graph and suppose $B \subseteq V$. A set of k paths from a vertex $a \in V$ to vertices in B is called an **a - B fan of size k** if any two of the paths have only the vertex a in common. If $a \in B$, then an a - B fan may contain the trivial path consisting of the vertex a only.

The following intermediate result will prove useful in our discussion of the characterisation of k -connected graphs by [Whitney \[4\]](#).

Lemma 12.10 *Let $G = (V, E)$ be a graph and suppose $k \in \mathbb{N}$. Then G is k -connected if and only if $|V| \geq k + 1$ and, for every subset $B \subseteq V$ with $|B| \geq k$ and $a \in V \setminus B$, there is an a - B fan of size k in G .*

Proof Suppose G is k -connected. Let $B' \subseteq B$ with $|B'| = k$ and let H be the graph obtained from G by adding a new vertex v and joining it to each vertex in B' . Then $\kappa(H) = k$ (see [Exercise 12.34](#)). Since the vertices a and v are not adjacent in H , $\kappa_H(a, v) \geq \kappa(H) = k$. Since the set B' is a (a, v) -vertex separating set in H , however, $\kappa_H(a, v) \leq k$. Consequently, $\kappa_H(a, v) = k$. Furthermore, $\kappa'_H(a, v) = \kappa_H(a, v)$ by [Theorem 12.9](#). Hence, there exist k internally disjoint a - v paths in H . Deleting v from these paths produces an a - B' fan, and therefore an a - B fan, of size k in G .

Conversely, suppose that G has at least $k + 1$ vertices, and that there is an a - B fan of size k in G for every subset $B \subseteq V$ with $|B| \geq k$ and $a \in V \setminus B$. In particular, for each vertex $v \in V$, if we take $B = V \setminus \{v\}$, then there is a v - B fan of size k in G , and so $d_G(v) \geq k$. Let u and v be nonadjacent vertices in G , and let $U = N(v)$. Then, $|U| \geq k$. By the fan condition, there is a u - U fan of size k in G . If v does not belong to any path in this fan, then each u - w path, with $w \in U$, in the u - U fan can be extended to a u - v path by adding to it the vertex v and the edge vw . Hence, $\kappa'_G(u, v) \geq k$. On the other hand, if v belongs to a path in this u - U fan, then the u - v subpath of this path, together with the remaining $k - 1$ paths in the fan and the edges joining v to the end-vertices of these paths, once again produces a set of k internally disjoint u - v paths in G , and so $\kappa'_G(u, v) \geq k$. But $\kappa_G(u, v) = \kappa'_G(u, v) \geq k$ by [Theorem 12.9](#). Hence, $\kappa(G) = \min\{\kappa_G(u, v) \mid u, v \in V \text{ and } uv \notin E\} \geq k$. ■

We are now in a position to state and prove [Whitney's characterisation of \$k\$ -connected graphs](#) which dates from 1932.

Theorem 12.11 ([Whitney's characterisation of \$k\$ -connected graphs](#)) *A graph G is k -connected for any $k \in \mathbb{N}$ if and only if there are at least k internally disjoint paths between each pair of distinct vertices of G .*

Proof Suppose there are at least k internally disjoint paths between each pair of distinct vertices of the graph $G = (V, E)$. Then, $|V| \geq k + 1$, implying, if G is complete, that G is k -connected. Assume, therefore, that G is not complete.

Let X be a minimum vertex cut in G , and let u and v be vertices in different components of $G - X$. Since X is a (u, v) -vertex separating set in G , $\kappa_G(u, v) \leq |X|$. Hence, $\kappa'_G(u, v) = \kappa_G(u, v)$ by [Theorem 12.9](#). By assumption, there are at least k internally disjoint u - v paths in G , and so $\kappa'_G(u, v) \geq k$. Consequently, $\kappa(G) = |X| \geq \kappa_G(u, v) = \kappa'_G(u, v) \geq k$.

Conversely, suppose that G is k -connected. Suppose, to the contrary, that G contains a pair u, v of vertices that are joined by at most $k - 1$ internally disjoint paths. If u and v are not adjacent, then $\kappa(G) \leq \kappa_G(u, v) = \kappa'_G(u, v) \leq k - 1$, a contradiction. Hence, u and v are adjacent. Let $H = G - uv$. Then H contains at most $k - 2$ internally disjoint u - v paths. By [Theorem 12.9](#), there therefore exists a (u, v) -vertex separating set X in H consisting of at most $k - 2$ vertices. Since $|V| \geq k + 1$, there is at least one vertex $w \notin X \cup \{u, v\}$ in G . Now the set X separates the vertex w in H from at least one of u and v , say from u . But then $X \cup \{v\}$ is a (u, w) -vertex separating set in G consisting of at most $k - 1$ vertices, and so $\kappa(G) \leq \kappa_G(u, w) \leq k - 1$, a contradiction. ■

Recall that, by [Theorem 12.3](#), a 2-connected graph has the property that every two of its vertices lie on a common cycle. This result was generalised to k -connected graphs by [Dirac \[2\]](#) in 1960. Our proof of this result again uses [Lemma 12.10](#).

Theorem 12.12 *If G is a k -connected graph with $k \geq 2$, then every k vertices of G lie on a common cycle of G .*

Proof Let X be a set of k vertices of G , and let C be a cycle in G which contains a maximum number of vertices from X . Suppose, to the contrary, that C does not contain all vertices in X and let x be a vertex of X that does not lie on C .

Suppose first that $V(C) \subset X$. Then, $|V(C)| < k$. There exist, by [Lemma 12.10](#), $|V(C)|$ internally disjoint paths joining the vertex x to each vertex in $V(C)$. Selecting two adjacent vertices u and v on the cycle C and replacing the edge uv of C by the u - x path and the x - v path in the fan we obtain a new cycle which contains more vertices from X than does C , a contradiction. We conclude that $V(C) \not\subset X$.

Let $X' = X \cap V(C) = \{x_1, \dots, x_\ell\}$. It follows from [Theorem 12.3](#) that $\ell \geq 2$. Let $x_{\ell+1} \in V(C) \setminus X'$. Since $k \geq \ell + 1$, [Lemma 12.10](#) again guarantees the existence of $\ell + 1$ internally disjoint paths P_i for $i \in [\ell + 1]$ from x to the vertices in $X' \cup \{x_\ell\}$. For each $i \in [\ell + 1]$, let v_i be the first vertex on P_i that is also a vertex of C (possibly, $v_i = x_i$), and let P'_i be the x - v_i subpath of P_i . The vertices $\{v_1, \dots, v_{\ell+1}\}$ partition $E(C)$ into $\ell + 1$ paths (or segments). Since C contains exactly ℓ vertices of X , at least one of these paths, say P , contains no interior vertex belonging to X . Let P be a v_r - v_s path ($1 \leq r, s \leq \ell + 1$ with $r \neq s$). Replacing P on the cycle C by the v_r - v_s path determined by P'_r and P'_s , we obtain a new cycle which contains more vertices from X than does C , again a contradiction. We conclude, therefore, that every vertex in X is also a vertex of C , as desired. ■

We remark that the converse of [Theorem 12.12](#) is clearly not true for $k \geq 3$. That is, for $k \geq 3$ there are many graphs for which every set of k vertices is in a cycle, but which are not k -connected — consider, as an example, any cycle containing k or more vertices.

❖ The reader should now be able to attempt Exercises [12.33–12.36](#).

12.5.2 The edge form of Menger's Theorem

In this section, we apply the **max-flow min-cut theorem** ([Theorem 7.5](#)) to derive an **edge-form of Menger's Theorem**. We again begin with some definitions.

Let u and v be distinct vertices of a graph G . Then an edge separating set Y of G is called a **(u, v) -edge separating set** if u and v are in different components of $G - Y$. For nonadjacent vertices u and v of G , let $\lambda_G(u, v)$ denote the minimum cardinality of a (u, v) -edge separating set in G . Two u - v paths are **edge-disjoint** if they share no common edge. Let $\lambda'_G(u, v)$ denote the largest number of pairwise edge-disjoint u - v paths in G . We state the following observation whose proof is left as an exercise.

Observation 12.13 $\lambda(G) = \min\{\lambda_G(u, v) \mid v \in V(G) \setminus \{u\}\}$ for any graph G and any vertex u in G .

Consider again the graph $G_{12.10}$ in [Figure 12.7](#). For this graph, $\lambda'_{G_{12.10}}(u, v) = \lambda_{G_{12.10}}(u, v) = 5$. The fact that $\lambda'_{G_{12.10}}(u, v) = \lambda_{G_{12.10}}(u, v)$ is again no coincidence, as we show next.

Theorem 12.14 (Edge form of Menger's Theorem [3]) *For any two distinct vertices u and v of a graph G it holds that $\lambda'_G(u, v) = \lambda_G(u, v)$.*

Proof Clearly, $\lambda_G(u, v) \geq \lambda'_G(u, v)$ since a (u, v) -edge separating set must contain at least one edge from each one of the paths in any collection of edge-disjoint u - v paths. We now show that $\lambda_G(u, v) \leq \lambda'_G(u, v)$. For this purpose, we transform the graph G into a network N with $V(N) = V(G)$ by replacing each edge by two arcs, one in each direction (so that the underlying digraph of N is symmetric), with source $s = u$, sink $t = v$ and with a capacity function c of N that assigns the value 1 to every arc of N .

Let f be a maximum flow in N . If N contains a directed cycle (including a directed 2-cycle) all of whose arcs a satisfy $f(a) = 1$, then we can reassign a flow of 0 to each arc on this cycle to produce a new maximum flow. Hence we may assume that there is no directed cycle (including a directed 2-cycle) all of whose arcs a satisfy $f(a) = 1$. In particular, for every pair of adjacent vertices x and y in G , at least one of $f(x, y)$ and $f(y, x)$ equals 0.

Let N' be the network obtained from N by deleting all arcs a for which $f(a) = 0$. Let D' be the underlying digraph of N' and let f' be the restriction of the flow f in N to N' . Then, $f(N) = f'(N')$. By the flow conservation constraint, $f^+(x) = f^-(x)$ for every vertex $x \in V(D) \setminus \{s, t\}$. Thus, since $f'(a) = 1$ for all arcs a in D' , it follows that $\text{id}_{D'}(x) = \text{od}_{D'}(x)$ for all $x \in V(D') \setminus \{s, t\}$. We proceed further with the following claim whose proof is left as an exercise (see [Exercise 12.39](#)).

Claim 12.15 *The largest number of arc-disjoint s - t paths in N' equals $f(N)$.*

Since every two arc-disjoint s - t paths in D' correspond to two edge-disjoint u - v paths in G , it follows by [Claim 12.15](#) that $\lambda'_G(u, v) \geq f(N)$. We show next that $f(N) \geq \lambda_G(u, v)$. By the **max-flow min-cut theorem** ([Theorem 7.5](#)), the value of a maximum flow in N equals the capacity of a minimum cut. Let (S, \bar{S}) be a minimum cut. Then, $f(N) = c(S, \bar{S})$. Let $F = \{xy \in E(G) \mid (x, y) \in (S, \bar{S})\}$. Since each arc in N has capacity 1, $|F| = c(S, \bar{S})$. Furthermore, since $u = s \in S$ and

$v = t \in \bar{S}$, the set F is a (u, v) -edge separating set in G , and so $\lambda_G(u, v) \leq |F| = c(S, \bar{S}) = f(N)$. Hence, $\lambda'_G(u, v) \geq f(N) \geq \lambda_G(u, v)$. Consequently, $\lambda'_G(u, v) = \lambda_G(u, v)$. ■

As a consequence of [Observation 12.13](#) and the proof of [Theorem 12.14](#), we have an easy method for computing the edge connectivity of a connected graph G of order n and size m . We select an arbitrary vertex u in G . For each of the $n - 1$ vertices $v \in V(G) \setminus \{u\}$, we then construct the network N described in the proof of [Theorem 12.14](#) (where $s = u$ and $t = v$) and use the [max-flow min-cut algorithm](#) ([Algorithm 18](#)) to determine the maximum flow in N which is precisely the value $\lambda_G(u, v)$. We then compute $\min\{\lambda_G(u, v) \mid v \in V(G) \setminus \{u\}\}$ which, by [Observation 12.13](#), is the edge-connectivity $\lambda(G)$ of G . The [max-flow min-cut algorithm](#) has worst-case time complexity $\mathcal{O}(nm^2)$, and so the worst-case time complexity of computing the $n - 1$ values $\lambda_G(u, v)$ for $v \in V(G) \setminus \{u\}$, and hence of determining the value of $\lambda(G)$, is $\mathcal{O}(n^2m^2)$.

We close this section demonstrating the working of the procedure described above by computing the edge connectivity of the graph $G_{12.11}$ in [Figure 12.8\(a\)](#). The network $N_{12.12}$ associated with the graph $G_{12.11}$, which is described in the proof of [Theorem 12.14](#), is shown in [Figure 12.8\(b\)](#).

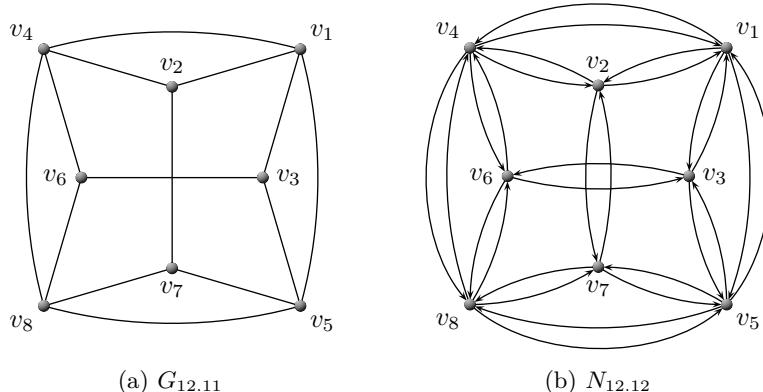


Figure 12.8: (a) A graph $G_{12.11}$ for which $\lambda(G_{12.11}) = 3$. (b) The network $N_{12.12}$ associated with the graph $G_{12.11}$, as described in the proof of [Theorem 12.14](#).

Each arc in $N_{12.12}$ has a flow capacity restriction of one unit. By choosing u to be the vertex v_1 , the maximum flows from u to v_i in the network $N_{12.12}$ are shown in [Table 12.2](#) for all $i = 2, \dots, 8$. These values were computed by means of [Algorithm 18](#). Since the smallest value in the table is 3, we conclude that $\lambda(G_{12.11}) = 3$.

v_2	v_3	v_4	v_5	v_6	v_7	v_8
3	3	4	4	3	3	4

Table 12.2: The maximum flows from the vertex $u = v_1$ to the vertex v_i in the network $N_{12.12}$ of [Figure 12.8\(b\)](#) for all $i = 2, \dots, 8$.

- ❖ The reader should now be able to attempt Exercises 12.37–12.42.

12.6 Computing the connectivity of a graph

In this section, we apply the [max-flow min-cut theorem](#) ([Theorem 7.5](#)) to determine the connectivity of a (connected) graph G . We begin by presenting an alternative proof of the [vertex form of Menger's Theorem](#) ([Theorem 12.9](#)) using the [max-flow min-cut theorem](#).

Alternative proof of Theorem 12.9 Let G be a graph and let u, v be a pair of nonadjacent vertices in G . Clearly, $\kappa_G(u, v) \geq \kappa'_G(u, v)$. We show that $\kappa_G(u, v) \leq \kappa'_G(u, v)$.

Let D be the digraph obtained from G as follows: Split every vertex x of G into two vertices x^- and x^+ , and add the arc (x^-, x^+) . Furthermore, if $xy \in E(G)$, then add the arcs (x^+, y^-) and (y^+, x^-) . Let N be the network with D as underlying graph, source $s = u^+$, sink $t = v^-$, and with a capacity function c that assigns the value 1 to every arc of the type (x^-, x^+) , and ∞ to all remaining arcs of D .

Since $uv \notin E(G)$, the sink s and source t are nonadjacent in D . If $u v_1 v_2 \dots v = t$ is a u - v path in G , then $s = u^+ v_1^- v_1^+ v_2^- v_2^+ \dots v^- = t$ is an s - t (directed) path in D . For each vertex x of G , $\text{od}(x^-) = 1$ and $\text{id}(x^+) = 1$ in D . Hence every s - t path in D is of the type $s = u^+ v_1^- v_1^+ v_2^- v_2^+ \dots v^- = t$. It follows that there is a one-to-one correspondence between the u - v paths in G and the s - t paths in D . Furthermore, two s - t paths in D are internally disjoint if and only if the corresponding u - v paths in G are internally disjoint. If two s - t paths in D share a common vertex, say x^- or x^+ , then they also share a common arc, namely (x^-, x^+) . Hence any two arc-disjoint s - t paths in D are necessarily internally disjoint paths. It follows that there is a one-to-one correspondence between the internally disjoint u - v paths in G and the arc-disjoint s - t paths in D .

Let f be a maximum flow in N . Then the flow along every arc of N is either 0 or 1 (see [Exercise 12.43](#)). Hence, the value of the flow $f(N)$ is finite. As in the proof of [Theorem 12.14](#), we may assume that there is no directed cycle in N all of whose arcs a satisfy $f(a) = 1$. Let N' and D' be as defined in the proof of [Theorem 12.14](#). Then, $f(N) = f(N')$. As in [Claim 12.15](#), we can show that the largest number of arc-disjoint s - t paths in N' is $f(N)$. Hence, since every pair of arc-disjoint s - t paths in N' correspond to a pair of internally disjoint u - v paths in G , it follows that $\kappa'_G(u, v) \geq f(N)$. We show next that $f(N) \geq \kappa_G(u, v)$. By the [max-flow min-cut theorem](#) ([Theorem 7.5](#)), $f(N) = c(S, \bar{S})$ where (S, \bar{S}) is a minimum cut (with $s \in S$ and $t \in \bar{S}$). Since $f(N)$ is finite, no arc in (S, \bar{S}) has infinite capacity. Therefore, each arc in (S, \bar{S}) has capacity 1 and is of the type (x^-, x^+) . Let $X = \{x \in V(G) \mid (x^-, x^+) \in (S, \bar{S})\}$. Then $|X| = c(S, \bar{S})$.

We show that X is a (u, v) -vertex separating set. Consider any u - v path $u v_1 v_2 \dots v = t$ in G . Then $s = u^+ v_1^- v_1^+ v_2^- v_2^+ \dots v^- = t$ is an s - t (directed) path in D and therefore must contain an arc (v_i^-, v_i^+) of (S, \bar{S}) , whence $v_i \in X$. It follows that X is a (u, v) -vertex separating set, and so $\kappa_G(u, v) \leq |X| = c(S, \bar{S}) = f(N)$. Hence, $\kappa'_G(u, v) \geq f(N) \geq \kappa_G(u, v)$. Consequently, $\kappa'_G(u, v) = \kappa_G(u, v)$. ■

As a consequence of [Observation 12.8](#) and the alternative proof of [Menger's Theorem](#) ([Theorem 12.9](#)) presented in this section, we can present an algorithm,

presented in pseudocode form as [Algorithm 29](#), for computing the connectivity of a connected graph.

Algorithm 29: The connectivity of a connected graph

Input : A connected graph G of order n and size m with vertex set $V(G) = \{v_1, \dots, v_n\}$ and edge set $E(G)$.

Output : The connectivity $\kappa(G)$ of G .

```

1  $\kappa \leftarrow n - 1$ 
2 if  $m = \binom{n}{2}$  then print  $\kappa$ ; stop
3 for  $i = 1$  to  $n - 1$  do
4   if  $i > \kappa + 1$  then print  $\kappa$ ; stop
5   else
6     for  $j = i + 1$  to  $n$  do
7       if  $v_i v_j \notin E(G)$  then
8         Construct the network  $N$  described in the alternative proof
         of Theorem 12.14 where  $s = v_i^+$  and  $t = v_j^-$ , and use
         Algorithm 18 to determine the maximum flow  $f(N)$  (which
         equals  $\kappa_G(v_i, v_j)$ )
9       if  $f(N) < \kappa$  then  $\kappa \leftarrow f(N)$ 

```

The following result guarantees the correct working of [Algorithm 29](#).

Theorem 12.16 [Algorithm 29](#) computes the connectivity $\kappa(G)$ of the (connected) input graph G .

Proof If $d_G(v_i) = n - 1$ for all $i \in [n]$, then $G \cong K_n$ and the output $\kappa = n - 1$ is the connectivity of G . Assume, therefore, that G is not complete. Then the algorithm terminates in [Step 4](#) at some integer $i > \kappa + 1$, and so $\kappa = \kappa_G(v_k, v_\ell)$ for some pair k, ℓ satisfying $1 \leq k \leq i - 1$ and $k < \ell \leq n$. Thus, by [Observation 12.8](#), $\kappa(G) \leq \kappa < i - 1$. Let S be a minimum vertex cut of G . Since $\kappa(G) \leq i - 2$, there exists a vertex v_r in the set $\{v_1, \dots, v_{i-1}\}$ such that $v_r \notin S$. Let v_s be a vertex that belongs to a component of $G - S$ not containing v_r . Then, $v_r v_s \notin E(G)$ and $\kappa(G) \leq \kappa_G(v_r, v_s) \leq |S| = \kappa(G)$. Hence, $\kappa(G) = \kappa_G(v_r, v_s)$. Since $r \in [i - 1]$, however, $\kappa_G(v_r, v_s)$ is computed before the algorithm terminates and, as such, $\kappa \leq \kappa_G(v_r, v_s) = \kappa(G)$. Consequently, $\kappa = \kappa(G)$ in [Step 4](#) of [Algorithm 29](#). ■

We demonstrate the working of [Algorithm 29](#) by applying it to the graph $G_{12.11}$ in [Figure 12.8\(a\)](#) of order $n = 8$ and size $m = 14$. The corresponding network $N_{12.13}$, described in the alternative proof of [Theorem 12.9](#) presented above, is shown in [Figure 12.9](#). The value of κ is initialised as 7 in [Step 1](#) of [Algorithm 29](#). Since $m \neq 28 = \binom{8}{2}$, the outer **for**-loop spanning [Steps 3–9](#) is entered. For $i = 1$ the **if**-test in [Step 4](#) fails and hence the inner **for**-loop spanning [Steps 6–9](#) is entered. This inner **for**-loop produces the maximum network flows from the source v_1^+ to the sink v_j^- in $N_{12.13}$ shown in the first row of [Table 12.3](#), during which the value of κ is updated downwards to 3.

For $i = 2$ the **if**-test in [Step 4](#) fails again and hence the inner **for**-loop spanning [Steps 6–9](#) is entered once more, producing the maximum network flows from the

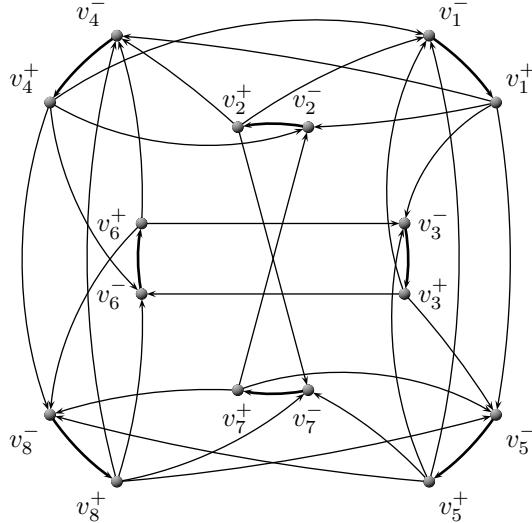


Figure 12.9: The network $N_{12.13}$ described in the alternative proof of [Theorem 12.9](#) which corresponds to the graph $G_{12.11}$. Edges shown in bold face have a flow capacity restriction of one unit, while the other edges are not restricted in terms of the flow they can accommodate.

source v_2^+ to the sink v_j^- in $N_{12.13}$ shown in the second row of [Table 12.3](#), during which no update is required for the value of $\kappa = 3$.

For $i = 3$ the **if**-test in [Step 4](#) fails yet again and so the inner **for**-loop spanning Steps [6–9](#) is entered again, producing the maximum network flows from the source v_3^+ to the sink v_j^- in $N_{12.13}$ shown in the third row of [Table 12.3](#), still not requiring an update for the value of $\kappa = 3$.

For $i = 4$ the **if**-test in [Step 4](#) fails one last time and so the inner **for**-loop spanning Steps [6–9](#) is entered again, producing the maximum network flows from the source v_4^+ to the sink v_j^- in $N_{12.13}$ shown in the last row of [Table 12.3](#), still not requiring an update for the value of $\kappa = 3$.

The **if**-test in [Step 4](#) succeeds for $i = 5$, and so the algorithm yields the connectivity $\kappa(G_{12.11}) = 3$ as output.

i	κ	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
1	7	—	—	—	—	3	3	3
2	3		3	—	3	3	—	3
3	3			3	—	—	3	3
4	3				3	—	3	—

Table 12.3: Output produced by [Algorithm 29](#) when applied to the graph $G_{12.11}$ in [Figure 12.8\(a\)](#) and network $N_{12.13}$ in [Figure 12.9](#).

We close this section with a discussion on the worst-case time complexity of [Algorithm 29](#). The **for**-loop spanning Steps [3–9](#) is performed at most $\kappa(G) + 2$ times. Each time the contents of this **for**-loop is performed (except the last), the **max-flow min-cut algorithm** ([Algorithm 18](#)) is called at most $n-i+1$ times with the vertex v_i^+

as source. Hence, there are $\mathcal{O}(n)$ maximum-flow problems that have to be solved with v_i^+ as source, and so the total number of maximum-flow problems that have to be solved is $\mathcal{O}(n\kappa(G))$. By Whitney's Theorem (Theorem 12.6), $\kappa(G) \leq \delta(G)$, and so

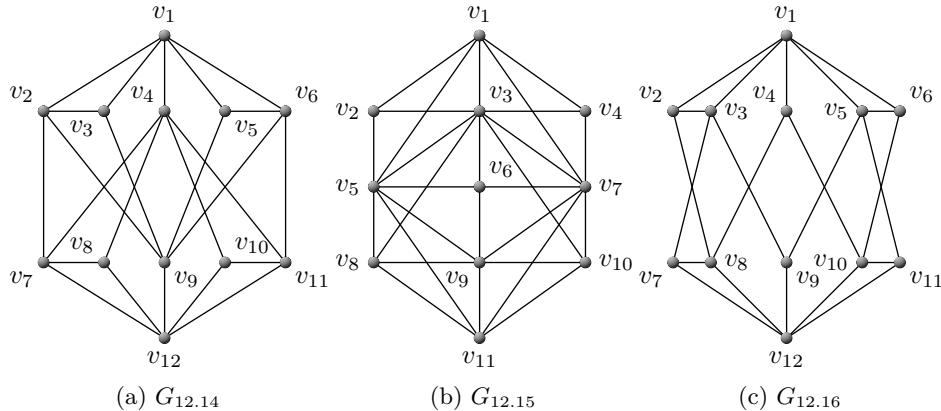
$$n\kappa(G) \leq n\delta(G) \leq \sum_{i=1}^n d_G(v_i) = 2m$$

by the Hand Shaking Lemma (Theorem 1.1). Consequently, $n\kappa(G) = \mathcal{O}(m)$. Since the max-flow min-cut algorithm has complexity $\mathcal{O}(nm^2)$, it follows that the connectivity of G can be computed in $\mathcal{O}(nm^3)$ time.

❖ The reader should now be able to attempt Exercises 12.43–12.44.

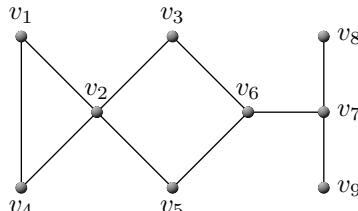
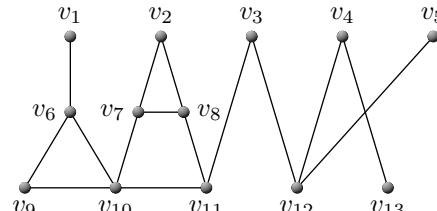
Exercises

- 12.1 Prove or disprove the following statement: Let G be a connected graph of order at least 3. Then G has a bridge if and only if G has a cut-vertex.
- 12.2 Prove that a connected cubic graph has a bridge if and only if G has a cut-vertex.
- 12.3 Find two vertex cuts of different cardinalities in each of the graphs in Figure 12.10.



- 12.4 Explain why any incomplete, connected graph with minimum degree δ contains a cut set of cardinality δ .
- 12.5 Prove or disprove the following statement: If G is an incomplete graph with a **universal vertex** (a vertex adjacent to every other vertex of G), then v belongs to every vertex cut of G .
- 12.6 Find two edge cuts of different cardinalities in each of the graphs in Figure 12.10.
- 12.7 Prove that if $\{u, v\}$ is a 2-edge cut of a graph G , then every cycle of G that contains u also contains v .

- 12.8 Prove that every minimal edge separating set of a graph G of order $n \geq 2$ is an edge cut of G .
- 12.9 Prove or disprove the following statement: If G is an incomplete graph with a universal vertex, then every edge cut of G contains an edge incident with G .
- 12.10 Let $\{e_1, e_2\}$ be an edge cut of a graph G . Show that every cycle of G containing e_1 must also contain e_2 .
- 12.11 Show that in any graph G , the number of edges that are common to a cycle of G and an edge cut of G is even.
- 12.12 Show that the deletion of the edges in an edge cut of minimum cardinality of a connected graph results in a disconnected graph with exactly two components. (Remark: Note that a similar result does not hold for vertex cuts of minimum cardinality.)
- 12.13 Use [Algorithm 28](#) to determine the blocks of each of the graphs in [Figure 12.11](#).

(a) $G_{12.17}$ (b) $G_{12.18}$ **Figure 12.11:** The graphs associated with [Exercise 12.13](#).

- 12.14 Prove that the relation \sim defined in [Section 12.3](#) is an equivalence relation on the edge set $E(G)$ of a graph G ; that is, for any edges $e, f, h \in E(G)$, verify that
- $e \sim e$ [reflexivity],
 - if $e \sim f$, then $f \sim e$ [symmetry], and
 - if $e \sim f$ and $f \sim h$, then $e \sim h$ [transitivity].
- 12.15 Prove that if G is a connected graph with at least one cut-vertex, then G has at least two end-blocks.
- 12.16 Let A denote the set of cut-vertices of a graph G and let B denote the set of its blocks. The **block graph** of G is a bipartite graph with partite sets A and B in which a vertex $a \in A$ is adjacent to a vertex $b \in B$ if a is an edge of b in G . Prove that the block graph of any connected graph is a tree.
- 12.17 Explain why any hamiltonian graph is 2-connected.
- 12.18 Show that the Petersen graph in [Figure 12.5\(c\)](#) is 3-connected.
- 12.19 Determine, by inspection, the connectivity of each of the graphs in [Figure 12.12](#). Motivate your answer.

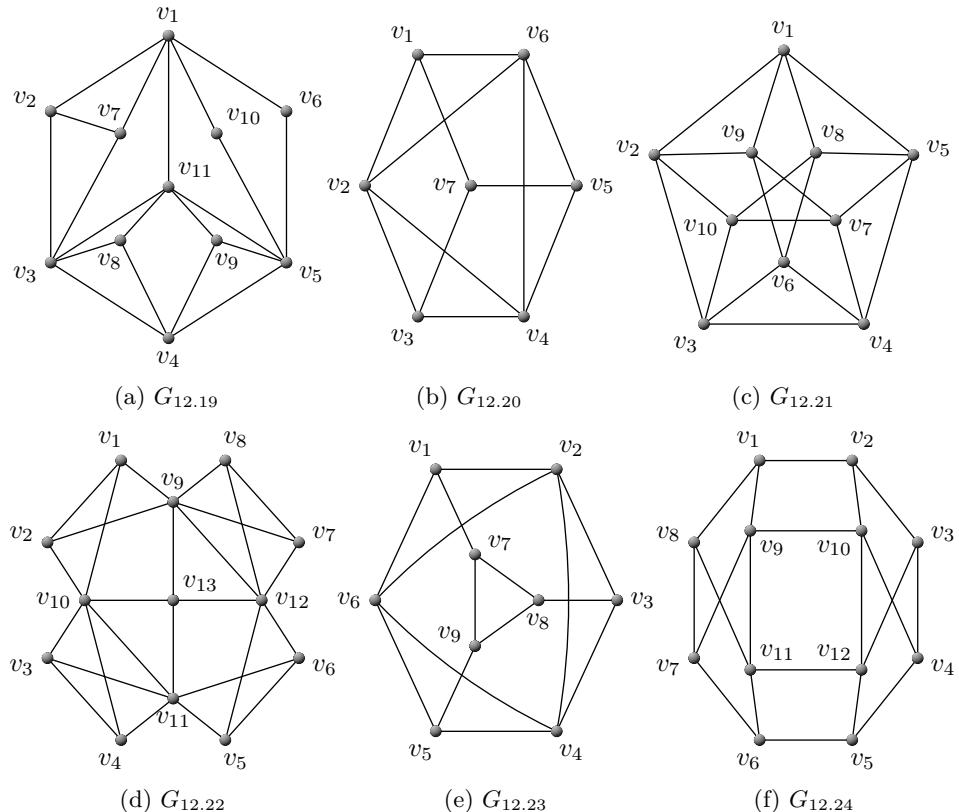


Figure 12.12: The graphs associated with Exercises 12.19 and 12.25.

- 12.20 Determine the connectivity of a complete k -partite graph.
- 12.21 Produce a 3-regular graph that is 2-connected but not 3-connected. (Hint: Start with a cycle of length 8 and add four edges.)
- 12.22 Define, for any graph G and any edge $e = xy$ of G , the so-called **contraction graph** G / e obtained by replacing the vertices x and y by a new vertex and joining this new vertex to all vertices that were adjacent to x or y in G .
 - Let $G \not\cong K_3$ be a 2-connected graph and let e be any edge of G . Show that either $G - e$ or G / e is again 2-connected.
 - Let G be a 3-connected graph and let $e = xy$ be any edge of G . Show that G / e is 3-connected if and only if $G - e$ is 2-connected.
- 12.23 Show that the result of [Theorem 12.5](#) is best possible by showing that for each integer $k \geq 2$, there exists a graph G of order $n \geq k + 1$ such that $\delta(G) = (n + k - 3)/2$ and $\kappa(G) < k$.
- 12.24 Determine, by inspection, the edge-connectivity of each of the graphs in [Figure 12.12](#). Motivate your answer.
- 12.25 Determine $\lambda(K_n)$.
- 12.26 Determine $\lambda(K_{p,q})$ where p and q are integers such that $1 \leq p \leq q$.

- 12.27 Prove or disprove the following statement: If H is a subgraph of a graph G , then
- $\kappa(H) \leq \kappa(G)$ and
 - $\lambda(H) \leq \lambda(G)$.
- 12.28 Show that if G is a k -connected graph, then $G \cup K_1$ is $(k+1)$ -connected and $(k+1)$ -edge connected.
- 12.29 Construct, for any integers a , b and c satisfying $1 \leq a \leq b \leq c$, a graph G with $\kappa(G) = a$, $\lambda(G) = b$ and $\delta(G) = c$.
- 12.30 Prove that if G is a cubic graph, then $\kappa(G) = \lambda(G)$.
- 12.31 Prove that if G is a graph of order n and minimum degree $\delta(G) \geq (n-1)/2$, then $\lambda(G) = \delta(G)$.
- 12.32 Use the algorithm described at the end of [Section 12.5](#) to compute the edge-connectivity of each of the graphs in [Figure 12.13](#).

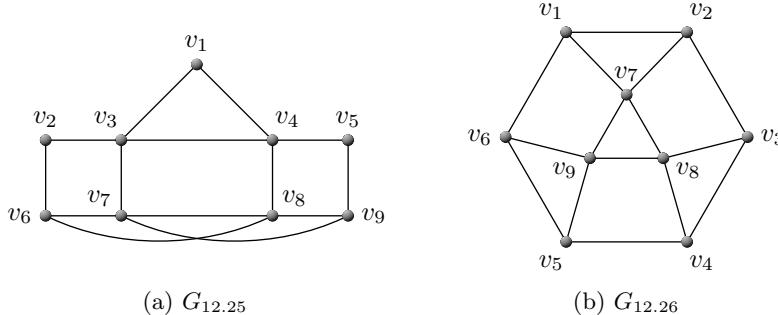


Figure 12.13: The graphs associated with [Exercise 12.32](#).

- 12.33 Prove [Observation 12.8](#).
- 12.34 Let v_1, \dots, v_k be k distinct vertices of a k -connected graph G . Let H be the graph obtained from G by adding a new vertex v and joining it to each of v_1, \dots, v_k . Show that $\kappa(H) = k$.
- 12.35 Find as many internally disjoint $x-y$ paths as possible in each of the graphs in [Figure 12.14](#).
- 12.36 Find two (x, y) -vertex separating sets of different cardinalities in each of the graphs in [Figure 12.14](#).
- 12.37 Prove that $n \geq k(\text{diam}(G) - 1) + 2$ for any k -connected graph of order n with diameter $\text{diam}(G)$.
- 12.38 Prove [Observation 12.13](#).
- 12.39 Prove [Claim 12.15](#).
- 12.40 Use the [edge form of Menger's Theorem](#) ([Theorem 12.14](#)) to prove that a graph G is k -edge connected if and only if every two vertices of G are connected by at least k edge-disjoint paths.

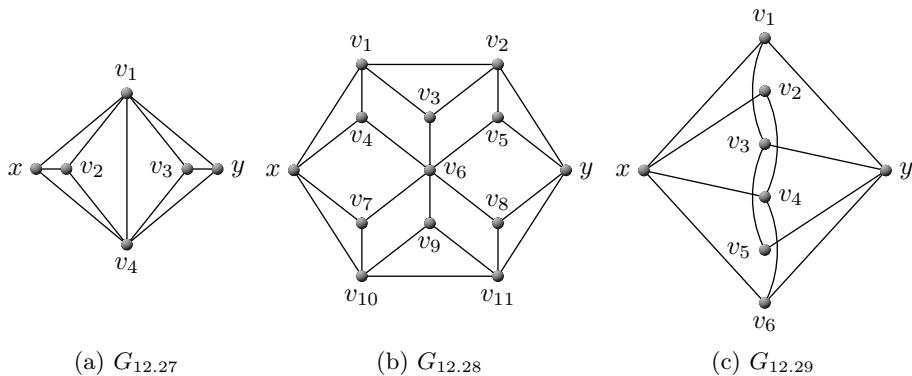


Figure 12.14: The graphs associated with Exercises 12.35 and 12.36.

- 12.41 Show that every k -connected graph of order at least $2k$ contains a cycle of length $2k$.
- 12.42 Prove that $m \geq k \cdot \text{diam}(G)$ for any k -edge connected graph G of size m with diameter $\text{diam}(G)$.
- 12.43 In the alternative proof of Menger's Theorem (Theorem 12.9) provided in Section 12.6, prove that if $f(a) \geq 1$ for some arc a of N , then $f(a) = 1$.
- 12.44 Use Algorithm 29 to compute the connectivity of each of the graphs in Figure 12.15.

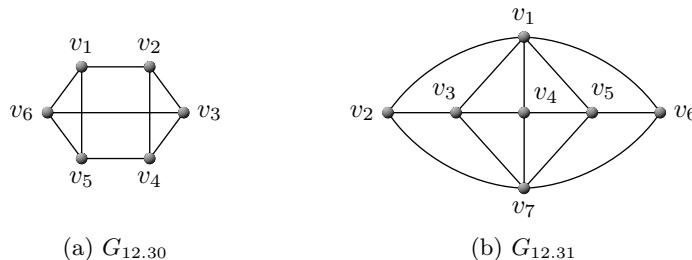


Figure 12.15: The graphs associated with Exercise 12.44.

Computer exercises

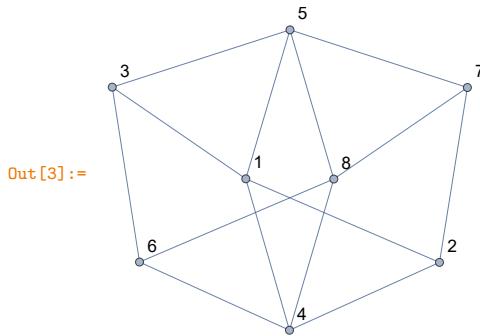
The commands `VertexConnectivity` and `EdgeConnectivity` may be used to compute respectively the connectivity $\kappa(G)$ and the edge connectivity $\lambda(G)$ of a graph G . For example, the commands

```
In[1]:= A11 = {{0, 1, 1, 1, 1, 0, 0, 0}, {1, 0, 0, 1, 0, 0, 1, 0}, {1, 0, 0, 0, 0, 1, 0, 0}, {1, 1, 0, 0, 0, 1, 0, 1}, {1, 0, 1, 0, 0, 0, 1, 1}, {0, 0, 1, 1, 0, 0, 0, 1}, {0, 1, 0, 0, 1, 0, 0, 1}, {0, 0, 1, 0, 0, 1, 0, 0}, {0, 0, 0, 1, 1, 1, 1, 0}};
In[2]:= MatrixForm[A11]
In[3]:= G11 = AdjacencyGraph[A11, VertexLabels -> "Name"]
In[4]:= VertexConnectivity[G11]
In[5]:= EdgeConnectivity[G11]
```

produce the adjacency matrix

$$\text{Out}[2]:= \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

and graphic representation



of the graph $G_{12.11}$ in [Figure 12.8\(a\)](#) as well as the connectivity and edge connectivity values

```
Out[4]:= 3
Out[5]:= 3
```

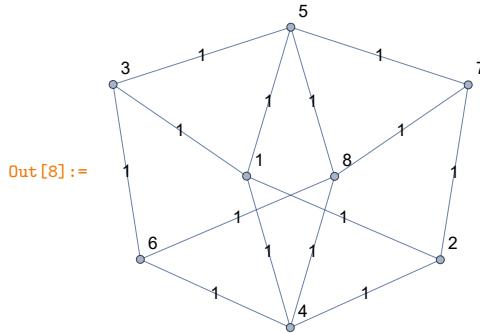
for the graph $G_{12.11}$. The value edge connectivity $\lambda(G_{12.11}) = 3$ returned by [MATHEMATICA](#) may be validated by means of the algorithmic procedure described at the end of [Section 12.5](#). For example, the commands

```
In[6]:= A11prime = {{Infinity, 1, 1, 1, 1, Infinity, Infinity, Infinity}, {1,
    Infinity, Infinity, 1, Infinity, Infinity, 1, Infinity}, {1, Infinity,
    Infinity, 1, 1, Infinity, Infinity}, {1, 1, Infinity,
    Infinity, 1, Infinity, 1}, {1, Infinity, 1, Infinity,
    Infinity, 1, 1}, {Infinity, Infinity, 1, 1, Infinity,
    Infinity, 1}, {Infinity, 1, Infinity, Infinity, 1, Infinity,
    Infinity, 1}, {Infinity, Infinity, Infinity, 1, 1, 1, Infinity}};
In[7]:= MatrixForm[A11prime]
In[8]:= N11 = WeightedAdjacencyGraph[A11prime, EdgeLabels -> "EdgeWeight",
    VertexLabels -> "Name"]
```

produce the adjacency matrix

$$\text{Out}[7]:= \begin{pmatrix} \infty & 1 & 1 & 1 & 1 & \infty & \infty & \infty \\ 1 & \infty & \infty & 1 & \infty & \infty & 1 & \infty \\ 1 & \infty & \infty & \infty & 1 & 1 & \infty & \infty \\ 1 & 1 & \infty & \infty & \infty & 1 & \infty & 1 \\ 1 & \infty & 1 & \infty & \infty & \infty & 1 & 1 \\ \infty & \infty & 1 & 1 & \infty & \infty & \infty & 1 \\ \infty & 1 & \infty & \infty & 1 & \infty & \infty & 1 \\ \infty & \infty & \infty & 1 & 1 & 1 & 1 & \infty \end{pmatrix}$$

and graphical representation



of the network $N_{12.12}$ in [Figure 12.8\(b\)](#) associated with the graph $G_{12.11}$, as defined in the proof of [\(the edge form of\) Menger's Theorem \(Theorem 12.14\)](#). Furthermore, the commands

```
In[9]:= n = 8;
In[10]:= kappa = n - 1;
In[11]:= For[i = 1, i <= n - 1, i++,
           If[i > kappa + 1,
               Print["kappa=", kappa]; Break[],
               For[j = i + 1, j <= n, j++,
                   If[A11[[i, j]] == 0,
                       f1 = FindMaximumFlow[N11prime, 2 i, 2 j - 1];
                       Print["Max Flow from i^+=", 2 i,
                             " (i=", i, ") to j^-=", 2 j - 1, " (j=", j, ") is ", f1]];
                   If[f1 < kappa, kappa = f1]]];
           Print["===="]
]
```

produce as output:

```
Out[11]:= Max Flow from i^+=2 (i=1) to j^-=11 (j=6) is 3
Max Flow from i^+=2 (i=1) to j^-=13 (j=7) is 3
Max Flow from i^+=2 (i=1) to j^-=15 (j=8) is 3
=====
Max Flow from i^+=4 (i=2) to j^-=5 (j=3) is 3
Max Flow from i^+=4 (i=2) to j^-=9 (j=5) is 3
Max Flow from i^+=4 (i=2) to j^-=11 (j=6) is 3
Max Flow from i^+=4 (i=2) to j^-=15 (j=8) is 3
=====
Max Flow from i^+=6 (i=3) to j^-=7 (j=4) is 3
Max Flow from i^+=6 (i=3) to j^-=13 (j=7) is 3
Max Flow from i^+=6 (i=3) to j^-=15 (j=8) is 3
=====
Max Flow from i^+=8 (i=4) to j^-=9 (j=5) is 3
Max Flow from i^+=8 (i=4) to j^-=13 (j=7) is 3
=====
kappa=3
```

Projects

This section contains three projects related to the connectivity and edge connectivity of a graph. In the first project, the focus is on the robustness of the SANReN backbone fibre-optic cable network which provides internet access to South Africa.

The reader is guided towards exploring the robustness of the network and discovering low-cost additional infrastructure links that, if added, would increase the network robustness. The aim in the second project is to investigate the connectivity and edge connectivity of a graph model of sea vessel tourism voyages between ports on the Pacific Rim. In the final project, the reader is asked to analyse the robustness of the London Underground system in terms of potential station and/or rail link unavailability.

Project 12.1: The South African fibre-optic network backbone

The purpose of this project is to explore the robustness of the SANReN backbone network of fibre-optic cables used for high-speed internet access in South Africa, as shown in [Figure 12.16](#).

Consider only the part of the network on South African soil (*i.e.* disregarding the Tenet/Seacom cable linking South Africa to Europe via the Indian Ocean and disregard the fact that the city of Durban represents a cut-vertex in the infrastructure).

Tasks

1. Produce a graphical representation of the graph $G_{12.32}$ induced by the network in [Figure 12.16](#) in which cities and service sites are denoted by vertices, and optic-fibre cables by edges. Take into account both the existing parts and the planned future extensions of the network.
2. Although the graph $G_{12.32}$ is 1-connected, it is not 2-connected. Explain why.
3. Explain why it is desirable that the SANReN backbone network should be 2-connected.
4. Propose a viable plan for laying additional fibre-optic cables (over and above the planned future extensions) so as to render the fibre-optic network 2-connected. By viable, we mean that cost should be taken into account. The cost of laying a fibre-optic cable increases with the length of the cable. It would, for example, certainly not be viable to lay a fibre-optic cable between one of the SKA sites and Makhado!
5. Why would it not be realistic to attempt to extend the SANReN network so that it is 3-connected?

Project 12.2: Pacific passenger liner sea routes

The purpose of this project is to explore the level of redundancy in the network of passenger liner sea routes between large ports in South-East Asia and on the North-American West Coast, as shown in [Figure 12.17](#). The graph in [Figure 12.17](#) represents traveller choices in terms of direct sea connections between the thirteen ports in question.

Tasks

1. Produce a graphical representation of the network $N_{12.34}$ of order 13 associated with $G_{12.33}$, as defined in the proof of [Theorem 12.14](#).

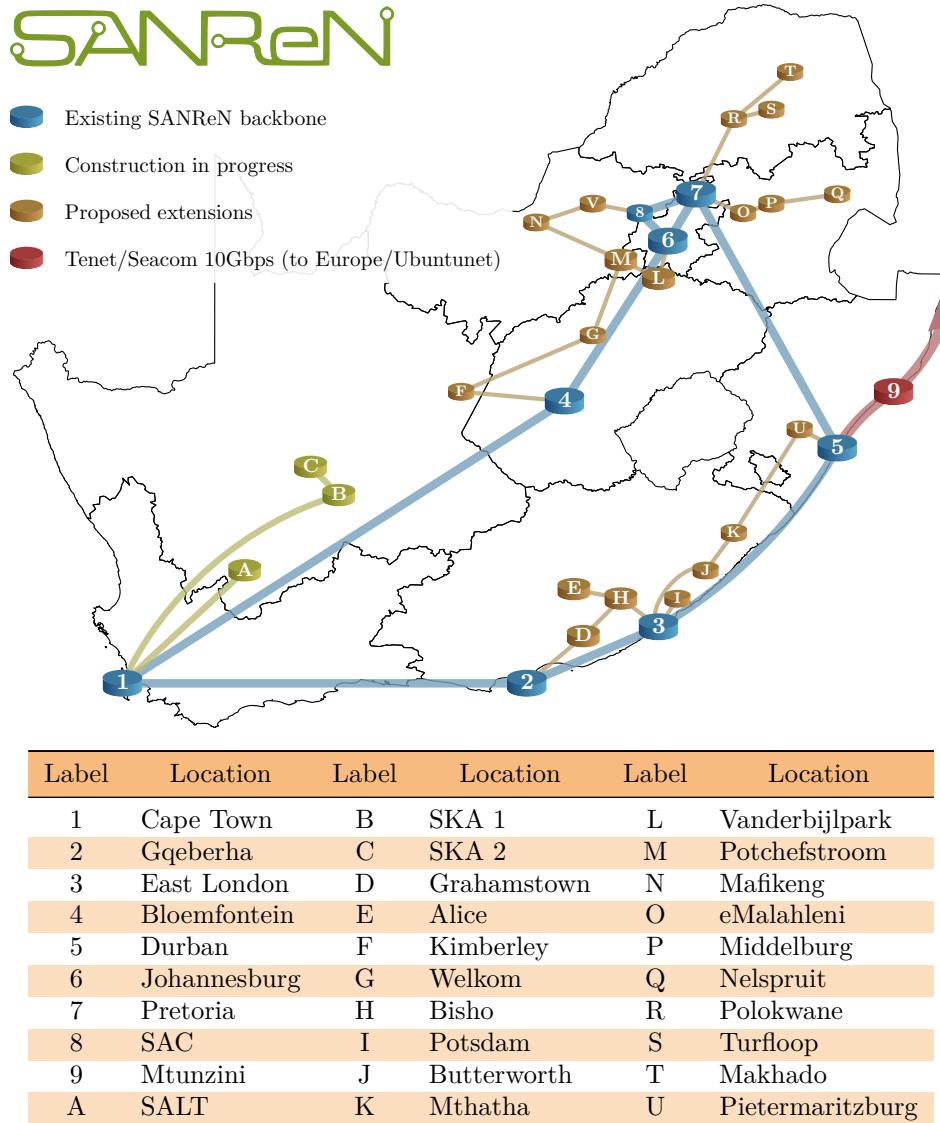


Figure 12.16: Graphical representation of the current and planned SANReN backbone network of fibre-optic cables used for internet access in South Africa.

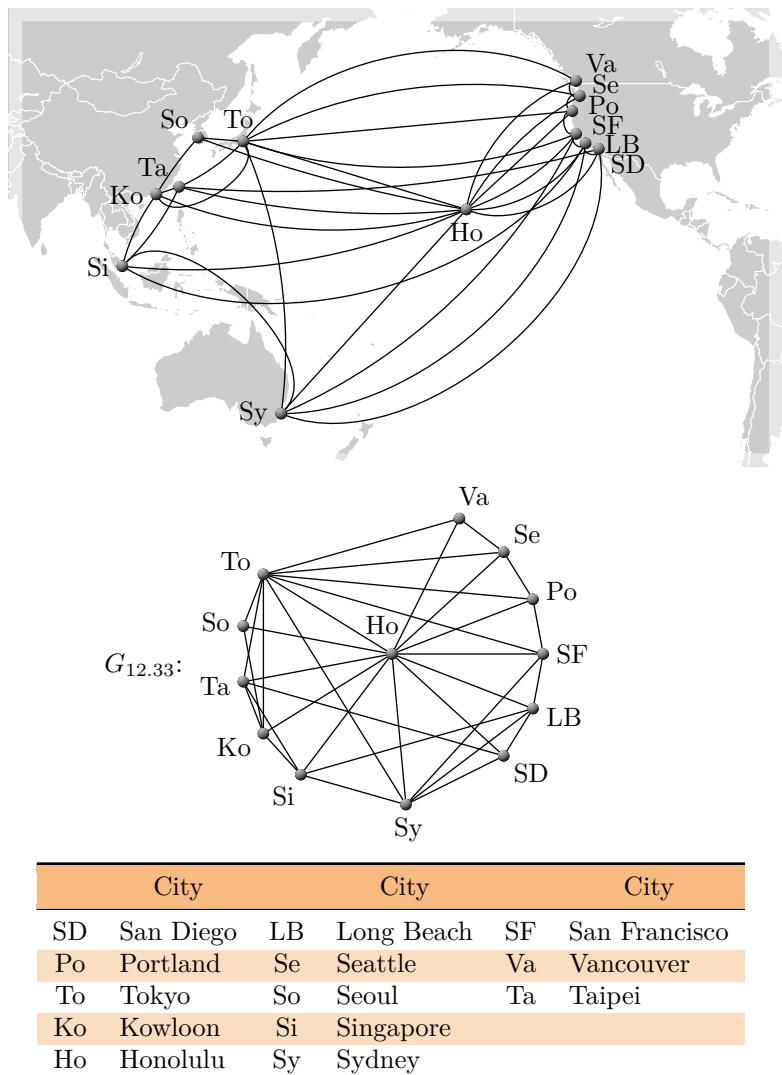


Figure 12.17: A graph $G_{12.33}$ representing the network of passenger liner sea routes between large ports on the Pacific Rim.

2. Use the network $N_{12.34}$ of [Task 1](#) and the algorithm described at the end of [Section 12.5](#) to compute the value of $\lambda(G_{12.33})$. (Hint: Use [MATHEMATICA](#), as explained in the [Computer Exercises](#).)
3. Explain in what practical sense the value of $\lambda(G_{12.33})$, as determined in [Task 2](#), is of concern to travel agents booking voyages for travellers in the network of [Figure 12.17](#).
4. Produce a graphical representation of the network $N'_{12.34}$ of order 26 associated with $G_{12.33}$, as defined in the alternative proof of [Theorem 12.9](#) in [Section 12.6](#).
5. Use the network $N'_{12.34}$ of [Task 4](#) and [Algorithm 29](#) to compute the value of $\kappa(G_{12.33})$. (Hint: Use [MATHEMATICA](#), as explained in the [Computer Exercises](#).)
6. Explain in what practical sense the value of $\kappa(G_{12.33})$, as determined in [Task 5](#), is of concern to travel agents booking voyages for travellers in the network of [Figure 12.17](#).

Project 12.3: The London Underground system

The purpose of this project is to explore the robustness of the London Underground system within the region roughly enclosed by the (yellow) Circle Line in the map in [Figure 12.18](#).

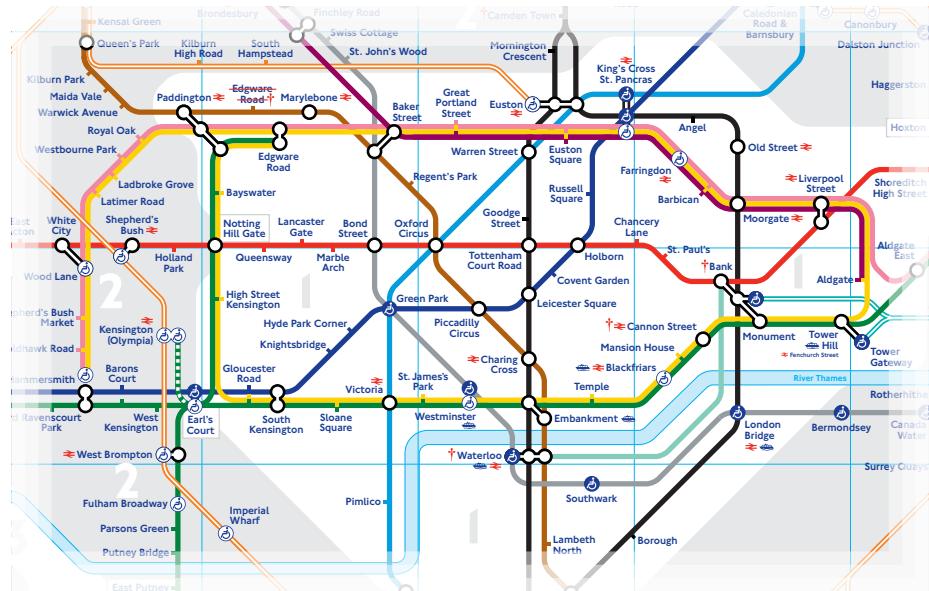


Figure 12.18: The underground system in Central London.

Consider that part of the London Underground system induced by the fifty five Zone-1 stations in [Table 12.4](#). These stations are serviced by the following ten underground lines: the Bakerloo Line (brown), the Central Line (red), the Circle Line (yellow), the District Line (green), the Hammersmith & City Line (pink), the

Jubilee Line (grey), the Metropolitan Line (purple), the Northern Line (black), the Piccadilly Line (dark blue) and the Victoria Line (light blue).

1 Angel	20 Great Portland Street	39 Paddington
2 Baker Street	21 Green Park	40 Piccadilly Circus
3 Bank	22 High Street Kensington	41 Queensway
4 Barbican	23 Holborn	42 Regent's Park
5 Bayswater	24 Hyde Park Corner	43 Russel Square
6 Blackfriars	25 King's Cross St Pancras	44 Sloane Square
7 Bond Street	26 Knightsbridge	45 South Kensington
8 Cannon Street	27 Lancaster Gate	46 Southwark
9 Chancery Lane	28 Leicester Square	47 St James's Park
10 Charing Cross	29 Liverpool Street	48 St Paul's
11 Covent Garden	30 London Bridge	49 Temple
12 Earl's Court	31 Mansion House	50 Tottenham Court Road
13 Edgware Road	32 Marble Arch	51 Tower Hill
14 Embankment	33 Marylebone	52 Victoria
15 Euston	34 Monument	53 Warren Street
16 Euston Square	35 Moorgate	54 Waterloo
17 Farringdon	36 Notting Hill Gate	55 Westminster
18 Gloucester Road	37 Old Street	
19 Goodge Street	38 Oxford Circus	

Table 12.4: Fifty five of the Zone-1 underground stations in Central London.

Tasks

1. Produce a graphical representation of the multigraph $G_{12.35}$ induced by the fifty five stations in [Table 12.4](#) within the travel network in [Figure 12.18](#), in which underground stations are denoted by vertices and underground lines by edges. If there is more than one line between two stations, this should be reflected by incorporating parallel edges into $G_{12.35}$. Use the numbers in [Table 12.4](#) to label the vertices of $G_{12.35}$.
2. Explain why both the connectivity and the edge connectivity of $G_{12.35}$ is 2.
3. Explain why the central part of the London Underground system is nevertheless an effective travel network, despite the small values of the connectivity and the edge connectivity of $G_{12.35}$.

Denote the largest number of internally disjoint paths from station i to station j in $G_{12.35}$ by $\kappa'_{G_{12.35}}(i, j)$ and the largest number of pairwise edge-disjoint paths from station i to station j in $G_{12.35}$ by $\lambda'_{G_{12.35}}(i, j)$.

4. Can $\kappa'_{G_{12.35}}(i, j) \neq \lambda'_{G_{12.35}}(i, j)$ for some pair of distinct values $i, j \in [55]$? Motivate your answer.
5. Determine, by inspection, the value of $\kappa'_{G_{12.35}}(2, 14)$. Produce $\kappa'_{G_{12.35}}(2, 14)$ internally disjoint travel routes from Baker Street to Embankment.
6. Determine, by inspection, the value of $\lambda'_{G_{12.35}}(12, 25)$. Produce $\lambda'_{G_{12.35}}(12, 25)$ pairwise edge-disjoint travel routes from Earl's Court to King's Cross St Pancras.

7. Explain in what practical sense the value of $\kappa'_{G_{12..35}}(i, j)$ is of concern to London city planners.
8. Explain in what practical sense the value of $\lambda'_{G_{12..35}}(i, j)$ is of concern to London city planners.

Further reading

- [1] G Chartrand and F Harary, 1968. *Graphs with prescribed connectivities*, pp. 61–63 in *Theory of Graphs*, Tihany, Budapest.
- [2] GA Dirac, 1960. *In abstrakten Graphen vorhandene vollständige 4-Graphen und ihre Unterteilungen*, Mathematische Nachrichten, **22**, pp. 61–85.
- [3] K Menger, 1927. *Zur allgemeinen Kurventheorie*, Fundamenta Mathematicae, **10**, pp. 96–115.
- [4] R Whitney, 1932. *Congruent graphs and the connectivity of graphs*, American Journal of Mathematics, **54**(1), pp. 150–168.



Planarity

In Central Spain, in mainly rain
Three houses stood upon the plain,
The houses of our mystery
To which, from realms of industry
Came wires with power to light and heat,
And pipes with gas to cook their meat
And other pipes with water sweet.

The owners said, “Where these things cross,
Burn, leak or short, we’ll suffer loss.
So let a graphman living near
Plan each from each to keep them clear.”

Tell them graphman, come in vain,
They’ll bear one cross that must remain
Explain the planeness of the plain.

— *Blanche Descartes*

Contents

13.1	Introduction	393
13.2	Properties of planar graphs	395
13.3	Which graphs are planar?	398
13.4	The crossing number of a graph	415
13.5	Other parameters related to planarity	420
13.6	Embedding on surfaces other than the plane	425
13.7	The Robertson-Seymour theorems	430
	Exercises	432
	Computer exercises	437
	Projects	438
	Further reading	444

13.1 Introduction

A **planar graph** is a graph that *can be drawn* in the plane in such a way that no two of its edges meet each other (intersect) except at a vertex with which they are both incident (*i.e.* there are no edge crossings). A graph that *is*

so drawn in the plane is said to be **embedded in the plane** or a **plane embedding**. A planar graph that is embedded in the plane is called a **plane graph**. Finally, a graph that is not planar is called **nonplanar**.

The complete graph K_4 in [Figure 13.1\(a\)](#) is drawn with intersecting edges (one edge crossing). Nevertheless, K_4 is a planar graph because it can be drawn in the plane without any of its edges intersecting. [Figure 13.1\(b\)](#) shows K_4 redrawn with no edges crossing. Thus, the graph in [Figure 13.1\(a\)](#), though planar, is not a plane graph: it is not an embedding of K_4 in the plane. On the other hand, the graph in [Figure 13.1\(b\)](#) is a plane graph. The process of embedding a planar graph in the plane is not unique. Indeed, a given planar graph can give rise to several different plane graphs, as we shall see later in this chapter.



Figure 13.1: Two drawings of the complete graph K_4 .

It is often not easy to determine whether or not a given graph is planar. Ponder, for a moment the following well-known puzzle, called the *three houses and three utilities problem*. Suppose we have three houses (H_1 , H_2 and H_3) and three utilities, namely *electricity* (E), *gas* (G) and *water* (W). Is it possible to connect each utility to each of the three houses without the utility connecting wires or pipes crossing? This situation may be modelled by the complete bipartite graph $K_{3,3}$ where the one partite set represents the three houses and the other the three utilities, as shown in [Figure 13.2\(a\)](#). The three houses and three utilities problem may be restated in graph theoretic terms as: Is $K_{3,3}$ a planar graph? Try as we may, we find that we cannot connect each utility to each of the three houses without causing a crossing of the utility connecting lines — there always seems to be one connection that we cannot make (see [Figure 13.2\(b\)](#)) ... but is this because $K_{3,3}$ is not planar, or is it merely because our drawing attempts are not ingenious enough?

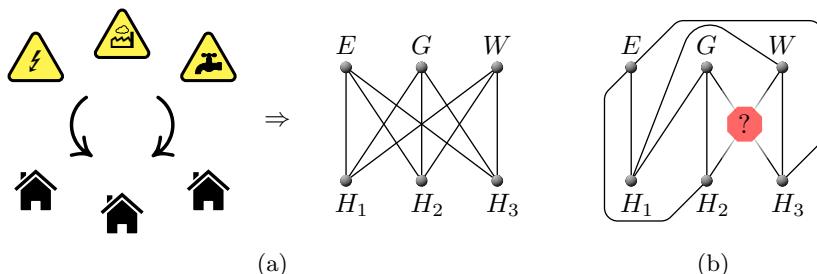


Figure 13.2: $K_{3,3}$ and the three houses and three utilities problem.

In this chapter we consider questions such as: “How can we efficiently decide whether a graph is planar?”, “How can we actually *prove* that a (nonplanar) graph is not planar?”, “Can we characterise those graphs that are planar?” and “If a

graph is not planar, what is the smallest number of edge crossings with which it may be drawn in the plane?" Extensions of these questions to embeddings of graphs on surfaces other than the plane are also considered.

We close this introduction by mentioning that planar graphs have many other, more practical applications besides solving puzzles such as the three houses and three utilities problem. One important modern application involves the layout of electric circuit boards. In the design of an electric circuit board, the objective is to locate several nodes on the board (a flat board of insulating material) and to connect certain pairs of these nodes by means of electrical wires (or conducting strips) printed directly onto the circuit board as required by the topology of the underlying electric circuit. These electrical wires may not cross, since this would lead to undesirable electrical contact or short-circuiting at crossing points. The following question arises naturally: Given an electric circuit, is it possible to connect the required nodes in such a way that no two of the connecting wires cross? In graph theoretic terms, the problem is to determine whether or not the graph associated with the electric circuit (where vertices represent nodes, and edges correspond to electrical wires connecting pairs of nodes) is planar.

❖ The reader should now be able to attempt Exercises 13.1–13.4.

13.2 Properties of planar graphs

The **faces** of a plane graph G are the connected pieces of the plane that remain after removal of the vertices and edges of G . Every plane graph contains exactly one unbounded face, which we call the **outer face**; the other faces are its **inner faces**. The vertices and the edges of G that are incident with a face F form a subgraph of G called the **boundary** of F .

To illustrate these concepts, consider the plane graph $G_{13.3}$ in Figure 13.3. This graph has five faces, labelled F_1, F_2, \dots, F_5 . The boundary of F_1 consists of the four vertices v_1, v_2, v_8, v_9 , and the four edges v_1v_2, v_2v_9, v_9v_8 and v_8v_1 . The boundary of the outer face consists of all the vertices of $G_{13.3}$, except v_9 , and the eight edges $v_1v_2, v_2v_3, v_3v_4, v_3v_5, v_5v_6, v_6v_7, v_7v_8$ and v_8v_1 . Observe that each cycle edge belongs to the boundaries of exactly two faces, while each bridge is on the boundary of only one face.

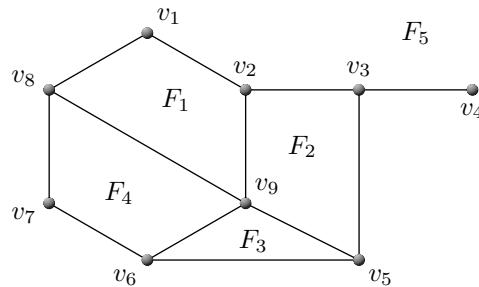


Figure 13.3: A plane graph $G_{13.3}$ with 9 vertices, 12 edges and 5 faces.

The plane graph $G_{13.3}$ in Figure 13.3 has $n = 9$ vertices, $m = 12$ edges, and $f = 5$ faces. Thus, if we subtract the number of edges from the number of vertices

and add the number of faces, we find that $n - m + f = 2$. It turns out that this equality is no coincidence; it holds for *any* connected plane graph. This well-known formula was discovered by Euler [15] in 1758 during his study of simply connected polyhedra, and is now a classical result in graph theory.

Theorem 13.1 (Euler's Formula) *If G is a connected plane graph of order n and size m with f faces, then $n - m + f = 2$.*

Proof By induction on m . If $m = 0$, then $G \cong K_1$ and so $n = 1$, $f = 1$ and $n - m + f = 2$. Hence the desired result holds for the base case, $m = 0$. Assume, as induction hypothesis, that the result holds for all connected plane graphs with fewer than $m \geq 1$ edges and let G be a connected plane graph of size m . Suppose that G has n vertices and f faces. We show that $n - m + f = 2$. If G is a tree, then $f = 1$ and, by Theorem 5.2, $m = n - 1$, so that $n - m + f = n - (n - 1) + 1 = 2$, and hence we have the desired formula. On the other hand, if G is not a tree, then G contains a cycle edge e , and $G - e$ is a connected plane graph of order n and size $m - 1$. Furthermore, the two faces incident with e in G produce one face in $G - e$, so that $G - e$ has $f - 1$ faces. Applying the induction hypothesis to $G - e$ we have $n - (m - 1) + (f - 1) = 2$, or equivalently, $n - m + f = 2$. ■

It follows from Euler's Formula that no matter how a connected planar graph is embedded in the plane, the number of faces of the resulting plane graph is always the same, thereby establishing the fact that the notion of *the number of faces of a connected planar graph* is well-defined. This would even hold if, for example, we sought to embed a planar graph on the sphere instead of in the plane. In fact, embedding a (planar) graph in the plane is topologically equivalent to embedding it on a sphere as may be seen with the aid of a so-called **stereographic projection**.

Consider a graph G that is embedded on a sphere \mathcal{S} . Position \mathcal{S} so that it is tangential to a plane \mathcal{P} at a point B of \mathcal{S} which is diametrically opposite the point A of \mathcal{S} , such that A does not coincide with any vertex or any edge of G . Then the drawing of G on \mathcal{S} may be projected onto \mathcal{P} by mapping each point $X \neq A$ of \mathcal{S} to the point X' on \mathcal{P} that intersects the line through A and X . As with plane graphs, the regions of G on the sphere \mathcal{S} are the connected pieces of \mathcal{S} that remain upon removal of the vertices and edges from \mathcal{S} . An example of such a (reversible!) stereographic projection is shown in Figure 13.4. The following interesting result is easily established by means of a stereographic projection.

Theorem 13.2 *The stereographic projection may be used to map any face of an arbitrary plane embedding of a (planar) graph G to the outer face of a (possibly) distinct plane embedding of G .*

Proof Let F be the outer face in a plane embedding of G and position the sphere \mathcal{S} as described in the paragraph above with respect to the plane \mathcal{P} so that the point A lies within F . Then the stereographic projection of $\mathcal{S} \setminus F$ onto \mathcal{P} produces a plane embedding of G in which F is mapped to the outer face. ■

The following useful result also follows immediately from the proof of Theorem 13.2 and the reversibility of the stereographic projection process described above.

Corollary 13.3 *For every vertex v of a planar graph G , there exists a plane embedding of G in which v lies on the boundary of the outer face.*

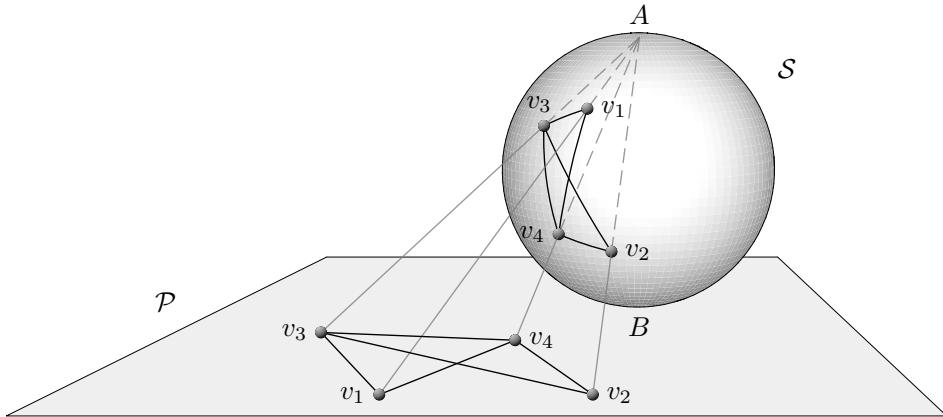


Figure 13.4: A stereographic projection of the drawing of a graph embedded on a sphere \mathcal{S} to the plane \mathcal{P} .

We next show that a planar graph of fixed order cannot have too many edges.

Theorem 13.4 *If G is a planar graph of order $n \geq 3$ and size m , then $m \leq 3n - 6$.*

Proof Consider an embedding of G in the plane, resulting in f faces. Every edge lies on the boundary of either one or two faces. Therefore, if the number of edges on the boundary of a face is summed over all the faces, the result is at most $2m$. The boundary of every face, however, contains at least three edges, and so the sum is at least $3f$. Hence, $3f \leq 2m$, or $f \leq 2m/3$. Applying [Theorem 13.1](#), we obtain $2 = n - m + f \leq n - m + 2m/3 = n - m/3$. Therefore, $m \leq 3n - 6$. ■

Another important property of a planar graph is that its minimum degree cannot be too large either, as made more precise in the following result.

Theorem 13.5 *Every planar graph contains a vertex of degree at most 5.*

Proof Let G be a planar graph of order n and size m with vertex set $\{v_1, \dots, v_n\}$. If $n \leq 6$, then no vertex of G has a degree exceeding 5. Assume, therefore, that $n \geq 7$. By [Theorem 13.4](#), we have that $m \leq 3n - 6$ and hence

$$\sum_{i=1}^n d(v_i) = 2m \leq 6n - 12 < 6n. \quad (13.1)$$

If, however, each vertex of G has a degree exceeding 5, then

$$\sum_{i=1}^n d(v_i) \geq 6n,$$

which contradicts (13.1). This contradiction shows that the degree of at least one vertex of G is at most 5. ■

We are now in a position to return to our question in the introduction of whether or not the graph $K_{3,3}$ is planar.

Theorem 13.6 *The graph $K_{3,3}$ is nonplanar.*

Proof Suppose, to the contrary, that $K_{3,3}$ is planar, and consider any embedding of $K_{3,3}$ in the plane, resulting in f faces. Since $K_{3,3}$ is a bipartite graph, it contains no triangles (see Theorem 2.3). Therefore, the boundary of every face contains at least four edges. Let x be the number of edges on the boundary of a face, summed over all f faces. Then, $x \geq 4f$. Since $K_{3,3}$ contains no bridges, however, every edge lies on the boundary of exactly two faces. Thus, the sum x counts each edge twice; that is, $x = 2m = 18$. Therefore, $4f \leq x = 18$, or equivalently, $f \leq \frac{9}{2}$. Consequently, $f \leq 4$. But, by Theorem 13.1, $f = 2 + m - n = 2 + 9 - 6 = 5$. This contradiction shows that $K_{3,3}$ is nonplanar. ■

Theorem 13.6 shows that in the three houses and three utilities problem it is impossible to connect each utility with each of the three houses without the utility lines crossing. As we shall see later in this chapter, $K_{3,3}$ plays a major role in the study of planar graphs, as does the complete graph K_5 , which is also nonplanar as we prove next.

Theorem 13.7 *The graph K_5 is nonplanar.*

Proof Suppose, to the contrary, that K_5 is planar. Since K_5 has $n = 5$ vertices and $m = 10$ edges, we have that $10 = m > 3n - 6 = 9$, which contradicts the result of Theorem 13.4. This contradiction shows that K_5 is nonplanar. ■

We remark that if a graph G contains a nonplanar subgraph H , then G is itself nonplanar, for otherwise a plane embedding of H would result from a plane embedding of G . In particular, we have the following immediate consequence of Theorems 13.6 and 13.7.

Corollary 13.8 *The graph $K_{r,s}$ is nonplanar for all $r \geq s \geq 3$, and the graph K_n is nonplanar for all $n \geq 5$.*

❖ The reader should now be able to attempt Exercises 13.5–13.22.

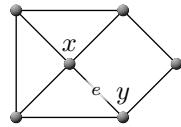
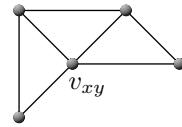
13.3 Which graphs are planar?

In this section we present two classic theorems, due to Kazimierz Kuratowski and Klaus Wagner, which characterise planar graphs, and we illustrate the working of an efficient algorithm for planarity testing. In order to present these results, however, we require the notions of a contraction, a subdivision and a graph minor.

13.3.1 Contractions, subdivisions and graph minors

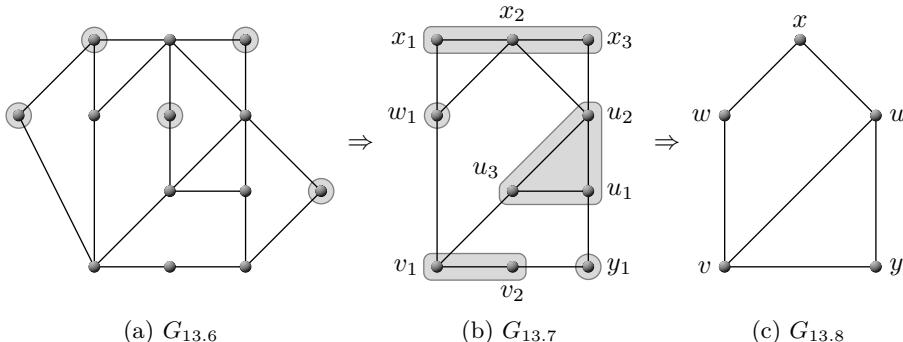
A useful operation in graph theory is that of contracting an edge into a vertex. Let $e = xy$ be an edge of a graph G . By **contracting** the edge e we mean replacing the vertices x and y in G by a new vertex v_{xy} and joining v_{xy} to all vertices that were adjacent to x or y in G . The resulting graph is called a **contraction** of G . We denote the graph obtained from G by contracting the edge e by G/e . In Figure 13.5, the graph $G_{13.5} = G_{13.4}/e$ is a contraction of $G_{13.4}$, obtained by contracting the edge $e = xy$.

Consider two graphs G and H . If the vertex set of H can be partitioned into sets $\{V_x \mid x \in V(G)\}$ such that $H[V_x]$ is connected and there is an edge joining two sets V_x and V_y in H if and only if $xy \in E(G)$, then we call H a **G -minor**

(a) $G_{13.4}$ (b) $G_{13.5}$ **Figure 13.5:** The graph $G_{13.5} = G_{13.4}/e$ is a contraction of the graph $G_{13.4}$.

graph and we call the sets V_x the **branch sets** of G . Intuitively, we obtain G from the graph H by contracting every branch set to a single vertex and deleting any parallel edges or loops that may arise. If F is a graph that contains a G -minor subgraph, then we call G a **minor** of F and write $G \preccurlyeq F$.

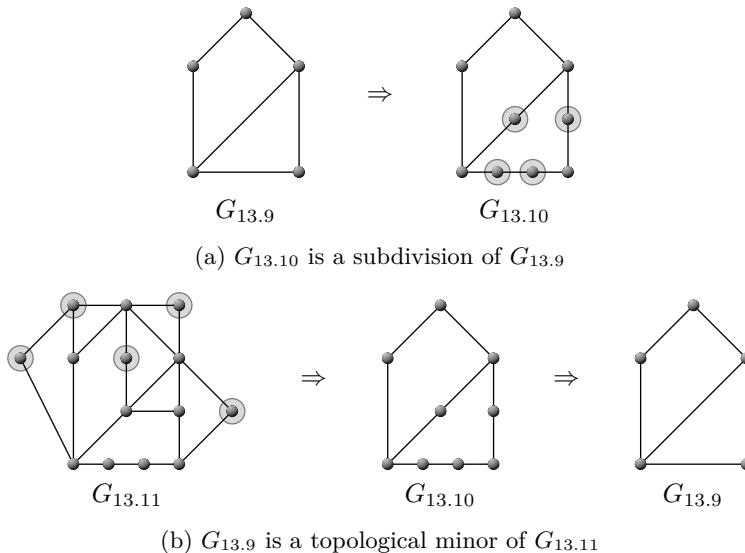
The graph $G_{13.7}$ in [Figure 13.6](#) is a subgraph of $G_{13.6}$, and $G_{13.7}$ is also a $G_{13.8}$ -minor graph with branch sets $V_u = \{u_1, u_2, u_3\}$, $V_v = \{v_1, v_2\}$, $V_w = \{w_1\}$, $V_x = \{x_1, x_2, x_3\}$ and $V_y = \{y_1\}$ corresponding to the vertices u, v, w, x and y of $G_{13.7}$. Hence $G_{13.6}$ contains a $G_{13.8}$ -minor subgraph (namely $G_{13.7}$), and so $G_{13.8}$ is a minor of $G_{13.6}$.

**Figure 13.6:** A minor $G_{13.8}$ of a graph $G_{13.6}$ together with a $G_{13.8}$ -minor graph $G_{13.7}$.

We remark that the graph G/e obtained by contracting a single edge $e = xy$ of a graph G is a minor of G , since the vertex set of G may be partitioned into branch sets with $\{x, y\}$ as one of the branch sets and with every other branch set consisting of a single vertex. Proof of the following more general observation is left as an exercise.

Theorem 13.9 *A graph H is a minor of a graph G if and only if H can be obtained from a subgraph of G by a series of edge contractions.*

An **elementary subdivision** of a nonempty graph G is the operation of removing some edge uv from G and adding a new vertex w together with the edges uw and vw to the resulting graph. We say that the edge uv has been **subdivided**. A **subdivision** of G is a graph obtained from G by a sequence of elementary subdivisions (including the possibility of none). A subdivision of G may therefore be thought of as a graph obtained by inserting vertices (of degree 2) into some of the edges of G . The graph $G_{13.10}$ in [Figure 13.7\(a\)](#) is a subdivision of the graph $G_{13.9}$.

**Figure 13.7:** Subdivisions and topological minors.

If H is a subdivision of a graph G , and if H is a subgraph of another graph F , then we say that G is a **topological minor** of F . We call the vertices of G in H the **branch vertices** of H ; the other vertices of H we call its **subdividing vertices**. The graph $G_{13.10}$ in Figure 13.7(b) is an example of a topological minor of the graph $G_{13.11}$.

Note that if a graph G is planar (nonplanar, respectively), then any subdivision of G is clearly planar (nonplanar, respectively). Furthermore, if a graph G contains a nonplanar subgraph, then G is nonplanar. Hence, we obtain the following result as a consequence of Theorems 13.6 and 13.7.

Theorem 13.10 *Any graph that contains K_5 or $K_{3,3}$ as a topological minor is nonplanar.*

❖ The reader should now be able to attempt Exercises 13.23–13.28.

13.3.2 Kuratowski's characterisation of planar graphs

A **minimal nonplanar graph** is a nonplanar graph with the property that each of its proper subgraphs is planar. We start our exposition of **Kuratowski's characterisation of planar graphs** by showing that every minimal nonplanar graph is connected and that no minimal nonplanar graph contains a cut-vertex.

Lemma 13.11 *Every minimal nonplanar graph is 2-connected.*

Proof Let G be a minimal nonplanar graph. If G were disconnected, then at least one component of G would be nonplanar, contradicting the minimality of G . Hence, G is connected. Suppose that G has a cut-vertex v . Let G_1, G_2, \dots, G_k be the components of $G - v$. For $i \in [k]$, let $H_i = G[V(G_i) \cup \{v\}]$. By the minimality of G , each subgraph H_i of G is planar. It follows from Corollary 13.3 that there is a planar embedding of each graph H_i with v on the boundary of the outer face.

Kazimierz Kuratowski was born in Warsaw, the Russian Empire (now Poland) on 2 February 1896. His studies at the University of Glasgow were interrupted by the outbreak of World War I, but he obtained a doctorate in mathematics at the University of Warsaw in 1921 and was appointed professor of mathematics at the Technical University of Lvov in 1927, where he worked with Banach and Ulam on some important results in measure theory. In 1934, he moved to the University of Warsaw, where he remained until his retirement. Poland had made a remarkable leap forward in mathematical teaching and research between the two world wars, but the entire education system was destroyed during World War II. Kuratowski played a major role as leader of the rebuilding process of the Polish education system. His main work was in the areas of topology and set theory, and he is best known for his characterisation of planar graphs (see [Theorem 13.14](#)). Kuratowski was honoured with many prizes and elections to academies. He died in Warsaw on 18 June 1980.



Biographic note 40: Kazimierz Kuratowski (1896–1980)

By merging these embeddings of the graphs H_i at the vertex v we obtain a planar embedding of G , contradicting the fact that G is nonplanar. This contradiction shows that G cannot have a cut-vertex; hence, G is 2-connected. ■

We require the following lemma in order to show that a connected nonplanar graph of minimum size containing neither K_5 nor $K_{3,3}$ as a topological minor is, in fact, 3-connected.

Lemma 13.12 *If G is a nonplanar graph with $\kappa(G) = 2$ such that $G = G_1 \cup G_2$, where $V(G_1) \cap V(G_2) = \{x, y\}$, then $G_1 + xy$ or $G_2 + xy$ is nonplanar.*

Proof For $i = 1, 2$, let $F_i = G_i + xy$ and suppose that F_1 and F_2 are both planar. Then it follows by [Theorem 13.2](#) that there is a planar embedding of F_1 and F_2 , in which xy is a boundary edge of the outer face. Combining these drawings of F_1 and F_2 by merging them along the edge xy produces an embedding of the graph $G + xy$ in the plane. If xy is already an edge of G , then we have an embedding of G in the plane. If xy is not an edge of G , then deleting this edge from the embedding of $G + xy$ in the plane once again produces an embedding of G in the plane. Both cases produce a contradiction, showing that at least one of F_1 or F_2 is nonplanar. ■

We are now finally in a position to prove our last intermediate result before presenting a formulation of [Kuratowski's characterisation of planar graphs](#).

Lemma 13.13 *A connected nonplanar graph of minimum size containing neither K_5 nor $K_{3,3}$ as a topological minor is 3-connected.*

Proof If G is a connected nonplanar graph of minimum size that contains neither K_5 nor $K_{3,3}$ as a topological minor, then the graph $G - e$ contains neither K_5 nor $K_{3,3}$ as a topological minor for any edge e of G . Hence, by our choice of G , the graph $G - e$ is planar. Thus, any proper subgraph of G is planar, implying that G is a minimal nonplanar graph. Hence, by Lemma 13.11, G is 2-connected.

Suppose that $\kappa(G) = 2$ and let $\{x, y\}$ be a cut-set of G . Then, $G = G_1 \cup G_2$ with $V(G_1) \cap V(G_2) = \{x, y\}$. For $i = 1, 2$, let $F_i = G_i + xy$. It follows from Lemma 13.12 that at least one of F_1 or F_2 is nonplanar. We may assume, without loss of generality, that F_1 is nonplanar. Since F_1 has fewer edges than does G , the graph F_1 contains K_5 or $K_{3,3}$ as a topological minor by our choice of the graph G . Hence, F_1 contains a subgraph H that is a subdivision of K_5 or $K_{3,3}$. If H is a subgraph of G , then G contains a subdivision of K_5 or $K_{3,3}$, a contradiction. Since $H - xy$ is a subgraph of G , it follows that $xy \in E(H) \setminus E(G)$. But replacing the edge xy in H by an x - y path in G_2 produces a subgraph of G that is a subdivision of K_5 or $K_{3,3}$, again a contradiction. We conclude from these contradictions that $\kappa(G) \geq 3$. ■

Surprisingly, Kuratowski [29] proved that the converse of Theorem 13.10 is true by showing that the only obstacle to the planarity of a graph G is the presence of K_5 or $K_{3,3}$ as a topological minor of G , as stated in the following theorem.

Theorem 13.14 (Kuratowski's Theorem) *A graph is planar if and only if it contains neither K_5 nor $K_{3,3}$ as a topological minor.*

As shown in Lemma 13.13, we can reduce the general case of Kuratowski's Theorem to the 3-connected case, which is the heart of the proof. To prove Kuratowski's Theorem for 3-connected graphs, we require the notion of a convex embedding of a planar graph. A **convex embedding** of a planar graph in the plane is an embedding of the graph in the plane such that each inner face is convex. In particular, in a convex embedding of a planar graph, every edge can be drawn as a straight line segment. To prove Kuratowski's Theorem for 3-connected graphs, we actually prove the following stronger result due to Tutte [47, 48].

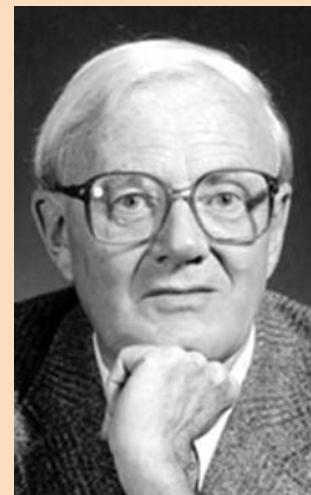
Theorem 13.15 (Tutte's Theorem) *If G is a 3-connected graph containing neither K_5 nor $K_{3,3}$ as a topological minor, then G has a convex embedding in the plane.*

Before presenting a proof of Tutte's Theorem, however, we first present three lemmas to facilitate the proof. The first lemma is due to Thomassen [44].

Lemma 13.16 (Thomassen's Lemma) *If G is a 3-connected graph of order at least 5, then there exists an edge e of G such that the graph G/e is again 3-connected.*

Proof Suppose, to the contrary, that there is no such edge e . Then, for every edge $e = xy \in E(G)$, the graph G/e contains a cut-set of size at most 2. Since $\kappa(G) \geq 3$, such a cut-set has size exactly 2 and contains the contracted vertex v_{xy} of G/e . Let z be the other vertex of this set. Then, any two vertices separated by $\{v_{xy}, z\}$ in G/e are separated by $\{x, y, z\}$ in G . Since $\kappa(G) \geq 3$, $\{x, y, z\}$ is a minimal separating set of G , and so each of x, y and z has a neighbour in every component of $G - \{x, y, z\}$.

William Thomas Tutte was born in Newmarket (Suffolk, England) on 14 May 1917. In 1935, he went up to Cambridge to study Chemistry until 1941, when he joined the cryptographic code breaking team at Bletchley Park during World War II. At Bletchley Park he was responsible for breaking the German cipher FISH. After the war he returned to Trinity College, Cambridge, but this time to study mathematics. Upon obtaining his doctorate in 1948 he was invited to take up a teaching position at the University of Toronto in Canada. In 1962, he joined the faculty at the newly established University of Waterloo where he remained until his retirement in 1984. There he established the Department of Combinatorics and Optimisation, which he helped develop into a world-class department. He is best known for his work on graph connectivity and hamiltonicity in graphs and matroids. He died in Waterloo on 2 May 2002.



Biographic note 41: William Tutte (1917–2002)

Among all the edges of G , choose the edge $e = xy$ and the vertex z such that the resulting disconnected graph $G - \{x, y, z\}$ has a component H with the largest possible number of vertices. Let u be a neighbour of z that belongs to a component of $G - \{x, y, z\}$ different from H . By assumption, G/uz is not 3-connected, and so there is a vertex v such that $\{u, v, z\}$ is a cut-set of G . We claim that some component of $G - \{u, v, z\}$ is larger than H .

Since each of x and y has a neighbour in H , the subgraph F of G induced by $V(H) \cup \{x, y\}$ is connected. If $v \notin V(F)$, then (since u and z are also not in $V(F)$) the graph F is contained in some component of $G - \{u, v, z\}$, contradicting the maximality of H . We may therefore assume that $v \in V(F)$ (possibly, $v \in \{x, y\}$). If $F - v$ is connected, then $F - v$ is contained in some component of $G - \{u, v, z\}$ and $|V(F - v)| = |V(H)| + 1 > |V(H)|$, a contradiction. We may therefore also assume that $F - v$ is disconnected. Since x and y are adjacent, at least one component C of $F - v$ contains neither x nor y . But then in $G - \{v, z\}$, the vertex u is disconnected from each vertex in $V(C)$, and so $G - \{v, z\}$ is disconnected, contradicting the 3-connectedness of G . ■

The next lemma shows that if a graph contains neither K_5 nor $K_{3,3}$ as a topological minor, then this property is preserved if we contract any edge of the graph.

Lemma 13.17 *Let G be a graph and let e be any edge of G . If G/e contains K_5 or $K_{3,3}$ as a topological minor, then G also contains K_5 or $K_{3,3}$ as a topological minor.*

Proof Let $e = xy$ and let v_{xy} be the contracted vertex of G/e . Suppose G/e contains K_5 or $K_{3,3}$ as a topological minor. Then G/e contains a subgraph H that is a subdivision of K_5 or $K_{3,3}$. We show that G also contains a subgraph that is

Carsten Thomassen was born in Grindsted, Denmark on 22 August 1948. He obtained his doctorate in mathematics at the University of Waterloo in 1976 under the supervision of Daniel Younger. He currently teaches discrete mathematics (more specifically, graph theory) at the Technical University of Denmark. Thomassen is editor of a number of international mathematics journals (including the Journal of Graph Theory, the Electronic Journal of Combinatorics, the journal Combinatorica, the Journal of Combinatorial Theory, Discrete Mathematics, and the European Journal of Combinatorics) and is the recipient of numerous honours and awards. He is also included on the ISI Web of Knowledge list of the 250 most cited mathematicians worldwide.



Biographic note 42: Carsten Thomassen (1948–present)

a subdivision of K_5 or $K_{3,3}$; that is, we show that G contains K_5 or $K_{3,3}$ as a topological minor.

If $v_{xy} \notin V(H)$, then H is a subgraph of G , and the desired result follows. Hence, we may assume $v_{xy} \in V(H)$. Suppose v_{xy} is not a branch vertex of H . Let u and v be the two neighbours of v_{xy} in H . If x or y , say x , is adjacent in G to both u and v , then removing v_{xy} from H and adding x , as well as the edges ux and vx , produces a subgraph of G that is a subdivision of K_5 or $K_{3,3}$, as desired. On the other hand, if each of x and y is adjacent in G to exactly one of u and v , say $\{ux, vy\} \subset E(G)$, then removing v_{xy} from H and adding the vertices x and y , together with the edges ux , xy and vy , produces a subgraph of G that is a subdivision of K_5 or $K_{3,3}$, as desired. Hence, we may assume that v_{xy} is a branch vertex of H .

If x is adjacent in G to all the neighbours of v_{xy} in H , then removing v_{xy} from H and adding x and the edges joining x to the neighbours of v_{xy} in H produces a subgraph of G that is a subdivision of K_5 or $K_{3,3}$, as desired.

If x is adjacent to all but one of the neighbours of v_{xy} in H , then removing v_{xy} from H and adding x and y , the edges joining x to all but one of the neighbours of v_{xy} in H and the edge joining y to the remaining neighbour of v_{xy} in H , produces a subgraph of G that is a subdivision of K_5 or $K_{3,3}$ (with x as the branch vertex corresponding to v_{xy}), as desired.

Hence, we may assume that neither x nor y is adjacent to at least two of the neighbours of v_{xy} in H . This implies that H is subdivision of K_5 and each of x and y is adjacent in G to exactly two of the four vertices adjacent to v_{xy} in H . Let x'_1 and x'_2 (y'_1 and y'_2 , respectively) be the two neighbours of v_{xy} in H that are adjacent in G to y (x , respectively). If x'_1 is a branch vertex in H , let $x_1 = x'_1$; otherwise, let x_1 be the branch vertex of H such that the $v_{xy}-x_1$ path contains the vertex x'_1 . Let x_2 , y_1 and y_2 be similarly defined.

Let F be the subgraph of G obtained from H by removing the vertex v_{xy} , adding the vertices x and y , adding the edges xy , xy'_1 , xy'_2 , yx'_1 and yx'_2 , and deleting the internal vertices, if any, and edges of the x_1-x_2 path and the y_1-y_2 path

in H that contain no branch vertex. Then, the graph F is a subdivision of $K_{3,3}$ in G , in which x , x_1 and x_2 are the branch vertices of the one partite set and y , y_1 and y_2 are the branch vertices of the other partite set. Hence G contains $K_{3,3}$ as a topological minor. ■

Our final intermediate result states that face boundaries consisting solely of cycles is a property of 2-connected graphs (and no other graphs).

Lemma 13.18 *Every face of a plane embedding of a graph G has a cycle as its boundary if and only if G is 2-connected.*

Proof Suppose the boundary of some face f of a plane embedding of G is not a cycle. Then the boundary of the face f is another type of closed walk containing some vertex, v say, twice. Hence, there is a simple closed curve C in the plane that starts and ends at the vertex v such that C leaves the vertex v between two edges of the boundary of the face f , remains entirely within f , and later returns to v between a different pair of edges, and intersects G only at v . Thus the curve C separates the plane into two faces, both of which contain parts of the graph G . Since the curve C intersects G only at v , it follows that v is a cut-vertex of G , in which case G is not 2-connected.

Conversely, suppose G has a cut-vertex v (*i.e.* is not 2-connected). Then G is the union of two graphs F and H having only the vertex v in common. In any plane embedding of G , there is a face f whose boundary contains an edge e of H incident with v followed immediately by an edge h of F also incident with v . We now traverse the boundary of f starting at v and then proceed along e . Since e is an edge of H and v is a cut-vertex, no edges in F are encountered until the boundary has returned to v . Since h is an edge of F and is in the boundary, and since the boundary is a closed walk ending at v , it follows that the boundary of f meets v at least twice; that is, the boundary of f is not a cycle. ■

We are now in a position to prove [Tutte's Theorem](#).

Proof of Theorem 13.15 We proceed by induction on the order n of a 3-connected graph G . If $n \leq 4$, then $G = K_4$, which has a convex embedding. This establishes the base case. Assume, as induction hypothesis, that $n \geq 5$ and that every 3-connected graph of order less than n containing neither K_5 nor $K_{3,3}$ as a topological minor has a convex embedding in the plane. Let G be a 3-connected graph of order n that contains neither K_5 nor $K_{3,3}$ as a topological minor.

By [Lemma 13.16](#), there exists an edge $e = xy$ of G such that the graph $H = G/e$ is again 3-connected. Let v_{xy} be the contracted vertex of H . By [Lemma 13.17](#), the graph H contains neither K_5 nor $K_{3,3}$ as a topological minor. Applying the induction hypothesis to the 3-connected graph H of order $n - 1$, there is a convex embedding of H in the plane. Consider such an embedding in which all edges from v_{xy} to its neighbours are straight line segments.

The plane graph obtained by deleting the edges incident with v_{xy} has a face (possibly the outer face) containing v_{xy} . Since H is 3-connected, it follows that $H - v_{xy}$ is 2-connected. Hence, by [Lemma 13.18](#), the face of $H - v_{xy}$ which contained v_{xy} is a cycle, C say. Since H is 3-connected, the vertex v_{xy} has at least three neighbours in H . Each neighbour of v_{xy} in H is adjacent in G to the vertex x or the vertex y , or to both x and y .

Suppose that x and y have three common neighbours a, b and c on C . Then the subgraph of G obtained from the cycle C by adding the vertices x and y , adding the edge xy and adding the six edges between $\{x, y\}$ and $\{a, b, c\}$ is a subdivision of K_5 , in which a, b, c, x and y are the branch vertices. Hence, G contains K_5 as a topological minor, a contradiction. Thus, there are at most two vertices on C adjacent to both x and y in G .

Suppose that the vertices a and c on C are adjacent to x in G , and that the vertices b and d on C are adjacent to y in G , such that the vertices occur in the order a, b, c, d on C . Then the subgraph of G obtained from the cycle C by adding the vertices x and y , and adding the five edges ax, by, cx, dy and xy is a subdivision of $K_{3,3}$, in which a, c and y are the branch vertices of the one partite set and b, d and x are the branch vertices of the other partite set. Hence, G contains $K_{3,3}$ as a topological minor, a contradiction.

Thus, there exist neighbours a and b of x on C in G such that all neighbours of x on C in G occur on a single segment from a to b on C and all neighbours of y on C in G occur on the other segment from a to b on C . We now obtain a convex embedding of G by placing x at v_{xy} in H and placing y in the wedge formed by ax, ab and bx . ■

- ❖ The reader should now be able to attempt Exercises 13.29–13.30.

13.3.3 Wagner's characterisation of planar graphs

In this section, we present a second characterisation of planar graphs, due to **Wagner** [52], which is often referred to as **Wagner's Theorem**.

Klaus Wagner was born on 31 March 1910. In 1937, he obtained a doctorate in topology at the University of Cologne where he also taught for many years. In 1970, he moved to the University of Duisburg where he retired in 1978. He is best known for his contributions in topological graph theory. His most famous result is his characterisation of planar graphs in terms of a forbidden set of graph minors (see [Theorem 13.19](#)). He also proved that a 4-connected graph is planar if and only if it does not contain K_5 as a minor. He is reported to have conjectured during the 1930s that in any finite set of graphs, one graph is isomorphic to a minor of another. This conjecture was proven correct in 2004 by **Neil Robertson** and **Paul Seymour**, and implies that any family of graphs that is closed under the minor ordering of graphs (as planar graphs are) can be characterised by a finite obstruction set. He died on 6 February 2000.



Biographic note 43: Klaus Wagner (1910–2000)

Theorem 13.19 (Wagner's Theorem) *A graph is planar if and only if it contains neither K_5 nor $K_{3,3}$ as a minor.*

Wagner’s Theorem is an immediate consequence of Kuratowski’s Theorem and the following result.

Lemma 13.20 *A graph contains K_5 or $K_{3,3}$ as a topological minor if and only if it contains K_5 or $K_{3,3}$ as a minor.*

Proof It is an easy exercise (see Exercise 13.27) to show that every topological minor of a graph is also its (ordinary) minor. This establishes the necessity. Furthermore, it is also an easy exercise (see Exercise 13.28) to show that if a graph contains $K_{3,3}$ as a minor, then it contains $K_{3,3}$ as a topological minor. Hence, it suffices to show that every graph containing K_5 as a minor also contains K_5 or $K_{3,3}$ as a topological minor.

Let G be a graph and suppose $K_5 \preccurlyeq G$. Among all K_5 -minor subgraphs of G , let H be a minimal such subgraph; that is, no proper subgraph of H is a K_5 -minor graph of G . Then every branch set of H induces a tree in H , and there is exactly one edge in H between any two branch sets. For each branch set V_x , let T_x be the subgraph of H obtained by adding to the tree $H[V_x]$ the four edges (and their incident vertices) joining it to the other four branches. Then, T_x is a tree. By the minimality of H , the tree T_x has exactly four leaves, namely the four vertices of T_x that lie in the branch sets different from V_x .

If each of the five trees T_x is a subdivision of $K_{1,4}$, then H is a subdivision of K_5 ; that is, G contains K_5 as a topological minor. On the other hand, suppose one of the trees T_x is not a subdivision of $K_{1,4}$. Then T_x has exactly two vertices of degree 3. Contracting V_x onto these two vertices, and contracting every other branch set to a single vertex, we obtain $K_{3,3}$ plus two edges (one edge in each partite set). It follows that $K_{3,3} \preccurlyeq G$, and so, by Exercise 13.28, the graph G contains $K_{3,3}$ as a topological minor. ■

❖ The reader should now be able to attempt Exercise 13.31.

13.3.4 A planarity testing algorithm

In 1961, Auslander and Perter [2] proved that the problem of deciding whether or not a graph is planar is, in fact, a polynomial time decision problem (*i.e.* is in the class P, in the notation of Chapter 3) by producing an algorithm of worst-case time complexity $\mathcal{O}(n^3)$ for solving this decision problem for a graph of order n . During the mid-1960s, however, two more efficient $\mathcal{O}(n^2)$ algorithms for planarity testing were put forward, one by Demoucron *et al.* [13] in 1964 and the other by Lempel *et al.* [30] in 1967. Moreover, both these algorithms are capable of producing a plane embedding of the input graph G if G is, in fact, planar — something that the original algorithm of Auslander and Perter cannot do. In a significant breakthrough during the mid-1970s, three planarity testing algorithms of $\mathcal{O}(n)$ worst-case time complexity were published, one by Hopcroft and Tarjan [27] in 1974 and another two (independently) by Even and Tarjan [16] and by Booth and Luecker [8], both in 1976. None of these linear algorithms are, however, capable of producing plane embeddings of planar graphs; they merely produce the binary output `true` or `false` to the question of whether the input graph is planar. Then, in 1985, Chiba *et al.* [10] improved the original 1967 algorithm of Lempel *et al.* [30], making it both linear and capable of providing a plane embedding if the input graph is planar. In 1993, Mehlhorn *et al.* [31] did the same for the 1974 algorithm of

[Hopcroft](#) and [Tarjan](#). An interesting additional capability appeared in the 1999 planarity testing algorithm of [Shih and Hsu](#) [41], which has a linear worst-case time complexity and produces either a plane embedding (if the input graph is planar) or explicit [Kuratowski](#) subgraphs (if the input graph is nonplanar). This algorithm is based on a depth-first search approach (see [Section 5.5](#)) and is similar to the algorithm by [Boyer and Myrvold](#) [9], which also has a linear worst-case time complexity and was also published (independently) in 1999.

As we shall see in this section, the blocks of a graph play a central role in deciding whether the entire graph is planar. In this section we present the simple and very popular 1964 algorithm of [Demoucron et al.](#) [13] for determining whether or not a block of order n is planar.

Recall, from the previous chapter, that a **block** of a graph G is a maximal subgraph of G containing no cut-vertices. The blocks of G partition the edge set of G and any two blocks have at most one vertex in common. Furthermore, if two blocks of G share a vertex, then this vertex is necessarily a cut-vertex of G . Finally, a block of G that contains exactly one cut-vertex of G is called an **end-block** of G . Consider, as an example, the graph $G_{13,12}$ shown in [Figure 13.8](#). This graph has six blocks, as indicated in the figure: B_1 and B_4 are complete graphs of order 2, B_2 is a complete graph of order 4, B_3 is the Petersen graph, B_5 is the cube and B_6 is a triangle. Of these blocks, B_1 , B_4 and B_6 are end-blocks.

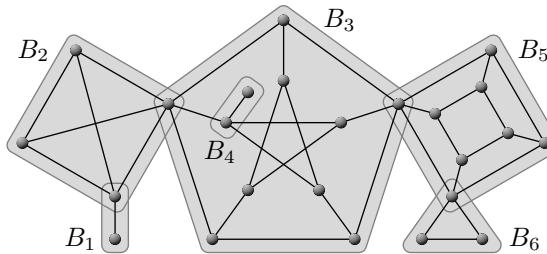


Figure 13.8: The graph $G_{13,12}$, which has six blocks, B_1, \dots, B_6 .

The following result shows that the blocks of a graph G are the fundamental subgraphs of G to consider when attempting to resolve the decision problem of whether or not G is planar.

Theorem 13.21 *A connected graph is planar if and only if each of its blocks is planar.*

Proof If G is a planar graph, then certainly each of its blocks is planar. We prove the converse by induction over the number of blocks of G . As base case for the induction process, observe that if G contains only one block B , which is planar, then $B = G$ and hence G must itself be planar. Assume, as induction hypothesis, that every graph with $b \geq 1$ blocks, each of which is planar, is itself a planar graph. Now consider, for the induction step, a graph G with $b + 1$ blocks, all of which are planar. Let B' be an end-block of G and let v be the cut-vertex of G that belongs to B' . Remove from G all vertices of B' different from v and call the resulting graph G' . Then it follows from the induction hypothesis that G' is a planar graph. Since B' is planar, it follows from [Corollary 13.3](#) that B' may be embedded as a plane block so that v lies on the boundary of the outer face. This plane block

may now be placed in any face of G' containing v on its boundary so that the two vertices of G' and B' labelled v coincide, thus producing a plane embedding of G and completing the induction step. ■

It follows from Theorems 13.11 and 13.21 that, in order to verify whether or not a given graph G is planar, we need only verify whether each of its blocks is planar. The notion of a so-called *H-fragment* (also sometimes called an *H-bridge*) of a graph plays an important role in the algorithm put forward by Demoucron *et al.* [13] for this verification process, and this notion stems from the early short-proof alternative for Kuratowski's Theorem by Dirac and Schuster [14] dating back to 1954. Define, for any subgraph H of a block G , a binary relation \sim on the edges of G that are not in H , denoted by $e \sim f$ if there exists a walk W in $G - E(H)$ whose first and last edges are e and f , and where no internal vertex of W belongs to H . Then the relation \sim is an equivalence relation on the edges of G that are not in H (see Exercise 13.32). The subgraphs of $G - E(H)$ induced by the resulting equivalence classes are called the **H -fragments** of G . The vertices of an H -fragment S that are also vertices of H are called the **vertices of attachment** of S . An **admissible face** of an H -fragment S is a face of H whose boundary contains all the vertices of attachment of S . The set of admissible faces of an H -fragment S is denoted by $\mathcal{F}_H(S)$.

Consider, as an example, the graph $G_{13.13}$ in Figure 13.9(a). Since $G_{13.13}$ contains no cut-vertex, it is clear that $G_{13.13}$ is itself a block. Consider the subgraph H_0 of $G_{13.13}$ induced by the vertices v_1, v_4 and v_5 shown in Figure 13.9(b), i.e. the triangle shown in bold face in Figure 13.9(a). The block $G_{13.13}$ has two H_0 -fragments S_1 and S_2 , as shown in Figure 13.9(c)–(d). The vertices of attachment of each H_0 -fragment are highlighted. If F_1 is the outer face of H_0 and F_2 is its inner face, as shown in Figure 13.9(b), then both these faces contain all the vertices of attachment of both H_0 -fragments. Hence $\mathcal{F}_{H_0}(S_1) = \mathcal{F}_{H_0}(S_2) = \{F_1, F_2\}$.

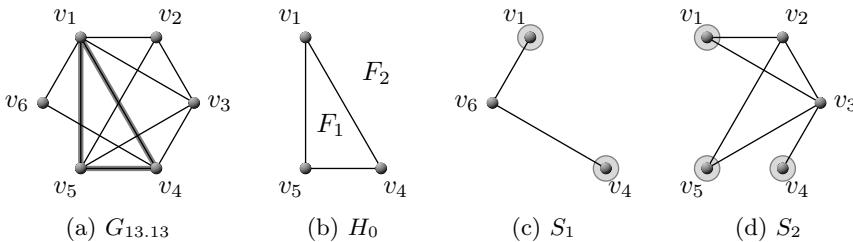


Figure 13.9: (a) A block $G_{13.13}$, (b) a subgraph H_0 of $G_{13.13}$, and (c)–(d) the two H_0 -fragments S_1 and S_2 of $G_{13.13}$.

An **extremal path** of an H -fragment S of a block G is a path in S starting at one of its vertices of attachment and ending at another of its vertices of attachment. Note that since G is a block, each of its H -fragments necessarily has at least two vertices of attachment and hence at least one extremal path, no matter which subgraph of G is chosen as H . The H_0 -fragment S_1 of $G_{13.13}$ in Figure 13.9(c) only has one extremal path, namely $v_1 v_6 v_4$, while the H_0 -fragment S_2 in Figure 13.9(d) has the five distinct extremal paths $v_1 v_2 v_5$, $v_1 v_3 v_5$, $v_1 v_2 v_3 v_5$, $v_1 v_3 v_4$ and $v_1 v_2 v_3 v_4$.

Two different H -fragments S and T of G are said to **compete** if (i) $\mathcal{F}_H(S) \cap \mathcal{F}_H(T) \neq \emptyset$, and (ii) for each face in $\mathcal{F}_H(S) \cap \mathcal{F}_H(T)$ there is an extremal path of S and an extremal path of T which cannot both be embedded into this face. The two H_0 -fragments S_1 and S_2 of $G_{13.13}$ in Figure 13.9(c)–(d) are not competing; simultaneous embeddings into the outer face F_1 of H_0 of the only extremal path of the H_0 -fragment S_1 together with each of the five distinct extremal paths of the H_0 -fragment S_2 are shown in Figure 13.10(a)–(e), respectively. The same can be done for the inner face F_2 of H_0 .

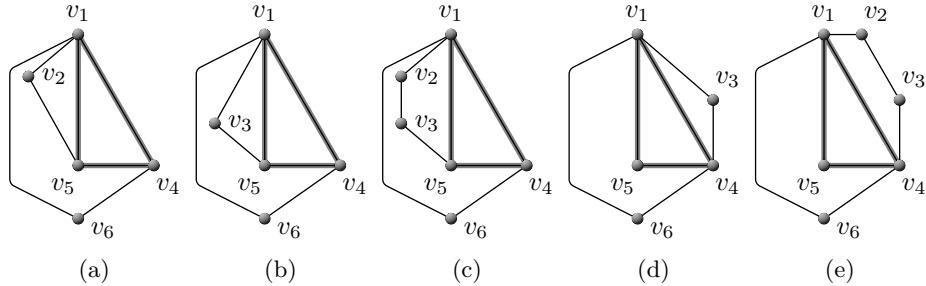


Figure 13.10: Simultaneous embeddings into the outer face F_2 of the subgraph $H_0 \subset G_{13.13}$ in Figure 13.9(b) of the only extremal path of the H_0 -fragment S_1 together with each of the five distinct extremal paths of the H_0 -fragment S_2 .

Suppose a block G has to be tested for planarity. The basic idea behind the planarity testing algorithm of Demoucron *et al.* [13] is that if a plane embedding of a subgraph H of G may be extended to a plane embedding of G , then every path connecting any two vertices of attachment of an H -fragment of G should appear wholly within a single face \mathcal{F} of G in that extension. The algorithm iteratively constructs plane embeddings of increasingly larger subgraphs of G that are eventually extended to a plane embedding of G itself if G is planar. The extension step of the algorithm works as follows. Upon (carefully) choosing a face \mathcal{F} whose boundary contains all vertices of attachment of an H -fragment B of G (so as to ensure that \mathcal{F} can accommodate B wholly within itself), an extremal path of B is embedded within \mathcal{F} . A pseudocode description of the this iterative procedure is presented as Algorithm 30.

Consider again, as an example, the block $G_{13.13}$ of order 6 in Figure 13.9(a), and take the cycle $v_1 v_4 v_5 v_1$ as the initial subgraph H_0 in Step 2 of Algorithm 30. A plane embedding of this cycle has two faces, namely the outer face F_1 and the inner face F_2 shown in row $i = 0$ of Table 13.1. As illustrated in Figure 13.9(b)–(c), the set of H_0 -fragments of $G_{13.13}$ is $f(G_{13.13} \setminus H_0) = \{S_1^{(0)}, S_2^{(0)}\}$; this fragment set is also depicted in row $i = 0$ of Table 13.1. The set of admissible faces for the fragment $S_1^{(0)}$ is $\mathcal{F}_{H_0}(S_1^{(0)}) = \{F_1, F_2\}$, since both the faces F_1 and F_2 contain all the vertices of attachment of $S_1^{(0)}$ (namely v_1 and v_4 , with these vertices clearly marked in row $i = 0$ of Table 13.1). Similarly, $\mathcal{F}_{H_0}(S_2^{(0)}) = \{F_1, F_2\}$. Since $|\mathcal{F}_{H_0}(S_1^{(0)})| = |\mathcal{F}_{H_0}(S_2^{(0)})| = 2$, we arbitrarily select the H_0 -fragment $S_1^{(0)}$ and embed the arbitrary extremal path $v_1 v_2 v_3 v_4$ of $S_1^{(0)}$ within the face F_1 in order to obtain the plane embedding H_1 depicted in row $i = 1$ of Table 13.1.

Algorithm 30: Planarity testing

Input : A block G of order n to be tested for planarity.

Output : A plane embedding of G if G is planar, or a flag **nonplanar** if G is nonplanar.

```

1  $i \leftarrow 0$ 
2  $H_i \leftarrow$  any cycle of  $G$  embedded in the plane
3 loop
4    $L_i \leftarrow$  list of admissible faces of  $H_i$ 
5    $f(G \setminus H_i) \leftarrow$  set of all  $H_i$ -fragments of  $G$ 
6   if  $f(G \setminus H_i) = \emptyset$  then return  $H_i$ , stop
7   for each  $H_i$ -fragment  $B \in f(G \setminus H_i)$  do
8      $\mathcal{F}(B) \leftarrow$  set of faces of  $H_i$  containing all vertices of attachment
      of  $B$ 
9     if  $\mathcal{F}(B) = \emptyset$  for some  $H_i$ -fragment  $B$  then return nonplanar, stop
10    else if  $|\mathcal{F}(B)| = 1$  for some  $H_i$ -fragment  $B$  then select some such  $B$ 
11    else select an arbitrary  $H_i$ -fragment  $B$ 
12    Choose any extremal path  $P$  of the selected  $H_i$ -fragment  $B$ 
13    Embed  $P$  across a single face in  $\mathcal{F}(B)$  to form a new graph  $H_{i+1}$ 
14     $i \leftarrow i + 1$ 
```

The set of H_1 -fragments of $G_{13.13}$ is $f(G_{13.13} \setminus H_1) = \{S_1^{(1)}, S_2^{(1)}, S_3^{(1)}, S_4^{(1)}\}$, as depicted in row $i = 1$ of [Table 13.1](#). The sets of admissible faces for these fragments are $\mathcal{F}_{H_1}(S_1^{(1)}) = \{F_1, F_2, F_3\}$, $\mathcal{F}_{H_1}(S_2^{(1)}) = \{F_1, F_3\}$, $\mathcal{F}_{H_1}(S_3^{(1)}) = \{F_1\}$ and $\mathcal{F}_{H_1}(S_4^{(1)}) = \{F_1\}$. Since there are fragments with only one admissible face, the fragment $S_3^{(1)}$ is arbitrarily selected in [Step 10](#) of the algorithm and its only extremal path $v_2 v_5$ is embedded within its only admissible face F_1 in order to obtain H_2 , as shown in row $i = 2$ of [Table 13.1](#).

The set of H_2 -fragments of $G_{13.13}$ is $f(G_{13.13} \setminus H_2) = \{S_1^{(2)}, S_2^{(2)}, S_3^{(2)}\}$, as depicted in row $i = 2$ of [Table 13.1](#). The sets of admissible faces for these fragments are $\mathcal{F}_{H_2}(S_1^{(2)}) = \{F_1, F_3\}$, $\mathcal{F}_{H_2}(S_2^{(2)}) = \{F_3\}$ and $\mathcal{F}_{H_2}(S_3^{(2)}) = \{F_1\}$. Since there are again fragments with only one admissible face, the fragment $S_2^{(2)}$ is arbitrarily selected in [Step 10](#) of the algorithm and its only extremal path $v_1 v_3$ is embedded within its only admissible face F_3 in order to obtain H_3 , as shown in row $i = 3$ of [Table 13.1](#).

The set of H_3 -fragments of $G_{13.13}$ is $f(G_{13.13} \setminus H_3) = \{S_1^{(3)}, S_2^{(3)}\}$, as depicted in row $i = 3$ of [Table 13.1](#). The sets of admissible faces for these fragments are $\mathcal{F}_{H_3}(S_1^{(3)}) = \{F_2, F_3\}$ and $\mathcal{F}_{H_3}(S_2^{(3)}) = \{F_1\}$. Now there is one fragment with only one admissible face, namely the fragment $S_2^{(3)}$. This fragment is therefore selected in [Step 10](#) of the algorithm and its only extremal path $v_3 v_5$ is embedded within its only admissible face F_1 in order to obtain H_4 , as shown in row $i = 4$ of [Table 13.1](#).

There is only one H_4 -fragment of $G_{13.13}$, namely $f(G_{13.13} \setminus H_4) = \{S_1^{(4)}\}$, as depicted in row $i = 4$ of [Table 13.1](#). The set of admissible faces for this fragment is $\mathcal{F}_{H_4}(S_1^{(4)}) = \{F_2, F_3\}$. This final fragment is selected in [Step 11](#) of the algorithm and its only extremal path $v_1 v_6 v_4$ is embedded within the admissible face F_2 in order to obtain H_5 , as shown in row $i = 5$ of [Table 13.1](#). At this stage no

i	H_i	$f(G_{13.13} \setminus H_i)$
0	 Admissible faces: $\mathcal{F}(S_1^{(0)}) = \{F_1, F_2\}$ $\mathcal{F}(S_2^{(0)}) = \{F_1, F_2\}$	
1	 Admissible faces: $\mathcal{F}(S_1^{(1)}) = \{F_1, F_2, F_3\}$ $\mathcal{F}(S_2^{(1)}) = \{F_1, F_3\}$ $\mathcal{F}(S_3^{(1)}) = \{F_1\}$ $\mathcal{F}(S_4^{(1)}) = \{F_1\}$	
2	 Admissible faces: $\mathcal{F}(S_1^{(2)}) = \{F_2, F_3\}$ $\mathcal{F}(S_2^{(2)}) = \{F_3\}$ $\mathcal{F}(S_3^{(2)}) = \{F_1\}$	
3	 Admissible faces: $\mathcal{F}(S_1^{(3)}) = \{F_2, F_3\}$ $\mathcal{F}(S_2^{(3)}) = \{F_1\}$	
4	 Admissible faces:	
5	 \emptyset	

Table 13.1: Results obtained when applying Algorithm 30 to the block $G_{13.13}$ in Figure 13.9(a).

fragments remain, and the algorithm terminates in [Step 6](#), returning the result that the block $G_{13.13}$ is indeed planar, H_5 being a plane embedding of $G_{13.13}$.

We prove the correctness of [Algorithm 30](#) in general by means of a series of three intermediate lemmas culminating in a theorem stating that the algorithm produces a plane embedding of a block G if and only if G is planar.

Lemma 13.22 *Suppose G is a block. For any two competing H_i -fragments S and T of G in [Step 11](#) of [Algorithm 30](#) it holds that $\mathcal{F}_{H_i}(S) = \mathcal{F}_{H_i}(T)$ and that $|\mathcal{F}_{H_i}(S)| = 2$.*

Proof By contradiction. Suppose S and T are competing H_i -fragments of G in [Step 11](#) of [Algorithm 30](#) for which $\mathcal{F}_{H_i}(S) \neq \mathcal{F}_{H_i}(T)$. Then there are at least three distinct faces F_S , F_T and F_{ST} with the properties that

$$F_S \in \mathcal{F}_{H_i}(S) \setminus \mathcal{F}_{H_i}(T), \quad F_T \in \mathcal{F}_{H_i}(T) \setminus \mathcal{F}_{H_i}(S) \quad \text{and} \quad F_{ST} \in \mathcal{F}_{H_i}(S) \cap \mathcal{F}_{H_i}(T).$$

Let s and t be extremal paths of S and T , respectively. Then s is embeddable into the face F_S and t is embeddable into the face F_T so that the pair (s, t) is embeddable outside of the face F_{ST} , as shown in [Figure 13.11\(a\)](#). But then the pair (s, t) is also embeddable within the face F_{ST} , as shown in [Figure 13.11\(b\)](#). This is, however, a violation of the defining property of competing fragments, showing that $\mathcal{F}_{H_i}(S) = \mathcal{F}_{H_i}(T)$.

To show that $\mathcal{F}_{H_i}(S)$ and $\mathcal{F}_{H_i}(T)$ contain exactly two fragments, an argument similar to the one above may be used, but with three faces $F_1, F_2, F_3 \in \mathcal{F}_{H_i}(S)$ — the details of this argument are left as an exercise. ■

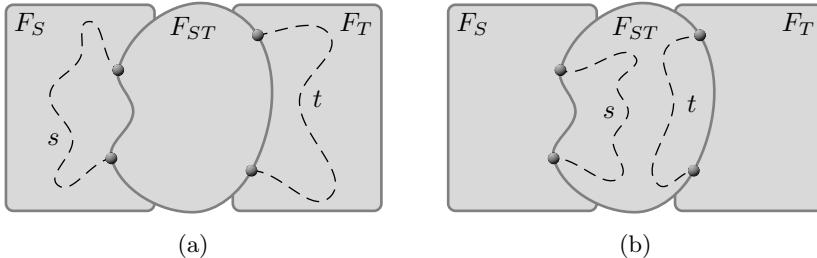


Figure 13.11: Two different ways of embedding the extremal paths s and t in the proof of [Lemma 13.22](#).

For our next lemma we need the notion of a fragment graph. Let H be a plane subgraph of a block G . Then the vertices of the so-called **fragment graph** of G with respect to H , denoted by $\Phi(G \setminus H)$, represent the H -fragments of G , and two vertices in $\Phi(G \setminus H)$ are adjacent if the corresponding two fragments are competing.

Lemma 13.23 *Suppose G is a block and that H is a plane subgraph of G . If G is planar and if each H -fragment of G has at least two admissible faces, then the fragment graph $\Phi(G \setminus H)$ is bipartite.*

Proof By contradiction. Suppose $\Phi(G \setminus H)$ is not bipartite. Then it follows from [Theorem 2.3](#) that there is a cycle $S_1 S_2 S_3 \cdots S_r S_1$ of odd length r in $\Phi(G \setminus H)$. But then it follows from [Lemma 13.22](#) that

$$\mathcal{F}_H(S_1) = \mathcal{F}_H(S_2) = \mathcal{F}_H(S_3) = \cdots = \mathcal{F}_H(S_r)$$

and that the fragments S_1, \dots, S_r each has the same two admissible faces, F_1 and F_2 (say). Let P_i be an extremal path of the fragment S_i , for all $i \in [r]$. The only way to extend the partial plane embedding H of G to a larger partial plane embedding of G by successive embeddings of the extremal paths is to embed P_i within the face F_1 if i is odd or within the face F_2 if i is even (or the other way around). But this successive embedding process results in a simultaneous embedding of the paths P_r and P_1 of the competing fragments S_r and S_1 into the face F_1 (since r is odd) — a contradiction. ■

We next show that each member of the sequence of subgraphs H_0, H_1, H_2, \dots produced by [Algorithm 30](#) is a plane graph.

Lemma 13.24 *If G is a planar block, then iteration i of the loop spanning Steps 3–14 in [Algorithm 30](#) produces a partial plane embedding H_i of G .*

Proof By induction over the iteration index i of the algorithm. Observe, as base case for the induction process, that if $i = 0$, then the subgraph H_i of the input graph G is a cycle embedded in the plane in [Step 2](#) of the algorithm. Assume, as induction hypothesis, that the algorithm yields a partial plane embedding H_k of G at iteration $i = k$ and that the set of fragments $f(G \setminus H_k)$ is nonempty. Then there are admissible faces for all fragments in $f(G \setminus H_k)$. We consider two separate cases for the embedding extension in [Steps 12](#) and [13](#) of the algorithm.

Case 1: If there are fragments in $f(G \setminus H_k)$ with only one admissible face each, then any such a fragment S is selected in [Step 10](#) of the algorithm and an extremal path P of S is embedded within the relevant admissible face. Since this extremal path P is embedded into a single face of H_k in [Steps 12](#) and [13](#) of the algorithm, no edge crossings are produced and the new subgraph H_{k+1} is still a plane graph.

Case 2: If all fragments in $f(G \setminus H_k)$ have at least two admissible faces each and there is no fragment competing with the fragment S selected in [Step 11](#) of the algorithm, then any extremal path P of S embedded within any one of the two admissible faces of S again produces no edge crossings, and hence the new partial embedding H_{k+1} produced in [Steps 12](#) and [13](#) of the algorithm is still a plane embedding. Therefore, suppose all fragments in $f(G \setminus H_k)$ have at least two admissible faces each and that the fragment S selected in [Step 11](#) of the algorithm competes with some other fragment T . We complete the proof by showing that there is no “wrong” choice of an admissible face F of H_k in [Step 13](#) of the algorithm within which to embed an extremal path P of S in the sense that some extremal path Q of T then cannot be embedded into any face of the partial embedding of G at a later iteration of the algorithm, because of the “necessity” of F for the embedding of Q . Suppose the above problem of choosing a “wrong” face for embedding occurs during the extension step of constructing H_{k+1} . We show that this situation can be rectified. Let ϕ be the component of the fragment graph $\Phi(G \setminus H_k)$ containing S . Then ϕ is bipartite by [Lemma 13.23](#) and hence every vertex in ϕ has the same two admissible faces, F_1 and F_2 (say), and no two of these vertices are competing with respect to the faces F_1 and F_2 . If we swap F_1 and F_2 in the partial plane embedding H_k of G , then another plane embedding of G is obtained which can be reduced to H_k and in this new partial embedding we no longer have the problem of having chosen the “wrong” embedding face during the extension process in [Steps 12](#) and [13](#) of the algorithm. ■

We are now in a position to state and prove our final result on the correctness of [Algorithm 30](#).

Theorem 13.25 *Algorithm 30 yields a plane embedding of a block G if and only if G is planar.*

Proof In the sequence of subgraphs H_0, H_1, H_2, \dots produced by [Algorithm 30](#) it holds that $m(H_i) > m(H_{i-1})$ for $i = 1, 2, 3, \dots$. Since $m(G)$ is finite, [Algorithm 30](#) therefore necessarily terminates at some index value $i = \Omega$ (say). If G is planar, then it follows from [Lemma 13.24](#) that each subgraph in the sequence $H_0, H_1, H_2, \dots, H_\Omega$ is a plane graph and hence that H_Ω is a plane embedding of G . If, however, the algorithm terminates in [Step 9](#) because there is an H_Ω -fragment without an admissible face, then G is nonplanar, since we have already shown (in the proof of [Lemma 13.24](#)) that there is no “wrong” choice of an admissible face of H_i in [Step 13](#) of the algorithm within which to embed an extremal path of an H_i -fragment S competing with another H_i -fragment T during any iteration $i < \Omega$ of the algorithm if G is planar. ■

We conclude this section with a result on the worst-case time complexity of [Algorithm 30](#). Proof of this result is left as an exercise.

Theorem 13.26 *Algorithm 30 has a worst-case time complexity of $\mathcal{O}(n^2)$, where n is the order of the input block G .*

❖ The reader should now be able to attempt Exercises [13.32–13.37](#).

13.4 The crossing number of a graph

The **(plane) crossing number** of a graph G , denoted by $\nu(G)$, is the smallest number of edge crossings (edge intersections at points other than vertices) with which G may be drawn in the plane. The crossing number of a graph G can always be realised by means of a so-called **simple drawing** of G , which has the following properties:

- (i) No edge crosses itself,
- (ii) two edges incident with some common vertex do not cross each other,
- (iii) intersection points of curves representing edges are crossing points, i.e. no two curves touch each other, and
- (iv) any two edges cross each other at most once.

We remark that in a drawing of a graph with the minimum number of crossings, the four properties mentioned above are required, for otherwise we can construct a different drawing of the same graph with fewer edge crossings, as illustrated in [Figure 13.12](#).

In addition to the four requirements mentioned above, we also require that in a simple drawing of a graph,

- (v) no more than two edges intersect at any point in the plane,
- so as to be sure that all crossings between pairs of edges are visible and can be counted.

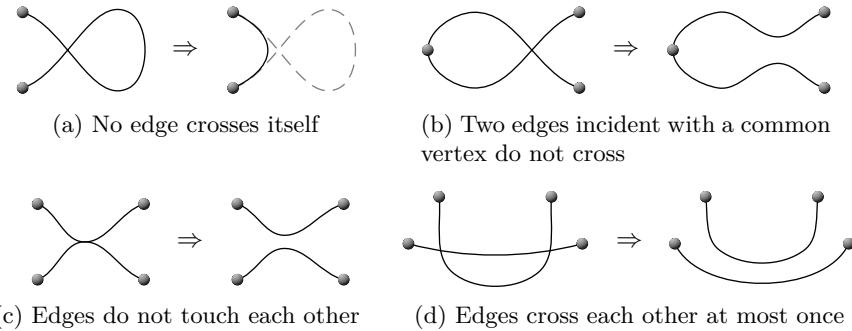


Figure 13.12: Four situations that are ruled out in simple drawings of graphs.

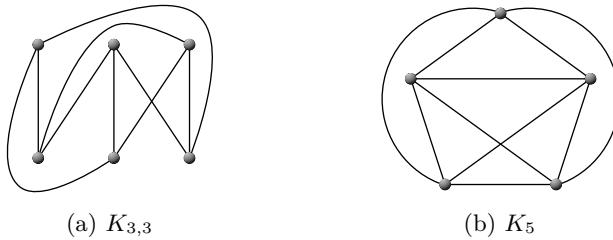


Figure 13.13: Simple plane drawings of $K_{3,3}$ and K_5 with one edge crossing.

The crossing number of any planar graph is therefore zero, while $\nu(K_{3,3}) = 1$ and $\nu(K_5) = 1$ in view of Theorems 13.6 and 13.7 as well as the simple drawings of $K_{3,3}$ and K_5 in Figure 13.13.

We next establish lower and upper bounds on the crossing number of a general graph in terms of its order and size.

Theorem 13.27 *If G is a graph of order n and size m , then*

$$m - 3n + 6 \leq \nu(G) \leq \binom{n}{4}.$$

Proof We show first that the crossing number is at most $\binom{n}{4}$. Arrange the vertices of G in a regular n -gon and join every two adjacent vertices of G by means of a straight line segment. In the resulting straight-line drawing of G , each set of four vertices contributes at most one edge crossing. Thus the number of edge crossings is at most the number of distinct sets of four vertices, namely $\binom{n}{4}$.

To establish the lower bound, consider a simple drawing of G in the plane with $\nu(G)$ crossings. Let H be the graph obtained from G by introducing a new vertex at each of the $\nu(G)$ crossing points. The new graph H is a plane graph of order $n + \nu(G)$. Since every newly formed vertex has degree 4, while the degree of every original vertex remains unchanged, it follows from the Handshaking Lemma (Theorem 1.1) that the number of edges in H is

$$\begin{aligned} |E(H)| &= \frac{1}{2} \sum_{v \in V(H)} d_H(v) = \frac{1}{2} \left(\sum_{v \in V(G)} d_H(v) + \sum_{v \in V(H) \setminus V(G)} d_H(v) \right) \\ &= \frac{1}{2} \sum_{v \in V(G)} d_G(v) + 2(|V(H)| - |V(G)|) = m + 2\nu(G). \end{aligned}$$

Using the bound in [Theorem 13.4](#) on the number of edges in a planar graph, we therefore have that $m + 2\nu(G) \leq 3(n + \nu(G)) - 6$, or equivalently that $\nu(G) \geq m - 3n + 6$. \blacksquare

To illustrate the lower bound in [Theorem 13.27](#), consider, as an example, the complete graph K_6 . By [Theorem 13.27](#), $\nu(K_6) \geq 15 - 18 + 6 = 3$. A drawing of K_6 with three edge crossings is shown in [Figure 13.14](#). Therefore we also have that $\nu(K_6) \leq 3$, and consequently, $\nu(K_6) = 3$.

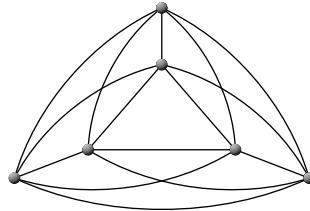


Figure 13.14: A drawing of K_6 realising its crossing number, $\nu(K_6) = 3$.

The first documented interest in the crossing number of a graph came from **Paul Turán** who wrote as follows about a problem he encountered while interred in a Nazi labour camp during World War II where he was forced to work in a brick factory [46]: “There were some kilns where the bricks were made and some open storage yards where the bricks were stored. All the kilns were connected by rail with all storage yards … the trouble was only at crossings. The trucks generally jumped the rails there, and the bricks fell out of them; in short this caused a lot of trouble and loss of time … the idea occurred to me that this loss of time could have been minimised if the number of crossings of the rails had been minimised. But what is the minimum number of crossings?”

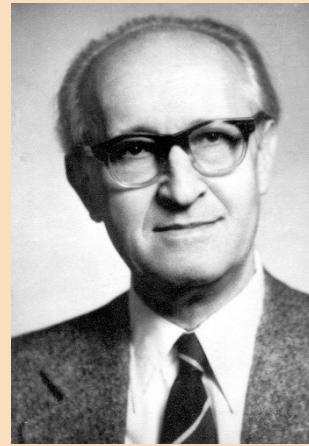
If there are p kilns and q storage yards, then the solution to Turán’s so-called **brickfactory minimisation problem** is the crossing number $\nu(K_{p,q})$. The answer to this problem is not known to this day. **Zarankiewicz** [55], however, showed that

$$\nu(K_{p,q}) \leq \left\lfloor \frac{p}{2} \right\rfloor \left\lfloor \frac{p-1}{2} \right\rfloor \left\lfloor \frac{q}{2} \right\rfloor \left\lfloor \frac{q-1}{2} \right\rfloor \quad (13.2)$$

for all positive integers p and q . In addition, **Zarankiewicz** [55] and **Urbanik** [49] independently claimed that the crossing number $\nu(K_{p,q})$ is actually equal to the upper bound in (13.2) — a claim now widely known as **Zarankiewicz’ Conjecture**. In fact, their attempted proofs of this claim were reprinted in a book, cited, and subsequently used by other authors. Paul Kainen and Gerhard Ringel, however, discovered flaws in their arguments, but despite numerous attempts, these flaws could not be corrected. Nevertheless, no counter example has yet been found showing that equality in (13.2) does not always hold. The interested reader is referred to **Guy** [23] for an account of the history of Zarankiewicz’ Conjecture.

A drawing of $K_{p,q}$ in which the number of edge crossings equals the upper bound in (13.2) may be realised by equipping the plane with a Cartesian system of axes, by placing $\lfloor \frac{q}{2} \rfloor$ vertices on the negative x -axis, $\lceil \frac{q}{2} \rceil$ vertices on the positive x -axis, $\lfloor \frac{p}{2} \rfloor$ vertices on the negative y -axis and $\lceil \frac{p}{2} \rceil$ vertices on the positive y -axis, and by drawing in the appropriate pq edges as straight lines. This construction is illustrated for $p = q = 5$ in [Figure 13.15](#).

Paul Turán was born in Budapest, Hungary on 18 August 1910. He obtained a doctorate in mathematics at the Eötvös Loránd University in 1935 under the supervision of Lipót Fejér. Turán found it difficult to secure an academic position because of discrimination against him as a result of his Jewish origins. In 1940, he was sent to a Nazi labour camp. **Paul Erdős**, who began corresponding with him regularly in 1934, managed to keep some contact with him while he was in the labour camp. He “founded” the mathematical branch of extremal graph theory while interred in the labour camp. He is also well known for his *brick factory optimisation problem*, dating back to World War II. After the war he held various positions (some of them visiting appointments) at many institutions, including the Eötvös Loránd University, Princeton University, and the Hungarian Academy of Sciences. He died of leukaemia in Budapest on 26 September 1976.



Biographic note 44: Paul Turán (1910–1976)

Kleitman [28] showed that equality holds in (13.2) for $p \leq 6$ and he also showed that the smallest counter example to Zarankiewicz’ Conjecture must occur for odd p and odd q (if such a counter example exists). Woodall [53] showed by means of a computer search that equality holds in (13.2) for $K_{7,7}$ and $K_{7,9}$. Hence the smallest unsettled instances of Zarankiewicz’ Conjecture are $K_{7,11}$ and $K_{9,9}$. The limit

$$\mathcal{C} = \lim_{p \rightarrow \infty} \frac{\nu(K_{p,p})}{\binom{p}{2}^2} \quad (13.3)$$

is, however, known to exist, although the value of this limit is not known. The drawing of $K_{p,q}$ described above (with $p = q$) and illustrated for $K_{5,5}$ in Figure 13.15 yields the bound $\mathcal{C} \leq \frac{1}{4}$. If equality in (13.2) always holds, then $\mathcal{C} = \frac{1}{4}$.

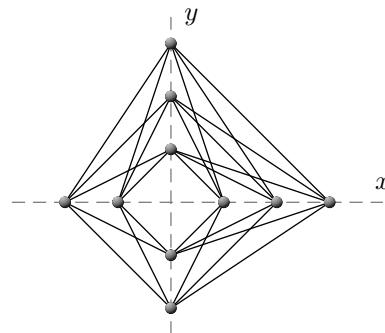


Figure 13.15: A simple plane drawing of $K_{5,5}$ realising the upper bound $\nu(K_{5,5}) \leq 16$ in (13.2). It is known that, in fact, $\nu(K_{5,5}) = 16$ (a result due to Kleitman [28]) — hence this drawing of $K_{5,5}$ is best possible.

Kazimierz Zarankiewicz was born in Częstochowa, Poland on 2 May 1902. He obtained a doctorate in mathematics at the University of Warsaw in 1923 and held temporary teaching positions at various institutions (including teaching illegally in an underground university during World War II) until finally being appointed professor of mathematics at the University of Warsaw in 1948. Zarankiewicz is credited for important work in topology, graph theory, complex functions and number theory. His work on triangular numbers $1, 3, 6, 10, 15, 21, \dots$ (a triangular number is a figurate number which may be represented in the form of a triangular grid of points where the first row contains a single element and each subsequent row contains one more element than the previous) inspired Sierpinski to work further on the topic, while Zarankiewicz collaborated with [Kuratowski](#) in the field of topology. He died in London on 5 September 1959.



Biographic note 45: Kazimierz Zarankiewicz (1902–1959)

For the complete graph of order n it is known that

$$\begin{aligned} \nu(K_n) &\leq \frac{1}{4} \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor =: U(n) \\ &= \begin{cases} \frac{1}{64}n(n-2)^2(n-4) & \text{if } n \text{ is even} \\ \frac{1}{64}(n-1)^2(n-3)^2 & \text{if } n \text{ is odd.} \end{cases} \end{aligned} \quad (13.4)$$

This 1972 result is due to [Guy](#) [24] who also showed that equality holds in (13.4) for all $n \leq 10$; a result extended to $n \leq 12$ by [Pan and Richter](#) [33] in 2007.

[Saaty](#) [40] presented a drawing construction which realises the upper bound in (13.4) where the vertices of K_n are placed on two concentric circles (see [Project 13.1](#)). The upper bound $U(n)$ in (13.4) may also be scaled to produce a lower bound on $\nu(K_n)$. It is, in fact, known that

$$0.8594 U(n) \leq \nu(K_n) \leq U(n)$$

for all positive integers n [12, 33, 35].

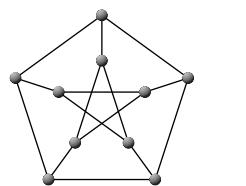
It is interesting to note that the smallest cubic graphs with crossing numbers 1, 2 and 3 are the utility graph $K_{3,3}$ (recall that a drawing of $K_{3,3}$ realising its crossing number $\nu(K_{3,3}) = 1$ is shown in [Figure 13.13\(a\)](#)), the Petersen graph and the Heawood graph, respectively. Conventional simple plane drawings of the Petersen graph and the Heawood graph, as well as drawings by [Exoo](#) [17] realising their crossing numbers, are shown in [Figure 13.16](#).

We close this section by remarking that [Garey and Johnson](#) [22] showed in 1983 that the decision problem associated with the crossing number of a general graph is NP-complete.

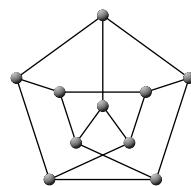
Daniel J Kleitman was born on 4 October 1934 and obtained a doctorate in physics from Harvard University in 1958. After postdoctoral fellowships at the Niels Bohr Institute and at Harvard, he was appointed assistant professor of physics at Brandeis in 1960. He subsequently joined the mathematics department at the Massachusetts Institute of Technology as associate professor of applied mathematics in 1966, and was promoted to full professor in 1969. His research interests include combinatorics, graph theory, genomics and operations research. He was a mathematics advisor (and extra!) in the blockbuster movie *Good Will Hunting*.



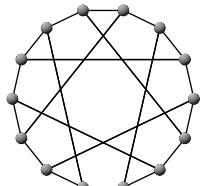
Biographic note 46: Daniel Kleitman (1934–present)



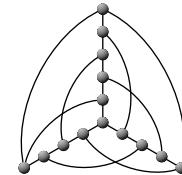
(a) Petersen graph



(b) Petersen graph with 2 edge crossings



(c) Heawood graph



(d) Heawood graph with 3 edge crossings

Figure 13.16: The Petersen graph and the Heawood graph. The drawings in (b) and (d) are due to Exoo [17].

- ❖ The reader should now be able to attempt Exercises 13.38–13.41 as well as Project 13.1.

13.5 Other parameters related to planarity

In this section, we review three graph parameters that are closely related to planarity and the crossing number of a graph. These parameters are the rectilinear crossing number, the thickness and the coarseness of a graph.

13.5.1 The rectilinear crossing number

The following remarkable theorem implies that there is nothing to be gained by drawing the edges of a graph as curves instead of straight line segments when attempting to establish an embedding of a graph in the plane. Although the the-

orem is now widely known as **Fáry's Theorem**, it was proved independently by Wagner [51] in 1936, by Fáry [18] in 1948 and by Stein [43] in 1951.

Theorem 13.28 (Fáry's Theorem) *Any planar graph can be drawn in the plane without edge crossings in such a manner that its edges are straight line segments.*

Proof of the theorem is left as an exercise (see [Project 13.2](#)). As a result of this theorem, an embedding of a planar graph G in the plane so that its edges are straight line segments is called a **Fáry embedding** of G in the plane.

In view of [Theorem 13.28](#) it is indeed surprising that the restriction of drawing graph edges as straight line segments may affect the minimum number of edge crossings for nonplanar graphs. In fact, there is a generalisation of the crossing number of a graph G , known as the **rectilinear crossing number** and denoted by $\bar{\nu}(G)$, which is the smallest number of edge crossings with which G may be drawn in the plane so that its edges are straight line segments. Clearly, the inequality $\nu(G) \leq \bar{\nu}(G)$ holds for any graph G . The smallest complete graph for which the crossing number and the rectilinear crossing number differ is K_8 . The crossing numbers and rectilinear crossing numbers of small complete graphs are shown in [Table 13.2](#).

n	3	4	5	6	7	8	9	10	11	12
$\nu(K_n)$	0	0	1	3	9	18	36	60	100	150
$\bar{\nu}(K_n)$	0	0	1	3	9	19	36	62	102	153

Table 13.2: The crossing numbers and rectilinear crossing numbers of small complete graphs. (Sequences [A000241](#) and [A014540](#) of Sloane [42], respectively.)

- ❖ The reader should now be able to attempt [Exercise 13.42](#) as well as [Project 13.2](#).

13.5.2 The thickness of a graph

The **thickness** of a graph G , denoted by $\theta(G)$, is the smallest number of pairwise edge-disjoint planar spanning subgraphs $G_1, G_2, \dots, G_\theta$ in a factorisation of G (*i.e.* so that $G_1 \oplus G_2 \oplus \dots \oplus G_\theta = G$).

This parameter has many applications, one of which again arises in the design of electric circuit boards. Recall, from the introduction, that in the design of an electric circuit board, the objective is to locate several nodes on the board (a flat board of insulating material) and to connect certain pairs of these nodes by means of electrical wires (or conducting strips) printed directly onto the circuit board according to the specified topology of the circuit. These electrical wires may not cross, since this would lead to undesirable electrical contact or short-circuiting at crossing points. If, however, large numbers of crossings are unavoidable, then electric circuits may be printed on several boards which are then sandwiched together using suitable, insulating interconnections. Each board in such a sandwich consists of a printed circuit without crossings. The following question then arises in order to save costs: What is the smallest number of such sandwich layers for a given

electric circuit topology? The answer to this question is the thickness $\theta(G)$ of the graph G modelling the circuit topology.

Clearly, $\theta(G) = 1$ if and only if G is planar. Since K_5 is nonplanar by [Theorem 13.7](#), it follows that $\theta(K_5) \geq 2$. The factorisation of K_5 into the two pairwise edge-disjoint planar spanning subgraphs in [Figure 13.17](#) shows that $\theta(K_5) \leq 2$, thereby establishing the thickness of K_5 as $\theta(K_5) = 2$.

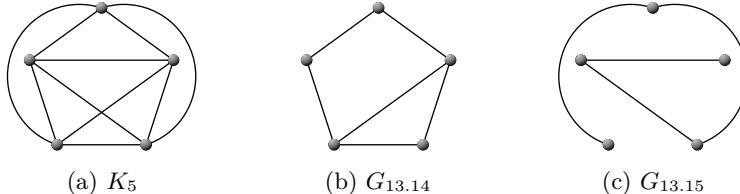


Figure 13.17: A decomposition of K_5 into two pairwise edge-disjoint planar spanning subgraphs $G_{13.14}$ and $G_{13.15}$.

As an immediate consequence of [Theorem 13.4](#), the following lower bound on $\theta(G)$ holds for any graph G of order at least 3.

Corollary 13.29 *If G is a graph of order $n \geq 3$ and size m , then*

$$\theta(G) \geq \left\lceil \frac{m}{3n - 6} \right\rceil.$$

[Corollary 13.29](#) may be used to derive the following good lower bound on the thickness of a complete graph.

Theorem 13.30 $\theta(K_n) \geq \left\lfloor \frac{n+7}{6} \right\rfloor$ for any positive integer n .

Proof The desired result clearly holds for $1 \leq n \leq 4$. We therefore assume that $n \geq 5$. Since the complete graph K_n has size $n(n - 2)/2$, it follows from [Corollary 13.29](#) that

$$\begin{aligned} \theta(K_n) &\geq \left\lceil \frac{n(n-1)}{6(n-2)} \right\rceil = \left\lceil \frac{(n+7)(n-2) - 6(n-2) + 2}{6(n-2)} \right\rceil \\ &= \left\lceil \frac{n+7}{6} - 1 + \frac{2}{6(n-2)} \right\rceil = \left\lceil q - 1 + \frac{r}{6} + \frac{2}{6(n-2)} \right\rceil \end{aligned}$$

by writing $n+7 = 6q+r$ for some integers q and r , where $0 \leq r \leq 5$. Since $n \geq 5$, it follows that $\frac{2}{6(n-2)} < \frac{1}{6}$ and $0 < \frac{r}{6} + \frac{2}{6(n-2)} < 1$, so that

$$\left\lceil q - 1 + \frac{r}{6} + \frac{2}{6(n-2)} \right\rceil = q = \left\lfloor q + \frac{r}{6} \right\rfloor = \left\lfloor \frac{n+7}{6} \right\rfloor. \quad \blacksquare$$

It was, in fact, established by [Beineke \[3\]](#), [Beineke and Harary \[6\]](#), [Vasak \[50\]](#), and [Alexeev and Gončakov \[1\]](#) that

$$\theta(K_n) = \begin{cases} \left\lfloor \frac{n+7}{6} \right\rfloor & \text{if } n \neq 9, 10 \\ 3 & \text{if } n = 9, 10. \end{cases}$$

It is also known for the complete bipartite graph $K_{p,q}$ that

$$\theta(K_{p,q}) = \left\lceil \frac{pq}{2(p+q-2)} \right\rceil,$$

except possibly when p and q are both odd — a result due to [Beineke et al.](#) [7] in 1964.

Lowell Wayne Beineke was born in 1939 and became the Schrey Professor of mathematics at Indiana University-Purdue University, Fort Wayne. He obtained a bachelor's degree in science from Purdue University in 1961 and a doctorate in mathematics from the University of Michigan in 1965 under the supervision of [Frank Harary](#). He is a well-known international scholar in graph theory, and is perhaps best known for his elegant characterisation of line graphs (also sometimes called *derived graphs*). His extensive research spans more than a hundred papers, and he has co-edited more than half a dozen books. He has held numerous visiting professorships at the University of Oxford and, together with [Gary Chartrand](#), was the first to work on the coarseness of graphs.



Biographic note 47: Lowell Beineke (1939–present)

- ❖ The reader should now be able to attempt Exercises [13.43–13.44](#).

13.5.3 The coarseness of a graph

The **coarseness** of a graph G , denoted by $\xi(G)$, is the largest number of pairwise edge-disjoint nonplanar subgraphs contained in G . The notion of coarseness was suggested by [Paul Erdős](#) to [Beineke](#) and [Chartrand](#), who wrote the first paper on the subject in 1968 [4]. Clearly, $\xi(G) = 0$ if and only if G is planar. Since the smallest number of edges that a nonplanar graph can possess is 9 (and then only if the graph is $K_{3,3}$), the following upper bound holds on the coarseness of a graph in terms of its size.

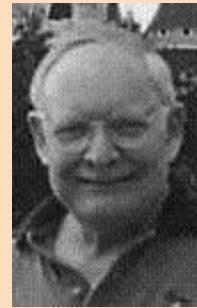
Theorem 13.31 *If G is a graph of size m , then $\xi(G) \leq \lfloor \frac{m}{9} \rfloor$.*

It is an interesting fact that the difference between the coarseness and the thickness of a graph can be made arbitrarily large, as stated in the following result.

Theorem 13.32 *For every positive integer p there exists a graph G such that $\xi(G) - \theta(G) = p$.*

Proof The complete bipartite graph $K_{3,3(p+2)}$ is equivalent to the edge-disjoint union of $K_{1,3(p+2)}$ and $K_{2,3(p+2)}$, each of which is planar. Therefore, $\theta(K_{3,3(p+2)}) = 2$. Furthermore, since $K_{3,3(p+2)}$ contains $p+2$ edge-disjoint copies of $K_{3,3}$, it follows that $\xi(K_{3,3(p+2)}) \geq p+2$. However, $K_{3,3(p+2)}$ has size $9(p+2)$ and so, by [Theorem 13.31](#), $\xi(K_{3,3(p+2)}) \leq p+2$. Hence, $\xi(K_{3,3(p+2)}) = p+2$, so that $\xi(K_{3,3(p+2)}) - \theta(K_{3,3(p+2)}) = p$. ■

Gary Theodore Chartrand was born on 24 August 1936 and is currently professor emeritus of mathematics at Western Michigan University. He obtained a doctorate in mathematics from Western Michigan University in 1964 under the supervision of Edward A Nordhaus. He accepted a teaching position at Western Michigan University in 1964, where he has been an emeritus member of the faculty since his retirement. He is best known for his contributions in graph colourings, distance in graphs, and graph domination. He is a prolific author, having authored or co-authored a large number of text books in graph theory and, together with [Lowell Beineke](#), was the first to work on the coarseness of graphs. He is also responsible for coining the term “outer planar.”



Biographic note 48: Gary Chartrand (1936–present)

The proof of [Theorem 13.32](#) also has the fascinating consequence that, for every integer p , it is possible to combine two *planar* graphs (by edge-disjoint union) into a graph which may, in turn, be decomposed into p edge-disjoint *nonplanar* graphs! The bounds

$$\frac{n(n-1)}{18} \geq \xi(K_n) \geq \begin{cases} \frac{n(n-3)}{18} & \text{if } n \equiv 0 \pmod{3} \\ \frac{(n-1)(n-4)}{18} & \text{if } n \equiv 1 \pmod{3} \\ \frac{(n-2)(n-5)}{18} & \text{if } n \equiv 2 \pmod{3} \end{cases} \quad (13.5)$$

on the coarseness of a complete graph are due to [Beineke and Chartrand](#) [4]. The upper bound in (13.5) follows directly from [Theorem 13.31](#) because the size of K_n is $\binom{n}{2} = n(n-1)/2$. The proofs of the lower bounds are more technical and are hence omitted. [Beineke and Chartrand](#) also established the asymptotic result

$$\lim_{n \rightarrow \infty} \frac{\xi(K_n)}{\frac{n^2}{18}} = 1.$$

The values of the thickness $\theta(K_n)$ and the coarseness $\xi(K_n)$ of the complete graph K_n are shown in [Table 13.3](#) for small values of n .

n	3	4	5	6	7	8	9	10	11	12
$\theta(K_n)$	1	1	2	2	2	2	3	3	3	3
$\xi(K_n)$	0	0	1	1	1	2	3	4	5	6

Table 13.3: The thickness (sequence [A124156](#) of Sloane [42]) and the coarseness of the complete graph K_n for small values of n .

Finally, the following bounds are known for the coarseness of the complete bipartite graph $K_{p,q}$, and are also due to [Beineke and Chartrand](#) [4].

Theorem 13.33 For the complete bipartite graph $K_{p,q}$,

$$\left\lfloor \frac{p}{3} \right\rfloor \left\lfloor \frac{q}{3} \right\rfloor \leq \xi(K_{p,q}) \leq \left\lfloor \frac{pq}{9} \right\rfloor.$$

Proof of this theorem is left as an exercise.

- ❖ The reader should now be able to attempt [Project 13.3](#).

13.6 Embedding on surfaces other than the plane

The reader might rightly wonder why we have thus far only concerned ourselves with drawings of graphs on specifically the plane and the sphere? The answer is that drawings on these surfaces are the easiest to conceptualise. But what about surfaces other than the plane and the sphere? Embedding a graph in the plane or, equivalently, on the sphere, is certainly not topologically equivalent to embedding it on a general orientable surface. This section is therefore devoted to showing how the theory of the previous sections generalises to other orientable surfaces.

13.6.1 The genus of a surface

Broadly speaking, the notion of orientability of a surface in Euclidean space is a measure of whether or not it is possible to make a consistent choice with respect to a surface normal vector at every point on the surface. Such a choice of a surface normal vector allows one to define a “clockwise” direction of traversal of any simple closed curve drawn on the surface. More specifically, a surface is said to be **orientable** if it is possible to choose a “clockwise” orientation consistently for traversing any simple closed curve drawn on the surface. An orientable surface in Euclidean space has the desirable property of having two distinct sides which we may think of as an *inside* and an *outside*.

Examples of orientable surfaces are the **sphere** and the **torus** shown in [Figure 13.18\(a\)–\(b\)](#). On the other hand, the **Möbius strip** shown in [Figure 13.18\(c\)](#) is not orientable. To see why this is the case, consider the simple closed curve \mathcal{C} drawn on the Möbius strip in [Figure 13.18\(c\)](#).

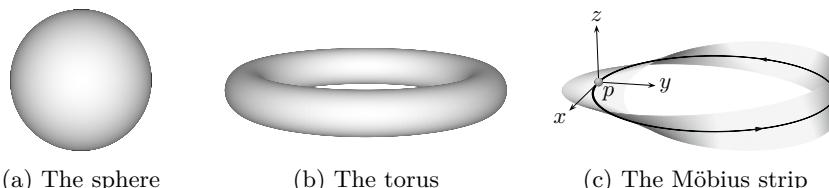


Figure 13.18: (a)–(b) Two orientable surfaces. (c) A surface that is not orientable.

Suppose we choose a “clockwise” traversal direction of \mathcal{C} according to the right-hand set of axes shown, where the x -direction is the direction of traversal, the y -direction points towards the “inside” of the curve and the z -direction defines a surface normal vector, pointing “upwards” in the figure. As we traverse \mathcal{C} from

the point p to where we encounter p again for the second time, the surface normal defined by the z -direction varies continuously until it points in the opposite direction (*i.e.* “downwards” in the figure) at the second encounter of the point p . This traversal shows that it is not possible to make a consistent choice of surface normal vector at every point on the Möbius strip — it is therefore not orientable; it does not have an “inside” and an “outside.”

In this section we consider only orientable surfaces and the possibility of embedding graphs in such surfaces instead of merely embedding them in the plane (or on the sphere), as considered previously in this chapter. For this purpose we require a measure capable of distinguishing between orientable surfaces that are not topologically equivalent. A topologically invariant property of a surface ideally suited to this purpose is the **genus of a surface**, which is the largest number of nonintersecting simple closed curves along which one may cut in the surface without disconnecting the surface.

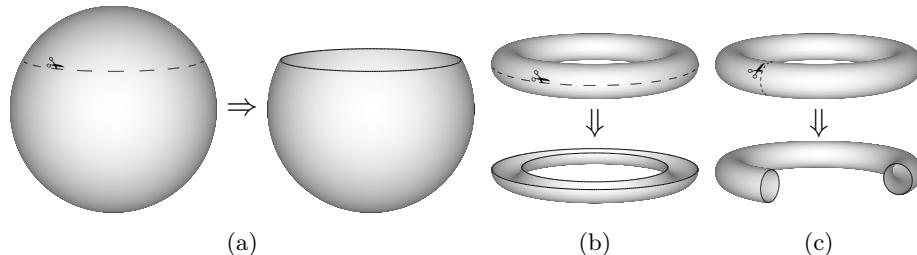


Figure 13.19: (a) The sphere has genus 0. (b)–(c) The torus has genus 1.

As may be seen in Figure 13.19(a), the genus of the sphere is 0, since cutting along *any* simple closed curve on the sphere disconnects it into two parts. For the torus it is possible to cut along a *single* simple closed curve without disconnecting it, as shown in Figure 13.19(b) or (c). One cannot, however, cut along *two* non-intersecting simple closed curves drawn on the torus without disconnecting the surface into two parts. The genus of the torus is therefore 1. Orientable surfaces of genus 0, 1, 2 and 3 are shown in Figure 13.20.

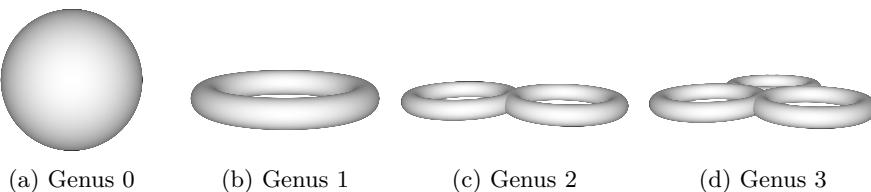


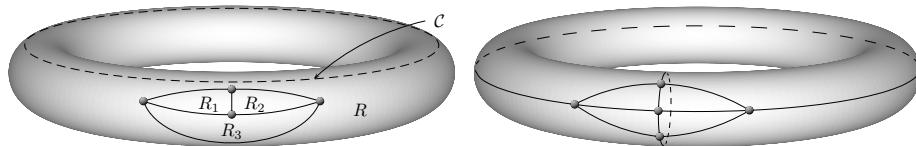
Figure 13.20: Orientable surfaces of genus 0, 1, 2 and 3.

❖ The reader should now be able to attempt Exercises 13.45–13.46.

13.6.2 The generalised crossing number of a graph

The notion of embedding a graph in the plane (or on the sphere, for that matter, by Theorem 13.2) may be extended naturally to orientable surfaces of higher genus.

More specifically, a graph is said to be **embedded** on an orientable surface (of arbitrary genus) if it is drawn on the surface in such a way that none of its edges intersect (except possibly at vertices). The notion of a face in a plane embedding generalises naturally to orientable surfaces of higher genus; the faces are those connected pieces of the surface that remain upon removal of the vertices and edges from the embedding. For orientable surfaces of genus $g \geq 1$, however, two kinds of faces can result — those that are so-called 2-cells and those that are not. A face is a **2-cell** if any simple, closed curve in the face can be deformed continuously (shrunk or contracted) *within that face* to a single point within the face (*i.e.* if the face is homeomorphic to an open disk). An embedding of a graph on an orientable surface is called a **2-cell embedding** if every face of the embedding is a 2-cell. The embedding of K_4 on the torus shown in [Figure 13.21\(a\)](#) is not a 2-cell embedding, since the face denoted by R is not a 2-cell; the simple closed dotted curve \mathcal{C} in R cannot be deformed continuously within R to a point in R . The embedding of K_5 on the torus shown in [Figure 13.21\(b\)](#) is, however, indeed a 2-cell embedding.



(a) An embedding of K_4 on the torus that
is not 2-cell (b) A 2-cell embedding of K_5 on the torus

Figure 13.21: Embeddings of K_4 and K_5 on the torus.

The following interesting generalisation of [Euler's Formula](#) for plane graphs (see [Theorem 13.1](#)) holds for an arbitrary orientable surface.

Theorem 13.34 *If a 2-cell embedding of a connected graph of order n and size m on an orientable surface of genus g results in f faces, then $n - m + f = 2 - 2g$.*

The notion of the crossing number of a graph may also be extended to orientable surfaces of higher genus. The **crossing number of a graph G on an orientable surface of genus g** , denoted by $\nu_g(G)$, is the smallest number of edge crossings (edge intersections at points other than vertices) with which G may be drawn on an orientable surface of genus g . Note that the plane crossing number of a graph is the special case of this new definition where $g = 0$ (*i.e.* $\nu(G) = \nu_0(G)$ for any graph G). The crossing number $\nu_1(G)$ of a graph G drawn on an orientable surface of genus 1 (topographically equivalent to the torus) is often called the **toroidal crossing number** of G . It naturally holds for any graph G that $\nu_{g_2}(G) \leq \nu_{g_1}(G)$ if $g_2 > g_1$.

Recall that K_5 cannot be embedded in the plane (and hence not on the sphere) since it has plane crossing number $\nu_0(K_5) = 1$ by [Theorem 13.7](#). Since K_5 can be embedded on the torus, however, as shown [Figure 13.21\(b\)](#), its toroidal crossing number is $\nu_1(K_5) = 0$.

It is left as an exercise to verify that the complete graphs K_6 and K_7 also have toroidal crossing number zero. The complete graph K_8 cannot, however, be embedded on the torus; its toroidal crossing number is $\nu_1(K_8) = 4$. The toroidal crossing numbers of small complete graphs are shown in [Table 13.4](#). Guy *et al.* [26]

Richard Kenneth Guy was born in Nuneaton (Warwickshire), England on 30 September 1916. He authored many scientific papers and books on topics in combinatorial game theory, graph theory and number theory, including various results on graph crossing numbers such as the upper bound (13.4). During the late 1950s, Guy discovered a unistable polyhedron with only 19 faces — the smallest yet found. He is also the father of the *glider* in Conway's *Game of Life*. Moreover, he was a well-known analyst of chess endgame studies, having composed more than 200 studies and being co-inventor of the Guy-Blandford-Roycroft classification scheme for chess endgame studies. Towards the end of his life, he was professor emeritus of mathematics at the University of Calgary. He died in Calgary on 9 March 2020.



Biographic note 49: Richard Guy (1916–2020)

n	3	4	5	6	7	8	9	10	11	12
$\nu_0(K_n)$	0	0	1	3	9	18	36	60	100	150
$\nu_1(K_n)$	0	0	0	0	0	4	9	23	42	70

Table 13.4: The plane and toroidal crossing numbers of small complete graphs. (Sequences [A000241](#) and [A014543](#) of Sloane [42], respectively.)

proved that

$$\frac{23}{210} \binom{n}{4} \leq \nu_1(K_n) \leq \frac{59}{216} \binom{n-1}{4}$$

for all $n \geq 10$. Guy and Jenkyns [25], furthermore, obtained the bounds

$$\frac{1}{15} \binom{p}{2} \binom{q}{2} \leq \nu_1(K_{p,q}) \leq \frac{1}{6} \binom{p-1}{2} \binom{q-1}{2}$$

on the toroidal crossing number of $K_{p,q}$ in 1969 and showed that the toroidal crossing number of $K_{3,q}$ is

$$\nu_1(K_{3,q}) = \left\lfloor \frac{(q-3)^2}{12} \right\rfloor$$

for all $q \in \mathbb{N}$. In 1996, this result was generalised to an arbitrary surface of genus g by Richter and Šíráň [34], who showed that

$$\nu_g(K_{3,q}) = \left\lfloor \frac{q}{4g+2} \right\rfloor \left[q - (2g+1) \left(1 + \left\lfloor \frac{q}{4g+2} \right\rfloor \right) \right].$$

The values of $\nu_g(K_{3,q})$ are shown in Table 13.5 for all $g \in \{0, 1, 2, 3\}$ and all $q \in \{3, \dots, 18\}$.

❖ The reader should now be able to attempt Exercises 13.47–13.51.

q	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\nu_0(K_{3,q})$	1	2	4	6	9	12	16	20	25	30	36	42	49	56	64	72
$\nu_1(K_{3,q})$	0	0	0	0	1	2	3	4	5	6	8	10	12	14	16	18
$\nu_2(K_{3,q})$	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	
$\nu_3(K_{3,q})$	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	

Table 13.5: The crossing number of $K_{3,q}$ on a surface of genus $g \in \{0, 1, 2, 3\}$.

13.6.3 The genus of a graph

The **genus of a graph** G , denoted by $g(G)$, is the smallest genus of an orientable surface on which G can be embedded. Clearly, $g(G) = 0$ if and only if G is planar. Since K_5 is nonplanar by [Theorem 13.7](#), it follows that $g(K_5) \geq 1$. However, $g(K_5) \leq 1$ by the toroidal embedding of K_5 shown in [Figure 13.21\(b\)](#). The genus of K_5 is thus established as $g(K_5) = 1$.

We state, without proof, an interesting result due to [Youngs](#) [54] which dates back to 1963.

Theorem 13.35 *If G is a connected graph that is embedded on an orientable surface of genus $g(G)$, then it is necessarily a 2-cell embedding.*

Using this result it is possible to establish a lower bound on the genus of a (connected) graph.

Theorem 13.36 *If G is a connected graph of order $n \geq 3$ and size m , then*

$$g(G) \geq \frac{m}{6} - \frac{n}{2} + 1.$$

Proof Suppose a connected graph G of order $n \geq 3$ and size m is embedded on a surface of genus $g(G)$, producing f faces. Then it follows from [Theorem 13.35](#) that every face is a 2-cell and hence from [Theorem 13.34](#) that $n - m + f = 2 - 2g(G)$. An identical argument as in the proof of [Theorem 13.4](#) shows that $3f \leq 2m$ and so

$$2 - 2g(G) = n - m + f \leq n - m + \frac{2m}{3},$$

from which the desired inequality follows. ■

[Theorem 13.36](#) may be used to establish the following lower bound on the genus of a complete graph.

Theorem 13.37 $g(K_n) \geq \lceil \frac{(n-3)(n-4)}{12} \rceil$ for all $n \geq 3$.

Verification of the details of the proof of this theorem is left as an exercise. [Ringel](#) and [Youngs](#) [37] showed in 1968 that equality always holds in the result of [Theorem 13.37](#), in other words that, in fact,

$$g(K_n) = \left\lceil \frac{(n-3)(n-4)}{12} \right\rceil$$

for all $n \geq 3$. The genus of the complete graph K_n is shown in [Table 13.6](#) for small values of n .

For complete bipartite graphs, [Ringel](#) [36] and [Beineke and Harary](#) [5] showed in 1965 that

$$g(K_{p,q}) = \left\lceil \frac{(p-2)(q-2)}{4} \right\rceil.$$

n	3	4	5	6	7	8	9	10	11	12
$\nu(K_n)$	0	0	1	3	9	18	36	60	100	150
$g(K_n)$	0	0	1	1	1	2	3	4	5	6

Table 13.6: The crossing number $\nu(K_n)$ and genus $g(K_n)$ of the complete graph K_n for small values of n . (Sequences [A000241](#) and [A000933](#) of Sloane [42], respectively.)

We conclude this section by remarking that the decision problem of determining, for a general graph G and a positive integer k , whether $g(G) \leq k$ is NP-complete; a 1989 result due to Thomassen [45].

- ❖ The reader should now be able to attempt Exercises [13.52–13.54](#).

13.7 The Robertson-Seymour theorems

There is a remarkable family of theorems, due to [Neil Robertson](#) and [Paul Seymour](#), which generalise the notion of the existence of a finite collection of forbidden minor embeddings in the characterisation of [Wagner](#) for planar graphs to surfaces of higher genus [19]. Robertson and Seymour called the statement of their most famous theorem [Wagner's Conjecture](#) (although Wagner denied ever conjecturing it) until they finally proved it in a series of twenty papers [38]–[39] spanning over 500 pages during the period 1983 to 2004. In order to state this theorem we require the notions of a binary relation and a partially ordered set.

George Neil Robertson was born in Canada in December 1938 and is currently distinguished professor of mathematics (topological graph theory) at Ohio State University. He obtained his doctorate in 1969 at the University of Waterloo under the supervision of [William Tutte](#) and joined the faculty of Ohio State University in the same year. He was promoted to associate professor in 1972 and was appointed as professor in 1984. During the period 1984–1996 he consulted for Bell Communications Research and has held many visiting faculty positions. He is best known for the theory that he developed together with [Paul Seymour](#) on graph minors during the period 1983–2004, culminating in the important results of Theorems [13.38](#) and [13.39](#).



Biographic note 50: Neil Robertson (1938–present)

A **binary relation** \mathcal{R} on a nonempty set S is a subset of the Cartesian product $S \times S$. If $(a, b) \in \mathcal{R}$, then we write $a \mathcal{R} b$. A binary relation \mathcal{R} on S is

- (i) **reflexive** if $a \mathcal{R} a$ for all $a \in S$,
- (ii) **antisymmetric** if, whenever $a \mathcal{R} b$ and $b \mathcal{R} a$, then $a = b$, and
- (iii) **transitive** if, whenever $a \mathcal{R} b$ and $b \mathcal{R} c$, then $a \mathcal{R} c$.

A binary relation \mathcal{R} on a set S which is reflexive, antisymmetric and transitive is called a **partially ordered binary relation** on S , in which case the set S together with \mathcal{R} is called a **partially ordered set**. Recall, from [Section 13.3.1](#), that the notation $H \preccurlyeq G$ is used to denote the fact that a graph H is a minor of a graph G . If \mathcal{G} is the set of all graphs and

$$\mathcal{R} = \{(H, G) \in \mathcal{G} \times \mathcal{G} \mid H \preccurlyeq G\},$$

then \mathcal{R} together with \mathcal{G} is a partially ordered set. Here we refer to the ordering denoted by the symbol “ \preccurlyeq ” as the **minor ordering** on the set of all graphs.

A set \mathcal{F} of graphs is **closed under the minor ordering of graphs** if, whenever $G \in \mathcal{F}$ and $H \preccurlyeq G$, it also holds that $H \in \mathcal{F}$. Since every subgraph of a planar graph is again planar and since contracting an edge of a planar graph again results in a planar graph, it follows that the family \mathcal{P} of all planar graphs is closed under the minor ordering of graphs.

An element H of a set \mathcal{F} of graphs (which is not necessarily closed under the minor ordering of graphs) is called a **minimal element with respect to the minor ordering of graphs** if every minor of H distinct from H does not belong to \mathcal{F} . For example, the graphs K_5 and $K_{3,3}$ are minimal elements of the family $\overline{\mathcal{P}}$ of all nonplanar graphs.

In general, the **complement** $\overline{\mathcal{F}}$ of a set \mathcal{F} of graphs is the set of all graphs that do not belong to \mathcal{F} . The so-called **obstruction** set for \mathcal{F} is the set of all graphs in the complement $\overline{\mathcal{F}}$ of \mathcal{F} that are minimal elements of $\overline{\mathcal{F}}$ with respect to the minor ordering of graphs. For example, the set $\mathcal{L}_0 = \{K_5, K_{3,3}\}$ is the obstruction set for the family \mathcal{P} of all planar graphs according to [Theorem 13.19](#). If \mathcal{F} is any collection of graphs that is closed under the minor ordering of graphs, then any graph G belongs to \mathcal{F} if and only if there is no graph H in the obstruction set for \mathcal{F} such that $H \preccurlyeq G$.

Although the following theorem has been shown to hold by [Robertson](#) and [Seymour](#), it is still widely known as [Wagner's Conjecture](#).

Theorem 13.38 ([Wagner's Conjecture](#)) *Every set of finite, undirected, unlabelled graphs contains only a finite number of minimal elements with respect to the minor ordering of graphs.*

Since the family of graphs that can be embedded on some orientable surface is closed under the minor ordering of graphs, [Wagner's Conjecture](#) has the following consequence in terms of graph embeddability on general orientable surfaces.

Theorem 13.39 ([Robertson-Seymour](#) Theorem) *For any integer $g \geq 0$ there is a finite obstruction set \mathcal{L}_g of graphs with the property that a graph G can be embedded on an orientable surface of genus g if and only if G does not contain, as a minor, any of the graphs in \mathcal{L}_g .*

It is interesting to note that, whereas the obstruction set $\mathcal{L}_0 = \{K_5, K_{3,3}\}$ is specified by [Wagner's Theorem](#) ([Theorem 13.19](#)), none of the other obstruction sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \dots$ in [Theorem 13.39](#) are known explicitly. It is even difficult to establish bounds on the cardinalities $|\mathcal{L}_1|, |\mathcal{L}_2|, |\mathcal{L}_3|, \dots$ of these sets. For example, it is conjectured that every minimal minor in the obstruction set \mathcal{L}_1 for graph embeddability on the torus has order less than 30, but this has yet to be verified [32]. Furthermore, [Gagarina et al.](#) [21] showed that for graphs with no $K_{3,3}$ -subdivisions, the toroidal obstruction set contains four minors.

Paul D Seymour was born in Plymouth (Devon, England) on 26 July 1950 and is currently Albert Baldwin Dod Professor of Mathematics at Princeton University, a position he took up in 1996. He obtained a doctorate in mathematics from the University of Oxford in 1975 and was appointed professor of mathematics at Ohio State University in 1983, where he worked in discrete mathematics, including graph theory, optimisation and combinatorics in general. During the period 1983–1996 he consulted for Bell Communications Research until he moved to Princeton. He won the Ostrowski Prize in 2004, the Fulkerson Prize four times (in 1979, 1994, 2006 and 2009) and the Pólya Prize twice (in 1983 and 2004). He has also received honorary doctorates from the University of Waterloo and from the Technical University of Denmark. He is best known for his pioneering work on regular matroids, unimodular matrices, the run-up to the 1976 proof of the [Four-colour Theorem](#) and his work on graph minors, culminating in the twenty papers with [Neil Robertson](#) which led to the important results of Theorems 13.38 and 13.39.



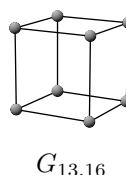
Biographic note 51: Paul Seymour (1950–present)

A significant consequence of [Theorem 13.39](#) is that it proves the existence of a polynomial-time algorithm for testing whether or not a given graph G of order n can be embedded on an orientable surface of specified genus $g \geq 0$. This follows from the finiteness of the relevant obstruction set, and by the following theorem which is also due to [Robertson](#) and [Seymour](#).

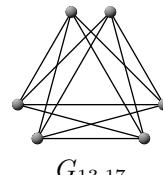
Theorem 13.40 *If G and H are arbitrary graphs, then there exists an efficient algorithm capable of determining whether or not H is a minor of G .*

Exercises

- 13.1 Suppose the three houses and three utilities problem were the four houses and two utilities problem. Would it then be possible to connect each utility to each house without the utility connecting lines crossing?
- 13.2 Show that the graph $K_{2,n}$ is planar for every positive integer n .
- 13.3 Produce a plane embedding of each of the following planar graphs:



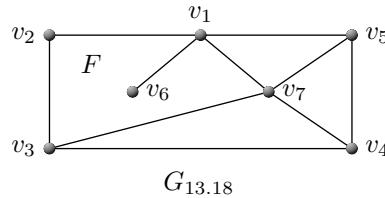
$G_{13.16}$



$G_{13.17}$

- 13.4 There are ten nonisomorphic, connected, planar graphs of order at most 4. Embed each of these graphs in the plane.

- 13.5 Determine, for the plane graph $G_{13.18}$ below, the vertices and the edges on
(a) the boundary of the face F and (b) the boundary of the outer face.



- 13.6 Verify Euler's formula ([Theorem 13.1](#)) for the graph $G_{13.18}$ in [Exercise 13.5](#) above.

- 13.7 A soccer ball consists of 32 faces, each of which is either a regular pentagon or a regular hexagon. Furthermore, exactly three faces meet at each vertex (two hexagonal faces and one pentagonal face), as shown in [Figure 13.22](#). Use [Euler's Formula](#) ([Theorem 13.1](#)) to determine how many pentagons and how many hexagons make up the soccer ball.



Figure 13.22: A soccer ball comprising 32 faces, each of which is either a (black) regular pentagon or a (white) regular hexagon.

- 13.8 Prove that any connected, planar graph has at least four vertices of degree at most 5.
- 13.9 Prove that any connected, nonplanar graph has either at least five vertices of degree at least 4 or at least six vertices of degree at least 3.
- 13.10 Prove, by induction on the size of the graph, that a plane graph is bipartite if and only if the boundary of each of its faces is a closed walk of even length.
- 13.11 Suppose $f \geq 2$ and $\ell \geq 1$ are integers with $f \times \ell$ even. Construct a planar graph with exactly f faces in which the boundary of every face is a closed walk of length ℓ .
- 13.12 Show that every connected planar graph can be factored into an edge-disjoint union of three forests.
- 13.13 Show that there exists no 6-connected planar graph.
- 13.14 Show that every **triangulated** plane graph (a plane graph in which each face is bounded by a 3-cycle) of order at least 4 is 3-connected.
- 13.15 Consider a plane drawing with f faces of a graph G of order n and size m comprising k components. Determine the value of $n - m + f$ in terms of k .
- 13.16 Show that the graph obtained from K_5 by removing an edge is planar.

- 13.17 What is the smallest number of edges that may be removed from K_6 in order to obtain a planar graph?
- 13.18 Show that there exists no r -regular, planar, bipartite graph for $r \geq 4$.
- 13.19 Prove or disprove the following statement: If G is a connected graph of order n and size $m = 3n - 6$, then G is planar.
- 13.20 (a) Show that if a shortest cycle in a planar graph of order n and size m has length k , then $m \leq \frac{k(n-2)}{k-2}$.
 (b) Deduce that a planar bipartite graph of order $n \geq 4$ has at most $2n - 4$ edges.
 (c) Use the result in (b) to show that $K_{3,3}$ is nonplanar.
- 13.21 Prove that if G is a graph of order 11, then G or its complement \bar{G} is nonplanar.
- 13.22 Use the fact that every cycle of the Petersen graph (see Figure 13.16(a)) has length at least 5 to prove that it is nonplanar. (Hint: Consider using Exercise 13.20(a).)
- 13.23 Prove Theorem 13.9, i.e. show that a graph H is a G -minor graph if and only if G can be obtained from H by a series of edge contractions. (Hint: Use induction on $|V(H)| - |V(G)|$.)
- 13.24 Show that $K_{2,2}$ is a subdivision of $K_{3,3}$.
- 13.25 Identify, in each of the nonplanar graphs of Figure 13.23, a subgraph that is either K_5 or $K_{3,3}$, or a subdivision of one of these graphs.

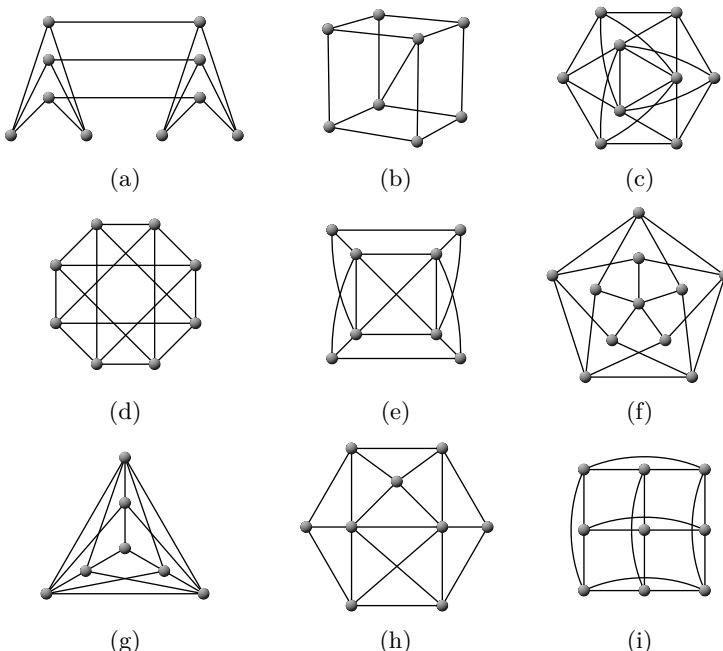


Figure 13.23: Nonplanar graphs associated with Exercise 13.25.

- 13.26 Produce a plane drawing of a connected graph with degree sequence $4, 4, 4, 4, 2, \dots$ which is not a subdivision of K_5 .
- 13.27 Show that every topological minor of a graph is also its (ordinary) minor.
- 13.28 Show that if a graph contains $K_{3,3}$ as a minor, then it contains $K_{3,3}$ as a topological minor.
- 13.29 Prove that the Petersen graph (shown in Figure 13.16(a)) is nonplanar by using Kuratowski's Theorem (Theorem 13.14).
- 13.30 Show that the result of Tutte's Theorem (Theorem 13.15) is not necessarily true if we replace the condition of 3-connectedness with a condition of 2-connectedness. (Hint: Give an example of a 2-connected graph that contains neither K_5 nor $K_{3,3}$ as a topological minor, but which does not have an embedding in the plane with all inner faces convex.)
- 13.31 Prove that the Petersen graph (shown in Figure 13.16(a)) is nonplanar by using Wagner's Theorem (Theorem 13.19).
- 13.32 Prove that the binary relation \sim used to define the notion of an H -fragment in Section 13.3.4 is, in fact, an equivalence relation.
- 13.33 Identify which of the graphs in Figure 13.24 are blocks, and which are not. Also state in each case for which values of $k \in \mathbb{N}$ the graphs are k -connected.

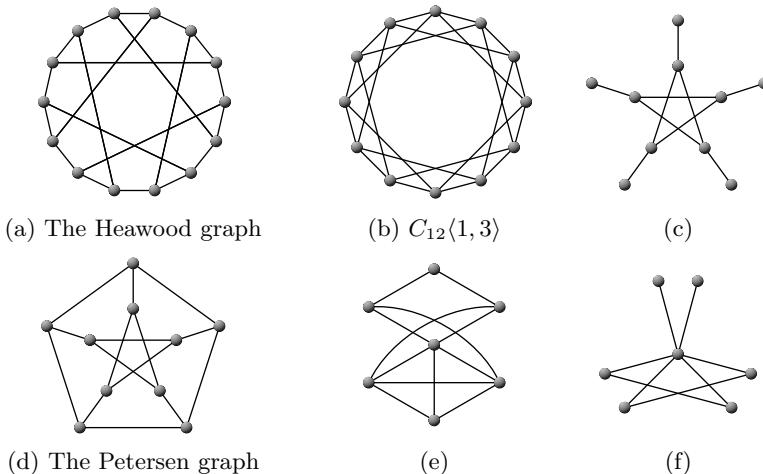


Figure 13.24: A collection of graphs associated with Exercise 13.33.

- 13.34 In each of the cases in Figure 13.25 produce drawings of the H -fragments of the graph, where H is the cycle that has been highlighted. Also state which faces of H are admissible for each of the H -fragments.
- 13.35 Use Algorithm 30 to test whether or not each of the graphs in Figure 13.26 is planar.
- 13.36 Complete the proof of Lemma 13.22.
- 13.37 Prove Theorem 13.26.

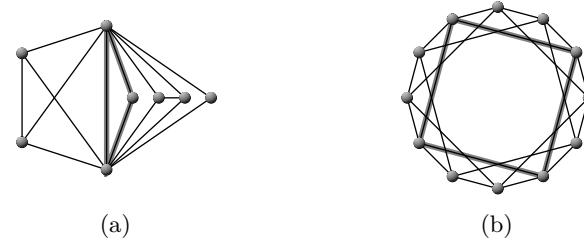


Figure 13.25: A collection of graphs associated with [Exercise 13.34](#).

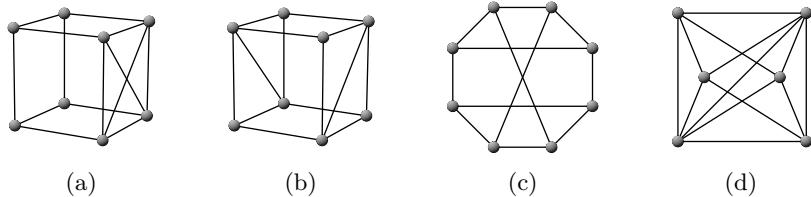


Figure 13.26: A collection of graphs associated with [Exercise 13.35](#).

- 13.38 Determine the crossing number $\nu(K_{2,2,2})$.
- 13.39 Determine the crossing number $\nu(K_{1,2,2,2})$.
- 13.40 Prove that $\nu(K_{2,2,3}) = 2$.
- 13.41 Produce a drawing of $K_{6,7}$ realising its crossing number $\nu(K_{6,7}) = 54$.
- 13.42 Produce a straight-line drawing of K_7 showing that $\nu(K_7) \leq \bar{\nu}(K_7) \leq 9$.
- 13.43 Determine the thickness of the Petersen graph shown in [Figure 13.24\(d\)](#).
- 13.44 Prove that $\theta(K_8) = 2$.
- 13.45 Produce a drawing of an orientable surface of genus 4.
- 13.46 Research, on the Internet, two examples of nonorientable surfaces other than the Möbius strip. Convince yourself in each case that the surface is indeed not orientable by showing that a consistent choice of surface normal cannot always be made (*i.e.* that the surface does not have a natural “inside” and “outside”).
- 13.47 Verify that the utility graph $K_{3,3}$ can be embedded on the torus.
- 13.48 Produce a 2-cell embedding of K_4 on the torus.
- 13.49 Produce an embedding of K_5 on an orientable surface of genus 2 which is not a 2-cell embedding.
- 13.50 Verify that the complete bipartite graphs $K_{3,4}$, $K_{3,5}$ and $K_{3,6}$ can all be embedded on the torus.
- 13.51 Produce a drawing of the complete bipartite graph $K_{3,7}$ with one edge crossing on the torus.
- 13.52 Prove that every embedding of a connected graph G on an orientable surface of genus $g(G)$ has the same number of faces.

13.53 Use [Theorem 13.36](#) to prove [Theorem 13.37](#).

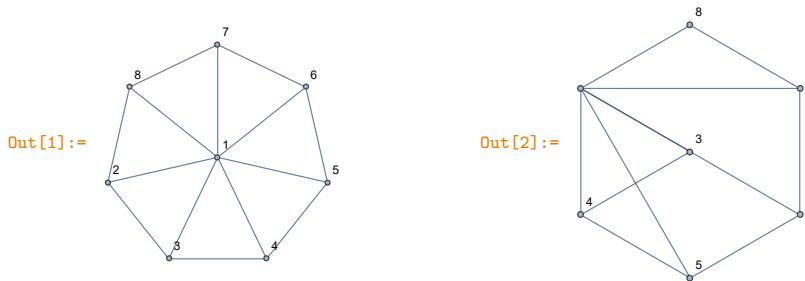
13.54 Let G be a connected graph. Explain why $g(G) \leq \nu(G)$.

Computer exercises

The [MATHEMATICA](#) command `EdgeContract[G, {u <-> v}]` produces the graph obtained by contracting the edge uv in a graph G . For example, the commands

```
In[1]:= WheelGraph[8, VertexLabels -> "Name"]
In[2]:= EdgeContract[WheelGraph[8, VertexLabels -> "Name"], {1 <-> 2}]
```

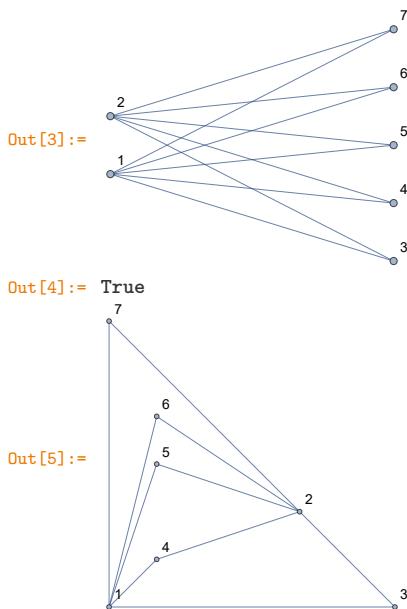
produce the output:



The command `PlanarGraphQ[G]`, furthermore, yields the boolean value **True** if G is a planar graph, or the boolean value **False** otherwise. In the case of a planar graph G , the attribute `GraphLayout -> "PlanarEmbedding"` may be used to produce a plane embedding of G . For example, the commands

```
In[3]:= CompleteGraph[{2, 5}, VertexLabels -> "Name"]
In[4]:= PlanarGraphQ[CompleteGraph[{2, 5}]]
In[5]:= CompleteGraph[{2, 5}, GraphLayout -> "PlanarEmbedding", VertexLabels -> "Name"]
```

produce the output:



- ❖ The reader should now be able to repeat Exercises 13.3 and 13.35 without pen and paper.

Projects

This section contains three projects. In the first, Saaty's simple drawing of K_n realising the bound (13.4) is described and the reader is guided through the process of counting the edge crossings. In the second, the reader is taken through Fisk's proof of [Fáry's Theorem](#). This proof is based on Chvátal's celebrated watchman theorem. The purpose of the final project is to establish the bounds, due to [Beineke](#) and [Chartrand](#), on the coarseness of a complete bipartite graph.

Project 13.1: Realising the plane crossing number of K_n

The purpose of this project is to guide the reader towards showing that the upper bound (13.4) on the crossing number $\nu(K_n)$ of a complete graph of order n is indeed realisable. We follow the approach due to [Saaty](#) [40] in 1967 of producing a simple drawing of K_n with exactly

$$\begin{aligned} U(n) &= \frac{1}{4} \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor \\ &= \begin{cases} \frac{1}{64}n(n-2)^2(n-4) & \text{if } n \text{ is even} \\ \frac{1}{64}(n-1)^2(n-3)^2 & \text{if } n \text{ is odd} \end{cases} \end{aligned}$$

edge crossings. There are actually four cases to consider, namely where $n = 4k$, $n = 4k+1$, $n = 4k+2$ or $n = 4k+3$ for some $k \in \mathbb{N}$. The drawing constructions in these four cases are similar in nature, differing only very slightly in technical detail. We shall start with the case $n = 4k+2$, describing it in detail, and then mentioning afterwards how the method of construction is altered for the other three cases.

Suppose $n = 4k+2$, for some $k \in \mathbb{N}$, and consider a drawing of K_n in which its vertices are arranged on two concentric circles, with $2k+1$ vertices placed in an equidistant manner on the inner circle and the other $2k+1$ vertices placed similarly on the outer circle. The drawing will consist of three types of edges:

Type 1 edges. Each edge of type 1 joins two vertices on the inner circle, and are drawn as straight line segments.

Type 2 edges. Each edge of type 2 joins two vertices on the outer circle, and they are drawn as curved line segments in a clockwise or anti-clockwise direction, choosing the direction that produces the shortest edge. These edges are drawn on or outside the outer circle.

Type 3 edges. Each edge of type 3 joins a vertex on the inner circle with a vertex on the outer circle. These edges are drawn either as straight or as curved line segments, so as to remain wholly between the two concentric circles.

An example of this kind of simple drawing for K_{10} is shown in [Figure 13.27](#). The reader may verify exhaustively that there are five crossings between edges

of type 1 (drawn as dotted lines) in this drawing, five crossings between edges of type 2 (drawn as grey curves), and fifty crossings between edges of type 3 (drawn as black curves), yielding a total of $U(10) = \frac{1}{4} \left\lfloor \frac{10}{2} \right\rfloor \left\lfloor \frac{9}{2} \right\rfloor \left\lfloor \frac{8}{2} \right\rfloor \left\lfloor \frac{7}{2} \right\rfloor = 60$ crossings.

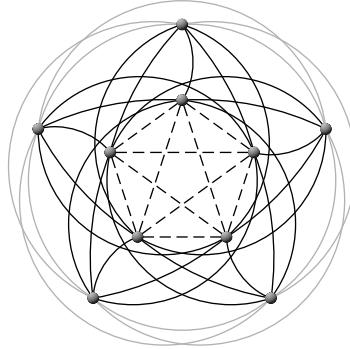


Figure 13.27: A simple drawing of K_{10} realising the upper bound (13.4) in terms of the number of edge crossings.

Tasks

1. The number of crossings between edges of type 1 may be found by considering a fixed vertex x on the inner circle, and counting the crossings produced between edges incident with x and vertices that are immediately adjacent to x on the inner circle, then counting the crossings produced between edges incident with x and vertices that are a distance 2 apart on the inner circle, and so on. Show that this count yields a total of

$$\frac{2k+1}{4} \left(\sum_{i=1}^{2k-2} i + \sum_{i=1}^{2k-3} i + \sum_{i=1}^{2k-4} i + \cdots + 2 + 1 \right) \quad (13.6)$$

edge crossings. (Hint: The factor $2k+1$ results because of the number of distinct ways of fixing x on the inner circle, while the factor of $\frac{1}{4}$ results because of counting every edge crossing four times in the above procedure — once from each of the four vertices that determine a pair of intersecting edges.)

2. Show that the quantity in (13.6) may be simplified to $\binom{2k+1}{4}$.
3. Convince yourself that the number of crossings between edges of type 2 equals the number of crossings between edges of type 1. (Hint: The number of edge crossings outside the outer circle may be obtained from those inside the inner circle by means of a stereographic projection in which the roles of edges of types 1 and 2 are interchanged.)

For the edge crossings that occur in the region between the two circles, consider a fixed vertex x on the inner circle and an axis of symmetry along the line through x and the corresponding vertex on the outer circle. Note that x is joined to k vertices on the outer circle that are to the “left” of this axis of symmetry and to k vertices on the outer circle that are to the “right” of this axis of symmetry. Let e_1 be the

edge incident with x and the first vertex on the outer circle “left” of the axis of symmetry. Edges emanating from the first vertex “left” of x on the inner circle cross the edge e_1 a total of k times. Edges emanating from the second vertex “left” of x on the inner circle cross the edge e_1 a total of $k - 1$ times, and so on. Edges emanating from the first vertex on the inner circle to the “right” of x cross the edge e_1 a total of $k - 2$ times, edges from the second vertex to the right of x cross the edge e_1 a total of $k - 3$ times, and so on. Combining these two series of crossings we find that the total number of edge crossings involving the edge e_1 is

$$k + (k - 1) + \cdots + 1 + 0 + 0 + 1 + \cdots + (k - 3) + (k - 2).$$

For the edge e_2 incident with x and the second vertex to the left of the axis of symmetry, a similar sequence occurs, but shifted over by unity. In other words the total number of edge crossings involving the edge e_2 is

$$(k + 1) + k + \cdots + 1 + 0 + 0 + 1 + \cdots + (k - 4) + (k - 3).$$

Continuing analogously, the corresponding number of edge crossings involving the edge e_k is

$$(2k - 1) + (2k - 2) + \cdots + 1 + 0.$$

The same procedure follows for the edges emanating from x to vertices on the right of the axis of symmetry. For the edge along the axis of symmetry, the corresponding series is

$$(k - 1) + (k - 2) + \cdots + 1 + 0 + 0 + 1 + \cdots + (k - 2) + (k - 1).$$

Tasks (continued)

4. Show that the grand total of edge crossings enumerated by the process described above is

$$2 \sum_{i=1}^{2k-1} \sum_{j=1}^i j. \quad (13.7)$$

5. Show that the enumeration process above yields the total number of crossings between edges of type 3 as

$$\frac{k(2k-1)(2k+1)^2}{3}. \quad (13.8)$$

(Hint: Multiply the quantity in (13.7) by $2k + 1$, because this is the number of ways in which the vertex x may be fixed, and then divide the answer by 2, because each edge crossing will have been counted exactly twice in the enumeration process described above.)

6. Add twice the quantity in Task 2 to the quantity in Task 5 to show that the simple drawing constructed above realises the upper bound $U(n)$ in (13.4). That is, show that

$$2 \binom{2k+1}{4} + \frac{k(2k-1)(2k+1)^2}{3} = \frac{1}{64} n(n-2)^2(n-4),$$

where $n = 4k + 2$.

Let us now consider the case where $n = 4k$, for some $k \in \mathbb{N}$. Exactly the same drawing construction is used with the only exception that the axes of symmetry introduced for Task 4 now pass through diametrically opposite vertices. It is immaterial on which side of a vertex edges are drawn in the region between the two circles as well as outside the outer circle to a diametrically opposite vertex, as long as the same side is used with respect to every symmetry axis.

Tasks (continued)

7. Convince yourself that exactly the same counting procedure as applied in Tasks 1–6 is applicable in this case.

We next consider the case where $n = 4k + 3$, for some $k \in \mathbb{N}$. Construct a drawing of K_n in which its vertices are arranged on two concentric circles (with $2k + 1$ vertices on the inner circle and another $2k + 1$ vertices on the outer circle) and where a single vertex is placed at the centre of both circles. The drawing still consists of three types of edges, as described above. Moreover, the central vertex is to be joined to *all* other vertices by straight lines. To render this additional feature possible, the vertices on the inner circle are rotated slightly so as to ensure that no edge between the central vertex and a vertex on the outer circle coincides with the corresponding vertex on the inner circle. The total number of crossings between edges of each type is the same as before, except for the additional crossings between the straight line edges incident with the central vertex; only crossings involving these edges have to be added to the quantities enumerated before.

Tasks (continued)

8. Show that in this case the number of edge crossings within the inner circle is $k^2(2k + 1)^2$.
9. Show that the number of edge crossings outside the outer circle is $\frac{1}{3}k(k + 1)(2k + 1)(4k - 1)$.
10. Show that the number of edge crossings in the region enclosed between the two circles is $\frac{1}{6}k(2k + 1)^2(k + 1)$.
11. Add the quantities in Tasks 8–10 together to show that the simple drawing constructed above realises the upper bound $U(n)$ in (13.4). That is, show that a total of $\frac{1}{64}(n - 1)^2(n - 3)^2$ edge crossings are achieved, where $n = 4k + 3$.

The drawing for the case $n = 4k + 1$ is identical to that for the case $n = 4k + 3$, except that the central vertex is placed slightly off-centre, so as not to interfere with edge crossings.

Tasks (continued)

12. Convince yourself that the enumeration process for the case $n = 4k + 3$ described above carries through for the case $n = 4k + 1$, and hence that the upper bound $U(n)$ in (13.4) is again realised in this final case.

Project 13.2: Proving Fáry's Theorem

Recall [Fáry's Theorem](#) ([Theorem 13.28](#)), stating that there exists a straight-line plane embedding of any planar graph. The purpose of this project is to guide the reader in a step-by-step fashion towards proving this result.

We first need an intermediate result, however. The so-called **art gallery problem** is a celebrated visibility problem in the field of computational geometry. It derives from the real-world problem of guarding the entire contents of an art gallery (modelled by a polygon representing its plan view), employing the smallest possible number of guards so that each point in the interior of the art gallery is within the line of sight of at least one guard. The problem was originally posed by Victor Klee to [Václav Chvátal](#) in 1973, who solved it shortly thereafter [[11](#)]. We guide the reader through [Fisk's](#) [[20](#)] simplified proof of Chvátal's result.

Let $n \geq 3$ be an integer. A set of points \mathcal{S} within a simple, plane polygon \mathcal{P}_n with n vertices (or corner points) is said to **guard** \mathcal{P}_n if, for every point p in the interior of \mathcal{P}_n , there exists some point $q \in \mathcal{S}$ such that the straight line segment between p and q lies wholly within \mathcal{P}_n .

Theorem 13.41 (Chvátal's Watchman Theorem) $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and sometimes necessary to guard a simple, plane polygon with n corner points.

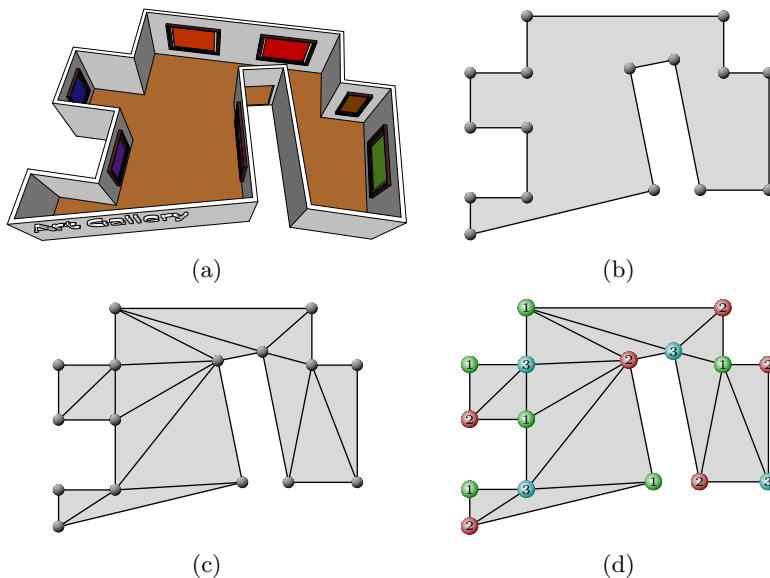


Figure 13.28: (a) A plan view of an art gallery layout with the museum interior shaded. (b) A polygon \mathcal{P}_{16} with 16 corner points representing the plan view of the gallery. (c) The inner-triangulated graph G associated with \mathcal{P}_{16} . (d) An assignment of the numbers 1, 2 and 3 to the vertices of G so that no two adjacent vertices of G receive the same colour.

Fisk's proof of [Theorem 13.41](#) proceeds by viewing the polygon as a graph whose vertices are the corner points of the polygon and whose edges are the sides of the polygon. The part of this (planar) graph corresponding to the interior of

the polygon is then **triangulated** to form a new graph G , *i.e.* edges are repeatedly added on the inside of the polygon between nonadjacent vertices until all faces of the interior are bounded by triangles. An example of a polygon \mathcal{P}_{16} , representing an art gallery with 16 corner points and its associated inner-triangulated graph G of order 16, are shown in [Figure 13.28\(a\)–\(c\)](#). Integer values are then assigned to the vertices of G — one value to each vertex — so that no two adjacent vertices in G receive the same integer value. This process of assigning integer values to the vertices of the inner-triangulated graph in [Figure 13.28\(c\)](#) is illustrated in [Figure 13.28\(d\)](#). Complete the proof by performing the following tasks:

Tasks

1. Prove that this process of assigning integer values to the vertices of G can always be performed using only the three integers 1, 2 and 3. (Hint: Consider the properties of the **dual graph** of G . The dual graph of G is a graph in which every vertex represents a face of G and in which two vertices are adjacent if the corresponding faces of G share a common edge along their boundaries.)
2. Explain why any subset of vertices of G assigned the same integer value forms a valid guard set \mathcal{S} of the original polygon.
3. Prove that some subset of vertices of G assigned the same integer value has cardinality no more than $\lfloor \frac{n}{3} \rfloor$.
4. Show that for every value $n \geq 3$ there is a polygon \mathcal{P}_n with n corner points which requires a set of exactly $\lfloor \frac{n}{3} \rfloor$ points to guard it.

We are now in a position to prove [Fáry's Theorem \(Theorem 13.28\)](#). Let G' be a planar graph of order $n \geq 3$ and let a , b and c be three vertices of G' . Fully triangulate G' to form a new (planar) graph G'' . We show, by induction over the order n of the triangulated graph G'' , that there exists a straight-line plane embedding of G'' so that the boundary of the outer face is the triangle on the three vertices a , b and c .

Tasks (continued)

5. Show that G'' has size $3n - 6$. (Hint: Use [Theorem 13.4](#) and the definition of triangulation.)
6. Establish the base case for the induction process corresponding to the special case where $n = 3$ (*i.e.* motivate why there exists a straight-line plane embedding of any triangulated planar graph G'' of order $n = 3$ on the three vertices a , b and c so that the boundary of the outer face is the triangle on the three vertices a , b and c).

Assume, as induction hypothesis, that there exists a straight-line plane embedding of any planar graph G'' of order $n = k \geq 3$ containing three vertices a , b and c so that the boundary of the outer face is the triangle on the three vertices a , b and c . Now consider, for the induction step, the case where $n = k + 1$.

Tasks (continued)

7. Explain why there is a vertex v in G'' with at most 5 neighbours that are distinct from a , b and c . (Hint: Use the result of [Task 4](#) of this project.)

8. Perform the induction step, forming a new graph G''' (of order k) by removing v from G'' and retriangulating the face \mathcal{F} formed by removing v . It follows by the induction hypothesis that G''' has a straight-line plane embedding so that the boundary of the outer face is the triangle on the three vertices a , b and c . Upon removal of the retriangulation edges in G''' that were not originally in G'' , a polygon with at most 5 corner points (*i.e.* the face \mathcal{F}) is recovered. Prove, by using [Chvátal's Watchman Theorem \(Theorem 13.41\)](#) that there is a point $p \in \mathcal{F}$ at which v may be placed (also inserting the edges that were originally incident with v) so as to form a straight-line embedding of G'' .

Project 13.3: The coarseness of a complete bipartite graph

The purpose of this project is to guide the reader in a step-by-step manner towards proving [Theorem 13.33](#), *i.e.* showing that the coarseness ξ of the complete bipartite graph $K_{p,q}$ satisfies

$$\left\lfloor \frac{p}{3} \right\rfloor \left\lfloor \frac{q}{3} \right\rfloor \leq \xi(K_{p,q}) \leq \left\lfloor \frac{pq}{9} \right\rfloor. \quad (13.9)$$

Tasks

1. Determine the size of $K_{p,q}$ and use [Theorem 13.31](#) to establish the upper bound on $\xi(K_{p,q})$ in (13.9).
2. Find a lower bound on the number of distinct copies of $K_{3,3}$ that appear as subgraphs of $K_{p,q}$.
3. Use [Theorem 13.6](#) and your result in [Task 2](#) above to establish the lower bound on $\xi(K_{p,q})$ in (13.9).
4. Deduce that if $p \equiv q \equiv 0 \pmod{3}$, then $\xi(K_{p,q}) = \frac{pq}{9}$.

Further reading

- [1] VB Alexeev and VS Gončakov, 1976. *Thickness of arbitrary complete graphs*, Matematicheskii Sbornik, **101(143)**, pp. 212–230.
- [2] L Auslander and S Perter, 1961. *On embedding graphs in the sphere*, Journal of Applied Mathematics and Mechanics, **10**, pp. 517–523.
- [3] LW Beineke, 1967. *The decomposition of complete graphs into planar subgraphs*, pp. 139–154 in F Harary (Ed), *Graph Theory and Theoretical Physics*, Academic Press, New York (NY).
- [4] LW Beineke and G Chartrand, 1968. *The coarseness of a graph*, Composito Mathematica, **19**, pp. 290–298.
- [5] LW Beineke and F Harary, 1965. *Inequalities involving the genus of a graph and its thickness*, Proceedings of the Glasgow Mathematical Association, **7**, pp. 19–21.
- [6] LW Beineke and F Harary, 1965. *The genus of the n-cube*, Canadian Journal of Mathematics, **17**, pp. 494–496.

- [7] LW Beineke, F Harary and JW Moon, 1964. *On the thickness of the complete bipartite graph*, Proceedings of the Cambridge Philosophical Society, **60**, pp. 1–6.
- [8] KS Booth and GS Luecker, 1976. *Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms*, Journal of Computer and System Sciences, **13**, pp. 335–379.
- [9] J Boyer and W Myrvold, 1999. *Stop minding your P's and Q's: A simplified $\mathcal{O}(n)$ planar embedding algorithm*, Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 140–146.
- [10] N Chiba, T Nishizeki, S Abe and T Ozawa, 1985. *A linear algorithm for embedding planar graphs using PQ-trees*, Journal of Computer and System Sciences, **30(1)**, pp. 54–76.
- [11] V Chvátal, 1975. *A combinatorial theorem in plane geometry*, Journal of Combinatorial Theory, Series B, **18**, pp. 39–41.
- [12] E de Klerk, DV Pasechnik and A Schrijver, 2007. *Reduction of symmetric semidefinite programs using the regular *-representation*, Mathematical Programming, **109**, pp. 613–624.
- [13] G Demoucron, Y Malgrange and R Pertuiset, 1964. *Graphes planaires: Reconnaissance et construction des représentations planaires topologiques*, Revue Française de Recherche Opérationnelle, **8(30)**, pp. 33–47.
- [14] GA Dirac and S Schuster, 1954. *A theorem of Kuratowski*, Proceedings of the Koninklijke Nederlandse Academie van Wetenschappen, Series A: Mathematical Sciences, **57**, pp. 343–348.
- [15] L Euler, 1758. *Demonstratio nunnularum insignium proprietatum quibus solida hedris planis inclusa sunt praedita*, Novi Commentarii academiae scientiarum Petropolitanae, **4**, pp. 140–160.
- [16] S Even and R Tarjan, 1976. *Computing an ST numbering*, Theoretical Computer Science, **2**, pp. 339–344.
- [17] G Exoo, *Rectilinear drawings of famous graphs*, URL: <http://isu.indstate.edu/ge/COMBIN/RECTILINEAR/>.
- [18] I Fáry, 1948. *On straight-line representation of planar graphs*, Acta Scientiarum Mathematicarum, **11**, pp. 229–233.
- [19] MR Fellows, 1987. *The Robertson-Seymour theorems: A survey of applications*, Contemporary Mathematics, **89**, pp. 1–18.
- [20] S Fisk, 1978. *A short proof of Chvátal's watchman theorem*, Journal of Combinatorial Theory, Series B, **24**, p. 374.
- [21] A Gagarina, W Myrvold and J Chambers, 2005. *Forbidden minors and subdivisions for toroidal graphs with no $K_{3,3}$'s*, Electronic Notes in Discrete Mathematics, **22**, pp. 151–156.
- [22] MR Garey and DS Johnson, 1983. *Crossing number is NP-complete*, SIAM Journal on Algebra and Discrete Mathematics, **4**, pp. 312–316.
- [23] RK Guy, 1969. *The decline and fall of Zarankiewicz's theorem*, pp. 63–69 in F Harary (Ed), *Proof Techniques in Graph Theory*, Academic Press, New York (NY).

- [24] RK Guy, 1972. *Crossing numbers in graphs*, pp. 111–124 in Y Alavi, DR Lick and AT White (Eds), *Graph Theory and Applications: Proceedings of the Conference at Western Michigan University, Kalamazoo*, Springer-Verlag, New York (NY).
- [25] RK Guy and TA Jenkyns, 1969. *The toroidal crossing number of $K_{m,n}$* , Journal of Combinatorial Theory, **6**, pp. 235–250.
- [26] RK Guy, TA Jenkyns and J Schaefer, 1968. *The toroidal crossing number of the complete graph*, Journal of Combinatorial Theory, **4**, pp. 376–390.
- [27] J Hopcroft and RE Tarjan, 1974. *Efficient planarity testing*, Journal of the Association for Computing Machinery, **21**, pp. 549–568.
- [28] DJ Kleitman, 1970. *The crossing number of $K_{5,n}$* , Journal of Combinatorial Theory, **9**, pp. 315–323.
- [29] K Kuratowski, 1930. *Sur le problème des courbes gauches en Topologie*, Fundamenta Mathematicae, **15**, pp. 271–283.
- [30] A Lempel, S Even and L Cederbaum, 1967. *An algorithm for planarity testing of graphs*, pp. 215–323 in *Theorey of Graphs*, Proceedings of an International Symposium held in Rome in 1966, Rome.
- [31] K Mehlhorn, P Mutzel and S Naher, 1993. *An implementation of the Hopcroft and Tarjan planarity test and embedding algorithm*, Technical Report MPI-1-93-151, Max Planck Institut fur Informatik, Saarbrucken.
- [32] B Mohar, 2001. *Graph minors and graphs on surfaces*, Invited talk at the 18th British Combinatorial Conference, Sussex.
- [33] S Pan and RB Richter, 2007. *The crossing number of K_{11} is 100*, Journal of Graph Theory, **56**, pp. 128–134.
- [34] RB Richter and J Širáň, 1996. *The crossing number of $K_{3,n}$ in a surface*, Journal of Graph Theory, **21**, pp. 51–54.
- [35] RB Richter and C Thomassen, 1997. *Relations between crossing numbers of complete and complete bipartite graphs*, American Mathematical Monthly, **104**, pp. 131–137.
- [36] G Ringel, 1965. *Das Geschlecht des vollständiger paaren Graphen*, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, **28**, pp. 139–150.
- [37] G Ringel and JWT Youngs, 1968. *Solution of the Heawood map-coloring problem*, Proceedings of the National Academy of Sciences, **60**, pp. 438–445.
- [38] N Robertson and PD Seymour, 1983. *Graph minors. I — Excluding a forest*, Journal of Combinatorial Theory, Series B, **35**, pp. 39–61.
- [39] N Robertson and PD Seymour, 2004. *Graph minors. XX — Wagner's conjecture*, Journal of Combinatorial Theory, Series B, **92**, pp. 325–357.
- [40] TL Saaty, 1967. *Two theorems on the minimum number of intersections for complete graphs*, Journal of Combinatorial Theory, **2**, pp. 571–584.
- [41] W-K Shih and W-L Hsu, 1999. *A new planarity test*, Theoretical Computer Science, **223(1-2)**, pp. 179–191.

- [42] NJA Sloane, *The on-line encyclopedia of integer sequences*, URL: <http://www.research.att.com/~njas/sequences>.
- [43] SK Stein, 1951. *Convex maps*, Proceedings of the American Mathematical Society, **2**, pp. 464–466.
- [44] C Thomassen, 1980. *Planarity and duality of finite and infinite graphs*, Journal of Combinatorial Theory, Series B, **29**, pp. 244–271.
- [45] C Thomassen, 1989. *The graph genus problem is NP-complete*, Journal of Algorithms, **10**, pp. 568–576.
- [46] P Turán, 1977. *A note of welcome*, Journal of Graph Theory, **1**, pp. 7–9.
- [47] WT Tutte, 1960. *Convex representations of graphs*, Proceedings of the London Mathematical Society, **10**, pp. 304–320.
- [48] WT Tutte, 1963. *How to draw a graph*, Proceedings of the London Mathematical Society, **13**, pp. 743–767.
- [49] K Urbanik, 1955. *Solution du problème posé par P. Turán*, Colloquium Mathematicum, **3**, pp. 200–201.
- [50] JM Vasak, 1976. *The thickness of the complete graph*, PhD thesis, University of Illinois at Urbana-Champaign, Champaign (IL).
- [51] K Wagner, 1936. *Bemerkungen zum Vierfarbenproblem*, Jahresbericht der Deutschen Mathematiker-Vereinigung, **46**, pp. 26–32.
- [52] K Wagner, 1937. *Über eine Eigenschaft der ebene Komplexe*, Mathematische Annalen, **114**, pp. 570–590.
- [53] DR Woodall, 1993. *Cyclic-order graphs and Zarankiewicz's crossing-number conjecture*, Journal of Graph Theory, **17**, pp. 657–671.
- [54] JWT Youngs, 1963. *Minimal embeddings and the genus of a graph*, Journal of Mathematics and Mechanics, **12**, pp. 303–315.
- [55] K Zarankiewicz, 1954. *On a problem of P. Turán concerning graphs*, Fundamenta Mathematicae, **41**, pp. 137–145.



Graph colouring

Contents

14.1	Introduction	449
14.2	Vertex colouring	451
14.3	Edge colouring	477
	Exercises	482
	Computer exercises	486
	Projects	488
	Further reading	497

14.1 Introduction

Suppose the mathematics department of a university wishes to schedule its examination timetable. The department offers ten courses, some of which may be taken in conjunction with each other (*i.e.* simultaneously, by the same students), and some which may not, as outlined in [Table 14.1](#). Of course, examinations for courses that may be taken in conjunction with one another cannot be scheduled in the same examination time slot, and the department wishes to draw up an examination time table with the minimum number of time slots, so as to minimise invigilation costs.

Course number	Course name	Courses that may be taken in conjunction
v_1	Calculus	v_2, v_5, v_6, v_7, v_9
v_2	Analytic geometry	v_1, v_5, v_9
v_3	Abstract algebra	v_6, v_8
v_4	Complex analysis	v_5, v_9, v_{10}
v_5	Functional analysis	v_1, v_2, v_4
v_6	Graph theory	v_1, v_3, v_7, v_8
v_7	Linear algebra	v_1, v_6
v_8	Number theory	v_3, v_6
v_9	Numerical analysis	v_1, v_2, v_4
v_{10}	Topology	v_4

Table 14.1: Courses offered by a mathematics department.

Time slot 1	Time slot 2	Time slot 3
1. Calculus	1. Analytic geometry	1. Functional analysis
2. Abstract algebra	2. Complex analysis	2. Graph theory
3. Topology	3. Linear algebra	3. Numerical analysis
	4. Number theory	

Table 14.2: Examination timetable corresponding to the vertex colouring of the course graph in Figure 14.1(b).

Being a mathematics department, they employ the power of graph theory to draw up the desired timetable. A graph, called the **course graph**, is constructed, in which the vertices represent the ten subjects offered by the department, and in which two vertices are adjacent if the corresponding courses may be taken in conjunction with each other, as shown in Figure 14.1(a). Suppose the vertices of the course graph are coloured so that each vertex is assigned a colour, ensuring that no two adjacent vertices receive the same colour. Then the colours may represent examination time slots in the sense that all courses corresponding to vertices coloured with colour 1 may be scheduled together without a clash in time slot 1 of the timetable, all courses corresponding to vertices coloured with colour 2 may be scheduled together without a clash in time slot 2 of the timetable, and so on. The problem of drawing up an examination timetable with the minimum number of time slots then reduces to the problem of colouring the vertices of the course graph with the minimum number of colours, in such a way that no two adjacent vertices receive the same colour. For the course graph in Figure 14.1(a) a colouring of the vertices in three colours, as shown in Figure 14.1(b), is best possible (why?). The examination timetable corresponding to this optimal colouring is shown in Table 14.2.

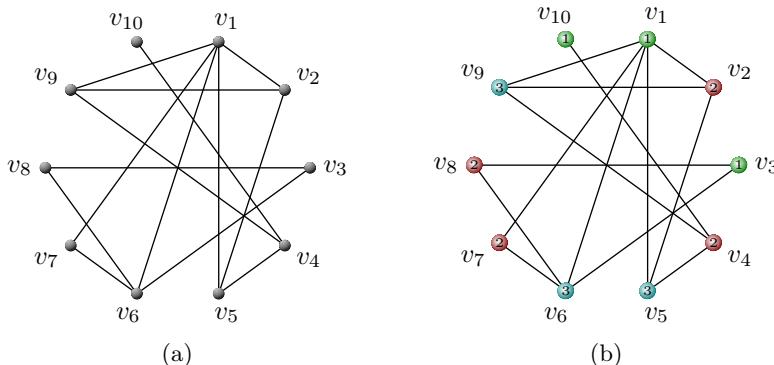


Figure 14.1: (a) The course graph corresponding to the courses in Table 14.1, and (b) a colouring of the vertices of the course graph in three colours.

This colouring problem is an instance of a classical problem in graph theory with a very rich history indeed, as we shall outline in this chapter.

14.2 Vertex colouring

Let us make the colouring problem outlined in the above introduction more precise. A (**proper**) **vertex colouring** of a graph G is an assignment of colours (typically represented by elements of some set) to the vertices of G , one colour to each vertex, so that no two adjacent vertices of G receive the same colour. If k colours are used, then the colouring is referred to as a (**proper**) **k -colouring** of G . A k -colouring of G thus partitions the vertex set of G into k vertex subsets, called **colour classes**, each containing vertices of the same colour.

The colouring in Figure 14.1(b) is therefore a 3-colouring of the course graph in Figure 14.1(a), with colour classes $\{v_1, v_3, v_{10}\}$, $\{v_2, v_4, v_7, v_8\}$, $\{v_5, v_6, v_9\}$.

14.2.1 k -Colourability

A graph G is **k -colourable** if there exists a k -colouring of G . The course graph in Figure 14.1(a) is therefore 3-colourable, but not 2-colourable.

Consider the decision problem, denoted by $D_{\text{colour}}(G, k)$, of deciding whether a given graph G of order n is k -colourable, for some $k \in \mathbb{N}$. Since the only graphs that are 1-colourable are empty (or edgeless) graphs, and since it may certainly be tested in polynomial time whether a graph is edgeless or not, it follows that $D_{\text{colour}}(G, 1) \in P$ (i.e. $D_{\text{colour}}(G, 1)$ can be solved in polynomial time). It turns out that it is also computationally easy to solve the decision problem $D_{\text{colour}}(G, 2)$.

Theorem 14.1 *A nontrivial graph is 2-colourable if and only if it is bipartite.*

Proof Suppose G is a bipartite graph with partite sets X and Y . Then no two vertices of X are adjacent, and similarly for Y . Hence there exists a 2-colouring of G with colour classes X and Y . Conversely, suppose X and Y are the colour classes of a 2-colouring of G . Then no two vertices of X are adjacent, and similarly for Y . Hence, G is bipartite with partite sets X and Y . ■

Suppose a graph G has ℓ components. Let v_i be an arbitrary vertex of the i -th component, let \mathcal{X}_i be the set of vertices in the i -th component of G at even distance from v_i and let \mathcal{Y}_i be the set of vertices in the i -th component of G at odd distance from v_i , for all $i \in [\ell]$. Then G is bipartite if and only if no two vertices in \mathcal{X}_i are adjacent and if no two vertices in \mathcal{Y}_i are adjacent, for all $i \in [\ell]$, in order to avoid odd cycles in the components of a bipartite graph (prohibited by Theorem 2.3). Since it may be tested in polynomial time whether or not these conditions are satisfied for the sets \mathcal{X}_i and \mathcal{Y}_i (see Project 14.1), it follows that $D_{\text{colour}}(G, 2) \in P$. The following result is due to Karp [37] and dates from 1972.

Theorem 14.2 $D_{\text{colour}}(G, k)$ is NP-complete for all $k > 2$.

Note that $D_{\text{colour}}(G, k) \in \text{NP}$ for all $k > 2$; a k -colouring of G is a certificate to an instance of the problem (given such a certificate, one may test in $\mathcal{O}(n^2)$ time whether the certificate is, in fact, a valid k -colouring of G). Karp [37] proved Theorem 14.2 by polynomial-time reducing an instance of 3-SAT to an instance of $D_{\text{colour}}(G, k)$.

It is, nevertheless, possible to solve the decision problem $D_{\text{colour}}(G, k)$ in polynomial time for certain graph classes and for arbitrary values of k , as we demonstrate next. Recall, from [Chapter 13](#), that a planar graph is a graph that may be embedded in the plane (that is, drawn in the plane without edge intersections). The following theorem solves the decision problem $D_{\text{colour}}(G, 5)$ for the class of planar graphs G .

Theorem 14.3 (Five-colour Theorem) *Every planar graph is 5-colourable.*

Proof We proceed by induction over the order of the planar graph. The result is certainly true for all graphs of order at most 5. Suppose therefore, as induction hypothesis, that every planar graph of order $n - 1$ is 5-colourable and let G be a plane graph embedding of a planar graph of order n . Then G has a vertex v of degree at most 5 by [Theorem 13.5](#). Since $G - v$ is a graph of order $n - 1$ it follows by the induction hypothesis that $G - v$ is 5-colourable, with the colours 1, 2, 3, 4 and 5 (say). If $d(v) < 5$, or if $d(v) = 5$ and one of these colours has not been used to colour the neighbours of v , then v may be coloured with that colour, producing a 5-colouring of G .

Assume, therefore, that $d(v) = 5$ and that all five colours have been used to colour the neighbours of v in $G - v$. Let v_1, \dots, v_5 be the five neighbours of v , arranged cyclically around v in the plane, as shown in [Figure 14.2\(a\)](#). Suppose that v_1 has been coloured with colour 1, that v_2 has been coloured with colour 2, and so on. We show that it is possible to recolour some of the vertices of $G - v$, including a neighbour of v , so that one of the five colours becomes available to colour v as well.

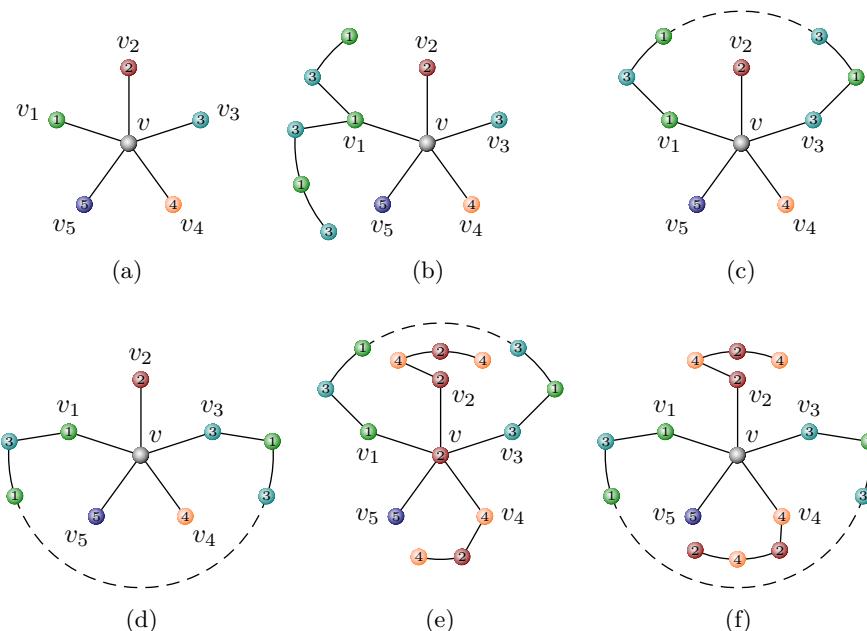


Figure 14.2: The vertex v and its five neighbours v_1, \dots, v_5 arranged cyclically around v in the proof of [Theorem 14.3](#).

Consider those vertices of $G - v$ that have been coloured with colours 1 and 3, including v_1 and v_3 , of course. Suppose there is no v_1 - v_3 path in $G - v$ all of whose vertices have been coloured with colours 1 and 3 (alternately) and consider all paths in $G - v$ starting at v_1 and whose vertices have been coloured with colours 1 and 3 (alternately). These paths induce a subgraph H of $G - v$ not containing v_3 , as shown in [Figure 14.2\(b\)](#). If the colours 1 and 3 are interchanged in H , a new 5-colouring of $G - v$ is produced in which the vertices v_1 and v_3 are both coloured with colour 3. Colour 1 may therefore be assigned to the vertex v in order to produce a 5-colouring of G .

Suppose, therefore, that there is indeed a v_1 - v_3 path P in $G - v$ all of whose vertices have been coloured with colours 1 and 3 (alternately). The path P together with the path v_3 - v_1 produces a cycle in G which either encloses v_2 or encloses both of v_4 and v_5 , as shown in [Figure 14.2\(c\)-\(d\)](#). Hence there exists no v_2 - v_4 path in G all of whose vertices have been coloured with colours 2 and 4 (alternately) — recall that G is a plane graph. Let H' be the subgraph of G induced by all the paths in G starting at v_2 and whose vertices have been coloured with colours 2 and 4 (alternately). Then H' does not contain v_4 , as shown in [Figure 14.2\(e\)-\(f\)](#). If the colours 2 and 4 are interchanged in H' , a new 5-colouring of $G - v$ is produced in which the vertices v_2 and v_4 are both coloured with colour 4. Colour 2 may therefore be assigned to the vertex v in order to produce a 5-colouring of G . ■

It is, in fact, possible to improve the [Five-colour Theorem](#) as follows.

Theorem 14.4 (Four-colour Theorem) *Every planar graph is 4-colourable.*

The price of this improvement, however, is that the proof of [Theorem 14.4](#) is by no means simple — in fact, it has a rich and colourful history! Interest in this theorem originally arose in the context of map colouring, when the theorem was still a conjecture.

When a map is coloured in an atlas, adjacent countries receive different colours so that one may distinguish between them. Naturally, the publisher of such an atlas would like to use as small a number of colours as possible to colour the map, in order to save on the cost of publishing the map. A graph representing the neighbouring topology of a map may be formed in which the countries are represented by vertices, as shown in [Figure 14.3\(a\)-\(b\)](#), and in which vertices representing countries with common borders of positive length are adjacent. The problem of colouring the original map, using a certain number of colours, is then equivalent to the problem of colouring the vertices of the resulting planar graph with the same number of colours (so that no two adjacent vertices receive the same colour).

Consider, as an example, the 4-colouring of the graph $G_{14.1}$ in [Figure 14.3\(c\)](#). This 4-colouring shows that $G_{14.1}$ is 4-colourable and hence that the original map of Australia in [Figure 14.3\(a\)](#) may be coloured using only four colours.

Cartographers had long noticed that it was possible to colour any map they could lay their hands on using only four colours (hence the [Four-colour Conjecture](#)), but since no proof of this result was available for a general map, there was no guarantee that some strange map could not be devised which could not be coloured with four colours. Until 1976, proving the [Four-colour Conjecture](#) was one of the most famous unsolved problems of the time in all of mathematics, ranking in stature with tantalizing unresolved problems such as *Fermat's last theorem*

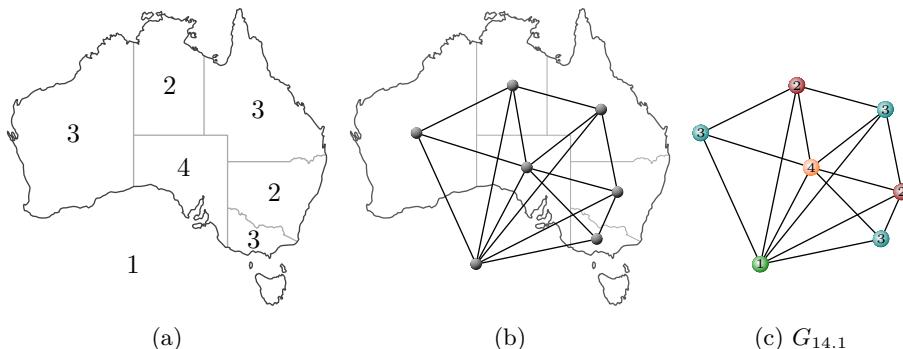


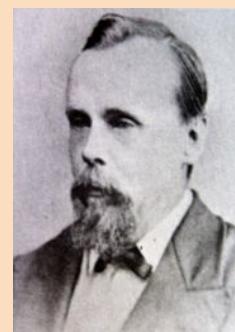
Figure 14.3: The provinces of Australia are represented in the graph $G_{14.1}$ in (c) by vertices and these vertices have the same colours as the provinces in the map in (a). Two vertices in $G_{14.1}$ are adjacent if the two provinces they represent have a common boundary in the original map, as indicated in (b).

(stating that $x^n + y^n = z^n$ has no integer solutions for x, y and z when $n > 2$ and which was subsequently proved by British mathematician Andrew Wiles in May 1995 [1, 41]), the *Riemann-hypothesis* (also Hilbert’s Eighth Problem [33], stating that the nontrivial zeros of the Riemann zeta-function $\zeta(s) = \sum_{n=1}^{\infty} n^{-s}$ all have real part $\frac{1}{2}$ [20, 41]) and the Goldbach Conjecture (stating that every even integer greater than 2 may be expressed as the sum of two primes [20]).

The [Four-colour Conjecture](#) seems to have been formulated for the first time by [Francis Guthrie](#) while he was a student at University College, London. He attempted to prove the conjecture, but was not satisfied with his proof [47]. He therefore mentioned the problem to his brother Frederick, also a student at University College, London [24]. Frederick Guthrie, in turn, asked his mathematics professor, [Augustus de Morgan](#) (for whom [De Morgan](#)’s Laws of set theory are named [18]), to verify the “fact” that any map drawn in the plane could be coloured with at most four colours, so that adjacent countries received different colours. [De Morgan](#) responded that he did not know that this was indeed a “fact.” [De Morgan](#) subsequently often spoke of this conjecture with other mathematicians and he is credited with writing an anonymous article in the April 14, 1860 issue of the journal *Athenaeum* in which he described the [Four-colour Conjecture](#). This is the first published reference to the conjecture [18, 47]. [De Morgan](#) probably also communicated the problem to the English mathematician [Arthur Cayley](#) and to a London lawyer [Alfred Bray Kempe](#) [24] who had studied mathematics under [Cayley](#) at Cambridge and devoted some of his time to mathematics throughout his life [46].

Almost twenty years after it had first appeared in print, the [Four-colour Conjecture](#) was raised as an open problem by [Cayley](#) at a meeting of the London Mathematical Society on 13 June, 1878 [14, 18, 47] as well as in an 1879 paper by [Cayley](#) in which he explained why this conjecture appeared to be so difficult to prove [15, 18, 35]. During that same year, [Kempe](#) published what seems to be the first attempted proof of the [Four-colour Conjecture](#) [18, 27, 39]. [Kempe](#) made use of what is today called *Kempe chains* to recolour parts of a map, where necessary, so as to be able to colour some uncoloured country (see [Project 14.2](#)). [Kempe](#) received great acclaim for his “proof.” Based on this argument, as well as his work on

Francis Guthrie was born in London on 22 January 1831. He was a student of John Lindley, professor of botany at University College, London while his brother Frederick was a student of [Augustus de Morgan](#), professor of mathematics also at University College, London. He obtained his bachelor's degree in the arts in 1850, and an LLB degree in 1852. Guthrie moved to South Africa on 10 April 1861, taking up the post of mathematics master at Graaff-Reinet College, where he delivered a course of acclaimed public lectures on botany in 1862. He moved to Cape Town in 1875 where he practiced at the Bar and edited a newspaper before becoming professor of mathematics at the South African College, which later became the University of Cape Town. He remained there from 1876 until his retirement in 1898, staying on his farm at Raapenberg. When Harry Bolus undertook to describe the family of *Ericaceae* for *Flora Capensis*, he enlisted Guthrie's aid and they collaborated until Guthrie's death. Before his death, Guthrie had made an extensive collection of the Cape Peninsula flora, which was eventually housed as the Guthrie Herbarium in the Botany Department at the University of Cape Town; this collection is still used for teaching and reference. Though Guthrie did not live to see the published work, he had the satisfaction of knowing that the greater part of the work on *Ericaceae* had been completed. He died in Claremont, Cape Town on 19 October 1899 and is buried in the old cemetery attached to St Thomas's Church in Rondebosch.



Biographic note 52: Francis Guthrie (1831–1899)

linkages [58], he was elected Fellow of the Royal Society and served as its treasurer for many years. He was knighted in 1912 [46].

The [Four-colour Theorem](#) continued to capture the imagination of many professional and amateur mathematicians. For example, Peter Guthrie Tait, Professor of Natural Philosophy at the University of Edinburgh described yet another “proof” [18, 46, 47]. It contained some clever ideas, but unfortunately also a number of basic errors [46]. Lewis Carroll, author of the famous children’s story “Alice in Wonderland,” created a game for two players in which one player designed a map for his or her opponent to four-colour. In 1889, the then Bishop of London (Frederick Temple), later Archbishop of Canterbury, published his own “proof” of the [Four-colour Theorem](#) [18, 47]. Temple had considered it sufficient to prove the [Four-colour Theorem](#) by showing that it is impossible to draw five mutually neighbouring regions in the plane, *i.e.* where each region borders the other four. If there is a map with five neighbouring regions, then the [Four-colour Theorem](#) is certainly false. From this last fact, Temple, thus made the incorrect logical deduction that if there is no a map with five neighbouring regions, then the [Four-colour Theorem](#) is true [58].

Unfortunately, more than a decade after [Kempe](#)’s original “proof” was published, the [Four-colour Theorem](#) returned to being the [Four-colour Conjecture](#), when [Kempe](#)’s proof was refuted in 1890 by [Heawood](#) in his first paper [30, 35].

Augustus de Morgan was born in Madurai, Madras Presidency, India on 27 June 1806. His father, Colonel Augustus de Morgan, held various appointments in India in the service of the East India Company. The family moved to England when Augustus was seven months old. In 1823, at the age of sixteen, he entered Trinity College, Cambridge, where his tutor was John Philips Higman. At the age of 22 years he was appointed professor of mathematics at University College, London, where he remained for many years and made numerous fundamental contributions, including the construction of inventories of the fundamental symbols of algebra and of the laws of algebra, formulating the laws of logic named after him, introducing the term *mathematical induction* and making its idea rigorous. Five years after his resignation from University College in 1866, De Morgan died of nervous prostration on 18 March 1871, following the death of his son. In addition to his considerable mathematical legacy, the headquarters of the London Mathematical Society is called *De Morgan House*, the student society of the Mathematics Department of University College, London is called the *Augustus de Morgan Society*, and the crater De Morgan on the Moon is named after him.



Biographic note 53: Augustus de Morgan (1806–1871)

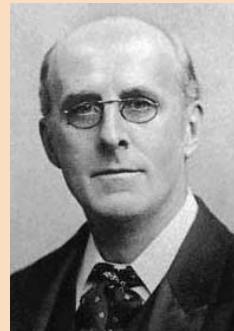
Heawood, a lecturer at Durham in England [46], stated almost apologetically that he had discovered an error in Kempe's proof that is so serious that he was unable to repair it [18]. Kempe reported the error to the London Mathematical Society himself, stating that he too could not correct the mistake [46]. The reader is invited to investigate the error in Kempe's proof attempt in Project 14.2 at the end of the chapter.

Although Kempe's work contained a flaw, it also contained a valuable contribution which formed the basis of many later attempts to prove the Four-colour Conjecture, including the successful attempt of Appel and Haken in 1976 [27]. Furthermore, Heawood was able to use Kempe's technique to prove the Five-colour Theorem. In fact, the proof we gave for Theorem 14.3 is Heawood's proof.

Despite the fact that it is not particularly difficult to prove the Five-colour Theorem, it is extraordinarily difficult to prove the Four-colour Theorem. It remained a conjecture for several decades until 1976, when the Four-colour Conjecture became the Four-colour Theorem for the second, and last, time. On June 21, 1976, Kenneth Appel and Wolfgang Haken of the University of Illinois, announced that, with the computer aid of John Koch, they had found a complete computer proof of the Four-colour Conjecture [2, 18, 28, 35]. Their proof made use of two fundamental ideas developed during the first half of the twentieth century, namely so-called unavoidable sets and reducible configurations.

It follows from Theorem 13.6 that every map has at least one country with five or fewer neighbours. This means that in every map there must be either a country with one, a country with two, a country with three, a country with four or a country with five neighbours. Hence, in every map at least one country from this

Alfred Bray Kempe was born in Kensington, London on 6 July 1849. He studied at Trinity College, Cambridge where [Arthur Cayley](#) was one of his teachers. He graduated with a bachelor's degree in the arts in 1872. Despite his interest in mathematics, he became a barrister, specialising in ecclesiastical law. He was nevertheless active as an amateur mathematician, developing the notion of straight line linkages and publishing his influential lectures on the subject in 1877. Kempe's universality theorem for linkages states that any bounded subset of an algebraic curve may be traced out by the motion of one of the joints in a suitably chosen linkage. Kempe's proof was, however, flawed, and the first complete proof was only provided in 2002, based on his ideas. The development of the notion of multisets may also be attributed to Kempe, although this was not recognised until long after his death. He was an avid mountain climber, mostly in Switzerland. Kempe was elected Fellow of the Royal Society in 1881 and served as president of the London Mathematical Society from 1892 to 1894. He was knighted in 1912, became the Chancellor for the Diocese of London and received an honorary degree from the University of Durham. He died in London on 21 April 1922.



Biographic note 54: Alfred Kempe (1849–1922)

collection cannot be avoided and such a collection is called an **unavoidable set** [5, 57]. A **reducible configuration** is any arrangement of countries that cannot occur in a minimal nonfour-colourable map. If a map contains a reducible configuration, then any colouring of the remainder of the map with four colours may be extended, perhaps after necessary local recolouring, to a colouring of the entire map [5, 57].

The approach in [Appel](#) and [Haken](#)'s proof of the [Four-colour Theorem](#) was to find an unavoidable set of reducible configurations [58]. To determine unavoidability they made use of a refinement of Heesch's method of **discharging** which assigns a charge of $6 - i$ to each vertex, where i is the degree of the vertex. One can then prove that a set of configurations is unavoidable if it is possible to distribute the charges so that the vertices of a triangulation all have negative charges [5, 6, 31, 57]. Since the set is unavoidable, every map must contain at least one of the configurations, but each configuration is reducible and thus cannot be contained in a minimal nonfour-colourable map. Consequently, no minimal nonfour-colourable map exists [58]. The unavoidable set in [Appel](#) and [Haken](#)'s proof has cardinality 1476 [53].

The [Four-colour Theorem](#) was the first major theorem to be proved with the aid of a computer — resulting in a proof that could not be verified directly by other mathematicians [46]. For this reason, as well as the fact that their proof is rather lengthy, the proof of [Appel](#) and [Haken](#) was met with skepticism at first, especially since the proposed solution had required hundreds of hours of computer calculations to test all 1476 configurations for reducibility. [Appel](#) and [Haken](#), in fact, remarked in 1986 that “this leaves the reader to face 50 pages containing text and diagrams, 85 pages filled with almost 2500 additional diagrams, and 400

Percy John Heawood was born in Newport, Shropshire, England on 8 September 1861. He studied at Queen Elizabeth's School, Ipswich and later at Exeter College, Oxford. He spent his career at Durham University, where he was appointed as lecturer in 1885, senior proctor of the university in 1901, professor of mathematics in 1910, and vice-chancellor during the period 1926–1928. GA Dirac remarked as follows about him in the Journal of the London Mathematical Society: “In his appearance, manners and habits of thought, Heawood was an extravagantly unusual man. He had an immense moustache and a meagre, slightly stooping figure. He usually wore an Inverness cape of strange pattern and manifest antiquity, and carried an ancient handbag. His walk was delicate and hasty, and he was often accompanied by a dog, which was admitted to his lectures. . . His transparent sincerity, piety and goodness of heart, and his eccentricity and extraordinary blend of naiveté and shrewdness secured for him not only the fascinated interest, but also the regard and respect of his colleagues.” He died in Durham, England on 24 January 1955. Durham University still annually awards a *Heawood Prize* in his honour to a student graduating in mathematics whose performance is outstanding in the final year.



Biographic note 55: Percy Heawood (1861–1955)

microfiche pages that contain further diagrams and thousands of individual verifications of claims made in the 24 lemmas in the main sections of text. In addition, the reader is told that certain facts have been verified with the use of about twelve hundred hours of computer time and would be extremely time-consuming to verify by hand. The papers are somewhat intimidating due to their style and length and few mathematicians have read them in any detail.” This prompted [Neil Robertson](#), [Daniel Sanders](#), [Paul Seymour](#) and [Robin Thomas](#) to publish a new proof of the [Four-colour Theorem](#) in 1997 [49]. Their 42-page proof is also computer assisted, but the computer programs are available for independent verification purposes. In addition, the proof yields a quadratic-time algorithm for four-colouring planar graphs.

The story of the [Four-colour Theorem](#) has over the years inspired a considerable amount of theory and has spawned many related problems. More than two dozen equivalent formulations of the theorem have been put forward (in terms of, for example, vector cross products, Lie algebras, and divisibility). Generalisations of the theorem have also been conjectured. One of these conjectures is by none other than [Percy Heawood](#):

Conjecture 14.5 *If G is a graph of genus g (see [Section 13.6.3](#)), then G is $(7 + \sqrt{48g + 1})/2$ -colourable.*

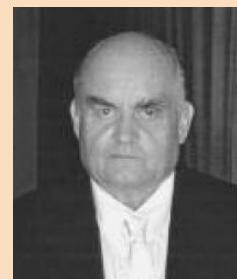
This conjecture has also had an interesting history. Peter Guthrie Tait attempted to prove [Conjecture 14.5](#) in 1880, but errors in his “proof” were discovered independently by [Julius Petersen](#) in 1891 and [William Tutte](#) in 1954.

Kenneth Ira Appel was born in Brooklyn, New York on 8 October 1932. He studied at the University of Michigan and later conducted research at the Institute for Defense Analyses in Princeton before he became a faculty member at the University of Illinois at Urbana-Champaign in 1961. Appel was chair of the mathematics department at the University of New Hampshire in Durham, New Hampshire during the period 1993–2002 and, although retired, still worked and occasionally taught there afterwards. He was elected Fellow of the American Mathematical Society in 2012. He died in Dover, New Hampshire, on 19 April 2013, at the age of 80, after a year-long battle with cancer. He was the father of the computer scientist Andrew Appel.



Biographic note 56: Kenneth Appel (1932–2013)

Wolfgang Haken was born in Berlin, Germany on 21 June 1928, but subsequently moved to the United States of America. He specialised in topology, and in particular on 3-manifolds. Haken introduced several important mathematical notions, including *Haken manifolds*, *Kneser-Haken finiteness*, and an expansion of the work of Kneser into a theory of normal surfaces. He adopted an algorithmic approach in much of his work, and he was one of the influential figures in algorithmic topology. One of his key contributions to this field is an algorithm for detecting whether or not a knot is unknotted. Wolfgang Haken was the recipient of the 1979 Fulkerson Prize of the American Mathematical Society for his solution with **Kenneth Appel** of the four-colour problem.



Biographic note 57: Wolfgang Haken (1928–present)

- ❖ The reader should now be able to attempt Exercises 14.1–14.5 and Projects 14.1–14.2.

14.2.2 Criticality and the chromatic number of a graph

The (**vertex**) **chromatic number** of a graph G , denoted by $\chi(G)$, is the smallest integer k for which G is k -colourable. If $\chi(G) = k$, then the graph G is said to be **k -chromatic**. A proper (vertex) colouring of G using $\chi(G)$ colours is called a **χ -colouring of G** . In view of the NP-completeness of the decision problem $D_{\text{colour}}(G, k)$ for an arbitrary graph and for $k > 2$, computing $\chi(G)$ for a general graph G is a hard problem. The notion of *criticality* with respect to the chromatic number of a graph, however, provides a useful tool during attempts to establish the value of this parameter.

It is a simple matter to verify that if H is a subgraph of G , then $\chi(H) \leq \chi(G)$. A graph G is, however, said to be **critical with respect to $\chi(G)$** if, for every proper

Peter Christian Julius Petersen was born on 16 June 1839 in Sorø on Zealand in the east of Denmark. He obtained a master's degree in mathematics in 1866 and a doctorate in mathematics in 1871, both from the University of Copenhagen. Petersen had a wide variety of interests in mathematics, including in geometry, complex analysis, number theory, mathematical physics, mathematical economics, cryptography and graph theory. In 1891, he wrote a seminal paper entitled *Die Theorie der regulären graphs* [48]. This contribution is widely considered to have been instrumental in the emergence of modern graph theory at the end of the nineteenth century. In 1898, he presented a counter example to Tait's claim about 1-factorability of 3-regular graphs, which is today known as the "Petersen graph." The graph has, in fact, often been used afterwards to refute conjectures. Petersen was a prolific author, contributing to many areas of mathematics and writing several textbooks. He died in Copenhagen on 5 August 1910.



Biographic note 58: Julius Petersen (1839–1910)

subgraph H of G , it holds that $\chi(H) < \chi(G)$. Also, a graph G is **k -critical with respect to $\chi(G)$** if it is k -chromatic and critical with respect to $\chi(G)$. In this section we shall omit the phrase "with respect to $\chi(G)$ " when referring to these concepts of criticality, instead referring only to the abbreviations *critical graph* and *k -critical graph* respectively, because the local context of usage of these terms in this section is clear.

Our first result reveals why the notion of k -criticality is useful when studying the chromatic number of a graph.

Theorem 14.6 *If G is a k -chromatic graph, then G contains a k -critical subgraph.*

Proof Suppose G is k -chromatic. If G is not k -critical, then it must contain a proper subgraph G_1 for which $\chi(G_1) = k$, for otherwise it would be possible to colour the vertices of G with fewer than k colours. Now either G_1 is k -critical, or it contains a proper subgraph G_2 for which $\chi(G_2) = k$. Proceeding in this way, we obtain a sequence

$$G \supsetneq G_1 \supsetneq G_2 \supsetneq G_3 \supsetneq \dots$$

in which every graph has chromatic number k . This sequence must terminate, since G is finite, and hence we conclude that there must be a k -critical graph G_ℓ in the sequence. ■

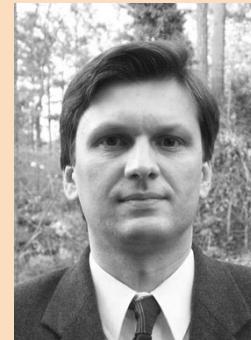
If G is a connected graph, then of course G is critical if and only if

$$\chi(G - e) < \chi(G) \tag{14.1}$$

for every edge e of G . This property, in fact, holds for *every* critical graph, as stated in the following theorem.

Theorem 14.7 *Every critical graph is connected.*

Robin Thomas was born on 22 August 1962 in what was then Czechoslovakia (now the Czech Republic). In 1985, he obtained a doctorate in mathematics from Charles University in Prague. He joined the faculty at the Mathematics Department of the Georgia Institute of Technology in Atlanta in 1989, where he became a Regents' Professor. Thomas was twice awarded the Fulkerson Prize for an outstanding paper in discrete mathematics — in 1994 for a paper on the Hadwiger conjecture with [Neil Robertson](#) and [Paul Seymour](#) [50], and again in 2009 for his proof of the Strong Perfect Graph Theorem together with Maria Chudnovsky, [Robertson](#) and [Seymour](#) [19]. In 2011, he was awarded the Karel Janeček Foundation Neuron Prize for Lifetime Achievement in Mathematics and in 2012 he became a Fellow of the American Mathematical Society. He died on 26 March 2020 after a long struggle against amyotrophic lateral sclerosis (ALS).



Biographic note 59: Robin Thomas (1962–2020)

Proof By contradiction. Assume, to the contrary, that G is a critical, disconnected graph. Then G contains $c \geq 2$ components G_1, \dots, G_c , and since $\chi(G) = \max\{\chi(G_1), \dots, \chi(G_c)\}$ it follows that there is at least one component, $G_\ell \subsetneq G$ (say), of G for which $\chi(G_\ell) = \chi(G)$, contradicting the criticality of G . We conclude that there is no disconnected, critical graph. ■

If $\chi(G) = 1$, then G is either trivial or edgeless, as mentioned before. Hence, G is 1-critical if and only if $G \cong K_1$ by [Theorem 14.7](#). If $\chi(G) = 2$, then it follows from [Theorem 14.1](#) that G is bipartite and has at least one edge. Hence it follows from [Theorem 14.7](#) that G is 2-critical if and only if $G \cong K_2$. We leave the characterisation of 3-critical graphs as an exercise.

It is possible to improve upon the result in [\(14.1\)](#), as stated in the following theorem.

Lemma 14.8 *If G is a k -critical graph, then $\chi(G - v) = \chi(G - e) = k - 1$ for every vertex v and every edge e of G .*

Proof Suppose G is a k -critical graph. Then $\chi(G - v) \leq k - 1$ and $\chi(G - e) \leq k - 1$ for every vertex v and every edge e of G , by definition.

Now suppose there exists a $(k - 2)$ -colouring of $G - e$ for some edge $e = uv$ of G . Then u may be recoloured with a new, $(k - 1)$ -th colour to yield a $(k - 1)$ -colouring of G , contradicting the fact that $\chi(G) = k$. We conclude that no $(k - 2)$ -colouring of $G - e$ exists, and hence that $\chi(G - e) \geq k - 1$ for any edge e of G .

Suppose there exists a $(k - 2)$ -colouring of $G - v$ for some vertex v of G . Then v may be assigned a new, $(k - 1)$ -th colour to yield a $(k - 1)$ -colouring of G , again contradicting the fact that $\chi(G) = k$. ■

[Lemma 14.8](#) may now be used to prove the following useful result.

Lemma 14.9 *If G is a k -critical graph with minimum degree δ , then $\delta \geq k - 1$.*

Proof By contradiction. Suppose, to the contrary, that $\delta \leq k - 2$ and let v be a vertex of minimum degree in G . Since G is k -critical, it follows from Lemma 14.8 that $\chi(G - v) = k - 1$. Because $\delta \leq k - 2$, at most $k - 2$ colours would have been used to colour the neighbours of v in any $(k - 1)$ -colouring of $G - v$, leaving at least one colour with which to colour v in order to obtain a $(k - 1)$ -colouring of G . But this contradicts the fact that $\chi(G) = k$. We conclude that $\delta \geq k - 1$. ■

Theorem 14.6 and Lemma 14.9 lead us to an upper bound on the chromatic number of a general graph G in terms of the maximum degree of G .

Theorem 14.10 $\chi(G) \leq \Delta + 1$ for every graph G with maximum degree Δ .

Proof If G is a k -chromatic graph, then G contains a k -critical subgraph H by Theorem 14.6. Furthermore, $\delta(H) \geq k - 1$ by Lemma 14.9 and hence $k \leq 1 + \delta(H) \leq 1 + \Delta(H) \leq 1 + \Delta(G)$. ■

❖ The reader should now be able to attempt Exercises 14.6–14.9.

14.2.3 Other bounds on the chromatic number of a graph

It is possible to establish the value of the chromatic number for certain very special graph classes analytically, as we demonstrate next.

Theorem 14.11

- (i) $\chi(K_n) = n$ for all $n \in \mathbb{N}$.
- (ii) $\chi(C_n) = \begin{cases} 2 & \text{if } n \geq 4 \text{ is even,} \\ 3 & \text{if } n \geq 3 \text{ is odd.} \end{cases}$
- (iii) $\chi(P_n) = 2$ for all $n \geq 2$.

Proof (i) Since every vertex v of the complete graph K_n is adjacent to $n - 1$ other vertices, at least n colours have to be used in any proper colouring of K_n (one colour for v and $n - 1$ colours for its neighbours). This shows that $\chi(K_n) \geq n$. An n -colouring of K_n may, however, be obtained by colouring each vertex of K_n with a different colour, from which we conclude that $\chi(K_n) \leq n$.

(ii) Suppose the vertex set of C_n is $\{v_0, \dots, v_{n-1}\}$, that all edges of C_n are of the form $v_i v_{i+1} \pmod{n}$ for $i = 0, \dots, n - 1$ and consider the case where $n \geq 4$ is an even integer. Since $C_n \not\cong K_1$, no 1-colouring of C_n exists, and so $\chi(C_n) \geq 2$. A 2-colouring of C_n may be obtained by colouring all the odd-indexed vertices of C_n with one colour and all the even-indexed vertices of C_n with another, showing that $\chi(C_n) \leq 2$ if n is even.

Now consider the case where $n \geq 3$ is an odd integer. Any 2-colouring of C_n necessarily has to follow the strategy of colouring an arbitrary vertex v of C_n with colour 1 and then proceed to colour the vertices of C_n at even distance from v with colour 1 and the vertices of C_n at odd distance from v with colour 2. But this results in a vertex w that is adjacent to both a vertex of colour 1 and a vertex of colour 2, and which therefore cannot be coloured, showing that $\chi(C_n) \geq 3$ if n is odd. A 3-colouring of C_n may, however, be obtained by following the 2-colouring strategy described above and then colouring the vertex w with colour 3, showing that $\chi(C_n) \leq 3$ if n is odd.

(iii) This part of the proof is left as an exercise. ■

[Theorem 14.11](#) shows that the general bound in [Theorem 14.10](#) is sharp — the bound is attained by complete graphs and by odd cycles. There is, however, no hope of establishing a formula for the chromatic number of a general graph in view of [Theorem 14.2](#). We therefore turn our attention towards establishing bounds on the chromatic number $\chi(G)$ of a general graph G . Recall, from [Chapter 3](#), that the **clique number** of a graph G , denoted by $\omega(G)$, is the order of the largest clique in G .

Theorem 14.12 $\omega(G) \leq \chi(G) \leq \frac{1}{2}(1 + \sqrt{8m + 1})$ for any graph G of size m .

Proof Since the vertices of a clique in G must all be assigned different colours in any colouring of G , it follows that $\chi(G)$ is at least as large as the order of the largest clique of G .

Suppose $\chi(G) = k$ and consider a k -colouring of G . If there were no edge between two colour classes X and Y of this colouring, then a $(k - 1)$ -colouring of G could be obtained by colouring the vertices in X and Y with the same colour, thus producing a colouring of G in fewer than the minimum number of colours — a contradiction. Hence G has at least one edge between every pair of colour classes, in which case $m \geq \binom{k}{2} = k(k - 1)/2$. Solving this inequality for k , we obtain the desired upper bound on $\chi(G)$. ■

That the bounds in [Theorem 14.12](#) are sharp, may be seen by taking $G \cong K_n$, in which case

$$\omega(G) = \chi(G) = n = \frac{1}{2}(1 + \sqrt{8\binom{n}{2} + 1}) = \frac{1}{2}(1 + \sqrt{8m + 1}).$$

The bounds in [Theorem 14.12](#) are, however, not of much practical value, because the upper bound is not very good for sparse graphs and because the lower bound is hard to compute in view of [Theorem 3.8](#).

We have already seen that the bound in [Theorem 14.10](#) is sharp. The following theorem, however, sheds more light on the result of [Theorem 14.10](#), showing that the upper bound in [Theorem 14.10](#) is attained by complete graphs and odd cycles only, and that it may be lowered by one for all other graphs. The theorem is due to [Brooks \[12\]](#) and dates from 1941.

Rowland Leonard Brooks was born in Lincolnshire, England on 6 February 1916. After studying at Trinity College, Cambridge he worked as a tax inspector. However, at Cambridge he worked with fellow Trinity students [William Tutte](#), Cedric Smith and Arthur Harold Stone on the problem of “Squaring the square” (partitioning rectangles and squares into unequal squares), both under their own names and under the pseudonym *Blanche Descartes*. Today Brooks is best known for proving [Theorem 14.13](#). He died on 18 June 1993.



Biographic note 60: Rowland Brooks (1916–1993)

Theorem 14.13 (Brooks' Theorem) $\chi(G) \leq \Delta(G)$ for any connected graph G which is neither a complete graph nor an odd cycle.

Proof If $\Delta(G) \leq 1$, then G is a complete graph. If $\Delta(G) = 2$, then G is a cycle or a path. If G is an odd cycle, then $\chi(G) = 3$; otherwise, if G is an even cycle or a path, then $\chi(G) \leq 2$ by Theorems 14.11(ii)–(iii). Hence, the desired result holds for a connected graph G with maximum degree $\Delta(G) \leq 2$. Assume, therefore, that G is not a complete graph and that $\Delta(G) \geq 3$. Let v be a vertex of maximum degree in G . Since G is connected and not complete, there exist two nonadjacent vertices u and w such that uv and vw are edges of G .

Let H_1 be the component of $G - \{u, w\}$ that contains v and arrange the vertices of H_1 in nonincreasing order of their distances from v in H_1 . Suppose this gives rise to the sequence $v_3, \dots, v_n (= v)$. Let $v_1 = u$ and $v_2 = w$. Observe that, for $i \in [n-1]$, v_i is adjacent to some v_j for all $j > i$. Assign colour 1 to the vertices v_1 and v_2 ; then successively colour v_3, \dots, v_n each with the first available colour in the list $1, \dots, \Delta$. By the construction of the sequence v_1, \dots, v_n each vertex v_i $i \in [n-1]$ is adjacent to some vertex v_j with $j > i$, and hence is adjacent to at most $\Delta - 1$ vertices v_j satisfying $j < i$. So, when its turn comes to be coloured, v_i is adjacent to at most $\Delta - 1$ vertices already coloured. Therefore, one of the colours $1, \dots, \Delta$ will be available to colour v_i . Finally, since v_n is adjacent to two vertices assigned colour 1, namely v_1 and v_2 , it is adjacent to at most $\Delta - 1$ vertices that have received distinct colours and can be assigned one of the colours $2, \dots, \Delta$ in order to produce a k -colouring of $G[V(H_1) \cup \{u, w\}]$ with $k \leq \Delta(G)$.

If vertices of G remain that have not yet been coloured, then $G - \{u, w\}$ is disconnected. Let H_2 be the union of the components of $G - \{u, w\}$ that do not contain v . Then $H_3 = G[V(H_2) \cup \{u, v, w\}]$ is connected. Order the vertices of H_2 in nonincreasing order of their distances from v in H_3 . Let w_1, \dots, w_m be the resulting sequence, and let $w_{m+1} = u$ and $w_{m+2} = w$. For each w_i ($i \in [m]$) there is a w_j with $j > i$ so that w_i is adjacent to w_j . As before, a k' -colouring of H_2 with $k' \leq \Delta$ may be obtained by successively colouring w_1, \dots, w_m , each with the first available colour in the list $1, \dots, \Delta$.

In order to obtain a colouring of G using at most Δ colours, some vertices of G may have to be recoloured. Suppose that some colour i is present in the neighbourhood of u in $G[V(H_2) \cup \{u, w\}]$ as well as in the neighbourhood of w in $G[V(H_2) \cup \{u, w\}]$. Suppose colour j is assigned to v . If $i \neq j$, then interchange the colours of the vertices that have been coloured with colours i and j in H_1 . Now both u and w are adjacent with at most $\Delta - 1$ colours in their combined neighbourhoods. Hence u and w can each be recoloured with a colour that does not yet appear in their combined neighbourhoods in order to produce a colouring of G using at most Δ colours.

Suppose now that there is no single colour assigned to vertices in the neighbourhoods of both u and w in $G[V(H_2) \cup \{u, w\}]$. Let \mathcal{C}_1 and \mathcal{C}_2 be the sets of colours assigned to the vertices of H_2 that are adjacent to u and w , respectively (i.e. $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$). If $1 \notin \mathcal{C}_1 \cup \mathcal{C}_2$, then u and w may retain colour 1 and a colouring of G using at most Δ colours has been found. Thus, assume, without loss of generality, that $1 \in \mathcal{C}_1$. Let \mathcal{D}_1 and \mathcal{D}_2 be the sets of colours assigned to the vertices of H_1 that are adjacent to u and w , respectively. From the k -colouring of H_1 it follows that $1 \notin \mathcal{D}_1$. If $|\mathcal{C}_1 \cup \mathcal{D}_1| < \Delta$, retain colour 1 for w and recolour u with one of the colours $1, \dots, \Delta$ to obtain a colouring of G using at most Δ colours.

Assume, therefore, that $|\mathcal{C}_1 \cup \mathcal{D}_1| = \Delta$. If $|\mathcal{C}_1 \cup \mathcal{C}_2| < \Delta$, let $i \in [\Delta] \setminus \{\mathcal{C}_1 \cup \mathcal{C}_2\}$. Then $i \in \mathcal{D}_1$. Interchange the colours of the vertices coloured 1 and i in

$G[V(H_1) \cup \{u, w\}]$ to obtain a colouring of G using at most Δ colours. Hence, assume that $|\mathcal{C}_1 \cup \mathcal{C}_2| = \Delta$. Since $d(u) = \Delta$, it follows that $|\mathcal{C}_2| = |\mathcal{D}_1|$. If $|\mathcal{C}_1| > 1$, let $i \in \mathcal{C}_1 \setminus \{1\}$ and interchange the colours of the vertices coloured i and j in H_1 . Retain colour 1 for w and recolour u with one of the colours $1, \dots, \Delta$ not used in its neighbourhood. Finally, suppose that $|\mathcal{C}_1| = 1$. Then $|\mathcal{C}_2| = |\mathcal{D}_1| = \Delta - 1$, and v is the only neighbour of w in H_1 . Let $i \in \mathcal{D}_1 \setminus \{j\}$. It follows that $i \neq 1$. Interchange the colours of the vertices coloured 1 and i in H_1 . Then v retains colour j ($\neq 1$). Since at least two vertices adjacent to u are now coloured with colour 1, it follows that u may be coloured with one of the colours $2, \dots, \Delta$ to produce a colouring of G using at most Δ colours. ■

The statement of [Theorem 14.13](#) suggests that complete graphs and odd cycles play an important role in determining whether or not a graph has a large chromatic number. It is obvious that if a graph G has a subgraph isomorphic to K_k , then $\chi(G) \geq k$. One might, therefore, be led to believe that in order to have a large chromatic number, a graph must contain a reasonably large clique. The following theorem, however, shows that this is not true and, in fact, that there exist triangle-free graphs (graphs containing no 3-cycles and hence no cliques of order $k \geq 3$) with arbitrarily large chromatic numbers. This fact has been established by several mathematicians, including [Descartes](#) [21] in 1947, [Zykov](#) [60] in 1949, and [Kelly and Kelly](#) [38] in 1954. The proof we present here is due to [Mycielski](#) [43] and dates from 1955.

Theorem 14.14 *For every positive integer k there is a triangle-free graph G for which $\chi(G) = k$.*

Proof By induction over k . For $k \in [3]$ the graphs K_1 , K_2 and C_5 satisfy the statement of the theorem, respectively. Assume, as induction hypothesis, that there exists a triangle-free graph G_{k-1} with vertex set $\{v_1, \dots, v_n\}$ for which $\chi(G_{k-1}) = k-1$. We use this graph to construct a triangle-free graph G_k for which $\chi(G_k) = k$, as follows. Add n new vertices $\mathcal{U} = \{u_1, \dots, u_n\}$ to the graph G_{k-1} and join u_i to the neighbours of v_i in G_{k-1} . Finally add another new vertex v to the resulting graph and join it to all the vertices in \mathcal{U} so as to obtain G_k . Then the graph G_k has order $2n+1$.

If G_k were to contain a triangle, such a triangle would have to contain exactly one vertex, u_i (say), from \mathcal{U} and two adjacent vertices, v_j and v_ℓ (say), from G_{k-1} since no two vertices in \mathcal{U} are adjacent and since G_{k-1} is triangle-free. But then $G_{k-1}[v_i, v_j, v_\ell]$ is a triangle, a contradiction. We conclude that G_k is indeed triangle-free.

It remains to be shown that $\chi(G_k) = k$. Any $(k-1)$ -colouring of G_{k-1} may be extended to a k -colouring of G_k by assigning to u_i the same colour assigned to v_i and by assigning the k -th colour to v . This shows that $\chi(G_k) \leq k$. Now suppose there exists a $(k-1)$ -colouring of G_k using the colours $1, \dots, k-1$. Recolouring if necessary, we may assume that v is coloured with colour $k-1$. Then no vertex $u_i \in \mathcal{U}$ is coloured with this colour. However, since $\chi(G_{k-1}) = k-1$ and $G_{k-1} \subset G_k$, the colour $k-1$ must be assigned to some vertices of G_{k-1} . Recolour each vertex v_i of G_{k-1} that is coloured with colour $k-1$ using the colour assigned to u_i . Since u_i is adjacent to every neighbour of v_i in G_{k-1} and since no two of the recoloured vertices are adjacent, this produces a $(k-2)$ -colouring of G_{k-1} , a contradiction, and so we conclude that $\chi(G_k) = k$. ■

Recall, from [Chapter 12](#), that the **girth** of a graph is the length of a shortest cycle in the graph (taken as infinite if the graph is acyclic). The result of [Theorem 14.14](#) was improved significantly by [Erdős and Wilson \[22\]](#) in 1977, who showed that there exist graphs with arbitrarily large girth *and* arbitrarily large chromatic numbers. The proof of this theorem by [Erdős](#) is, however, probabilistic in nature and is hence deferred to [Chapter 20](#), where we consider the topic of probabilistic arguments.

Theorem 14.15 (Erdős) *For any positive integers k and ℓ there is a graph G with girth $g(G) > \ell$ for which $\chi(G) > k$.*

We close this section with the landmark 1956 result by [Nordhaus and Gaddum \[45\]](#) on the chromatic number of a graph and its complement. We require as intermediate result the following lemma due to [Welsh and Powell \[56\]](#).

Lemma 14.16 *For any graph G ,*

$$\chi(G) \leq 1 + \max_H \{\delta(H)\},$$

where the maximum is taken over all induced subgraphs H of G .

Proof Let $k = \max_H \{\delta(H)\}$, where the maximum is taken over all induced subgraphs H of G . Then clearly $\delta(G) \leq k$.

We define an ordering of the vertices of G as follows: Let v_n be a vertex of minimum degree in G (therefore $d_G(v_n) \leq k$). Let $G_{n-1} = G - v_n$. Since G_{n-1} is an induced subgraph of G , it follows that $\delta(G_{n-1}) \leq k$. Let v_{n-1} be a vertex of minimum degree in G_{n-1} (therefore $d_{G_{n-1}}(v_{n-1}) \leq k$). Let $G_{n-2} = G_{n-1} - v_{n-1}$, and continue this process until all the vertices v_1, \dots, v_n of G have been ordered. Then G_i is the graph induced by the vertices $\{v_1, \dots, v_i\}$ (here $G_n = G$) and v_i has degree at most k in G_i , for all $i \in [n]$.

If we now colour the vertices of G in the order v_1, \dots, v_n , using the first available colour not already used to colour any lower-indexed neighbours of v_i when colouring v_i , we require at most $k+1$ colours. ■

The proof of the result by [Nordhaus and Gaddum](#) presented here was suggested by Kronk to [Chartrand and Mitchem \[17\]](#).

Theorem 14.17 (Nordhaus and Gaddum) *For every graph G of order n ,*

- (i) $n \leq \chi(G) \cdot \chi(\bar{G}) \leq (n+1)^2/4$, and
- (ii) $2\sqrt{n} \leq \chi(G) + \chi(\bar{G}) \leq n+1$.

Proof We first prove the lower bound in (i). Consider a $\chi(G)$ -colouring of G and a $\chi(\bar{G})$ -colouring of \bar{G} . Using these colourings, a colouring of $K_n = G \oplus \bar{G}$ may be obtained by assigning as colour to the vertex v of K_n , the ordered pair (c, d) , where c is the colour assigned to v in G and d is the colour assigned to v in \bar{G} . Since every two vertices of K_n are adjacent in either G or \bar{G} , they are assigned different colours in at least one of G or \bar{G} . Thus, the resulting assignment of colours to vertices of K_n is a proper colouring of K_n using at most $\chi(G) \cdot \chi(\bar{G})$ colours, from which we deduce that $n = \chi(K_n) \leq \chi(G) \cdot \chi(\bar{G})$, the desired lower bound.

Next we establish the upper bound in (ii). Let $k = \max_H \{\delta(H)\}$, where the maximum is taken over all induced subgraphs H of G . We show, by contradiction, that

$$\max_F \{\delta(F)\} \leq n - k - 1, \tag{14.2}$$

where the maximum is taken over all induced subgraphs F of \bar{G} . Assume, to the contrary, that there is an induced subgraph F of \bar{G} for which $\delta(F) \geq n - k$. Then each vertex of F has degree at most $n - 1 - \delta(F) \leq k - 1$ in G . Let H be an induced subgraph of G for which $\delta(H) = k$. Then $V(F) \cap V(H) = \emptyset$ and so $|V(F)| \leq n - |V(H)| \leq n - (k + 1)$, implying that $\delta(F) \leq n - k - 2$, which is a contradiction, thereby establishing (14.2). It therefore follows from Lemma 14.16 that $\chi(G) \leq k + 1$ and $\chi(\bar{G}) \leq n - k$. Hence $\chi(G) + \chi(\bar{G}) \leq (k + 1) + (n - k) = n + 1$, the desired upper bound.

Since the arithmetic mean of two positive numbers is always at least as large as their geometric mean, it follows that

$$\sqrt{n} \leq \sqrt{\chi(G) \cdot \chi(\bar{G})} \leq \frac{\chi(G) + \chi(\bar{G})}{2},$$

which establishes the desired lower bound in (ii). The upper bound in (i) follows similarly readily from the inequality relating the geometric and arithmetic means of two positive numbers. ■

❖ The reader should now be able to attempt Exercises 14.10–14.13.

14.2.4 Computing good vertex colourings heuristically

Let us now turn our attention to algorithmic procedures by which good colourings of graphs may be obtained efficiently. Perhaps the most elementary vertex colouring heuristic, the **sequential colouring heuristic**, is a greedy algorithm which takes a labelling of the vertex set of a graph G as input and assigns the first available colour that has not yet been assigned to any of its neighbours to each vertex in turn, in the order in which the vertices of G have been labelled. The **sequential colouring heuristic** is presented in pseudocode form as Algorithm 31.

Algorithm 31: Sequential colouring heuristic

Input : A graph G of order n with vertex set $\{v_1, \dots, v_n\}$.

Output : A proper (vertex) colouring of G .

```

1 for  $i = 1$  to  $n$  do
2   Assign the smallest colour to  $v_i$  that has not yet been assigned to any
     of its neighbours
3 print the resulting colouring of  $G$ 
```

As an illustration of the **sequential colouring heuristic**, suppose the vertices of the well-known Grötzsch graph are labelled as in Figure 14.4(a). The 4-colouring of this graph shown in the same figure was obtained by the **sequential colouring algorithm**.

Because of its greedy nature, the **sequential colouring heuristic** does not necessarily produce a minimum colouring (*i.e.* a χ -colouring) for any graph G . The number of colours used by the **sequential colouring heuristic** depends in a sensitive manner on the sequence in which the vertices are labelled — especially in large graphs, as illustrated in [55, p. 86], but will always use at most $\Delta + 1$ colours, as stated in the next result (whose proof we leave as an exercise).

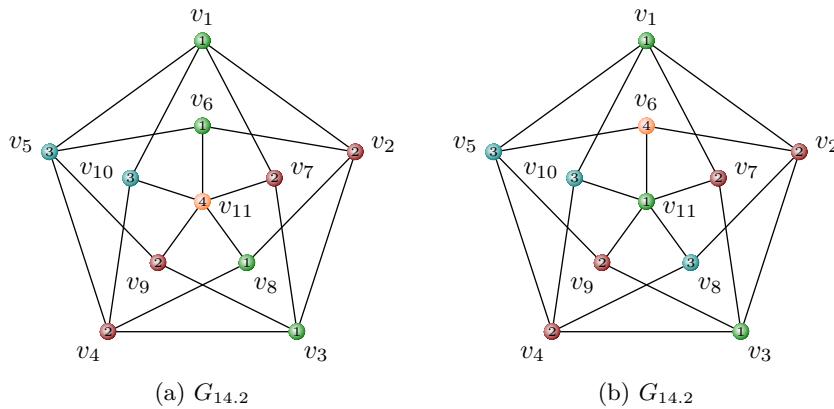


Figure 14.4: 4-Colourings of the Grötzsch graph obtained via (a) the sequential colouring heuristic and (b) the largest-first or Welsh-Powell colouring heuristic.

Theorem 14.18 Let G be an input graph of order n and size m .

- (i) Algorithm 31 returns a proper colouring of G using at most $\Delta(G)+1$ colours.
- (ii) The worst-case time complexity of Algorithm 31 is $\mathcal{O}(mn)$.

Generally speaking, if a vertex has large degree and many of its neighbours have already been coloured during an application of the [sequential colouring heuristic](#), then it is more likely that the vertex will have to be coloured with a previously unused colour, thus perhaps resulting in the use of an unnecessarily large number of colours. To remedy this situation, the vertices are labelled in nonincreasing order of their degrees in the [Welsh-Powell colouring heuristic](#) [56], also known as the [largest-first heuristic](#), which then proceeds with the colouring process in exactly the same manner as in the [sequential colouring heuristic](#). The process is formalised in pseudocode form as [Algorithm 32](#).

Algorithm 32: Welsh-Powell colouring heuristic

Input : A graph G of order n .

Output : A proper (vertex) colouring of G .

- 1 Label the vertices of $G \{v_1, \dots, v_n\}$ in nonincreasing order of their degrees
 - 2 **for** $i = 1$ **to** n **do**
 - 3 | Assign the smallest colour to v_i that has not yet been assigned to any of its neighbours
 - 4 **print** the resulting colouring of G
-

In the Grötzsch graph in [Figure 14.4\(b\)](#), vertex v_{11} is the only vertex of degree 5 and is thus coloured first by the [Welsh-Powell colouring heuristic](#). Next, the vertices v_1, \dots, v_5 , all of degree 4, may be coloured in any sequence. Finally, the vertices v_6, \dots, v_{10} may be coloured in any sequence, since they all have degree 3. A 4-colouring of the Grötzsch graph obtained by the [Welsh-Powell colouring heuristic](#) with the vertex sequence $v_{11}, v_1, \dots, v_{10}$, is shown in [Figure 14.4\(b\)](#).

The number of colours used in the [Welsh-Powell colouring heuristic](#) is related to Welsh and Powell's upper bound on the chromatic number of a graph in [Lemma 14.17](#).

ma 14.16. This colouring procedure also does not always yield a minimum colouring, since different sequences of nonincreasing degree may still exist (for example, in regular graphs) but will also always use at most $\Delta + 1$ colours.

The Welsh-Powell colouring heuristic may further be refined, based on the intuition that if two vertices have equal degree, then the one having the more densely coloured neighbourhood will be harder to colour later during the colouring procedure. Let the **colour degree** of a vertex v of a graph G be the number of different colours already assigned to vertices in $N_G(v)$ in a partial colouring of G (*i.e.* during the course of a colouring procedure). Then an adapted version of the Welsh-Powell colouring heuristic, called **Brélaz's colouring heuristic**, is obtained if, among the uncoloured vertices with maximum colour degree, the vertex with largest degree in the uncoloured subgraph is chosen to be coloured next. Therefore, in Brélaz's heuristic the order in which the vertices should be coloured is not determined beforehand, but is determined as the algorithm proceeds.

Algorithm 33: Brélaz's colouring heuristic

Input : A graph G of order n .

Output : A proper (vertex) colouring of G .

- 1 Label the vertices of G v_1, \dots, v_n in nonincreasing order of their degrees
 - 2 Colour a vertex of minimum degree with colour 1
 - 3 Select a vertex of maximum colour degree and colour it with the smallest available colour (in the case of a tie, select any vertex of maximum degree in the uncoloured subgraph)
 - 4 **if all vertices are coloured then stop**
 - 5 **else go to Step 3**
 - 6 **print** the resulting colouring of G
-

Daniel Brélaz was born in Lausanne, Switzerland on 4 January 1950. He received a degree in mathematics from École Polytechnique Fédérale de Lausanne in 1975, and afterwards taught mathematics. Later he became a Swiss politician and member of the Green Party of Switzerland. In 1975, he joined the Group for the Protection of the Environment in Lausanne and in 1978 he was one of the first environmentalists elected to the National Council of Switzerland (the parliament of Switzerland) in the Grand Council of Vaud. He was re-elected in 1982–1983.



Biographic note 61: Daniel Brélaz (1950–present)

Applying Brélaz's colouring heuristic, given in pseudocode form as Algorithm 33, to the Grötzsch graph in Figure 14.4, vertex v_{11} is still coloured first as by the Welsh-Powell colouring heuristic. The vertices v_6, \dots, v_{10} all have colour degree 1 and a degree of 2 in the uncoloured subgraph, $G_{14.2} - v_{11}$, shown in

Figure 14.5(a). Any one of these vertices may therefore be coloured next. Suppose v_6 is chosen to be coloured next. Then $v_2, v_5, v_7, v_8, v_9, v_{10}$ all have maximum colour degree 1, and v_2 and v_5 have a degree of 3 in the uncoloured subgraph, $G_{14.2} - \{v_6, v_{11}\}$, shown in **Figure 14.5(b)**, while v_7, v_8, v_9, v_{10} have a degree of 2. From v_2 and v_5 , v_2 is chosen to be coloured next. At this point v_8 is the only uncoloured vertex with maximum colour degree and is coloured next. All uncoloured vertices now have the same colour degree (namely 1). In the uncoloured subgraph, $G_{14.2} - \{v_2, v_6, v_8, v_{11}\}$ shown in **Figure 14.5(c)**, vertices v_1, v_3, v_4 and v_5 all have a degree of 3. Suppose v_1 is chosen to be coloured next. Continuing in this manner v_5, v_4, v_{10} and v_3 are coloured in this order. Finally, v_7 and v_9 may be coloured in any order. This leads to the 4-colouring of the Grötzsch graph in **Figure 14.5(d)**.

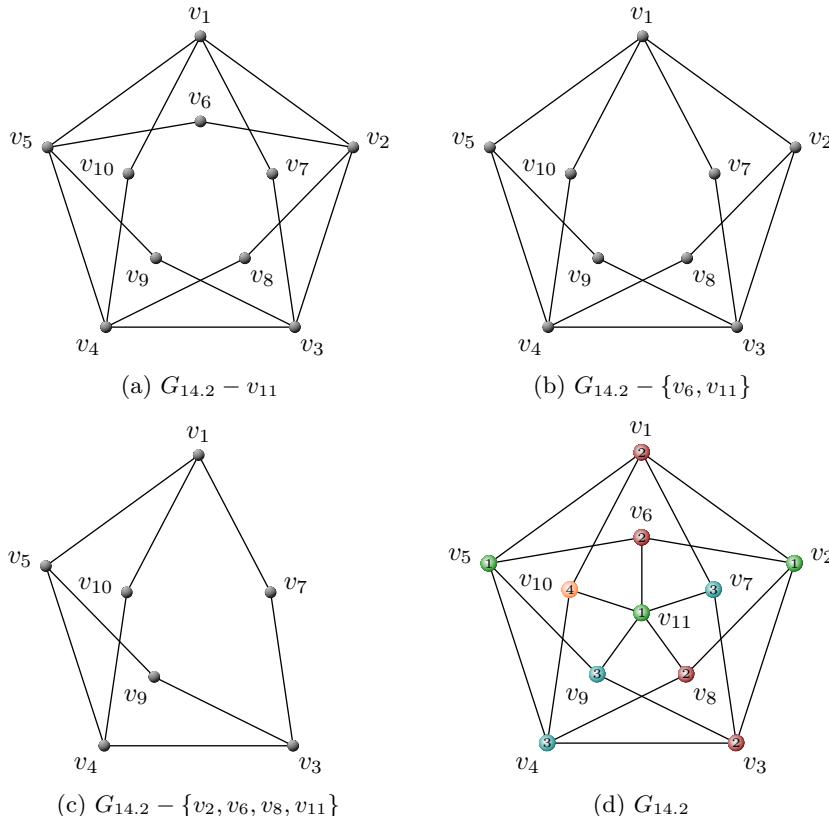


Figure 14.5: Subgraphs and final colouring produced during the course of Brélaz's colouring heuristic when applied to the Grötzsch graph.

A reverse strategy of the same basic idea that a vertex with large degree is more likely to force one to use a previously unused colour, gives rise to the so-called **smallest-last colouring heuristic**, formalised in pseudocode form as [Algorithm 34](#). In this heuristic, due to Matula *et al.* [42], the ordering of the vertices is determined in reverse by repeatedly selecting one of the vertices of minimum degree, placing it just before the previously placed vertex in the order in which the vertices should be coloured and deleting the selected vertex from the graph. In

the remaining subgraph, a vertex of minimum degree is again selected, placed just before the previously ordered vertex, and deleted from the graph. This process continues until all vertices have been positioned in a colouring order. Thereafter, the colouring proceeds as in the [sequential colouring heuristic](#).

Algorithm 34: Smallest-last colouring heuristic

Input : A graph G of order n .

Output : A proper (vertex) colouring of G .

- 1 Select a vertex of minimum degree in G and label it v_n
 - 2 $\mathcal{S} \leftarrow \{v_n\}$, $i \leftarrow n - 1$
 - 3 **if** $|\mathcal{S}| = n$ **then**
 - 4 Colour the vertices $\{v_1, \dots, v_n\}$ of G by means of the sequential colouring heuristic ([Algorithm 31](#))
 - 5 **stop**
 - 6 Select a vertex of minimum degree in $G - \mathcal{S}$ and label it v_i
 - 7 $\mathcal{S} \leftarrow \mathcal{S} \cup \{v_i\}$, $i \leftarrow i - 1$
 - 8 **go to Step 3**
-

In the Grötzsch graph in [Figure 14.4](#), the vertices v_6, \dots, v_{10} all have a minimum degree of 3 and so any one may be selected first. Adopting the convention of choosing the smallest subscripted vertex in case of a tie, the vertex v_6 is selected to be coloured last. In the graph $G_{14.2} - v_6$, the vertices $v_2, v_5, v_7, \dots, v_{10}$ now all have a minimum degree of 3. Choose v_2 next. After deleting v_2 from $G_{14.2} - v_6$, the resulting graph has only one vertex of degree 2, namely v_8 , and so v_8 should be coloured just before v_2 . For clarity, the current subgraph, $G_{14.2} - \{v_2, v_6, v_8\}$ is shown in [Figure 14.6\(a\)](#). As may be seen in [Figure 14.6\(a\)](#), the graph $G_{14.2} - \{v_2, v_6, v_8\}$ is 3-regular. Thus, v_1 is chosen to be inserted in the reverse colouring order next. In $G_{14.2} - \{v_1, v_2, v_6, v_8\}$ the vertices v_7 and v_{10} both have minimum degree 1. The vertex v_7 is therefore chosen next. Continuing in this manner, v_{10} and then v_{11} are inserted in the reverse colouring order. Finally, in $G_{14.2} - \{v_1, v_2, v_6, v_7, v_8, v_{10}, v_{11}\}$ all remaining vertices have minimum degree 2. Thus v_3 is chosen next, resulting in the final selection of v_4, v_5 and v_9 in that order. Therefore the colouring sequence for the [smallest-last colouring heuristic](#) is $v_9, v_5, v_4, v_3, v_{11}, v_{10}, v_7, v_1, v_8, v_2, v_6$ and the resulting 4-colouring of the Grötzsch graph is shown in [Figure 14.6\(b\)](#).

The [smallest-last colouring heuristic](#) generally achieves slightly better results than the [largest-first](#) or [Welsh-Powell colouring heuristic](#), as demonstrated by Brélaz [7], while, in general, Brélaz's heuristic outperforms both of these heuristics [28, pp 231, 233]. Furthermore, all four of the above heuristics use a technique called **successive augmentation**, where the vertices are coloured in sequence, one at a time, without any attempt to improve the colouring by means of perturbation, until all vertices have been coloured and the final colouring is obtained. Any of the above heuristics may, however, be improved (as Brélaz's [7] testing showed) by incorporating a colour interchange approach. According to this approach, whenever a new colour has to be used to colour a vertex v , all pairs i and j of colours already used, are considered. If it so happens that in the graph induced by vertices coloured with colours i and j , the vertex v is adjacent to vertices coloured with

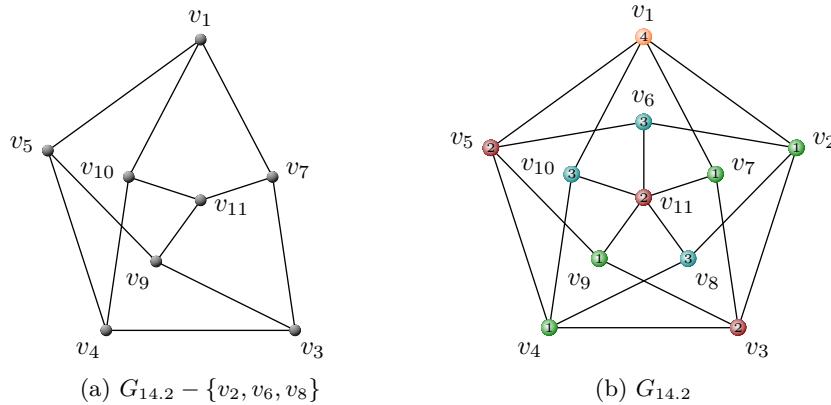


Figure 14.6: Colouring of the Grötzsch graph via the [smallest-last colouring algorithm](#).

one of the colours, then the colours in some of the bipartite components are interchanged so that v is adjacent to vertices of colour i only. Then v can be assigned colour j and a new colour is no longer required to colour v .

- ❖ The reader should now be able to attempt Exercises 14.14–14.18 and Projects 14.3–14.5.

14.2.5 An exact algorithm for vertex colouring

Although heuristic colouring methods, such as the four procedures outlined above, are often capable of producing a χ -colouring of a graph G , this is, of course, not guaranteed for every graph. In order to appreciate the computational complexity of computing the chromatic number of a graph G of order n exactly, it is worth pondering the computational expense of a natural (albeit naive) brute-force approach whereby every possible k -partition of the vertex set of G is tested to determine whether it is a k -colouring of G . If no k -colouring of G can be found in this way, then G is not k -colourable and the bound $\chi(G) > k$ has been established. Otherwise the bound $\chi(G) \leq k$ has been established. If this procedure is repeated for $k = 3, 4, 5, \dots$ until a proper k -colouring of G is obtained for the first time, then clearly $\chi(G) = k$. The worst-case time complexity of this naive procedure may be estimated by noting that the number of k -partitions of the vertex set of G into k subsets of cardinalities n_1, \dots, n_k is the well-known multinomial coefficient

$$\binom{n}{n_1, \dots, n_k} = \frac{n!}{n_1! \cdots n_k!}$$

where, of course, $n_1 + \cdots + n_k = n$. Therefore, the total number of k -partitions of the vertex set of G is

$$\sum_{\substack{(n_1, \dots, n_k): \\ n_1 + \cdots + n_k = n}} \binom{n}{n_1, \dots, n_k} = \sum_{\substack{(n_1, \dots, n_k): \\ n_1 + \cdots + n_k = n}} \binom{n}{n_1, \dots, n_k} 1^{n_1} 1^{n_2} \cdots 1^{n_k} \quad (14.3)$$

$$= \underbrace{(1 + 1 + \cdots + 1)}_{k \text{ terms}}^n = k^n \quad (14.4)$$

by the multinomial theorem [51, p. 11]. It is concluded that the worst-case time complexity of the above naive exact colouring procedure for a graph G of order n is $\mathcal{O}(\chi(G)^n) = \mathcal{O}(\Delta^n)$, which is an exponential function of the order of G .

Of course, exact algorithms used in practice are not as naive in their approach as the exact procedure described above. One of the first exact graph colouring algorithms, due to Brown [13] in 1972, is based on the idea of avoiding redundancy and is essentially a tree-search procedure employing a backtracking technique. Brown's idea of avoiding redundancy states that it is useless to consider colourings that may be obtained from one another simply by interchanging some of the colours. Another exact algorithm for graph colouring is that of Herrmann and Hertz [32], dating from 2002, which employs the notion of graph criticality.

We describe and illustrate the working of [Brown's Algorithm](#) in the remainder of this section. Suppose we wish to find the chromatic number of a graph G of order n with vertex set $\{v_1, \dots, v_n\}$. We assume that the vertices of G are labelled in the order in which they would be coloured according to [Brélaz's colouring heuristic](#) ([Algorithm 33](#)). [Brown's Algorithm](#), in fact, starts by applying [Brélaz's colouring heuristic](#) to G in order to determine an upper bound q on $\chi(G)$. The actual vertex colouring returned by the heuristic is, however, not utilised in [Brown's Algorithm](#); it is only the number of colours employed by the heuristic that is of importance.

The algorithm attempts to find a vertex colouring of G using $q - 1$ colours. If this is possible, the value of the upper bound q is lowered by one, and the process is repeated until no vertex colouring of G using one fewer colour can be found, at which point the current upper bound is returned as the value of the chromatic number of G . Each attempt at colouring the vertices of G with one fewer colour is essentially a back-tracking depth-first search. The vertices of G are coloured in the order in which they are labelled, assigning the smallest available colour i from a list containing one fewer colour than the current upper bound to each vertex until a vertex is reached for which no colour is available, in which case backtracking occurs to the last visited vertex for which more than one colour was available. This vertex is then recoloured with the second smallest available colour (a colour larger than i), after which the depth-first search colouring strategy proceeds again, perhaps until another backtracking is required. This process is repeated until all vertices have been coloured or until it has been ascertained that no vertex colouring of G exists utilising one colour fewer than the current upper bound.

[Brown's Algorithm](#) is given in pseudocode form as [Algorithm 35](#). The algorithm determines the initial upper bounding value q on $\chi(G)$ in [Step 1](#) and colours the vertex v_1 with colour 1 in [Step 2](#). It then keeps track of a number of parameters as it progresses with colouring the other vertices of G . One of the most important of these parameters is a parameter ℓ which captures the number of colours already used in a vertex colouring of G . The value of ℓ , when colouring the vertex v_k , is recorded as L_k and this value is updated when necessary. The parameters ℓ and L_k are both initialised as 1 when $k = 1$. These initialisations all take place in [Step 3](#) of [Algorithm 35](#).

A set \mathcal{U}_k of colours available for use when colouring the vertex v_k is also maintained and updated when necessary, taking into account colours already assigned to the neighbours of v_k . Define \mathbb{N}_p as the set of the first p natural numbers, *i.e.* $\mathbb{N}_p = [p]$. Then \mathcal{U}_k is the set \mathbb{N}_{q-1} from which the colours already assigned to the neighbours of v_k (with indices smaller than k) have been removed. The set \mathcal{U}_1 is

Algorithm 35: Brown's algorithm

Input : A graph G with vertex set $\{v_1, \dots, v_n\}$
Output : The value of $\chi(G)$ together with a χ -colouring of G

```

1  $q \leftarrow$  number of colours used by Brélaz' heuristic when colouring  $G$ 
2 colour vertex  $v_1$  with colour 1
3  $\mathcal{U}_1 \leftarrow \emptyset$ ,  $L_1 \leftarrow 1$ ,  $\ell \leftarrow 1$ ,  $k \leftarrow 2$ 
4  $\mathcal{U}_k \leftarrow$  the elements of  $\mathbb{N}_{q-1}$  not used to colour the neighbours of  $v_k$ 
5 if  $\mathcal{U}_k = \emptyset$  then
6   if  $k = 1$  then return  $\chi(G) = q$  and the vertex colouring; stop
7   else  $k \leftarrow k - 1$ ,  $\ell \leftarrow L_k$ , go to Step 5
8 else
9    $i \leftarrow$  smallest colour in  $\mathcal{U}_k$ 
10   $\mathcal{U}_k \leftarrow \mathcal{U}_k \setminus \{i\}$ 
11  colour vertex  $v_k$  with colour  $i$ 
12 else
13   if  $i > \ell$  then  $\ell \leftarrow \ell + 1$ 
14   if  $k = n$  then
15     store the vertex colouring
16      $q \leftarrow \ell$ ,  $j \leftarrow$  smallest vertex label coloured with colour  $q$ 
17      $k \leftarrow j - 1$ ,  $\ell \leftarrow q - 1$ , go to Step 4
18   else  $L_k \leftarrow \ell$ ,  $k \leftarrow k + 1$ , go to Step 4

```

initialised in [Step 3](#) as the empty set, because colouring the vertex with colour 1 is without loss of generality.

Whereas Steps 1–3 are only executed once, Steps 4–18 are repeated each time the vertex k is coloured, for $k = 2, \dots, n$. If $k = n$ in [Step 14](#), then a vertex colouring of G using $\ell < q$ colours has been achieved, at which point the colouring is stored in [Step 15](#) for potential future reference, upon which the value of q is lowered to ℓ in [Step 16](#) and a colouring in $\ell - 1$ is attempted in [Step 17](#). Backtracking occurs if $\mathcal{U}_k = \emptyset$ in [Step 5](#) in order to avoid that the value of ℓ exceeds $q - 1$. The stopping condition of the algorithm occurs in [Step 6](#) when backtracking is attempted, but all colouring alternatives have been exhausted for smaller indexed vertices. Otherwise, the value of ℓ is reset to L_k and an alternative colouring of v_{k-1} is attempted in [Step 7](#).

The actual colouring of v_k takes place in Steps 9–11, where the smallest colour i in the set \mathcal{U}_k of available colours is assigned to v_k and the set \mathcal{U}_k is updated in case later backtracking is required so as to avoid duplicating colourings. The value of ℓ is increased if necessary in [Step 13](#) and stored in [Step 18](#). The value of k is also incremented in [Step 18](#), after which the loop spanning Steps 4–18 is repeated.

Let us illustrate the working of [Algorithm 35](#) by means of an example. Consider the graph $G_{14.3}$ of order 10 in [Figure 14.7\(a\)](#) whose vertices have been numbered in the order in which they would be coloured according to Brélaz's colouring heuristic. Applying Brélaz's colouring heuristic to $G_{14.3}$, the colouring in [Figure 14.7\(b\)](#) results, utilising four colours. Therefore $q = 4$ is an upper bound on $\chi(G_{14.3})$. A vertex colouring of $G_{14.3}$ utilising only three colours is therefore initially attempted by Brown's Algorithm. After performing the initialisations in Steps 1–3 of [Algo-](#)

rithm 35, the parameter values in the first row of Table 14.3 results. The column headed “Iteration” in the table contains the number of times that the loop spanning Steps 4–18 has been executed.

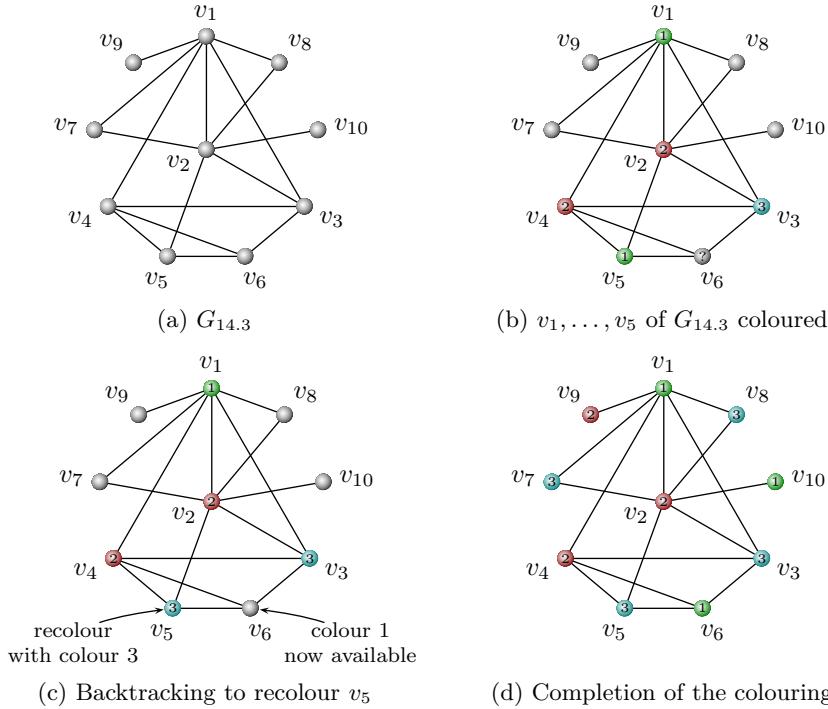


Figure 14.7: Application of Algorithm 35 to determine the chromatic number $\chi(G_{14.3})$ of the graph $G_{14.3}$ in (a). The colouring process occurs in a depth-first manner until it is found in (b) that v_6 cannot be coloured without using a fourth colour. Backtracking then occurs, recolouring v_5 with colour 3 in (c), after which the colouring process resumes in a depth-first manner until all vertices are coloured in (d), using only three colours.

When entering the loop spanning Steps 4–18 for the first time ($k = 2$), the list of colours available for colouring v_2 is determined to be $\mathcal{U}_2 = \{2, 3\}$. Of these colours, colour 2 is assigned to v_2 , and the list of potentially available alternative colours for v_2 is updated to $\mathcal{U}_2 = \{3\}$. The parameter updates $\ell \leftarrow 2$ (in Step 13), as well as $L_2 \leftarrow 2$ and $k \leftarrow 3$ (in Step 18), are performed.

Upon entering the loop spanning Steps 4–18 for the second time ($k = 3$), the list of colours available for colouring v_3 is found to be $\mathcal{U}_3 = \{3\}$. Hence colour 3 is assigned to v_3 , and the list of potentially available alternative colours for v_3 is updated as $\mathcal{U}_3 \leftarrow \emptyset$ (this value may later be updated if backtracking is required). The parameter updates $\ell \leftarrow 3$ (in Step 13), as well as $L_3 \leftarrow 3$ and $k \leftarrow 4$ (in Step 18), are performed.

When we enter the loop spanning Steps 4–18 for the third time ($k = 4$), the list of colours available for colouring v_4 is found to be $\mathcal{U}_4 = \{2\}$. Hence colour 2 is assigned to v_4 , and the list of potentially available alternative colours for v_4

Iteration	k	\mathcal{U}_k	i	ℓ	L_k	q	j	Action
0	1	\emptyset	—	1	1	4	—	Colour v_1 with colour 1
1	2	$\{2, 3\} \rightarrow \{3\}$	2	2	2	4	—	Colour v_2 with colour 2
2	3	$\{3\} \rightarrow \emptyset$	3	3	3	4	—	Colour v_3 with colour 3
3	4	$\{2\} \rightarrow \emptyset$	2	3	3	4	—	Colour v_4 with colour 2
4	5	$\{1, 3\} \rightarrow \{3\}$	1	3	3	4	—	Colour v_5 with colour 1
5	6	\emptyset	v_6 cannot be coloured, so backtrack to recolour v_5					
	5	$\{3\} \rightarrow \emptyset$	3	3	3	4	—	Colour v_5 with colour 3
6	6	$\{1\} \rightarrow \emptyset$	1	3	3	4	—	Colour v_6 with colour 1
7	7	$\{3\} \rightarrow \emptyset$	3	3	3	4	—	Colour v_7 with colour 3
8	8	$\{3\} \rightarrow \emptyset$	3	3	3	4	—	Colour v_8 with colour 3
9	9	$\{2, 3\} \rightarrow \{3\}$	2	3	3	4	—	Colour v_9 with colour 2
10	10	$\{1, 3\} \rightarrow \{3\}$	3	3	3	4	—	Colour v_{10} with colour 1
	2	$\{3\} \rightarrow \emptyset$	3	—	2	3	3	—
	1	\emptyset	... stop and return $q = 3$ as chromatic number					

Table 14.3: Algorithmic progression when applying [Algorithm 35](#) to the graph $G_{14.3}$ in [Figure 14.7](#).

is updated as $\mathcal{U}_4 \leftarrow \emptyset$ (again, this value may later be updated if backtracking is required). The parameter updates $L_4 \leftarrow 3$ and $k \leftarrow 5$ are performed (in [Step 18](#)).

When executing the loop spanning Steps [4–18](#) for the fourth time ($k = 5$), the list of colours available for colouring v_5 is determined to be $\mathcal{U}_5 = \{1, 3\}$. Of these colours, colour 1 is assigned to v_5 , and the list of potentially available alternative colours for v_5 is updated to $\mathcal{U}_5 = \{3\}$. The parameter updates $L_5 \leftarrow 3$ and $k \leftarrow 6$ are performed (in [Step 18](#)).

When entering the loop spanning Steps [4–18](#) for the fifth time ($k = 6$), it is, however, found that none of the three colours is available to colour v_2 (*i.e.* $\mathcal{U}_6 = \emptyset$). We therefore backtrack in [Step 5](#), reducing the value of k to $k \leftarrow 5$ again, recolouring the vertex v_5 with colour 3 (instead of colour 1 as previously) and updating the list of potentially available alternative colours for v_5 to $\mathcal{U}_5 \leftarrow \emptyset$ in [Steps 9–11](#). The parameter update $k \leftarrow 6$ is again performed (in [Step 18](#)) before entering the loop spanning Steps [4–18](#) for the sixth time. Now the list of colours available for colouring v_6 is determined to be $\mathcal{U}_6 = \{1\}$. Hence colour 1 is assigned to v_6 , and the list of potentially available alternative colours for v_6 is updated to $\mathcal{U}_6 \leftarrow \emptyset$. The parameter updates $L_6 \leftarrow 3$ and $k \leftarrow 7$ are performed (in [Step 18](#)).

When we enter the loop spanning Steps [4–18](#) for the seventh time ($k = 7$), the list of colours available for colouring v_7 is found to be $\mathcal{U}_7 = \{3\}$. Hence colour 3 is assigned to v_7 , and the list of potentially available alternative colours for v_7 is updated to $\mathcal{U}_7 \leftarrow \emptyset$. The parameter updates $L_7 \leftarrow 3$ and $k \leftarrow 8$ are performed (in [Step 18](#)).

Upon entering the loop spanning Steps [4–18](#) for the eighth time ($k = 8$), the list of colours available for colouring v_8 is similarly found to be $\mathcal{U}_8 = \{3\}$. Hence colour 3 is also assigned to v_8 , and the list of potentially available alternative colours for v_8 is updated to $\mathcal{U}_8 \leftarrow \emptyset$. The parameter updates $L_8 \leftarrow 3$ and $k \leftarrow 9$ are performed (in [Step 18](#)).

As we enter the loop spanning Steps [4–18](#) for the ninth time ($k = 9$), the list of colours available for colouring v_9 is found to be $\mathcal{U}_9 = \{2, 3\}$. Of these colours, colour 2 is assigned to v_9 , and the list of potentially available alternative colours

for v_9 is updated to $\mathcal{U}_9 \leftarrow \{3\}$. The parameter updates $L_9 \leftarrow 3$ and $k \leftarrow 10$ are performed (in Step 18).

Upon entering the loop spanning Steps 4–18 for the second last time ($k = 10$), the list of colours available for colouring v_{10} is found to be $\mathcal{U}_{10} = \{1, 3\}$. Hence colour 1 is assigned to v_{10} , and the list of potentially available alternative colours for v_{10} is updated as $\mathcal{U}_{10} \leftarrow \{3\}$. Since $k = 10 (= n)$ in Step 14, the colouring in Figure 14.7(d) is stored and the upper bound on the chromatic number of the graph $G_{14.3}$ is lowered to $q \leftarrow 3$ in Step 16. Since the smallest indexed vertex coloured with colour 3 is v_3 , the parameter value $j \leftarrow 3$ is assigned, and the parameters $k \leftarrow 2$ and $\ell \leftarrow 2$ are restored in Step 17. The set \mathcal{U}_2 is, however, empty and so the updates $k \leftarrow 1$ and $\ell \leftarrow 1$ are performed in Step 7. Since $\mathcal{U}_1 = \emptyset$ and $k = 1$ in Steps 5–6, the algorithm terminates with the optimal vertex colouring in Figure 14.7(d) and the chromatic number $\chi(G_{14.3}) = 3$ as output.

❖ The reader should now be able to attempt Exercise 14.19.

14.3 Edge colouring

A **(proper) edge colouring** of a graph G is an assignment of colours to the edges of G , one colour to each edge, so that no two edges incident with the same vertex of G receive the same colour. If k colours are used, then the colouring is referred to as a **k -edge colouring** of G . A k -edge colouring of G therefore partitions the edge set of G into k edge subsets, again called **colour classes** and each containing edges of the same colour. A graph G is said to be **k -edge colourable** if there exists a k -edge colouring of G .

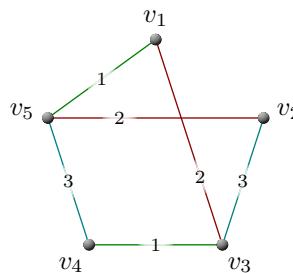


Figure 14.8: A 3-edge colouring of the graph $G_{14.4}$.

The edge colouring in Figure 14.8 is an example of a 3-edge colouring of the graph $G_{14.4}$ with colour classes $\{v_1v_5, v_3v_4\}$, $\{v_1v_3, v_2v_5\}$, and $\{v_2v_3, v_4v_5\}$. The graph $G_{14.4}$ in Figure 14.8 is therefore 3-edge colourable, but not 2-edge colourable (why not?).

14.3.1 The chromatic index of a graph

The **chromatic index** of a graph G , denoted by $\chi'(G)$, is the smallest integer k for which G is k -edge colourable. If $\chi'(G) = k$ for some graph G , then G is said to be **k -edge chromatic**. A proper edge colouring of G using $\chi'(G)$ colours is called a **χ' -edge colouring of G** . The graph $G_{14.4}$ in Figure 14.8 is therefore 3-edge chromatic.

It is obvious that the chromatic index of a graph G is bounded from below by its maximum degree $\Delta(G)$, because the edges incident to a vertex of maximum degree require different colours, i.e. $\chi'(G) \geq \Delta(G)$. The following theorem, due to König [40] and dating from 1916, states that for bipartite graphs this lower bound of $\Delta(G)$ colours always suffices.

Theorem 14.19 $\chi'(G) = \Delta(G)$ for any bipartite graph G .

Proof By induction over the size m of G . If $m = 0$, then $\chi'(G) = \Delta(G) = 0$, while if $m = 1$, then $\chi'(G) = \Delta(G) = 1$. Assume therefore, as induction hypothesis, that $m \geq 2$ and that $\chi'(H) = \Delta(H)$ for all bipartite graphs H of size smaller than m .

Let G be a bipartite graph of size m with maximum degree Δ and suppose $uv \in E(G)$. If $\Delta(G - uv) = \Delta - 1$, then, by the induction hypothesis, there is a $(\Delta - 1)$ -edge colouring of $G - uv$. Such an edge-colouring may be extended to a Δ -edge colouring of G by assigning a new colour to the edge uv . Suppose, therefore, that $\Delta(G - uv) = \Delta$. By the induction hypothesis, there is a Δ -edge colouring of $G - uv$ using the colours $1, \dots, \Delta$.

In $G - uv$, each of u and v is incident with at most $\Delta - 1$ edges. Hence there are colours $\alpha, \beta \in [\Delta]$ such that u is not incident with an edge of colour α and v is not incident with an edge of colour β . If $\alpha = \beta$, then we may extend the Δ -edge colouring of $G - uv$ to a Δ -edge colouring of G by colouring the edge uv with colour α , in which case $\chi(G) = \Delta(G)$. Similarly, if colour β is not used to colour any edge incident with u , then we may colour the edge uv with colour β to obtain a Δ -edge colouring of G , in which case $\chi'(G) = \Delta(G)$ again.

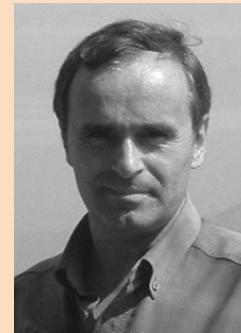
Assume, therefore, that $\alpha \neq \beta$ and that u is incident with an edge of colour β . Let F be the subgraph of G consisting of all paths emanating from u whose edges are alternately coloured β and α . Since each vertex in $G - uv$ is incident with at most one edge of colour α and incident with at most one edge of colour β , the subgraph F is a path starting at u . If $v \in V(F)$, then since v is not incident with an edge of colour β , F is a u - v path starting with an edge of colour β and ending with an edge of colour α , thus having even length. But then $F + uv$ is an odd cycle, contradicting the fact that G is bipartite (see Theorem 2.3). Hence $v \notin V(F)$. If the colours of the edges of F are interchanged, then a new Δ -edge colouring of $G - uv$ results in which neither u nor v is incident with an edge of colour β . We may therefore colour the edge uv with colour β to obtain a Δ -edge colouring of G , in which case $\chi'(G) = \Delta(G)$ yet again. ■

Although there are only very rough estimates for the value of the (vertex) chromatic number of a general graph, there is a surprisingly strong result bounding the chromatic index of any graph. Vizing [54] and Gupta [29] proved independently, in 1964 and 1966, respectively, that the chromatic index of a graph always takes one of two values, as stated in the next theorem.

Theorem 14.20 (Vizing's Theorem) $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$ for any graph G .

Proof By induction over the size m of G . The result is immediate if $m = 0$ or $m = 1$. Assume therefore, as induction hypothesis, that $m \geq 2$ and that $\chi'(H) \leq \Delta(H)$ for all graphs H of size smaller than m . Let G be a bipartite graph of size m with maximum degree Δ .

Vadim Georgievich Vizing was born in Kiev on 25 March 1937. Because his mother was half-German, the Soviet authorities forced his family to move to Siberia in 1947. After completing his undergraduate studies in mathematics in Tomsk State University in 1959, he started work on his doctorate at the Steklov Institute of Mathematics in Moscow, on the subject of function approximation, but he left in 1962 without completing his degree. He returned to Novosibirsk, working at the Russian Academy of Sciences during the period 1962–1968 and earning a doctorate there in 1966. After holding various positions, he moved to Odessa in 1974, where he taught mathematics for many years at the Academy for Food Technology. The result of [Theorem 14.20](#) was published in 1964 when Vizing was working in Novosibirsk and is a natural continuation of the work of Claude Shannon, who showed that the edges of any multigraph can be coloured with at most $3\Delta/2$ colours. Vizing also made other significant contributions to the field of graph colouring, including the introduction of list colouring, the formulation of the total colouring conjecture (still unsolved to this day and stating that the edges and vertices of any graph can together be coloured with at most $\Delta + 2$ colours), [Vizing's Conjecture](#) (also unsolved and concerning the domination number of cartesian products of graphs as we describe later in [Chapter 16](#)), and the 1974 definition of the modular product of graphs as a way of reducing subgraph isomorphism problems to finding maximum cliques in graphs. He also proved a stronger version of [Brook's Theorem](#) that applies to list colouring. From 1976, Vizing stopped working on graph theory and studied scheduling problems instead, only returning to graph theory again as late as 1995. He died on 23 August 2017.



Biographic note 62: Vadim Vizing (1937–2017)

We shall show that if we have coloured all but one of the edges of G with colours $1, \dots, \Delta + 1$, then the last edge can also be coloured with one of these colours. We refer to edges coloured with colour i as **i -edges** and say that a colour is **represented** at a vertex v if it is assigned to some edge incident with v ; otherwise, we say that the colour is **missing** at v . If a colour α is represented at a vertex v and a colour β is missing at v , then the subgraph consisting of all paths emanating at v whose edges are alternately coloured α and β is a single path (since each vertex is incident with at most one α -edge and at most one β -edge) that starts at v . We call such a path the **(α, β) -path from v** .

For any edge $uv \in E(G)$, there is a $(\Delta + 1)$ -edge colouring of $G - uv$ by the induction hypothesis. We may assume that if the colour α is missing at u and the colour β is missing at v , then the (β, α) -path from u is a u - v path, for otherwise we could interchange the colours α and β along this path and colour the edge uv with the colour β to produce a $(\Delta + 1)$ -edge colouring of G .

Let $uv_1 \in E(G)$. By the induction hypothesis there is a $(\Delta + 1)$ -edge colouring \mathcal{C}_1 of $G - uv_1$ using the colours $1, \dots, \Delta + 1$. Let α and β_1 be colours missing

at u and v_1 , respectively, in \mathcal{C}_1 . Let v_2, \dots, v_k be a maximal sequence of distinct neighbours of u in G such that colour β_i is missing at v_i and the edge uv_i has colour β_{i-1} for all $i = 2, \dots, k$.

For $i \in \{2, \dots, k\}$, let \mathcal{C}_i be the $(\Delta + 1)$ -edge colouring of $G - uv_i$ obtained from \mathcal{C}_1 by uncolouring the edge uv_i , colouring the edge uv_ℓ with the colour β_ℓ for $\ell \in [\ell - 1]$, and leaving the colours of all other edges unchanged. Note that in each of these edge colourings the same colours are missing from u .

We now return to the edge colouring \mathcal{C}_1 . Let $\beta = \beta_k$. Suppose the colour β is missing from u . Then the colouring \mathcal{C}_k may be extended to a $(\Delta + 1)$ -edge colouring of G by colouring the edge uv_k with colour β . Hence we may assume that the colour β is represented at u (but is missing from v_k).

By the maximality of k , the edge uv_j is coloured β for some $j \in \{2, \dots, k - 1\}$. Let P be the (α, β) -path from u in $G - uv_k$. By our earlier assumption, this path starts at u and ends at v_k . Furthermore, it starts with a β -edge and ends with an α -edge. Since the edge uv_{j-1} is coloured with colour β in \mathcal{C}_k , the initial edge of P is the edge uv_{j-1} . Since the colour β is missing from v_j in \mathcal{C}_k , it follows that $v_j \notin V(P)$. Moreover, all the edges of $P - u$ are coloured the same in \mathcal{C}_1 as in the colouring \mathcal{C}_k , and so there is no $v_{j-1}-v_j$ path whose edges are alternately coloured α and β in \mathcal{C}_1 .

Note that in \mathcal{C}_1 and in \mathcal{C}_{j-1} the colour β is missing from v_{j-1} . Let Q be the (α, β) -path from u in $G - uv_{j-1}$. By our earlier assumption, Q starts at u and ends at v_{j-1} . Furthermore, it starts with a β -edge and ends with an α -edge. Since the edge uv_j is coloured β in \mathcal{C}_{j-1} , the initial edge of Q is the edge uv_j . The edges of $Q - u$ are coloured the same in \mathcal{C}_1 as in the colouring \mathcal{C}_{j-1} , and so there exists a $v_{j-1}-v_j$ path whose edges are alternately coloured α and β in \mathcal{C}_1 , producing a contradiction. ■

❖ The reader should now be able to attempt Exercises 14.20–14.22 and Project 14.6.

14.3.2 Criticality with respect to $\chi'(G)$

It is a simple matter to verify that if H is a subgraph of G , then $\chi'(H) \leq \chi'(G)$. A graph G is **critical with respect to $\chi'(G)$** if, for every subgraph H of G , it holds that $\chi'(H) < \chi'(G)$. Also, a graph G is **k -critical with respect to $\chi'(G)$** if $\chi'(G) = k$ and G is critical with respect to $\chi'(G)$.

We quote two useful criticality results from Wallis [55, §7.5] without proof, because the proofs are technical and rather lengthy in nature.

Theorem 14.21 *If G is a critical graph with respect to $\chi'(G)$, then every vertex of G is adjacent to at least two vertices of maximum degree in G .*

Theorem 14.22 *If G is a critical graph with respect to $\chi'(G)$, then G has at least $\Delta(G) - \delta(G) + 2$ vertices of maximum degree.*

The converse results of Theorems 14.21 and 14.22 are useful in the sense that they may be used in some instances to verify that a graph is not critical with respect to $\chi'(G)$: If a graph contains a vertex that is not adjacent to at least two vertices of maximum degree or if the graph does not have at least $\Delta(G) - \delta(G) + 2$ vertices of maximum degree, then it cannot be critical with respect to $\chi'(G)$.

❖ The reader should now be able to attempt Exercises 14.23–14.24.

Robin James Wilson was born in the United Kingdom on 5 December 1943. He studied at Balliol College, Oxford and at the University of Pennsylvania. He later became professor of mathematics at the Open University, stipendiary lecturer at Pembroke College, Oxford and professor of geometry at Gresham College, London. On occasion, he guest teaches at Colorado College. Robin Wilson's academic interests lie in graph colouring problems and algebraic properties of graphs. He also researches the history of mathematics, focussing on British mathematics and mathematics during the 17th century and the period 1860–1940, as well as on the history of graph theory and combinatorics. Wilson was editor-in-chief of the European Mathematical Society Newsletter during the period 1999–2003. He is the son of Harold Wilson, former Prime Minister of the United Kingdom.



Biographic note 63: Robin Wilson (1943–present)

14.3.3 Computing the chromatic index of a graph

According to [Theorem 14.20](#), the set of all graphs may be partitioned into two classes, namely the class of graphs G for which $\chi'(G) = \Delta(G)$ (called **Class 1** graphs) and the class of graphs G for which $\chi'(G) = \Delta(G) + 1$ (called **Class 2** graphs). [Erdős and Wilson \[22\]](#) proved in 1977 that the probability that a graph of order n is a Class 1 graph approaches 1 as $n \rightarrow \infty$. In this sense there are considerably more Class 1 graphs than Class 2 graphs. The decision problem of determining whether a randomly generated graph is a Class 1 graph or a Class 2 graph was, however, shown to be NP-complete by [Holyer \[34\]](#) in 1981.

The following result may nevertheless be used in some instances to verify that a graph is a Class 2 graph.

Theorem 14.23 *If G is a graph of order n and size m for which $m > \lfloor \frac{n}{2} \rfloor \Delta(G)$, then G is a Class 2 graph.*

Proof By contradiction. Suppose, to the contrary, that G is a graph of order n , of size m , with maximum degree Δ and that $m > \lfloor \frac{n}{2} \rfloor \Delta$, but that G is a Class 1 graph. Consider a Δ -edge colouring of G with colour classes $\mathcal{C}_1, \dots, \mathcal{C}_\Delta$. Then, since no colour class can contain more than $\lfloor \frac{n}{2} \rfloor$ edges,

$$m = \sum_{i=1}^{\Delta} |\mathcal{C}_i| \leq \left\lfloor \frac{n}{2} \right\rfloor \Delta,$$

a contradiction. ■

The body of work on the use of critical graphs with respect to classifying various graph classes as Class 1 or 2 graphs is large, but for our purposes it will suffice to state only two further useful results. Since the proofs of these results are technical in nature, we again quote the results from [Wallis \[55, §7.5\]](#) without proof.

Theorem 14.24 *Every Class 2 graph G contains a k -critical subgraph with respect to $\chi'(G)$ for each $k \in \{2, \dots, \Delta(G)\}$.*

Theorem 14.25 Every Class 2 graph has at least three vertices of maximum degree.

The converse of [Theorem 14.25](#) is especially useful in the sense that it may be used in some instances to verify that a graph is a Class 1 graph, for if a graph does not have at least three vertices of maximum degree, then it cannot be a Class 2 graph.

❖ The reader should now be able to attempt Exercises [14.25–14.27](#).

Exercises

14.1 Verify that each of the graphs in [Figure 14.9](#) is 3-colourable.

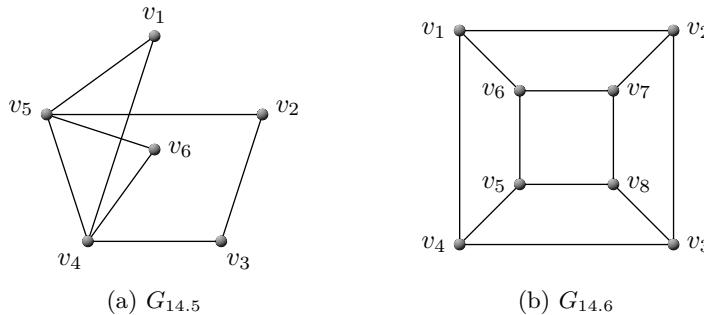


Figure 14.9: Graphs that are 3-colourable.

- 14.2 (a) Show that the map of Africa in [Figure 14.10](#) is 4-colourable, where two countries may be coloured with the same colour if they share no common border of positive length.
 (b) Is the map of Africa in [Figure 14.10](#) 3-colourable? Explain.
- 14.3 (a) Give an example of a planar graph that is not 3-colourable.
 (b) Give an example of a triangle-free graph that is not 3-colourable.
- 14.4 Prove or disprove the following statement: If a graph G is 4-colourable, then G is planar.
- 14.5 Suppose G is a connected graph of order n and size $m \leq n$. Prove that G is 3-colourable.
- 14.6 Prove that a graph G is 3-critical with respect to $\chi(G)$ if and only if G is an odd cycle.
- 14.7 Prove that if $\chi(G) = k$ for some graph G , then G contains at least k vertices, each of degree at least $k - 1$.
- 14.8 Prove or disprove the following statement: If G is a k -chromatic graph, then G contains a subgraph isomorphic to K_k .
- 14.9 Suppose G is a k -critical graph with respect to $\chi(G)$ for some $k \geq 2$. Prove that G contains a $(k - 1)$ -critical subgraph.
- 14.10 (a) Prove [Theorem 14.11\(iii\)](#).



Figure 14.10: Map of the African continent.

- (b) Find the chromatic number of the wheel graph $W_n = C_n + K_1$.
- 14.11 Find the chromatic number of the Petersen graph, the Grötzsch graph and the Heawood graph shown in [Figure 14.11](#).

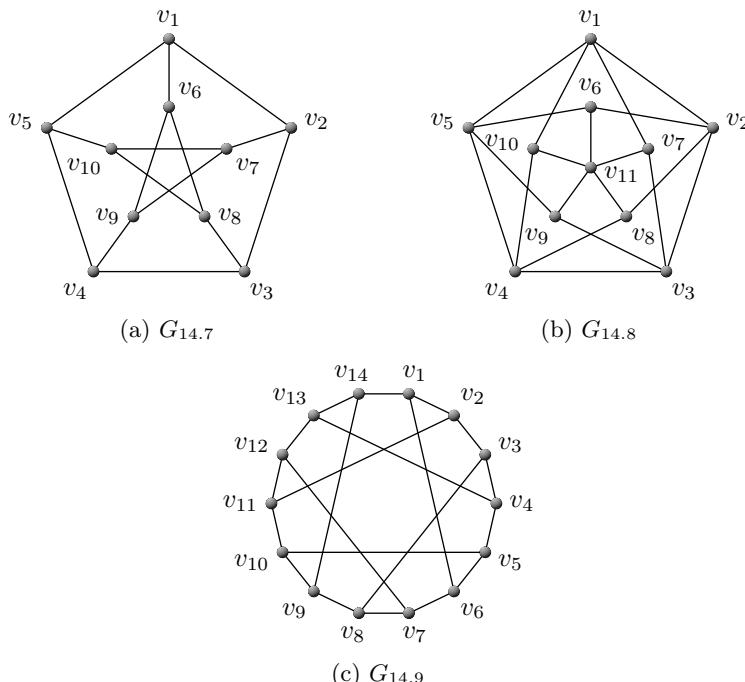


Figure 14.11: (a) The Petersen graph, (b) the Grötzsch graph and (c) the Heawood graph.

- 14.12 Prove that $\chi(G \square H) = \max\{\chi(G), \chi(H)\}$ for any graphs G and H .
- 14.13 (a) Suppose τ is the length of a longest path in a graph G . Prove that $\chi(G) \leq \tau + 1$.
 (b) Produce, for each positive integer τ , a graph G with chromatic number $\tau + 1$ and in which any longest path has length τ .
- 14.14 Use the *sequential colouring heuristic* ([Algorithm 31](#)) to find vertex colourings for the two graphs in [Figure 14.12](#).
- 14.15 Prove [Theorem 14.19](#).
- 14.16 Use the *largest-first* or *Welsh-Powell heuristic* ([Algorithm 32](#)) to find (vertex) colourings for the two graphs in [Figure 14.12](#).
- 14.17 Use *Brélaz's heuristic* ([Algorithm 33](#)) to find (vertex) colourings for the two graphs in [Figure 14.12](#).
- 14.18 Use the *smallest-last heuristic* ([Algorithm 34](#)) to find (vertex) colourings for the two graphs in [Figure 14.12](#).
- 14.19 Use *Brown's Algorithm* ([Algorithm 35](#)) to determine the chromatic numbers of the two graphs in [Figure 14.12](#).

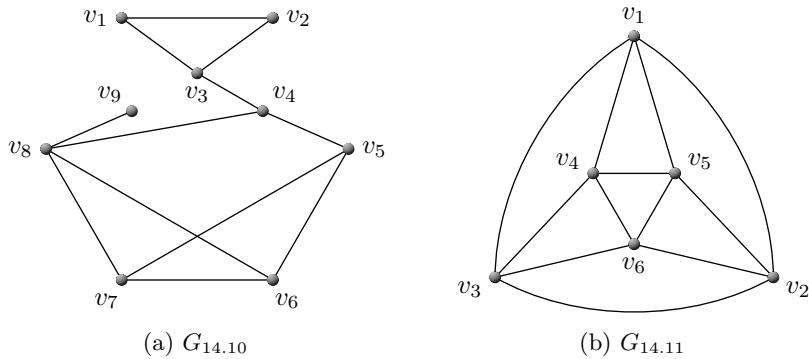


Figure 14.12: The graphs $G_{14.10}$ and $G_{14.11}$.

14.20 Prove that

$$\chi'(K_n) = \begin{cases} n & \text{if } n \text{ is odd,} \\ n - 1 & \text{if } n \text{ is even.} \end{cases}$$

14.21 Find the chromatic index of the Petersen graph, the Grötzsch graph and the Heawood graph shown in Figure 14.11.

14.22 Show that every hamiltonian, cubic graph is 3-edge chromatic.

14.23 Which of the graphs in Figure 14.13 are critical with respect to the chromatic index χ' ? Motivate.

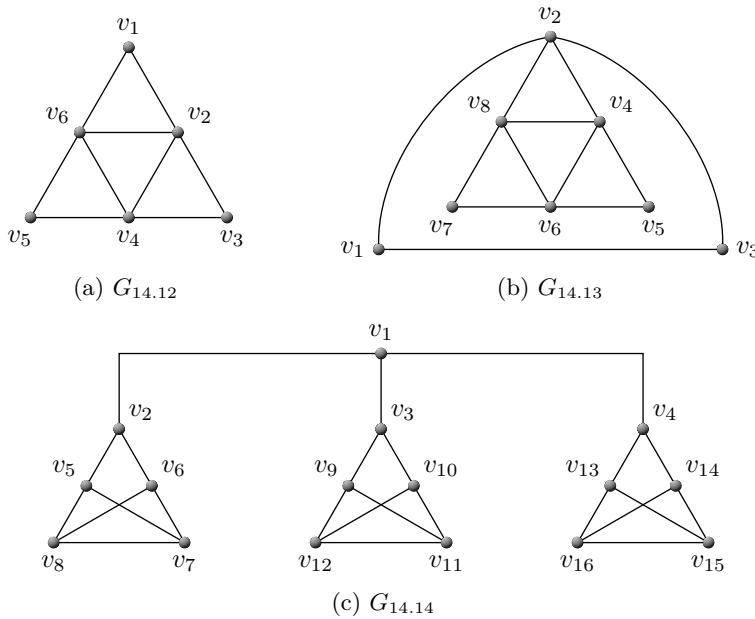


Figure 14.13: Critical with respect to the chromatic index χ' ?

14.24 Suppose G is a Δ -critical graph with respect to $\chi'(G)$ and that uv is an edge of G . Prove that $d(u) + d(v) \geq \Delta + 2$.

14.25 Prove that:

- (a) For every graph G of order 5 and size 7, $\chi'(G) = 4$.
- (b) Every graph of order 5 and size 8 is a Class 1 graph.

14.26 Prove that all trees are Class 1 graphs.

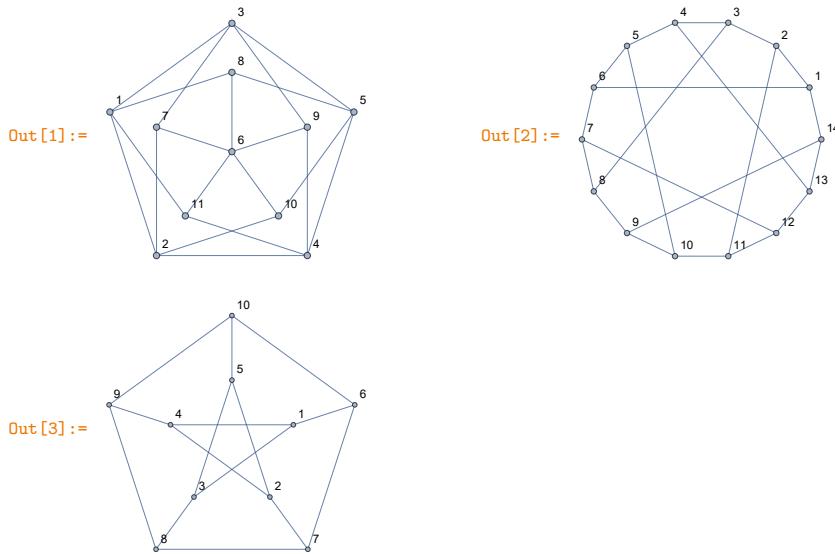
14.27 Prove that all nonempty, regular graphs of odd order are Class 2 graphs.

Computer exercises

A host of well-known graphs are known to **MATHEMATICA** and may be retrieved by means of built-in commands. In [Exercise 14.11](#) we encountered three famous graphs, namely the Grötzsch graph, the Heawood graph and the Petersen graph. These graphs are known to **MATHEMATICA** by the names `GroetzschGraph`, `HeawoodGraph` and `PetersenGraph`, respectively. For example, the commands

```
In[1]:= F = GraphData["GroetzschGraph", "Graph", "Labeled"]
In[2]:= G = GraphData["HeawoodGraph", "Graph", "Labeled"]
In[3]:= H = PetersenGraph[VertexLabels -> "Name"]
```

yield the output:



The chromatic number of a graph may be determined by utilising the notion of a chromatic polynomial. The command `ChromaticPolynomial[G, k]` may be used to produce this polynomial for the graph G . The coefficient of the term x^k in this polynomial represents the number of vertex colourings of G using k colours. Moreover, the chromatic index of G may be found by determining the chromatic number of its line graph. The line graph of G is another graph $L(G)$ representing adjacencies between the edges of G . The line graph $L(G)$ may be constructed by creating a vertex in $L(G)$ for each edge in G , and inserting an edge in $L(G)$ for every two edges in G that share a vertex in common. For example, the commands

```
In[4]:= ChromaticNumber[g_] := (k = 1; While[ ChromaticPolynomial[g, k] == 0,
k++]; k)
```

```
In[5]:= ChromaticNumber[F]
In[6]:= ChromaticNumber[LineGraph[G]]
```

produce the following chromatic number and chromatic index values associated with the Grötzsch graph:

```
Out[5]:= 4
Out[6]:= 3
```

Brélaz's heuristic may be implemented in **MATHEMATICA** by means of the following function:

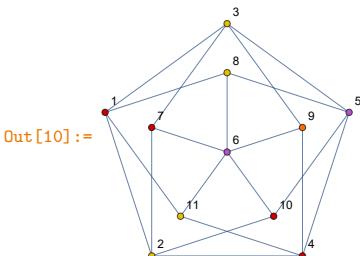
```
In[7]:= Brelaz[g_] := Module[{m = 0, v = VertexCount[g], cd, color, p, nc, e},
    cd = color = ConstantArray[0, v];
    While[m >= 0, p = Position[cd, m][[1, 1]];
        e = AdjacencyList[g, p];
        nc = Append[color[[e]], 0];
        color[[p]] = Min[Complement[Range[Max[nc] + 1], nc]];
        cd[[p]] = -2 v;
        Scan[(cd[[#]]++) &, e];
        m = Max[cd]];
    color]
```

The function may be applied to the Grötzsch graph by means of the commands:

```
In[8]:= colF = Brelaz[F]
In[9]:= vertexpartition = Table[Flatten[Position[colF, i]], {i, 1, Max[colF]}]
In[10]:= HighlightGraph[F, vertexpartition]
```

which yield respectively the following list of colours assigned to the vertices of the graph, the list of corresponding vertex colour classes, and a graphical representation of the vertex colouring:

```
Out[8]:= {1, 2, 2, 1, 3, 3, 1, 2, 4, 1, 2}
Out[9]:= {{1, 4, 7, 10}, {2, 3, 8, 11}, {5, 6}, {9}}
```



Brélaz's heuristic may similarly be implemented in **MATHEMATICA** to determine the chromatic index of a graph by defining the following function in terms of its line graph:

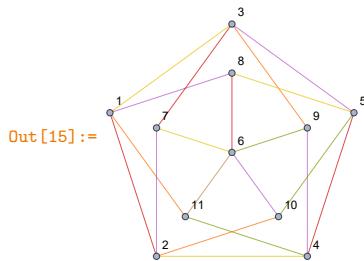
```
In[11]:= EdgeColoring[g_] := Module[{c = Brelaz[LineGraph[g]], e, se},
    e = Map[Sort, List@@@EdgeList[g]];
    se = Sort[MapIndexed[Prepend[#2, #1] &, e]];
    Map[Last, Sort[Reverse[MapThread[Prepend, {se, c}], 2]]]
]
```

The function may be applied to the Grötzsch graph by means of the commands:

```
In[12]:= colF = EdgeColoring[F]
In[13]:= partition = Table[ Flatten[Position[colF, i]], {i, 1, Max[colF]}];
In[14]:= edgepartition = Table[EdgeList[F][[partition[[i]] ]], {i, 1,
Length[partition]} ]
In[15]:= HighlightGraph[F, edgepartition]
```

which yield respectively the following list of colours assigned to the edges of the graph, the list of corresponding edge colour classes, and a graphical representation of the edge colouring:

```
Out[12]:= {1, 2, 3, 4, 2, 3, 4, 3, 1, 4, 1, 3, 5, 2, 5, 2, 1, 5, 3, 6}
Out[14]:= {{1 ↔ 2, 3 ↔ 7, 4 ↔ 5, 6 ↔ 8}, {1 ↔ 3, 2 ↔ 4, 5 ↔ 8, 6 ↔ 7}, {1 ↔ 8,
           2 ↔ 7, 3 ↔ 5, 4 ↔ 9, 6 ↔ 10}, {1 ↔ 11, 2 ↔ 10, 3 ↔ 9}, {4 ↔ 11,
           5 ↔ 10, 6 ↔ 9}, {6 ↔ 11}}
Out[15]:= 
```



- ❖ The reader should now be able to repeat Exercises 14.1, 14.2, 14.11, 14.17 and 14.21 without pen and paper.

Projects

This section contains six projects. In the first project, the reader is required to design and analyse an algorithm for deciding whether a given graph is bipartite, while in the second project, the reader is invited to identify and explain the error in Alfred Kempe's 1879 attempted proof of the [Four-colour Theorem](#). Two further projects (the third and last projects) are devoted to timetabling applications involving respectively vertex and edge colourings of appropriate graphs. The remaining two projects are dedicated to generalised colourings related to the types of colourings described in this chapter, namely maximum degree colourings, path colourings and clique colourings. While these generalised colourings are not described explicitly in this chapter, they all admit the notion of classical vertex colouring as a special case.

Project 14.1: An algorithm for deciding bipartiteness

We saw, in [Theorem 14.1](#), that the class of bipartite graphs is precisely the class of 2-colourable graphs. In the paragraph directly following [Theorem 14.1](#), we furthermore described a simple method of determining whether a graph is bipartite or not.

Tasks

1. Design an algorithm, called **Bipartite**, which takes a graph G of order n and size m as input, and returns **true** if G is a bipartite graph or **false** otherwise. Write down the algorithm in pseudocode form.
2. Illustrate the correct working of the algorithm by considering first an input graph that is, in fact, bipartite, and then one that is not.

3. Prove that $D_{\text{colour}}(G, 2) \in \mathcal{P}$ for any graph G by estimating the worst-case time complexity of Algorithm Bipartite (in [Task 1](#)), showing that this complexity is a polynomial function of the parameter values n and m .

Project 14.2: Kempe's proof of the Four-colour Theorem

Alfred Bray Kempe used an inductive argument, known as the *method of Kempe chains*, in his 1879 attempted proof of the [Four-colour Theorem](#) (in the context of map colouring) [39]. Here is his attempted proof...

Consider a map in which every region is coloured, using the colours 1, 2, 3 or 4, except for one region, say X. If this final region X is not surrounded by regions of all four colours, then a colour remains to be assigned to X. Hence, suppose that regions of all four colours surround X. If X is surrounded by regions A, B, C, D in order, coloured 1, 2, 3 and 4, respectively, then there are two cases to consider, as illustrated in [Figure 14.14](#): (a) there is no chain of adjacent regions from A to C alternately coloured 1 and 3, or (b) there is a chain of adjacent regions from A to C alternately coloured 1 and 3.

Case (a). Change the colour of region A to colour 3, and then interchange the colours in all chains of successive regions coloured alternately with colours 1 and 3, starting from region A, as illustrated in [Figure 14.15\(a\)](#). Since region C is not in any such chain, it remains coloured 3 and there is now no region adjacent to region X coloured 1. Colour region X, using colour 1.

Case (b). There is no chain of successive regions coloured alternatively with colours 2 and 4, starting from region D and ending in region B, as illustrated in [Figure 14.15\(b\)](#). (Such a chain cannot cross the chain of successive regions coloured alternatively with colours 1 and 3, starting from region A.) Hence, **Case (a)** holds for D and one may interchange colours 2 and 4 as above, and finally colour region X, using colour 4.

Task

Identify and explain the error in Kempe's attempted proof.

Project 14.3: Examination timetabling

Suppose a university needs to draw up an examination timetable. The examination timetable should accommodate forty courses, as listed in [Table 14.4](#). The numbers of courses that may be taken in conjunction with each course (*i.e.* courses that may be taken simultaneously, by the same students) are listed in the third column of the table.

Tasks

1. Construct a course graph of order 40 for the university in which the vertices represent the various courses offered, and in which two vertices are adjacent if they may be taken in conjunction with one another.
2. Find the best lower and upper bounds on the chromatic number of the course graph in [Task 1](#) at your disposal via the theorems in this chapter.
3. Use the [Sequential colouring heuristic](#) (Algorithm 31) to colour the vertices of the course graph in [Task 1](#).

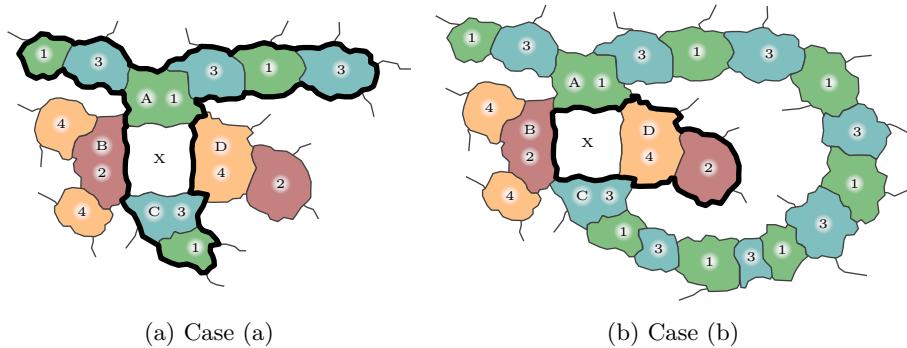


Figure 14.14: An illustration of the two cases that may arise when all four surrounding regions of an uncoloured region X , are already coloured. (a) There is no chain of adjacent regions from A to C alternately coloured 1 and 3. (b) There is a chain of adjacent regions from A to C alternately coloured 1 and 3.

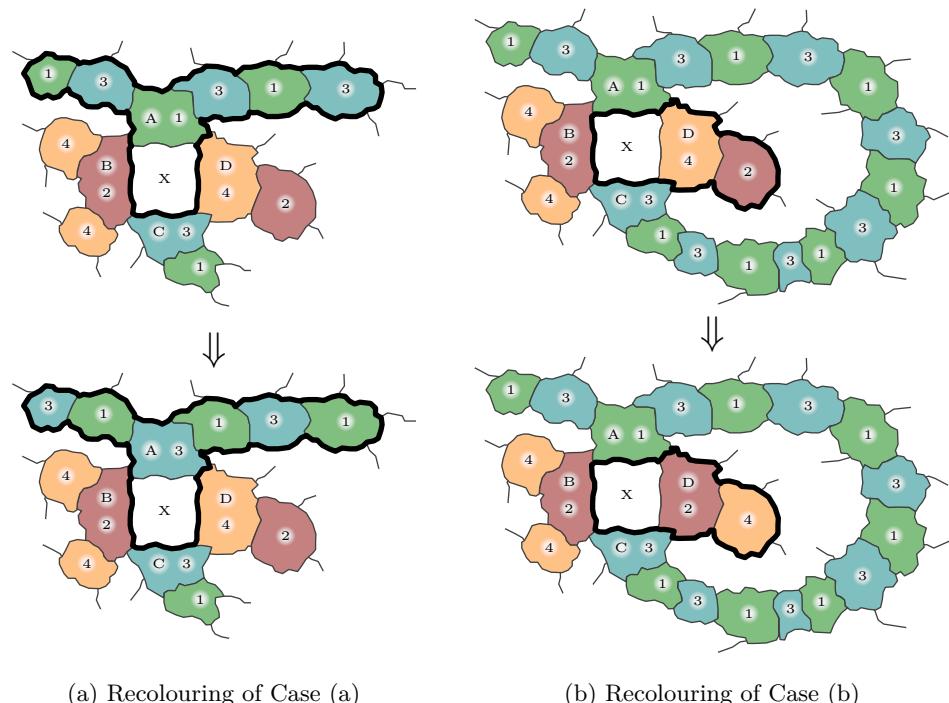


Figure 14.15: Kempe chains used to recolour parts of a map.

No.	Course	Courses in conjunction
1	Accounting	3, 11, 14, 20, 26, 37
2	Agricultural Economics	14, 26, 29
3	Applied Mathematics	1, 22, 23, 28, 37, 39
4	Archaeology	15, 18, 28
5	Biochemistry	6, 7, 29, 40
6	Biology	5, 7, 13, 29, 40
7	Biostatistics	5, 6, 29, 40
8	Chemistry	23, 28, 29, 37, 39, 40
9	Commercial Law	11, 14, 17, 21, 32, 34
10	Dutch	12, 16, 17, 18, 27, 33, 35, 36, 38
11	Economics	1, 9, 14, 20, 26, 37
12	English	10, 16, 17, 18, 21, 24, 27, 33
13	Entomology	6, 29, 40
14	Financial Planning	1, 2, 9, 11, 20, 22, 26, 32
15	Geography	4, 18, 27, 35, 36
16	German	10, 12, 17, 24, 33
17	Greek	9, 10, 12, 16, 18, 21, 35, 38
18	History	4, 10, 12, 17, 27, 28, 31, 34, 35, 36, 38
19	Human Movement Science	25, 28, 29, 40
20	Industrial Psychology	1, 11, 14, 22, 26, 37
21	Latin	9, 12, 17, 32, 34, 35
22	Logistics	3, 14, 20, 23, 37
23	Mathematics	3, 8, 22, 28, 37, 39
24	Music	12, 16, 33
25	Occupational Therapy	19, 29, 33, 36
26	Operations Research	1, 2, 11, 14, 20, 32, 37
27	Philosophy	10, 12, 15, 18, 30, 35, 36
28	Physics	3, 4, 8, 19, 23, 37, 39
29	Physiology	2, 5, 6, 7, 8, 13, 19, 25, 40
30	Political Philosophy	27, 31, 33, 34, 38
31	Political Science	18, 30, 34
32	Private Law	9, 11, 14, 21, 26, 34
33	Psychology	10, 12, 16, 24, 25, 30, 38, 40
34	Roman Law	9, 18, 21, 31, 32
35	Scriptural Doctrine	10, 15, 17, 18, 21, 27
36	Sociology	10, 15, 18, 25, 27, 30
37	Statistics	1, 3, 8, 11, 20, 22, 23, 26, 28, 39
38	Theology	10, 17, 18, 30, 33
39	Theoretical Physics	3, 8, 23, 28, 37
40	Zoology	5, 6, 7, 8, 13, 19, 29, 33

Table 14.4: Courses to be accommodated in a university examination timetable.

4. Construct a timetable corresponding to the colouring obtained in [Task 3](#). Your answer should be in the following form:

Slot 1	Slot 2	...	Slot i	...	Slot k
Course (1, 1)	Course (2, 1)	...	Course (i , 1)	...	Course (k , 1)
Course (1, 2)	Course (2, 2)	...	Course (i , 2)	...	Course (k , 2)
Course (1, 3)	Course (2, 3)	...	Course (i , 3)	...	Course (k , 3)
:	:	:	:	:	:

The columns Course (i , 1), Course (i , 2), Course (i , 3),... should be sorted alphabetically for all $i \in [k]$ and the number of each course in [Table 14.4](#) should also be listed.

5. Use *Brélaz's colouring heuristic* ([Algorithm 33](#)) to colour the vertices of the course graph in [Task 1](#).
6. Construct a timetable corresponding to the colouring obtained in [Task 5](#). Your answer should be in the same form as that of [Task 4](#).
7. Which of the timetables in Tasks 4 or 6 is the best from a practical point of view? Motivate.

Project 14.4: Shared resource scheduling

The users of a computer network may be considered to be in conflict if they require access to the same file servers simultaneously. Some threshold of conflict may, however, usually be tolerated during file retrieval. For example, each user of the network may find the slowdown incurred during file retrieval simultaneously with, say two other users acceptable, but a conflict with three or more users may perhaps cause a slowdown that is unacceptable to the users.

A so-called *file access graph* may be constructed in which the vertices represent the network users and in which two vertices are adjacent if the corresponding users require access to the same file servers. For example, a network comprising eight users and three file servers is shown in [Figure 14.16\(a\)](#). The lines in the figure indicate to which file server each user requires access. The corresponding file access graph is shown in [Figure 14.16\(b\)](#), where the users have been labelled u_1, \dots, u_8 .

A *file access schedule* is an assignment of users to a number of time slots during which those users may gain access to file servers, so as to limit the amount of slowdown during any particular time slot in accordance with the acceptable threshold of conflict. The file access schedule is *optimal* if it consists of the minimum number of time slots. Construction of an optimal file access schedule may be accomplished by colouring the vertices of the file access graph with the minimum number of colours in such a way that the maximum degree of each colour class induced subgraph is at most some nonnegative integer value d representing the threshold of conflict — such a colouring is called a **d -maximum degree colouring**. In our example above, $d = 2$ and a 2-maximum degree colouring of the file access graph in two colours is also shown in [Figure 14.16\(b\)](#). This number of colours is clearly optimal, since the maximum degree of the file access graph is $\Delta = 7 > 2$. The corresponding optimal file access schedule is given in [Table 14.5](#).

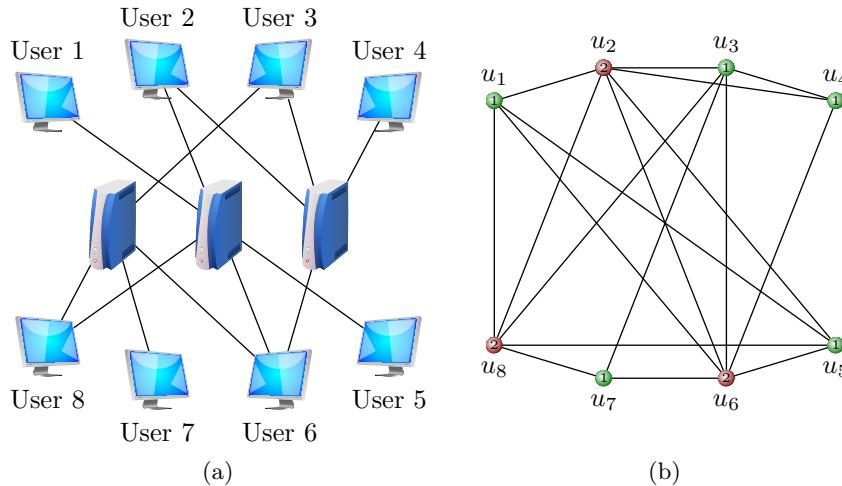


Figure 14.16: (a) Schematic representation of the users of a computer network.
(b) The corresponding file access graph for the computer system.

Time slot	Users
1	\$u_1, u_3, u_4, u_5, u_7\$
2	\$u_2, u_6, u_8\$

Table 14.5: Optimal file access schedule for the computer user network in Figure 14.16(a) according to the generalised vertex colouring in Figure 14.16(b).

This application gives rise to a new kind of (vertex) chromatic number. The **d -th maximum degree chromatic number** of a graph G , denoted by $\chi_d^\Delta(G)$, is the smallest integer k for which a d -maximum degree colouring of G exists (for some fixed value of $d \in \mathbb{N}$). The (classical) chromatic number of a graph is therefore a special case of the d -th maximum degree chromatic number, in the sense that $\chi(G) = \chi_0^\Delta(G)$ for any graph G .

The interested reader is referred to [44] for further applications of, and algorithms for, the scheduling problem described above.

Tasks

1. Prove that $1 \leq \chi_{d+1}^\Delta(G) \leq \chi_d^\Delta(G)$ for any graph G and any nonnegative integer d .
2. Find the maximum degree chromatic sequence $\chi_0^\Delta(G), \chi_1^\Delta(G), \chi_2^\Delta(G), \dots$ for each of the following graphs:
 - (a) P_n ;
 - (b) C_n ; and
 - (c) K_n .
3. Suppose there are thirteen employees at an IT company. The company owns four file servers and the programming duties of the employees are such that they require access to these file servers, as listed in Table 14.6.

Employee	Access to file server
Bernie	1, 3
Brian	2, 3
Christiaan	1, 3, 4
Fanie	2, 4
Jancke	1, 2, 3
Janneke	1, 2
Johan	3, 4
Louw	1, 4
Renier	1, 2, 3
Robert	1
Ryan	1, 2, 3, 4
Samantha	2
Thorsten	2, 3

Table 14.6: Employees of an IT firm and their file server access requirements.

- (a) Construct a file access graph Ψ for the employees of the company.
- (b) Determine maximum degree colourings for the file access graph Ψ in [Task 3\(a\)](#) using $\chi_d^\Delta(\Psi)$ colours, for the following thresholds of conflict:
 - (i) $d = 0$;
 - (ii) $d = 1$; and
 - (iii) $d = 2$.
- (c) Construct optimal file access schedules corresponding to the colourings in [Task 3\(b\)](#).

Project 14.5: Other generalised colourings

In addition to the (generalised) notion of a maximum degree (vertex) colouring introduced in [Project 14.4](#), there is a host of other ways to generalise the classical notion of a (proper) vertex colouring described in this chapter. In this project we explore just two of these notions of generalised (vertex) colouring.

Path Colourings

Chartrand *et al.* [16] introduced the notion of a path colouring of a graph. A **t -th path colouring** of a graph G is an assignment of colours to the vertices of G , one colour to each vertex, so that the length of a longest path in each colour class induced subgraph of G is at most t . If k colours are used, then the colouring is referred to as a **$\tau(t, k)$ -colouring** of G .

The **t -th path chromatic number** of a graph G , denoted by $\chi_t^\tau(G)$, is the smallest integer k for which a $\tau(t, k)$ -colouring of G exists (for some fixed value of $t \in \mathbb{N}$). The (classical) chromatic number of a graph is therefore a special case of the t -th path chromatic number in the sense that $\chi(G) = \chi_1^\tau(G)$ for any graph G . Finally, the path chromatic sequence of a graph G is merely the infinite sequence $\chi_1^\tau(G), \chi_2^\tau(G), \chi_3^\tau(G), \dots$

The interested reader is referred to [4, 36, 59] for further results on path colourings of graphs.

Tasks

1. Prove that $1 \leq \chi_{t+1}^\tau(G) \leq \chi_t^\tau(G)$ for any graph G and any $t \in \mathbb{N}$.
2. Prove that $\chi_t^\tau(G) \leq \chi(G) \leq t\chi_t^\tau(G)$ for any graph G and any $t \in \mathbb{N}$.
3. Find the t -th path chromatic sequence for each of the following graphs:
 - (a) P_n ;
 - (b) C_n ; and
 - (c) K_n .

Clique Colourings

The notion of a clique colouring of a graph was introduced by [Sachs \[52\]](#) and has subsequently been studied comprehensively [\[8, 9, 10, 11, 25\]](#). A **t -th clique colouring** of a graph G is an assignment of colours to the vertices of G , one colour to each vertex, so that the order of a largest clique in each colour class induced subgraph of G is at most t . If k colours are used, then the colouring is referred to as an **$\omega(t, k)$ -colouring** of G .

The **t -th clique chromatic number** of a graph G , denoted by $\chi_t^\omega(G)$, is the smallest integer k for which an $\omega(t, k)$ -colouring of G exists (for some fixed value of $t \in \mathbb{N}$). The (classical) chromatic number of a graph is therefore also a special case of the t -th clique chromatic number in the sense that $\chi(G) = \chi_1^\omega(G)$ for any graph G . Finally, the clique chromatic sequence of a graph G is merely the infinite sequence $\chi_1^\omega(G), \chi_2^\omega(G), \chi_3^\omega(G), \dots$

Tasks (continued)

4. Prove that $\chi_t^\omega(G) \leq \lceil \chi(G)/t \rceil$ for any graph G and any $t \in \mathbb{N}$.
5. Prove that $\chi_t^\omega(G) \geq \lceil \omega(G)/t \rceil$ for any graph G and any $t \in \mathbb{N}$.
6. Prove that $1 \leq \chi_{t+1}^\omega(G) \leq \chi_t^\omega(G)$ for any graph G and any $t \in \mathbb{N}$.
7. Find the t -th clique chromatic sequence for each of the following graphs:
 - (a) P_n ;
 - (b) C_n ; and
 - (c) K_n .

Project 14.6: Class timetabling

Suppose the mathematics department of a university has to draw up a class timetable. It offers sixteen courses, as listed in [Table 14.7](#), which are presented by four faculty members, Dr Durbach, Dr Peacock, Prof Scott and Prof Plumb.

The first digit of each course code indicates the year of study during which a student may take the course (for example, Math 101 may be taken during the first year of study, whilst Math 302 may be taken during the third year of study).

Course code	Course topic	Lecturer
Math 101	College algebra	Dr Durbach
Math 102	Calculus	Dr Durbach
Math 103	Real analysis	Dr Peacock
Math 104	Vector algebra	Prof Scott
Math 201	Linear algebra	Prof Scott
Math 202	Graph theory	Dr Durbach
Math 203	Numerical analysis	Prof Plumb
Math 204	Mathematical biology	Dr Peacock
Math 301	Vector calculus	Prof Plumb
Math 302	Set theory	Prof Plumb
Math 303	Number theory	Dr Durbach
Math 304	Mathematics of finance	Prof Scott
Math 401	Group theory	Dr Peacock
Math 402	Differential equations	Prof Scott
Math 403	Complex analysis	Prof Plumb
Math 404	Topology	Dr Peacock

Table 14.7: Courses offered by a mathematics department, and the lecturers responsible for the courses.

A first-year student may take any of the four first-year courses — hence, these courses should not clash in the timetable — and similarly for the other years of study. The timetable should also not clash with respect to the lecturers who present the courses.

The department would like to draw up a class timetable pursuing the following objectives: First and foremost, the number of lecture rooms required to implement the timetable should be a minimum (because the department wishes to minimise the cost of infrastructure associated with its timetable) and then the department would also like to utilise this minimum number of venues as equally frequently as possible (for example, a single period per week in one of the classrooms, and ten periods per week in another is unacceptable).

Tasks

1. Construct a bipartite graph of order 20, called the *lecture graph*, where one partite set has cardinality 4 and represents the four lecturers, where the other partite set has cardinality 16 and represents the courses offered, and where the edge set indicates which lecturer presents which courses.
2. A class timetable may be constructed by finding an edge colouring of the lecture graph in which the colours of edges incident with the four first-year subjects are distinct, and similarly for the second-, third- and fourth-year subjects. The edge colours in such a colouring represent different periods of the timetable in the sense that all edges coloured with colour 1 indicate courses (given by specific lecturers) that should be scheduled for lectures during time slot 1, and similarly for the other colours. Since no vertex representing a lecturer will have more than one edge with the same colour incident with it, the timetable will be clash-free with respect to the lecturers'

teaching responsibilities, and since the vertices representing courses offered during a particular year of study are incident with edges of distinct colours, the timetable will be clash-free with respect to courses that may be taken in conjunction with one another (*i.e.* simultaneously, by the same students). The department's objectives may be achieved by finding an edge colouring of the lecture graph using the minimum number of colours, and utilising the colours as equally often as possible. Find such an optimal edge colouring of the lecture graph in [Task 1](#).

3. Prove that your edge colouring in [Task 2](#) is indeed optimal, *i.e.* that the number of colours in your edge colouring of the lecture graph is as small as possible, and that the colours are utilised as equally often as possible.

Further reading

- [1] American Mathematical Society, *Fermat's last theorem*, URL: <http://www.ams.org/new-in-math/fermat.html>.
- [2] K Appel, W Haken and J Koch, 1977. *Every planar map is four-colourable*, Illinois Journal of Mathematics, **21**, pp. 429–567.
- [3] R Balakrishnan and K Ranganathan, 2000. *A Textbook of Graph Theory*, Springer-Verlag, New York (NY), Chapter VII.
- [4] G Benadé, I Broere, B Jonck and M Frick, 1996. *Uniquely $(m, k)^\tau$ -colourable graphs and k - τ -saturated graphs*, Discrete Mathematics, **162**, pp. 13–22.
- [5] B Bollobás, 1998. *Modern Graph Theory*, Springer-Verlag, New York (NY).
- [6] J Bradbury, *Four-colour problem*, URL: <http://www.cs.queensu.ca/~bradbury>.
- [7] D Brélaz, 1979. *New methods to colour the vertices of a graph*, Communications of the Association for Computing Machinery, **22**, pp. 251–256.
- [8] I Broere and M Frick, 1985. *On the order of colour critical graphs*, Congressus Numerantium, **47**, pp. 125–130.
- [9] I Broere and M Frick, 1990. *On the order of uniquely (k, m) -colourable graphs*, Discrete Mathematics, **82**, pp. 225–232.
- [10] I Broere and M Frick, 1990. *Two results on generalized chromatic numbers*, Quaestiones Mathematicae, **13**, pp. 183–190.
- [11] I Broere and M Frick, 1991. *A characterization of the sequence of generalized chromatic numbers of a graph*, pp. 179–186 in Y Alavi (Ed), *Graph Theory, Combinatorics and Applications: Proceedings of the Sixth Quadrennial International Conference on the Theory and Applications of Graphs*, John Wiley & Sons, New York (NY).
- [12] RL Brooks, 1941. *On colouring the nodes of a network*, Proceedings of the Cambridge Philosophical Society, **37**, pp. 194–197.
- [13] JR Brown, 1972. *Chromatic scheduling and the chromatic number problem*, Management Science, **19(4)**, pp. 456–463.
- [14] A Cayley, 1878. *Open problem*, Proceedings of the London Mathematical Society, **9**, p. 148.

- [15] A Cayley, 1879. *On the colouring of maps*, Proceedings of the Royal Geographical Society (New Series), **1**, pp. 259–261.
- [16] G Chartrand, DP Geller and ST Hedetniemi, 1968. *A generalization of the chromatic number*, Proceedings of the Cambridge Philosophical Society, **64**, pp. 265–271.
- [17] G Chartrand and J Mitchem, 1971. *Graphical theorems of the Nordhaus-Gaddum class*, pp. 55–61 in M Capobianco, JB Frechen and M Krolik (Eds), *Recent Trends in Graph Theory*, Springer, Berlin.
- [18] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY), Chapter 7.
- [19] M Chudnovsky, N Robertson, P Seymour and R Thomas, 2006. *The strong perfect graph theorem*, Annals of Mathematics, **164**(1), pp. 51–229.
- [20] R Crandall and C Pomerance, 2001. *Prime Numbers: A Computational Perspective*, Springer-Verlag, New York (NY).
- [21] B Descartes, 1947. *A three-colour problem*, Eureka, **9**, p. 21.
- [22] P Erdős and RJ Wilson, 1977. *On the chromatic index of almost all graphs*, Journal of Combinatorial Theory, **23**(2), pp. 255–257.
- [23] JR Evans and E Minieka, 1992. *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, New York (NY), Chapter 8.
- [24] LR Foulds, 1992. *Graph Theory Applications*, Springer-Verlag, New York (NY).
- [25] M Frick, 1986. *Generalised colourings of graphs*, PhD Dissertation, Rand Afrikaans University, Johannesburg, South Africa.
- [26] MR Garey and DS Johnson, 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*, WH Freeman, New York (NY).
- [27] A Gibbons, 1985. *Algorithmic Graph Theory*, Cambridge University Press, Cambridge.
- [28] R Gould, 1988. *Graph Theory*, The Benjamin/Cummings Publishing Company, Menlo Park.
- [29] RP Gupta, 1966. *The chromatic index and the degree of a graph*, Notices of the American Mathematical Society, **13**, p. 719.
- [30] PJ Heawood, 1890. *Map-colour theorem*, Quarterly Journal of Pure and Applied Mathematics, **24**, pp. 332–338.
- [31] H Heesch, 1969. *Untersuchungen zum Vierfarben-problem*, B-I-Hochschulskripten, Bibliographisches Institut, Mannheim.
- [32] F Herrmann and A Hertz, 2002. *Finding the chromatic number by means of critical graphs*, ACM Journal of Experimental Algorithms, **7**, pp. 1–12.
- [33] D Hilbert, 1902. *Mathematical problems*, Bulletin of the American Mathematical Society, Lecture delivered at the International Congress of Mathematicians in Paris in 1900.
- [34] I Holyer, 1981. *The NP-completeness of edge coloring*, SIAM Journal on Computing, **10**, pp. 718–720.

- [35] TR Jensen and B Toft, 1995. *Graph Colouring Problems*, John Wiley & Sons, New York (NY).
- [36] G Johns and F Saba, 1989. *On the path chromatic number of a graph*, Annals of the New York Academy of Sciences, **576**, pp. 275–280.
- [37] RM Karp, 1972. *Reducibility among combinatorial problems*, pp. 85–103 in RE Miller and JW Thatcher (Eds), *Complexity of Computer Computations*, Plenum, New York (NY).
- [38] JB Kelly and LM Kelly, 1954. *Paths and circuits in critical graphs*, American Journal of Mathematics, **76**, pp. 786–792.
- [39] AB Kempe, 1879. *On the geographical problems of four colours*, American Journal of Mathematics, **2**, pp. 193–200.
- [40] D König, 1916. *Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre*, Mathematische Annalen, **77**, pp. 453–465.
- [41] R Kumanduri and C Romero, 1998. *Number Theory with Computer Applications*, Prentice Hall, Upper Saddle River (NJ).
- [42] DW Matula, G Marble and JD Isaacson, 1972. *Graph colouring algorithms*, pp. 109–122 in RC Read (Ed), *Graph Theory and Computing*, Academic Press, New York (NY).
- [43] J Mycielski, 1955. *Sur le coloriage des graphes*, Colloquium Mathematicum, **3**, pp. 161–162.
- [44] I Nieuwoudt and JH van Vuuren, 2012. *Algorithms for a shared resource scheduling problem in which some level of conflict is tolerable*, Journal of Scheduling, **15(6)**, pp. 681–702.
- [45] EA Nordhaus and JW Gaddum, 1956. *On complementary graphs*, The American Mathematical Monthly, **63**, pp. 175–177.
- [46] JJ O'Connor and EF Robertson, *The four-colour theorem*, URL: http://www-groups.dcs.st-and.ac.uk/~history/%20HistTopics/The_four-colour_theorem.html.
- [47] O Oellermann, *The four-colour problem and its connection to South African flora*, URL: <http://www.uwinnipeg.ca/~ooellerm/guthrie/FourColor.html>.
- [48] J Petersen, 1891. *Die Theorie der regulären graphs*, Acta Mathematica, **15(1)**, pp. 193–220.
- [49] N Robertson, D Sanders, P Seymour and R Thomas, 1997. *The four-colour theorem*, Journal of Combinatorial Theory, Series B, **70(1)**, pp. 2–44.
- [50] N Robertson, P Seymour and R Thomas, 1993. *Hadwiger's conjecture for K_6 -free graphs*, Combinatorica, **13(3)**, pp. 279–361.
- [51] SM Ross, 2002. *A First Course in Probability*, Sixth edition, Prentice Hall, Upper Saddle River (NJ).
- [52] H Sachs, 1969. *Finite graphs*, pp. 175–184 in *Recent Progress in Combinatorics, Proceedings of the Third Waterloo Conference on Combinatorics*, Academic Press, New York (NY).

- [53] R Thomas, *The four-colour theorem*, URL: <http://www.math.gatech.edu/~thomas/FC/fourcolor.html>.
- [54] VG Vizing, 1964. *On an estimate of the chromatic class of p -graphs*, Diskret. Analiz. **3**, pp. 25–30.
- [55] WD Wallis, 2000. *A Beginner's Guide to Graph Theory*, Birkhäuser, Boston (MA), Chapter 7.
- [56] DJA Welsh and MB Powell, 1967. *An upper bound for the chromatic number of a graph and its application to timetabling problems*, The Computer Journal, **10**, pp. 85–87.
- [57] DB West, 1996. *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River (NJ), §2.4.
- [58] R Wilson, 2002. *Four Colours Suffice — How the Map Problem was Solved*, Penguin Books, London.
- [59] D Woodall, 1990. *Improper colourings of graphs*, pp. 45–63 in R Nelson and RJ Wilson (Eds), *Graph Colourings*, Longman Scientific and Technical, Harlow.
- [60] AA Zykov, 1949. *On some properties of linear complexes (Russian)*, Mat. Sbornik, **24**, pp. 163–188.



Oriented graphs

Contents

15.1	Introduction	501
15.2	Strong orientations	501
15.3	Tournaments	503
15.4	Higher-order rankings in tournaments	506
15.5	Almost traceable and almost hamiltonian orient digraphs	512
	Exercises	517
	Computer exercises	518
	Projects	520
	Further reading	523

15.1 Introduction

A digraph D is **symmetric** if, whenever (u, v) is an arc of D , then (v, u) is also an arc of D . Studying symmetric digraphs is, of course, essentially the same as studying (undirected) graphs. We therefore rather focus our attention on other types of digraphs that occur frequently. Of special interest is so-called asymmetric digraphs, also called oriented graphs, which is the topic of this chapter.

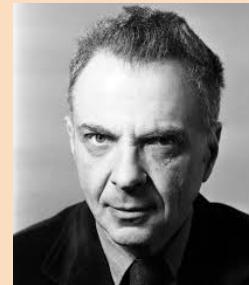
An **asymmetric digraph** or **oriented graph** D is a digraph that can be obtained from a graph G by assigning a direction to (that is, orienting) each edge of G . The resulting digraph D is called an **orientation** of G . Therefore, if D is an oriented graph, then at most one of (u, v) and (v, u) is an arc of D for every pair of distinct vertices u and v of D .

15.2 Strong orientations

Recall from Section 2.5 that a digraph D is **strong** if, for every pair u and v of distinct vertices, D contains both a u - v path and a v - u path. A **strong orientation** of G is an orientation of G that produces a strong digraph.

As a hypothetical application of oriented graphs, consider the following traffic flow problem. Suppose that traffic congestion necessitates a town council to convert all the streets (currently two-way streets) in the town to one-way streets. This can

The American mathematician and statistician **Herbert Ellis Robbins** was born on 12 January 1915 in New Castle, Pennsylvania. He obtained a doctorate in mathematics from Harvard University in 1938 under the supervision of [Hassler Whitney](#) and started his academic career as an instructor at New York University during the period 1939–1941. After World War II, he taught at the University of North Carolina at Chapel Hill from 1946 to 1952. In 1953, he accepted a position as professor of mathematical statistics at Columbia University where he remained until his retirement in 1985. He was the co-author, with Richard Courant, of the popular book entitled *What is Mathematics?* His research interests were in topology, measure theory, statistics, and a variety of other fields, including graph theory. In graph theory, he is best known for [Theorem 15.1](#) which states that graphs that have strong orientations are exactly the 2-edge-connected graphs. He died on 12 February, 2001.



Biographic note 64: Herbert Robbins (1915–2001)

clearly be done by orienting the edges of the graph corresponding to the two-way street system. In the resulting oriented graph it must, however, be possible to travel from any location to any other location. This leads to the following question: Which graphs have strong orientations? Two obvious necessary conditions for a graph to have a strong orientation are that it must be connected and must be bridgeless. [Robbins \[8\]](#) showed that these two conditions are, in fact, also sufficient.

Theorem 15.1 ([8]) *A nontrivial graph has a strong orientation if and only if it is connected and contains no bridge.*

Proof Let G be a nontrivial graph. As observed earlier, if G has a strong orientation, then it is connected and contains no bridge. To prove the sufficiency, suppose that G is a connected, bridgeless graph, but assume, to the contrary, that G has no strong orientation. Since G contains no bridge, every edge of G is a cycle edge. In particular, G contains at least one cycle $C : v_1 v_2 \dots v_k v_1$. If we direct the edges $v_1 v_2, v_2 v_3, \dots, v_{k-1} v_k, v_k v_1$ as $(v_1, v_2), (v_2, v_3), \dots, (v_{k-1}, v_k), (v_k, v_1)$ we produce a directed cycle C . Hence the subgraph C of G has a strong orientation. Among all subgraphs of G , let H be one of maximum order that has a strong orientation. Thus, the order of H is at least the order of a maximum cycle in G , and so $|V(H)| \geq 3$. If $|V(H)| = |V(G)|$, then we can extend any strong orientation of H to a strong orientation of G by orienting the edges in $E(G) \setminus E(H)$ arbitrarily. This contradicts our assumption that G has no strong orientation. Hence, $3 \leq |V(H)| < |V(G)|$.

Since G is connected, there must be an edge $e = v_1 v_2$ joining a vertex $v_1 \in V(H)$ and a vertex $v_2 \in V(G) \setminus V(H)$. Since G has no bridge, there is a cycle $C : v_1 v_2 \dots v_k v_1$ containing the edge e . Let v_t be the first vertex on C following v_1 that belongs to $V(H)$. Then, $3 \leq t \leq k$. Let P be the v_1 - v_t path on C that contains the vertex v_2 . Then the ends of P belong to H , while no internal vertex of P belongs to H . Let F be the subgraph of G obtained from H by adding to it the path P between the vertices v_1 and v_t . As observed, no internal vertex of the added path P

belongs to H . We can extend any strong orientation of H to an orientation of H' (say) by orienting the edges of P from v_1 to v_t .

We claim that this orientation of H' is strong. Let u, v be a distinct pair of vertices in H' . We show that there is both a $u-v$ path and a $v-u$ path in our orientation of H' . If both u and v are in $V(H)$, then this is immediate since H is strong. Suppose, therefore, that exactly one of u or v is in $V(H)$. Without loss of generality, we may assume that $u \in V(H)$ and that $v \in V(P) \setminus V(H)$. Since H is strong, there is a (directed) $u-v_1$ path in H . Extending this path by following the (directed) subpath of P from v_1 to v produces a $u-v$ path in H' . Since H is strong, there is a (directed) v_t-u path in H . Extending the (directed) subpath of P from v to v_t by following this v_t-u path in H produces a $v-u$ path in H' . Thus, if exactly one of u or v is in $V(H)$, then there is both an $u-v$ path and a $v-u$ path in our orientation of H' . Similarly, if both u and v are internal vertices of P , then there is both a $u-v$ path and a $v-u$ path in our orientation of H' . Our orientation of H' is therefore strong. This contradicts the choice of H , however. We deduce, therefore, that G has a strong orientation. ■

❖ The reader should now be able to attempt Exercises 15.1–15.5.

15.3 Tournaments

A **tournament** is an orientation of a complete graph. Thus, a tournament T is a digraph with the property that for every pair u and v of distinct vertices, exactly one of (u, v) or (v, u) is an arc of T . There is one tournament of order 1 and one of order 2 (up to isomorphism). There are two (nonisomorphic) tournaments of order 3, as shown in Figure 15.1.

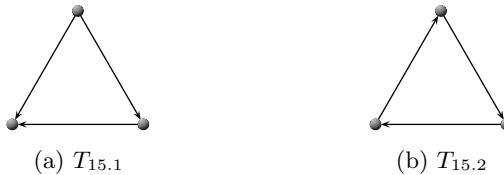


Figure 15.1: Two tournaments of order 3.

Tournaments are the class of oriented graphs that have received the most attention in the literature. This popularity stems from the prevalence of round-robin tournaments in various types of sports, where each player (or team) plays every other player (or team) exactly once and ties are not permitted. The results of such a round-robin tournament with n players can be modelled by a tournament T on n vertices, where the vertices represent the n players and an arc (u, v) indicates that player u defeated player v in the tournament. The number of victories by a player v is therefore the number $\text{od}(v)$. For this reason, the outdegree of a vertex v in a tournament is called the **score of v** .

A tournament may not have a clear overall “winner” since there may be more than one vertex with maximum outdegree. The following result, however, shows that although there need not be a clear winner, it is always possible to find a player x such that, for every other player y , x either beats y or else x beats some

player who beats y . Such a player x is called a **king**. Thus, if x is a king in a tournament T , then $d(x, y) \leq 2$ for every vertex y in T .

Theorem 15.2 *Every tournament has a king.*

Proof Let T be a tournament and let v be a vertex of maximum score in T . Thus, $\text{od}(v) \geq \text{od}(u)$ for every vertex $u \in V(T)$. Let S be the set of all vertices of T adjacent from v . Thus, for each $w \in S$, (v, w) is an arc of T and $d(v, w) = 1$. On the one hand, if $V(T) = S \cup \{v\}$, then $d(v, w) \leq 1$ for every $w \in V(T)$. On the other hand, suppose $U = V(T) \setminus (S \cup \{v\}) \neq \emptyset$ and let $u \in U$. Then, (u, v) is an arc of T . If u is adjacent to all of the vertices in S , then since u is also adjacent to v , $\text{od}(u) \geq |S| + |\{v\}| = \text{od}(v) + 1$, contradicting our choice of v . Hence, u is adjacent from some vertex in S . It follows that $d(v, u) = 2$. Hence, $d(v, w) \leq 2$ for every $w \in V(T)$. In both cases, v is a king in T . ■

Theorem 15.2 was first discovered by the sociologist Landau [4] when studying pecking orders among chickens. In this context, Theorem 15.2 claims that among any group of chickens, a chicken x that pecks the largest number of chickens dominates the group in the sense that for every other chicken y , x either pecks y or x pecks some chicken that pecks y .

As a consequence of the proof of Theorem 15.2, we have the following result.

Corollary 15.3 *Every vertex of maximum score in a tournament is a king.*

A tournament T is furthermore **transitive** if, whenever (u, v) and (v, w) are arcs of T , then (u, w) is also an arc of T . The tournament $T_{15.1}$ in Figure 15.1 is a transitive tournament, while the tournament $T_{15.2}$ in the same figure is not a transitive tournament. The following result characterises those tournaments that are transitive in terms of the scores of vertices in the tournament.

Theorem 15.4 *A tournament T is transitive if and only if every two vertices of T have distinct scores.*

Proof First we consider the necessity. Suppose that T is a transitive tournament. Let u and v be two distinct vertices of T . Without loss of generality, we may assume that (u, v) is an arc of T . Let S be the set of all vertices of T adjacent from v . Thus, for each $w \in S$, (v, w) is an arc. Since T is transitive, (u, w) is also an arc of T for each $w \in S$. Therefore, $\text{od}(u) \geq |S| + |\{v\}| = \text{od}(v) + 1$ and so every two vertices of T have distinct scores.

Next, we show the sufficiency. Suppose that every two vertices of T have distinct scores. Renaming vertices if necessary, we may assume that the score of v_i is $i - 1$ for all $i \in [n]$. Therefore, v_n is adjacent to every other vertex of T . The vertex v_{n-1} is adjacent to every other vertex of T except for v_n . The vertex v_{n-2} is adjacent to every other vertex of T except for v_{n-1} and v_n , and so on. Thus, $E(T) = \{(v_i, v_j) \mid 1 \leq j < i \leq n\}$. We show that T is transitive. Let (v_i, v_j) and (v_j, v_k) be arcs of T . Then, $i > j > k$. Since $i > k$, (v_i, v_k) is an arc of T . Hence, T is a transitive tournament. ■

A sequence s_1, s_2, \dots, s_n of nonnegative integers is called a **score sequence** if there exists a tournament T of order n whose vertices can be labelled v_1, v_2, \dots, v_n such that the score of v_i is s_i ; that is, $\text{od}(v_i) = s_i$ for all $i \in [n]$. As an immediate consequence of Theorem 15.4, we know precisely which sequences are score sequences of transitive tournaments.

Corollary 15.5 *A nondecreasing sequence \mathcal{S} of n nonnegative integers is a score sequence of a transitive tournament if and only if \mathcal{S} is the sequence $0, 1, \dots, n - 1$.*

As a further consequence of [Theorem 15.4](#), we have the following result.

Corollary 15.6 *For every positive integer n , there is precisely one transitive tournament of order n (up to isomorphism).*

We establish next a property of tournaments that is unique to transitive tournaments.

Theorem 15.7 *A tournament is transitive if and only if it contains no (directed) cycles.*

Proof Let T be a transitive tournament and assume, to the contrary, that T contains a cycle, say $C : v_1 v_2 \cdots v_k v_1$. Since (v_1, v_2) and (v_2, v_3) are arcs of T , so is (v_1, v_3) . Since (v_1, v_3) and (v_3, v_4) are arcs of T , (v_1, v_4) is similarly also an arc of T . Continuing in this way, (v_1, v_i) is an arc of T for all $i \in \{2, \dots, k\}$. This, however, contradicts the fact that (v_k, v_1) is an arc of T . We conclude that T contains no cycles.

For the converse, suppose T is a tournament that contains no cycles. Let (u, v) and (v, w) be arcs of T . Since T contains no cycles, (w, u) is not an arc of T . Thus, (u, w) is an arc of T , and so T is transitive. ■

Recall that a (directed) path in a digraph D which contains all vertices of D is called a **hamiltonian path**. We show next that every tournament contains a hamiltonian path.

Theorem 15.8 ([7]) *Every tournament contains a (directed) hamiltonian path.*

Proof Let $P : v_1 v_2 \cdots v_k$ be a longest (directed) path in a tournament T of order n . Assume, to the contrary, that P is not a hamiltonian path. Let v be a vertex of T not on P . Since P is a longest path, neither (v, v_1) nor (v_k, v) are arcs of T . Hence, (v_1, v) and (v, v_k) are arcs of T . This implies that there is an integer $i \in [k - 1]$, such that both (v_i, v) and (v, v_{i+1}) are arcs of T . Replacing the path $v_i v_{i+1}$ of length 1 on P with the path $v_i v v_{i+1}$ of length 2 produces a path of length exceeding that of P , contradicting our choice of P . We deduce, therefore, that P is a hamiltonian path. ■

Recall, from [Section 2.5](#), that a **k -cycle** of a digraph is a cycle of length k . A digraph D of order $n \geq 3$ is said to be **vertex-pancyclic** if each vertex of D is contained in a k -cycle for every $k \in \{3, \dots, n\}$.

Theorem 15.9 ([5]) *Every nontrivial strong tournament is vertex-pancyclic.*

Proof Let T be a strong tournament of order $n \geq 3$, and let v be a vertex of T . We show that v is contained in a k -cycle for every $k \in \{3, \dots, n\}$. We proceed by induction on k .

First we consider the case when $k = 3$. Since T is strong, $\text{od}(v) > 0$ and $\text{id}(v) > 0$. Thus, both the sets $N^+(v)$ and $N^-(v)$ are nonempty. Since T is strong, there is a path from v to each vertex in $N^-(v)$. Every such path contains an arc (u, w) for some $u \in N^+(v)$ and $w \in N^-(v)$. Hence, $v u w v$ is a 3-cycle containing v . This establishes the base case.

Assume, as induction hypothesis, that $3 \leq k < n$ and that v lies on a k -cycle of T . We show that v also lies on a $(k+1)$ -cycle of T . Let $C : v = v_0 v_1 \cdots v_{k-1} v_0$ be a k -cycle of T containing v .

On the one hand, suppose that there is a vertex u not on C that is adjacent to at least one vertex of C and adjacent from at least one vertex of C . This implies that there is an integer $i \in \{0, 1, \dots, k-1\}$, such that both (v_i, v) and (v, v_{i+1}) are arcs of T (where all subscripts are expressed modulo k). Replacing the path $v_i v_{i+1}$ of length 1 on C with the path $v_i v v_{i+1}$ of length 2, however, produces a $(k+1)$ -cycle that contains v , as desired.

On the other hand, suppose that every vertex not on C is either adjacent to every vertex of C or is adjacent from every vertex of C . Let A denote the set of all vertices not on C that are adjacent to every vertex of C , and let B denote the set of all vertices not on C that are adjacent from every vertex of C . Then, $V(T) = A \cup B \cup V(C)$. If $A = \emptyset$, then there would be no path from a vertex in B to a vertex in C . If $B = \emptyset$, then there would be no path from a vertex in C to a vertex in A . Both cases contradict the fact that T is strong. Hence, $|A| \geq 1$ and $|B| \geq 1$. Since T is strong, there is a path from each vertex $v \in V(C)$ to each vertex in A . Every such path contains an arc (b, a) for some $b \in B$ and $a \in A$. Hence, $a v = v_0 v_1 \cdots v_{k-1} b a$ is a $(k+1)$ -cycle containing v , as desired. ■

Recall, from Chapter 11, that a cycle in a digraph D containing all vertices of D is called a **hamiltonian cycle**. A digraph that contains a hamiltonian cycle is called **hamiltonian**. As a consequence of Theorem 15.9, we have a characterisation of those tournaments that are strong.

Corollary 15.10 *A tournament of order at least 3 is strong if and only if it is hamiltonian.*

Proof Let T be a tournament of order $n \geq 3$. By Theorem 15.9, every vertex of T is contained in a k -cycle for every $k \in \{3, \dots, n\}$. In particular, every vertex belongs to an n -cycle. Thus, T is hamiltonian. This establishes the necessity. To prove the sufficiency, suppose that T is hamiltonian. Then, T contains a spanning subdigraph, namely a hamiltonian cycle of T , that is strong, and therefore T is itself strong. ■

❖ The reader should now be able to attempt Exercises 15.6–15.10.

15.4 Higher-order rankings in tournaments

In the previous section, a method was suggested for allocating scores to players in a round-robin sports tournament modelled by a tournament T of order n , based on the outdegrees of the vertices of T modelling these players. This method involved counting the number of players beaten by a particular player during the tournament, and gave rise to a score vector, $\underline{\mathbf{t}}^{(1)}$ (say). Hence, the score of player v_i (*i.e.* the i -th entry of $\underline{\mathbf{t}}^{(1)}$) is the outdegree or number of directed walks of length 1 in T , originating at the vertex v_i . It is clear that the i -th entry of $\underline{\mathbf{t}}^{(1)}$ is the i -th row sum of the $n \times n$ adjacency matrix \mathbf{A} of T , *i.e.*

$$\underline{\mathbf{t}}^{(1)} = \mathbf{A}\underline{\mathbf{e}}, \quad (15.1)$$

where \underline{e} is an n -column vector containing only ones. This method of scoring players may be thought of as a first-order method of scoring (hence the superscript ⁽¹⁾ in (15.1)), because one only considers the *number* of players that were beaten by a particular player in order to allocate a score to that player, irrespective of *how good* the players are that were beaten by the player in question. One may, however, argue that beating two poor players is not necessarily better than beating one good player.

But then the question arises how one might measure *how good* the players are that are beaten by a particular player during a tournament. One way of achieving such a measurement is to count the number of players beaten by a particular player, who themselves have beaten at least one other player. This gives rise to a second-order score vector, $\underline{t}^{(2)}$ (say). In this case the second-order score of player v_i (*i.e.* the i -th entry of $\underline{t}^{(2)}$) is the number of directed walks of length 2 in T , originating at the vertex v_i .

One may continue this process by rather counting the number of players beaten by a particular player, who themselves have beaten at least one other player who, in turn, also has beaten at least one player. This gives rise to a third-order score vector, $\underline{t}^{(3)}$ (say), whose i -th entry is the number of directed walks of length 3 in T , originating at the vertex v_i . As k increases, more and more information is incorporated into the score vector $\underline{t}^{(k)}$ and it makes sense to wonder about the limit

$$\underline{\Phi} = \lim_{k \rightarrow \infty} \underline{t}^{(k)}, \quad (15.2)$$

which incorporates as much information as possible into the score vector. The question arises how to evaluate $\underline{t}^{(k)}$ efficiently and how to use such information to estimate $\underline{\Phi}$ in (15.2). The following theorem provides a method for such evaluations.

Theorem 15.11 *Let \mathbf{A} be the adjacency matrix of a tournament T . Then the entry in row i and column j of the matrix power \mathbf{A}^k is the number of directed v_i - v_j walks of length k in T .*

Proof We proceed by induction on k . Observe, as base case, that the entry in row i and column j of the matrix \mathbf{A} is 1 if the arc (v_i, v_j) , which is a directed v_i - v_j walk of length 1, is present in T , or 0 otherwise. Because there can be at most one v_i - v_j walk of length 1 in T , the result is certainly true for $k = 1$.

Assume, as induction hypothesis, that the theorem holds for $k = m$, *i.e.* that the entry $a_{ij}^{(m)}$ in row i and column j of the matrix power \mathbf{A}^m is the number of directed v_i - v_j walks of length m in T . Then, because $\mathbf{A}^{m+1} = \mathbf{A}^m \mathbf{A}$, the entry in row i and column j of the matrix power \mathbf{A}^{m+1} is given by

$$a_{ij}^{(m+1)} = \sum_{\ell=1}^n a_{i\ell}^{(m)} a_{\ell j}, \quad (15.3)$$

where n is the order of T . But now it follows, by the induction hypothesis, that

$$a_{i\ell}^{(m)} a_{\ell j} = \begin{cases} a_{i\ell}^{(m)} & \text{if } a_{\ell j} = 1 \\ 0 & \text{if } a_{\ell j} = 0 \end{cases}$$

is the number of directed v_i - v_j walks of length $m+1$ with v_ℓ as second last vertex, as illustrated in [Figure 15.2](#). If we consider every vertex as a possible vertex candidate for v_ℓ (*i.e.* if we let ℓ vary over the range $1, \dots, n$) and add together the number of directed v_i - v_j walks of length $m+1$ thus found, we count all directed v_i - v_j walks of length $m+1$, which is exactly the number obtained in [\(15.3\)](#). ■

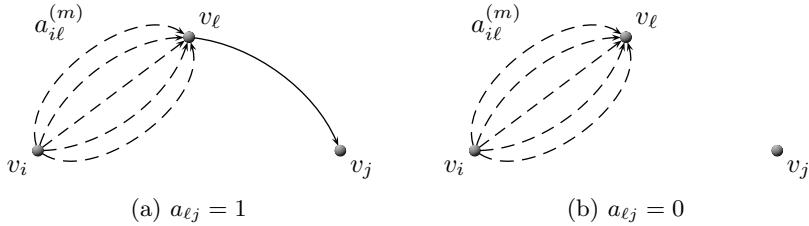


Figure 15.2: The number of directed v_i - v_j walks of length $m+1$ with v_ℓ as second last vertex.

It follows, by [Theorem 15.11](#), that the product $\mathbf{A}^k \underline{\mathbf{e}}$, capturing the row sums of the matrix power \mathbf{A}^k , contains as its i -th entry the number of directed walks of length k originating at the vertex v_i in T . But this is exactly the k -th order score of the player corresponding to the vertex v_i in T , and hence the k -th order score vector of the tournament is given by

$$\underline{\mathbf{t}}^{(k)} = \mathbf{A}^k \underline{\mathbf{e}}, \quad (15.4)$$

which is, in retrospect, an obvious generalisation of [\(15.1\)](#). We are therefore interested in investigating the limiting process

$$\underline{\Phi} = \lim_{k \rightarrow \infty} \mathbf{A}^k \underline{\mathbf{e}}. \quad (15.5)$$

It turns out that a multiplicative scaling of the limit approach rates in [\(15.5\)](#) may be obtained for certain tournaments by utilising techniques from linear algebra, as we show next.

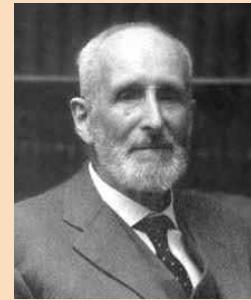
A square matrix \mathbf{A} is **primitive** if there exists a natural number k such that all entries of \mathbf{A}^k are strictly positive. The following theorem from the realm of linear algebra (whose proof falls beyond the scope of this book) is part of a famous result on primitive matrices known as the [Perron-Frobenius Theorem](#). A weaker form of this theorem was proved by [Perron](#) [6] in 1907 for positive matrices, and was extended by [Frobenius](#) [3] in 1912 to primitive matrices. The interested reader may consult [1, Chapter 2] for more information on this theorem.

Theorem 15.12 (Perron-Frobenius Theorem) *If \mathbf{A} is an $n \times n$ primitive, non-negative matrix, then the eigenvalue λ of \mathbf{A} with largest modulus is real and has multiplicity one. Furthermore, if $\underline{\mathbf{e}}$ is an n -column vector containing only ones, then*

$$\lim_{k \rightarrow \infty} \frac{\mathbf{A}^k}{\lambda^k} \underline{\mathbf{e}} = \underline{\mathbf{x}}$$

for some eigenvector $\underline{\mathbf{x}}$ of \mathbf{A} associated with λ , whose entries are all nonzero and have the same sign.

Oskar Perron was born on 7 May 1880 in Frankenthal, Germany. He obtained a doctorate in mathematics in 1902 from the Ludwig-Maximilian University of Munich in Germany. Thereafter, he studied further at Tübingen University and at Göttingen University, where he worked with David Hilbert. In 1910, Perron accepted a position as extraordinary professor at Tübingen University and in 1914, he became an ordinary professor at Heidelberg. After World War I (during which he won an iron cross), Perron returned to Heidelberg, where he remained until 1922 when he took up a chair at Ludwig-Maximilian University in Munich. He formally retired in 1951, but continued to teach certain courses at Munich until 1960. Even after ending his teaching career at the age of 80, he published 18 papers between 1964 and 1973. He wrote numerous books on topics such as continued fractions and algebra, but perhaps most remarkable of all Perron's books was his text on noneuclidean geometry published when he was 82. He was awarded an honorary doctorate from the University of Tübingen in 1956, an honorary doctorate from the University of Mainz in 1960 and the Bavarian Order of Merit in 1959. He died on 22 February 1975 in Munich.



Biographic note 65: Oskar Perron (1880–1975)

The significance of [Theorem 15.12](#) is that it may be used to gain information about the limiting process in [\(15.5\)](#) and, in particular, its componentwise limiting value approach rates in the case of strong tournaments. In order to invoke this theorem, let us first prove that the adjacency matrix of any strong tournament is a primitive matrix.

Theorem 15.13 *If A is the adjacency matrix of a strong tournament T of order $n \geq 5$, then $A^{\text{diam}(T)+3} > \mathbf{0}$.*

Proof We show that there exists a directed v_i - v_j walk of length $\text{diam}(T) + 3$ for every pair of vertices v_i and v_j in T (even when $i = j$).

Suppose $d(v_i, v_j) = d_{ij}$. Then there exists a directed v_i - v_j path P in T of length d_{ij} . The idea behind the proof is to use the vertex-pancyclic property of T according to [Theorem 15.9](#) in order to append the path P with cycles of appropriate length so as to produce a directed v_i - v_j walk of length $\text{diam}(T) + 3$.

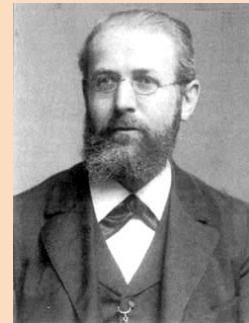
Clearly, $0 \leq d_{ij} \leq \text{diam}(T) \leq n - 1$. The directed cycles appended to P must have a combined length of $\text{diam}(T) + 3 - d_{ij}$. We know that

$$3 \leq \text{diam}(T) - d_{ij} + 3 \leq (n - 1) + 3 = n + 2.$$

If $\text{diam}(T) - d_{ij} + 3 \leq n$, then it follows by [Theorem 15.9](#) that v_j is contained in a directed cycle $C^{(1)}$ of length $\text{diam}(T) - d_{ij} + 3$. The path P , followed by the cycle $C^{(1)}$ is, therefore, a v_i - v_j walk of length $\text{diam}(T) + 3$, as desired.

Suppose now that $\text{diam}(T) - d_{ij} + 3 = n + 1$. Since $n \geq 5$, it follows, again by [Theorem 15.9](#), that v_j is contained in a directed $(n - 2)$ -cycle $C^{(2)}$ as well as in a directed 3-cycle $C^{(3)}$. The path P , followed by the cycle $C^{(2)}$, followed by the cycle $C^{(3)}$ is, therefore, a v_i - v_j walk of length $d_{ij} + (n - 2) + 3 = \text{diam}(T) + 3$, as desired.

Ferdinand Georg Frobenius was born on 26 October 1849 in Charlottenburg, Germany. In 1870, he received a doctorate in mathematics (differential equations) from the University of Berlin under the supervision of Karl Weierstrass. After teaching in Berlin until 1874, he took up a position as professor of mathematics at the Eidgenössische Polytechnikum in Zürich where he remained for seventeen years. In 1893, he returned to Berlin, where he succeeded Leopold Kronecker as professor of mathematics and was elected to the Prussian Academy of Sciences. He made a large number of important contributions in both group theory and number theory. He also supervised several doctoral students who themselves later made important contributions in mathematics, including [Edmund Landau](#) (1899), Issai Schur (1901) and Robert Remak (1910). He died on 3 August 1917 in Berlin.



Biographic note 66: Ferdinand Frobenius (1849–1917)

Finally, suppose $\text{diam}(T) - d_{ij} + 3 = n + 2$. Since $n \geq 5$, it follows, yet again by [Theorem 15.9](#), that v_j is contained in a directed $(n - 1)$ -cycle $C^{(4)}$ as well as in a directed 3-cycle $C^{(3)}$. The path P , followed by the cycle $C^{(3)}$, followed by the cycle $C^{(4)}$ is, therefore, a v_i - v_j walk of length $d_{ij} + (n - 1) + 3 = \text{diam}(T) + 3$, as desired. ■

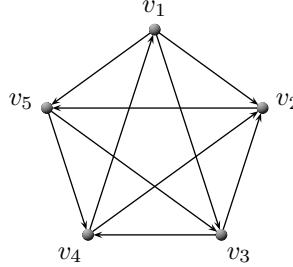
It should be clear to the reader that if T is a strong tournament (which means, by [Theorem 15.13](#), that the adjacency matrix \mathbf{A} of T is primitive), then the limit [\(15.5\)](#) does not exist (why?). We would, however, still like to be able to evaluate the relative rates at which the entries of $\underline{\Phi}$ in [\(15.5\)](#) approach infinity. Note that our inability to determine the particular multiplicative scaling of the limit approach rates in [Theorem 15.12](#) for strong tournaments T (*i.e.* which of the infinitude of eigenvectors associated with the eigenvalue λ of largest modulus to choose as \underline{x}) does not present a problem, since we are only interested in the *relative* approach rates of the limiting process in [\(15.5\)](#).

Let \mathbf{A} be the adjacency matrix of a strong tournament of order n . Then it follows, by Theorems [15.11](#) and [15.12](#), that the eigenvalue, λ , of \mathbf{A} with largest modulus is real and has multiplicity one. Dividing the expressions in [\(15.4\)](#) and [\(15.5\)](#) by λ^k and taking limits as $k \rightarrow \infty$ on both sides yields the identity

$$\lim_{k \rightarrow \infty} \frac{\underline{t}^{(k)}}{\lambda^k} = \lim_{k \rightarrow \infty} \frac{\mathbf{A}^k}{\lambda^k} \underline{e} = \underline{x}, \quad (15.6)$$

where \underline{x} is some eigenvector of \mathbf{A} associated with the eigenvalue λ , whose entries are all nonzero and have the same sign according to [Theorem 15.12](#). Because any real multiple of an eigenvector of \mathbf{A} associated with the eigenvalue λ is again an eigenvector of \mathbf{A} associated with λ , it follows by [\(15.6\)](#) that the relative magnitudes of the entries of any eigenvalue \underline{x}' of \mathbf{A} associated with λ captures the relative componentwise approach rates towards infinity in the limiting process [\(15.5\)](#).

Let us demonstrate, by means of an example, how the above eigenvector technique may be used to determine the infinite-order scores for players of the tournament $T_{15.3}$ shown in [Figure 15.3](#).



[Figure 15.3:](#) A tournament $T_{15.3}$ of order 5.

The eccentricities of the vertices in $T_{15.3}$ are $e(v_1) = 2$, $e(v_2) = 3$, $e(v_3) = 2$, $e(v_4) = 2$ and $e(v_5) = 2$. Hence $\text{diam}(T_{15.3}) = 3$, so that $A_{T_{15.3}}^{3+3} > \mathbf{0}$ by [Theorem 15.13](#), where $\mathbf{A}_{T_{15.3}}$ is the adjacency matrix of $T_{15.3}$. The matrix powers $\mathbf{A}_{T_{15.3}}$, $\mathbf{A}_{T_{15.3}}^2$, $\mathbf{A}_{T_{15.3}}^3$, $\mathbf{A}_{T_{15.3}}^4$, $\mathbf{A}_{T_{15.3}}^5$ and $\mathbf{A}_{T_{15.3}}^6$ are shown in [Figure 15.4](#), confirming the above result, and hence that $\mathbf{A}_{T_{15.3}}$ is a primitive matrix.

$$\begin{aligned} \mathbf{A}_{T_{15.3}} &= \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} & \mathbf{A}_{T_{15.3}}^2 &= \begin{bmatrix} 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 2 \\ 1 & 2 & 0 & 1 & 0 \end{bmatrix} \\ \mathbf{A}_{T_{15.3}}^3 &= \begin{bmatrix} 2 & 3 & 1 & 2 & 1 \\ 1 & 2 & 0 & 1 & 0 \\ 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 2 & 3 & 1 \\ 1 & 2 & 1 & 0 & 3 \end{bmatrix} & \mathbf{A}_{T_{15.3}}^4 &= \begin{bmatrix} 2 & 5 & 3 & 2 & 5 \\ 1 & 2 & 1 & 0 & 3 \\ 1 & 3 & 2 & 4 & 1 \\ 3 & 5 & 1 & 3 & 1 \\ 0 & 2 & 4 & 4 & 3 \end{bmatrix} \\ \mathbf{A}_{T_{15.3}}^5 &= \begin{bmatrix} 2 & 7 & 7 & 8 & 7 \\ 0 & 2 & 4 & 4 & 3 \\ 4 & 7 & 2 & 3 & 4 \\ 3 & 7 & 4 & 2 & 8 \\ 4 & 8 & 3 & 7 & 2 \end{bmatrix} & \mathbf{A}_{T_{15.3}}^6 &= \begin{bmatrix} 8 & 17 & 9 & 14 & 9 \\ 4 & 8 & 3 & 7 & 2 \\ 3 & 9 & 8 & 6 & 11 \\ 2 & 9 & 11 & 12 & 10 \\ 7 & 14 & 6 & 5 & 12 \end{bmatrix} \end{aligned}$$

[Figure 15.4:](#) The matrix powers $\mathbf{A}_{T_{15.3}}$, $\mathbf{A}_{T_{15.3}}^2$, $\mathbf{A}_{T_{15.3}}^3$, $\mathbf{A}_{T_{15.3}}^4$, $\mathbf{A}_{T_{15.3}}^5$ and $\mathbf{A}_{T_{15.3}}^6$.

The first six higher-order score vectors of $T_{15.3}$ are given in [Figure 15.5](#). The first-order ranking vector $\underline{t}^{(1)}$, for example, suggests that player v_1 is a clear winner, that players v_3 , v_4 and v_5 are jointly second, and that player v_2 is the clear looser, denoted by the ranking $v_1, \{v_3, v_4, v_5\}, v_2$. In this ranking it is not possible to distinguish between the players v_3 , v_4 and v_5 . In a ranking based on score vectors

of an order higher than one, however, it is indeed possible to distinguish further between these players. For example, the second-order score vector $\underline{t}^{(2)}$ suggests the finer ranking $v_1, \{v_4, v_5\}, v_3, v_2$. In this ranking, players v_4 and v_5 have clearly performed better than v_3 , but it is still not possible to distinguish between the players v_4 and v_5 based on the second-order score vector alone. The question arises whether such a distinction will be possible based on higher-order score vectors?

$$\underline{t}^{(1)} = \mathbf{A}_{T_{15.3}} \underline{e} = [3 \ 1 \ 2 \ 2 \ 2]^T \quad \underline{t}^{(2)} = \mathbf{A}_{T_{15.3}}^2 \underline{e} = [5 \ 2 \ 3 \ 4 \ 4]^T$$

$$\underline{t}^{(3)} = \mathbf{A}_{T_{15.3}}^3 \underline{e} = [9 \ 4 \ 6 \ 7 \ 7]^T \quad \underline{t}^{(4)} = \mathbf{A}_{T_{15.3}}^4 \underline{e} = [17 \ 7 \ 11 \ 13 \ 13]^T$$

$$\underline{t}^{(5)} = \mathbf{A}_{T_{15.3}}^5 \underline{e} = [31 \ 13 \ 20 \ 24 \ 24]^T \quad \underline{t}^{(6)} = \mathbf{A}_{T_{15.3}}^6 \underline{e} = [57 \ 24 \ 37 \ 44 \ 44]^T$$

Figure 15.5: The higher order score vectors $\underline{t}^{(1)}, \underline{t}^{(2)}, \underline{t}^{(3)}, \underline{t}^{(4)}, \underline{t}^{(5)}$ and $\underline{t}^{(6)}$ of the tournament $T_{15.3}$ in Figure 15.5.

The eigenvalues of $\mathbf{A}_{T_{15.3}}$ are $\lambda_1 \approx 1.8393$, $\lambda_{2,3} \approx -0.5 \pm 1.3229i$ and $\lambda_{4,5} \approx -0.4196 \pm 0.6063i$. Hence the moduli of these eigenvalues are $|\lambda_1| \approx 1.8393$, $|\lambda_2| = |\lambda_3| \approx 1.4142$ and $|\lambda_4| = |\lambda_5| \approx 0.7373$, confirming the result of Theorem 15.12 that the eigenvalue λ_1 of $\mathbf{A}_{T_{15.3}}$ with largest modulus is real and has multiplicity one. Eigenvectors of $\mathbf{A}_{T_{15.3}}$ associated with the eigenvalues $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and λ_5 are $\underline{x}_1 \approx [0.5990 \ 0.2514 \ 0.3880 \ 0.4623 \ 0.4623]^T$,

$$\underline{x}_{2,3} \approx \begin{bmatrix} 0.3753 \pm 0.0450i \\ 0.3753 \pm 0.0450i \\ -0.3753 \mp 0.0450i \\ -0.1281 \mp 0.5189i \\ -0.2471 \mp 0.4740i \end{bmatrix} \quad \text{and} \quad \underline{x}_{4,5} \approx \begin{bmatrix} 0.1078 \mp 0.5657i \\ 0.0949 \pm 0.4342i \\ 0.5059 \mp 0.0068i \\ -0.3031 \mp 0.1247i \\ -0.3031 \mp 0.1247i \end{bmatrix},$$

respectively. It therefore follows, by (15.6), that an infinite-order score vector would suggest the ranking $v_1, \{v_4, v_5\}, v_3, v_2$, which is still not able to distinguish between players v_4 and v_5 . In fact, for this example, the limiting ranking is already achieved based on the second-order score vector, but this will, of course, not always be the case.

The above example shows that the eigenvector method of determining higher order rankings for strong tournaments is not perfect, although it often gives a more refined ranking than the outdegree method of the previous section, in the sense that usually it is possible to distinguish between certain players via the eigenvector method who achieved a tie via the outdegree method.

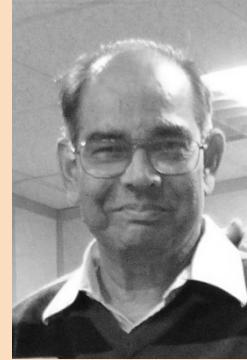
- ❖ The reader should now be able to attempt Exercise 15.11 and Project 15.1.

15.5 Almost traceable and almost hamiltonian oriented digraphs

The reader will recall the discussion on the existence of hypotraceable and hypohamiltonian graphs in Chapter 11. In this section, we briefly consider similar existence questions for oriented graphs instead of graphs and digraphs in general.

An oriented graph G is **hypotraceable** if G does not possess a (directed) hamiltonian path, but deleting any vertex from G results in an oriented graph that does contain a (directed) hamiltonian path. An oriented graph G is similarly **hypohamiltonian** if G is nonhamiltonian, but deleting any vertex from G results in a hamiltonian oriented graph.

Uppaluri Siva Ramachandra Murty (or USR Murty as he prefers to write his name) was born on 23 December 1940. He received a doctorate in mathematics in 1967 from the Indian Statistical Institute in Calcutta, with a thesis on extremal graph theory. His supervisor was the well-known statistician Dr CR Rao. Murty is professor emeritus in the Department of Combinatorics and Optimisation at the University of Waterloo, having worked there since 1967. He is well known for his work in matroid theory and graph theory. He is best known in graph theory circles for his textbook entitled *Graph Theory with Applications* co-authored with Adrian Bondy. Murty has served as a managing editor and co-editor-in-chief of the prestigious Journal of Combinatorial Theory, Series B.



Biographic note 67: USR Murty (1940–present)

The Indian/Canadian mathematician [USR Murty](#) first raised the question of the existence of hypohamiltonian oriented graphs in 1976. An answer came in the affirmative in 1978 when Danish mathematician [Carsten Thomassen](#) proved the following result.

Theorem 15.14 ([10]) *If $a \geq 3$ and $b \geq 3$, then the cartesian product $\vec{C}_a \square \vec{C}_b$ of oriented cycles is a hypohamiltonian oriented graph if $b = ma - 1 \geq 3$ for some $m \geq 1$, or $a = 3$ and $b = 6k + 4$ for some $k \geq 0$.*

Theorem 15.14 implies the existence of infinitely many hypohamiltonian oriented graphs, and in particular of orders 12, 15, 20, 24, 28, 30, 33, 42, 44, 45, 48, ... The smallest of these oriented graphs is shown in Figure 15.6. The theorem makes no claim, however, about the existence or otherwise of hypohamiltonian oriented graphs of the intermittent orders. It is therefore natural to ask: *For which values of n does there exist a hypohamiltonian oriented graph of order n ?*

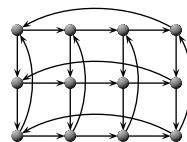


Figure 15.6: The hypohamiltonian oriented graph $\vec{C}_3 \square \vec{C}_4$ of order 12.

This question remained unanswered for thirty six years until German mathematicians Arnfried Kemnitz and [Ingo Schiermeyer](#) called for its resolution in 2014.

Ingo Schiermeyer was born in 1960 in Paderborn, Germany. He obtained a doctorate in mathematics at RWTH Aachen University in 1988, and completed his habilitation in 1993. In 1995, he started his academic career as a professor at the Cottbus campus of the Brandenburg University of Technology in Brandenburg, Germany. In 1999, he accepted a position as professor of applied discrete mathematics in the Institute of Discrete Mathematics and Algebra at the Technische Universität Braunschweig, Freiberg. He has worked in many areas of graph theory, including Ramsey theory, independence in graphs, rainbow connections, and colorings. He is organiser of the annual C5 Graph Theory Workshop on cycles, colorings, cliques, claws and closures.



Biographic note 68: Ingo Schiermeyer (1960–present)

Together with South African mathematicians [Susan van Aardt](#), [Marietjie Frick](#) and [Alewyn Burger](#), they were able to prove the following existence result.

Theorem 15.15 ([12]) *There exists a hypohamiltonian oriented graph of order n if and only if $n \geq 9$.*

The proof of [Theorem 15.15](#) consists of the construction of six infinite classes of hypohamiltonian oriented graphs, as illustrated in [Figure 15.7](#).

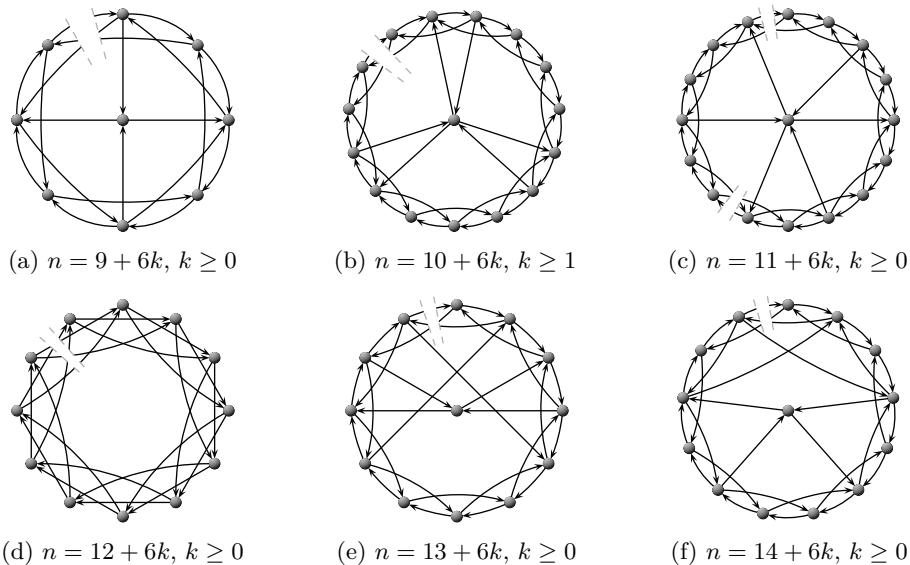


Figure 15.7: Six infinite classes of hypohamiltonian oriented graphs.

Note that the class of graphs in [Figure 15.7\(b\)](#) does not include an oriented graph of order 10. Although such a digraph exists, it does not conform to the

Marietjie Frick was born in Potgietersrus, South Africa, in 1948 and attended school in Johannesburg. She obtained a master's degree in mathematics (group theory) from the Australian National University, and a doctorate in mathematics (graph theory) in 1986 from the then Rand Afrikaans University (now called the University of Johannesburg) in South Africa under the supervision of Izak Broere. The next year she spent at Rand Afrikaans University as a postdoctoral researcher and in 1988, she was appointed as a lecturer at the University of South Africa (UNISA). She is currently an extraordinary professor of mathematics within the Department of Mathematics and Applied Mathematics at the University of Pretoria. Her research interests are in longest paths and cycles in graphs, vertex partition problems, and local properties of graphs. She is well known for her important contributions to the *Path Partition Conjecture* (PPC) on which she has worked for more than two decades. She was the organiser of many successful Salt Rock Graph Theory Workshops that have attracted international graph theorists from various parts of the world, including the United States of America, Canada, Denmark, Germany and Mexico.



Biographic note 69: Marietjie Frick (1948–present)

construction pattern illustrated in the figure. Construction of a hypohamiltonian oriented graph of order 10 will be left as an exercise.

Van Aardt, Frick and Burger also considered the existence of planar hypohamiltonian oriented graphs. They proved that there is no planar hypohamiltonian oriented graph of order $n \leq 8$ and produced the smallest planar hypohamiltonian oriented graph, of order 9, shown in [Figure 15.8](#).

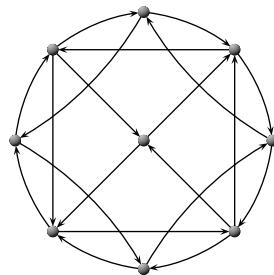


Figure 15.8: A planar hypohamiltonian oriented graph of order 9.

Based on a construction using the digraph in [Figure 15.8](#) as starting point, Van Aardt, Burger and Frick were able to produce an infinite class of planar hypohamiltonian oriented graphs, resulting in the following existence theorem.

Theorem 15.16 ([1]) *There exists a planar hypohamiltonian oriented graph of order $9 + 6k$ for every integer $k \geq 0$.*

Susan van Aardt was born in Paarl, South Africa in 1966 and was schooled in various towns in the Western Cape. She attended Stellenbosch University where she obtained a doctorate in numerical analysis in 1996 and was appointed as lecturer in the Department of Mathematics, Applied Mathematics and Astronomy at the University of South Africa in 1997. Her interest in graph theory was sparked after attending various departmental colloquia on the topic. In 2001, she started collaborating with her colleague [Marietjie Frick](#) on the *Directed Path Partition Conjecture* (DPPC). Her main research interests within the discipline of graph theory are longest paths and cycles in graphs and digraphs, especially hypotraceable and hypohamiltonian oriented graphs. Together with [Marietjie Frick](#) she has co-hosted various graph theory workshops in Salt Rock, KwaZulu-Natal in South Africa.



Biographic note 70: Susan van Aardt (1966–present)

A complete characterisation of orders for which there exist hypohamiltonian oriented graphs remains an open problem, however.

Existence questions related to hypotraceability of oriented graphs have also been considered. In 2007, the Slovak mathematician Peter Katrenič noted the existence of the hypotraceable oriented graph of order 13 in [Figure 15.9\(a\)](#) and asked whether there exist smaller hypotraceable oriented graphs.

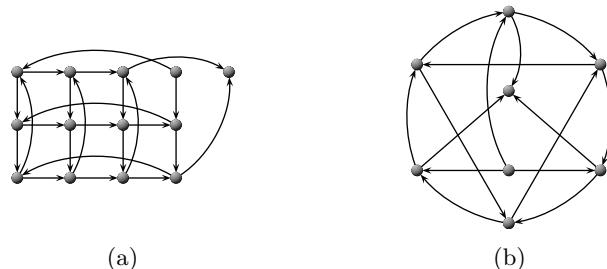


Figure 15.9: (a) A hypotraceable oriented graph of order 13. (b) A hypotraceable oriented graph of order 8.

Together with [Van Aardt](#), Katrenič subsequently found the hypotraceable oriented graph of order 8 in [Figure 15.9\(b\)](#). This prompted them, together with [Frick](#) and Danish mathematician Morten Nielsen, to establish the following existence result in 2011.

Theorem 15.17 ([13]) *There exists a hypotraceable oriented graph of order n for every $n \geq 8$ except possibly for $n = 9, 11$.*

Burger [2] settled the two outstanding cases of [Theorem 15.17](#) by showing that there is no hypotraceable oriented graph of order 9, but that there are thousands of hypotraceable oriented graphs of order 11. We therefore have the following result.

Corollary 15.18 ([2, 13]) *There exists a hypotraceable oriented graph of order n if and only if $n = 8$ or $n \geq 10$.*

❖ The reader should now be able to attempt [Exercise 15.12](#).

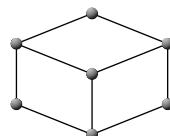
Alewyn Petrus Burger was born just outside Worcester, South Africa on 4 January 1968. He obtained an honours degree in applied mathematics from Stellenbosch University in 1989 as well as master's and doctoral degrees in mathematics from the *University of South Africa* (UNISA) in 1994 and 1999, respectively. Thereafter, he held post-doctoral positions at UNISA (2000–2001), the University of Victoria, Canada (2002–2004) and Stellenbosch University (2005–2007). In 2008, he accepted a research position within the Department of Logistics at Stellenbosch University, focussing on computational aspects of various research topics in graph theory. His interests in graph theory lie within the areas of Ramsey theory, graph domination, traceability of graphs and graph-drawing. He has also worked in design theory (Latin squares and lottery designs). He maintains a wide research network with collaborators in Canada, Denmark, Germany, Mexico and South Africa.



Biographic note 71: Alewyn Burger (1968–present)

Exercises

15.1 Does the graph shown below have a strong orientation? If so, orient its edges so that the resulting digraph is strong.



15.2 Prove or disprove the following statement: A nontrivial graph that has a strong orientation contains no cut-vertex.

15.3 Prove or disprove the following statement: If G is a graph that has a strong orientation, then there is a cycle in G containing any two specified vertices of G .

15.4 Prove that if D is an oriented graph of order 4 such that $D - v$ is strong for every vertex v of D , then D is strong.

15.5 Prove that if D is an oriented graph of order 5 such that $D - u - v$ is strong for every pair u, v of vertices of D , then D is strong.

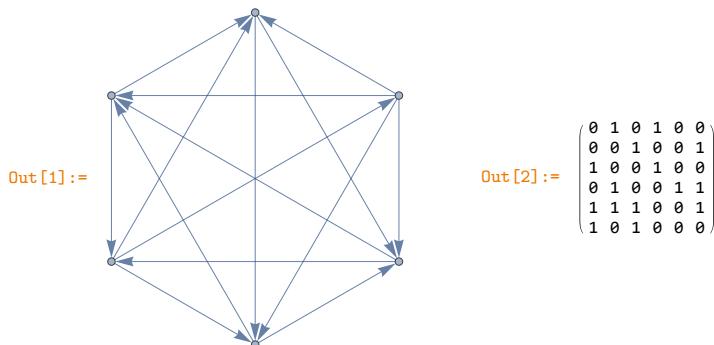
- 15.6 Prove that if u and v are distinct vertices in a tournament T that contains both a $u-v$ path and a $v-u$ path, then $d(u, v) \neq d(v, u)$.
- 15.7 Prove that if the distance from a vertex u to a vertex v in a tournament is $d(u, v) = k$, then $\text{id}(u) \geq k - 1$.
- 15.8 By reversing the direction of an arc (u, v) we mean deleting the arc (u, v) and inserting the arc (v, u) . Prove that if T is a strong tournament of order 5 such that every tournament obtained from T by reversing the direction of any specified arc is strong, then every vertex of T has score 2.
- 15.9 Prove that if v is a vertex of a tournament T of maximum indegree, then $d(u, v) \leq 2$ for every vertex u of T .
- 15.10 Prove that if an even number of players play in a round-robin sports tournament, then it is *not* possible for all players to tie for first place.
- 15.11 Prove that the adjacency matrix of a tournament T of order n is primitive if and only if T is strong and $n \geq 4$.
- 15.12 Construct a hypohamiltonian oriented graph of order 10.

Computer exercises

The **MATHEMATICA** command `DirectedGraph[G, "Random"]` may be used to assign a random orientation to an undirected graph G . For example, the commands

```
In[1]:= K6 = DirectedGraph[CompleteGraph[6], "Random"]
In[2]:= MatrixForm[AdjacencyMatrix[K6]]
```

may produce the output:

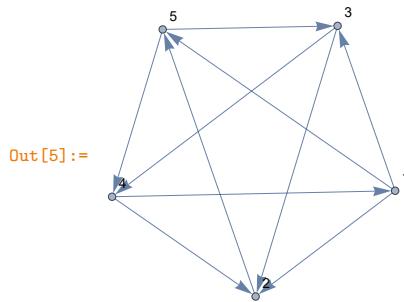


Recall, from Chapter 1, that the tournament $T_{15.3}$ in Figure 15.3 may be generated in **MATHEMATICA** by means of the commands

```
In[3]:= m = {{0, 1, 1, 0, 1}, {0, 0, 0, 0, 1}, {0, 1, 0, 1, 0}, {1, 1, 0, 0, 0},
           {0, 0, 1, 1, 0}};
In[4]:= MatrixForm[m]
In[5]:= G = AdjacencyGraph[m, VertexLabels -> "Name"]
In[6]:= UndirectedGraphQ[G]
```

which produce the output:

$$\text{Out}[4]:= \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$



Out[6]:= False

Recall, also from [Chapter 1](#), that the command `VertexOutDegree[G]` produces a vector containing the outdegrees of vertices of a directed graph G . If G is a tournament, this vector is the first-order score vector of G . Also, the command `MatrixPower[M, n]` returns the matrix M raised to the n -th power. For example, the commands

```
In[7]:= VertexOutDegree[G]
In[8]:= Table[VertexEccentricity[G, v], {v, 1, 5}]
In[9]:= GraphRadius[G]
In[10]:= Diam = GraphDiameter[G]
In[11]:= MatrixForm[MatrixPower[m, Diam + 3]]
```

produce the output:

```
Out[10]:= 3
Out[11]:=  $\begin{pmatrix} 8 & 17 & 9 & 14 & 9 \\ 4 & 8 & 3 & 7 & 2 \\ 3 & 9 & 8 & 6 & 11 \\ 2 & 9 & 11 & 12 & 10 \\ 7 & 14 & 6 & 5 & 12 \end{pmatrix}$ 
Out[12]:= {{1, 2, 3, 4, 5}}
Out[13]:= {{1, 2, 3, 4, 5}}
Out[14]:= True
```

confirming the results following directly after [Figure 15.3](#) and in [Figure 15.2](#) of [Section 15.4](#). Furthermore, recall from [Chapter 2](#), that the **MATHEMATICA** command `ConnectedComponents[G]` returns a list of components in an undirected graph G . If, however, G is directed, then it returns a list of strongly connected components of G . Furthermore, the command `WeaklyConnectedComponents[G]` returns a list of weakly connected components of a directed graph G . Also, recall, from [Chapter 11](#), that the command `HamiltonianGraphQ[G]` yields the boolean value **True** if G is a hamiltonian graph or digraph, or the boolean value **False** otherwise. Furthermore, the command `FindHamiltonianCycle[G]` produces a (directed) hamiltonian cycle of a (di)graph G if it is hamiltonian (or an empty list if it is not). For example, the commands

```
In[12]:= WeaklyConnectedComponents[G]
In[13]:= ConnectedComponents[G]
In[14]:= HamiltonianGraphQ[G]
In[15]:= FindHamiltonianCycle[G]
```

produce the output:

```
Out[12]:= {{1, 2, 3, 4, 5}}
Out[13]:= {{1, 2, 3, 4, 5}}
Out[14]:= True
Out[15]:= {{1 ↔ 2, 2 ↔ 5, 5 ↔ 3, 3 ↔ 4, 4 ↔ 1}}
```

Finally, the command **Eigensystem**[M] returns a list of the form {values, vectors}, where values is a list of eigenvalues of the square matrix M (this list may be retrieved on its own by means of the command **Eigenvalues**[M]) and where vectors is a list of eigenvectors of M (this list may also be retrieved on its own by means of the command **Eigenvectors**[M]) associated with these eigenvalues, listed in the same order as the eigenvalues. These lists are computed analytically; to obtain numerical approximations of the results, the command **N**[•] may be used. For example, the command

```
In[16]:= N[Eigensystem[m]]
```

produces the output:

```
Out[16]:= {{1.83929, -0.5 + 1.32288i, -0.5 - 1.32288i, -0.419643 + 0.606291i,
-0.419643 - 0.606291i}, {{1.2956, 0.543689, 0.839287, 1., 1.}, {-0.25
- 0.661438i, -0.25 - 0.661438i, 0.25 + 0.661438i, -0.75 + 0.661438i,
1.}, {-0.25 + 0.661438i, -0.25 + 0.661438i, 0.25 - 0.661438i, -0.75 -
0.661438i, 1.}, {0.352201 + 1.72143i, -0.771845 - 1.11514i, -1.41964 +
0.606291i, 1., 1.}, {0.352201 - 1.72143i, -0.771845 + 1.11514i,
-1.41964 - 0.606291i, 1., 1.}}}
```

Projects

This section contains two projects. The first concerns a graph theoretic model of a round-robin sports tournament containing six players. The reader is guided in a step-by-step fashion to explore the properties of this tournament in respect of the results established in this chapter, and to pronounce on the ranking of the tournament players. In the second project, the reader's attention is drawn to the fact that the contents of this chapter, which was presented mainly in the context of sports tournaments, is also applicable in subjective preference modelling.

Project 15.1: Tournament rankings

Consider the tournament $T_{15.4}$ of order six shown in Figure 15.10.

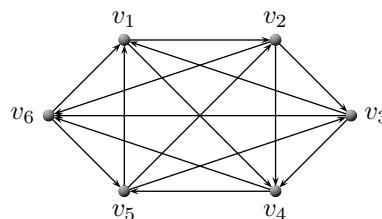


Figure 15.10: A tournament $T_{15.4}$ of order six.

Tasks

1. Verify that the tournament $T_{15.4}$ is strong, by constructing a distance matrix $\mathbf{D}(T_{15.4})$ for the tournament and testing whether $\mathbf{D}(T_{15.4}) < \infty$.
2. Use your results in [Task 1](#) to compute the radius and diameter of $T_{15.4}$.
3. Verify the result of [Theorem 15.9](#) for the tournament $T_{15.4}$ by producing, for each vertex v_i of $T_{15.4}$, cycles of lengths 3, 4, 5 and 6 containing v_i .
4. Use the method of outdegrees to produce a (first-order) score vector for the tournament $T_{15.4}$, and deduce a suitable ranking of the players in the tournament according to your score vector.
5. Suppose the adjacency matrix of the tournament $T_{15.4}$ is $\mathbf{A}_{T_{15.4}}$.
 - (a) Use [MATHEMATICA](#) to verify that $\mathbf{A}_{T_{15.4}}^{\text{diam}(T_{15.4})+3} > \mathbf{0}$.
 - (b) Use [MATHEMATICA](#) to compute the eigenvalues of $\mathbf{A}_{T_{15.4}}$, as well as a set of associated eigenvectors.
 - (c) Use your results in [Task 5\(b\)](#) to refine the ranking of players in $T_{15.4}$ obtained in [Task 4](#).

[Project 15.2: Subjective preference selection problem](#)

In a *subjective preference selection problem* (SPSP), the objective is to select a most pleasing alternative from a set of alternatives, based on the relative performances of the alternatives in respect of some selection criterion. What distinguishes this problem from other types of selection problems is that there is no universally accepted objective function for scoring the alternatives according to the selection criterion. Instead, every decision maker has his or her own relative preferences of the alternatives with respect to the adopted selection criterion.

In 1980, [Saaty](#) [9] proposed a method for eliciting subjective preference information with respect to a set of alternatives, called the *analytic hierarchy process*¹ (AHP). According to the AHP, the decision maker is required to express preference information about the alternatives in pairs according to some preference criterion and in terms of a so-called *judgement scale*. More specifically, the decision maker should specify that alternative i is a_{ij} times more pleasing or attractive than alternative j . This is done for all pairs of alternatives, in the process forming a so-called *pairwise comparison matrix* \mathbf{A} containing a_{ij} as the entry in row i and column j . For purposes of consistency, it is required that $a_{ij} = 1/a_{ji}$ for all pairs of alternatives i and j , and that the main diagonal of \mathbf{A} should contain unit entries.

According to the squared judgement scale², for example, the entries in the pairwise comparison matrix are specified by the decision maker based on the guidelines in [Table 15.1](#).

¹Although the AHP is, strictly speaking, a multi-criteria decision support tool within the realm of subjective preference modelling, it is described here in the single-criterion special case, where the single criterion is something akin to aesthetic pleasure or desirability of alternatives.

²There are also many other judgement scales that can be used in conjunction with the AHP.

a_{ij}	Description
1	Alternatives i and j are equally desirable
9	Alternative i is weakly more desirable than alternative j
25	Alternative i is strongly more desirable than alternative j
64	Alternative i is very strongly more desirable than alternative j
81	Alternative i is absolutely more desirable than alternative j

Table 15.1: Descriptions of the meaning of an entry a_{ij} in row i and column j of a pairwise comparison matrix involving alternative i relative to alternative j , according to the squared judgement scale. Values in the set $\{4, 16, 36, 49\}$ may be used as intermediate specifications.

Tasks

1. Explain in what sense an AHP pairwise comparison matrix, as described above, can be considered an extension or generalisation of the notion of an adjacency matrix of a tournament.
2. Does the [Perron-Frobenius Theorem \(Theorem 15.12\)](#) make any pronouncement on whether or not the eigenvalue with largest modulus of an AHP pairwise comparison matrix is real-valued? Motivate.
3. Suppose a decision maker specifies the following pairwise comparison matrix in respect of twelve alternatives a_1, \dots, a_{12} within a subjective preference selection setting:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} & a_{19} & a_{10} & a_{11} & a_{12} \\ a_{12} & 1 & 25 & 9 & 1/4 & 1/4 & 4 & 4 & 1/4 & 1/4 & 4 & 25 & 4 \\ a_{13} & 1/25 & 1 & 1/4 & 1/49 & 1/49 & 1/25 & 1/9 & 1/49 & 1/36 & 1/25 & 4 & 1/9 \\ a_{14} & 1/9 & 4 & 1 & 1/16 & 1/16 & 1/9 & 1/4 & 1/16 & 1/9 & 1/9 & 4 & 1/4 \\ a_{15} & 4 & 49 & 16 & 1 & 1/4 & 4 & 4 & 1/4 & 4 & 4 & 49 & 4 \\ a_{16} & 4 & 49 & 16 & 4 & 1 & 4 & 4 & 1/4 & 4 & 4 & 49 & 4 \\ a_{17} & 1/4 & 25 & 9 & 1/4 & 1/4 & 1 & 4 & 1/4 & 1/4 & 1/4 & 25 & 4 \\ a_{18} & 1/4 & 9 & 4 & 1/4 & 1/4 & 1/4 & 1 & 1/4 & 1/4 & 1/4 & 9 & 4 \\ a_{19} & 4 & 49 & 16 & 4 & 4 & 4 & 4 & 1 & 4 & 4 & 49 & 4 \\ a_{10} & 4 & 36 & 9 & 1/4 & 1/4 & 4 & 4 & 1/4 & 1 & 4 & 36 & 4 \\ a_{11} & 1/4 & 25 & 9 & 1/4 & 1/4 & 4 & 4 & 1/4 & 1/4 & 1 & 25 & 4 \\ a_{12} & 1/25 & 1/4 & 1/4 & 1/49 & 1/49 & 1/25 & 1/9 & 1/49 & 1/36 & 1/25 & 1 & 1/9 \\ a_{13} & 1/4 & 9 & 4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 9 & 1 \end{bmatrix}$$

Use [MATHEMATICA](#) to compute an eigenvector of \mathbf{A} corresponding to the eigenvalue with largest modulus.

4. Use your eigenvector in [Task 3](#) above to rank the alternatives a_1, \dots, a_{12} from most desirable to least desirable, based on the pairwise comparison information supplied by the decision maker.

Further reading

- [1] A Berman and RJ Plemmons, 1994. *Nonnegative Matrices in the Mathematical Sciences*, Society for Industrial, Applied Mathematics, Philadelphia, (PA).
- [2] AP Burger, 2013. *Computational results on the traceability of oriented graphs of small order*, Electronic Journal of Combinatorics, **20(4)**, #P23.
- [3] FG Frobenius, 1912. *Über Matrizen aus nicht negativen Elementen*, Königliche Gesellschaft der Wissenschaften, pp. 456–477.
- [4] HG Landau, 1953. *On dominance relations and the structure of animal societies. III. The condition for a score structure*. Bulletin of Mathematical Biophysics, **15**, pp. 143–148.
- [5] JW Moon, 1966. *On subtournaments of a tournament*, Canadian Mathematical Bulletin, **9**, pp. 297–301.
- [6] O Perron, 1907. *Zur Theorie der Matrizen*, Mathematische Annalen, **64(2)**, pp. 248–263.
- [7] L Rédei, 1934. *Ein kombinatorischer Satz*, Acta Litt. Szeged, **7**, pp. 39–43.
- [8] HE Robbins, 1939. *A theorem on graphs, with an application to a problem in traffic control*, American Mathematical Monthly, **46**, pp. 281–283.
- [9] TL Saaty, 1980. *The Analytic Hierarchy Process: Planning, Priority Setting, Resources Allocation*, McGraw-Hill, New York (NY).
- [10] C Thomassen, 1978. *Hypohamiltonian graphs and digraphs*, pp. 557–571 in Y Alavi and DR Lick (Eds), *Theory and Applications of Graphs*, Springer, Berlin.
- [11] SA van Aardt, AP Burger and M Frick, 2013. *An infinite family of planar hypohamiltonian oriented graphs*, Graphs and Combinatorics, **29(4)**, pp. 729–733.
- [12] SA van Aardt, AP Burger, M Frick, A Kemnitz and I Schiermeyer, 2015. *Hypohamiltonian oriented graphs of all possible orders*, Graphs and Combinatorics, **31(6)**, pp. 1821–1831.
- [13] SA van Aardt, M Frick, P Katrenič and MH Nielsen, 2011. *The order of hypotraceable oriented graphs*, Discrete Mathematics, **311(14)**, pp. 1273–1280.



PART

Topics in
Modern Graph Theory



Domination in graphs

Contents

16.1	Introduction	527
16.2	The domination number	529
16.3	The independent domination number	555
16.4	The irredundance number	569
16.5	Total domination	576
	Exercises	594
	Computer exercises	595
	Projects	599
	Further reading	615

16.1 Introduction

For integers $m \geq n \geq 1$, consider a lottery scheme in which a player participates by purchasing a playing set \mathcal{L} of any number of n -element subsets (or n -sets), called **tickets**, from a universal set $\mathcal{U}_m = [m]$ prior to the winning draw. A winning ticket is drawn randomly from the universal set \mathcal{U}_m . The participant wins a prize, called a **k -prize**, if at least k numbers in one of the tickets that belong to his/her playing set \mathcal{L} match those in the winning ticket, for some integer $k \in [n]$. We denote such a lottery system by the triple $\langle m, n; k \rangle$.

Suppose a participant of a lottery system $\langle m, n; k \rangle$ wishes to play in such a way that (s)he is ensured to win a k -prize. One option is for the participant to buy all possible $\binom{m}{n}$ (distinct) lottery tickets, but that would prove to be too costly an exercise. A more realistic option is for the participant to find the smallest number of lottery tickets that must be purchased in order to be guaranteed a k -prize. Such a strategy would enable the participant to maximise his/her profit. The participant therefore wishes to choose a smallest possible playing set \mathcal{L} in such a way that there will necessarily be at least one ticket (n -set) in \mathcal{L} which contains at least k numbers of \mathcal{U}_m matching those of the winning ticket, no matter which winning ticket is chosen from \mathcal{U}_m . Such a playing set we call a **k -optimal playing set**. The cardinality of a k -optimal playing set is called the **lottery number** $L(m, n; k)$. A playing set of cardinality $L(m, n; k)$ is called an **optimal playing set**.

In most lotteries around the world today, players of the lottery select six numbers on a ticket, from a set of 49 numbers, and are awarded a prize if they have

at least $k \geq 3$ numbers in common with the set of six winning numbers, drawn at random by a governing body. Using our terminology, this is the lottery $\langle 49, 6; k \rangle$ which is used in various countries including France, Germany, Greece, Poland, South Africa, Spain and the United Kingdom, as well as in several states in Canada and the United States of America.

Consider, as an example, the unrealistically small lottery $\langle 5, 3; 2 \rangle$ in which each ticket is a 3-element subset of $\mathcal{U}_5 = [5]$. In order to be guaranteed a 2-prize in this lottery, a player may buy the two tickets $t_1 = \{1, 2, 3\}$ and $t_2 = \{2, 4, 5\}$ to form the playing set $\mathcal{L} = \{t_1, t_2\}$. Suppose that t^* is chosen as the winning ticket. If t^* does not have at least two numbers in common with the ticket t_1 , then t^* contains both numbers 4 and 5 and therefore has at least two numbers in common with the ticket t_2 . Hence no matter which winning ticket is chosen, it will have at least two numbers in common with the ticket t_1 or the ticket t_2 . Since each of the $\binom{5}{2} = 10$ possible winning tickets share at least two numbers with at least one of the tickets in the playing set \mathcal{L} , we have $L(5, 3; 2) \leq |\mathcal{L}| = 2$. For any given ticket t , however, a ticket containing both numbers that are missing from t does not have at least two numbers in common with t . Hence there is no playing set comprising one ticket which can guarantee the player a 2-prize in the lottery $\langle 5, 3; 2 \rangle$, and so $L(5, 3; 2) > 1$. Consequently, $L(5, 3; 2) = 2$ and $\mathcal{L} = \{t_1, t_2\}$ is a 2-optimal playing set.

The problem of determining the value of a lottery number, $L(m, n; k)$, is known as the **lottery problem** and has been studied in the combinatorial literature since the 1960s. The lottery problem may be translated into the realm of graph theory by defining a so-called **lottery graph**, denoted by $G\langle m, n; k \rangle$, for a lottery system $\langle m, n; k \rangle$, in which the vertices represent all the tickets that may possibly be played in the lottery, and in which two vertices are adjacent if the corresponding tickets share at least k numbers. The vertex set of $G\langle m, n; k \rangle$ consists of all $\binom{m}{n}$ possible n -element subsets of $\mathcal{U}_m = [m]$, while two vertices in $G\langle m, n; k \rangle$ are adjacent if the corresponding two n -element subsets share a common k -subset. The lottery graph $G\langle 5, 3; 2 \rangle$ is shown in Figure 16.1, with the optimal playing set $\mathcal{L} = \{\{1, 2, 3\}, \{2, 4, 5\}\}$ indicated by highlighted vertices.

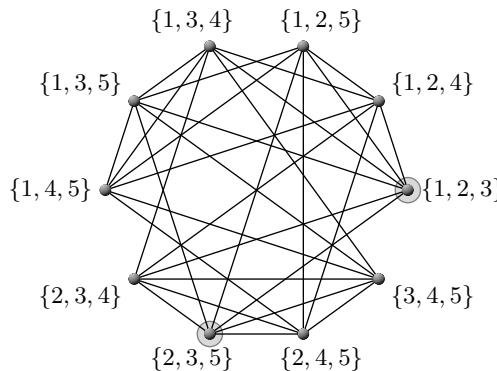


Figure 16.1: The lottery graph $G\langle 5, 3; 2 \rangle$.

Let S be a set of vertices in the lottery graph $G\langle m, n; k \rangle$ with the property that every vertex not in the set S is adjacent to at least one vertex in S . Then the tickets corresponding to the vertices in S form a playing set for a lottery

system $\langle m, n; k \rangle$ which guarantees the player a k -prize. Such a set S is called a dominating set of the lottery graph. A dominating set of minimum cardinality in the lottery graph $G\langle m, n; k \rangle$ corresponds to a k -optimal playing set for the lottery system $\langle m, n; k \rangle$. Thus, domination in graphs can be applied to study the lottery problem. In this chapter, we consider the properties of dominating sets in graphs.

- ❖ The reader should now be able to attempt [Project 16.1](#).

16.2 The domination number

First we recall the concept of the neighbourhood of a vertex and of a set of vertices in a graph. Let $G = (V, E)$ be a graph, and let v be a vertex in V . Recall, from [Chapter 1](#), that the **open neighbourhood** of v in G is $N(v) = \{u \in V \mid uv \in E\}$ and the **closed neighbourhood** of v is $N[v] = \{v\} \cup N(v)$. A vertex in $N(v)$ is called a **neighbour** of v . For a set $S \subseteq V$, its **open neighbourhood** is the set $N(S) = \bigcup_{v \in S} N(v)$ and its **closed neighbourhood** is the set $N[S] = N(S) \cup S$.

Moreover, we say that G is **F -free** if G does not contain a graph F as an induced subgraph. In particular, we say a graph is **claw-free** if it is $K_{1,3}$ -free and that it is **quadrilateral-free** if it is C_4 -free.

A **dominating set** S of a graph G is a set of vertices of G such that each vertex not in S is adjacent to a vertex of S . Equivalently, a set S of vertices of $G = (V, E)$ is a dominating set if $N[S] = V$. A vertex v in G is said to **dominate** itself and each of its neighbours. Thus, if S is a dominating set in G , then every vertex of G is dominated by at least one vertex of S . The **domination number** of G , denoted by $\gamma(G)$, is the minimum cardinality of a dominating set of G . A dominating set of cardinality $\gamma(G)$ is called a **γ -set of G** or a **minimum dominating set** of G . If X and Y are subsets of vertices in G , we say that X dominates Y if $Y \subseteq N[X]$. In particular, if X dominates V , then X is a dominating set in G .

To illustrate these definitions, consider the Petersen graph $G_{16.1}$ shown in [Figure 16.2](#). The set $S = N(v_1) = \{u_2, v_2, v_5\}$ is a dominating set of $G_{16.1}$, and so $\gamma(G_{16.1}) \leq |S| = 3$. Since $G_{16.1}$ is a cubic graph, each vertex dominates four vertices. Therefore, no set of two vertices dominates all ten vertices in the graph, and so $\gamma(G_{16.1}) \geq 3$. Consequently, $\gamma(G_{16.1}) = 3$. One can, in fact, show that the Petersen graph $G_{16.1}$ has exactly ten distinct γ -sets, namely the ten open neighbourhoods $N(v)$ corresponding to the ten vertices v in the graph.

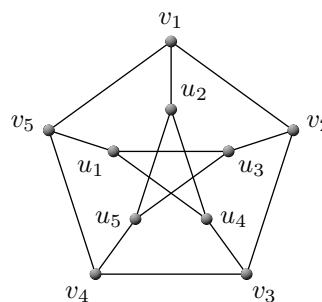


Figure 16.2: The Petersen graph $G_{16.1}$.

For the complete graph, $\gamma(K_n) = 1$. For the path and cycle, it is a simple exercise to show that $\gamma(P_n) = \gamma(C_n) = \lceil \frac{n}{3} \rceil$ for all $n \geq 3$. Since the early 1970s, domination in graphs has received considerable attention in the literature due to its many and varied applications.

The earliest ideas of dominating sets date back to problems involving chess pieces, where one wishes to cover or dominate various opposing pieces or various squares of the chessboard. In 1862, [De Jaenisch](#) [29] posed the problem of finding the minimum number of queens that can be placed on a chessboard so that each square of the chessboard is dominated by at least one of the queens. A graph may be formed from an 8×8 chessboard in which the vertices represent the squares of the board and in which two vertices are adjacent if a queen situated on one square can occupy the other square in a single move. This graph is referred to as the **queen's graph**, denoted by Q_8 . The minimum number of queens that cover all the squares of an 8×8 chessboard is the domination number $\gamma(Q_8)$. For the queen's graph, five queens suffice; that is, $\gamma(Q_8) = 5$.

The classical problems of covering chessboards with the minimum number of chess pieces rekindled interest in dominating concepts. Ultimately the theory of domination was formalised by [Berge](#) [5] and [Ore](#) [91] in 1962. [Ore](#) coined the term “domination number,” although [Berge](#) was the first to define it as a graph parameter (then called the coefficient of external stability).

Some applications of dominating sets include the following. [Berge](#) [5] mentioned the problem of keeping a number of strategic locations under surveillance by a set of radar stations. The minimum number of radar stations needed to survey all the locations is the domination number of the associated graph. In a similar vein, [Liu](#) [82] discussed the application of domination to communications in a network, where a dominating set represents a set of cities which, acting as transmission stations, can transmit messages to every city in the network.

The notion of domination is also a standard one in coding theory. If one defines a graph whose vertices are the n -dimensional vectors with coordinates chosen from $[p]$ and two vertices are adjacent if they differ in one coordinate, then sets of vectors which are (n, p) -covering sets, single error correcting codes, or perfect covering sets are all dominating sets of the graph with certain additional properties. See, for example, [Kalfleisch et al.](#) [73].

As a further example, a desirable property of a committee chosen from a collection of people might be that every nonmember knows at least one member of the committee, for ease of communication. A committee with this property is a dominating set of the acquaintance graph of the set of people.

- ❖ The reader should now be able to attempt [Project 16.2](#).

16.2.1 Minimal dominating sets

If S is a dominating set of a graph G , then so too is every superset of S . Not every subset of S is, however, necessarily a dominating set. A **minimal dominating set** of a graph G is a dominating set of G that contains no dominating set of G as a proper subset. For example, the sets $S_1 = N(v_1) = \{u_2, v_2, v_5\}$, $S_2 = \{u_1, v_1, v_3, u_5\}$, and $S_3 = \{v_1, v_2, v_3, v_4, v_5\}$ are all minimal dominating sets of the Petersen graph $G_{16.1}$ shown in [Figure 16.2](#). Hence, the Petersen graph contains minimal dominating sets of cardinalities 3, 4 and 5.

Early work on the topic of domination focused on properties of minimal dominating sets. In order to state these properties, we introduce some additional notation. Let $G = (V, E)$ be a graph. Let $S \subseteq V$ be a subset of vertices in G and let v be a vertex in V . The **S -private neighbourhood** of a vertex $v \in S$ is defined as $\text{pn}[v, S] = \{w \in V \mid N[w] \cap S = \{v\}\}$, while its **S -external private neighbourhood** is defined as $\text{epn}[v, S] = \text{pn}[v, S] \cap (V \setminus S)$ and its **S -internal private neighbourhood** by $\text{ipn}[v, S] = \text{pn}[v, S] \cap S$. Hence, $\text{pn}[v, S] = \text{epn}[v, S] \cup \text{ipn}[v, S]$. We remark that either v is adjacent to some other vertex in S , in which case $\text{ipn}[v, S] = \emptyset$, or v is isolated in $G[S]$, in which case $\text{ipn}[v, S] = \{v\}$.

We begin with a classical result of Ore [91].

Theorem 16.1 *If D is a dominating set of a graph G , then D is a minimal dominating set of G if and only if $\text{epn}[v, D] \neq \emptyset$ or $\text{ipn}[v, D] \neq \emptyset$ for each $v \in D$.*

Proof Suppose, first, that D is a minimal dominating set of G . Then, for each vertex v of D , the set $D \setminus \{v\}$ is not a dominating set of G . Hence there is a vertex $w \in V \setminus (D \setminus \{v\})$ that is adjacent to no vertex of $D \setminus \{v\}$. If $w \in D$, then $w = v$ and v is isolated in $G[D]$, and so $\text{ipn}[v, D] = \{v\}$. If $w \in V \setminus D$, then, since every vertex not in D is adjacent to some vertex of D , w is adjacent to v and to no other vertex of D , and so $w \in \text{epn}[v, D]$. Conversely, if $\text{epn}[v, D] \neq \emptyset$ or $\text{ipn}[v, D] \neq \emptyset$ for each vertex $v \in D$, then $D \setminus \{v\}$ is not a dominating set of G for any such vertex v . ■

The next result follows immediately from [Theorem 16.1](#).

Corollary 16.2 *If $G = (V, E)$ is a graph with no isolated vertex and D is a minimal dominating set of G , then $V \setminus D$ is a dominating set of G .*

Proof By [Theorem 16.1](#), $\text{epn}[v, D] \neq \emptyset$ or $\text{ipn}[v, S] \neq \emptyset$ for each vertex $v \in D$. If $\text{epn}[v, D] \neq \emptyset$, then clearly v is adjacent to some vertex of $V \setminus D$. If $\text{ipn}[v, S] \neq \emptyset$, then v is isolated in $G[D]$ and, by hypothesis, v is adjacent to some vertex in $V \setminus D$. Hence every vertex of D is adjacent to some vertex of $V \setminus D$. ■

Bollobás and Cockayne [7] established the following property of minimum dominating sets in graphs.

Theorem 16.3 *If G is a graph with no isolated vertex, then there exists a γ -set D of G in which $\text{epn}[v, D] \neq \emptyset$ for every vertex $v \in D$.*

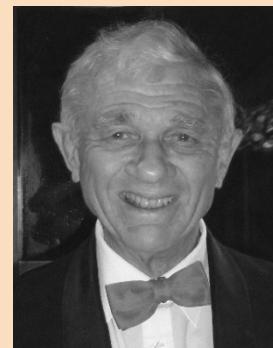
Proof Among all γ -sets of G , let D be one for which the number of edges in $G[D]$ is a maximum, and let $v \in D$. If $\text{epn}[v, D] = \emptyset$, then every vertex of $V \setminus D$ adjacent to v is adjacent to some other vertex of D . By [Theorem 16.1](#), $\text{ipn}[v, D] \neq \emptyset$, implying that v is isolated in $G[D]$. By hypothesis, v is adjacent to a vertex w in $V \setminus D$. But then $(D \setminus \{v\}) \cup \{w\}$ is a γ -set of G whose induced subgraph contains at least one edge incident with w , and hence has more edges than the subgraph induced by D . This contradicts our choice of D . Hence, $\text{epn}[v, D] \neq \emptyset$. ■

Béla Bollobás was born in 1943, in Hungary. He obtained a first doctorate in discrete geometry under the supervision of László Tóth and **Paul Erdős** at Budapest University in 1967. In 1972, he received a second doctorate in functional analysis (on Banach algebras) under the supervision of Frank Adams from Cambridge University. He is well known for his significant contributions to many areas of combinatorics, including extremal graph theory and random graph theory. He is a prolific researcher, having co-authored many journal papers, and is author of at least a dozen books.



Biographic note 72: Béla Bollobás (1943–present)

Ernie Cockayne was born in Leicester, England, in 1940. He obtained a master's degree from Oxford University in 1965 and a doctorate from the University of British Columbia, in 1967. He spent his entire academic career in the Department of Mathematics and Statistics of the University of Victoria, Canada, from 1966 until his retirement in 2005, the last 25 years as full professor. He was a pioneer in the development of the theory of domination, independence and irredundance in graphs. He has co-authored many journal papers, about half of them with [Kieka Mynhardt](#).



Biographic note 73: Ernie Cockayne (1940–present)

16.2.2 Bounds on the domination number

In this section we investigate bounds on the domination number of a graph. Throughout this section, we let $\mathcal{G}_{n,m}$ denote the class of all graphs of order n and size m without isolated vertices. We prove that all the essential upper bounds on the domination number of a graph $G \in \mathcal{G}_{n,m}$ can be written in the unified form

$$\gamma(G) \leq \frac{an + bm}{2a + b} \tag{16.1}$$

for real constants $b \geq 0$ and $a > -\frac{b}{2}$. As a special case of a more general result (see [12]), we have the following lemma which shows that the inequalities $b \geq 0$ and $a > -\frac{b}{2}$ must hold in (16.1).

Lemma 16.4 ([12]) *Let $a, b \in \mathbb{R}$. Then $an + bm > 0$ for every graph $G \in \mathcal{G}_{n,m}$ if and only if $b \geq 0$ and $a > -\frac{b}{2}$.*

Proof Suppose the inequality $an + bm > 0$ holds for every graph $G \in \mathcal{G}_{n,m}$ and for each pair of real numbers a and b . If $b < 0$, then there exists a sufficiently large integer ℓ (simply take $\ell > (b - 2a)/b$) such that the complete graph on ℓ vertices achieves a negative value for $an + bm$. On the other hand, for all pairs a, b where $a \leq -\frac{b}{2}$, the graph $G = K_2$ with $n = 2$ and $m = 1$ shows that $an + bm$ is not always positive on $\mathcal{G}_{n,m}$. Hence, both conditions $b \geq 0$ and $a > -\frac{b}{2}$ are necessary.

To prove the sufficiency, suppose that a and b are real numbers satisfying $b \geq 0$ and $a > -\frac{b}{2}$. Let G be an arbitrary graph in $\mathcal{G}_{n,m}$. If $a > 0$, the inequality $an + bm > 0$ trivially holds. Hence, we may assume that $a \leq 0$, for otherwise the desired result holds. Observe that $n \leq 2m$ holds since $\delta(G) \geq 1$. This implies that $an + bm \geq (2a + b)m > 0$, thereby establishing the sufficiency. ■

In view of Lemma 16.4, a general way to formulate our problem is as follows: “Determine the shape of the surface $\Phi(x, y, z)$ which is the subset of $D = \{(x, y, z) \mid y \geq 0 \text{ and } x > -\frac{y}{2}\} \subset \mathbb{R}^3$ defined by the rule

$$z = \sup_{G \in \mathcal{G}_{n,m}} \frac{\gamma(G)}{xn + ym}.$$

In other words, determine $z = z(x, y)$ as a function of x and y .

We prove the following result due to Bujtás *et al.* [12].

Theorem 16.5 *The surface $\Phi(x, y, z)$ is determined by*

$$z(x, y) = \frac{1}{2x + y}.$$

Proof Let $y \geq 0$ and $x > -\frac{y}{2}$ be arbitrary real numbers and $G \in \mathcal{G}_{n,m}$. We observe that the graph $G = K_2$ with $n = 2$ and $m = 1$ has $\gamma(G) = 1$. Hence, a simple general lower bound for z is given by

$$z(x, y) \geq \frac{1}{2x + y}.$$

It therefore suffices to prove that

$$\gamma(G) \leq \frac{xn + ym}{2x + y} \tag{16.2}$$

for every $G \in \mathcal{G}_{n,m}$. For notational simplicity, let $\varphi(G) = (xn + ym)/(2x + y)$ denote the right-hand side of the inequality (16.2). Therefore, we wish to show that $\gamma(G) \leq \varphi(G)$.

Suppose that $x \leq 0$ (and still $x > -\frac{y}{2}$). Let $G \in \mathcal{G}_{n,m}$. The set formed by sequentially adding a vertex incident with each edge of G produces a dominating set of size at most m , and so $\gamma(G) \leq m$. Hence, since $n \leq 2m$ holds for the graph G which contains no isolated vertex, we obtain the chain of inequalities

$$\varphi(G) = \frac{xn + ym}{2x + y} \geq \frac{2xn + ym}{2x + y} = m \geq \gamma(G),$$

noting that x is nonpositive, which establishes the desired inequality. Hence, we may assume that $x > 0$ (and still $y \geq 0$). We apply induction on $n + m$ to prove that $\gamma(G) \leq \varphi(G)$ for every graph G without isolated vertices, and for any two real numbers x, y with $x > 0$, $y \geq 0$. Since $n \geq 2$ and $m \geq 1$, the assertion is obvious whenever G has a dominating vertex since then $\gamma(G) = 1$ and $\varphi(G) \geq 1$. In particular, this establishes the base case when $n + m = 3$.

Let $G \in \mathcal{G}$ and assume as induction hypothesis that the result holds for all graphs $F \in \mathcal{G}$ with $n(F) + m(F) < n(G) + m(G) = n + m$. If G is a disconnected graph with components G_1, G_2, \dots, G_k , then

$$\gamma(G) = \sum_{i=1}^k \gamma(G_i) \quad \text{and} \quad \varphi(G) = \sum_{i=1}^k \varphi(G_i).$$

Hence, both sides of the inequality (16.2) are additive with respect to vertex-disjoint union. We may therefore assume, without loss of generality, that G is connected, for otherwise we could apply the result to each component.

Suppose that G is not a tree. Let e be a cycle edge of G , and consider the graph $G - e$. Then $G - e$ is connected. Since removing edges does not decrease the domination number, we have that $\gamma(G) \leq \gamma(G - e)$. By induction, $\gamma(G - e) \leq \varphi(G - e)$. However, $\varphi(G - e) < \varphi(G)$. Consequently, $\gamma(G) < \varphi(G)$ and we have strict inequality in (16.2).

We may therefore assume that G is a tree. Suppose that $\text{diam}(G) > 2$. Then G has an edge e such that each component of $G - e$ has more than one vertex. Let G_1 and G_2 denote the two components of $G - e$. Applying the induction hypothesis to each component of $G - e$, we have that $\gamma(G) \leq \gamma(G_1) + \gamma(G_2) \leq \varphi(G_1) + \varphi(G_2) < \varphi(G)$ and again we obtain strict inequality in (16.2). Finally, if G is a tree with $\text{diam}(G) = 2$, then we are back to the basic case that G has a dominating vertex. ■

An equivalent formulation of [Theorem 16.5](#) gives us the following general upper bound on the domination number of a graph, due to [Bujtás et al. \[12\]](#).

Theorem 16.6 *The bound $\gamma(G) \leq an + bm$ is valid for every graph $G \in \mathcal{G}_{n,m}$ if and only if both $2a + b \geq 1$ and $b \geq 0$ hold.*

Proof Suppose the bound $\gamma(G) \leq an + bm$ is valid for every graph $G \in \mathcal{G}_{n,m}$. Since every graph has domination number at least one, $an + bm > 0$, and so, by [Lemma 16.4](#), $b \geq 0$ and $a > -\frac{b}{2}$. Furthermore, since $\gamma(G)/(an + bm) \leq 1$, [Theorem 16.5](#) implies that $1/(2a + b) = z(a, b) \leq 1$. Thus, both $2a + b \geq 1$ and $b \geq 0$ hold.

Conversely, suppose that both $2a + b \geq 1$ and $b \geq 0$ hold. Then $a > -\frac{b}{2}$ and $b \geq 0$, and so, by [Theorem 16.5](#), $z(a, b) = 1/(2a + b)$. Since $2a + b \geq 1$, we have that $z(a, b) \leq 1$ and therefore $\gamma(G)/(an + bm) \leq 1$ holds for every graph $G \in \mathcal{G}_{n,m}$. ■

To illustrate the result of [Theorem 16.6](#), let $G \in \mathcal{G}_{n,m}$. Taking $(a, b) = (\frac{1}{3}, \frac{1}{3})$, we have that $\gamma(G) \leq (n + m)/3$. Taking $(a, b) = (\frac{1}{4}, \frac{1}{2})$, we have that $\gamma(G) \leq (n + 2m)/4$. Taking $(a, b) = (\frac{1}{2}, 0)$, we have the following upper bound on the domination number in terms of the order due to [Ore \[91\]](#).

Corollary 16.7 ([91]) *If G is a graph of order n with no isolated vertex, then $\gamma(G) \leq \frac{n}{2}$.*

We remark that [Corollary 16.7](#) can also be deduced from [Corollary 16.2](#) or [Theorem 16.3](#) (see [Exercise 16.1](#)). A large family of graphs attaining the bound in [Corollary 16.7](#) can be established using the following transformation of a graph. The **ℓ -corona** of a graph G is the graph of order $(\ell + 1)|V(G)|$ obtained from G by attaching a path of length ℓ to each vertex of G so that the resulting paths are pairwise vertex disjoint. The 1-corona of G , denoted by $\text{cor}(G)$, is also called the **corona**

of G . Hence the corona $\text{cor}(G)$ of G is that graph obtained from G by adding a pendant edge to each vertex of G . We observe that the corona of a graph is a graph containing no isolated vertex and with domination number exactly half its order.

[Payan and Xuong](#) [92] characterised those graphs with no isolated vertex and with domination number exactly half their order. A subset of vertices of a graph G is **independent** if no two vertices in the set are adjacent in G .

Theorem 16.8 *If G is a graph of order n with no isolated vertex, then $\gamma(G) = \frac{n}{2}$ if and only if the components of G are C_4 or $\text{cor}(H)$ for some connected graph H .*

Proof The sufficiency is obvious. To prove the necessity, we note that $\gamma(G) = \frac{n}{2}$ if and only if the domination number of any component of G equals half its order. Hence, without loss of generality, we may assume that $G = (V, E)$ is connected. By [Theorem 16.3](#), we know that there exists a γ -set D of G in which $\text{epn}[v, D] \neq \emptyset$ for every vertex $v \in D$. Let $D = \{v_1, \dots, v_{n/2}\}$. For each vertex v_i of D , let $w_i \in \text{epn}[v, D]$. Thus, $w_i \in V \setminus D$ and $N(w_i) \cap D = \{v_i\}$. Since $\gamma(G) = \frac{n}{2}$, it holds that $V = D \cup \{w_1, \dots, w_{n/2}\}$. It follows that the only edges joining D and $V \setminus D$ are the edges $v_i w_i$, $i \in [\frac{n}{2}]$. If $n = 2$, then $G = K_2 = \text{cor}(K_1)$. If $n = 4$, then either $G = C_4$ or $G = P_4 = \text{cor}(K_2)$. Hence, we may assume that $n \geq 6$.

We show first that $d(v_i) = 1$ or $d(w_i) = 1$ for every $i \in [\frac{n}{2}]$. If this is not the case, then, renaming vertices if necessary, we may assume that $d_G(v_1) \geq 2$ and $d(w_1) \geq 2$. Then $v_1 v_i \in E$ and $w_1 w_j \in E$ for some i and j where $1 < i, j \leq \frac{n}{2}$. If $i \neq j$, then G contains a spanning subgraph H that is isomorphic to $P_6 \cup (\frac{n-6}{2})K_2$ (here the path P_6 is the path $v_j w_j w_1 v_1 v_i w_i$). If $i = j$, then, since G is connected, there must be an edge joining a vertex in $\{v_1, v_i, w_1, w_i\}$ to a vertex in $V \setminus \{v_1, v_i, w_1, w_i\}$. We may assume that $v_i v_k$ is such an edge. Thus, once again, G contains a spanning subgraph H that is isomorphic to $P_6 \cup (\frac{n-6}{2})K_2$ (here the path P_6 is the path $w_k v_k v_i v_1 w_1 w_i$). Since adding edges to a graph does not increase its domination number, $\gamma(G) \leq \gamma(H) = 2 + (n-6)/2 < \frac{n}{2}$, contradicting the fact that $\gamma(G) = \frac{n}{2}$. Hence, $d(v_1) = 1$ or $d(w_1) = 1$. More generally, $d(v_i) = 1$ or $d(w_i) = 1$ for every $i \in [\frac{n}{2}]$.

Let v be a vertex of G of degree at least two. Renaming vertices if necessary, we may assume that $v = v_1$. We show that every vertex in $V \setminus D$ has degree one in G . Let $w \in V \setminus D$. Since G is connected, there is a $v-w$ path P in G . Let w' be the first vertex on P not in D and let v' be the vertex that immediately precedes w' on P . Then, $v' = v_i$ and $w' = w_i$ for some $i \in [\frac{n}{2}]$. Since $d(v_i) \geq 2$, we have that $d(w_i) = 1$. Thus, $w = w_i$ and $d(w) = 1$. This is true for every vertex $w \in V \setminus D$. In particular, the set $V \setminus D$ is an independent set. Thus, since G is a connected graph, the subgraph $H = G[D]$ induced by the set D is a connected graph. Consequently, $G = \text{cor}(H)$ for some connected graph H , as claimed. ■

Bounds for graphs with minimum degree at least two

If we restrict the minimum degree $\delta(G)$ of G to be at least two, then the upper bound on the domination number in [Corollary 16.7](#) due to Ore can be improved from one-half its order to two-fifths its order, except for seven exceptional graphs (one of order four and six of order seven). More precisely, [McCuaig and Shepherd](#) [85] proved the following result. We omit the proof.

Theorem 16.9 *If G is a connected graph of order n with $\delta(G) \geq 2$ and $G \notin \mathcal{B}$, where \mathcal{B} is the collection of seven graphs shown in [Figure 16.3](#), then $\gamma(G) \leq \frac{2}{5}n$.*

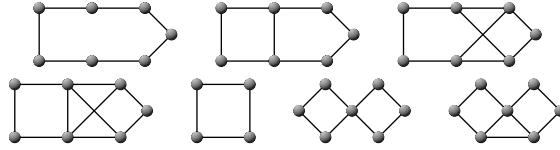


Figure 16.3: The family \mathcal{B} of exceptional graphs.

As an immediate consequence of [Theorem 16.9](#), we have the following result.

Theorem 16.10 *If G is a connected graph of order $n \geq 8$ with $\delta(G) \geq 2$, then $\gamma(G) \leq \frac{2}{5}n$.*

To show that the bound of [Theorem 16.10](#) is sharp, McCuaig and Shepherd [85] introduced a family \mathcal{F} of graphs constructed as follows. We denote the graph obtained from a 4-cycle C_4 by adding a pendant edge to one of its vertices by $L_{4,1}$ which we call a **key**. We define a **unit** to be a graph that is isomorphic to a 5-cycle C_5 or to a key $L_{4,1}$, and we call a unit a **cycle unit** or a **key unit** according to whether it is a cycle or a key, respectively. In a cycle unit, we select two vertices at distance two apart in the unit and we call these two vertices the **link vertices** of the unit, while in a key unit we call the vertex of degree one the **link vertex** of the unit.

Let \mathcal{F} denote the family of all graphs G that are obtained from the disjoint union of at least two units, each of which is a cycle unit or a key unit (called the **units** of G), by adding edges in such a way that G is connected and every added edge joins two link vertices. A graph in the family \mathcal{F} with three cycle units and two key units is shown in [Figure 16.4](#) with the link vertices highlighted.

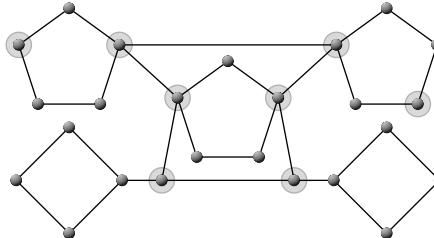


Figure 16.4: A graph in the family \mathcal{F} .

Let G be an arbitrary graph in the family \mathcal{F} . Then, G has order $n = 5k$ for some integer $k \geq 2$. Every dominating set of G contains at least two vertices from each unit in G , and so $\gamma(G) \geq 2k = \frac{2}{5}n$. Since G is a connected graph of order $n \geq 10$, it follows from [Theorem 16.10](#) that $\gamma(G) \leq \frac{2}{5}n$. Consequently, $\gamma(G) = \frac{2}{5}n$ for every graph $G \in \mathcal{F}$ of order n .

Bounds for graphs with minimum degree at least three

If we restrict the minimum degree to be at least three, then the upper bound on the domination number in [Theorem 16.10](#) due to McCuaig and Shepherd can be improved from two-fifths its order to three-eighths its order. The following result is due to Reed [100]. We omit the proof, which uses ingenious counting arguments.

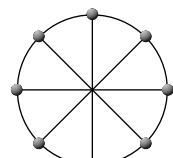
Theorem 16.11 If G is a graph of order n with $\delta(G) \geq 3$, then $\gamma(G) \leq \frac{3}{8}n$.

Bruce Reed is a Canadian mathematician and computer scientist. He obtained a doctorate in 1986 from McGill University, under the supervision of the Czech mathematician, Vašek Chvátal. Before returning to McGill University to occupy a Canada Research Chair in Graph Theory as a professor of computer science, he held positions at the University of Waterloo, Carnegie Mellon University, and the French National Centre for Scientific Research. He is best known for his significant contributions to a variety of areas within graph theory, including coloring, connectivity, and domination, and is an expert on the probabilistic method and random graphs. He is a prolific researcher, having co-authored many journal papers, and is co-author of the book *Graph colouring and the probabilistic method* with Michael Molloy. He also co-authored several other books.

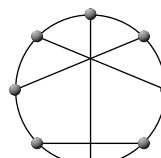


Biographic note 74: Bruce Reed (1962–present)

We remark that the two nonplanar cubic graphs of order $n = 8$ (shown in Figure 16.5) both have domination number three and achieve the upper bound in Theorem 16.11.



(a) $G_{16.2}$



(b) $G_{16.3}$

Figure 16.5: The two nonplanar cubic graphs of order eight.

That there are connected graphs of arbitrarily large order and minimum degree three satisfying the bound in Theorem 16.11 may be seen as follows. Let H' be any connected graph. For each vertex v of H' , add a (disjoint) copy of the cubic graph $G_{16.3}$ shown in Figure 16.5(b) and identify any one of its vertices that is in a triangle with v . Let H denote the resulting graph and let \mathcal{H} denote the family of all such graphs H . When $H' = P_3$, the resulting graph H is shown in Figure 16.6.

Let H be an arbitrary graph in the family \mathcal{H} . Then, H has order $n = 8k$ for some integer $k \geq 1$. Every dominating set in H contains at least three vertices from each copy of $G_{16.3}$ in H , and so $\gamma(H) \geq 3k = \frac{3}{8}n$. By Theorem 16.11, $\gamma(H) \leq \frac{3}{8}n$. Consequently, $\gamma(H) = \frac{3}{8}n$ for every graph $H \in \mathcal{H}$ of order n .

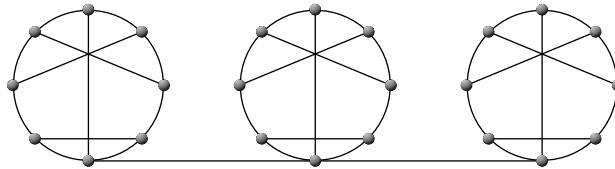


Figure 16.6: A graph $H \in \mathcal{H}$.

Bounds for cubic graphs

As a special case of [Theorem 16.11](#), every connected cubic graph G on n vertices has domination number $\gamma(G) \leq \frac{3}{8}n$. [Reed \[100\]](#) conjectured that this upper bound can be improved to $\gamma(G) \leq \lceil \frac{n}{3} \rceil$. [Kostochka and Stodolsky \[77\]](#), however, disproved this conjecture by constructing a connected cubic graph G on 60 vertices with $\gamma(G) = 21$ and presented a sequence $\{G_k\}_{k=1}^{\infty}$ of connected cubic graphs for which

$$\lim_{k \rightarrow \infty} \frac{\gamma(G_k)}{|V(G_k)|} \geq \frac{8}{23} = \frac{1}{3} + \frac{1}{69}.$$

[Kelmans \[74\]](#) constructed a smaller counter example (with 54 vertices) to [Reed's](#) conjecture and an infinite sequence of 2-connected cubic graphs H_k for which

$$\lim_{k \rightarrow \infty} \frac{\gamma(H_k)}{|V(H_k)|} \geq \frac{1}{3} + \frac{1}{60}.$$

Sasha (Alexandr) Kostochka was born in the city of Chelyabinsk, Russia in 1951. He obtained a doctorate in mathematics from the Sobolev Institute of Mathematics at the Academy of Sciences of the USSR in 1978. He currently is a professor in the Department of Mathematics at the University of Illinois at Urbana-Champaign. He is a world-leading researcher in graph theory and is best known for his significant contributions to colourings in graphs, extremal graph theory, and graph minors. He is a prolific author.



Biographic note 75: [Alexandr Kostochka \(1951–present\)](#)

[Kostochka and Stodolsky \[78\]](#) subsequently proved that the two nonplanar cubic graphs $G_{16.2}$ and $G_{16.3}$ shown in [Figure 16.5](#) are the only connected cubic graphs that achieve the three-eights bound in [Reed's Theorem 16.11](#) by showing that if G is a connected cubic graph of order $n \geq 10$, then $\gamma(G) \leq \frac{4}{11}n$. [Kostochka and Stocker \[76\]](#) improved the [Kostochka-Stodolsky](#) $\frac{4}{11}$ -upper bound on the domination number to a $\frac{5}{14}$ -upper bound.

Theorem 16.12 *If G is a connected cubic graph of order $n \geq 10$, then $\gamma(G) \leq \frac{5}{14}n$.*

[Kostochka and Stocker \[76\]](#) further showed that the bound $\lfloor \frac{5}{14}n \rfloor$ in [Theorem 16.12](#) is sharp for $8 < n \leq 18$.

Let $\mathcal{G}_{\text{cubic}}^n$ denote the family of all connected cubic graphs of order n . As a consequence of the above results we have that

$$0.35 = \frac{1}{3} + \frac{1}{60} \leq \sup_{G \in \mathcal{G}_{\text{cubic}}^n} \left(\lim_{n \rightarrow \infty} \frac{\gamma(G)}{n} \right) \leq \frac{5}{14} = \left(\frac{1}{3} + \frac{1}{42} \right) \approx 0.357142.$$

It remains an open problem, however, to determine what this supremum is. Indeed, the problem of determining a sharp upper bound on the domination number of a connected cubic graph in terms of its order is one of the major outstanding problems in domination theory.

Bounds in terms of maximum degree

Bounds on the domination number of a graph can also be given in terms of the maximum degree of the graph. The lower bound of the following theorem is due to Walikar *et al.* [127], while the upper bound is due to Berge [6].

Theorem 16.13 *If G is a graph of order n with maximum degree Δ , then*

$$\left\lceil \frac{n}{1 + \Delta} \right\rceil \leq \gamma(G) \leq n - \Delta.$$

Proof Let $G = (V, E)$ be a graph of order n and let $\Delta(G) = \Delta$. We first consider the lower bound. Let D be a γ -set of G . Then, $V \setminus D \subseteq \bigcup_{v \in D} N(v)$, implying that $|V \setminus D| \leq \Delta \cdot |D|$. Consequently, $n - \gamma(G) \leq \Delta\gamma(G)$, and so $\gamma(G) \geq \lceil n/(1 + \Delta) \rceil$. To show that $\gamma(G) \leq n - \Delta$, let v be a vertex of G of maximum degree Δ . Then, $V \setminus N(v)$ is a dominating set of cardinality $n - \Delta$. ■

Let G be a graph of order n with domination number equal to one, and so G contains a vertex of degree $n - 1$. Then the corona $\text{cor}(G)$ of G is a graph of order $2n$ with maximum degree $\Delta(\text{cor}(G)) = n$ and domination number $\gamma(\text{cor}(G)) = n$. This shows that the upper bound in [Theorem 16.13](#) is attainable. Furthermore, it follows from the proof of [Theorem 16.13](#) that if $\gamma(G) = n/(1 + \Delta)$, then the sets $N(v)$, where $v \in D$, are pairwise disjoint and $d(v) = \Delta$ for each $v \in D$. Thus, V can be partitioned into subsets V_1 and V_2 with $\gamma(G) = |V_1| \leq |V_2|$ satisfying the following three conditions: (i) V_1 is an independent set, (ii) each vertex of V_2 is adjacent to a unique vertex of V_1 , and (iii) each vertex of V_1 has degree Δ .

Recall from [Chapter 12](#) that $\kappa(G)$ denotes the (vertex) connectivity of G . Since $\kappa(G) \leq \Delta(G)$ for every graph G , we have the following upper bound on the domination number in terms of the (vertex) connectivity of the graph, which was first observed by Walikar *et al.* [127].

Corollary 16.14 *If G is a graph of order n , then $\gamma(G) \leq n - \kappa(G)$.*

If G is a complete graph or an empty graph, then the bound in [Corollary 16.14](#) is attained.

Vizing's edge bound

Vizing [122] showed that if a graph of a given order has sufficiently many edges, then it is guaranteed to have a dominating set of some specified cardinality.

Theorem 16.15 *If G is a graph of order n and size m with domination number $\gamma \geq 2$, then $m \leq (n - \gamma)(n - \gamma + 2)/2$.*

Proof We proceed by induction on the order n . If $n = 2$, then $G = \bar{K}_2$ and the inequality is satisfied. This establishes the base case. Assume, as induction hypothesis, that $n \geq 3$ and that every graph of order less than n and with domination number at least 2 satisfies the inequality, and let G be an (n, m) -graph with domination number $\gamma \geq 2$.

Assume first that $\gamma \geq 3$. Let v be a vertex of G of maximum degree Δ . Then, by Theorem 16.13, $|N(v)| = d_G(v) = \Delta \leq n - \gamma$. Thus, we may write $|N(v)| = \Delta = n - \gamma - r$ where $0 \leq r \leq n - \gamma - 1$. Let $S = V \setminus N[v]$. Then $|S| = \gamma + r - 1$. If $u \in N(v)$, then the set $(S \setminus N(u)) \cup \{u, v\}$ is a dominating set of G , and therefore $|S \setminus N(u)| + 2 \geq \gamma$. Thus, $\gamma + r - 1 - |N(u) \cap S| + 2 \geq \gamma$, and so $|N(u) \cap S| \leq r + 1$ for each vertex $u \in N(v)$. Hence, the number N (say) of edges between $N[v]$ and S is at most $\Delta(r + 1)$.

Let G_S denote the subgraph $G[S]$ induced by S , and let D_S be a γ -set of G_S . Then $D_S \cup \{v\}$ is a dominating set of G . Hence, $\gamma \leq |D_S \cup \{v\}|$, and so $\gamma(G_S) = |D_S| \geq \gamma - 1 (\geq 2)$. By the induction hypothesis, the number of edges in G_S is

$$\begin{aligned} m(G_S) &\leq (|S| - \gamma(G_S))(|S| - \gamma(G_S) + 2)/2 \\ &\leq (\gamma + r - 1 - (\gamma - 1))(\gamma + r - 1 - (\gamma - 1) + 2)/2 \\ &= r(r + 2)/2. \end{aligned}$$

Therefore, the number of edges m in G satisfies

$$\begin{aligned} 2m &= \sum_{u \in N[v]} d_G(u) + \sum_{u \in S} d_G(u) \\ &\leq \Delta(\Delta + 1) + (2m(G_S) + N) \\ &\leq \Delta(\Delta + 1) + r(r + 2) + \Delta(r + 1) \\ &= \Delta(\Delta + 1) + (n - \gamma - \Delta)(n - \gamma - \Delta + 2) + \Delta(n - \gamma - \Delta + 1) \\ &= (n - \gamma)(n - \gamma + 2) - \Delta(n - \gamma - \Delta) \\ &\leq (n - \gamma)(n - \gamma + 2). \end{aligned}$$

Equivalently, $m \leq (n - \gamma)(n - \gamma + 2)/2$, as desired. The result also holds when $\gamma = 2$, since adding an isolated vertex to G yields a graph G' with $\gamma(G') \geq 3$, $m(G') = m(G)$, $\Delta(G') = \Delta(G)$ and $n(G') - \gamma(G') = n(G) - \gamma(G)$. ■

Vizing [122] constructed the following family of graphs $G_{n,\gamma}$ of order n with domination number $\gamma \geq 2$ that have size $m(G_{n,\gamma}) = \lfloor \frac{1}{2}(n - \gamma)(n - \gamma + 2) \rfloor$.

If $\gamma = 2$, then let $G_{n,2}$ be the graph obtained as follows: For n even, let $G_{n,2}$ be obtained from a complete graph K_n by removing the edges of a perfect matching. For n odd, let $G_{n,2}$ be obtained from a complete graph K_n by removing a set of edges that induce $P_3 \cup ((n - 3)/2)K_2$ (*i.e.* removing an almost perfect matching). Then

$$m(G_{n,2}) = \binom{n}{2} - \left\lceil \frac{n}{2} \right\rceil = \left\lfloor \frac{n(n - 2)}{2} \right\rfloor = \left\lfloor \frac{(n - \gamma)(n - \gamma + 2)}{2} \right\rfloor.$$

Thus, for this graph $G_{n,2}$ the required equality holds. If $\gamma > 2$, then let $G_{n,\gamma}$ be the graph obtained by adding to the graph $G_{n-\gamma+2,2}$ a set of $\gamma - 2$ isolated vertices. Then this graph has domination number γ . Furthermore,

$$m(G_{n,\gamma}) = m(G_{n-\gamma+2,2}) = \left\lfloor \frac{1}{2}(n-\gamma+2-2)(n-\gamma+2) \right\rfloor = \left\lfloor \frac{1}{2}(n-\gamma)(n-\gamma+2) \right\rfloor.$$

Vizing's result in [Theorem 16.15](#) has been generalised by Fulman [40] and others. If we restrict the graph to being connected, then this result has been improved by Sanchis [102, 103]. Rautenbach [97] showed that the square dependence on n and γ in Vizing's result reduces to a linear dependence on n, γ and the maximum degree Δ .

Dieter Rautenbach was born in Solingen, Germany in 1972. He obtained a doctorate in mathematics from RWTH Aachen in 1998, and his Habilitation in mathematics in March 2002 from the same university. He held his first professorship at the Forschungsinstitut für Diskrete Mathematik of Bonn University from May 2003 until August 2006. Thereafter, he moved to the Institut für Mathematik of TU Ilmenau as a professor for mathematics from September 2006 until September 2010. In October 2010, he became the director of the Institut für Optimierung und Operations Research at Ulm University. He is best known for his significant contributions to cycles, matchings, independence and domination in graphs. He is a prolific researcher, having co-authored many journal papers.



Biographic note 76: Dieter Rautenbach (1972–present)

The family of graphs $G_{n,\gamma}$ constructed by Vizing all have maximum degree $\Delta = n - \gamma$. Hence the result of [Theorem 16.15](#) is sharp when $\Delta = n - \gamma$. Sanchis [102] showed that the bound of [Theorem 16.15](#) can be improved slightly if $\Delta < n - \gamma$.

Theorem 16.16 *If G is a graph of order n and size m with domination number $\gamma \geq 2$ and with maximum degree Δ satisfying $\Delta \leq n - \gamma - 1$, then $m \leq (n - \gamma)(n - \gamma + 1)/2$.*

Proof For $1 \leq \Delta \leq n - \gamma - 1$, let $f(\Delta) = (n - \gamma)(n - \gamma + 2) - \Delta(n - \gamma - \Delta)$. From the proof of [Theorem 16.15](#), we have $2m \leq f(\Delta)$. The parabolic function $f(\Delta) = \Delta^2 - (n - \gamma)\Delta + (n - \gamma)(n - \gamma + 2)$ achieves its maximum value at one of its endpoints, namely $\Delta = 1$ or $\Delta = n - \gamma - 1$. Since $f(1) = f(n - \gamma - 1) = (n - \gamma)(n - \gamma + 1) + 1$, which is odd, and since $m \leq \lfloor \frac{1}{2}f(\Delta) \rfloor$, we have $m \leq \frac{1}{2}(n - \gamma)(n - \gamma + 1)$, as desired. ■

As an immediate corollary of [Vizing's theorem](#), we have an upper bound for the domination number of a graph in terms of its order and size. A corresponding lower bound was given by Berge [6].

Theorem 16.17 *If G is a graph of order n and size m with domination number γ , then*

$$n - m \leq \gamma \leq n + 1 - \sqrt{1 + 2m}.$$

Furthermore, $\gamma = n - m$ if and only if each component of G is a star.

Proof By Vizing's Theorem 16.15, we have $(n - \gamma)(n - \gamma + 2) - 2m \geq 0$, or, equivalently, $(n - \gamma)^2 + 2(n - \gamma) - 2m \geq 0$. Therefore, since $n - \gamma \geq 0$, we have $n - \gamma \geq -1 + \sqrt{1 + 2m}$, or, equivalently, $\gamma \leq n + 1 - \sqrt{1 + 2m}$. This establishes the upper bound. Since $\gamma \geq 1$, the lower bound is immediate for $m \geq n$. Suppose, therefore, that $m < n$. Let G_1, \dots, G_k be the components of G ($k \geq 1$), where G_i is a graph of order n_i and size m_i . Then $m_i \geq n_i - 1$ with equality if and only if G_i is a tree. Now,

$$m = \sum_{i=1}^k m_i \geq \sum_{i=1}^k (n_i - 1) = n - k,$$

and so, $k \geq n - m$ with equality if and only if each component G_i is a tree. Hence,

$$\gamma = \sum_{i=1}^k \gamma(G_i) \geq \sum_{i=1}^k 1 = k \geq n - m.$$

Furthermore, $\gamma = n - m$ if and only if $k = n - m$ (and so, each component G_i is a tree) and $\gamma(G_i) = 1$ for all $i \in [k]$. Thus, $\gamma = n - m$ if and only if each component of G is a star. ■

16.2.3 Vizing's Conjecture

Recall, from Chapter 1, that for a pair of graphs G and H , the **Cartesian product** $G \square H$ of G and H is the graph whose vertex set is $V(G) \times V(H)$. Two vertices (g_1, h_1) and (g_2, h_2) are adjacent in $G \square H$ if either $g_1 = g_2$ and $h_1 h_2$ is an edge in H , or $h_1 = h_2$ and $g_1 g_2$ is an edge in G . The following conjecture was made by Vizing [123] in 1968, after being posed by him as a problem in [121].

Conjecture 16.18 (Vizing's Conjecture) *For every pair of graphs G and H , $\gamma(G \square H) \geq \gamma(G)\gamma(H)$.*

Although Vizing's Conjecture has been proven true for several classes of graphs, the conjecture has yet to be settled for all graphs and is arguably the main open problem in the area of domination theory. Using what is called the *double projection approach*, Clark and Suen [16] made a breakthrough in 2000 by proving that $\gamma(G \square H) \geq \frac{1}{2}\gamma(G)\gamma(H)$ for all graphs G and H . Their idea nicely incorporates the product structure of $G \square H$. The proof we present here is a simplification of that given in [16].

First we need the notion of a fiber of a graph. For a vertex g of G , the subgraph of $G \square H$ induced by the set $\{(g, h) \mid h \in V(H)\}$ is called an **H -fiber** and is denoted by ${}^g H$. Similarly, for $h \in V(H)$, the **G -fiber**, G^h , is the subgraph induced by $\{(g, h) \mid g \in V(G)\}$. We note that all G -fibers are isomorphic to G and that all H -fibers are isomorphic to H .

We will also have need of projection maps from the Cartesian product $G \square H$ to one of the factors G or H , or to a fiber. The **projection to H** is the map $p_H : V(G \square H) \mapsto V(H)$ defined by $p_H(g, h) = h$, while the **projection to G** is the map $p_G : V(G \square H) \mapsto V(G)$ defined by $p_G(g, h) = g$.

Theorem 16.19 *For all graphs G and H , $\gamma(G \square H) \geq \frac{1}{2}\gamma(G)\gamma(H)$.*

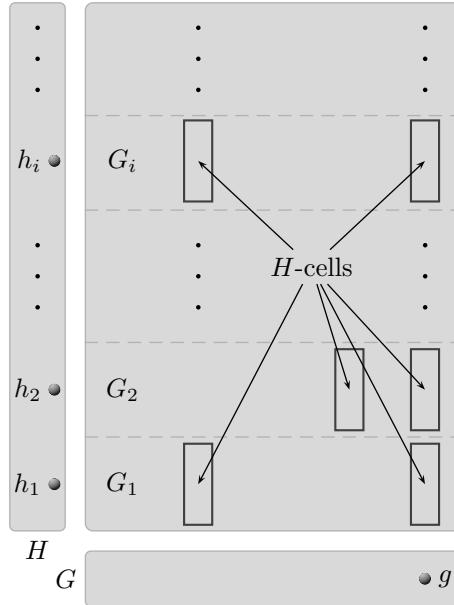


Figure 16.7: Clark-Suen partition.

Proof Let H be a graph with $\gamma(H) = k$ and let $\{h_1, \dots, h_k\}$ be a γ -set of H . Consider a partition $\{\pi_1, \dots, \pi_k\}$ of $V(H)$ chosen so that $h_i \in \pi_i$ and $\pi_i \subseteq N[h_i]$ for each $i \in [k]$. Let G_i be the subgraph of $G \square H$ induced by $V(G) \times \pi_i$. For a vertex g of G , the set of vertices $\{g\} \times \pi_i$ is called an **H -cell**, as illustrated in Figure 16.7.

Let D be a γ -set of $G \square H$ and let $i \in [k]$. If a vertex in an H -cell $\{g\} \times \pi_i$ is not dominated by D from within the H -fiber ${}^g H$, then it is adjacent to a vertex of D in G_i that does not belong to ${}^g H$. But then the projection $p_G(D \cap V(G_i))$ dominates the vertex g in G . Let n_i denote the number of H -cells in G_i such that all vertices from the H -cell $\{g\} \times \pi_i$ are dominated by D from within the corresponding H -fiber. For each such H -cell $\{g\} \times \pi_i$ that is dominated by D from within ${}^g H$, we add the vertex g to the set of vertices in $p_G(D \cap V(G_i))$. The resulting set forms a dominating set of G of cardinality at most $|D \cap V(G_i)| + n_i$. Hence, $|D \cap V(G_i)| + n_i \geq \gamma(G)$ and thus, summing over all $i \in [k]$, we have

$$|D| + \sum_{i=1}^k n_i \geq \gamma(G)\gamma(H). \quad (16.3)$$

On the other hand, let g be an arbitrary vertex of G . For each H -cell $\{g\} \times \pi_i$ that is dominated by D from within ${}^g H$, the vertices in π_i are dominated by the projection $p_H(D \cap V({}^g H))$ in H . For each H -cell $\{g\} \times \pi_i$ that is not dominated by D from within ${}^g H$, we add the vertex h_i to the set of vertices in $p_H(D \cap V({}^g H))$. Since the vertex h_i dominates the vertices of π_i in H , the resulting set forms a dominating set of H . If m_g denotes the number of H -cells in ${}^g H$ that are dominated by D from within ${}^g H$, this resulting dominating set has cardinality $|D \cap V({}^g H)| + (k - m_g)$. Hence, $k = \gamma(H) \leq |D \cap V({}^g H)| + (k - m_g)$, or, equivalently, $m_g \leq |D \cap V({}^g H)|$.

Summing over all vertices g in G , we have

$$|D| \geq \sum_{g \in V(G)} m_g. \quad (16.4)$$

The quantities $\sum_{i=1}^k n_i$ and $\sum_{g \in G} m_g$, however, both count the number of H -cells that are dominated by D from within the corresponding H -fiber, and so

$$\sum_{i=1}^k n_i = \sum_{g \in V(G)} m_g. \quad (16.5)$$

Since $|D| = \gamma(G \square H)$, the inequalities (16.3), (16.4) and (16.5) therefore yield the bound $2\gamma(G \square H) \geq \gamma(G)\gamma(H)$. ■

The factor $\frac{1}{2}$ of Theorem 16.19 comes from the double counting of the vertices of the γ -set D of $G \square H$. Aharoni and Szabó [1] modified this approach to settle the conjecture for chordal graphs, where a graph is **chordal** if each of its cycles of length four or more has a **chord**, which is an edge joining two vertices that are not adjacent in the cycle.

Theorem 16.20 *Chordal graphs satisfy Vizing's Conjecture.*

In 2010, Suen and Tarr [110] presented the following improvement of Theorem 16.19.

Theorem 16.21 *For all graphs G and H ,*

$$\gamma(G \square H) \geq \frac{1}{2}\gamma(G)\gamma(H) + \frac{1}{2}\min\{\gamma(G), \gamma(H)\}.$$

In 2019, Zerbib [129] further strengthened the result of Theorem 16.21.

Theorem 16.22 ([129]) *For all graphs G and H ,*

$$\gamma(G \square H) \geq \frac{1}{2}\gamma(G)\gamma(H) + \frac{1}{2}\max\{\gamma(G), \gamma(H)\}.$$

The double projection approach of Clark and Suen can also be applied to claw-free graphs. As a consequence of a stronger result in [9], we have the following lower bound on the domination number for products of claw-free graphs with arbitrary graphs.

Theorem 16.23 *If G is a claw-free graph, then for any graph H without isolated vertices, $\gamma(G \square H) \geq \frac{1}{2}\gamma(G)(\gamma(H) + 1)$.*

For a detailed discussion of Vizing's Conjecture, we refer the reader to Hartnell and Rall [50], and to the survey papers on Vizing's Conjecture in [9, 10].

16.2.4 Nordhaus-Gaddum type bounds

In 1972, Jaeger and Payan [71] established the first Nordhaus-Gaddum type result on the domination number. The proof presented below is based on that of Cockayne [17].

Douglas Rall was born on 1949 in West Union, Iowa. He obtained a doctorate in mathematics at the University of Iowa in 1976 under the supervision of Frank Kosier. He has been a faculty member at Furman University in Greenville, South Carolina since 1976. In 1990, he was awarded the Alester B Furman, Jr & Janie Earle Furman Award for Meritorious Teaching. He served as chair of the Mathematics Department during the period 1998–2002 and occupied the Furman University Faculty Chair during the period 1999–2001. He is known for his significant contributions to domination theory, and in particular for his work on [Vizing's Conjecture](#) related to the domination number of the Cartesian product of graphs. He has published many journal papers and is co-author of the book titled *Topics in graph theory: Graphs and their cartesian product*, published in 2008 by CRC Press.



Biographic note 77: Douglas Rall (1949–present)

Theorem 16.24 *Let G be a graph of order $n \geq 2$. Then the following holds:*

- (i) $3 \leq \gamma(G) + \gamma(\bar{G}) \leq n + 1$, and
- (ii) $2 \leq \gamma(G)\gamma(\bar{G}) \leq n$.

Proof The lower bounds in (i) and (ii) follow immediately from the observation that if $\gamma(G) = 1$ or $\gamma(\bar{G}) = 1$, then $\gamma(\bar{G}) \geq 2$ or $\gamma(G) \geq 2$, respectively. That these lower bounds are sharp may be seen by considering the graph $G = K_{1,n-1}$ with $\gamma(G) = 1$ and $\gamma(\bar{G}) = 2$.

Next we verify the upper bound in (i). Let $G = (V, E)$ be a graph of order n . If G has an isolated vertex, then $\gamma(G) \leq n$ and $\gamma(\bar{G}) = 1$, while if \bar{G} has an isolated vertex, then $\gamma(G) = 1$ and $\gamma(\bar{G}) \leq n$. In both cases, $\gamma(G) + \gamma(\bar{G}) \leq n + 1$. On the other hand, if G and \bar{G} have no isolated vertices, then, by [Corollary 16.7](#), $\gamma(G) \leq \frac{n}{2}$ and $\gamma(\bar{G}) \leq \frac{n}{2}$, and so $\gamma(G) + \gamma(\bar{G}) \leq n$. That this bound is sharp may be seen by taking $G = K_n$ or $\bar{G} = K_n$.

It remains to verify the upper bound in (ii). The upper bound is immediate if $\gamma(G) = 1$. Hence, in what follows we assume that $\gamma = \gamma(G) \geq 2$. Let $D = \{v_1, \dots, v_\gamma\}$ be a γ -set of G . Partition V into γ subsets V_1, \dots, V_γ such that (a) $v_i \in V_i$ and all vertices in V_i are dominated by v_i in G , and (b) the sum over all integers $i \in [k]$ of the number of vertices in V_i adjacent to all other vertices in V_i is a maximum.

We now show that each set V_i , where $i \in [k]$, is a dominating set in \bar{G} . If this is not the case, then there exists a vertex x in V_j that is adjacent to no vertex of V_i in \bar{G} for some i and j . Thus, the vertex x is adjacent in G to every vertex of V_i . If $x = v_j$, then $D \setminus \{v_i\}$ is a dominating set of G of cardinality less than γ , which is impossible. Hence, $x \in V_j \setminus \{v_j\}$. If x is adjacent to every other vertex of V_j in G , then $D \setminus \{v_i, v_j\} \cup \{x\}$ is a dominating set of G of cardinality less than $\gamma(G)$, which is again impossible. Therefore, x is adjacent in G to every vertex of V_i but not to every vertex of V_j . But then removing the vertex x from V_j and placing it in V_i produces a new partition that has more vertices adjacent to all others in their set than does the original partition, producing a contradiction. Therefore each set V_i

Sandi Klavžar was born in 1962 in Ljubljana, Slovenia. He obtained a doctorate from the University of Ljubljana in 1990 and afterwards held a position at the University of Maribor. Since 2008 he has been professor of mathematics at the University of Ljubljana and in part at the University of Maribor. He is an expert in graph products, metric and chemical graph theory, graph domination, and the Tower of Hanoi puzzle. He has co-authored many journal papers, and is a co-author of several books including *Topics in Graph Theory*, *Handbook of Product Graphs*, *The Tower of Hanoi—Myths and Maths*. His extensive editorial work includes associate editorship of Discrete Applied Mathematics.



Biographic note 78: Sandi Klavžar (1962–present)

is a dominating set in \bar{G} , and so $|V_i| \geq \gamma(\bar{G})$ for all i . Hence,

$$n = \sum_{i=1}^{\gamma(G)} |V_i| \geq \gamma(G)\gamma(\bar{G}). \quad \blacksquare$$

[Payan and Xuong](#) [92] characterised those graphs G of order n for which $\gamma(G)\gamma(\bar{G}) = n$.

Theorem 16.25 *If G is a graph of order $n \geq 2$, then $\gamma(G)\gamma(\bar{G}) = n$ if and only if G is one, or the complement of one, of the following graphs: K_n , the disjoint union of 4-cycles C_4 and the corona $\text{cor}(H)$ of H for any graph H , and $K_3 \square K_3$ (the Cartesian product of two triangles).*

Before we present the next result, recall from [Chapter 1](#) that a graph G is **factorable** into the factors G_1, G_2, \dots, G_t if these factors are pairwise edge-disjoint and

$$\bigcup_{i=1}^t E(G_i) = E(G).$$

If G is factored into G_1, G_2, \dots, G_t , then we write $G = G_1 \oplus G_2 \oplus \dots \oplus G_t$, which is called a **factorisation** of G . The upper bounds of [Theorem 16.24](#) can be restated as follows.

Theorem 16.26 *Let $n \geq 2$ be an integer. If $K_n = G_1 \oplus G_2$, then $\gamma(G_1) + \gamma(G_2) \leq n + 1$ and $\gamma(G_1)\gamma(G_2) \leq n$.*

Corresponding upper bounds for three factors were obtained by [Goddard et al.](#) [47].

Theorem 16.27 *Let $n \geq 3$ be an integer. If $K_n = G_1 \oplus G_2 \oplus G_3$, then $\gamma(G_1) + \gamma(G_2) + \gamma(G_3) \leq 2n + 1$.*

Wayne Goddard was born in Durban, South Africa in 1965. He obtained doctorates in mathematics from the University of Natal, Durban in 1989, and from the Massachusetts Institute of Technology, Cambridge (MA) in 1992. He started his academic career at the University of Pennsylvania in the Department of Mathematics, before returning to South Africa where he held a professorship at the University of Natal, Durban, in its Computer Science Department. In 2002, he moved to Clemson University in the United States, where he currently is jointly in the School of Computing and the Department of Mathematical Sciences. He is managing editor of the journal *Discrete Mathematics*. He is best known for his significant contributions to many areas of graph theory, including coloring and domination in graphs, as well as graph algorithms. He is a prolific researcher, having co-authored many journal papers in addition to writing two textbooks, including *Introducing the Theory of Computation*.



Biographic note 79: Wayne Goddard (1965–present)

Proof We show first that $\gamma(G_2) + \gamma(G_3) \leq \gamma(G_2 \oplus G_3) + n$. Let D be a γ -set of $G_2 \oplus G_3$. For $i \in \{2, 3\}$, let D_i be the set of vertices in G_i not dominated by the set D . Then, $D \cup D_i$ is a dominating set of G_i for each $i \in \{2, 3\}$. Furthermore, $D_2 \cup D_3 \subseteq V(K_n) \setminus D$. Since the vertices not dominated by the set D in G_2 are disjoint from those not dominated by D in G_3 , $D_2 \cap D_3 = \emptyset$. Hence, $|D_2| + |D_3| = |D_2 \cup D_3| \leq n - |D|$, and so $|D| + |D_2| + |D_3| \leq n$. Thus, $\gamma(G_2) + \gamma(G_3) \leq |D \cup D_2| + |D \cup D_3| = 2|D| + |D_2| + |D_3| \leq |D| + n = \gamma(G_2 \oplus G_3) + n$. Since $G_2 \oplus G_3 = \bar{G}_1$, we have by [Theorem 16.26](#) $\gamma(G_1) + \gamma(G_2 \oplus G_3) \leq n + 1$. Therefore, $\gamma(G_1) + \gamma(G_2) + \gamma(G_3) \leq \gamma(G_1) + \gamma(G_2 \oplus G_3) + n \leq 2n + 1$. ■

The following theorem is also due to [Goddard et al.](#) [47].

Theorem 16.28 *Let $n \geq 3$ be an integer. If $K_n = G_1 \oplus G_2 \oplus G_3$, then the maximum value of the product $\gamma(G_1)\gamma(G_2)\gamma(G_3)$ is $n^3/27 + \mathcal{O}(n^2)$.*

That is, there exist constants c_1 and c_2 such that the maximum triple product always lies between $n^3/27 + c_1n^2$ and $n^3/27 + c_2n^2$. To illustrate a realisation of the upper bound in [Theorem 16.28](#) via a general construction, let $n \geq 12$ be a multiple of 3 and let (A, B, C) denote a partition of the vertex set of $G = K_n$ with $|A| = |B| = |C| = \frac{n}{3} \geq 4$. We now construct factors G_1 , G_2 and G_3 as follows. The sets A , B and C are the sets of vertices isolated in G_1 , G_2 and G_3 , respectively. Furthermore, G_1 has all the edges between B and C , G_2 has all the edges between A and C , and G_3 has all the edges between A and B . Finally, each of $G_1[B]$, $G_1[C]$, $G_2[A]$, $G_2[C]$, $G_3[A]$, and $G_3[B]$ has no isolated vertex. Therefore, $\gamma(G_1) = |A| + 2 = (n+6)/3$, $\gamma(G_2) = |B| + 2 = (n+6)/3$, and $\gamma(G_3) = |C| + 2 = (n+6)/3$. This construction consequently yields, as a lower bound, the maximum product of three positive integers summing to $n+6$. This shows that the maximum product is at least $n^3/27 + 2n^2/3$.

16.2.5 NP-completeness of the domination problem

The decision problem $D_{\text{domset}}(G, k)$, associated with the domination number takes the following form: Given a graph G and a positive integer k , decide whether or not G has a dominating set of size at most k .

[Garey and Johnson](#) [41] were the first to show that $D_{\text{domset}}(G, K)$, is NP-complete by constructing a polynomial-time reduction from the prototype NP-complete problem 3-SAT (described in [Chapter 3](#)) to $D_{\text{domset}}(G, k)$. Consider a set $X = \{x_1, \dots, x_n\}$ of boolean variables, and a set $\mathcal{C} = \{C_1, \dots, C_m\}$ of 3-element sets, called clauses, where each clause $C \in \mathcal{C}$ contains three distinct occurrences of either a variable $x \in X$ or its complement \bar{x} conjoined by the **or**-operator. The decision problem 3-SAT asks whether \mathcal{C} is satisfiable; that is, whether there is an assignment of the values **true** and **false** to the boolean variables in X such that at least one variable in each clause in \mathcal{C} is assigned the value **true**.

Theorem 16.29 $D_{\text{domset}}(G, k)$ is NP-complete.

Proof Let $G = (V, E)$ be a graph. It is obvious that $D_{\text{domset}}(G, k)$ is a member of the class NP since, given an arbitrary set $D \subseteq V$ with $|D| \leq k$, we can verify in polynomial time whether D is a dominating set.

Given an instance \mathcal{C} of 3-SAT, we construct a graph $G_{\mathcal{C}}$ whose order is polynomially bounded in the size of \mathcal{C} such that \mathcal{C} is satisfiable if and only if $G_{\mathcal{C}}$ has a dominating set of cardinality at most k .

For every boolean variable x occurring in \mathcal{C} we introduce a copy of K_3 , which we call G_x , that contains two specified vertices labelled x and \bar{x} . Furthermore, for every clause C of \mathcal{C} we introduce a copy of K_1 and we label its vertex by C . If the literal x occurs in clause C , we add an edge joining the specified vertex x in G_x to the specified vertex C . (See, for example, [Figure 16.8](#) where $\mathcal{C} = \{C_1, C_2\}$ with $C_1 = \{x_1, \bar{x}_2, x_3\}$ and $C_2 = \{x_1, x_3, x_4\}$.) Let $G_{\mathcal{C}}$ denote the resulting graph.

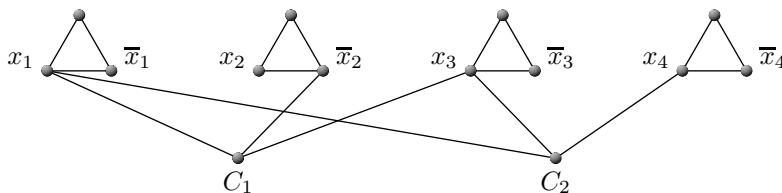


Figure 16.8: A reduction from 3-SAT to $D_{\text{domset}}(G, k)$.

Let \mathcal{C} use n boolean variables and contain m clauses. Suppose \mathcal{C} is satisfiable and consider a satisfying truth assignment for \mathcal{C} . We now identify a set S of vertices of G as follows. For each variable $x \in \mathcal{C}$ assigned the value **true**, we put x in S . For each variable $x \in \mathcal{C}$ assigned the value **false**, we put \bar{x} in S . Each triangle in G then contains exactly one vertex in S . Furthermore, each clause vertex C is dominated by at least one vertex of S since, by assumption, each clause contains at least one variable whose value is assigned **true** and, by construction, the corresponding vertex belongs to S . Hence, S is a dominating set of G of cardinality n .

Conversely, we assume now that G has a dominating set S of cardinality at most n . We show that \mathcal{C} has a corresponding satisfying truth assignment. In order to dominate the vertex in a triangle G_x distinct from x and \bar{x} , the triangle G_x

contains at least one vertex in S . Therefore, $|S| \geq n$. Consequently, $|S| = n$ and each triangle contains exactly one vertex in S . Thus, at most one of the two vertices x and \bar{x} in every triangle G_x can be in S . Furthermore, the set S contains no clause vertex. Since S is a dominating set, each clause vertex is therefore dominated by at least one vertex that belongs to a triangle. Hence, the vertices in S corresponding to literals indicate a satisfying truth assignment for \mathcal{C} . The truth value of a variable x for which neither x nor \bar{x} is in S can be set arbitrarily.

Finally we show that the construction for creating an instance of $D_{\text{domset}}(G, k)$ from an instance of 3-SAT can be performed in polynomial time. The length of an instance of 3-SAT is $3n + m$ since \mathcal{C} is specified by m sets of size three plus n variables. The graph G constructed from this instance \mathcal{C} has order $3n + m$ and size $3n+3m$. Hence the order of G , which is at most a constant times the cardinality of \mathcal{C} , is polynomially bounded in the size of \mathcal{C} . ■

[Theorem 16.29](#) suggests that we should not expect to find a polynomial time algorithm to determine the domination number of an arbitrary graph. If we wish to find a polynomial time algorithm for computing the domination number of a graph, we will need to restrict the instances to certain classes of graphs. There are, however, relatively few such classes which allow us to compute the domination number in polynomial time. Such classes include, among others, the well-studied families of trees, block graphs, interval graphs, cographs, strongly chordal graphs, and permutation graphs. $D_{\text{domset}}(G, k)$ remains NP-complete where instances are restricted to, among others, bipartite graphs, split graphs and chordal graphs. For an excellent discussion on the complexity of the graph domination problem, we refer the reader to Chapter 12 in [57].

16.2.6 A polynomial-time tree domination algorithm

As indicated in the previous paragraph, a wide variety of polynomial time algorithms for computing the domination number for certain graph classes have been proposed. Here we present a linear algorithm due to [Mitchell et al.](#) [89] for computing the domination number of an arbitrary tree.

For ease of presentation, and without loss of generality, we consider rooted trees. Recall, from [Chapter 5](#), that in a **rooted tree** we distinguish one vertex r , called the **root**, from all the other vertices in the tree. For each vertex $v \neq r$ of T , the **parent** of v is the neighbour of v on the unique r - v path, while a **child** of v is any other neighbour of v . A **descendant** of v is a vertex u such that the unique r - u path contains v . Thus, every child of v is a descendant of v . We let $C[v]$ and $D[v]$ denote the set of children and descendants, respectively, of v , and we define $D(v) = D[v] \cup \{v\}$. The **maximal subtree** at v is the subtree of T induced by $D(v)$, and is denoted by T_v . Recall that a **leaf** of T is a vertex of degree 1, while a **support vertex** of T is a vertex adjacent to a leaf.

Recall, furthermore, that we commonly draw the root of a rooted tree T at the top of the graphical representation of T with the remaining vertices at appropriate horizontal levels below r , depending on their distances from r . Given a rooted tree T with the root labelled vertex 1 and with $V(T) = [n]$, we represent T by a data structure called a **parent array** in which the parent of a vertex labelled i is given by $\text{Parent}[i]$ with the special convention that $\text{Parent}[1] = 0$ (to indicate

that the vertex labelled 1 has no parent). A tree $T_{16.4}$ and its parent array are shown in Figure 16.9.

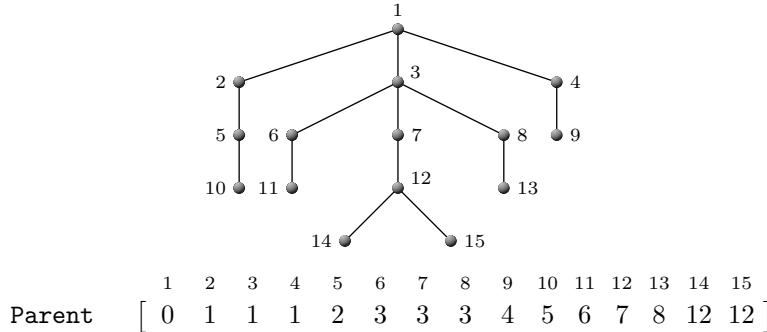


Figure 16.9: A rooted tree $T_{16.4}$ with its parent array.

We are now in a position to present an efficient algorithm for computing the domination number of a rooted tree T . Initially, $S = \emptyset$ and all vertices are labelled **Bound**. As the algorithm progresses, vertices are added to the set S , depending on their labels. When a vertex i is encountered, its current label **Label**[i] together with that of its parent **Parent**[i] are used to possibly relabel **Parent**[i] to **Free** or **Required**. Once a vertex has been labelled **Required** its label does not change again. Upon completion of the algorithm, the set S consists of all vertices labelled **Required**. The algorithm is presented in pseudocode form as Algorithm 36.

Algorithm 36: Tree domination

Input : A rooted tree $T = (V, E)$ rooted at a vertex labelled 1 with $V = [n]$ and represented by an array $\text{Parent}[1 \dots n]$.

Output : A γ -set S of T

```

1  $S \leftarrow \emptyset$ 
2 for  $i = 1$  to  $n$  do
3    $\text{Label}[i] \leftarrow \text{Bound}$ 
4 for  $i = n$  down to 2 do
5   if  $\text{Label}[i] = \text{Bound}$  then
6      $\text{Label}[\text{Parent}[i]] \leftarrow \text{Required}$ 
7   else if  $\text{Label}[i] = \text{Required}$  then
8      $S \leftarrow S \cup \{i\}$ 
9     if  $\text{Label}[\text{Parent}[i]] = \text{Bound}$  then
10       $\text{Label}[\text{Parent}[i]] \leftarrow \text{Free}$ 
11 if ( $\text{Label}[1] = \text{Bound}$ ) or ( $\text{Label}[1] = \text{Required}$ ) then
12    $S \leftarrow S \cup \{1\}$ 

```

Let us illustrate the working of Algorithm 36 by means of an example. Suppose we wish to find a minimum dominating set in the (rooted) tree $T_{16.4}$ in Figure 16.9.

ure 16.9. Initially, we label all vertices as **Bound** in [Step 3](#) of the algorithm; that is, $\text{Label}[i] = \text{Bound}$ for all $i \in [15]$.

We now proceed to [Step 4](#) of the algorithm, and first consider vertex 15. Since $\text{Label}[15] = \text{Bound}$ and $\text{Parent}[15] = 12$, we change the label of vertex 12 from **Bound** to **Required**; that is, $\text{Label}[12] = \text{Required}$. We next consider vertex 14. Since $\text{Label}[14] = \text{Bound}$ and $\text{Parent}[15] = 12$, the label of vertex 12 remains unchanged as **Required**. In a similar way as for vertex 15, when we consider vertex 13, we change the label of its parent, vertex 8, from **Bound** to **Required**; that is, $\text{Label}[8] = \text{Required}$.

We next consider vertex 12. Since $\text{Label}[12] = \text{Required}$, we add vertex 12 to the set S in [Step 8](#) of the algorithm. Thus, currently $S = \{12\}$. Furthermore, since $\text{Parent}[12] = 7$ and $\text{Label}[7] = \text{Bound}$, we change the label of vertex 7 to **Free**; that is, $\text{Label}[7] = \text{Free}$.

Considering vertices 11, 10 and 9 in turn results in the labels 6, 5 and 4 of their parents, respectively, changing to **Required**; that is, after these three steps of the algorithm, we have $\text{Label}[4] = \text{Label}[5] = \text{Label}[6] = \text{Required}$.

We next consider vertex 8. Since $\text{Label}[8] = \text{Required}$, we add vertex 8 to the set S in [Step 8](#) of the algorithm. Thus, currently $S = \{8, 12\}$. Furthermore, since $\text{Parent}[8] = 3$ and $\text{Label}[3] = \text{Bound}$, we change the label of vertex 3 from **Bound** to **Free** in [Step 10](#) of the algorithm; that is, $\text{Label}[3] = \text{Free}$.

We now consider vertex 7. Since its label is **Free**, we immediately move on to the next vertex, namely vertex 6. Since $\text{Label}[6] = \text{Required}$, we add vertex 6 to the set S in [Step 8](#) of the algorithm. Thus, currently $S = \{6, 8, 12\}$. Furthermore, since $\text{Parent}[6] = 3$ and $\text{Label}[3] = \text{Free}$, the label of vertex 3 remains unchanged as **Free** in [Step 10](#) of the algorithm; that is, $\text{Label}[3] = \text{Free}$.

Since $\text{Label}[5] = \text{Required}$, we add vertex 5 to the set S in [Step 8](#) of the algorithm. Thus, currently $S = \{5, 6, 8, 12\}$. Furthermore, since $\text{Parent}[5] = 2$ and $\text{Label}[2] = \text{Bound}$, we change the label of vertex 2 from **Bound** to **Free**; that is, $\text{Label}[2] = \text{Free}$. In a similar way, since $\text{Label}[4] = \text{Required}$ and $\text{Parent}[4] = 1$, we add vertex 4 to the set S and change the label of vertex 1 from **Bound** to **Free**; that is, $\text{Label}[1] = \text{Bound}$. Thus, currently $S = \{4, 5, 6, 8, 12\}$.

Since the label of vertex 3 is **Free**, we move on to the next vertex, namely vertex 2. Since the label of vertex 2 is **Free**, we move on to the next vertex, namely vertex 1. Finally, since the label of vertex 1 is **Free**, we do not add vertex 1 to the set S in [Step 12](#) of the algorithm. The algorithm now terminates and the resulting set $S = \{4, 5, 6, 8, 12\}$ is a minimum dominating set of T .

We now verify the validity of [Algorithm 36](#). We may assume that the vertices of the rooted tree T are labelled $1, \dots, n$ so that for $i < j$, $d(1, i) \leq d(1, j)$, i.e. vertex i is at a level smaller than or equal to that of vertex j .

Theorem 16.30 *If T is a tree, then the set produced by [Algorithm 36](#) is a γ -set of T .*

Proof Let S be the set produced by [Algorithm 36](#). By construction, S is a dominating set of T and therefore $\gamma(T) \leq |S|$. It remains to show that $\gamma(T) = |S|$. Suppose, to the contrary, that $\gamma(T) < |S|$. Among all γ -sets of T , let D be chosen so that the largest labelled vertex $k \in (S \setminus D) \cup (D \setminus S)$ is as small as possible. Thus, $D \cap \{k+1, \dots, n\} = S \cap \{k+1, \dots, n\}$. Let T_k denote the maximal subtree at k ; that is, T_k consists of k and all descendants of k . Moreover, let $D_k = D \cap V(T_k)$ and let $S_k = S \cap V(T_k)$. We consider two cases.

Case 1: $k \in S$. In this case, $k \notin D$. Now every vertex of T_k , except possibly for k , is dominated by D_k . Thus, since $k \notin D$, the set D_k dominates $V(T_k) \setminus \{k\}$. By our choice of k , $D_k = S_k \setminus \{k\}$. Let ℓ be an arbitrary child of k (by construction, the set S contains no leaf, and so the vertex k has at least one child.) Therefore, $k < \ell$. Since ℓ is dominated by D_k , either $\ell \in D_k$ or ℓ is adjacent to a vertex of D_k . If $\ell \in D_k$, then, since $\ell \geq k + 1$, $\ell \in S$ and therefore the vertex ℓ is labelled **Required** in [Algorithm 36](#). If, on the other hand, $\ell \notin D_k$, then ℓ has a child that belongs to D_k . This child of ℓ is a descendant of vertex k and therefore has label greater than k , and so, by our choice of k , also belongs to S . Thus, ℓ has a child that is labelled **Required** in [Algorithm 36](#) and therefore ℓ is labelled **Free** in [Algorithm 36](#). Hence, every child of the vertex k is labelled **Required** or **Free**. This implies that when the vertex k is considered during the execution of the algorithm, its label is either **Free** (if it has a child labelled **Required**) or **Bound** (if every child has label **Free**). Since all vertices $i \in S$ with $i > 1$ are labelled **Required**, it follows that $k = 1$ and that the vertex 1 is labelled **Bound** with every child of vertex 1 having label **Free**. But then the set D does not dominate the vertex 1, a contradiction.

Case 2: $k \in D$. In this case, $k \notin S$. Now every vertex of T_k , except possibly for k , is dominated by S_k . Thus, since $k \notin S$, the set S_k dominates $V(T_k) \setminus \{k\}$. By our choice of k , $S_k = D_k \setminus \{k\}$. Since $|S| < |D|$, it follows that $k > 1$. Let k^* denote the parent of k , and let $D^* = (D \setminus \{k\}) \cup \{k^*\}$. Then D^* is a dominating set of T of cardinality $|D| = \gamma(T)$. But the largest vertex in $(S \setminus D) \cup (D \setminus S)$ is less than k , which contradicts our choice of k .

Since both Cases 1 and 2 produce a contradiction, we conclude that $\gamma(T) \geq |S|$. Consequently, $\gamma(T) = |S|$, and so S is a γ -set of T as required. ■

For a more detailed discussion on domination algorithms, including polynomial time algorithms for computing the domination number of interval graphs and permutation graphs, we refer the reader to Chapter 12 in [\[57\]](#).

16.2.7 The upper domination number

The **upper domination number** $\Gamma(G)$ of a graph G is the maximum cardinality of a minimal dominating set of G . A minimal dominating set of cardinality $\Gamma(G)$ is called a **Γ -set of G** . We observe that if G has no isolated vertex and if v is an arbitrary vertex of G , then $V(G) \setminus \{v\}$ is a dominating set in G , implying that the set $V(G)$ is not a minimal dominating set. Hence we have the following observation.

Observation 16.31 *If G is a graph of order n with no isolated vertex, then $\Gamma(G) \leq n - 1$.*

That the trivial upper bound in [Observation 16.31](#) is sharp, may be seen by considering a star $K_{1,n-1}$ where $n \geq 3$. The set of $n - 1$ leaves in such a star form a minimal dominating set and so $\Gamma(K_{1,n-1}) \geq n - 1$. Hence by [Observation 16.31](#), $\Gamma(K_{1,n-1}) = n - 1$. If, however, we impose a regularity condition on the graph, then the upper domination number can be improved to at most one-half its order. In order to prove this, we consider an edge weight function on a dominating set in a graph, as defined in [\[108\]](#). For disjoint subsets X and Y of vertices in a

graph $G = (V, E)$, we denote the set of edges between X and Y by $G[X, Y]$, or simply $[X, Y]$ if the graph G is clear from the context.

Let S be a dominating set of G . An **edge weight function** of $S \subseteq V$ is defined to be a function $\psi_S : E \mapsto [0, 1]$ that assigns to each edge in $G[S]$ and each edge in $G[V \setminus S]$ a weight of 0 and that assigns to each edge in $[S, V \setminus S]$ a weight in $(0, 1]$ in such a way that for each vertex $v \in V \setminus S$, the weight 1 is shared equally among the edges joining v to S . Thus, if e is an edge joining $v \in V \setminus S$ to S , then $\psi(e) = 1/d_S(v)$, where $d_S(v)$ denotes the number of vertices in S adjacent to v . Since S is a dominating set in G , the sum of the weights of the edges joining each vertex in $V \setminus S$ to S is 1. Hence it follows that

$$\sum_{e \in [S, V \setminus S]} \psi_S(e) = |V \setminus S| = n - |S|. \quad (16.6)$$

Next we define a **vertex weight function** of S , denoted by ϕ_S , that assigns to each vertex $v \in S$ the sum of the weights of the edges that join v to $V \setminus S$. Since every edge in $G[S]$ has weight 0, we have that

$$\phi_S(v) = \sum_{e \in [\{v\}, V \setminus \{v\}]} \psi_S(e) = \sum_{e \in [\{v\}, V \setminus S]} \psi_S(e).$$

Since the number of edges in $[S, V \setminus S]$ with one endpoint in S is the same as the number of edges in $[S, V \setminus S]$ with one endpoint in $V \setminus S$, we can count the edges in $[S, V \setminus S]$ in two ways, which yields the equation

$$\sum_{e \in [S, V \setminus S]} \psi_S(e) = \sum_{v \in S} \phi_S(v). \quad (16.7)$$

Finally, we define a **vertex weight sum** of S , denoted by $\xi(S)$, to be the sum over all vertices in S of the weights assigned by ϕ_S ; that is,

$$\xi(S) = \sum_{v \in S} \phi_S(v).$$

Hence, from (16.6) and (16.7), the relationship

$$\xi(S) = n - |S| \quad (16.8)$$

holds for every dominating set S in the graph G . We are now in a position to prove the following result.

Theorem 16.32 ([108]) *For every regular graph G of order n with no isolated vertex, $\Gamma(G) \leq \frac{n}{2}$.*

Proof Let G be a k -regular graph on n vertices for some integer $k \geq 1$ and let D be a Γ -set of G . We use the edge weight function ψ_D and vertex weight function ϕ_D to count the number of vertices in D relative to n . Recall that if e is an edge joining $v \in V \setminus D$ to D , then $\psi_D(e) = 1/d_D(v)$. Thus, $\frac{1}{k} \leq \psi_D(e) \leq 1$ for every edge $e \in [D, V \setminus D]$.

We show that $\phi_D(v) \geq 1$ for each $v \in D$. Let A be the set of isolated vertices in $G[D]$ and let $B = D \setminus A$. Each vertex $a \in A$ is joined by k edges to $V \setminus D$. Since $\psi_D(e) \geq \frac{1}{k}$ for every edge joining D to $V \setminus D$, we therefore have $\phi_D(a) \geq k(1/k) = 1$ for each $a \in A$. No vertex $b \in B$ is an isolated vertex in $G[D]$, and so $\text{ipn}[b, D] \neq \{b\}$. Since $\text{ipn}[v, D] \in \{\emptyset, \{v\}\}$ for every $v \in D$, it therefore follows that $\text{ipn}[b, D] = \emptyset$. Hence, by [Theorem 16.1](#) it follows that $\text{epn}[b, D] \neq \emptyset$. But every edge that joins b to a vertex in $\text{epn}[b, D]$ is assigned weight 1 under the function ψ , and so $\phi_D(b) \geq 1$ for each $b \in B$. Thus, $\phi_D(v) \geq 1$ for each $v \in D$, and so $\xi(D) \geq |D|$. Recall that $n - |D| = \xi(D)$, by [\(16.8\)](#), and hence $n - |D| \geq |D|$. As a consequence, $\Gamma(G) = |D| \leq \frac{n}{2}$. ■

In order to characterise regular graphs with no isolated vertices and with upper domination number one-half their order, we recall an important class of graphs called circulant graphs, defined in [Chapter 1](#). A **circulant graph** $C_n\langle L \rangle$ with **connection set** $L \subseteq [\lfloor \frac{n}{2} \rfloor]$ is a graph on n vertices in which the i -th vertex is adjacent to the $(i + j)$ -th and $(i - j)$ -th vertices for each j in L , where the vertices are arranged in a circular fashion. More precisely, if $L = \{j_1, \dots, j_r\} \subseteq [\lfloor \frac{n}{2} \rfloor]$, then the circulant graph $C_n\langle L \rangle$ is the graph with vertex set $\{v_0, \dots, v_{n-1}\}$ and edge set $\{v_i v_{i+j} \pmod n \mid i \in [n-1] \text{ and } j \in \{j_1, \dots, j_r\}\}$. For $k \geq 4$ even and $\ell \geq k$ with ℓ even, let $L_{k,\ell} = [\frac{k}{2}-1] \cup \{\frac{\ell}{2}\}$, while for $k \geq 3$ odd and $\ell \geq k$, let $L_{k,\ell} = [(k-1)/2]$. In both cases, the circulant graph $C_\ell\langle L_{k,\ell} \rangle$ is a $(k-1)$ -regular graph on ℓ vertices. For example, the circulant graph $C_8\langle L_{5,8} \rangle = C_8\langle 1, 2 \rangle$ shown in [Figure 16.10\(a\)](#) is a 4-regular graph on eight vertices, while the circulant graph $C_{10}\langle L_{6,10} \rangle = C_{10}\langle 1, 2, 5 \rangle$ shown in [Figure 16.10\(b\)](#) is a 5-regular graph on ten vertices.

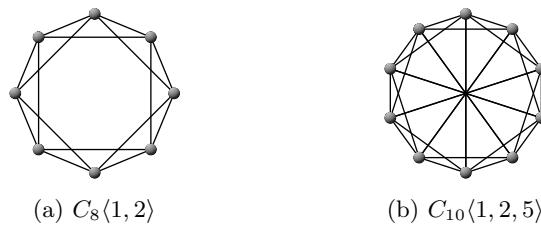


Figure 16.10: Two circulant graphs.

In view of the circulant graphs $C_\ell\langle L_{k,\ell} \rangle$ constructed above, we observe that for every two positive integers k and ℓ , with $\ell \geq k$ and where ℓ is even whenever k is even, there always exist $(k-1)$ -regular graphs on ℓ vertices. Conversely, since every graph has an even number of vertices of odd degree by [Corollary 1.2](#), every $(k-1)$ -regular graph on ℓ vertices satisfies $\ell \geq k$ with ℓ even whenever k is even.

Let \mathcal{G}_{reg} be the family of connected regular graphs constructed as follows. Let $k \geq 1$ and $\ell \geq k$ be arbitrary fixed integers, provided that ℓ is even whenever k is even. As observed earlier, for every such pair of integers k and ℓ we note that there exist $(k-1)$ -regular graphs on ℓ vertices. Let F_1 and F_2 be disjoint $(k-1)$ -regular graphs (not necessarily connected) on ℓ vertices with $V(F_1) = \{u_1, \dots, u_\ell\}$ and $V(F_2) = \{v_1, \dots, v_\ell\}$. Furthermore, let F be the graph obtained from the disjoint union $F_1 \cup F_2$ by joining u_i to v_i for each $i \in [\ell]$. Finally, let \mathcal{G}_{reg} be the family of all graphs thus constructed which are, in addition, connected.

It was shown in [108] that equality is achieved in [Theorem 16.32](#) if and only if every component of G is a bipartite regular graph or belongs to the family \mathcal{G}_{reg} .

As a simple application of [Theorem 16.32](#), we determine the upper domination number of the Petersen graph. As observed earlier, the set $S_3 = \{v_1, v_2, v_3, v_4, v_5\}$ is a minimal dominating set of the Petersen graph $G_{16.1}$ shown in [Figure 16.2](#), and so $\Gamma(G_{16.1}) \geq |S_3| = 5$. Applying [Theorem 16.32](#) to the Petersen graph, which is a 3-regular graph on 10 vertices, we have that $\Gamma(G_{16.1}) \leq 5$. Consequently, $\Gamma(G_{16.1}) = 5$.

We close this section by noting that in 1996, [Nowakowski and Rall](#) [90] made the natural [Vizing](#)-like conjecture for the upper domination number of Cartesian products of graphs. A proof for this conjecture was found by [Brešar](#) [8].

Theorem 16.33 *For any graphs G and H , $\Gamma(G \square H) \geq \Gamma(G)\Gamma(H)$.*

Boštjan Brešar was born in 1968 in Kranj, Slovenia. He obtained a bachelor's degree in applied mathematics at the University of Ljubljana in 1994. He also obtained both a master's in computer science (1998) and a doctorate in mathematics (2000) at the University of Maribor. He started his academic career in 2001 as an assistant professor of mathematics at the University of Maribor, and was rapidly promoted to a full professor of mathematics in 2011. From 2012 to 2016 he served as deputy dean for research and graduate studies in the Faculty of Natural Sciences and Mathematics, University of Maribor. He is known for his significant contributions to domination theory, and in particular for his work on [Vizing's Conjecture](#) related to the domination number of the Cartesian product of graphs. He has published extensively.



Biographic note 80: Boštjan Brešar (1968–present)

- ❖ The reader should now be able to attempt Exercises [16.1–16.9](#).

16.3 The independent domination number

An **independent dominating set**, abbreviated as an ID-set, of G is a set that is both dominating and independent in G . The **independent domination number** of G , denoted by $i(G)$, is the minimum cardinality of an ID-set. Recall that the **independence number** of G , denoted by $\alpha(G)$, is the maximum cardinality of an independent set in G . An ID-set of G of cardinality $i(G)$ is called an *i-set of G* . We adopt a similar notation for other parameters. The independent domination number was introduced by [Cockayne and Hedetniemi](#) in [25].

Recall that the **queen's graph** \mathcal{Q}_8 may be formed from an 8×8 chessboard by taking the squares as the vertices and in which two vertices are adjacent if a queen situated on one of the corresponding squares can occupy the other square in a

single move. The minimum number of mutually nonattacking queens that cover all the squares of a chessboard is the independent domination number $i(\mathcal{Q}_8)$. For the queen's graph \mathcal{Q}_8 , we note that $i(\mathcal{Q}_8) = 7$ while $\gamma(\mathcal{Q}_8) = 5$. Properties of these parameters are explored further in [Project 16.2](#) at the end of the chapter.

For the complete graph, $i(K_n) = 1$, while for a path and cycle, $i(P_n) = i(C_n) = \lceil \frac{n}{3} \rceil$ for all $n \geq 3$. For the complete bipartite graph, $i(K_{r,s}) = \min\{r, s\}$.

16.3.1 The domination inequality chain

If no proper superset of an independent set S is an independent set of G , then S is a **maximal independent set** of G . [Berge \[6\]](#) was the first to observe the following result.

Proposition 16.34 *A set S of vertices in a graph G is an independent dominating set if and only if S is maximal independent in G .*

Proof If S is a maximal independent set of vertices in a graph G , then S is independent and every vertex not in S is adjacent to some vertex of S ; that is, S is an independent dominating set. Conversely, if S is an independent dominating set, then S is independent and every vertex not in S is adjacent to some vertex of S ; that is, S is maximal independent. ■

Therefore, every maximal independent set is a dominating set. This is why the minimum cardinality of a maximal independent set in G is called the independent domination number of G . [Berge \[6\]](#) was also the first to observe the next result.

Proposition 16.35 *Every maximal independent set of vertices in a graph G is a minimal dominating set in G .*

Proof Let S be a maximal independent set of vertices in a graph G . By [Proposition 16.34](#), S is a dominating set. Since S is independent, every vertex of S is adjacent to no other vertex of S , and so $\text{ipn}[v, S] = \{v\}$ for every vertex $v \in S$. Hence, by [Theorem 16.1](#), S is a minimal dominating set. ■

As a consequence of Propositions 16.34 and 16.35 we have the following inequality chain.

Theorem 16.36 *For every graph G , $\gamma(G) \leq i(G) \leq \alpha(G) \leq \Gamma(G)$.*

The independent domination number of a graph is therefore sandwiched between the domination and independence numbers. Indeed, this is part of the canonical domination inequality chain first observed by [Cockayne et al. \[26\]](#) in 1978. That the bounds $\gamma(G) \leq i(G) \leq \alpha(G)$ are sharp may be seen by taking G to be the corona $\text{cor}(K_{1,t})$ of a star $K_{1,t}$ for which $\gamma(G) = i(G) = \alpha(G) = t + 1$. The difference between each of these parameters can, however, be made arbitrarily large. For example, for $2 \leq s \leq t$, let G be the graph obtained from the disjoint union of two stars $K_{1,s}$ and $K_{1,t}$ by adding an edge joining the central vertices of each star (such a tree is called a **double star**, denoted $S(s,t)$). Then $\gamma(G) = 2$, $i(G) = s + 1$ and $\alpha(G) = s + t$. The following stronger result, which may be found in [45], holds.

Theorem 16.37 *Given any integers $2 \leq a \leq b \leq c$, there exists a graph G with $\gamma(G) = a$, $i(G) = b$ and $\alpha(G) = c$.*

Proof Let G be obtained from the disjoint union of $a - 1$ stars $K_{1,b}$ and a star $K_{1,c}$ as follows. For $i \in [a - 1]$, let G_i be a star $K_{1,b}$ with centre v_i and leaves $v_{i1}, v_{i2}, \dots, v_{ib}$, and let G_a be a star $K_{1,c}$ with centre v_a and leaves $v_{a1}, v_{a2}, \dots, v_{ac}$. Form a clique K_a on the set of central vertices v_1, \dots, v_a of the stars. For $i \in [b]$, form a clique K_a on the set of leaves $v_{1i}, v_{2i}, \dots, v_{ai}$, one leaf from each star. Finally, if $c > b$, add all possible edges from the set $\{v_{1b}, v_{2b}, \dots, v_{a-1,b}\}$ to the set $\{v_{a,b-1}, \dots, v_{ac}\}$. Then, $\gamma(G) = a$, $i(G) = b$ and $\alpha(G) = c$. For example, the set of central vertices of the stars is a γ -set of G , while the sets $V(G_1) \setminus \{v_1\}$ and $V(G_a) \setminus \{v_a\}$ are an i -set and an α -set of G , respectively. ■

For any two graph parameters λ and μ , we define a graph G to be a $\langle \lambda, \mu \rangle$ -graph if $\lambda(G) = \mu(G)$.

16.3.2 $\langle \gamma, i \rangle$ -graphs

It remains an open problem to characterise the $\langle \gamma, i \rangle$ -graphs. A necessary and sufficient forbidden subgraph list characterising $\langle \gamma, i \rangle$ -graphs is impossible to obtain since the addition of a new vertex adjacent to all vertices of a graph G produces a graph G' containing G as an induced subgraph with $\gamma(G') = i(G') = 1$.

The first result involving forbidden subgraphs that implies equality of the parameters γ and i was that presented by Allan and Laskar [2] who proved the following result.

Theorem 16.38 *Every claw-free graph is a $\langle \gamma, i \rangle$ -graph.*

Proof Let G be a claw-free graph. Among all γ -sets of G , let D be chosen so that $G[D]$ has minimum size. Suppose, to the contrary, that D is not independent. Then there exist vertices u and v in D that are adjacent. Thus, $\text{pn}[v, D] \subseteq V \setminus D$. By the minimality of D , the set $\text{pn}[v, D]$ is nonempty. By the claw-freeness of G , the set $\text{pn}[v, D]$ forms a clique. Therefore, for any $v' \in \text{pn}[v, D]$, the set $D' = (D \setminus \{v\}) \cup \{v'\}$ is a γ -set of G such that $G[D']$ has a smaller size than $G[D]$, a contradiction. ■

The **line graph** $L(G)$ of a graph G is that graph whose vertices can be put in one-to-one correspondence with the edges of G in such a way that two vertices of $L(G)$ are adjacent if and only if the corresponding edges of G are adjacent. Since the line graph $L(G)$ of a graph G has no induced subgraph isomorphic to $K_{1,3}$, we have the following immediate consequence of [Theorem 16.38](#).

Corollary 16.39 *The line graph of any graph is a $\langle \gamma, i \rangle$ -graph.*

This improves an earlier result due to Mitchell and Hedetniemi [88] that the line graph of a tree is a $\langle \gamma, i \rangle$ -graph. Later, Topp and Volkmann [116] found sixteen graphs F such that being F -free implies that F is a $\langle \gamma, i \rangle$ -graph.

16.3.3 Domination-perfect graphs

Motivated by the concept of perfect graphs in the chromatic sense, Sumner and Moore [111] defined a graph G to be **domination perfect** if $\gamma(H) = i(H)$ for every induced subgraph H of G . As a consequence of [Theorem 16.38](#), we have the following result.

Corollary 16.40 *Claw-free graphs are domination perfect.*

By Corollary 16.40, line graphs are also domination perfect. In [111], Sumner and Moore established that it is not necessary to test every induced subgraph of a graph in order to determine whether it is domination perfect.

Theorem 16.41 *A graph is domination perfect if and only if $\gamma(H) = i(H)$ for every induced subgraph H of G with $\gamma(H) = 2$.*

Building on several other results, including those by Fulman [39], Zverovich and Zverovich [130], and Topp and Volkmann [116], Zverovich and Zverovich [131] finally provided the following forbidden induced subgraph characterisation of domination perfect graphs in terms of seventeen forbidden induced subgraphs.

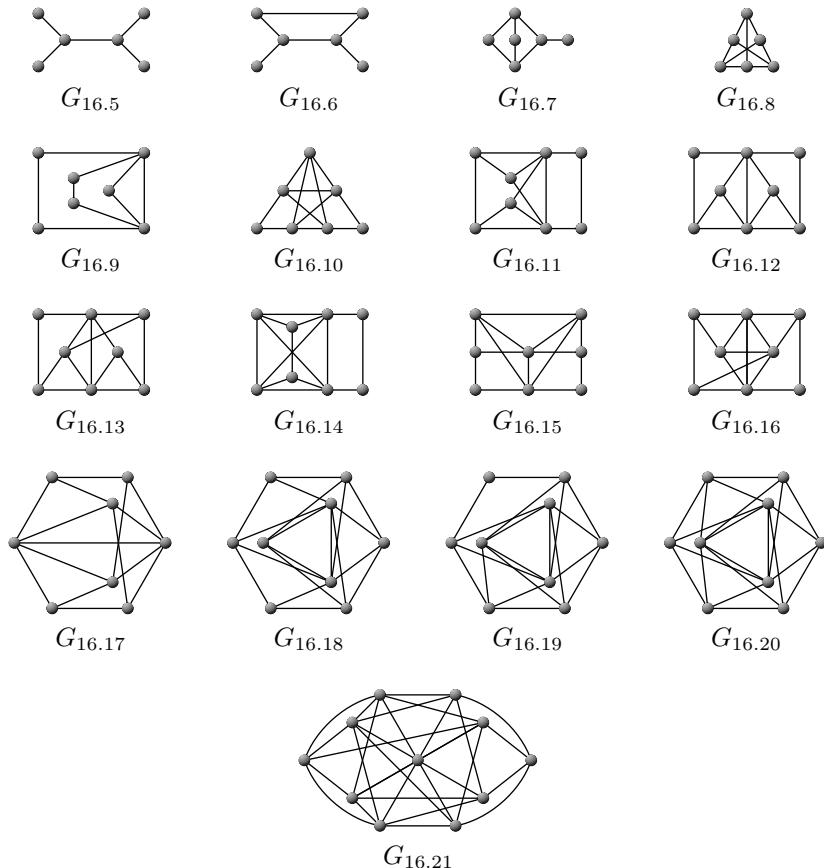


Figure 16.11: Seventeen minimal domination imperfect graphs $G_{16.5}-G_{16.21}$.

Theorem 16.42 *A graph is domination perfect if and only if it does not contain one of the seventeen graphs $G_{16.5}-G_{16.21}$ shown in Figure 16.11 as an induced subgraph.*

Lutz Volkmann (born in 1944) is an emeritus professor of mathematics at RWTH Aachen University, Germany, who has worked in various areas of mathematics. Volkmann is a prolific author on functions of a complex variable and combinatorics — more specifically in graph theory. His publications include various books, such as *Meromorphe Funktionen und Differentialgleichungen* in 1985 (co-authored by Gerhard Jank) and *Graphen an allen Ecken und Kanten* in 2011. He is the namesake of *Volkmann trees*. For many molecular structure descriptors, the Volkmann tree $V_{n,d}$ is extremal among n -vertex trees in which no vertex has degree greater than d . Volkmann supervised fifteen doctoral theses, and six of his students pursued academic careers. He is also a sportsman, and in 2014 ran a half marathon in 2 hours and 6 minutes. In 2015, he celebrated his golden wedding anniversary together with his wife Hannelore.



Biographic note 81: Lutz Volkmann (1944–present)

A graph G is called **minimal domination imperfect** if G is not domination perfect and $\gamma(H) = i(H)$, for every proper induced subgraph H of G . We remark that the seventeen subgraphs presented in the characterisation of domination perfect given in [Theorem 16.42](#) are all minimal domination imperfect graphs. [Theorem 16.42](#) therefore states that there are exactly seventeen minimal domination imperfect graphs.

16.3.4 $\langle i, \alpha \rangle$ -graphs

At the other end of the inequality chain in [Theorem 16.36](#), a graph G is defined to be **well-covered** if the independent domination number of G is equal to its independence number. Thus, a graph is well-covered if and only if it is an $\langle i, \alpha \rangle$ -graph. We note that all maximal independent sets in a well-covered graph have the same size. Equivalently, every independent set is contained in a maximum independent set of the graph. For example, the balanced complete bipartite graphs are well-covered. The concept of well-covered graphs was introduced by [Plummer \[94\]](#).

[Ravindra \[99\]](#) characterised well-covered bipartite graphs. For this purpose, if $e = uv$ is an edge in a bipartite graph, let G_e denote the subgraph induced by $N[u] \cup N[v]$. Recall, from [Chapter 9](#), that a **matching** in a graph G is a set of independent edges in G and that if every vertex of G is incident with an edge of M , then M is called a **perfect matching** in G .

Theorem 16.43 *A connected bipartite graph G is well-covered if and only if it contains a perfect matching M such that, for every edge $e \in M$, G_e is a complete bipartite graph.*

As an immediate consequence of [Theorem 16.43](#), we have a characterisation of well-covered trees.

Corollary 16.44 *A tree is well-covered if and only if it is K_1 or the corona of a tree.*

This result was extended by Finbow and Hartnell [36] who showed that a graph of girth at least 8 is well-covered if and only if the graph is the corona of a graph of girth at least 8. Later, Finbow *et al.* [37] characterised well-covered graphs of girth at least 5. A number of other classes of well-covered graphs have also been completely described, including well-covered block graphs and unicyclic graphs (by Topp and Volkmann [115]), well-covered cubic graphs (by Campbell *et al.* [14]), well-covered graphs that contain neither 4- nor 5-cycles (by Finbow *et al.* [38]), and 4-connected claw-free well-covered graphs (by Hartnell and Plummer [51]), among other classes. For a survey on well-covered graphs we refer the reader to Plummer [95].

16.3.5 Bounds for $K_{1,k}$ -free graphs

The claw-free result due to Allan and Laskar [2] in [Theorem 16.38](#) was extended as follows by Bollobás and Cockayne [7].

Theorem 16.45 *If G is a $K_{1,k+1}$ -free graph for any integer $k \geq 2$, then $i(G) \leq (k-1)\gamma(G) - (k-2)$.*

Proof Let $G = (V, E)$ be a graph, let D be a γ -set of G and let I be a maximal independent set of vertices in $G[D]$. Furthermore, let Y denote the set of all vertices in $V \setminus D$ that are adjacent to no vertex of I in G . Finally, let X be a maximal independent set of vertices in $G[Y]$. Then, $I \cup X$ is a maximal independent set in G . By [Proposition 16.34](#), $I \cup X$ is therefore an independent dominating set of G . We show that each vertex of $D \setminus I$ is adjacent to at most $k-1$ vertices of X . If this is not the case, then there is a vertex $v \in D \setminus I$ that is adjacent to (at least) k vertices of X . By the maximality of the set I , the vertex v is adjacent to some vertex of I . Hence, there exists an independent set of $k+1$ vertices adjacent to v , and so G contains an induced $K_{1,k+1}$, a contradiction. Therefore, each vertex of $D \setminus I$ is adjacent to at most $k-1$ vertices of X . Since every vertex of Y (and hence of X) is adjacent to some vertex of $D \setminus I$ in G , we have that $|X| \leq (k-1)(|D| - |I|)$. Hence, $i(G) \leq |I| + |X| \leq |I| + (k-1)(|D| - |I|) \leq (k-1)|D| - (k-2)|I| \leq (k-1)\gamma(G) - (k-2)$. ■

Setting $k = 2$ in [Theorem 16.45](#) we have the result of [Theorem 16.38](#). Zverovich and Zverovich [130] proved that the inequality in [Theorem 16.45](#) actually holds for a wider class of graphs.

Theorem 16.46 *For $k \geq 2$, if G does not contain two induced subgraphs $K_{1,k+1}$ having different centres and exactly one edge in common, then $i(G) \leq (k-1)\gamma(G) - (k-2)$.*

16.3.6 Bounds in terms of maximum degree

The upper and lower bounds on the domination number in terms of the maximum degree established in [Theorem 16.13](#) also hold for the independent domination number. The proof is almost identical.

Theorem 16.47 *For any graph G of order n and with maximum degree Δ ,*

$$\left\lceil \frac{n}{1+\Delta} \right\rceil \leq i(G) \leq n - \Delta.$$

Proof Let $G = (V, E)$ be a graph of order n with maximum degree Δ . The lower bound follows immediately from the lower bound in [Theorem 16.13](#) and the inequality $\gamma(G) \leq i(G)$. To prove the upper bound, let x be a vertex of G of maximum degree Δ and let X be a maximal independent set in $G[V \setminus N(v)]$. Necessarily, $x \in X$. The set X is an independent dominating set of cardinality $|X| \leq |V \setminus N(v)| = n - \Delta$. ■

16.3.7 Bounds relating i and γ

Using [Theorem 16.3](#), Bollobás and Cockayne [7] proved the following upper bound on the independent domination number.

Theorem 16.48 *If G is an isolate-free graph of order n with ordinary domination number γ , then $i(G) \leq n + 2 - \gamma - \lceil \frac{n}{\gamma} \rceil$.*

Proof By [Theorem 16.3](#), there exists a γ -set D of G such that $\text{epn}[v, D] \neq \emptyset$ for every $v \in D$. For each vertex $v \in D$, let $v' \in \text{epn}[v, D]$. By the Pigeonhole Principle, there is a vertex $y \in D$ that is adjacent to at least $(n - |D|)/|D|$ vertices of $V \setminus D$. Let D' be a maximal independent set containing y . Then D' contains no vertex from the set $N(y)$ and $|N(y)| \geq \lceil (n - |D|)/|D| \rceil$. Furthermore, D' contains at most one of x and x' for every vertex $x \in D \setminus \{y\}$. It follows that $i(G) \leq |D'| \leq n - (\gamma - 1) - \lceil (n - \gamma)/\gamma \rceil$, or, equivalently, $i(G) \leq n + 2 - \gamma - \lceil \frac{n}{\gamma} \rceil$. ■

16.3.8 Bounds relating i and n

Let $i(n, \delta)$ denote the maximum of the independent domination numbers over all graphs of order n and with minimum degree δ . Applying basic results from calculus to the upper bound in [Theorem 16.48](#) one obtains the following bound, first noted by Gimbel and Vestergaard [42].

Theorem 16.49 *$i(n, 1) \leq n + 2 - 2\sqrt{n}$ for all $n \in \mathbb{N}$.*

Proof By [Theorem 16.48](#), $i(G) \leq n + 2 - \gamma - \lceil \frac{n}{\gamma} \rceil \leq n + 2 - \gamma - \frac{n}{\gamma}$. Since the function $\gamma + \frac{n}{\gamma}$ attains a minimum at $\gamma = \sqrt{n}$, the expression $n + 2 - \gamma - \frac{n}{\gamma}$ is maximised at $n + 2 - 2\sqrt{n}$. ■

For the extremal graphs, add $m - 1$ pendent edges to each vertex of the clique K_m . Let G denote the resulting graph. Then G has order $n = m^2$ and $i(G) = (m - 1)^2 + 1 = n + 2 - 2\sqrt{n}$. We note that the extremal graphs all have minimum degree $\delta = 1$. For $\delta \geq 2$, we have the following bounds on the independent domination number.

Theorem 16.50 *$i(n, \delta) \leq \frac{2}{3}(n - \delta)$ if $\frac{1}{7}(n - 2) \leq \delta \leq \frac{n}{4}$.*

Theorem 16.51 *$i(n, \delta) \leq n + 3\delta - 2\sqrt{\delta(n + 2\delta - 4)} - 2$ for any $\delta \geq 2$.*

The bound in [Theorem 16.50](#) is due to Haviland [52], while the bound in [Theorem 16.51](#) is due to Glebov and Kostochka [43].

Favaron [33] made the following conjecture.

Conjecture 16.52 *$i(n, \delta) \leq n + 2\delta - 2\sqrt{\delta n}$ for any $\delta \geq 1$.*

Odile Favaron was born Zink in Paris in 1938. She completed her state thesis (doctorate and habilitation) in 1986 under the supervision of Jean-Claude Bermond. She started her academic career at the Universities of Besançon and of Paris-Sud, before accepting an associate professor position in the mathematics department of the University Paris-Sud in Orsay, where she worked until her retirement in 2003. She worked in graph theory in the Laboratoire de Recherche en Informatique (LRI). She is best known for her significant contributions to graph domination theory, and is an expert on claw-free graphs. She is a prolific researcher, having co-authored many journal papers.



Biographic note 82: Odile Favaron (1938–present)

Favaron [33] showed that if [Conjecture 16.52](#) is true, then the bound is attained for every fixed positive integer δ by infinitely many graphs. When $\delta = 2$, the Glebov-Kostochka bound in [Theorem 16.51](#) is equivalent to the bound in Favaron's [Conjecture 16.52](#). In 1999, Sun and Wang [113] presented a proof of [Conjecture 16.52](#) for all $\delta \geq 1$.

16.3.9 Bounds for bipartite graphs

Since every connected bipartite graph, being 2-colourable, is the union of two independent sets, each of which dominates the other, we have the following bound on the independent domination number of a bipartite graph.

Theorem 16.53 *If G is a connected bipartite graph of order n , then $i(G) \leq \frac{n}{2}$.*

That the bound in [Theorem 16.53](#) is sharp may be seen by taking a balanced complete bipartite graph $K_{n/2, n/2}$ for $n \geq 2$ even. Even for arbitrarily large diameter, the bound of $\frac{n}{2}$ in [Theorem 16.53](#) is sharp. Consider, for example, an even cycle C_{2k} and add $r \geq 1$ pendant edges to each vertex of the cycle. Let G denote the resulting graph. Then G has order $n = 2(r+1)k$ and $i(G) = (r+1)k$.

The bound of $\frac{n}{2}$ in [Theorem 16.53](#) is essentially best possible even for connected bipartite graphs of arbitrarily large order n and with fixed minimum degree δ . To see this, let $\delta \geq 1$ and $k \geq 2$ be fixed integers, and consider k disjoint copies of $K_{\delta, \delta}$. Add a new vertex v and join v to every vertex in one of the partite sets from each of the k copies of the $K_{\delta, \delta}$. Call the resulting graph G , and note that G has order $n = 2k\delta + 1$ and that v has degree $k\delta$ in G . Let S be a maximal independent set in G . If $v \in S$, then $|S| = (n+1)/2$; otherwise, $|S| = (n-1)/2$. Thus, $i(G) = (n-1)/2$.

Rautenbach and Volkmann [98] posed as an open problem a characterisation of connected bipartite graphs achieving equality in the bound of [Theorem 16.53](#). Such a characterisation was given by Ma and Chen [84]. For this purpose, they introduced the following notation. For a bipartite graph G with partite sets X and Y , and for an arbitrary subset $S \subseteq X$, let $C(S) = S \cup \{v \in X \setminus S \mid N(v) \subseteq N(S)\}$.

Theorem 16.54 Let G be a connected bipartite graph of order n with partite sets X and Y . Then $i(G) = \frac{n}{2}$ if and only if $|C(S)| \geq |N(S)|$ for every subset $S \subseteq X$.

Proof First we prove the necessity. Suppose $i(G) = \frac{n}{2}$. Since $i(G) \leq \min\{|X|, |Y|\} \leq \frac{n}{2}$, we have that $|X| = |Y| = \frac{n}{2}$. If there exists a subset $S \subseteq X$ such that $|C(S)| < |N(S)|$, then $C(S) \cup (Y \setminus N(S))$ is an independent dominating set of G . Hence, $i(G) < |C(S)| + |Y \setminus N(S)| = |Y| + |C(S)| - |N(S)| < |Y| = \frac{n}{2}$, a contradiction. Therefore, $|C(S)| \geq |N(S)|$ for every subset $S \subseteq X$.

Next we prove the sufficiency. Suppose $|C(S)| \geq |N(S)|$ for every subset $S \subseteq X$. Suppose, to the contrary however, that $i(G) < \frac{n}{2}$. Let I be an i -set of G , let $I_X = I \cap X$ and let $I_Y = I \cap Y$. Then $|I| = |I_X| + |I_Y| < \frac{n}{2}$. Since I is an independent dominating set of G , each vertex in $X \setminus I_X$ is dominated by I_Y and each vertex in $Y \setminus I_Y$ is dominated by I_X . Hence, $N(I_X) = Y \setminus I_Y$ and $N(I_Y) = X \setminus I_X$. Furthermore, $N(x) \not\subseteq N(I_X)$ for every vertex $x \in X \setminus I_X$. Hence, $C(I_X) = I_X$. Similarly, $C(I_Y) = I_Y$. By assumption, $|C(I_X)| \geq |N(I_X)|$ and $|C(I_Y)| \geq |N(I_Y)|$. Thus, $|I_X| + |I_Y| = |C(I_X)| + |C(I_Y)| \geq |N(I_X)| + |N(I_Y)| = n - |I_X| - |I_Y|$, and so $|I_X| + |I_Y| \geq \frac{n}{2}$. This contradicts our earlier observation that $|I_X| + |I_Y| < \frac{n}{2}$. Therefore, $i(G) \geq \frac{n}{2}$. ■

16.3.10 Bounds for trees

As a special case of [Theorem 16.53](#), it holds that $i(T) \leq \frac{n}{2}$ for every tree T and this bound is sharp. Favaron [34] provided the following bound for trees which was originally conjectured by McFall and Nowakowski [86].

Theorem 16.55 If T is a tree of order $n \geq 2$ with ℓ leaves, then $i(G) \leq (n + \ell)/3$.

The bound in [Theorem 16.55](#) is achieved, for example, by the path P_{3k+1} where $k \geq 1$. Other trees achieving equality include the corona $T = \text{cor}(T')$ of a tree T' on n' vertices. Such a corona T has order $n = 2n'$ with $\ell = n'$ leaves satisfying $i(T) = n' = (n + \ell)/3$. Favaron [34] also characterised the extremal trees in [Theorem 16.55](#).

16.3.11 Bounds for cubic graphs

Lam *et al.* [79] established an upper bound on the independent domination number of a cubic graph.

Theorem 16.56 For any connected cubic graph G of order n other than $K_{3,3}$, $i(G) \leq \frac{2}{5}n$.

Equality in [Theorem 16.56](#) holds for the prism $C_5 \square K_2$. Goddard and Henning [45] conjectured that the graphs $K_{3,3}$ and $C_5 \square K_2$ are the only exceptions for an upper bound of $\frac{3}{8}n$ on the independent domination number.

Conjecture 16.57 If G is a connected cubic graph of order $n > 10$, then $i(G) \leq \frac{3}{8}n$.

Two infinite families $\mathcal{G}_{\text{cubic}}$ and $\mathcal{H}_{\text{cubic}}$ of connected cubic graphs with independent domination number three-eighths their orders were constructed in [45] as follows. For $k \geq 1$, consider two copies of the path P_{4k} with respective vertex sequences $a_1 b_1 c_1 d_1 \dots a_k b_k c_k d_k$ and $w_1 x_1 y_1 z_1 \dots w_k x_k y_k z_k$. For each $i \in [k]$, join a_i to w_i , b_i to x_i , c_i to y_i , and d_i to z_i . To complete the construction of graphs in $\mathcal{G}_{\text{cubic}}$, join a_1 to d_k and w_1 to z_k .

For $\ell \geq 1$, consider a copy of the cycle $C_{3\ell}$ with vertex sequence $a'_1 b'_1 c'_1 \dots a'_{\ell} b'_{\ell} c'_{\ell} a'_1$. Let $\mathcal{L} = [\ell]$. For each $i \in \mathcal{L}$, add the vertices $\{w'_i, x'_i, y'_i, z'_i, z''_i\}$ and join a_i to w'_i , b_i to x'_i , and c_i to y'_i . To complete the construction of graphs in $\mathcal{H}_{\text{cubic}}$, for each $i \in \mathcal{L}$ and $j \in \{1, 2\}$, join z'_i to each of the vertices w'_i, x'_i , and y'_i .

Graphs in the families $\mathcal{G}_{\text{cubic}}$ and $\mathcal{H}_{\text{cubic}}$ are illustrated in Figure 16.12.

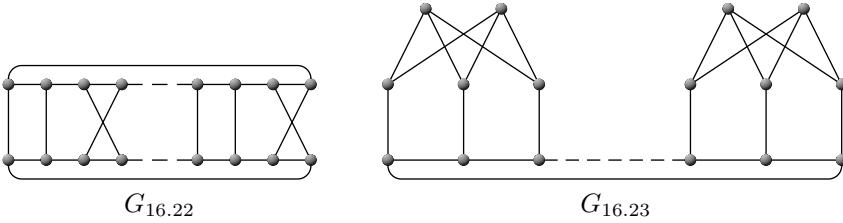


Figure 16.12: Two graphs $G_{16.22} \in \mathcal{G}_{\text{cubic}}$ and $G_{16.23} \in \mathcal{H}_{\text{cubic}}$.

It was shown in [45] that every graph in the family $\mathcal{G}_{\text{cubic}} \cup \mathcal{H}_{\text{cubic}}$ is a connected cubic graph with independent domination number three-eighths its order.

Proposition 16.58 *If $G \in \mathcal{G}_{\text{cubic}} \cup \mathcal{H}_{\text{cubic}}$ has order n , then $i(G) = \frac{3}{8}n$.*

Hence if Conjecture 16.57 is true, then the bound is sharp. We remark that the family $\mathcal{G}_{\text{cubic}}$ provides a simple example of a family of 3-connected cubic graphs with the difference between the domination and independent domination numbers arbitrarily large.

Proposition 16.59 *If $G \in \mathcal{G}_{\text{cubic}}$ has order n , then either $n = 16k$ or $n = 16k + 8$ for some integer $k \geq 1$. In both cases, $i(G) - \gamma(G) \geq k$.*

We close this section on cubic graphs with two conjectures. The first conjecture was posed by Verstra  te [120].

Conjecture 16.60 *If G is a connected cubic graph of order n and with girth at least 6, then $i(G) \leq \frac{n}{3}$.*

We remark that the girth six requirement in Verstra  te's Conjecture is essential, since the generalised Petersen graph $G_{16.24}$ of order $n = 14$ shown in Figure 16.13 has independent domination number $i(G_{16.24}) = 5 > \frac{n}{3}$. Perhaps the graph $G_{16.24}$ is the only exception when relaxing the girth condition from six to five.

The following conjecture is due to Goddard and Henning [45].

Conjecture 16.61 *If $G \neq K_{3,3}$ is a connected bipartite cubic graph of order n , then $i(G) \leq \frac{4}{11}n$.*

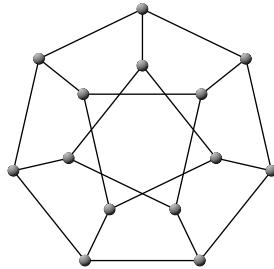


Figure 16.13: The generalised Petersen graph $G_{16,24}$ of order 14.

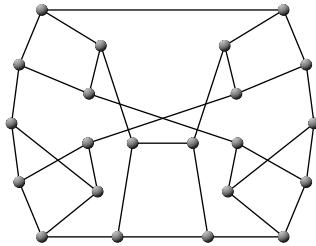


Figure 16.14: The bipartite cubic graph $G_{16,25}$ with $i(G_{16,25}) = \frac{4}{11}n$.

We remark that [Southey \[106\]](#) confirmed the truth of [Conjecture 16.61](#) for $n \leq 26$. If [Conjecture 16.61](#) is true, then the bound is achieved by the bipartite cubic graph $G_{16,25}$ of order $n = 22$ with $i(G_{16,25}) = 8$ shown in [Figure 16.14](#).

[Henning et al. \[64\]](#) have shown that [Conjecture 16.61](#) is true if the girth is at least six; that is, [Conjecture 16.61](#) holds if the graph is quadrilateral-free.

16.3.12 Bounds for regular graphs

As an immediate consequence of the domination inequality chain ([Theorem 16.36](#)) and the upper bound on the upper domination number for regular graphs ([Theorem 16.32](#)), we have the following upper bound on the independent domination number.

Theorem 16.62 *For every regular graph G of order n with no isolated vertex, $i(G) \leq \frac{n}{2}$.*

As observed in [\[46\]](#), if equality holds in [Theorem 16.62](#), then every component of G is a copy of $K_{k,k}$. We state this formally as follows.

Theorem 16.63 ([\[46\]](#)) *For all integers $k \geq 1$, if G is a connected k -regular graph of order n , then $i(G) \leq n/2$, with equality if and only if $G = K_{k,k}$.*

For most values of δ the upper bound in [Theorem 16.62](#) on $i(G)$ for a δ -regular graph seems far from best possible. [Favaron \[33\]](#) was the first to improve the bound for large δ as follows.

Theorem 16.64 *If G is a δ -regular graph of order n with $\delta \geq \frac{n}{2}$, then $i(G) \leq n - \delta$, with equality attained only by complete, balanced multipartite graphs.*

[Haviland \[52, 53\]](#) improved the upper bound of [Theorem 16.64](#) as follows for values of δ satisfying $\frac{n}{4} \leq \delta \leq \frac{n}{2}$. Note that $(3 - \sqrt{5})/2 \approx 0.381966$.

Theorem 16.65 If G is a δ -regular graph of order n with $\delta \geq \frac{n}{2}$, then

$$i(G) \leq \begin{cases} n - \sqrt{n\delta} & \text{if } \frac{n}{4} \leq \delta \leq (3 - \sqrt{5})\frac{n}{2} \\ \delta & \text{if } (3 - \sqrt{5})\frac{n}{2} \leq \delta \leq \frac{n}{2}. \end{cases}$$

We establish next a best possible upper bound on the ratio of the independence number and the domination number of a regular graph. Let G be a connected k -regular graph of order n . If $k = 1$, then $G = K_2$ and $i(G) = \gamma(G) = 1$. If $k = 2$, then G is a cycle C_n of order n and $i(G) = \gamma(G) = \lceil \frac{n}{3} \rceil$. Thus, for $k \in \{1, 2\}$, we have $i(G)/\gamma(G) = 1$. Hence, the ratio $i(G)/\gamma(G)$ is only of interest when $k \geq 3$.

In 2012, Goddard *et al.* [46] showed that if G is a connected k -regular graph and $k = 3$, then the ratio $i(G)/\gamma(G) \leq \frac{k}{2}$, with equality if and only if $G = K_{k,k}$. In 2020, Babikir and Henning [4] extended this result to larger values of k , namely $k \in \{4, 5, 6\}$. A natural question posed in [4] is whether this result is true for all integers $k \geq 3$. This question was answered in the affirmative in 2021 by Knor *et al.* [75].

Theorem 16.66 ([75]) If G is a connected k -regular graph for any integer $k \geq 3$, then

$$\frac{i(G)}{\gamma(G)} \leq \frac{k}{2},$$

with equality if and only if $G = K_{k,k}$.

Proof For $k \geq 3$, let G be a connected k -regular graph. Let X be a γ -set of G and let $Y = V(G) \setminus X$. Let t be the number of edges in the subgraph $G[X]$ of G induced by X , and so $t = m(G[X])$. If $t = 0$, then X is an ID-set of G , implying that $i(G) = \gamma(G)$, and so the desired bound on the ratio is immediate noting that in this case $i(G)/\gamma(G) = 1 < \frac{3}{2} \leq \frac{k}{2}$. Hence, we may assume that $t \geq 1$.

Suppose first that $t < \gamma(G)/2$. Let X_1 be a maximal independent set in $G[X]$, and let $X_2 = X \setminus X_1$. If a vertex in X_2 has no neighbour in X_1 , then it could be added to X_1 to produce a larger independent set in $G[X]$, contradicting the maximality of the independent set X_1 . Every vertex in X_2 therefore has at least one neighbour in X_1 , implying that $t \geq |X_2|$. Thus, $|X| = |X_1| + |X_2| \leq |X_1| + t$, and so $|X_1| \geq |X| - t$. We can therefore write $|X_1| = |X| - t + s$ for some integer $s \geq 0$. Let $X_2 = \{x_1, \dots, x_{t-s}\}$, and let N_i be the set of neighbours of x_i that belong to Y , that is, $N_i = N_G(x_i) \cap Y$ for $i \in [t-s]$. Also, let

$$Y_1 = \bigcup_{i=1}^t N_i.$$

As observed earlier, every vertex in X_2 has at least one neighbour in X_1 and therefore at most $k-1$ neighbours in Y . Thus, $|N_i| \leq k-1$ for $i \in [t-s]$, implying that $|Y_1| \leq (k-1)(t-s)$. Let $Y_{1,1}$ be the set of vertices in Y_1 that have a neighbour in X_1 , and so the set X_1 dominates the set $Y_{1,1}$. Let $Y_{1,2} = Y_1 \setminus Y_{1,1}$, and let I_1 be an ID-set in $G[Y_{1,2}]$. Since $I_1 \subseteq Y_{1,2} \subseteq Y_1$, we have $|I_1| \leq |Y_1| \leq (k-1)(t-s)$. Since X is a dominating set of G , the set X_1 dominates the set $Y \setminus Y_1$. As observed earlier, the set X_1 dominates both X_2 and $Y_{1,1}$. By definition, the set I_1 dominates the set $Y_{1,2}$. The set $X_1 \cup I_1$ is therefore a dominating set of G . Further, $X_1 \cup I_1$

is an independent set. Therefore, $X_1 \cup I_1$ is an ID-set of G , implying that

$$\begin{aligned} i(G) &\leq |X_1| + |I_1| \leq (|X| - t + s) + (k - 1)(t - s) \\ &= |X| + (k - 2)(t - s) \leq |X| + (k - 2)t. \end{aligned}$$

By supposition, $t < \gamma(G)/2$. Thus, since $|X| = \gamma(G)$, the above inequality chain implies that

$$i(G) < \gamma(G) + (k - 2)\gamma(G)/2 = \frac{k}{2}\gamma(G),$$

or, equivalently, $i(G)/\gamma(G) < \frac{k}{2}$. Hence, we may assume that $t \geq \gamma(G)/2$, for otherwise the desired result follows (with strict inequality). Under this assumption, we show that $\gamma(G) \geq \frac{n}{k}$. Let $[X, Y]$ be the set of edges between X and Y . Counting edges incident with vertices in X , the k -regularity of the graph G implies that

$$k|X| = \sum_{x \in X} d_G(x) = |[X, Y]| + 2t, \quad (16.9)$$

noting that the edges in $G[X]$ are double counted when the degrees of the vertices in X are summed. Since X is a dominating set of G , every vertex of Y has at least one neighbour in X , and so $|[X, Y]| \geq |Y|$. Hence, since $|X| = \gamma(G)$ and $t \geq \gamma(G)/2$, we have by (16.9) that

$$|Y| \leq |[X, Y]| = k|X| - 2t \leq k\gamma(G) - \gamma(G) = (k - 1)\gamma(G).$$

Therefore, $n = |X| + |Y| \leq \gamma(G) + (k - 1)\gamma(G) = k\gamma(G)$, or, equivalently, $\gamma(G) \geq \frac{n}{k}$. By Theorem 16.63, $i(G) \leq \frac{n}{2}$, with equality if and only if $G = K_{k,k}$, and so

$$\frac{i(G)}{\gamma(G)} \leq \frac{n/2}{n/k} = \frac{k}{2}. \quad (16.10)$$

This establishes the desired upper bound in the statement of the theorem. Suppose, next, that G is a connected k -regular graph satisfying $i(G)/\gamma(G) = \frac{k}{2}$. In this case, we must have equality in (16.10). In particular, $i(G) = \frac{n}{2}$, implying by Theorem 16.63 that $G = K_{k,k}$. ■

In the special case when $k = 3$, we have by Theorem 16.66 that if G is a connected cubic graph, then $i(G)/\gamma(G) \leq \frac{3}{2}$, with equality if and only if $G = K_{3,3}$. In 2013, Southey and Henning [107] showed that if we exclude $K_{3,3}$, then the upper bound in Theorem 16.66 can be improved considerably. In order to state their result, recall that the 5-prism $C_5 \square K_2$, shown in Figure 16.15, is the Cartesian product of a 5-cycle with a copy of K_2 .

Theorem 16.67 *If $G \neq K_{3,3}$ is a connected cubic graph, then $i(G)/\gamma(G) \leq \frac{4}{3}$, with equality if and only if $G = C_5 \square K_2$.*

The following question was posed in [45].

Question 16.68 *Is it true that if $G \neq K_{4,4}$ is a connected 4-regular graph, then $i(G)/\gamma(G) \leq \frac{3}{2}$?*

If Question 16.68 can be answered affirmatively, then the bound is achieved, for example, by the 4-regular graphs $G_{16.26}$ and $G_{16.27}$ shown in Figure 16.16. Both graphs have domination number 4 and independent domination number 6.

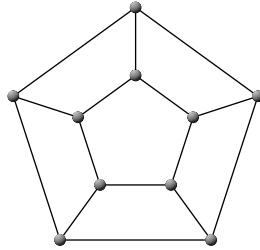


Figure 16.15: The 5-prism $C_5 \square K_2$.

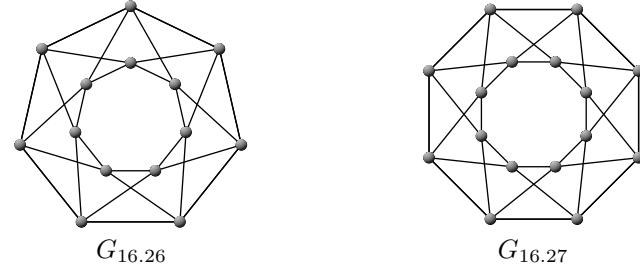


Figure 16.16: The graphs $G_{16.26}$ and $G_{16.27}$.

16.3.13 Bounds for triangle-free graphs

The independent domination numbers of triangle-free graphs were investigated by Haviland [54] and later by Goddard and Lyle [48] who proved the following bounds (where part (iii) was also established in [104]).

Theorem 16.69

- (i) There exists a triangle-free graph G of order n with $i(G) = n - \mathcal{O}(n)$.
- (ii) Furthermore, let G be a triangle-free graph of order n . If $\delta(G) \geq \frac{3}{20}n$, then $i(G) \leq \frac{n}{2}$, and this bound is sharp for $\delta(G) \leq \frac{n}{4}$.
- (iii) If $\delta(G) \geq \frac{n}{4}$, then $i \leq \max\{n - 2\delta(G), \delta(G)\}$, and this bound is sharp.

Equality is obtained for graphs such as the following: Consider the path P_4 , $v_1 v_2 v_3 v_4$, and replace each vertex v_i by an independent set A_i with the same neighbourhood, where $|A_1| = |A_4| = \frac{n}{2} - \delta$ and $|A_2| = |A_3| = \delta$.

Goddard and Lyle [48] constructed triangle-free graphs G with $i(G) > \frac{n}{2}$ for all $0 < \delta(G) < \frac{n}{10}$ as follows. For a positive integer δ , let G_δ be obtained from the corona $\text{cor}(C_5)$ of a 5-cycle by replacing each end-vertex by an independent set of size $\frac{n}{5} - \delta$ and replacing each vertex of the 5-cycle by an independent set of size δ . Then, $i(G_\delta) > \frac{n}{2}$ for all $\delta < \frac{n}{10}$. They posed the following question.

Question 16.70 Is it true that every triangle-free graph G of order n with minimum degree $\delta(G) \geq \frac{n}{10}$ satisfies $i(G) \leq \frac{n}{2}$?

16.3.14 Nordhaus-Gaddum type bounds

Nordhaus-Gaddum type bounds on the sum of the independent domination numbers of a graph and its complement are easy to establish.

Observation 16.71 *If G is a graph of order $n \geq 2$, then $3 \leq i(G) + i(\bar{G}) \leq n + 1$.*

Proof The lower bound follows immediately from the observation that if $i(G) = 1$ or $i(\bar{G}) = 1$, then $i(\bar{G}) \geq 2$ or $i(G) \geq 2$, respectively. That this lower bound is sharp may be seen by considering the graph $G = K_{1,n-1}$ with $i(G) = 1$ and $i(\bar{G}) = 2$. Applying the upper bound $i(G) \leq n - \Delta(G)$ established in [Theorem 16.47](#), we have that $i(G) + i(\bar{G}) \leq 2n - (\Delta(G) + \Delta(\bar{G})) \leq 2n - (\Delta(G) + \delta(\bar{G})) = 2n - (n-1) = n+1$. That this bound is sharp may be seen by considering $G = K_n$ or $\bar{G} = K_n$. ■

If one does not allow isolated vertices in either G or \bar{G} , then the upper bound in [Observation 16.71](#) can be improved as follows [44].

Theorem 16.72 *If G is a graph of order $n \geq 2$ such that neither G nor \bar{G} has an isolated vertex, then $i(G) + i(\bar{G}) \leq n + 4 - \lfloor 2\sqrt{n} \rfloor$, and this bound is sharp.*

Trivially, if G is a graph of order $n \geq 2$, then $2 \leq i(G)i(\bar{G})$. Finding sharp upper bounds on the product of the independent domination numbers of a graph and its complement, however, has proven to be more challenging. This problem has been studied by several authors, including those of [21] and [23], before the problem was finally settled as follows by [Goddard and Henning](#) [44].

Theorem 16.73 *Define $b(n) = \lfloor (n+4)/4 \rfloor \lfloor (n+6)/4 \rfloor$. Then, for all graphs G of order n ,*

$$i(G)i(\bar{G}) \leq \begin{cases} n & \text{if } n \leq 7 \\ b(n) + 1 & \text{if } n = x^2 \text{ for } x \text{ odd, or } n = x^2 - 1 \text{ for } x \text{ even} \\ b(n) & \text{otherwise,} \end{cases}$$

and this bound is best possible for all n .

16.4 The irredundance number

Let $G = (V, E)$ be a graph, let S be a set of vertices in G and let v be a vertex in G . Recall that the **S -private neighbourhood** of v is defined by $\text{pn}[v, S] = \{w \in V \mid N[w] \cap S = \{v\}\}$, while its **S -external private neighbourhood** is defined by $\text{epn}[v, S] = \text{pn}[v, S] \cap (V \setminus S)$ and its **S -internal private neighbourhood** by $\text{ipn}[v, S] = \text{pn}[v, S] \cap S$. Hence, $\text{pn}[v, S] = \text{epn}[v, S] \cup \text{ipn}[v, S]$.

The following definitions are due to [Cockayne et al.](#) [26]. If $\text{pn}[v, S] \neq \emptyset$, then the vertex v is called an **irredundant vertex**, while if $\text{pn}[v, S] = \emptyset$, then v is called a **redundant vertex**. Furthermore, the set S is an **irredundant set** if every vertex in S is an irredundant vertex. Thus, the set S is an irredundant set if every vertex in S has at least one S -private neighbour. A set of vertices that is not irredundant is called a **redundant set**. Consequently, a set S of vertices in a graph G is a redundant set if it contains a redundant vertex.

Since $\text{pn}[v, S] = \text{epn}[v, S] \cup \text{ipn}[v, S]$, a set S is an irredundant set if, for each $v \in S$, $\text{epn}[v, S] \neq \emptyset$ or $\text{ipn}[v, S] \neq \emptyset$. Hence, by [Theorem 16.1](#), we have the following observation.

Observation 16.74 *Every minimal dominating set of a graph G is an irredundant set of G .*

We further observe that every independent set of G is an irredundant set of G . A property P of sets of vertices is said to be **hereditary** if, whenever a set S has property P , so does every proper subset of S , while a property P is **superhereditary** if, whenever a set S has property P , so does every proper superset of S . For example, the property P of being an irredundant set is hereditary, since every proper subset of an irredundant set is also an irredundant set. Moreover, the property P of being a dominating set is superhereditary, since every proper superset of a dominating set is also a dominating set. We remark that the property P of being an independent set is also hereditary.

The **irredundance number** of G , denoted by $\text{ir}(G)$, is the minimum cardinality taken over all maximal irredundant sets of vertices of G , while the **upper irredundance number** of G , denoted by $\text{IR}(G)$, is the maximum cardinality of an irredundant set of G . An irredundant set of G of cardinality $\text{ir}(G)$ is called an **ir-set** of G , and similarly for the upper irredundance number IR .

Consider, for example, the set $S = \{b, c, h, i\}$ of vertices in the graph $T_{16.28}$ shown in Figure 16.17. Since $\text{epn}[b, S] = \{a\}$, $\text{epn}[c, S] = \{d\}$, $\text{epn}[h, S] = \{g\}$, and $\text{epn}[i, S] = \{j\}$, we have that $\text{epn}[v, S] \neq \emptyset$ for each $v \in S$. Hence, S is an irredundant set of $T_{16.28}$. Moreover, S is a maximal irredundant set of $T_{16.28}$. To see this, we note that adding the vertex a to the set S results in a set in which the vertex b has no private neighbour; that is, $\text{pn}[b, S \cup \{a\}] = \emptyset$. Adding either d or e to the set S produces a set in which the vertex c has no private neighbour. By symmetry, adding g or j or k to S produces a set in which h or i has no private neighbour. Finally, adding the vertex f to S results in a set in which the vertex f has no private neighbour. Hence, S is a maximal irredundant set in $T_{16.28}$, and so $\text{ir}(T_{16.28}) \leq 4$. Suppose S' is an irredundant set in $T_{16.28}$ with $|S'| \leq 3$. Then, $|S' \cap \{a, b, c, d, e\}| \leq 1$ or $|S' \cap \{g, h, i, j, k\}| \leq 1$. Renaming vertices if necessary, we may assume that $|S' \cap \{a, b, c, d, e\}| \leq 1$ and that $S' \cap \{a, b\} = \emptyset$. But then $S' \cup \{a\}$ is an irredundant set. Therefore, no irredundant set in $T_{16.28}$ consisting of three or fewer vertices is maximal, and so $\text{ir}(T_{16.28}) \geq 4$. Consequently, $\text{ir}(T_{16.28}) = 4$ and S is an ir-set of $T_{16.28}$. A maximal irredundant set need therefore not be a dominating set and, strictly speaking, the irredundance number is not a domination parameter. One can also show that $\gamma(T_{16.28}) = 5$, $\text{IR}(T_{16.28}) = 6$ and that $\{b, d, f, h, j\}$ and $\{a, c, e, g, i, k\}$ are examples of a γ -set and of an IR-set of $T_{16.28}$, respectively.

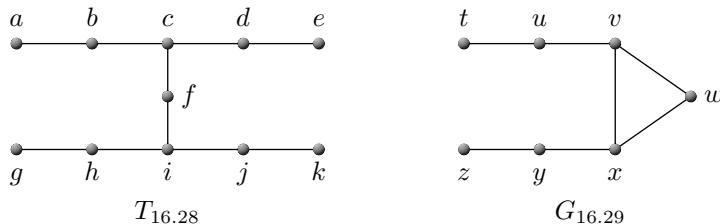


Figure 16.17: The graphs $T_{16.28}$ and $G_{16.29}$.

16.4.1 Maximal irredundant sets

Bollobás and Cockayne [7] were the first to observe the following result.

Proposition 16.75 *Every minimal dominating set of a graph G is a maximal irredundant set of G .*

Proof Let S be a minimal dominating set of a graph G . By Observation 16.74, S is an irredundant set of G . Hence, it remains to be shown that S is maximal. Suppose that this is not the case. Then there is a maximal irredundant set S' in G that properly contains the set S . Let $v \in S' \setminus S$. Since S' is an irredundant set, $\text{pn}[v, S'] \neq \emptyset$. Thus, $\text{ipn}[v, S'] \neq \emptyset$ or $\text{epn}[v, S'] \neq \emptyset$. If $\text{ipn}[v, S'] \neq \emptyset$, then v is isolated in $G[S']$ and is therefore not dominated by S in G . If $\text{epn}[v, S'] \neq \emptyset$, then let $v' \in \text{epn}[v, S']$. Therefore, the vertex $v' \in V \setminus S' \subset V \setminus S$ and v' is not dominated by S in G . In both cases, we contradict the assumption that S is a dominating set in G . ■

Volkmann and Zverovich [126] provided the following necessary and sufficient condition for an irredundant set to be maximal.

Proposition 16.76 *Let X be an irredundant set of a graph G and let $U = V(G) \setminus N[X]$. The set X is a maximal irredundant set if and only if, for every vertex $v \in N[U]$, the vertex v dominates $\text{pn}[x, X]$ for some vertex $x \in X$.*

Proof Let $G = (V, E)$ be a graph, let X be an irredundant set of G and let $U = V \setminus N[X]$. Suppose that X is a maximal irredundant set. Suppose, to the contrary, however, that there is a vertex $v \in N[U]$ that does not dominate $\text{pn}[x, X]$ for every vertex $x \in X$ and consider the set $X' = X \cup \{v\}$. Since X is an irredundant set, $\text{pn}[x, X] \neq \emptyset$ for each vertex $x \in X$. Hence, since v does not dominate $\text{pn}[x, X]$, we have that $\text{pn}[x, X'] \neq \emptyset$ for each vertex $x \in X$. If $v \in U$, then $v \in \text{pn}[v, X']$. If $v \notin U$, then v is adjacent to some vertex $u \in U$, implying that $u \in \text{pn}[v, X']$. In both cases, therefore, $\text{pn}[v, X'] \neq \emptyset$. Hence, $\text{pn}[x', X'] \neq \emptyset$ for each vertex $x' \in X'$. Consequently, the set X' is an irredundant set, contradicting the maximality of X . This establishes the necessity.

To prove the sufficiency, suppose that for every vertex $v \in N[U]$, the vertex v dominates $\text{pn}[x, X]$ for some vertex $x \in X$. Let u be an arbitrary vertex in $V \setminus X$. If $u \in N[U]$, then u dominates $\text{pn}[x, X]$ for some vertex $x \in X$. For such a vertex $x \in X$, we have $\text{pn}[x, X \cup \{u\}] = \emptyset$. If $u \notin N[U]$, then $N[u] \subseteq N[X]$. Consequently, $\text{pn}[u, X \cup \{u\}] = \emptyset$. Hence, for every vertex $u \in V \setminus X$, the set $X \cup \{u\}$ is not irredundant, and so X is a maximal irredundant set. ■

16.4.2 The domination inequality chain revisited

Since every minimal dominating set is a maximal irredundant set, we can extend the inequality chain in Theorem 16.36 as follows.

Theorem 16.77 *For every graph G , $\text{ir}(G) \leq \gamma(G) \leq i(G) \leq \alpha(G) \leq \Gamma(G) \leq \text{IR}(G)$.*

The inequality chain above was first observed by Cockayne et al. [26] in 1978. The inequality $\text{ir}(G) \leq \gamma(G)$ may be strict as is illustrated by the graph $T_{16.28}$ in Figure 16.17, where $\text{ir}(T_{16.28}) = 4$ and $\gamma(T_{16.28}) = 5$. Also, for the graph $G_{16.29}$ in Figure 16.17, we have $\text{ir}(G_{16.29}) = 2$ and $\gamma(G_{16.29}) = 3$. The set $\{v, x\}$ is a maximal irredundant set of the graph $G_{16.29}$. The difference between the irredundance number and the domination number cannot, however, be too large as is shown

by the following inequality relating the irredundance number of a graph and its domination number. This inequality was independently discovered by Allan and Laskar [2] as well as by Bollobás and Cockayne [7].

Theorem 16.78 *For every graph G , $\gamma(G) \leq 2\text{ir}(G) - 1$.*

Proof Let $S = \{v_1, \dots, v_k\}$ be an ir-set of G . Since S is an irredundant set, $\text{pn}[v, S] \neq \emptyset$ for each vertex $v \in S$. For $i = [k]$, let $u_i \in \text{pn}[v_i, S]$ (possibly, $u_i = v_i$). Let $S' = \{u_1, \dots, u_k\}$ and let $D = S \cup S'$. We show that D is a dominating set of G . If there is a vertex u not dominated by D , then for every vertex $v \in D$ we have that $u \notin N[v]$ and $v \notin N[u]$. Hence, $u \in \text{pn}[u, S \cup \{u\}]$. Furthermore, for every vertex $v_i \in S$, we note that $u_i \in \text{pn}[v_i, S \cup \{u\}]$. Hence, $\text{pn}[v, S \cup \{u\}] \neq \emptyset$ for every $v \in S \cup \{u\}$. Thus, $S \cup \{u\}$ is an irredundant set, contradicting the maximality of S and so D is a dominating set of G . If D is a minimal dominating set of G , then by Proposition 16.75, the set D would be a maximal irredundant set of G , contradicting the maximality of S . Hence, D contains a proper subset D' that is a dominating set of G , and so $\gamma(G) \leq |D'| \leq |D| - 1 \leq 2k = 2\text{ir}(G)$. ■

The domination chain stated in Theorem 16.77 suggests the following question. Given an integer sequence $1 \leq s_1 \leq s_2 \leq s_3 \leq s_4 \leq s_5 \leq s_6$, does there exist a graph G for which $\text{ir}(G) = s_1$, $\gamma(G) = s_2$, $i(G) = s_3$, $\alpha(G) = s_4$, $\Gamma(G) = s_5$ and $\text{IR}(G) = s_6$? If such a graph G exists, then $(s_1, s_2, s_3, s_4, s_5, s_6)$ is called a **domination sequence**. Suppose we wish to determine which sequences are domination sequences.

By definition of the six parameters in the domination chain, a necessary condition for a sequence to be a domination sequence is the condition $1 \leq s_1 \leq s_2 \leq s_3 \leq s_4 \leq s_5 \leq s_6$. If $\text{ir}(G) = 1$, then, by Theorem 16.78, $\gamma(G) \leq 1$. Consequently, $\gamma(G) = 1$ which, in turn, implies that $i(G) = 1$. Hence, the condition $s_1 = 1$ implies that $s_3 = 1$ is also a necessary condition for a sequence to be a domination sequence. By Theorem 16.78, the condition $s_2 \leq 2s_1 - 1$ is a further necessary condition. If $\alpha(G) = 1$, then G is a complete graph, and so $\Gamma(G) = \text{IR}(G) = 1$. Therefore, the condition $s_4 = 1$ implies that $s_6 = 1$ is another necessary condition for a sequence to be a domination sequence. Surprisingly, these four trivial necessary conditions are, in fact, also sufficient conditions for a sequence to be a domination sequence, as was shown by Cockayne and Mynhardt [27].

Theorem 16.79 *A sequence $(s_1, s_2, s_3, s_4, s_5, s_6)$ of integers is a domination sequence if and only if the following four conditions hold:*

- (i) $1 \leq s_1 \leq s_2 \leq s_3 \leq s_4 \leq s_5 \leq s_6$,
- (ii) $s_1 = 1$ implies that $s_3 = 1$,
- (iii) $s_4 = 1$ implies that $s_6 = 1$, and
- (iv) $s_2 \leq 2s_1 - 1$.

Cockayne *et al.* [22] have shown that graphs exist having distinct values for all six parameters mentioned in the domination chain in Theorem 16.78. The independence number of a bipartite graph is, however, always equal to its upper irredundance number.

Theorem 16.80 *For every bipartite graph G , $\alpha(G) = \Gamma(G) = \text{IR}(G)$.*

Kieka (Christina) Mynhardt was born in Cape Town, South Africa in 1953. She obtained a doctorate in mathematics from Rand Afrikaans University in 1979. She started her academic career at the University of Pretoria in its Department of Mathematics in 1976, moved to the Department of Mathematics, Applied Mathematics and Astronomy at the University of South Africa (UNISA) in 1978, and was promoted to full professor there in 1992. In 1994, she became the first female National Research Foundation A-rated scientist in South Africa. She is currently a professor in the Department of Mathematics and Statistics at the University of Victoria, Canada. She is best known for her significant contributions to domination in graphs, a field in which she is a world leader.



Biographic note 83: Kieka Mynhardt (1953–present)

Proof Let $G = (V, E)$ be a bipartite graph with partite sets \mathcal{L} and \mathcal{R} . Let X be an IR-set of G , and let I be the set of isolated vertices in $G[X]$. Furthermore, let

$$\mathcal{L}_1 = I \cap \mathcal{L}, \quad \mathcal{L}_2 = (X \cap \mathcal{L}) \setminus I, \quad \mathcal{R}_1 = I \cap \mathcal{R}, \quad \mathcal{R}_2 = (X \cap \mathcal{R}) \setminus I,$$

where one or more of these sets may be empty. Since X is an irredundant set, $\text{pn}[x, X] \neq \emptyset$ for each vertex $x \in X$. In particular, $\text{pn}[x, X] \neq \emptyset$ for each vertex $x \in \mathcal{R}_2$. Let x be an arbitrary vertex in \mathcal{R}_2 . Since x is not isolated in $G[X]$, we note that $\text{ipn}[x, X] = \emptyset$, and so $\text{epn}[x, X] \neq \emptyset$. Let $x' \in \text{epn}[x, X]$. Then, $x' \in V \setminus X$ and $N(x') \cap X = \{x\}$. Moreover, since $x \in \mathcal{R}$, we note that $x' \in \mathcal{L}$. Let $A = \{x' \mid x \in \mathcal{R}_2\}$. Then $|A| = |\mathcal{R}_2|$ and $A \subset \mathcal{L}$. Furthermore, no vertex of A is adjacent to any vertex of \mathcal{R}_1 . Therefore the set $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{R}_1 \cup A$ is an independent set of G . Hence, $\alpha(G) \geq |\mathcal{L}_1| + |\mathcal{L}_2| + |\mathcal{R}_1| + |A| = |X| = \text{IR}(G)$. By Theorem 16.78, $\alpha(G) \leq \text{IR}(G)$. Consequently, $\alpha(G) = \Gamma(G) = \text{IR}(G)$. ■

16.4.3 Irredundance perfect graphs

It remains an open problem to characterise the $\langle \text{ir}, \gamma \rangle$ -graphs. A necessary and sufficient forbidden subgraph list characterising $\langle \text{ir}, \gamma \rangle$ -graphs is impossible to obtain since the addition of a new vertex adjacent to all vertices of a graph G produces a graph H containing G as an induced subgraph with $\text{ir}(H) = \gamma(H) = 1$. The first result involving forbidden subgraphs that implies equality of the parameters ir and γ was that presented by Laskar and Pfaff [80] who proved the following result.

Theorem 16.81 Every chordal graph that contains neither of the graphs $T_{16.28}$ and $G_{16.29}$ in Figure 16.17 as an induced subgraph is an $\langle \text{ir}, \gamma \rangle$ -graph.

A graph G is called **irredundance perfect** if $\text{ir}(H) = \gamma(H)$ for every induced subgraph H of G . As an immediate consequence of Theorem 16.72, we have the following two corollaries.

Corollary 16.82 A chordal graph G is irredundance perfect if and only if G contains neither of the graphs $T_{16.28}$ and $G_{16.29}$ in Figure 16.17 as an induced subgraph.

Corollary 16.83 A tree T is irredundance perfect if and only if T is $T_{16.28}$ -free, where $T_{16.28}$ is the tree shown in Figure 16.17.

Corollary 16.82 implies that the graph $G_{16.29}$ is a graph G of smallest order, namely 7, for which $\text{ir}(G) < \gamma(G)$, while Corollary 16.83 implies that the tree $T_{16.28}$ is a tree T of smallest order, namely 11, for which $\text{ir}(T) < \gamma(T)$.

Favaron [32] provided the following sufficient condition for a graph G to have $\text{ir}(G) = \gamma(G) = i(G)$.

Theorem 16.84 If a graph G is both claw-free and $G_{16.29}$ -free, where $G_{16.29}$ is the graph shown in Figure 16.17, then $\text{ir}(G) = \gamma(G) = i(G)$.

The requirement that G is both claw-free and $G_{16.29}$ -free is essential in Theorem 16.84. For example, the graph $G_{16.29}$ is claw-free but $\text{ir}(G_{16.29}) < \gamma(G_{16.29})$, while the tree $T_{16.28}$ is $G_{16.29}$ -free but $\text{ir}(T_{16.28}) < \gamma(T_{16.28})$. Favaron [32] also established a sufficient condition for a graph to be irredundance perfect in terms of six forbidden subgraphs.

Theorem 16.85 Every graph that does not contain the graphs P_6 , C_6 , $2P_4$ or any of the graphs $G_{16.30}$, $G_{16.31}$, and $G_{16.32}$ in Figure 16.18 as induced subgraphs is irredundance perfect.

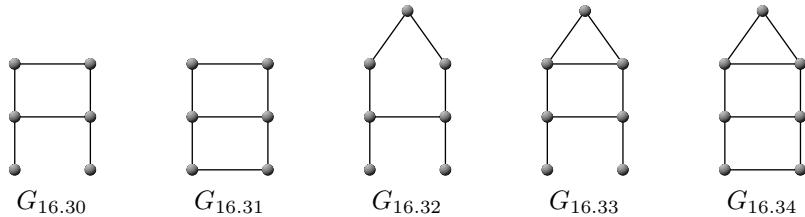


Figure 16.18: Forbidden subgraphs for irredundance perfect graphs.

Favaron [32], however, conjectured as follows that not all six graphs in Theorem 16.85 are needed as forbidden subgraphs for an irredundance perfect graph.

Conjecture 16.86 Every P_6 -free graph that does not contain the graphs $G_{16.30}$ or $G_{16.31}$ in Figure 16.18 as induced subgraphs is irredundance perfect.

The next two conjectures on irredundance perfect graphs were posed by Faudree *et al.* [31] and Puech [96], respectively.

Conjecture 16.87 Every P_5 -free graph is irredundance perfect.

Conjecture 16.88 Every P_6 -free graph with no induced subgraph isomorphic to the graphs $G_{16.33}$ or $G_{16.34}$ in Figure 16.18 is irredundance perfect.

Henning [59] proved [Conjecture 16.86](#) for a graph G such that $\text{ir}(H) \leq 4$ for every induced subgraph H of G . [Conjecture 16.86](#) was completely proven independently by Puech [96] as well as by Volkmann and Zverovich [124]. [Conjecture 16.87](#) follows from a result of Puech [96] that a graph is irredundance perfect if it does not contain P_6 and the graph obtained from the graph $G_{16.33}$ in [Figure 16.18](#) by deleting a vertex of degree 1 as induced subgraphs. [Conjecture 16.88](#) was proven by Volkmann and Zverovich [126]. We observe that [Conjecture 16.88](#) implies both [Conjecture 16.86](#) and [16.87](#), while [Conjectures 16.86](#) and [16.87](#) do not imply each other.

A graph G is **minimal irredundance imperfect** if G is not irredundance perfect and $\text{ir}(H) = \gamma(H)$ for every proper induced subgraph H of G . From [Theorem 16.41](#), a graph is domination perfect if and only if $\gamma(H) = i(H)$ for every induced subgraph H of G with $\gamma(H) = 2$. Henning [59] conjectured that a graph is irredundance perfect if and only if $\text{ir}(H) = \gamma(H)$ for every induced subgraph H of G with $\text{ir}(H) \leq 4$. This conjecture may be restated as follows: A graph is irredundance perfect if and only if every minimal irredundance imperfect graph G satisfies $\text{ir}(G) \leq 4$. This conjecture was disproven by Volkmann and Zverovich [125] who constructed the minimal irredundance imperfect graph $G_{16.35}$ in [Figure 16.19](#) of order 16 with $\text{ir}(G) = 5$.

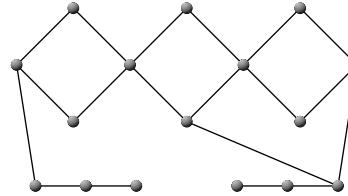


Figure 16.19: A minimal irredundance imperfect graph $G_{16.35}$ with $\text{ir}(G_{16.35}) = 5$ and $\gamma(G_{16.35}) = 6$.

Volkmann and Zverovich [125], in turn, posed the following conjecture.

Conjecture 16.89 *A graph is irredundance perfect if and only if $\text{ir}(H) = \gamma(H)$ for every induced subgraph H of G with $\text{ir}(H) \leq 5$.*

16.4.4 Bounds involving the minimum and maximum degrees

Cockayne and Mynhardt [28] obtained the following lower bound for the irredundance number of a graph in terms of its order and maximum degree.

Theorem 16.90 *If G is a graph of order n with maximum degree $\Delta \geq 2$, then $\text{ir}(G) \geq 2n/(3\Delta)$.*

Favaron [33] provided the following upper bound for the upper irredundance number of a graph in terms of its order and minimum degree.

Theorem 16.91 *If G is a graph of order n with minimum degree δ , then $\text{IR}(G) \leq n - \delta$.*

Proof Let $G = (V, E)$ be a graph of order n and with minimum degree δ . Let X be an IR-set of G and let $x \in X$. If x is isolated in $G[X]$, then $N(x) \subseteq V \setminus X$. Therefore, $\delta \leq d_G(x) = |N(x)| \leq |V \setminus X| = n - |X|$, or, equivalently, $IR(G) = |X| \leq n - \delta$. Hence, we may assume that no vertex in X is isolated in $G[X]$, for otherwise the desired result is immediate. Suppose that x is adjacent to k vertices in $G[X]$. Let A denote these k neighbours of x in X and let B denote the remaining neighbours of x , if any. Then, $B \subseteq V \setminus X$. Furthermore, $|A| = k$ and $|B| = d_G(x) - k \geq \delta - k$. Since X is an irredundant set, $pn[v, X] \neq \emptyset$ for each vertex $v \in X$. In particular, $pn[v, X] \neq \emptyset$ for each vertex $v \in A$. Let v be an arbitrary vertex in A . Since v is not isolated in $G[X]$, we note that $ipn[v, X] = \emptyset$, and so $epn[v, X] \neq \emptyset$. Let $v' \in epn[v, X]$. Then $v' \in V \setminus X$ and $N(v') \cap X = \{v\}$. Let $A' = \{v' \mid v \in A\}$. Then $A' \subseteq V \setminus X$, $A' \cap B = \emptyset$ and $|A'| \geq |A| = k$. Thus, $n - |X| = |V \setminus X| \geq |A'| + |B| \geq |A| + |B| = d_G(x) \geq \delta$. Equivalently, $IR(G) = |X| \leq n - \delta$. ■

The extremal graphs that achieve the lower bound in [Theorem 16.90](#), as well as the extremal graphs that achieve the lower bound in [Theorem 16.91](#), were characterised by [Cockayne and Mynhardt](#) [28].

16.5 Total domination

A **total dominating set**, abbreviated as a TD-set, of a graph $G = (V, E)$ is a set S of vertices of G for which each vertex $v \in V$ is adjacent to a vertex of S . That is, $N(S) = V$. The **total domination number** of G , denoted by $\gamma_t(G)$, is the minimum cardinality of a TD-set. A TD-set of cardinality $\gamma_t(G)$ is called a **γ_t -set of G** . Every graph G without isolated vertices has a TD-set, since $S = V$ is such a set. If X and Y are subsets of vertices in G , we say that **X totally dominates Y** if $Y \subseteq N(X)$. In particular, if X totally dominates V , then X is a TD-set of G . A **minimal total dominating set** of G is a TD-set that contains no TD-set of G as a proper subset. The **upper total domination number** $\Gamma_t(G)$ of a graph G is the maximum cardinality of a minimal TD-set of G . A minimal TD-set of cardinality $\Gamma_t(G)$ we call a **Γ_t -set of G** . The notion of total domination in graphs was introduced by [Cockayne et al.](#) [19] and is now well-studied in graph theory.

The literature on the subject of total domination in graphs has been surveyed and detailed in [69]. A survey of total domination in graphs can also be found in [62].

To illustrate the definitions detailed above, consider once again the Petersen graph $G_{16.1}$ in [Figure 16.2](#). The set $S = N[v_1] = \{v_1, u_2, v_2, v_5\}$ is a TD-set of $G_{16.1}$, and so $\gamma_t(G_{16.1}) \leq |S| = 4$. Since $G_{16.1}$ is a cubic graph, each vertex totally dominates three vertices, implying that no set of three or fewer vertices totally dominates all ten vertices in the graph, and so $\gamma_t(G_{16.1}) \geq 4$. Consequently, $\gamma_t(G_{16.1}) = 4$. The sets $\{u_1, u_2, u_3, u_4, u_5\}$ and $\{v_1, v_2, v_4, u_1, u_3, u_5\}$ are both also minimal TD-sets of the Petersen graph $G_{16.1}$ in [Figure 16.2](#). Hence, the Petersen graph contains minimal TD-sets of cardinalities 4, 5 and 6. We show later (see [Theorem 16.115](#)) that the cardinality of a minimal TD-set in a 3-regular graph of order n cannot exceed $n/(2-1/3) = \frac{3}{5}n$. In particular, every minimal TD-set of the Petersen graph contains at most six vertices, and so $\Gamma_t(G_{16.1}) \leq 6$. Consequently, $\Gamma_t(G_{16.1}) = 6$.

16.5.1 Minimal total dominating sets

Let $G = (V, E)$ be a graph. In this section, we follow the graph theory terminology and concepts in [69]. Let $S \subseteq V$ be a subset of vertices in G and let v be a vertex in V . The **open S -private neighbourhood** of v is the set $\text{pn}(v, S) = \{w \in V \mid N(w) \cap S = \{v\}\}$. We remark that the sets $\text{pn}[v, S] \setminus S$ and $\text{pn}(v, S) \setminus S$ are equivalent and define the **S -external private neighbourhood** of v to be this set, abbreviated $\text{epn}[v, S]$ or $\text{epn}(v, S)$. The **open S -internal private neighbourhood** is defined by $\text{ipn}(v, S) = \text{pn}(v, S) \cap S$. We define an **S -external private neighbour** of v to be a vertex in $\text{epn}(v, S)$ and an **S -internal private neighbour** of v to be a vertex in $\text{ipn}(v, S)$. The following property of minimal TD-sets was established by Cockayne *et al.* [19].

Proposition 16.92 ([19]) *Let S be a TD-set of a graph G . Then S is a minimal TD-set of G if and only if, for each $v \in S$, $\text{epn}(v, S) \neq \emptyset$ or $\text{ipn}(v, S) \neq \emptyset$.*

Proof Suppose, first, that S is a minimal TD-set of G . Then the set $S \setminus \{v\}$ is not a TD-set of G for each vertex v of S . Hence, there is a vertex $w \in V$ that is adjacent to no vertex of $S \setminus \{v\}$. Since v is adjacent to a vertex of $S \setminus \{v\}$, we note that $v \neq w$. If $w \in S \setminus \{v\}$, then $w \in \text{ipn}(v, S)$. If $w \in V \setminus S$, then, since every vertex not in S is adjacent to some vertex of S , w is adjacent to v and to no other vertex of S , and so $w \in \text{epn}(v, S)$. Conversely, if $\text{epn}(v, S) \neq \emptyset$ or $\text{ipn}(v, S) \neq \emptyset$ for each vertex $v \in S$, then for each such vertex v , the set $S \setminus \{v\}$ is not a TD-set of G . Hence, S is a minimal TD-set. ■

If we impose the requirement in Proposition 16.92 that the minimal TD-set be a minimum TD-set, then the following result in [61] shows that the property of a minimal TD-set established in Proposition 16.92 can be strengthened as follows. We shall need this property of minimum TD-sets in subsequent proofs.

Proposition 16.93 *If G is a connected graph of order $n \geq 3$, and G is not the complete graph K_n , then G has a minimum TD-set S such that the following holds: For every vertex $v \in S$, $\text{epn}(v, S) \neq \emptyset$ or there exists a vertex $w \in \text{ipn}(v, S)$ with $\text{epn}(w, S) \neq \emptyset$.*

Proof Let $G = (V, E)$ be a connected graph of order $n \geq 3$ other than the complete graph K_n . For a subset $S \subseteq V$, let $\xi(S)$ be the number of vertices v in S such that $\text{epn}(v, S) \neq \emptyset$ or v is adjacent to a vertex $u \in S$ with $\text{epn}(u, S) \neq \emptyset$. Among all γ_t -sets of G , let S be chosen so that

- (i) the number of edges in $G[S]$ is a maximum, and
- (ii) subject to (i), $\xi(S)$ is maximised.

Let A be the set of vertices $v \in S$ with $\text{epn}(v, S) \neq \emptyset$. Furthermore, let B be a minimum set of vertices of $S \setminus A$ such that $G[A \cup B]$ has no isolated vertex. Necessarily, $|B| \leq |A|$. Finally, let $C = S \setminus (A \cup B)$. Suppose $C = \emptyset$, and so $S = A \cup B$. If $v \in B$, then by the minimality of the set B , there exists a vertex $w \in A$ whose only neighbour in $G[A \cup B] = G[S]$ is the vertex v , implying that $w \in \text{ipn}(v, S)$ and $\text{epn}(w, S) \neq \emptyset$. Therefore, for every vertex $v \in S$, $\text{epn}(v, S) \neq \emptyset$ or there exists a vertex $w \in \text{ipn}(v, S)$ with $\text{epn}(w, S) \neq \emptyset$. Hence, it suffices to show that $C = \emptyset$.

Suppose, to the contrary, that $C \neq \emptyset$. For each vertex $v \in C$ we note that $\text{epn}(v, S) = \emptyset$, and so, by [Proposition 16.92](#), $\text{ipn}(v, S) \neq \emptyset$. Hence, since $G[A \cup B]$ has no isolated vertex, we infer that $G[C] \cong mK_2$ for some $m \geq 1$ and that each vertex of C has degree 1 in $G[S]$. Let $u_i v_i$, $1 \leq i \leq m$, be the distinct edges of $G[C]$. For $i \in [m]$, let $N_i = N(\{u_i, v_i\}) \setminus \{u_i, v_i\}$. Since G is connected and $n \geq 3$, we note that $N_i \neq \emptyset$. Furthermore, since each vertex of C has degree 1 in $G[S]$, we note that $N_i \subseteq V \setminus S$. For $i \in [m]$, we select an arbitrary vertex $w_i \in N_i$. Renaming vertices if necessary, we may assume that $v_i w_i \in E$. If w_i is adjacent to some vertex of S different from u_i and v_i , then $(S \setminus \{u_i\}) \cup \{w_i\}$ is a γ_t -set of G whose induced subgraph contains more edges than the subgraph induced by S , contradicting our choice of S . Hence each vertex of N_i is adjacent to no vertex of $S \setminus \{u_i, v_i\}$. Therefore, the sets N_1, \dots, N_m are pairwise disjoint subsets of S . Moreover, since $\text{epn}(v, S) = \emptyset$ for every vertex $v \in C$, we have that $N(w) \cap S = \{u_i, v_i\}$ for every vertex $w \in N_i$.

Suppose $S = \{u_1, v_1\}$. Since $G \neq K_n$, the set N_1 contains two nonadjacent vertices. We may assume w_1 is chosen so that it is not adjacent to some vertex of N_1 . Then $D = \{v_1, w_1\}$ is a γ_t -set of G such that the number of edges in $G[D]$ is equal to the number in $G[S]$, but with $\xi(D) > \xi(S)$, contradicting our choice of S . Hence, $S \neq \{u_1, v_1\}$. Since G is connected, we may choose the vertex w_i to be a vertex in N_i that is adjacent to a vertex $y_i \in V \setminus (S \cup N_i)$.

We show next that there is no edge joining a vertex of N_i and a vertex of N_j where $1 \leq i < j \leq m$. If this is not the case, then renaming vertices if necessary, we may assume $w_1 w_2 \in E$. But then $(S \setminus \{u_1, u_2\}) \cup \{w_1, w_2\}$ is a γ_t -set of G whose induced subgraph contains more edges than the subgraph induced by S , a contradiction. Hence, there is no edge joining a vertex of N_i and a vertex of N_j . Thus, for each $i \in [m]$, the vertex $y_i \notin \cup_{j=1}^m N_j$, and so y_i is adjacent to a vertex of $S \setminus C$.

If, for $i \in [m]$, w_i is adjacent to every other vertex of N_i , then $(S \setminus \{u_i, v_i\}) \cup \{w_i, y_i\}$ is a γ_t -set of G whose induced subgraph contains more edges than the subgraph induced by S , a contradiction. Therefore, w_i is not adjacent to at least one vertex of N_i .

We now consider the set $S' = (S \setminus \{u_1\}) \cup \{w_1\}$. Since S' is a TD-set of G of cardinality $\gamma_t(G)$, we note that S' is a γ_t -set of G . Furthermore, since w_1 is not adjacent to at least one vertex of N_1 , we note that $\text{epn}(v_1, S') \neq \emptyset$. If $\text{epn}(a, S') \neq \emptyset$ for all $a \in A$, then $G[S']$ has the same size as $G[S]$ but with $\xi(S') \geq |A| + |B| + |\{v_1\}| > |A| + |B| = \xi(S)$, a contradiction. Hence, $\text{epn}(a, S') = \emptyset$ for some $a \in A$. For such a vertex $a \in A$, the vertex w_1 therefore dominates the set $\text{epn}(a, S)$. Let a be an arbitrary such vertex in A . We show that there is a vertex $a' \in \text{ipn}(a, S')$ such that $\text{epn}(a', S') \neq \emptyset$. Since $\text{epn}(a, S') = \emptyset$, [Proposition 16.92](#) implies that $\text{ipn}(a, S') \neq \emptyset$. Let $a^* \in \text{epn}(a, S)$. Furthermore, let $a' \in \text{ipn}(a, S')$, and so $a' \in S$ and $N(a') \cap S = \{a\}$. If $a' \in B$, then $(S' \setminus \{a'\}) \cup \{a^*\}$ is a γ_t -set of G whose induced subgraph contains more edges than the subgraph induced by S , a contradiction. As a result, $a' \in A$. If w_1 dominates the set $\text{epn}(a', S)$, then $(S' \setminus \{a', u_1\}) \cup \{a^*, w_1\}$ is a γ_t -set of G whose induced subgraph contains more edges than the subgraph induced by S , a contradiction. Therefore, $\text{epn}(a', S') \neq \emptyset$. Hence, for each vertex $a \in A$ with $\text{epn}(a, S') = \emptyset$, there exists a vertex $a' \in \text{ipn}(a, S')$ with $\text{epn}(a', S') \neq \emptyset$. As a result, the set S' is a γ_t -set of G such that the number of edges in $G[S']$ is

equal to the number in $G[S]$, but with $\xi(S') > \xi(S)$, contradicting our choice of S . Consequently, $C = \emptyset$. \blacksquare

For a subset S of vertices in a graph G , the **open boundary** of S is the set $\text{OB}(S) = \{v \mid |\text{N}(v) \cap S| = 1\}$; that is, $\text{OB}(S)$ is the set of vertices totally dominated by exactly one vertex in S . A minimal TD-set can be characterised by its open boundary as shown in the following result by Hedetniemi *et al.* [58].

Proposition 16.94 *If S is a TD-set of a graph G , then S is a minimal TD-set of G if and only if $\text{OB}(S)$ totally dominates S .*

Proof Suppose first that $\text{OB}(S)$ totally dominates S . Thus, $S \subseteq \text{N}(\text{OB}(S))$, and so every vertex in S has a neighbour in $\text{OB}(S)$. Let $v \in S$. Then v is adjacent to a vertex $u \in \text{OB}(S)$, and so $\text{N}(u) \cap S = \{v\}$. If $u \notin S$, then $u \in \text{epn}(v, S)$. If $u \in S$, then $u \in \text{ipn}(v, S)$. Hence, $\text{epn}(v, S) \neq \emptyset$ or $\text{ipn}(v, S) \neq \emptyset$ for every vertex $v \in S$. Thus, by Proposition 16.92, S is a minimal TD-set.

To prove the necessity, suppose that S is a minimal TD-set and let $v \in S$. By Proposition 16.92, $\text{epn}(v, S) \neq \emptyset$ or $\text{ipn}(v, S) \neq \emptyset$. If $\text{epn}(v, S) \neq \emptyset$, then v has a neighbour u outside S such that $\text{N}(u) \cap S = \{v\}$, and so $u \in \text{OB}(S)$. If, on the other hand, $\text{ipn}(v, S) \neq \emptyset$, then v has a neighbour u inside S such that $\text{N}(u) \cap S = \{v\}$, and so $u \in \text{OB}(S)$. In both cases, $\text{OB}(S)$ totally dominates v . \blacksquare

A classical result in domination theory is that if S is a minimal dominating set of a graph G with no isolated vertex, then $V(G) \setminus S$ is also a dominating set of G (see Theorem 16.3). Thus, the vertex set of every graph without any isolates can be partitioned into two dominating sets. It is not the case, however, that the vertex set of every graph can be partitioned into a dominating set and a TD-set, even if every vertex has degree at least 2. For example, the vertex set of a 5-cycle C_5 cannot be partitioned into a dominating set and a TD-set. This counter example is the only exception, as shown by Henning and Southey [65].

Theorem 16.95 *If G is a graph with minimum degree at least two that contains no C_5 -component, then $V(G)$ can be partitioned into a dominating set and a total dominating set.*

Proof Let $G = (V, E)$ be a graph with minimum degree $\delta \geq 2$ that contains no C_5 -component. For a subset $S \subseteq V$ and a vertex $v \in S$, we say v is an *S-bad* vertex if $\text{N}[v] \subseteq S$, and that v is an *S-weak* vertex if v has degree 1 in $G[S]$ and its neighbour in S is an *S-bad* vertex. Among all TD-sets of G , let S be chosen so that

- (i) the number of *S-bad* vertices is minimised, and
- (ii) subject to (i), the number of *S-weak* vertices is minimised.

Assume that there is at least one *S-bad* vertex. Let v be such a vertex. If v has no *S-weak* neighbour, then $S' = S \setminus \{v\}$ is a TD-set of G with fewer *S'-bad* vertices than *S-bad* vertices, contradicting our choice of S . Hence we may assume that every *S-bad* vertex has at least one *S-weak* neighbour.

Let w be an *S-weak* vertex. Since $\delta \geq 2$, w is adjacent to at least one vertex in $V \setminus S$. If $\text{epn}(w, S) = \emptyset$, then $S' = S \setminus \{w\}$ is a TD-set of G with fewer *S'-bad* vertices than *S-bad* vertices, contradicting our choice of S . Hence, $|\text{epn}(w, S)| \geq 1$. For each *S-weak* vertex w , let $w' \in \text{epn}(w, S)$. Since $\delta \geq 2$, w' is adjacent to at least one vertex in $V \setminus S$ and $\text{N}[w'] \setminus \{w\} \subseteq V \setminus S$.

We show next that every S -weak vertex has degree 2 in G . As defined earlier, let w be an S -weak vertex and suppose that $d(w) \geq 3$. Then, $S' = S \cup \{w'\}$ is a TD-set of G that satisfies condition (i), but with fewer S' -weak vertices than S -weak vertices, contradicting our choice of S . Hence, every S -weak vertex has degree 2 in G .

As defined earlier, let v be an S -bad vertex. Then v has at least one S -weak neighbour. For $k \geq 1$, let $W = \{w_1, \dots, w_k\}$ be the set of all S -weak neighbours of v . Then, $N(w_i) = \{v, w'_i\}$ for $i \in [k]$. Let $W' = \{w'_1, \dots, w'_k\}$. If every vertex in W' is adjacent to a vertex in $V \setminus (S \cup W')$, then $S' = (S \cup W') \setminus \{v\}$ is a TD-set of G with fewer S' -bad vertices than S -bad vertices, contradicting our choice of S . Hence, renaming vertices if necessary, we may assume that $N[w'_1] \subseteq W' \cup \{w_1\}$ and that $w'_1 w'_2$ is an edge of G .

If $d(v) \geq 3$, then $S' = (S \cup \{w'_1, w'_2\}) \setminus \{w_1, w_2\}$ is a TD-set of G with fewer S' -bad vertices than S -bad vertices, contradicting our choice of S . Thus, $d(v) = 2$, implying that $w' = \{w'_1, w'_2\}$ and $N(w'_1) = \{w_1, w'_2\}$. Hence, each of v, w_1, w'_1 and w_2 has degree 2 in G and $C : v, w_1, w'_1, w'_2, v$ is an induced 5-cycle in G .

Since G contains no C_5 -component, the vertex w'_2 is adjacent to some vertex not in the 5-cycle C . But then $S' = (S \cup \{w'_1, w'_2\}) \setminus \{v, w_1\}$ is a TD-set of G with fewer S' -bad vertices than S -bad vertices, contradicting our choice of S . We deduce, therefore, that the TD-set S contains no S -bad vertices. Hence, $V \setminus S$ is a dominating set of G . ■

We remark that the minimum degree condition of [Theorem 16.95](#) cannot be relaxed to $\delta \geq 1$. For example, the 2-corona of an arbitrary graph cannot be partitioned into a dominating set and a total dominating set; recall that the 2-corona of a graph H is the graph obtained from H by attaching a path of length 2 to each vertex of H so that the resulting paths are vertex disjoint.

16.5.2 Bounds on the total domination number

In this section, we investigate bounds on the total domination number of a graph in terms of its order. [Cockayne et al.](#) [19] obtained the following upper bound on the total domination number of a connected graph in terms of the order of the graph.

Theorem 16.96 *If G is a connected graph of order $n \geq 3$, then $\gamma_t(G) \leq \frac{2}{3}n$.*

[Brigham et al.](#) [11] characterised connected graphs of order at least 3 with total domination number exactly two-thirds their order as follows.

Theorem 16.97 *If G be a connected graph of order $n \geq 3$, then $\gamma_t(G) = \frac{2}{3}n$ if and only if $G \in \{C_3, C_6\}$ or G is the 2-corona of some connected graph H .*

The results of [Theorems 16.96](#) and [16.97](#) may readily be deduced from the property of minimum TD-sets stated earlier in [Proposition 16.93](#).

Proof of Theorems 16.96 and 16.97 Let $G = (V, E)$ be a connected graph of order $n \geq 3$. If $G = K_n$, then $\gamma_t(G) = 2 \leq \frac{2}{3}n$. Furthermore, if $\gamma_t(G) = \frac{2}{3}n$, then $G = C_3$. Hence, we may assume that $G \neq K_n$. Let S be a γ_t -set of G satisfying the statement of [Proposition 16.93](#), let $A = \{v \in S \mid \text{epn}(v, S) \neq \emptyset\}$ and let $B = S \setminus A$. By [Proposition 16.93](#), each vertex $v \in B$ is adjacent to at

least one vertex of A which is adjacent to v but to no other vertex of S . Hence, $|S| = |A| + |B| \leq 2|A|$. Let C be the set of all S -external private neighbours. Thus, if $w \in C$, then $w \in \text{epn}(v, A)$ for some vertex $v \in A$. Furthermore, $C \subseteq V \setminus S$ and since each vertex of A has at least one S -external private neighbour, we note that $|C| \geq |A|$. As a result,

$$n - |S| = |V \setminus S| \geq |C| \geq |A| \geq \frac{1}{2}|S|, \quad (16.11)$$

and so $\gamma_t(G) = |S| \leq \frac{2}{3}n$. This establishes the bound of [Theorem 16.96](#). Suppose that $\gamma_t(G) = \frac{2}{3}n$ (and still $G \neq K_n$). Then we must have equality throughout the inequality chain (16.11). In particular, $|A| = |B| = |C|$ and $V \setminus S = C$. We deduce that each vertex of A therefore has degree 2 in G and is adjacent to a unique vertex of B and a unique vertex of C . Hence, G contains a spanning subgraph that consists of r disjoint copies of a path P_3 on three vertices, where $r = \frac{n}{3}$.

Suppose that both sets B and C contain a vertex of degree 2 or more in G . Then, since G is connected, the graph G contains a spanning subgraph H where $H = C_6$ or $H = P_9 \cup (r-3)P_3$. If $H = P_9 \cup (r-3)P_3$, then $\gamma_t(H) = 5 + 2(r-3) = 2r-1 < \frac{2}{3}n$. Since adding edges to a graph does not increase its total domination number, $\gamma_t(G) \leq \gamma_t(H) < \frac{2}{3}n$, contrary to our assumption. As a consequence, $H = C_6$. But then G can contain no additional edges not in H , and so $G = H = C_6$.

Hence, we may assume that every vertex in B has degree 1 in G or every vertex in the set C has degree 1 in G , for otherwise $G = C_6$. If every vertex of B has degree 1 in G , then by the connectivity of G , the subgraph $G[C]$ is connected and G is the 2-corona of the graph $G[C]$, while if every vertex of C has degree 1 in G , then the subgraph $G[B]$ is connected and G is the 2-corona of the graph $G[B]$. ■

16.5.3 Bounds for graphs with minimum degree at least two

If G is a graph of order n that consists of a disjoint union of 3-cycles and 6-cycles, then $\gamma_t(G) = \frac{2}{3}n$. Hence, the upper bound in [Theorem 16.96](#) cannot be improved if we simply restrict the minimum degree to be two. If, however, we impose the additional restriction that G is connected, then [Sun \[112\]](#) showed that the upper bound in [Theorem 16.96](#) can be improved.

Theorem 16.98 *If G is a connected graph of order n with minimum degree at least two, then $\gamma_t(G) \leq \lfloor \frac{4}{7}(n+1) \rfloor$.*

The bound of [Sun \[112\]](#) in [Theorem 16.98](#) can be improved slightly if we forbid six graphs of small orders. Let $G_{16.36}$ and $G_{16.37}$ be the two graphs shown in [Figure 16.20\(a\)](#) and [\(b\)](#), respectively. We omit the proof of the following result due to [Henning \[61\]](#), which is relatively lengthy and technical.

Theorem 16.99 *If $G \notin \{C_3, C_5, C_6, C_{10}, G_{16.36}, G_{16.37}\}$ is a connected graph of order n with minimum degree at least two, then $\gamma_t(G) \leq \frac{4}{7}n$.*

In order to characterise connected graphs of large order with total domination number exactly four-sevenths their order, let $\mathcal{G}_{\geq 2}$ be the family of all graphs that



Figure 16.20: The graphs $G_{16.36}$ and $G_{16.37}$.

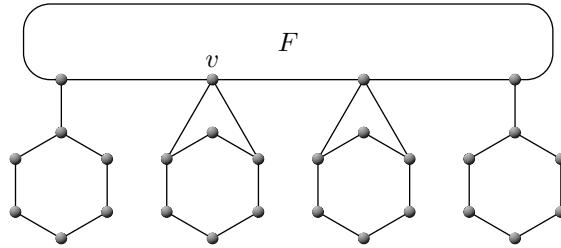


Figure 16.21: A graph $G_{16.38}$ in the family $\mathcal{G}_{\geq 2}$.

can be obtained from a connected graph F of order at least 3, as follows. For each vertex v of F , add a 6-cycle and join v either to one vertex of this cycle or to two vertices at distance 2 on this cycle. A graph $G_{16.38}$ in the family $\mathcal{G}_{\geq 2}$ is illustrated in [Figure 16.21](#) (here the graph F is a 4-cycle). The following characterisation is also due to Henning [61].

Theorem 16.100 *If G is a connected graph of order $n > 14$ with minimum degree at least two and $\gamma_t(G) = \frac{4}{7}n$, then $G \in \mathcal{G}_{\geq 2}$.*

We observe that every graph in the family $\mathcal{G}_{\geq 2}$ of extremal graphs of large order that achieve equality in the bound of [Theorem 16.100](#) has cut-vertices as well as induced 6-cycles. It is therefore a natural question to ask whether this upper bound of $\frac{4}{7}n$ can be improved if we restrict our attention to 2-connected graphs or to connected graphs that are C_6 -free. This question was answered in the affirmative by Henning and Yeo [68].

Theorem 16.101 *If G is a 2-connected graph of order $n \geq 19$, then $\gamma_t(G) \leq \frac{6}{11}n$.*

To illustrate the sharpness of [Theorem 16.101](#), let $k \geq 2$ be an integer and let $\mathcal{G}_{2\text{conn}}$ be the family of all graphs that can be obtained from a 2-connected graph H of order $2k$ that contains a perfect matching M , as follows. For each edge $e = uv$ in the matching M , subdivide the edge e three times, add a 6-cycle, select two vertices u' and v' at distance 2 apart on this cycle, and join u to u' and v to v' . The edges uv' and $u'v$ are optional edges that may be added. Let G denote the resulting graph of order $n = 11k$. Then, $\gamma_t(G) = 6k = \frac{6}{11}n$. A graph $G_{16.39}$ in the family $\mathcal{G}_{2\text{conn}}$ with $k = 4$ that is obtained from an 8-cycle H is shown in [Figure 16.22](#). The following result is also due to Henning and Yeo [68].

Theorem 16.102 *If G is a connected graph of order $n \geq 19$ with minimum degree at least two that has no induced 6-cycle, then $\gamma_t(G) \leq \frac{6}{11}n$.*

Anders Yeo was born in Australia on 14 October 1970 and schooled in Denmark. He obtained a doctorate in mathematics and computer science at Odense University, Denmark in 1997 under the supervision of Jørgen Bang-Jensen. He accepted a one-year postdoctoral position at the University of Victoria, Canada, followed by a two-year postdoctoral position at the University of Aarhus, Denmark. He started his academic career as a lecturer in the Department of Computer Science at Royal Holloway, University of London in 2001 where he was rapidly promoted to a Reader in 2003. He is currently a full professor at the University of Southern Denmark in Odense. He is an expert in graph theory, digraphs, algorithms, combinatorial optimisation, combinatorics and fixed parameter tractability. He has co-authored many journal papers, about half of them with Gregory Gutin. He has co-authored two books, *Total domination in graphs* and *Transversals in linear uniform hypergraphs*, both with Michael Henning. Yeo is an accomplished sportsman and has been included in the Danish national squash team.



Biographic note 84: Anders Yeo (1970–present)

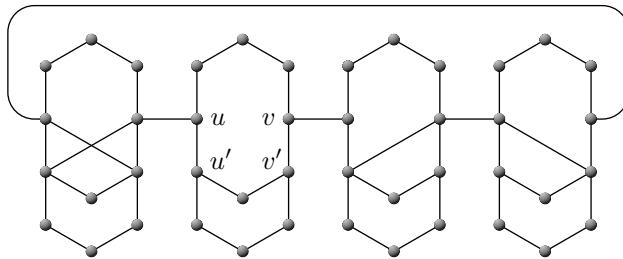


Figure 16.22: A graph $G_{16,39}$ in the family $\mathcal{G}_{2\text{conn}}$.

To illustrate the sharpness of [Theorem 16.102](#), let $\mathcal{G}_{C_6\text{free}}$ be the family of all graphs that can be obtained from a connected C_6 -free graph H of order at least two, as follows. For each vertex v of H , add a 10-cycle and join v to exactly one vertex of this cycle. Each graph $G \in \mathcal{H}$ is a connected C_6 -free graph of order n with $\gamma_t(G) = \frac{6}{11}n$. A graph $G_{16,40}$ in the family $\mathcal{G}_{C_6\text{free}}$ is illustrated in [Figure 16.23](#) (here the graph H is a 4-cycle).

16.5.4 Bounds for graphs with minimum degree at least three

If we restrict the minimum degree to be at least three, then the upper bound in [Theorem 16.100](#) on the total domination number of a graph with large order can be improved.

Theorem 16.103 *If G is a graph of order n with minimum degree at least three, then $\gamma_t(G) \leq \frac{n}{2}$.*

The proof we present of [Theorem 16.103](#) is an elegant proof due to Archdeacon *et al.* [3]. Their key lemma is a result about bipartite graphs. Their proof is by

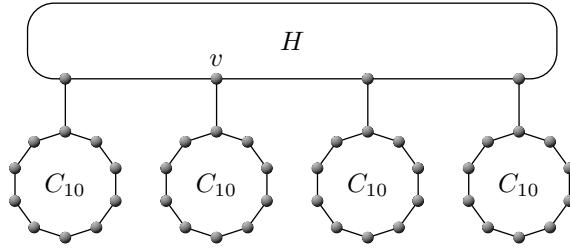


Figure 16.23: A graph $G_{16.40}$ in the family $\mathcal{G}_{C_6\text{free}}$.

induction on $|V(G)| + |E(G)|$ and uses Brooks' colouring of Theorem 14.13 and some clever counting arguments. We prove here a slight strengthening of their lemma. Recall that for disjoint vertex subsets X and Y , $[X, Y]$ denotes the set of all edges of G between X and Y .

Lemma 16.104 *Let $\delta \in \{1, 2, 3\}$ and let G be a bipartite graph with partite sets (X, Y) whose vertices in Y are of degree at least δ . Then there exists a set $A \subseteq X$ of size at most $|X \cup Y| / (\delta + 1)$ that dominates Y .*

Proof Let $|X| = x$ and $|Y| = y$. When $\delta = 1$, we choose for each vertex in Y an adjacent vertex in X to form the set A , and so $|A| \leq \min\{x, y\} \leq \frac{1}{2}(x + y)$ as desired. Hence we may assume $\delta \in \{2, 3\}$.

We proceed by induction on $|V(G)| + |E(G)|$. The smallest graph described by the lemma is $K_{1,\delta}$, for which the statement holds. This establishes our base case. If there exists a vertex v in Y of degree at least $\delta + 1$, then we delete any edge e incident with v . Applying the inductive hypothesis to the graph $G - e$, there exists a set $A \subseteq X$ of the desired size that dominates Y in $G - e$ and therefore also in G . So we may assume the vertices in Y are all of degree exactly δ .

If there exists an isolated vertex $v \in X$, then we apply the inductive hypothesis to $G - v$ to produce the desired set A . So we may assume that each vertex in X has degree at least 1. If there exists a vertex v in X of degree at least δ , then delete that vertex and all its neighbours. Let G' denote the resulting graph, and let $X' = X \setminus \{v\}$ and $Y' = Y \setminus N(v)$. Applying the inductive hypothesis to G' , there exists a set $A' \subseteq X'$ of size at most $|X' \cup Y'| / (\delta + 1) \leq |X \cup Y| / (\delta + 1) - 1$. Adding v into the subset A from this smaller graph G' yields our desired subset for G . So we may assume each vertex in X has degree at most $\delta - 1$.

Suppose $\delta = 2$. Then each vertex in X has degree exactly 1 and each vertex in Y has degree 2. Thus, $|[X, Y]| = x = 2y$. For each vertex in Y we now choose an adjacent vertex in X to form the set A . Then $|A| \leq y = \frac{1}{3}(x + y)$ as desired.

Suppose $\delta = 3$. Then each vertex in X has degree 1 or 2 and each vertex in Y has degree 3. For $i \in [2]$, let X_i denote the vertices in X of degree i , and let $|X_i| = x_i$. Then, $x = x_1 + x_2$ and $|[X, Y]| = x_1 + 2x_2 = 3y$. Our aim is therefore to find a set $A \subseteq X$ of size at most $(x + y)/4 = x_1/3 + 5x_2/12$ that dominates Y .

Let F be the graph with vertex set $V(F) = X_2$ and where two vertices are adjacent in F if and only if they have a common neighbour in G . Since each vertex in X_2 has degree 2 in G and each vertex in Y has degree 3, $\Delta(F) \leq 4$. Furthermore, since each vertex in Y has degree 3, no component of F is the complete graph K_5 . Hence, by Theorem 14.13, $\chi(F) \leq 4$, and so F has an independent set S of size at least $x_2/4$. Since the vertices in the set S have disjoint neighbourhoods in G ,

$|N_G(S)| = 2|S|$. For each vertex $y \in Y \setminus N_G(S)$, we choose an adjacent vertex in $X \setminus S$ and call the resulting set of such vertices S' . Then $A = S \cup S'$ dominates Y . Since $|S'| \leq |Y \setminus N_G(S)| = y - 2|S|$, we have $|A| = |S| + |S'| \leq y - |S| \leq y - x_2/4 = (x_1 + 2x_2)/3 - x_2/4 = x_1/3 + 5x_2/12$, as desired. ■

As a consequence of Lemma 16.104, we have the following upper bound on the total domination number of a graph with small minimum degree in terms of its order.

Theorem 16.105 *Let H be a graph of order n and let $\delta \in \{1, 2, 3\}$. If $\delta(H) \geq \delta$, then $\gamma_t(H) \leq 2n/(\delta + 1)$, and this bound is sharp.*

Proof Let $V(H) = \{v_1, \dots, v_n\}$ and let G be the bipartite graph constructed from H as follows. Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_n\}$. For each edge $v_i v_j$ in H , add the two edges $x_i y_j$ and $x_j y_i$ in G . We note that G has partite sets X and Y . A TD-set A in H corresponds to a subset $A \subset X$ in G adjacent to every vertex in Y . The result now follows by Lemma 16.104. ■

As a special case of Theorem 16.105 when $\delta = 3$, we have the result of Theorem 16.103.

The result of Theorem 16.105 was, in fact, known, albeit clothed in hypergraph terminology, well before the elegant graph theoretic proof presented by Archdeacon *et al.* [3] in 2004. Chvátal and McDiarmid [15] and Tuza [117] independently established an important result (see Theorem 16.107) about transversals in hypergraphs that readily implies Theorem 16.105. The idea of using transversals in hypergraphs to obtain results on total domination in graphs, however, first appeared in a paper by Thomassé and Yeo [114] submitted in 2003 but only published in 2007. Up to that time, the transition from total domination in graphs to transversals in hypergraphs seemed to pass by unnoticed. Subsequent to the Thomassé-Yeo paper, several results on total domination in graphs have been obtained using transversals in hypergraphs that appear very difficult to prove using graph theoretic techniques. For example, the proofs of Theorems 16.101 and 16.102 stated earlier demonstrate an interplay between graph theory and transversals in hypergraphs. We therefore take a brief hypergraph excursion before returning to the interplay between total domination in graphs and transversals in hypergraphs.

A hypergraph excursion. Hypergraphs are systems of sets which are conceived as natural extensions of graphs. More precisely, a **hypergraph** $H = (V, E)$ is a finite set V of elements, called **vertices**, together with a finite multiset E of arbitrary subsets of V , called **edges**. A k -uniform hypergraph is a hypergraph in which every edge has size k . Every (simple) graph is therefore a 2-uniform hypergraph, and so graphs are special hypergraphs.

A **transversal** (or **edge cover** or **hitting set** in the computer science literature) in H is a subset of the vertices of H which has a nonempty intersection with each edge of H . The **transversal number** $\tau(H)$ of H is the minimum cardinality of a transversal in H . A transversal in H of cardinality $\tau(H)$ is called a **$\tau(H)$ -transversal**.

For a graph $G = (V, E)$, we denote by H_G the **open neighbourhood hypergraph**, abbreviated ONH, of G ; that is, $H_G = (V, C)$ is the hypergraph with vertex set V and with edge set C consisting of the open neighbourhoods of vertices of V

in G . Every TD-set in G contains a vertex from the open neighbourhood of each vertex in G , and is therefore a transversal in H_G . In particular, if S is a γ_t -set of G , then S is a transversal in H_G , and so $\tau(H_G) \leq |S| = \gamma_t(G)$. Conversely, every transversal in H_G contains a vertex from the open neighbourhood of each vertex of G , and is therefore a TD-set in G . In particular, if T is a $\tau(H_G)$ -transversal, then T is a TD-set in G , and so $\gamma_t(G) \leq |T| = \tau(H_G)$. Consequently, $\gamma_t(G) = \tau(H_G)$. The transversal number of the ONH of a graph is therefore precisely the total domination number of the graph. We state this result, first observed by Thomassé and Yeo [114], in the following observation.

Observation 16.106 *If G is a graph with no isolated vertex and H_G is the ONH of G , then $\gamma_t(G) = \tau(H_G)$.*

As remarked in [69], “perhaps much of the recent interest in total domination in graphs arises from the fact that total domination in graphs can be translated to the problem of finding transversals in hypergraphs. The main advantage of considering hypergraphs rather than graphs is that the structure is easier to handle — for example, we can often restrict our attention to uniform hypergraphs where every edge has the same size.”

Chvátal and McDiarmid [15] and Tuza [117] independently established the following result about transversals in hypergraphs. A proof of this result can also be found in [69, Theorem 5.8].

Theorem 16.107 *If $H = (V, E)$ is a hypergraph in which all edges have size at least three, then $4\tau(H) \leq |V| + |E|$.*

Using Theorem 16.107, the result of Theorem 16.103 follows readily.

Proof of Theorem 16.103 (using hypergraphs) Let G be a graph of order n with minimum degree at least three, and let H_G be the ONH of G . Then, each edge of H_G has size at least 3. Furthermore, H has n vertices and n edges. By Theorem 16.107, there exists a transversal in H of size at most $(n + n)/4 = \frac{n}{2}$. Hence, $\gamma_t(G) = \tau(H_G) \leq \tau(H) \leq \frac{n}{2}$. ■

Two infinite families $\mathcal{K}_{\text{cubic}}$ and $\mathcal{L}_{\text{cubic}}$ of connected cubic graphs (described below) with total domination number one-half their orders were constructed in [35], which shows that the bound of Theorem 16.103 is sharp. For $k \geq 2$, consider two copies of the path P_{2k} with respective vertex sequences $a_1, b_1, a_2, b_2, \dots, a_k, b_k$ and $c_1, d_1, c_2, d_2, \dots, c_k, d_k$. For each $i \in [k]$, join a_i to d_i and b_i to c_i . To complete the construction of graphs in $\mathcal{K}_{\text{cubic}}$ ($\mathcal{L}_{\text{cubic}}$, respectively), join a_1 to c_1 and b_k to d_k (a_1 to b_k and c_1 to d_k , respectively). Two graphs $G_{16.41}$ and $G_{16.42}$ in the families $\mathcal{K}_{\text{cubic}}$ and $\mathcal{L}_{\text{cubic}}$ are illustrated in Figure 16.24.

The generalised Petersen graph of order 16 (in Figure 16.25) also achieves equality in Theorem 16.103. Hence we have two infinite families of connected cubic graphs that achieve equality in the upper bound of Theorem 16.103, as well as one connected cubic graph of order 16. But are there any other extremal connected graphs? Henning and Yeo [67] showed that there are no other extremal connected graphs. To do this, they first characterised connected hypergraphs that achieve equality in the upper bound of Theorem 16.107. Then, using the interplay between total domination in graphs and transversals in hypergraphs, they characterised connected graphs with minimum degree at least three that achieve equality in the bound of Theorem 16.103.

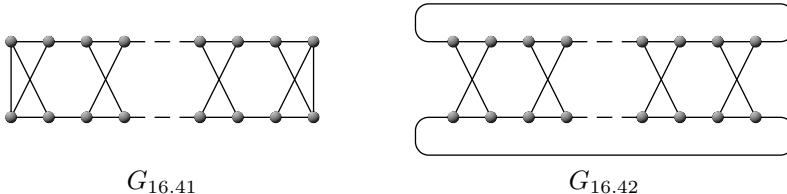


Figure 16.24: Cubic graphs $G_{16.41} \in \mathcal{K}_{\text{cubic}}$ and $G_{16.42} \in \mathcal{L}_{\text{cubic}}$.

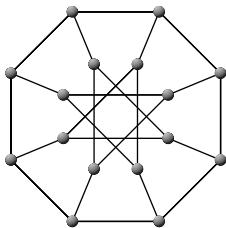


Figure 16.25: A generalised Petersen graph $G_{16.43}$ of order 16.

Theorem 16.108 *If G is a connected graph with minimum degree at least three and total domination number one-half its order, then $G \in \mathcal{K}_{\text{cubic}} \cup \mathcal{L}_{\text{cubic}}$ or $G = G_{16.43}$ is the generalised Petersen graph of order 16, shown in Figure 16.25.*

16.5.5 Bounds for graphs with minimum degree at least four

Here we consider the case where the minimum degree is at least four. Thomassé and Yeo [114] proved the following beautiful hypergraph result.

Theorem 16.109 *Every 4-uniform hypergraph of order n and size m has a transversal with no more than $(5n + 4m)/21$ vertices.*

Stéphan Thomassé was born in France on 28 December 1968. He obtained a doctorate in mathematics at the Université Claude Bernard Lyon 1 in Lyon, France in 1995 where he accepted a position as a mathematics assistant professor in 1997. In 2006, he became a full professor in the Department of Computer Science at the Université Montpellier 2. He is currently a full professor in the Department of Computer Science at the École Normale Supérieure (ENS) de Lyon which is part of the Université de Lyon. He has made significant contributions in many areas of graph theory and digraph theory.



Biographic note 85: Stéphan Thomassé (1968–present)

By the **complement** \bar{H} of a hypergraph H we mean the hypergraph with the same vertex set as H and where e is an edge of H if and only if $V(H) \setminus e$ is an edge of \bar{H} . Consider, for example, the complement \bar{F}_7 of the Fano plane F_7 (shown

in Figure 16.26). The Fano plane has the nice property that every two distinct vertices belong to exactly one common edge and every two distinct edges intersect in exactly one common vertex. Let $\{u, v\}$ be any two distinct vertices in \bar{F}_7 , and let e be the (unique) edge of F_7 that contains both u and v . Then, $V(F_7) \setminus e$ is an edge of \bar{F}_7 that is not covered by $\{u, v\}$, implying that no two vertices in \bar{F}_7 form a transversal in \bar{F}_7 . Hence, $\tau(\bar{F}_7) \geq 3$. Any set of three vertices that do not belong to a common edge in F_7 , however, form a transversal in \bar{F}_7 , and so $\tau(\bar{F}_7) \leq 3$. Consequently, $\tau(\bar{F}_7) = 3$. Since \bar{F}_7 is a 4-uniform hypergraph of order $n = 7$ and size $m = 7$, we observe that the complement of the Fano plane achieves equality in Theorem 16.109.

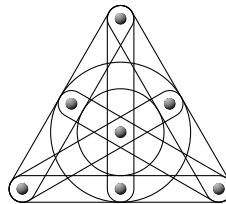


Figure 16.26: The Fano-plane F_7 .

As an immediate consequence of Theorem 16.109, we have the following upper bound on the total domination number of a graph with minimum degree at least four in terms of its order, due to Thomassé and Yeo [114].

Theorem 16.110 *If G is a graph of order n with minimum degree at least four, then $\gamma_t(G) \leq \frac{3}{7}n$.*

Proof Let H_G be the ONH of G . Then each edge of H_G has size at least 4. Let H be obtained from H_G by shrinking all edges of H_G , if necessary, to edges of size 4. Then, H is a 4-uniform hypergraph with n vertices and n edges. By Theorem 16.109, there exists a transversal in H of size at most $(5n + 4m)/21 = 9n/21$. Hence, $\gamma_t(G) = \tau(H_G) \leq \tau(H) \leq \frac{3}{7}n$. ■

The **incidence bipartite graph** $G_H(X, Y)$ of a hypergraph H is the bipartite graph with partite sets $X = V(H)$ and $Y = E(H)$, and where there is an edge between $x \in X$ and $y \in Y$ if and only if x belongs to the hyperedge y in H . Furthermore, the **bipartite complement** of a bipartite graph G with partite sets X and Y is the bipartite graph with the same vertex set and partite sets as G and where there is an edge between $x \in X$ and $y \in Y$ in the relative complement of G if and only if there is no edge between $x \in X$ and $y \in Y$ in G . The incidence bipartite graph of the complement of the Fano plane (or, equivalently, the relative complement of the Heawood graph) shown in Figure 16.27 achieves equality in the bound of Theorem 16.110.

Yeo [128] showed that the incidence bipartite graph of the complement of the Fano plane is, in fact, the unique connected graph achieving equality in the bound of Theorem 16.110.

Theorem 16.111 *If G is a connected graph of order n with minimum degree at least four, then $\gamma_t(G) \leq \frac{3}{7}n$, with equality if and only if G is the bipartite complement of the Heawood graph.*

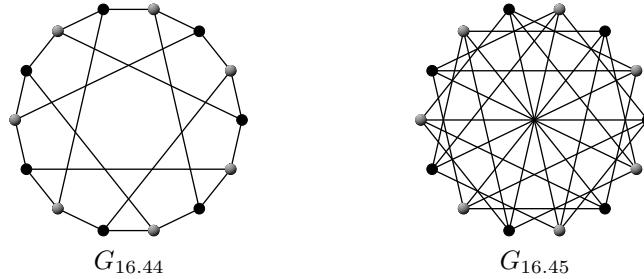


Figure 16.27: The Heawood graph $G_{16.44}$ and its bipartite complement $G_{16.45}$.

16.5.6 A heuristic bound

The decision problem underlying computation of the total domination number takes the following form: Given a graph $G = (V, E)$ and a positive integer k , does G have a TD-set of cardinality at most k ? We denote this decision problem by $D_{\text{tdomset}}(G, k)$.

$D_{\text{tdomset}}(G, k)$ is NP-complete for various graph classes, including (among others) bipartite graphs [93], chordal graphs [81], and claw-free graphs [87]. We therefore turn our attention toward developing an efficient approximation algorithm that can quickly find a TD-set whose cardinality is “close” to the cardinality of a minimum TD-set.

One such heuristic is the following greedy approach towards finding a TD-set in a general graph G with minimum degree $\delta \geq 2$ and order n . Let H_G be the ONH of G . Then each edge of H_G has size at least δ . Let H be obtained from H_G by shrinking all edges of H_G , if necessary, to edges of size δ . Then H is a δ -uniform hypergraph with n vertices and n edges. We construct a set T of vertices of H as follows. Select a vertex v of maximum degree in H , delete v and all edges incident with v from H , and let $T = \{v\}$. Note that the resulting hypergraph $H - v$ is a δ -uniform hypergraph with at most n vertices. In this hypergraph we select a vertex of maximum degree, delete that vertex and all edges incident with it from this hypergraph, and then add the deleted vertex to the set T . We continue this process until there are no edges left. By construction, the resulting set T hits every edge of H and is therefore a transversal of H . It was shown in [66] that $|T| \leq (1 + \ln \delta)n/\delta$. Every transversal of H is a transversal of H_G , and every transversal of H_G is a TD-set of G . Hence the set T produced by this greedy algorithm is a TD-set of G of cardinality at most $(1 + \ln \delta)n/\delta$. The time complexity of this greedy algorithm, which we call the *greedy total domination algorithm*, was shown in [66] to be $\mathcal{O}(n + \delta n)$. Hence we have the following result.

Theorem 16.112 *If G is a graph of order n with minimum degree $\delta \geq 2$, then the greedy total domination algorithm produces a TD-set T in G satisfying*

$$|T| \leq \left(\frac{1 + \ln \delta}{\delta} \right) n.$$

The time complexity of the algorithm is $\mathcal{O}(n + \delta n)$.

We remark that if the minimum degree δ is large, then the upper bound of $(1 + \ln \delta)n/\delta$ on the total domination number can also be established using probabilistic methods. For sufficiently large δ , the TD-set T produced by the greedy total

domination algorithm is essentially optimal, as can be deduced from the following result due to Thomassé and Yeo [114].

Theorem 16.113 *For any $\epsilon > 0$ and for sufficiently large δ , there exists a δ -uniform hypergraph H with n vertices and n edges satisfying*

$$\tau(H) > \left(\frac{(1 - \epsilon) \ln \delta}{\delta} \right) n.$$

16.5.7 Summary of bounds

Known upper bounds on the total domination number of a graph G in terms of its order n are summarised in [Table 16.1](#).

Condition(s)	Bound	Reference(s)
$\delta(G) \geq 1; n \geq 3; G$ is connected	$\gamma_t(G) \leq \frac{2}{3}n$	[19]
$\delta(G) \geq 2; n \geq 11; G$ is connected	$\gamma_t(G) \leq \frac{4}{7}n$	[61]
$\delta(G) \geq 3$	$\gamma_t(G) \leq \frac{n}{2}$	[3, 15, 117]
$\delta(G) \geq 4$	$\gamma_t(G) \leq \frac{3}{7}n$	[114]
$\delta(G) \geq 2$	$\gamma_t(G) \leq \left(\frac{1 + \ln \delta}{\delta} \right) n$	[66]

Table 16.1: Upper bounds on the total domination number of a graph G of order n .

16.5.8 The upper total domination number

The **upper total domination number** of a graph G , denoted by $\Gamma_t(G)$, is the maximum cardinality of a minimal TD-set in G . We call a minimal TD-set of cardinality $\Gamma_t(G)$ a **Γ_t -set of G** . We observe that if G has maximum degree greater than 1, then G necessarily contains a vertex, v say, such that no neighbour of v has degree 1. Furthermore, $V(G) \setminus \{v\}$ is a TD-set in G , implying that the set $V(G)$ is not a minimal dominating set. Hence we have the following observation.

Observation 16.114 *If G is a graph of order n with maximum degree greater than 1, then $\Gamma_t(G) \leq n - 1$.*

The trivial upper bound in [Observation 16.114](#) is sharp, as may be seen by taking G to be the graph obtained by subdividing every edge in the star $K_{1,(n-1)/2}$ exactly once, where n is odd and $n \geq 5$. The set of $n - 1$ leaves and support vertices form a minimal TD-set in G , and so $\Gamma_t(G) \geq n - 1$. Hence, by [Observation 16.114](#), $\Gamma_t(G) = n - 1$. If, however, we impose a regularity condition on the graph, then using edge weighting functions on dominating sets, as defined in [Section 16.2.7](#), it is possible to show that this upper bound can be greatly improved.

Theorem 16.115 ([108]) *For every k -regular graph G of order n with no isolated vertex, $\Gamma_t(G) \leq n/(2 - \frac{1}{k})$.*

Proof Let G be a k -regular graph of order n without isolated vertices for some $k \geq 1$ and let T be a Γ_t -set of G . We use the edge weight function ψ_T and vertex weight function ϕ_T to count the number of vertices in T relative to n . Recall, from Section 16.2.7, that if $e \in [T, V \setminus T]$, then $\psi_T(e) = 1/d_T(v)$, and so $\frac{1}{k} \leq \psi_T(e) \leq 1$.

We show that, on average, $\psi_T(v) \geq 1 - \frac{1}{k}$ for each vertex $v \in T$. Let $A = \{v \in V \mid \text{ipn}(v, T) \neq \emptyset\}$ and let $B = T \setminus A$. By Proposition 16.92, $\text{epn}(v, T) \neq \emptyset$ or $\text{ipn}(v, T) \neq \emptyset$ for each vertex $v \in T$, and so for each $v \in B$, we have $\text{epn}(v, T) \neq \emptyset$. For $X \in \{A, B\}$, let $X_1 = \{v \in X \mid v \in \text{ipn}(u, T) \text{ for some } u \in T\}$ and let $X_2 = X \setminus X_1$. We remark that A_1, A_2, B_1 and B_2 are pairwise disjoint and that $T = A_1 \cup A_2 \cup B_1 \cup B_2$. We consider the weight assigned by the function ϕ_T to vertices from each of these sets in turn.

If $v \in A_1$, then $v \in \text{ipn}(u, T)$ for some $u \in T$. Hence, v has exactly $k - 1$ neighbours in $V \setminus T$ and so $\phi_T(v) \geq (k - 1)\frac{1}{k} = 1 - \frac{1}{k}$. If $v \in A_2$, then possibly v has no neighbours in $V \setminus T$ and we have $\phi_T(v) \geq 0$. If $v \in B_1$, then $v \in \text{ipn}(u, T)$ for some $u \in T$ and hence v has exactly $k - 1$ neighbours in $V \setminus T$. Furthermore, $\text{epn}(v, T) \neq \emptyset$. Therefore, under the function ψ_T , at least one edge joining v to $V \setminus T$ is assigned weight 1 and each of the remaining $k - 2$ edges joining v to $V \setminus T$ is assigned weight at least $\frac{1}{k}$. Thus, $\phi_T(v) \geq 1 + (k - 2)\frac{1}{k} = 2(1 - \frac{1}{k})$. Finally, if $v \in B_2$, then $\text{epn}(v, T) \neq \emptyset$ and so at least one edge incident with v has weight 1. As a result, $\phi_T(v) \geq 1 > 1 - \frac{1}{k}$. Summing the weights over all vertices in T we therefore obtain the inequality

$$\xi(T) = \sum_{v \in T} \phi_T(v) \geq \left(1 - \frac{1}{k}\right)(|A_1| + 2|B_1| + |B_2|). \quad (16.12)$$

We now show that $|B_1| \geq |A_2|$. Let $t = |A_2|$. If $t = 0$, the result is immediate. Hence, we may assume that $t \geq 1$. Let $A_2 = \{a_1, \dots, a_t\}$. For $i \in [t]$ we remark that $\text{ipn}(a_i, T) \neq \emptyset$ and we let $b_i \in \text{ipn}(a_i, T)$. Since a_i is the unique neighbour of b_i in T , we have $b_i \neq b_j$ for $i \neq j$. Let $i \in [t]$. If $b_i \in A$, then $\text{ipn}(b_i, T) \neq \emptyset$, and so necessarily $a_i \in \text{ipn}(b_i, T)$, contradicting the fact that $a_i \in A_2$. Hence, $b_i \in B$ and since $b_i \in \text{ipn}(a_i, T)$ we have that $b_i \in B_1$. Thus, $\{b_1, \dots, b_t\} \subseteq B_1$, and so $|B_2| \geq t = |A_2|$, as desired. Therefore, $|A_1| + 2|B_1| + |B_2| \geq |A_1| + |A_2| + |B_1| + |B_2| = |T|$ and so, from inequality (16.12), it follows that

$$\xi(T) \geq \left(1 - \frac{1}{k}\right)|T|. \quad (16.13)$$

By equation (16.8), $n - |T| = \xi(T)$, and so by inequality (16.13) it holds that $n - |T| \geq \left(1 - \frac{1}{k}\right)|T|$. Thus, $\Gamma_t(G) = |T| \leq n/(2 - \frac{1}{k})$. ■

We remark that the upper total domination number and the upper domination number of a graph with no isolated vertex are incomparable. More specifically, for each positive integer k , there exist graphs G and H such that $\Gamma(G) - \Gamma_t(G) = k$ and $\Gamma_t(H) - \Gamma(H) = k$. Dorbec *et al.* [30], however, established the following relationship.

Theorem 16.116 *For every graph G of order n with no isolated vertex,*

$$\left(\frac{2}{n-1}\right)\Gamma(G) \leq \Gamma_t(G) \leq 2\Gamma(G).$$

In order to prove Theorem 16.116, we require the following key lemma. Recall that for a graph $G = (V, E)$ and a subset $S \subseteq V$, $d_S(v)$ denotes the number of vertices in S that are adjacent to v .

Lemma 16.117 ([30]) *Every Γ_t -set of G contains as a subset a minimal dominating set S such that $|S| \geq \frac{1}{2}\Gamma_t(G)$ and $\text{epn}(v, S) \geq 1$ for each $v \in S$.*

Proof Let D be a Γ_t -set of G and let

$$\begin{aligned} A &= \{v \in D \mid \text{epn}(v, D) \geq 1\}, \\ B &= \{v \in D \setminus A \mid d_A(v) \geq 1\}, \quad \text{and} \\ C &= D \setminus (A \cup B). \end{aligned}$$

Then, $D = A \cup B \cup C$. Let $v \in B \cup C$. Then, $\text{epn}(v, D) = \emptyset$, and so by Proposition 16.92, $|\text{ipn}(v, D)| \geq 1$. Let $v' \in \text{ipn}(v, D)$, and so $v' \in D$ and $N(v') \cap D = \{v\}$. Thus, $d_D(v') = 1$. Since v' is adjacent only to v and $v \notin A$, we have that $v' \notin B$.

Suppose $v \in C$. Then $v' \notin A$ since $d_A(v) = 0$, and so $v' \in C$. Hence, $\text{epn}(v', D) = \emptyset$, and so, by Proposition 16.92, $|\text{ipn}(v', D)| \geq 1$. The vertex v is, however, the only neighbour of v' in $G[D]$, implying that $\text{ipn}(v', D) = \{v\}$; that is, $N(v) \cap D = \{v'\}$, and so $d_D(v) = 1$. Hence, if $C \neq \emptyset$, then $G[C] = \frac{|C|}{2}K_2$ and for each $v \in C$, $d_D(v) = 1$. We call two adjacent vertices in $G[C]$ *partners* in C . Let (X, Y) be partite sets in the graph $G[C]$, and so each vertex in X (resp., in Y) is adjacent in $G[D]$ only to its partner in Y (resp., in X). For each $x \in X$, let y_x be the partner of x in C .

Suppose $v \in B$. Since $G[C] = \frac{|C|}{2}K_2$ and each vertex of C has degree 1 in $G[D]$, the vertex v is not adjacent to any vertex of C . In particular, $v' \in A \cup B$. As shown earlier, $v' \notin B$. Hence, $v' \in A$. Thus, for each $v \in B$, $\text{pn}(v, D) \subseteq A$. This, in turn, implies that $|A| \geq \sum_{v \in B} |\text{ipn}(v, D)| \geq |B|$.

Let $U = V(G) \setminus (D \cup N(A) \cup N(X))$ be the set of vertices in $V(G) \setminus D$ not dominated by A or X in G . Since D is a TD-set of G , the set U is dominated by $B \cup Y$. Let B_Y be a minimum subset of $B \cup Y$ that dominates U . Thus, for each $v \in B_Y$, it follows that $|\text{epn}(v, B_Y) \cap U| \geq 1$.

We now consider the set $S = A \cup B_Y \cup X$. Since $B \subseteq N(A)$ and $Y \subseteq N(X)$, the set S dominates D . By construction, the set S also dominates $V(G) \setminus D$. Thus, S dominates $V(G)$, but S is not necessarily a minimal dominating set of G . We now construct a minimal dominating set of G from S as follows. We consider the vertices in X in turn, and for each vertex $x \in X$ we systematically delete x from S if $\text{epn}(x, S) = \emptyset$ at each stage in the resulting set S . (Observe that if the partner $y_x \in Y$ of x in C is not in S , then $y_x \in \text{epn}(x, S)$, and so x is not deleted from S .) Let X^* be the resulting subset of vertices of X that belong to the set S upon the completion of this process. Thus, $S = A \cup B_Y \cup X^*$ and, by construction, $|\text{epn}(v, S)| \geq 1$ for each $v \in S$. If $x \in X \setminus X^*$, then the partner of x in C is in the set S , implying that S dominates C . Since $B \subseteq N(A)$, the set B is dominated by S . Therefore, the set S dominates D . By construction, the set S also dominates $V(G) \setminus D$. Hence, S is a minimal dominating set of G .

It remains to show that $|S| \geq \frac{1}{2}\Gamma_t(G)$. For every vertex $x \in X$, the set S contains at least one of x and its partner in C , whence $|S \cap C| \geq |X| = \frac{1}{2}|C|$. As shown earlier, $|A| \geq |B|$, and so,

$$|S| \geq |A| + |S \cap C| \geq |A| + \frac{1}{2}|C| \geq \frac{1}{2}(|A| + |B| + |C|) = \frac{1}{2}|D| = \frac{1}{2}\Gamma_t(G). \quad \blacksquare$$

Theorem 16.116 now follows readily from Lemma 16.117.

Proof of Theorem 16.116 Let D be a Γ_t -set of G . By Lemma 16.117, there is a minimal dominating set S of G such that $S \subseteq D$ and $|S| \geq \frac{1}{2}\Gamma_t(G)$. Hence, $\Gamma(G) \geq |S| \geq \frac{1}{2}\Gamma_t(G)$. This establishes the upper bound. The lower bound follows from the observations that $\Gamma(G) \leq n - 1$ (with equality if and only if G is the star $K_{1,n-1}$) and $\Gamma_t(G) \geq 2$. ■

16.5.9 Total domination edge critical graphs

In this section, we briefly consider the notion of total domination edge critical graphs. A graph G is **total domination edge critical** if $\gamma_t(G + e) < \gamma_t(G)$ for every edge $e \in E(\bar{G}) \neq \emptyset$. Furthermore, if $\gamma_t(G) = k$, then we say that G is a **k_t -critical graph**. Thus if G is k_t -critical, then its total domination number is k and the addition of any edge decreases the total domination number. In particular, we note that, by definition, the complete graph on at least two vertices is not a 2_t -critical graph. Furthermore, since $\gamma_t(G) \geq 2$ for every graph G with no isolated vertex, we note that if G is a k_t -critical graph, then $k \geq 3$. The study of total domination edge critical graphs was initiated by Van der Merwe [118]. It was shown in [119] that the addition of an edge to a graph can change the total domination number by at most two. A proof of this observation is left as an exercise (see Exercise 16.12).

Total domination edge critical graphs G with the property that $\gamma_t(G) = k$ and $\gamma_t(G + e) = k - 2$ for every edge $e \in E(\bar{G})$ are called **k_t -supercritical graphs**. Thus, if G is k_t -supercritical, then its total domination number is k and the addition of any edge decreases the total domination number by two. The following result characterises k_t -supercritical graphs. Recall that a *clique* is a complete subgraph and a *nontrivial clique* is a clique on at least two vertices.

Theorem 16.118 *A graph G is k_t -supercritical if and only if $k \geq 4$ is even and G is the disjoint union of $\frac{k}{2}$ nontrivial cliques.*

Proof Let G be a k_t -supercritical graph. Then, $\gamma_t(G) = k$ and $\gamma_t(G + e) = k - 2$ for every edge $e \in E(\bar{G})$. Since $\gamma_t(F) \geq 2$ for every graph F with no isolated vertex, we note, in particular, that $\gamma_t(G + e) \geq 2$, implying that $k \geq 4$. We first show that G is P_3 -free. For the sake of contradiction, let xyz be a path in G where x and z are not adjacent in G , and let $S_{x,z}$ be a γ_t -set of $G + xz$. Since G is k_t -supercritical, we have that $|S_{x,z}| = \gamma_t(G + xz) = k - 2$. The set $S_{x,z} \cup \{y\}$ is, however, a TD-set in G of cardinality less than k , a contradiction. Therefore, G is P_3 -free. The only graphs that are P_3 -free are those in which every component is a clique. Since G has no isolated vertex, every component of G is therefore a nontrivial clique. Furthermore, since $\gamma_t(G) = k$ and every γ_t -set of G contains two vertices from each (clique) component of G , the graph G is the disjoint union of $\frac{k}{2}$ nontrivial cliques. Conversely, if $k \geq 4$ is even and G is the disjoint union of $\frac{k}{2}$ nontrivial complete graphs, then $\gamma_t(G) = k$ and $\gamma_t(G + e) = k - 2$ for every edge $e \in E(\bar{G})$, implying that G is k_t -supercritical. ■

In contrast to k_t -supercritical graphs, a characterisation of k_t -critical graphs in general is much more difficult and not yet known for any value of $k \geq 3$. In the special case of 3_t -critical graphs, some properties of these graphs have been established. In particular, the following result was established in [119].

Theorem 16.119 *If G is a 3_t -critical graph, then $2 \leq \text{diam}(G) \leq 3$.*

Proof Suppose, to the contrary, that $\text{diam}(G) \geq 4$. Let v_0 and v_4 be two vertices at distance 4 apart in G and consider the edge $e = v_0v_4 \in E(\bar{G})$. Since G is a 3_t -critical graph, $\gamma_t(G + e) = 2$. Let S be a γ_t -set of $G + e$. If neither v_0 nor v_4 belongs to S , then S is a TD-set of G , implying that $\gamma_t(G) = 2$, a contradiction. Hence, renaming v_0 and v_4 , if necessary, we may assume that $v_0 \in S$. Let v denote the remaining vertex in S , and so $S = \{v, v_0\}$. If $v = v_4$, then the vertex v_2 is not dominated by S in $G + e$. On the other hand, if $v \neq v_4$, then the vertex v is a neighbour of v_0 in G . But then the vertex v_3 is not dominated by S in $G + e$. In both cases, the set S is not a TD-set in $G + e$, a contradiction. Therefore, $\text{diam}(G) \leq 3$. Since a complete graph is not 3_t -critical, $\text{diam}(G) \geq 2$. ■

- ❖ The reader should now be able to attempt Exercises 16.10–16.12 and Projects 16.3–16.6.

Exercises

- 16.1 List all the minimal dominating sets of the graph $G_{16.1}$ in Figure 16.2.
- 16.2 Prove Corollary 16.7 using Corollary 16.2 or Theorem 16.3.
- 16.3 Show that $\gamma(P_n) = \gamma(C_n) = \lceil \frac{n}{3} \rceil$ for all $n \geq 3$.
- 16.4 Show that if G is a disconnected graph, then $\gamma(\bar{G}) \leq 2$.
- 16.5 Determine a sharp lower bound on the domination number of a connected graph in terms of its diameter.
- 16.6 Define the **boundary** of a set S to be the set $B(S) = \{v \mid |N[v] \cap S| = 1\}$, that is, $B(S)$ is the set of vertices dominated by exactly one vertex in S . Prove that a dominating set S is a minimal dominating set if and only if $B(S)$ dominates S .
- 16.7 Prove that if a connected graph G of order n does not contain either P_4 or C_4 as an induced subgraph, then $\gamma(G) = 1$, that is, $\Delta(G) = n - 1$.
- 16.8 Using Lemma 16.4 and Corollary 16.14, show that if G is a graph of order n such that neither G nor \bar{G} has an isolated vertex, then $\gamma(G) + \gamma(\bar{G}) \leq (n + 4)/2$. (This was first proved by Joseph and Arumugam [72].)
- 16.9 A set S in a graph G is a **packing** of G if the vertices in S are pairwise at distance at least 3 apart in G , i.e. if $u, v \in S$, then $d_G(u, v) \geq 3$. Equivalently, S is a packing if the closed neighbourhoods of vertices in S are pairwise disjoint. The **packing number** of a graph G , denoted by $\rho(G)$, is the maximum cardinality of a packing of G . Prove that
 - (a) $\rho(G) \leq \gamma(G)$ for any graph G ;
 - (b) if T is a tree of order at least three, then there is a γ -set of T that contains no leaf of T ;
 - (c) if T is a tree of order at least three, then there is a maximum packing of T that contains one leaf-neighbour of every support vertex; and
 - (d) $\rho(T) = \gamma(T)$ for any tree T .

16.10 A set S in a graph G is an **open packing** of G if the open neighbourhoods of vertices in S are pairwise disjoint in G . The **open packing number** of G , denoted by $\rho^0(G)$, is the maximum cardinality of an open packing of G . Prove that

- (a) $\rho^0(G) \leq \gamma_t(G)$ for any graph G ;
- (b) if T is not a star, there is a γ_t -set of T that contains no leaf of T ;
- (c) if T is not a star, there is a maximum open packing of T that contains one leaf neighbour of every support vertex; and
- (d) $\rho^0(T) = \gamma_t(T)$ for any tree T .

16.11 Design a linear algorithm for computing the total domination number of a tree and prove the correctness of your algorithm.

16.12 Prove that the addition of an edge to a graph can change the total domination number by at most two.

Computer exercises

The **MATHEMATICA** command `AdjacencyList[G, S]` returns a list of vertices adjacent in the graph G to vertices in the set S , while the command `Complement[S, T]` returns the set of all vertices in the set S that are not in the set T . Moreover, the command `Join[A, B]` concatenates the two lists A and B , while the command `Union[A, B]` returns a sorted list of all the distinct elements that appear in the lists A and/or B . Finally, the command `Select[A, crit]` picks out all elements $a \in A$ for which `crit[a]` evaluates to `True`, while the commands `Length[S]` and `VertexCount[G]` return respectively the number of elements in the list S and the order of the graph G .

Using the commands above it is possible to define a function `ClosedNhood[G, S]` which returns the union of a vertex subset S of a graph G and all the vertices adjacent to vertices in S , as follows:

```
In[1]:= ClosedNhood[G_, S_] := Union[AdjacencyList[G, S], S]
```

We can then define a function `PrivateNeighbours[G, S, v]` as follows which returns the set of private neighbours of the vertex v within a vertex subset S of the graph G :

```
In[2]:= PrivateNeighbours[G_, S_, v_] :=
  Complement[
    ClosedNhood[G, S],
    ClosedNhood[G, Complement[S, {v}]]]
```

The native functions `DominatingSetQ[G, S]`, `TotalDominatingSetQ[G, S]` and `IrredundantSetQ[G, S]` below return the boolean value `True` if the vertex subset S is respectively a dominating set, a total dominating set, or an irredundant set of the graph G , or `False` otherwise:

```
In[3]:= DominatingSetQ[G_, S_] :=
  Length[ClosedNhood[G, S]] == VertexCount[G]
In[4]:= TotalDominatingSetQ[G_, S_] :=
  Length[AdjacencyList[G, S]] == VertexCount[G]
```

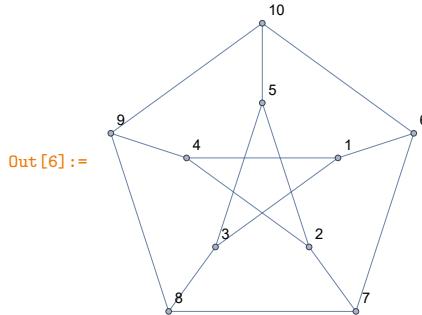
```
In[5]:= IrredundantSetQ[G_, S_] :=
Length[
Select[S, PrivateNeighbours[G, S, #] != {} &]
] == Length[S]
```

There is also a built-in function `IndependentVertexSetQ[G, S]` in **MATHEMATICA**, which returns the boolean value `True` if the vertex subset S is an independent set in the graph G .

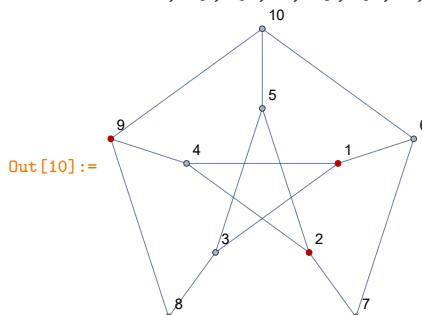
The functions above may be used to compute dominating sets, total dominating sets and irredundant sets of a given cardinality for a given graph. For example, the commands

```
In[6]:= P = PetersenGraph[VertexLabels -> "Name"]
In[7]:= ss3 = Subsets[VertexList[P], {3}];
In[8]:= ss4 = Subsets[VertexList[P], {4}];
In[9]:= domsets = Select[ss3, DominatingSetQ[P, #] &]
In[10]:= HighlightGraph[P, domsets[[1]]]
In[11]:= totdomsets = Select[ss4, TotalDominatingSetQ[P, #] &]
In[12]:= HighlightGraph[P, totdomsets[[1]]]
```

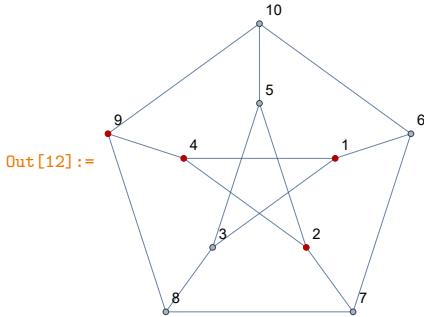
display the Petersen graph P , return a list of all dominating sets of cardinality 3 in P (highlighting the first of these sets in a graphical representation of P), and return a list of all total dominating sets of cardinality 4 in P (highlighting the first of these sets in a graphical representation of P), as follows:



```
Out[9]:= {{1, 2, 9}, {1, 5, 8}, {1, 7, 10}, {2, 3, 10}, {2, 6, 8}, {3, 4, 6}, {3, 7, 9}, {4, 5, 7}, {4, 8, 10}, {5, 6, 9}}
```



```
Out[11]:= {{1, 2, 4, 9}, {1, 3, 4, 6}, {1, 3, 5, 8}, {1, 6, 7, 10}, {2, 3, 5, 10}, {2, 4, 5, 7}, {2, 6, 7, 8}, {3, 7, 8, 9}, {4, 8, 9, 10}, {5, 6, 9, 10}}
```



The functions `MinimalDominatingSetQ[G, S]` may furthermore be defined as follows, to return the boolean value `True` if the vertex subset S is a minimal dominating set of the graph G :

```
In[13]:= MinimalDominatingSetQ[G_, s_] :=
Module[{i},
  If[!DominatingSetQ[G, s], Return[False]];
  For[i = 1, i <= Length[s], i++,
    If[DominatingSetQ[G, Complement[s, {s[[i]]}]], Return[False]]];
  ];
  Return[True]
]
```

Similarly, `MaximalIrredundantSetQ[G, S]` returns the boolean value `True` if the vertex subset S is a maximal irredundant set of the graph G :

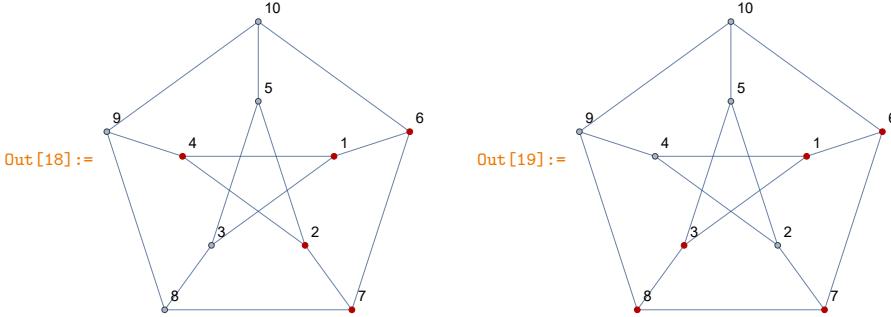
```
In[14]:= MaximalIrredundantSetQ[G_, S_] :=
Module[{i, R},
  If[!IrredundantSetQ[G, S], Return[False]];
  R = Complement[VertexList[G], S];
  For[i = 1, i <= Length[R], i++,
    If[IrredundantSetQ[G, Join[S, {R[[i]]}]], Return[False]]];
  ];
  Return[True]
]
```

The functions above may be used to compute minimal dominating sets and maximal irredundant sets of a given cardinality for a given graph. For example, the commands

```
In[15]:= ss5 = Subsets[VertexList[P], {5}];
In[16]:= mindomsets5 = Select[ss5, MinimalDominatingSetQ[P, #] &]
In[17]:= maxirsets5 = Select[ss5, MaximalIrredundantSetQ[P, #] &]
In[18]:= HighlightGraph[P, mindomsets5[[2]]]
In[19]:= HighlightGraph[P, maxirsets5[[5]]]
```

return lists of all minimal dominating sets and all maximal irredundant sets of cardinality 5 in the Petersen graph P (highlighting the second minimal dominating set and the fifth maximal irredundant set thus computed in a graphical representation of P), as follows:

```
Out[16]:= {{1, 2, 3, 4, 5}, {1, 2, 4, 6, 7}, {1, 3, 4, 8, 9}, {1, 3, 5, 6, 10}, {1,
 3, 6, 7, 8}, {1, 4, 6, 9, 10}, {2, 3, 5, 7, 8}, {2, 4, 5, 9, 10}, {2,
 4, 7, 8, 9}, {2, 5, 6, 7, 10}, {3, 5, 8, 9, 10}, {6, 7, 8, 9, 10}}
Out[17]:= {{1, 2, 3, 4, 5}, {1, 2, 4, 6, 7}, {1, 3, 4, 8, 9}, {1, 3, 5, 6, 10}, {1,
 3, 6, 7, 8}, {1, 4, 6, 9, 10}, {2, 3, 5, 7, 8}, {2, 4, 5, 9, 10}, {2,
 4, 7, 8, 9}, {2, 5, 6, 7, 10}, {3, 5, 8, 9, 10}, {6, 7, 8, 9, 10}}
```



It is also possible (and computationally cheaper!) to compute only a single dominating set of a specified cardinality for a given graph (if such a set exists), rather than computing all such sets. This may be achieved by invoking the following function, which returns a dominating set of cardinality n for the graph G if such a set exists (otherwise returning the empty set):

```
In[20]:= DominatingSet[G_, n_] :=
Module[{ss, i},
ss = Subsets[VertexList[G], {n}];
For[i = 1, i <= Length[ss], i++,
If[DominatingSetQ[G, ss[[i]]], Return[ss[[i]]]]
];
Return[{}];
]
```

The following function employs an interval-halving procedure to compute a dominating set of minimum cardinality (*i.e.* a γ -set) for a graph G :

```
In[21]:= gammaSet[G_] :=
Module[{lower, upper, m, ups, DEG, order, s},
DEG = Max[VertexDegree[G]];
order = VertexCount[G];
(* bounds from Theorem 16.13 *)
lower = Ceiling[order/(1 + DEG)];
upper = Min[Floor[order/2], order - DEG];
s = DominatingSet[G, lower];
If[Length[s] > 0,
upper = lower; ups = s, (* True *)
ups = DominatingSet[G, upper] (* False *)
];
While[upper - lower > 1,
m = Ceiling[(lower + upper)/2];
s = DominatingSet[G, m];
If[Length[s] > 0, upper = m; ups = s, lower = m];
];
ups
]
```

Applying the function above, the following γ -set is computed for the Petersen graph P , as follows:

```
In[22]:= gammaSet[P]
Out[22]:= {1, 2, 9}
```

Projects

This section contains six projects. The first project is devoted to an investigation of properties of lottery numbers, which are the domination numbers of certain graphs induced by combinatorial structures (as described in the introduction to this chapter), while in the second project the reader is guided in a step-by-step exploration aimed at establishing the domination numbers and independence numbers of the graphs induced by various chess pieces. Thereafter, applications of the notion of total domination are considered in the third project, and this is followed by an exploration in the fourth project of a variation on the theme of ordinary domination in graphs, called Roman domination. Two further variations on the classical notion of graph domination are finally considered in the fifth and sixth projects, namely distance domination and $\langle r, s \rangle$ -domination, respectively.

Project 16.1: The lottery problem

Consider a lottery scheme in which a player participates by selecting n -sets of numbers, called **tickets**, from a universal set $[m]$, for some integers m, n satisfying $n \leq m$. Suppose a winning ticket (also consisting of n numbers) is drawn randomly from the universal set, and that the player wins a prize, called a **k -prize**, if at least k numbers in at least one of his/her tickets match those in the winning ticket, for some integer $k \leq n$. Recall from the introduction of this chapter that a lottery of this form is denoted by the triple $\langle m, n; k \rangle$.

A number of combinations of lottery parameters m, n and k used world-wide in real national and state lotteries appear in [Table 16.2](#). Suppose we are interested in finding the smallest number of lottery tickets that a participant in a lottery of the form $\langle m, n; k \rangle$ must select in order to be guaranteed a k -prize. Denote the answer to this question by the **lottery number** $L(m, n; k)$. Consider, as an example, the unrealistically small lottery $\langle 5, 3; 2 \rangle$. In order to be guaranteed a 2-prize in this lottery, a player may buy the two tickets $\{1, 2, 3\}$ and $\{2, 4, 5\}$.

Tasks

1. Verify that each of the $\binom{5}{3} = 10$ possible winning tickets in the lottery $\langle 5, 3; 2 \rangle$ share at least two numbers with at least one of the tickets in the above ticket playing set $\{\{1, 2, 3\}, \{2, 4, 5\}\}$, and hence that $L(5, 3; 2) \leq 2$.
2. Verify that there is no playing set comprising one ticket which can guarantee the player a 2-prize in the lottery $\langle 5, 3; 2 \rangle$, and hence that $L(5, 3; 2) > 1$.

From the above we deduce that $L(5, 3; 2) = 2$. Recall, furthermore, that the problem of determining the value of a lottery problem is known as the **lottery problem** and has been studied in the combinatorial literature since the 1960s.

As described in [Section 16.1](#), the lottery problem may be translated into the realm of graph theory by defining a so-called **lottery graph**, denoted by $G\langle m, n; k \rangle$ for a lottery of the form $\langle m, n; k \rangle$, in which the vertices represent all the tickets that may possibly be played in the lottery, and in which two vertices are adjacent if the corresponding tickets share at least k numbers. The lottery graph $G\langle 5, 3; 2 \rangle$ was presented as an example in [Figure 16.1](#), with the optimal ticket playing set $\{\{1, 2, 3\}, \{2, 4, 5\}\}$ indicated by highlighted vertices.

State or country	Lottery
Malaysia	$\langle 10, n; k \rangle$
Yugoslavia	$\langle 24, 12; k \rangle$
West Virginia	$\langle 25, 6; k \rangle$
Chile, Venezuela	$\langle 25, 15; k \rangle$
Illinois, Lithuania	$\langle 30, 5; k \rangle$
Iowa	$\langle 30, 6; k \rangle$
Chile	$\langle 30, 7; k \rangle$
Wisconsin	$\langle 31, 5; k \rangle$
Colorado, Kansas	$\langle 32, 5; k \rangle$
District of Columbia, Michigan	$\langle 33, 5; k \rangle$
New Mexico, Turkey, Virginia	$\langle 34, 5; k \rangle$
Norway	$\langle 34, 7; k \rangle$
Arizona, Connecticut, Latvia, Massachusetts, Slovak Republic, Slovenia, South Dakota	$\langle 35, 5; k \rangle$
Kansas	$\langle 35, 6; k \rangle$
Hungary, Sweden	$\langle 35, 7; k \rangle$
Florida, Indiana, Kazakhstan, Yugoslavia	$\langle 36, 5; k \rangle$
Maine, New Hampshire, Vermont, Wisconsin	$\langle 36, 6; k \rangle$
China, Denmark	$\langle 36, 7; k \rangle$
Montana, Ohio, Texas	$\langle 37, 5; k \rangle$
Jamaica	$\langle 37, 6; k \rangle$
Iceland, Nebraska	$\langle 38, 5; k \rangle$
Australia, Delaware	$\langle 38, 6; k \rangle$
California, Georgia, Iowa, Malta, Maryland, Montana, New York, Pennsylvania, South Dakota	$\langle 39, 5; k \rangle$
District of Columbia	$\langle 39, 6; k \rangle$
Croatia	$\langle 39, 7; k \rangle$
Czech Republic, New Jersey	$\langle 40, 5; k \rangle$
Ghana, Kazakhstan, Louisiana, New Zealand, Perú	$\langle 40, 6; k \rangle$
Arizona	$\langle 41, 6; k \rangle$
Minnesota	$\langle 42, 5; k \rangle$
Belgium, Colorado, Ireland, Maine, Malaysia, Massachusetts, New Hampshire, Philippines, Puerto Rico, Taiwan, Vermont	$\langle 42, 6; k \rangle$
Japan	$\langle 43, 6; k \rangle$
Missouri, Portugal, Texas, Uruguay	$\langle 44, 5; k \rangle$
Australia, Connecticut, Missouri	$\langle 44, 6; k \rangle$
Argentina, Australia, Austria, Croatia, Hungary, Israel, Netherlands, Perú, Philippines, Singapore, Switzerland, Ukraine, Yugoslavia	$\langle 45, 6; k \rangle$
California	$\langle 47, 5; k \rangle$
Hong Kong	$\langle 47, 6; k \rangle$
British Columbia, Québec, Western Canada	$\langle 47, 7; k \rangle$
Denmark, Finland, Indiana, Oregon	$\langle 48, 6; k \rangle$
Malaysia	$\langle 49, 4; k \rangle$
Alberta, British Columbia, Colorado, Connecticut, Delaware, France, Georgia, Germany, Greece, Idaho, Iowa, Kansas, Kentucky, Louisiana, Malaysia, Manitoba, Maryland, Massachusetts, Minnesota, Missouri, Montana, New Hampshire, New Jersey, New Mexico, Ohio, Philippines, Poland, Québec, Rhode Island, Slovak Republic, South Africa, South Dakota, Spain, Turkey, United Kingdom, Virginia, Washington, Western Canada, Wisconsin	$\langle 49, 6; k \rangle$
Georgia, Illinois, Maryland, Massachusetts, Michigan, New Jersey, Virginia	$\langle 50, 6; k \rangle$

Table 16.2: Lottery parameters m , n and k used in real national and state lotteries world-wide.

Tasks (continued)

3. What is the order of the lottery graph $G\langle m, n; k \rangle$?
4. Prove that the lottery graph $G\langle m, n; k \rangle$ is r -regular, where

$$r = \sum_{i=k}^{n-1} \binom{n}{i} \binom{m-n}{n-i}.$$

(Hint: Consider an arbitrary ticket of $\langle m, n; k \rangle$ and count the number of tickets that share exactly i numbers with this ticket, where i is any integer in the set $\{k, \dots, n-1\}$.)

5. What is the size of the lottery graph $G\langle m, n; k \rangle$?

With the exception of a few small classes of lottery numbers, no closed-form formula is known for $L(m, n; k)$, for general values of m , n and k .

Tasks (continued)

6. Prove that $L(m, m; k) = 1$ for all $k \in [m]$.
7. Prove that $L(m, n; n) = \binom{m}{n}$ for all $n \in [m]$.
8. Prove that $L(m, n; 1) = \lfloor \frac{m}{n} \rfloor$ for all $n \in [m]$.

All of the tasks listed above were from results contained within the doctoral thesis of [Gründlingh \[49\]](#).

The problem of computing the lottery number $L(m, n; k)$ is NP-hard for general values of m , n and k . Let us therefore consider the small lottery $\langle 7, 3; 2 \rangle$.

Werner R Gründlingh was born in Durban, South Africa on 3 July 1978. He grew up in the vineyard-rich town of Stellenbosch, close to Cape Town, where he also attended Stellenbosch University and obtained a bachelor's degree with majors in applied mathematics and computer science. He obtained a doctorate in applied mathematics in 2004 under the supervision of Jan van Vuuren. After briefly teaching at his home university, he emigrated to British Columbia, Canada in 2006 to further pursue a professional career at the University of Victoria. He currently works for the Government of British Columbia as an economic analyst where he is involved in research projects related to statistics, discrete mathematics, neural networks, time series analyses and forecasting. In his spare time he manages multiple apiaries while being an avid beekeeper.



Biographic note 86: Werner Gründlingh (1978–present)

Tasks (continued)

9. Draw the lottery graph $G\langle 7, 3; 2 \rangle$. Place the vertices on the edge of an imaginary circle with the ticket $\{1, 2, 3\}$ at 3 o'clock, with the other tickets spaced equally around the circle and numbered in lexicographic order in an anti-clockwise direction. Use a full A4 page. (Hint: This graph has order 35 and size 210.)
10. Verify that the lottery graph $G\langle 7, 3; 2 \rangle$ is 12-regular, and use this fact to show that $L(7, 3; 2) > 2$.
11. Prove that, in fact, $L(7, 3; 2) > 3$ by considering all $\binom{7}{3} = 35$ playing sets comprising three tickets.
12. Prove that $L(7, 3; 2) \leq 4$ by producing a playing set of cardinality 4 that will guarantee a 2-prize.

The lottery examples considered up to this point have been unrealistically small. A popular set of lottery parameters is $\langle 49, 6; k \rangle$ where $k \in [6]$, as may be seen in [Table 16.2](#). The lottery numbers $L(49, 6; 1) = 8$, $L(49, 6; 2) = 19$ and $L(49, 6; 6) = 13\,983\,816$ in this class are known. The lottery numbers $L(49, 6; 3)$, $L(49, 6; 4)$ and $L(49, 6; 5)$ have, however, not yet been determined — in fact, computing these numbers is a well-known open problem in design theory. Best known bounds on these numbers are $89 \leq L(49, 6; 3) \leq 163$, $1\,352 \leq L(49, 6; 4) \leq 3\,977$ and $62\,151 \leq L(49, 6; 5) \leq 151\,771$, respectively. A playing set of 163 tickets guaranteeing a prize in the lottery $\langle 49, 6; 3 \rangle$ is shown in [Table 16.3](#).

Convince yourself that computing the lottery number $L(49, 6; 3)$ constitutes a formidable problem, by answering the following questions:

Tasks (continued)

13. What is the order of the lottery graph $G\langle 49, 6; 3 \rangle$?
14. What is the degree of regularity of the lottery graph $G\langle 49, 6; 3 \rangle$?
15. What is the size of the lottery graph $G\langle 49, 6; 3 \rangle$?
16. Suppose the vertices of $G\langle 49, 6; 3 \rangle$ were to be arranged 1cm apart along the edge of a circle, as described in [Task 9](#). How many 100m \times 50m soccer pitches would fit into this circle?

Project 16.2: Domination and independence in chess graphs

Recall that a placement of a number of copies of a particular chess piece is said to form a **dominating set** for an $n \times n$ chess board if each cell is reachable by at least one of the pieces within a single move. Similarly, a placement of a number of copies of a particular chess piece is said to form an **independent set** for an $n \times n$ chess board if no piece is reachable by any other piece within a single move. Now consider, for each of the chess pieces *kings*, *queens*, *knight*s, *bishop*s and *rook*s the following questions:

Question (I) What is the cardinality of a smallest dominating set for an $n \times n$ chess board?

01 02 03 04 05 49	02 06 10 14 18 49	06 09 11 13 18 19	23 25 30 37 46 48	25 31 35 39 44 47
01 02 06 11 17 20	02 07 11 15 19 49	07 08 10 12 18 19	23 26 29 32 39 45	25 31 36 42 44 47
01 02 07 10 16 21	02 08 12 16 20 49	07 08 11 13 20 21	23 26 32 35 42 45	25 34 37 40 45 48
01 02 08 13 15 18	02 09 13 17 21 49	07 09 10 13 14 15	23 26 34 36 41 47	26 28 41 43 45 48
01 02 09 12 14 19	03 04 06 08 18 20	07 09 11 12 16 17	23 27 32 39 40 41	26 29 30 35 45 47
01 03 06 10 15 19	03 04 07 09 19 21	08 09 14 16 18 21	23 27 36 38 45 47	26 29 32 45 46 47
01 03 07 11 14 18	03 04 10 12 14 16	08 09 15 17 19 20	23 28 31 35 37 42	26 29 36 42 45 46
01 03 08 12 17 21	03 04 11 13 15 17	10 11 14 17 19 21	23 29 30 36 39 42	26 30 32 34 41 42
01 03 09 13 16 20	03 05 06 09 14 17	10 11 15 16 18 20	23 29 37 43 44 48	26 30 32 36 39 45
01 04 06 13 14 21	03 05 07 08 15 16	12 13 14 17 18 20	23 30 31 33 43 46	26 34 35 39 41 46
01 04 07 12 15 20	03 05 10 13 18 21	12 13 15 16 19 21	23 30 34 38 40 46	26 37 38 40 43 44
01 04 08 11 16 19	03 05 11 12 19 20	22 23 25 31 34 48	23 30 35 36 46 47	27 28 29 31 34 37
01 04 09 10 17 18	03 06 11 16 21 49	22 23 25 35 42 43	23 33 35 42 44 48	27 29 33 34 44 48
01 05 06 12 16 18	03 07 10 17 20 49	22 23 28 30 44 46	24 25 26 27 37 48	27 29 34 35 38 41
01 05 07 13 17 19	03 08 13 14 19 49	22 24 26 27 28 44	24 25 28 33 41 45	27 30 31 37 43 44
01 05 08 10 14 20	03 09 12 15 18 49	22 24 31 41 45 48	24 25 31 38 44 45	27 30 32 38 42 45
01 05 09 11 15 21	04 05 06 07 10 11	22 24 33 37 40 41	24 26 27 31 33 43	27 30 35 40 41 47
01 06 07 08 09 49	04 05 08 09 12 13	22 25 27 29 34 43	24 26 29 34 40 45	27 32 34 35 36 40
01 10 11 12 13 49	04 05 14 15 18 19	22 25 30 36 39 43	24 27 34 39 42 47	27 35 38 39 45 46
01 14 15 16 17 49	04 05 16 17 20 21	22 25 32 43 46 47	24 28 38 40 43 48	27 36 40 41 42 46
01 18 19 20 21 49	04 06 12 17 19 49	22 26 31 38 40 48	24 29 30 32 40 42	28 30 31 36 37 39
02 03 06 07 12 13	04 07 13 16 18 49	22 26 33 37 38 45	24 29 35 39 40 46	28 31 32 37 46 47
02 03 08 09 10 11	04 08 10 15 21 49	22 27 28 31 46 48	24 30 34 35 38 47	28 32 35 39 43 48
02 03 14 15 20 21	04 09 11 14 20 49	22 28 29 38 41 44	24 32 35 36 38 41	28 32 36 42 43 48
02 03 16 17 18 19	05 06 13 15 20 49	22 28 34 40 44 45	24 34 36 38 42 46	28 35 36 43 47 48
02 04 06 09 15 16	05 07 12 14 21 49	22 29 30 31 33 48	24 37 41 43 44 45	28 39 42 43 47 48
02 04 07 08 14 17	05 08 11 17 18 49	22 32 33 35 36 37	25 26 28 33 38 40	29 31 33 38 41 43
02 04 10 13 19 20	05 09 10 16 19 49	22 33 37 39 42 47	25 26 31 40 41 44	30 33 36 39 44 48
02 04 11 12 18 21	06 07 14 16 19 20	22 34 37 43 44 46	25 28 29 33 44 46	31 33 34 40 43 45
02 05 06 08 19 21	06 07 15 17 18 21	23 24 27 30 41 46	25 28 30 33 34 43	32 33 44 46 47 48
02 05 07 09 18 20	06 08 10 13 16 17	23 24 29 36 40 47	25 29 37 38 41 48	38 39 40 41 42 47
02 05 10 12 15 17	06 08 11 12 14 15	23 24 32 34 38 39	25 31 32 35 36 44	
02 05 11 13 14 16	06 09 10 12 20 21	23 25 27 28 33 37	25 31 32 39 42 44	

Table 16.3: A playing set of 163 tickets guaranteeing a player a 3-prize in the lottery $\langle 49, 6; k \rangle$ [49].

Question (II) What is the cardinality of a largest independent set for an $n \times n$ chess board?

In order to develop a feeling for what is involved when attempting to answer these questions, observe that the cardinality of a smallest dominating set for a standard 8×8 chess board by queens is not more than 5. There are 638 different ways to form a dominating set of five queens on a standard 8×8 chess board (excluding reflexive and rotational symmetries) — see Figure 16.28 for one such solution. But is there perhaps a dominating set of four queens for an 8×8 chess board?

Furthermore, the cardinality of a largest independent set for a standard 8×8 chess board by queens is not smaller than 8. There are twelve different ways to form an independent set of eight queens on a standard 8×8 chess board (excluding reflexive and rotational symmetries) — see Figure 16.29 for these solutions. But is there perhaps an independent set of nine queens for an 8×8 chess board?

Tasks

- Verify that the placement in Figure 16.28 is, in fact, a dominating set, by convincing yourself that every cell of the chess board is in the line of attack of at least one queen.

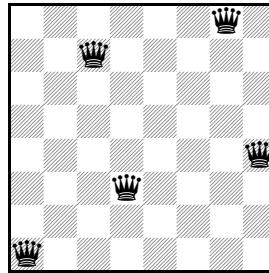


Figure 16.28: A dominating set of five queens on a standard 8×8 chess board.

2. Verify that each of the placements in [Figure 16.29](#) is, in fact, an independent set, by convincing yourself that no queen is in the line of attack of any other queen.

In order to translate the general questions posed above into the realm of graph theory, let us define the notion of a chess graph for the various chess pieces. The vertices of a **chess graph** for a particular chess piece represent the cells of the board, and two vertices are adjacent if the chess piece in question can move between the corresponding cells in one go. Call the graphs corresponding to a *king*, a *queen*, a *knight*, a *bishop* and a *rook* the **king's graph**, the **queen's graph**, the **bishop's graph**, the **knight's graph** and the **rook's graph**, denoted by \mathcal{K}_n , \mathcal{Q}_n , \mathcal{B}_n , \mathcal{N}_n and \mathcal{R}_n , respectively, for an $n \times n$ chess board. The graphs \mathcal{K}_4 , \mathcal{Q}_4 , \mathcal{B}_4 , \mathcal{N}_4 and \mathcal{R}_4 are shown in [Figure 16.30](#) as examples.

The graphs \mathcal{K}_n , \mathcal{Q}_n , \mathcal{B}_n , \mathcal{N}_n and \mathcal{R}_n all have order n^2 .

Tasks (continued)

3. Determine the sizes of the graphs \mathcal{K}_n , \mathcal{Q}_n and \mathcal{R}_n .

It is clear that the answers to [Question \(I\)](#) posed in the introduction to this project are the domination numbers $\gamma(\mathcal{K}_n)$, $\gamma(\mathcal{Q}_n)$, $\gamma(\mathcal{B}_n)$, $\gamma(\mathcal{N}_n)$ and $\gamma(\mathcal{R}_n)$, whilst the answers to [Question \(II\)](#) are the independence numbers $\alpha(\mathcal{K}_n)$, $\alpha(\mathcal{Q}_n)$, $\alpha(\mathcal{B}_n)$, $\alpha(\mathcal{N}_n)$ and $\alpha(\mathcal{R}_n)$. It would seem easier to answer [Question \(II\)](#) than it is to answer [Question \(I\)](#). Let us therefore consider [Question \(II\)](#) first.

Tasks (continued)

4. Prove that

$$\alpha(\mathcal{K}_n) = \begin{cases} \frac{1}{4}n^2 & \text{if } n \text{ is even,} \\ \frac{1}{4}(n+1)^2 & \text{if } n \text{ is odd.} \end{cases}$$

(Hint: A largest independent set of sixteen kings for a standard 8×8 chess board is shown in [Figure 16.31\(a\)](#).)

5. Prove that $\alpha(\mathcal{Q}_n) = n$ for all $n \in \mathbb{N}$.
6. Prove that $\alpha(\mathcal{B}_n) = 2n - 2$ for all $n \in \mathbb{N}$.
7. Prove that

$$\alpha(\mathcal{N}_n) = \begin{cases} \frac{1}{2}n^2 & \text{if } n \text{ is even,} \\ \frac{1}{2}(n^2 + 1) & \text{if } n \text{ is odd.} \end{cases}$$

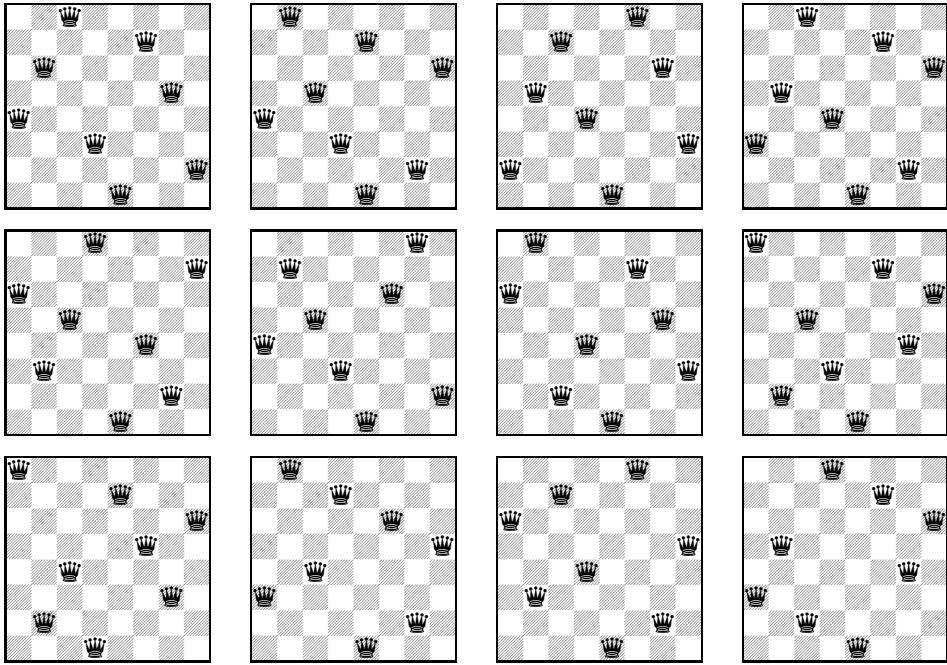


Figure 16.29: All twelve independent sets of eight queens on a standard 8×8 chess board.

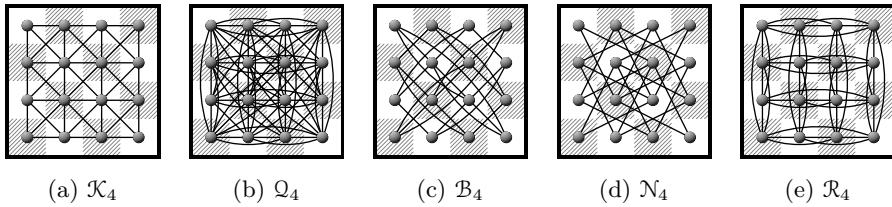


Figure 16.30: The chess graphs \mathcal{K}_4 , \mathcal{Q}_4 , \mathcal{B}_4 , \mathcal{N}_4 and \mathcal{R}_4 .

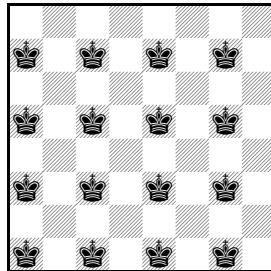
8. Prove that $\alpha(\mathcal{R}_n) = n$ for all $n \in \mathbb{N}$.

Question (I) has not yet been solved in closed form for general values of n in the case of the king's graph, the knight's graph and the queen's graph. Of these three open problems, the latter is certainly the most famous; it is known as the **queen's domination problem**. Values of $\gamma(\mathcal{Q}_n)$ are shown in Table 16.4 for $n \in \{3, \dots, 16\}$.

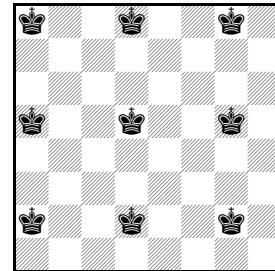
It is known that $\gamma(\mathcal{Q}_n) \geq (n - 1)/2$ for all $n \in \mathbb{N}$ and it is believed that $\gamma(\mathcal{Q}_n) \rightarrow \frac{n}{2}$ as $n \rightarrow \infty$, but nobody to date has been able to prove this!

Tasks (continued)

9. Find dominating sets of 3, 3, 4 and 5 queens for respectively a 5×5 , a 6×6 , a 7×7 and a 9×9 chess board.



(a)



(b)

Figure 16.31: (a) A largest independent set of sixteen kings and (b) a smallest dominating set of nine kings for a standard 8×8 chess board.

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\gamma(\mathcal{Q}_n)$	1	2	3	3	4	5	5	5	5	6	7	8	9	9

Table 16.4: Solutions to the queen's domination problem for $n \in \{3, \dots, 16\}$. See sequence [A075458](#) of Sloane [105].

Values of $\gamma(\mathcal{N}_n)$ are shown in [Table 16.5](#) for $n \in \{3, \dots, 16\}$, and a dominating set of twelve knights for an 8×8 chess board is shown in [Figure 16.32\(a\)](#).

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\gamma(\mathcal{N}_n)$	4	4	5	8	10	12	14	16	21	24	28	32	36	40

Table 16.5: Values of $\gamma(\mathcal{N}_n)$ for $n \in \{3, \dots, 16\}$. See sequence [A006075](#) of Sloane [105].

Tasks (continued)

10. Find dominating sets of 5, 8 and 10 knights for respectively a 5×5 , a 6×6 , and a 7×7 chess board.

For the bishop's graph and the rook's graph, [Question \(I\)](#) has been solved in closed form for general values of n .

Tasks (continued)

11. Prove that $\gamma(\mathcal{B}_n) = n$ for all $n \in \mathbb{N}$. (Hint: See [Figure 16.32\(b\)](#).)
12. Prove that $\gamma(\mathcal{R}_n) = n$ for all $n \in \mathbb{N}$.

Project 16.3: Total domination, policing and auditing

The notion of total domination may play a role in prison guarding and auditing applications. Consider, as an example of a prison guarding application of total domination, the **guarding capability graph** $G_{16,46}$ in [Figure 16.33\(a\)](#). The vertices g_1, \dots, g_5 on the left-hand side of the graph represent prison guards, whilst

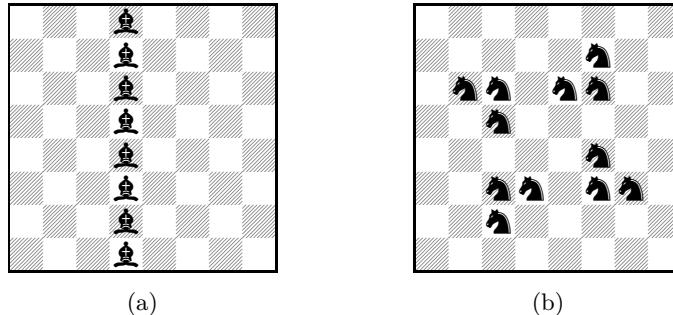


Figure 16.32: (a) A smallest dominating set of eight bishops and (b) a smallest dominating set of twelve knights for a standard 8×8 chess board.

the vertices p_1, \dots, p_8 on the right-hand side represent prison cells, each populated by prisoners having committed similar crimes. Based on the experience and seniority of the prison guards and the crimes of the prisoners in question, guards are capable of monitoring certain prison cells — these capabilities are indicated by edges joining vertex pairs of the form $g_i p_j$. Furthermore, in order to safeguard, to some extent, against irregularities as a result of possible guard-prisoner relationships or favouritism, each prison guard, in turn, has to be observed or overseen by at least one other prison guard. Guards of similar seniority are capable of overseeing each other and these capabilities are indicated by means of edges joining vertex pairs of the form $g_i g_j$. Determining the minimum number of guards that have to be on duty at any given time, involves finding, among the vertices g_1, \dots, g_5 , a total dominating set of $G_{16.46}$.

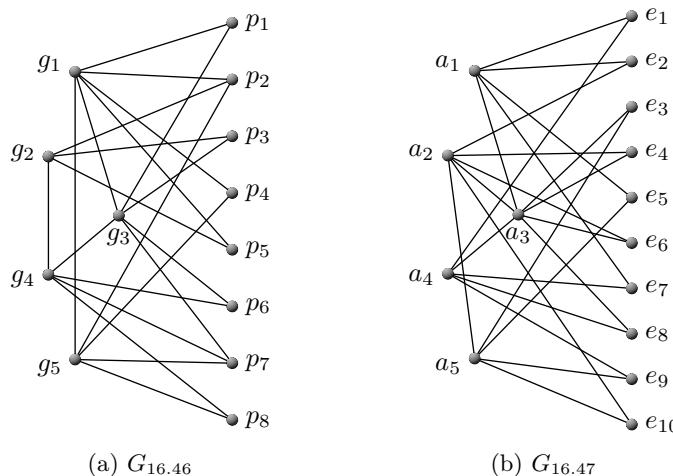


Figure 16.33: The guarding capability graph $G_{16.46}$ and the auditing capability graph $G_{16.47}$.

Consider, as an example of an auditing application of total domination, the **auditing capability graph** $G_{16.47}$ in Figure 16.33(b). The vertices a_1, \dots, a_5 on the left-hand side of the graph represent auditors employed by an auditing firm,

whilst the vertices e_1, \dots, e_{10} on the right-hand side represent employees of an accounting firm. Based on the qualifications of the auditors and the job descriptions of the accounting firm employees, auditors are capable of auditing the paper trail left by certain accounting firm employees — these capabilities are indicated by edges joining vertex pairs of the form $a_i e_j$. Furthermore, in order to guard against irregularities as a result of possible auditor-accounting firm employee relationships or corruption, the work of each auditor has to be audited by at least one other auditor. Auditors capable of auditing each other's work are indicated by means of edges joining vertex pairs of the form $a_i a_j$. Determining the minimum number of auditors that have to be contracted during an audit, involves finding, among the vertices a_1, \dots, a_5 , a total dominating set of $G_{16.47}$.

Tasks

1. Find, among the vertices g_1, \dots, g_5 , a total dominating set of the guarding capability graph $G_{16.46}$ in [Figure 16.33\(a\)](#).
2. Find, among the vertices a_1, \dots, a_5 , a total dominating set of the auditing capability graph $G_{16.47}$ in [Figure 16.33\(b\)](#).

Project 16.4: Roman domination

In his article “Defend the Roman Empire!”, [Stewart \[109\]](#) describes a fascinating application of domination. According to [Luttwak \[83\]](#), the Roman empire had approximately fifty fighting *legions* at its disposal (each comprising various infantry and cavalry *units*) to defend even the farthest reaches of its territories during its occupation of Europe in the third century A.D. Losing much of its power thereafter, the empire had only twenty five legions available during the century that followed. Emperor Constantine the Great (274–337 A.D.) was faced with the problem of deploying this limited number of legions to protect the empire from frequent uprisings against the authority of Rome.

A grouping of six legions, called a *field army*, was deemed sufficient to secure any single region of the empire. Hence, the emperor had four field armies at his disposal. Consider the eight regions of the empire shown in [Figure 16.34\(a\)](#). A deployment of these field armies was considered able to secure the empire if every region was either occupied by a field army, or was directly adjacent to a region that was occupied by two field armies. By decree of the emperor, two field armies had to be stationed in a region, before one would be allowed to move to an unoccupied, neighbouring region to quell an uprising. This decree was an attempt to ensure that the region vacated by the moving field army could not be attacked successfully.

Tasks

1. What was the minimum number of field armies required to secure the empire in accordance with the imperial decree described above?
2. If the available number of field armies was smaller than this minimum, how should the field armies have been stationed in order to defend the largest number of regions?

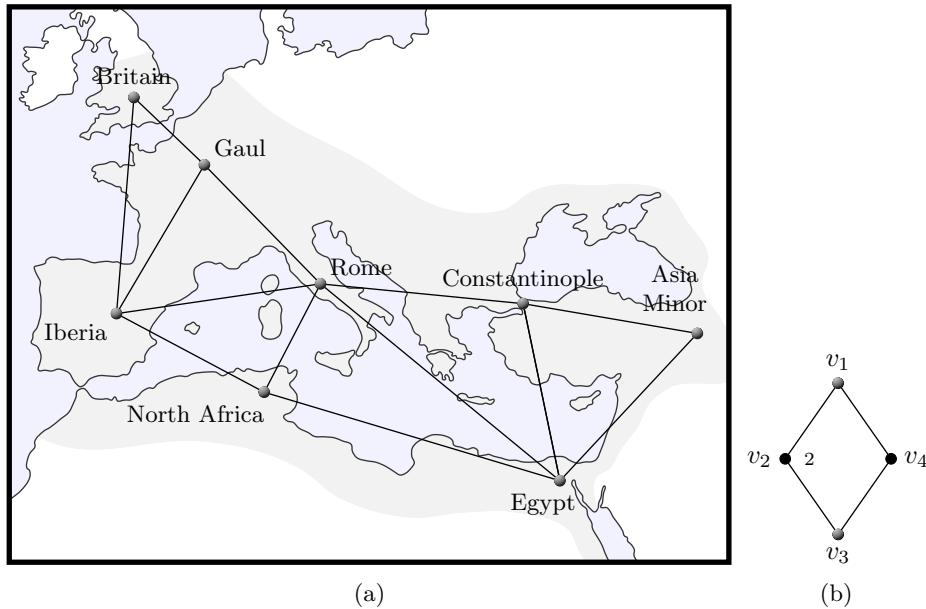


Figure 16.34: (a) A map of the eight regions (Britain, Gaul, Iberia, Rome, Constantinople, Asia Minor, Egypt and North Africa) of the Roman empire during the 3rd and 4th centuries A.D. with a graph model of those regions superimposed on the map. (b) An example of a Roman dominating set for the 4-cycle (solid vertices contain armies).

The sought-after military protection strategy described above has given rise to an active research field within graph theory, called *graph protection* [20, 24, 63] in which various protection strategies have been considered. In particular, a subset of the vertex set $V(G)$ of a graph G is a **Roman dominating set** if every vertex in $V(G)$ either has at least one field army assigned to it, or is adjacent to a vertex with two field armies assigned to it. The **Roman domination number** of G is the cardinality of a smallest Roman dominating set of G , and is denoted by $\gamma_R(G)$. An example of a smallest Roman dominating set for the 4-cycle C_4 is shown in Figure 16.34(b). The “2” next to the vertex v_2 indicates that two field armies are assigned to that vertex, while only one field army is assigned to the vertex v_3 . According to the imperial decree, the field army at the vertex v_3 may therefore not move. Any one of the field armies at the vertex v_2 is, however, able to move to either v_1 or v_4 , should there be an uprising.

In Task 1 above the reader was, therefore, asked to determine the Roman domination number of the graph underlying the map of the empire shown in Figure 16.34(a).

Tasks (continued)

3. Prove that $\gamma(G) \leq \gamma_R(G) \leq 2\gamma(G)$ for any graph G , where $\gamma(G)$ denotes the ordinary domination number of a graph, as usual.
4. Prove that $\gamma_R(G) \geq 2n/(\Delta + 1)$ for any graph with maximum degree $\Delta \geq 1$.

5. Prove that

$$\gamma_R(K_n) = \begin{cases} 1 & \text{if } n = 1, \\ 2 & \text{otherwise,} \end{cases}$$

for the complete graph K_n of order $n \in \mathbb{N}$.

6. Prove that $\gamma_R(P_n) = \lceil 2n/3 \rceil$ for the path P_n of order $n \in \mathbb{N}$.
 7. Prove that $\gamma_R(C_n) = \lceil 2n/3 \rceil$ for the cycle C_n of order $n \in \mathbb{N}$.
 8. Consider the complete bipartite graph $K_{p,q}$ with $p, q \in \mathbb{N}$ satisfying $p \leq q$.
 Prove that

$$\gamma_R(K_{p,q}) = \begin{cases} 2 & \text{if } p = 1, \\ 3 & \text{if } p = 2, \\ 4 & \text{if } p \geq 3. \end{cases}$$

9. Show that

$$\gamma_R(P_n \square P_k) \leq 2\left(\left\lceil \frac{nk}{5} \right\rceil + \left\lceil \frac{n}{5} \right\rceil + \left\lceil \frac{k}{5} \right\rceil\right)$$

for the cartesian product $P_n \square P_k$ (a rectangular grid graph).

Project 16.5: Distance domination

Schools sometimes provide buses which pick up children close to their homes, transport them to school in the morning and transport them back home in the afternoon. Such buses normally operate under a rule that no child should have to walk further than some specified distance in order to reach the nearest bus stop. This means that the route taken by the bus has to be designed to adhere to this rule.

David Erwin was born in London, United Kingdom in 1972 and moved to South Africa while very young. He received a master's degree from the University of Natal, Durban in South Africa in 1995 under the supervision of Henda Swart and a doctorate at Western Michigan University in 2001 under the supervision of [Gary Chartrand](#), both in graph theory. After a three-year period as Dorwart Visiting Assistant Professor at Trinity College in Hartford, Connecticut, he joined the School of Mathematical Sciences at the University of KwaZulu-Natal in 2004. In 2012, he moved to the Department of Mathematics and Applied Mathematics at the University of Cape Town, where he is a member of the Laboratory for Discrete Mathematics and Theoretical Computer Science. He is the author of a number of publications in different areas of graph theory and has also been managing editor (graph theory) for the journal *Utilitas Mathematica*.



Biographic note 87: David Erwin (1972–present)

This situation gives rise to the notion of distance domination. For $k \in \mathbb{N}$, a subset D_k of the vertex set $V(G)$ of a graph G is a **distance- k dominating set** of G if

every vertex of G is at distance at most k from some vertex in D_k . The cardinality of a smallest distance- k dominating set of G is called the **distance- k domination number** of G , denoted by $\gamma(G; k)$. Note that in the special case where $k = 1$, the notion of classical graph domination is recovered, and hence that $\gamma(G; 1) = \gamma(G)$ for any graph G .

The bus routing problem may be solved by finding a distance- k dominating set (which is as small as possible) for the graph underlying the street plan of the region concerned (*i.e.* whose vertices represent street intersections or dead-ends, and whose edges represent streets), ensuring that the vertex representing the school bus stop is in the distance- k dominating set.

Consider, for example, the idealised street map in Figure 16.35, where vertices represent street intersections and where edges represent streets. Suppose the school is located at the circled intersection and that the homes of the children are located anywhere along the edges of the graph. Suppose further that there is a rule that no child should have to walk more than the combined length of two street blocks to reach the nearest bus stop. The solid vertices in the figure represent a distance-2 dominating set, which may be taken as bus stops, and the arrows indicate a shortest, closed bus route from the school to these bus stops.

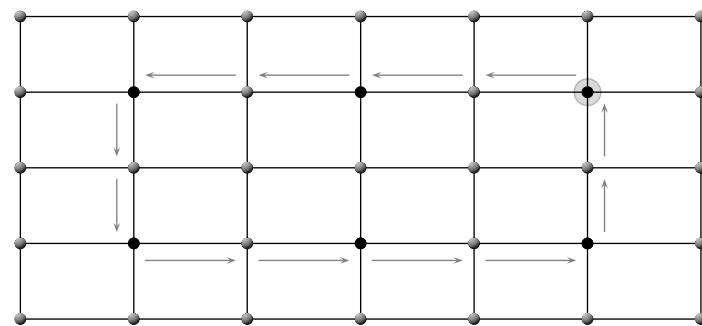


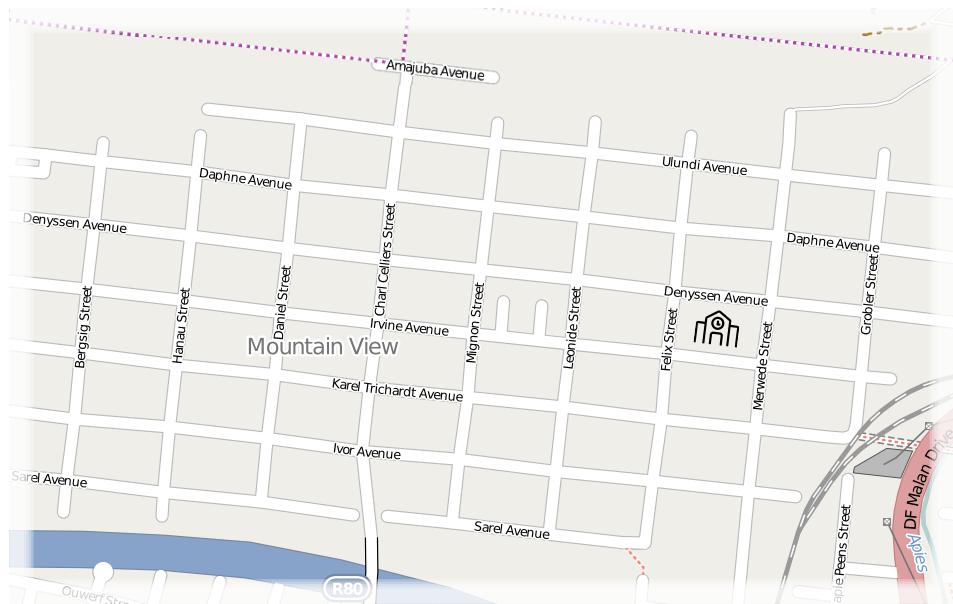
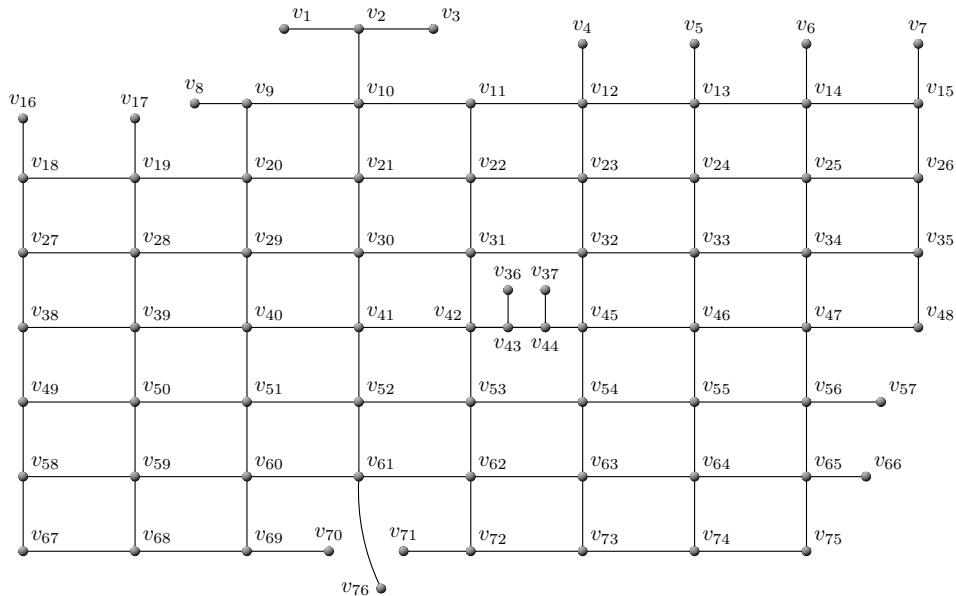
Figure 16.35: A closed route through a distance-2 dominating set of the grid graph $P_5 \square P_7$.

Task

Use the method described above to find a closed route for a bus of Bergsig Primary School in the *Tshwane* suburb of *Mountain View* shown in Figure 16.36(a) such that no child is required to walk further than two street blocks in order to reach the nearest bus stop. Indicate the bus stops by means of circled vertices and the route by means of arrows on the graph in Figure 16.36(b), as was done in Figure 16.35. The school bus stop is on the corner of **MERWEDE** and **DENYSSEN** streets, and your route should be as short as possible.

Project 16.6: $\langle r, s \rangle$ -domination

Suppose a network facility location problem is modelled by means of a graph G , and let r and s be nonnegative integers. The placement of a number of commodities on the vertices of G so that at least s commodities are located in the vicinity of (*i.e.* in the closed neighbourhood of) every vertex, and with no more than r commodities

(a) Street map of the *Tshwane* suburb of *Mountain View*.

(b) Underlying graph of the street map in (a).

Figure 16.36: A map and graph model of the *Tshwane* suburb of *Mountain View*.

Peter Dankelmann was born on 7 October 1962 and educated in Germany. After completing a doctorate at RWTH Aachen University under the supervision of [Lutz Volkmann](#), he joined the University of Natal in Durban in 1993, initially as a postdoctoral researcher, to work with the late Henda Swart. In 2012, he moved to the University of Johannesburg. He has published many research articles and his main research interests are distances in graphs and digraphs, domination and connectivity. The Academy of Science of South Africa elected him as a member in 2007. He is also an accomplished chess player, having won the title of provincial chess champion twice. He is editor-in-chief of the journal *Communications in Combinatorics and Optimization*.



Biographic note 88: Peter Dankelmann (1962–present)

placed at any vertex, is called an $\langle r, s \rangle$ -dominating set of G . The cardinality of a smallest $\langle r, s \rangle$ -dominating set of G is called the $\langle r, s \rangle$ -domination number of G and is denoted by $\gamma\langle G, r, s \rangle$. Note that in the special case where $r = s = 1$, the notion of classical graph domination is recovered, and hence that $\gamma\langle G, 1, 1 \rangle = \gamma(G)$ for any graph G .

The problem of finding $\gamma\langle G, r, s \rangle$ for some graph G was suggested by [Cockayne](#) [18] and often arises in logistics management applications, where the aim is to have sufficient commodities within easy reach of every vertex (so as to satisfy some customer demand), without over-investing by placing too many commodities at any vertex.

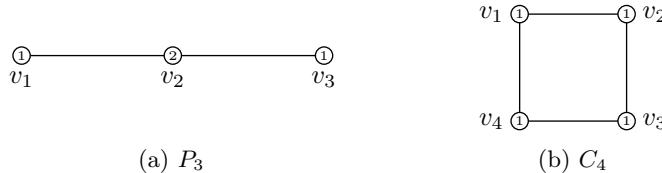


Figure 16.37: Minimum cardinality $\langle 2, 3 \rangle$ -dominating sets for P_3 and C_4 .

Consider, as examples, the 4-cycle C_4 and the path P_3 of order 3. A $\langle 2, 3 \rangle$ -dominating set of P_3 is shown in [Figure 16.37\(a\)](#), whilst a $\langle 2, 3 \rangle$ -dominating set of C_4 is shown in [Figure 16.37\(b\)](#). A number inside a vertex in the figure represents the number of commodities placed at the vertex in question. The $\langle 2, 3 \rangle$ -dominating sets shown in [Figure 16.37](#) therefore each has cardinality 4, which is a minimum in both cases, yielding the results $\gamma\langle P_3, 2, 3 \rangle = 4$ and $\gamma\langle C_4, 2, 3 \rangle = 4$, respectively.

Tasks

1. Prove that a graph G with minimum degree δ possesses an $\langle r, s \rangle$ -dominating set if and only if $s \in [(\delta + 1)r]$.
2. Determine $\gamma\langle P_6, 2, 3 \rangle$ and $\gamma\langle P_6, 3, 2 \rangle$ for the path P_6 of order 6.

Adriana Roux was born in Calvinia, South Africa on 4 November 1982. She obtained a doctorate in Operations Research at Stellenbosch University in 2014 under the supervision of Jan van Vuuren. Her doctoral thesis was on $\langle r, s \rangle$ -domination. She currently teaches applied discrete mathematics at Stellenbosch University. Her main research interests are related to domination and irredundance parameters in graphs, specifically towards their criticality with respect to the removal of vertices and edges.



Biographic note 89: Adriana Roux (1982–present)

3. Determine $\gamma\langle C_6, 2, 3 \rangle$ and $\gamma\langle C_6, 3, 2 \rangle$ for the cycle C_6 of order 6.

Recall that a subset P of the vertex set $V(G)$ of a graph G is a **packing** of G if the closed neighbourhoods of any two distinct vertices in P are disjoint (*i.e.* if $N[u] \cap N[v] = \emptyset$ for any $u, v \in P$, with $u \neq v$). The **packing number** of G is the cardinality of a largest packing of G and is denoted by $\rho(G)$. For the graphs in Figure 16.37 it holds that $\rho(P_3) = 1$ and $\rho(C_4) = 1$, for example.

Burger and Van Vuuren [13] proved in 2008 that

$$\left\lceil \frac{sn}{\Delta + 1} \right\rceil \leq \gamma\langle G, r, s \rangle \leq rn - \rho \min\{r(\delta + 1) - s, r\} \quad (16.14)$$

for any graph G of order n with minimum degree δ , maximum degree Δ and packing number ρ , for all $s \in [(\delta + 1)r]$.

Tasks (continued)

4. If $r = s = 1$, then the bounds in (16.14) reduce to

$$\left\lceil \frac{n}{\Delta + 1} \right\rceil \leq \gamma(G) \leq n - \rho. \quad (16.15)$$

- (a) Identify any infinite class of graphs for which the lower bound in (16.15) is attained.
- (b) Identify any infinite class of graphs for which the upper bound in (16.15) is attained.
5. Use the bounds in (16.14) to determine $\gamma\langle K_n, r, s \rangle$ for the complete graph K_n of order $n \in \mathbb{N}$, where $s \in [nr]$.
6. Prove that $\gamma\langle C_n, r, s \rangle = \lceil sn/3 \rceil$ for the cycle C_n of order $n \in \mathbb{N}$, where $s \in [3r]$.
7. Find an $\langle r, s \rangle$ -dominating set for the Cartesian product $C_n \square C_n$ of two cycles of order n , where $s \in [3r]$.

For more information on $\gamma\langle r, s \rangle$ -domination, the reader is referred to the doctoral thesis of Roux [101].

Teresa W Haynes was born in 1953 in Pikeville, Kentucky. She obtained a doctorate in computer science at the University of Central Florida under the direction of Robert Brigham. She started her academic career at East Tennessee State University, where she is currently a professor in the Department of Mathematics and Statistics. She is best known for her outstanding contributions to domination theory in graphs. She is a prolific researcher in various areas of graph theory, having co-authored many journal papers, and is a co-author of *Fundamentals of Domination in Graphs* and co-editor of *Domination in Graphs, Advanced Topics*.



Biographic note 90: Teresa Haynes (1953–present)

Peter J Slater was born in 1946 in the United States of America. He obtained a doctorate in mathematics from the University of Iowa in 1973 under the supervision of Tom Price and **Stephen T Hedetniemi**. He taught at Cleveland State University, before holding a National Research Council Postdoctoral Fellowship at the National Bureau of Standards for one year. After spending six years in the applied mathematics division at Sandia Laboratories in Albuquerque, New Mexico, he took up a position in the Department of Mathematical Sciences at the University of Alabama in Huntsville in 1981, where he was later promoted to full professor with a joint position in the Department of Mathematical Sciences and the Department of Computer Science. He was an internationally known graph theorist and mathematician. Perhaps best known for his co-authorship of the authoritative book *Fundamentals of domination in graphs*, he was also co-editor of the book *Domination in graphs: Advanced topics*. He published prolifically and was especially known for having introduced or pioneered many graph theoretic concepts that are widely researched today. He died on 27 September 2016.



Biographic note 91: Peter Slater (1946–2016)

Further reading

The books by **Haynes et al.** [56, 57] deal exclusively with domination in graphs. Recent survey articles on domination in graphs can be found in [55] and [60]. For a comprehensive bibliography of papers on dominating sets in graphs, we refer the reader to excellent bibliography compiled in [57] which contains over 1 200 entries.

- [1] R Aharoni and T Szabó, 2009. *Vizing's conjecture for chordal graphs*, Discrete Mathematics, **309**, pp. 1766–1768.

- [2] RB Allan and R Laskar, 1978. *On domination and independent domination numbers of a graph*, Discrete Mathematics, **23**, pp. 73–76.
- [3] D Archdeacon, J Ellis-Monaghan, D Fischer, D Froncek, PCB Lam, S Seager, B Wei and R Yuster, 2004. *Some remarks on domination*, Journal of Graph Theory, **46**, pp. 207–210.
- [4] A Babikir and MA Henning, 2020. *Domination versus independent domination in graphs of small regularity*, Discrete Mathematics, Manuscript 111727.
- [5] C Berge, 1962. *Theory of Graphs and its Applications*, Methuen, London, pp. 40–51.
- [6] C Berge, 1973. *Graphs and Hypergraphs*, North-Holland, Amsterdam.
- [7] B Bollobás and EJ Cockayne, 1979. *Graph-theoretic parameters concerning domination, independence, and irredundance*, Journal of Graph Theory, **3**, pp. 241–249.
- [8] B Brešar, 2005. *Vizing-like conjecture for the upper domination of Cartesian products of graphs — The proof*, Electronic Journal of Combinatorics, **12**, p. 6.
- [9] B Brešar, P Dorbec, W Goddard, BL Hartnell, MA Henning, S Klavžar and D Rall, 2012. *Vizing's conjecture: A survey and recent results*, Journal of Graph Theory, **69**, pp. 46–76.
- [10] B Brešar, BL Hartnell, MA Henning, K Kuenzel and DF Rall, 2021. *A new framework to approach Vizing's conjecture*, Discussiones Mathematicae Graph Theory, **41(3)**, pp. 749–762.
- [11] RC Brigham, JR Carrington and RP Vitray, 2000. *Connected graphs with maximum total domination number*, Journal of Combinatorial Mathematics and Combinatorial Computing, **34**, pp. 81–96.
- [12] C Bujtás, MA Henning and ZS Tuza, 2012. *Transversals and domination in uniform hypergraphs*, European Journal of Combinatorics, **33**, pp. 62–71.
- [13] AP Burger and JH van Vuuren, 2008. *On the $\langle r, s \rangle$ -domination number of a graph*, Ars Combinatoria, **88**, pp. 257–271.
- [14] SR Campbell, MN Ellingham and GF Royle, 1993. *A characterization of well-covered cubic graphs*, Journal of Combinatorial Mathematics and Combinatorial Computing, **13**, pp. 193–212.
- [15] V Chvátal and C McDiarmid, 1992. *Small transversals in hypergraphs*, Combinatorica, **12**, pp. 19–26.
- [16] WE Clark and S Suen, 2000. *An inequality related to Vizing's conjecture*, Electronic Journal of Combinatorics, **7(N4)**.
- [17] EJ Cockayne, *Private communication*.
- [18] EJ Cockayne, *Towards a theory of $\langle r, s \rangle$ -domination in graphs*, Unpublished manuscript.
- [19] EJ Cockayne, RM Dawes and ST Hedetniemi, 1980. *Total domination in graphs*, Networks, **10**, pp. 211–219.

- [20] EJ Cockayne, PA Dreyer, SM Hedetniemi and ST Hedetniemi, 2004. *Roman domination in graphs*, Discrete Mathematics, **278**, pp. 11–22.
- [21] EJ Cockayne, O Favaron, H Li and G MacGillivray, 1991. *The product of the independent domination numbers of a graph and its complement*, Discrete Mathematics, **90**, pp. 313–317.
- [22] EJ Cockayne, O Favaron, C Payan and AG Thomason, 1981. *Contributions to the theory of domination, independence and irredundance in graphs*, Discrete Mathematics, **33**, pp. 249–258.
- [23] EJ Cockayne, G Fricke and CM Mynhardt, 1995. *On a Nordhaus-Gaddum type problem for independent domination*, Discrete Mathematics, **138**, pp. 199–205.
- [24] EJ Cockayne, PJP Grobler, WR Gründlingh, J Munganga and JH van Vuuren, 2005. *Protection of a graph*, Utilitas Mathematica, **67**, pp. 19–32.
- [25] EJ Cockayne and ST Hedetniemi, 1977. *Towards a theory of domination in graphs*, Networks, **7**, pp. 247–261.
- [26] EJ Cockayne, ST Hedetniemi and DJ Miller, 1978. *Properties of hereditary hypergraphs and middle graphs*, Canadian Mathematical Bulletin, **21**, pp. 461–468.
- [27] EJ Cockayne and CM Mynhardt, 1993. *The sequence of upper and lower domination, independence and irredundance numbers of a graph*, Discrete Mathematics, **122**, pp. 89–102.
- [28] EJ Cockayne and CM Mynhardt, 1997. *Irredundance and maximum degree in graphs*, Combinatorics, Probability and Computing, **6**, pp. 153–157.
- [29] CF de Jaenisch, 1862. *Applications de l'Analyse Mathematique an Jenudes Echecs*, Petrograd.
- [30] P Dorbec, MA Henning and DF Rall, 2008. *On the upper total domination number of Cartesian products of graphs*, Journal of Combinatorial Optimization, **16**, pp. 68–80.
- [31] R Faudree, O Favaron and H Li, 1998. *Independence, domination, irredundance, and forbidden pairs*, Journal of Combinatorial Mathematics and Combinatorial Computing, **26**, pp. 193–212.
- [32] O Favaron, 1986. *Stability, domination and irredundance in a graph*, Journal of Graph Theory, **10**, pp. 429–438.
- [33] O Favaron, 1988. *Two relations between the parameters of independence and irredundance*, Discrete Mathematics, **70**, pp. 17–20.
- [34] O Favaron, 1992. *A bound on the independent domination number of a tree*, Vishwa International Journal of Graph Theory, **1**, pp. 19–27.
- [35] O Favaron, MA Henning, CM Mynhardt and J Puech, 2000. *Total domination in graphs with minimum degree three*, Journal of Graph Theory, **34**, pp. 9–19.
- [36] AS Finbow and BL Hartnell, 1983. *A game related to covering by stars*, Ars Combinatoria, **16**, pp. 189–198.

- [37] AS Finbow, BL Hartnell and RJ Nowakowski, 1993. *A characterization of well covered graphs of girth 5 or greater*, Journal of Combinatorial Theory, **57**, pp. 44–68.
- [38] AS Finbow, BL Hartnell and RJ Nowakowski, 1994. *A characterization of well-covered graphs that contain neither 4- nor 5-cycles*, Journal of Graph Theory, **18**, pp. 713–721.
- [39] J Fulman, 1993. *A note on the characterization of domination perfect graphs*, Journal of Graph Theory, **17**, pp. 47–51.
- [40] J Fulman, 1994. *A generalization of Vizing's theorem on domination*, Discrete Mathematics, **126**, pp. 403–406.
- [41] MR Garey and DS Johnson, 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York (NY).
- [42] J Gimbel and PD Vestergaard, 1995. *Inequalities for total matchings of graphs*, Ars Combinatoria, **39**, pp. 109–119.
- [43] NI Glebov and AV Kostochka, 1998. *On the independent domination number of graphs with given minimum degree*, Discrete Mathematics, **188**, pp. 261–266.
- [44] W Goddard and MA Henning, 2003. *Nordhaus-Gaddum bounds for independent domination*, Discrete Mathematics, **268**, pp. 299–302.
- [45] W Goddard and MA Henning, 2013. *Independent domination in graphs: A survey and recent results*, Discrete Mathematics, **313**, pp. 839–854.
- [46] W Goddard, MA Henning, J Lyle and J Southey, 2012. *On the independent domination number of regular graphs*, Annals of Combinatorics, **16(4)**, pp. 719–732.
- [47] W Goddard, MA Henning and HC Swart, 1992. *Some Nordhaus-Gaddum-type results*, Journal of Graph Theory, **16**, pp. 221–231.
- [48] W Goddard and J Lyle, 2012. *Independent dominating sets in triangle-free graphs*, Journal of Combinatorial Optimization, **23(1)**, pp. 9–20.
- [49] WR Gründlingh, 2004. *Two new combinatorial problems involving dominating sets for lottery schemes*, PhD thesis, Stellenbosch University, Stellenbosch.
- [50] B Hartnell and DF Rall, 1998. *Domination in Cartesian products: Vizing's conjecture*, pp. 163–189 in *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York (NY).
- [51] BL Hartnell and MD Plummer, 1996. *On 4-connected claw-free well-covered graphs*, Discrete Applied Mathematics, **64**, pp. 57–65.
- [52] J Haviland, 1991. *On minimum maximal independent sets of a graph*, Discrete Mathematics, **94**, pp. 95–101.
- [53] J Haviland, 1995. *Independent domination in regular graphs*, Discrete Mathematics, **143**, pp. 275–280.
- [54] J Haviland, 2008. *Independent domination in triangle-free graphs*, Discrete Mathematics, **308**, pp. 3545–3550.

- [55] TW Haynes, 1997. *Domination in graphs: A brief overview*, Journal of Combinatorial Mathematics and Combinatorial Computing, **24**, pp. 225–237.
- [56] TW Haynes, ST Hedetniemi and PJ Slater, 1997. *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York (NY).
- [57] TW Haynes, ST Hedetniemi and PJ Slater, 1997. *Fundamentals of Domination in Graphs*, Marcel Dekker, New York (NY).
- [58] ST Hedetniemi, DP Jacobs, R Laskar and D Pillone, *Open perfect neighborhood sets in graphs*, Unpublished manuscript.
- [59] MA Henning, 1995. *Irredundance perfect graphs*, Discrete Mathematics, **142**, pp. 125–135.
- [60] MA Henning, 1996. *Domination in graphs: A survey*, pp. 139–172 in G Chartrand and M Jacobson (Eds), *Surveys in Graph Theory*, Congressus Numerantium, Utilitas Mathematica Publishing, Winnipeg.
- [61] MA Henning, 2000. *Graphs with large total domination number*, Journal of Graph Theory, **35**, pp. 21–45.
- [62] MA Henning, 2009. *Recent results on total domination in graphs: A survey*, Discrete Mathematics, **309**, pp. 32–63.
- [63] MA Henning and ST Hedetniemi, 2003. *Defending the Roman empire — A new strategy*, Discrete Mathematics, **266**, pp. 239–251.
- [64] MA Henning, C Löwenstein and D Rautenbach, 2014. *Independent domination in subcubic bipartite graphs of girth at least six*, Discrete Applied Mathematics, **162**, pp. 399–403.
- [65] MA Henning and J Souayah, 2008. *A note on graphs with disjoint dominating and total dominating sets*, Ars Combinatoria, **89**, pp. 159–162.
- [66] MA Henning and A Yeo, 2007. *A transition from total domination in graphs to transversals in hypergraphs*, Quaestiones Mathematicae, **30**, pp. 417–436.
- [67] MA Henning and A Yeo, 2008. *Hypergraphs with large transversal number and with edge sizes at least three*, Journal of Graph Theory, **59**, pp. 326–348.
- [68] MA Henning and A Yeo, 2009. *Total domination in 2-connected graphs and in graphs with no induced 6-cycles*, Journal of Graph Theory, **60**, pp. 55–79.
- [69] MA Henning and A Yeo, 2013. *Total domination in graphs*, Springer, New York (NY).
- [70] W Imrich, S Klavžar and DF Rall, 2008. *Topics in Graph Theory: Graphs and Their Cartesian Product*, AK Peters/CRC Press, Wellesley (MA).
- [71] F Jaeger and C Payan, 1972. *Relations du type Nordhaus-Gaddum pour le nombre d'absorption d'un graphe simple*, Comptes Rendus de l'Academie des Sciences, **274**, pp. 728–730.
- [72] JP Joseph and S Arumugam, *A note on domination in graphs*, Unpublished manuscript.
- [73] JG Kalfleisch, RG Stanton and JD Horton, 1971. *On covering sets and error-correcting codes*, Journal of Combinatorial Theory, **11A**, pp. 233–250.

- [74] A Kelmans, 2006. *Counterexamples to the cubic graph domination conjecture*, RUTCOR Research Report, Rutgers University, Camden (NJ).
- [75] M Knor, R Škrekovski and A Tepeh, 2021. *Domination versus independent domination in regular graphs*, Journal of Graph Theory, **98**, pp. 525–530.
- [76] AV Kostochka and C Stocker, 2009. *A new bound on the domination number of connected cubic graphs*, Siberian Electronic Mathematical Reports, **6**, pp. 465–504.
- [77] AV Kostochka and BY Stodolsky, 2005. *On domination in connected cubic graphs*, Discrete Mathematics, **304**, pp. 45–50.
- [78] AV Kostochka and BY Stodolsky, 2009. *An upper bound on the domination number of n -vertex connected cubic graphs*, Discrete Mathematics, **309**, pp. 1142–1162.
- [79] PCB Lam, WC Shiu and L Sun, 1999. *On independent domination number of regular graphs*, Discrete Mathematics, **202**, pp. 35–144.
- [80] RC Laskar and J Pfaff, 1983. *Domination and irredundance in graphs*, Technical Report, Department of Mathematical Sciences, Clemson Univiversity, Clemson (SC).
- [81] RC Laskar, J Pfaff, SM Hedetniemi and ST Hedetniemi, 1984. *On the algorithmic complexity of total domination*, SIAM Journal on Algebraic and Discrete Methods, **5**, pp. 420–425.
- [82] CL Liu, 1968. *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York (NY).
- [83] EN Luttwak, 1976. *The grand strategy of the Roman empire*, Johns Hopkins University Press, Baltimore (MD).
- [84] D-X Ma and X-G Chen, 2004. *A note on connected bipartite graphs having independent domination number half their order*, Applied Mathematics Letters, **17**, pp. 959–962.
- [85] W McCuaig and B Shepherd, 1989. *Domination in graphs with minimum degree two*, Journal of Graph Theory, **13**, pp. 749–762.
- [86] JD McFall and R Nowakowski, 1980. *Strong independence in graphs*, Congressus Numerantium, **29**, pp. 639–656.
- [87] AA McRae, 1994. *Generalizing NP-completeness proofs for bipartite and chordal graphs*, PhD thesis, Clemson University, Clemson (SC).
- [88] S Mitchell and ST Hedetniemi, 1977. *Edge domination in trees*, pp. 489–509 in F Hoffman, L Lesniak-Foster, D McCarthy, RC Mullin, KB Reid and RG Stanton (Eds), *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing*, Congressus Numerantium, Utilitas Mathematica Publishing, Winnipeg.
- [89] SL Mitchell, EJ Cockayne and ST Hedetniemi, 1979. *Linear algorithms on recursive representations of trees*, Journal of Computer and System Sciences, **18**, pp. 76–85.
- [90] RJ Nowakowski and DF Rall, 1996. *Associative graph products and their independence, domination and coloring numbers*, Discussiones Mathematicae, **16**, pp. 53–79.

- [91] Ø Ore, 1962. *Theory of graphs*, Volume 38, American Mathematical Society, Providence (RI), pp. 206–212.
- [92] C Payan and NH Xuong, 1982. *Domination-balanced graphs*, Journal of Graph Theory, **6**, pp. 23–32.
- [93] J Pfaff, RC Laskar and ST Hedetniemi, 1983. *NP-completeness of total and connected domination and irredundance for bipartite graphs*, Technical Report, Department of Mathematical Sciences, Clemson University, Clemson (SC).
- [94] MD Plummer, 1970. *Some covering concepts in graphs*, Journal of Combinatorial Theory, **8**, pp. 91–98.
- [95] MD Plummer, 1993. *Well-covered graphs: A survey*, Quaestiones Mathematicae, **16**, pp. 253–287.
- [96] J Puech, 1998. *Irredundance perfection and P_6 -free graphs*, Journal of Graph Theory, **29**, pp. 239–255.
- [97] D Rautenbach, 1990. *A linear Vizing-like relation between the size and the domination number of a graph*, Journal of Graph Theory, **31**, pp. 297–302.
- [98] D Rautenbach and L Volkmann, 2002. *Independent domination and matchings in graphs*, Discrete Mathematics, **259**, pp. 325–330.
- [99] G Ravindra, 1977. *Well-covered graphs*, Journal of Combinatorics, Information and System Sciences, **2**, pp. 20–21.
- [100] BA Reed, 1996. *Paths, stars and the number three*, Combinatorics, Probability and Computing, **5**, pp. 277–295.
- [101] A Roux, 2014. *On the $\langle r, s \rangle$ -domination number of a graph*, PhD thesis, Stellenbosch University, Stellenbosch.
- [102] LA Sanchis, 1991. *Maximum number of edges in connected graphs with a given domination number*, Discrete Mathematics, **87**, pp. 64–72.
- [103] LA Sanchis, 2000. *On the number of edges in graphs with a given connected domination number*, Discrete Mathematics, **214**, pp. 193–210.
- [104] W Chee Shiu, X-G Chen and W Hong Chan, 2010. *Triangle-free graphs with large independent domination number*, Discrete Optimization, **7**, pp. 86–92.
- [105] NJA Sloane, *The online encyclopedia of integer sequences*, URL: <http://www.research.att.com/~njas/sequences/>.
- [106] J Southey, *Personal communication*, 2018.
- [107] J Southey and MA Henning, 2013. *Domination versus independent domination in cubic graphs*, Discrete Mathematics, **313(11)**, pp. 1212–1220.
- [108] J Southey and MA Henning, 2013. *Edge weighting functions on dominating sets*, Journal of Graph Theory, **72**, pp. 346–360.
- [109] I Stewart, 1999. *Defend the Roman Empire!*, Scientific American, **12**, pp. 136–138.
- [110] S Suen and J Tarr, 2010. *An improved inequality related to Vizing's conjecture*, The Electronic Journal of Combinatorics, **19(1)**, p. 8.

- [111] DP Sumner and JL Moore, 1979. *Domination perfect graphs*, Notices of the American Mathematical Society, **26**, A–569.
- [112] L Sun, 1995. *An upper bound for the total domination number*, Journal of the Beijing Institute of Technology, **4(2)**, pp. 111–114.
- [113] L Sun and J Wang, 1999. *An upper bound for the independent domination number*, Journal of Combinatorial Theory, Series B, **76(2)**, pp. 240–246.
- [114] S Thomassé and A Yeo, 2007. *Total domination of graphs and small transversals of hypergraphs*, Combinatorica, **27**, pp. 473–487.
- [115] J Topp and L Volkmann, 1990. *Well-covered and well-dominated block graphs and unicyclic graphs*, Mathematica Pannonica, **1/2**, pp. 55–66.
- [116] J Topp and L Volkmann, 1991. *On graphs with equal domination and independent domination numbers*, Discrete Mathematics, **96**, pp. 75–80.
- [117] Z Tuza, 1990. *Covering all cliques of a graph*, Discrete Mathematics, **86**, pp. 117–126.
- [118] LC van der Merwe, 1998. *Total domination edge critical graphs*, PhD thesis, University of South Africa, Pretoria.
- [119] LC van der Merwe, TW Haynes and CM Mynhardt, 1998. *Total domination edge critical graphs*, Utilitas Mathematica, **54**, pp. 229–240.
- [120] J Verstraëte, *Personal communication*.
- [121] VG Vizing, 1963. *The Cartesian product of graphs*, Vyčislitel’nye Sistemy, Novosibirsk, **9**, pp. 30–43.
- [122] VG Vizing, 1965. *A bound on the external stability number of a graph*, Doklady Akademii Nauk, **164**, pp. 729–731.
- [123] VG Vizing, 1968. *Some unsolved problems in graph theory*, Russian Mathematical Surveys, **23(6)**, pp. 125–141.
- [124] L Volkmann and VE Zverovich, 1997. *A proof of Favaron’s conjecture and a disproof of Henning’s conjecture on irredundance perfect graphs*, Proceedings of the 5th Twente Workshop on Graphs and Combinatorial Optimization, Twente, pp. 215–217.
- [125] L Volkmann and VE Zverovich, 2002. *A disproof of Henning’s conjecture on irredundance perfect graphs*, Discrete Mathematics, **254**, pp. 539–554.
- [126] L Volkmann and VE Zverovich, 2002. *Proof of a conjecture on irredundance perfect graphs*, Journal of Graph Theory, **41**, pp. 292–306.
- [127] HB Walikar, BD Acharya and E Sampathkumar, 1979. *Recent developments in the theory of domination in graphs*, Lecture Notes in Mathematics, Volume 1.
- [128] A Yeo, *Improved bound on the total domination in graphs with minimum degree four*, Unpublished manuscript.
- [129] S Zerbib, 2019. *An improved bound in Vizing’s conjecture*, Graphs and Combinatorics, **35(6)**, pp. 1401–1404.
- [130] IE Zverovich and VE Zverovich, 1991. *A characterization of domination perfect graphs*, Journal of Graph Theory, **15**, pp. 109–114.

- [131] IE Zverovich and VE Zverovich, 1995. *An induced subgraph characterization of domination perfect graphs*, Journal of Graph Theory, **20**, pp. 375–395.



Ramsey theory

Contents

17.1	Introduction	625
17.2	The classical two-colour Ramsey problem	628
17.3	Two-colour extensions	633
17.4	The multi-colour Ramsey problem	635
17.5	Multipartite Ramsey theory	636
17.6	Requirements other than that of a subgraph	641
	Exercises	644
	Computer exercises	645
	Projects	647
	Further reading	649

Ponder, for a moment, the remarkable fact that in *any* party of six people, at least three people are mutual acquaintances (know each other in pairs) or at least three people are mutual strangers (are pairwise strange to each other)! This fact is known as a *Friendship Theorem* or a *Theorem on Friends and Strangers*.¹

17.1 Introduction

To see why this Friendship Theorem holds, let us model the situation by means of a so-called **acquaintance graph** $G_{17.1}$ of order 6 whose vertices denote the six people, with two vertices being adjacent if and only if the corresponding people are acquaintances. Let v be a vertex of $G_{17.1}$. Since there are five other vertices in $G_{17.1}$, at least three of these vertices, x , y and z (say), are either (i) adjacent to v in $G_{17.1}$ or (ii) adjacent to v in the graph complement $\bar{G}_{17.1}$ as shown in Figure 17.1(a). Suppose (i) holds. If xy , xz or yz is an edge in $G_{17.1}$, then we have a triangle in $G_{17.1}$, representing three mutual acquaintances. If not, then we have a triangle in $\bar{G}_{17.1}$ representing three mutual strangers. Alternatively, suppose (ii) holds. If xy , xz or yz is an edge in $\bar{G}_{17.1}$, then we have a triangle in $\bar{G}_{17.1}$, representing three mutual strangers. If not, then we have a triangle in $G_{17.1}$ representing three mutual acquaintances.

¹A proof of this theorem (in slightly altered form) was required in the 1953 Putnam competition [34].

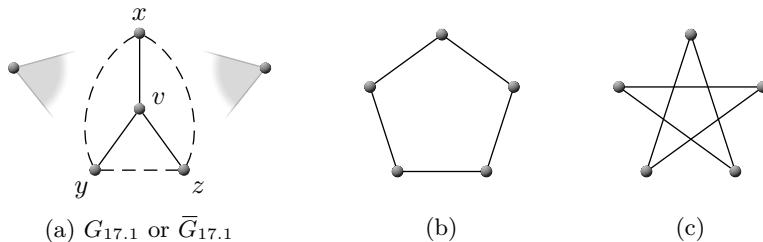


Figure 17.1: Acquaintance graphs.

The Friendship Theorem above may be understood in the context of a so-called Ramsey number. The (classical) **Ramsey number** $r = r(m, n)$ is the smallest natural number r with the property that if G is *any* graph of order r , then G contains the complete graph K_m as subgraph, or \bar{G} contains K_n as subgraph, or perhaps both.

In terms of Ramsey numbers we have just shown above that $r(3, 3) \leq 6$ — *i.e.* if G is *any* graph of order 6, then G contains the complete graph K_3 as subgraph, or \bar{G} contains K_3 as subgraph (or perhaps both). But is a graph of order 6 the *smallest* graph for which the above proof works? To see why $r(3, 3) > 5$, consider the 5-cycle C_5 shown in Figure 17.1(b). The complement of C_5 is again a 5-cycle, shown in Figure 17.1(c). It is clear that neither C_5 nor its complement contains K_3 as subgraph. Hence we cannot claim that if G is *any* graph of order 5, then G contains the complete graph K_3 as subgraph, or \bar{G} contains K_3 as subgraph (or perhaps both). We conclude that in a party of five or fewer people, the situation can occur where there are neither three mutual acquaintances nor three mutual strangers. Therefore, $r(3, 3) = 6$.

Frank Plumpton Ramsey was born in Cambridge, England on 22 February 1903. He entered Trinity College, Cambridge, to study mathematics and became a senior scholar in 1921, graduating as a wrangler in the Mathematical Tripos of 1923. He went to Vienna for a short while, returning to Cambridge where he was elected Fellow of King's College, Cambridge in 1924. Although Ramsey was a fellow of mathematics, he worked on a remarkable range of topics in mathematics, logic, philosophy and economics during his short academic career. He is best known for Theorem 17.1 and for his systematic attempt to base the mathematical theory of probability on the notion of partial belief. This attempt, as well as his work in economics, came about mainly because Ramsey was a close friend of the economist John Maynard Keynes (1883–1946). Ramsey suffered an attack of jaundice and was taken to Guy's Hospital in London for an operation. He died following the operation on 19 January 1930 at the age of 26.



Biographic note 92: Frank Ramsey (1903–1930)

The parameter $r(m, n)$ is called a *Ramsey* number in honour of British philosopher, economist and logician [Frank Plumpton Ramsey](#) (1903–1930) who during the 1920s was interested in decision procedures for systems of formal logic. During the course of his work he proved a famous theorem which was originally concerned with the colouring of subsets of a finite set, but which may be restated as follows in the language of graph theory [53].

Theorem 17.1 ([Ramsey's Theorem](#)) *For every $r \in \mathbb{N}$ there exists an $m \in \mathbb{N}$ such that, for every graph G of order at least r , the graph G itself or its complement \bar{G} contains the complete graph K_m of order m as subgraph.*

The popularity of [Ramsey's Theorem](#) began with its rediscovery in a classic 1935 paper of [Erdős and Szekeres](#) [27], to which we shall allude again later in this chapter. After a long embryonic stage, [Theorem 17.1](#) gave rise to what is today known as Ramsey theory, a subdiscipline of combinatorial analysis in which smallest structures are sought which necessarily induce smaller substructures of a specified nature. By many accounts Ramsey theory finally found its rightful place in combinatorial analysis during the 1973 Combinatorial Conference at Balatonfűred in Hungary, where close to thirty papers were devoted to topics in Ramsey theory. The interested reader is referred to [39, 53, 63] for fascinating accounts of the life and work of [Frank Ramsey](#), as well as the enormous amount of subsequent work spawned by [Theorem 17.1](#).

At first glance the result of [Theorem 17.1](#) may seem surprising. Upon reflection, however, one realizes that in order to force a complete subgraph K_m of order m in some graph G of order $n > m$ for values of m that are not trivially small when compared to n , many edges are required in G . Therefore, one may attempt to prevent K_m from appearing as a subgraph of G (i) by taking G to have too few edges, in which case its complement will have many edges, or (ii) by constructing G so that it has many edges, but “in the wrong places,” in which case one would attempt to impose too strong a structure on G to prevent K_m from occurring as a subgraph in \bar{G} as well (*i.e.* edges will be in the “correct places” in the graph complement \bar{G}). The existence result of [Theorem 17.1](#) may, of course, be generalised to the case where the complete subgraphs forced in the graph or its complement are not of the same order, giving rise to so-called off-diagonal Ramsey numbers of the form $r(m, n)$, where $m \neq n$.

We have seen that $r(3, 3) = 6$, and the following infinite class of Ramsey numbers is established similarly easily.

Theorem 17.2 $r(2, n) = n$ for all $n \geq 2$.

Proof Let G be a graph of order $n \geq 2$. If G has an edge, then G contains K_2 as subgraph; otherwise \bar{G} contains K_n as subgraph. Hence $r(2, n) \leq n$. Now consider the empty graph $H \cong \bar{K}_{n-1}$ of order $n - 1$. Since H does not contain K_2 as subgraph and \bar{H} does not contain K_n as subgraph, it follows that $r(2, n) > n - 1$. Consequently, $r(2, n) = n$. ■

❖ The reader should now be able to attempt Exercises [17.1–17.2](#).

17.2 The classical two-colour Ramsey problem

An intuitively more appealing way of thinking about Ramsey numbers stems from the realisation that if G is a graph of order r , then G may be obtained from a partition of the edge set of the complete graph K_r into two subsets, namely $E(G)$ and $E(\bar{G})$. We may think of this partition as a bi-colouring of the edges of K_r , and hence redefine the **Ramsey number** $r = r(m, n)$ as the smallest natural number such that, if the edges of K_r were to be bi-coloured arbitrarily using the colours red and blue, then either a complete graph of order m with all its edges coloured red or a complete graph of order n with all its edges coloured blue will be observed. We call such a bi-colouring of the edges of K_r a **red-blue edge colouring** of K_r , and we say that either a **red K_m** or a **blue K_n** appears or is forced as a subgraph of K_r , for short.

One advantage of this alternative definition of a Ramsey number is that it is very easy to see that the Ramsey numbers $r(m, n)$ and $r(n, m)$ must be equal. This follows by the ubiquity of the names of the colours “red” and “blue.” The following useful recursive upper bound on the Ramsey number $r(m, n)$ for any $m, n \geq 2$ is also easily established using the above alternative definition of a Ramsey number.

Theorem 17.3 *For any integers $m, n \geq 3$, the recursive inequality $r(m, n) \leq r(m - 1, n) + r(m, n - 1)$ holds. Moreover, if $r(m - 1, n)$ and $r(m, n - 1)$ are both even, then strict inequality holds.*

Proof Let $s = r(m - 1, n) + r(m, n - 1)$ and consider an arbitrary red-blue edge colouring of the complete graph K_s . We show that a red K_m or a blue K_n necessarily results as subgraph. Let v be an arbitrary vertex of K_s . Consider the edges joining v to the remaining $s - 1 = r(m - 1, n) + r(m, n - 1) - 1$ vertices. If there are at most $r(m - 1, n)$ red edges and at most $r(m, n - 1)$ blue edges, then v is joined to at most $s - 2$ vertices, a contradiction. Hence, there are $r(m - 1, n)$ red edges or $r(m, n - 1)$ blue edges incident with v . Without loss of generality, let us suppose that there are $s' = r(m - 1, n)$ red edges incident with v . Consider the graph $H \cong K_{s'}$ induced by the vertices joined to v by red edges and consider the red-blue edge colouring of H induced by the red-blue edge colouring of G . If H contains a blue K_n as subgraph, then so too does G . If not, then since $s' = r(m - 1, n)$, H contains a red K_{m-1} as subgraph. This subgraph together with the vertex v and the red edges joining v to it is a red subgraph K_m of G . It follows that $r(m, n) \leq s$, establishing the desired recursive inequality for the Ramsey number $r(m, n)$.

Suppose, next, that $r(m - 1, n)$ and $r(m, n - 1)$ are both even. Let $t = r(m - 1, n) + r(m, n - 1) - 1$ and consider an arbitrary red-blue edge colouring of the complete graph K_t . We show that a red K_m or a blue K_n necessarily occurs as a subgraph. Since t is odd, some vertex v of G' is incident with an even number of red edges (the subgraph induced by the red edges must have a vertex of even degree, since there are an even number of odd vertices in any graph by [Corollary 1.2](#)). If there are $r(m - 1, n)$ red edges incident with v , then proceeding as in the first part of the proof, a red K_m or a blue K_n occurs as subgraph. On the other hand, if v is incident with fewer than $r(m - 1, n)$ red edges, then since $r(m - 1, n)$ is even, v is incident with at most $r(m - 1, n) - 2$ red edges. But then v is incident with at least $r(m, n - 1)$ blue edges. Proceeding as in the first

part of the proof, a red K_m or a blue K_n once again occurs as a subgraph. Hence, $r(m, n) \leq t < r(m-1, n) + r(m, n-1)$. \blacksquare

The recursive upper bound in [Theorem 17.3](#) may be used to establish the values of small Ramsey numbers as we demonstrate in the following theorem, which is due to [Greenwood and Gleason \[36\]](#), and dates back to 1955.

Theorem 17.4

- (i) $r(3, 4) = 9$,
- (ii) $r(3, 5) = 14$.

Proof (i) It follows from [Theorems 17.2](#) and [17.3](#) that $r(3, 4) < r(2, 4) + r(3, 3) = 4 + 6 = 10$, because both $r(2, 4)$ and $r(3, 3)$ are even. Therefore $r(3, 4) \leq 9$.

Consider the red-blue edge colouring of K_8 shown in [Figure 17.2\(a\)–\(b\)](#) where the edges are coloured red and blue according to the circulant graph structures $C_8\langle 1, 4 \rangle$ and $C_8\langle 2, 3 \rangle$, respectively. Since $K_3 \not\subseteq C_8\langle 1, 4 \rangle$ and $K_4 \not\subseteq C_8\langle 2, 3 \rangle$, it follows that $r(3, 4) > 8$.

(ii) The Ramsey number $r(3, 5)$ may be established by methods similar to those in the proof of part (i); this is left as an exercise. \blacksquare



Figure 17.2: A red-blue edge colouring of K_8 for the proof of [Theorem 17.4\(i\)](#).

Note that an *analytical argument* was employed to establish an upper bound of 9 on the Ramsey number $r(3, 4)$ in [Theorem 17.4\(i\)](#), while a coinciding lower bound was found merely by producing a *counter example* in the form of a single red-blue edge colouring of K_{9-1} . This is a common dual approach towards establishing values of Ramsey numbers, often *employing computers* in the search for a suitable lower bound counter example.

In 1935, [Erdős and Szekeres \[27\]](#) produced the following beautiful proof of an upper bound in closed form on the Ramsey number $r(m, n)$ for any natural numbers $m, n \geq 2$.

Theorem 17.5 *If $m, n \geq 2$ are natural numbers, then*

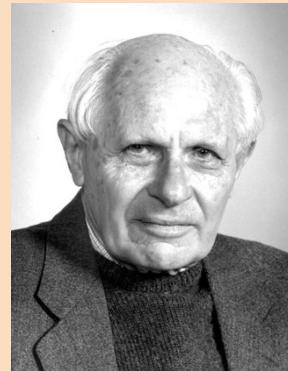
$$r(m, n) \leq \binom{m+n-2}{m-1}. \quad (17.1)$$

Proof By the strong form of induction over $k = m + n$. It follows from [Theorem 17.2](#) that equality holds in (17.1) if $m = 2$, for any $n \geq 2$, so that the bound in (17.1) is sharp for $4 \leq k \leq 5$, which serves as base case for the induction process. We therefore assume that $m, n \geq 3$ (*i.e.* $k \geq 6$) in the remainder of the proof.

Assume, as induction hypothesis, that

$$r(s, t) \leq \binom{s+t-2}{s-1}$$

George Szekeres was born in Budapest, Hungary on 29 May 1911 as Szekeres György and received a bachelor's degree in chemistry at the Technical University of Budapest. Thereafter, he started working in Budapest as an analytical chemist and in 1936 married Esther Klein. In order to escape Nazi persecution, Szekeres moved to Shanghai, China, where his son Peter was born. In China, he experienced World War II, the Japanese occupation and the start of the communist revolution. After the war he moved to Australia, where he worked as a mathematician at the University of Adelaide (1948–1963) and at the University of New South Wales from 1963 until his retirement in 1975. He was fond of setting problems for competitions, such as the annual mathematical olympiads and an undergraduate competition organised by the University of Sydney Mathematics Society. In 2001, the Australian Mathematical Society created the *George Szekeres Medal* in his honour and in 2002, he was made a Member of the Order of Australia (AM) "for service to mathematics and science, particularly as a contributor to education and research, to the support and development of the University of New South Wales Mathematics Competition and the Australian Mathematical Olympiad Team." George and his wife Esther died on the same day — 28 August 2005 — in Adelaide, Australia.



Biographic note 93: George Szekeres (1911–2005)

for all integers $s, t \geq 2$ satisfying $s + t < k$, where $k \geq 6$. Now let $m, n \geq 3$ be integers satisfying $m + n = k$. It follows from the induction hypothesis that

$$r(m-1, n) \leq \binom{m+n-3}{m-2} \quad \text{and} \quad r(m, n-1) \leq \binom{m+n-3}{m-1}.$$

It is left as an exercise to verify that

$$\binom{m+n-3}{m-2} + \binom{m+n-3}{m-1} = \binom{m+n-2}{m-1}, \quad (17.2)$$

which is known as *Pascal's identity*. It follows that

$$r(m-1, n) + r(m, n-1) \leq \binom{m+n-2}{m-1},$$

as desired. ■

It is also possible to establish analytic lower bounds on Ramsey numbers. A proof of the following lower bound, due to Spencer [62] in 1975, may proceed by the probabilistic method, but we postpone such a proof to the following chapter.

Theorem 17.6 $r(m, m) \geq m\sqrt{2^{m+1}}/e$ for any natural number $m \geq 2$.

The lower bound above may be generalised to off-diagonal Ramsey numbers (*i.e.* Ramsey numbers of the form $r(m, n)$, where $m \neq n$) by means of the following useful growth property showing that the grid of Ramsey numbers $r(m, n)$ is increasing in the directions of both its arguments m and n . We leave the proof of this growth property, as well as the generalisation of the lower bound, as exercises.

Theorem 17.7 *If $m, n, s, t \in \mathbb{N}$ satisfy the inequalities $2 \leq s \leq m$ and $2 \leq t \leq n$, then $r(s, t) \leq r(m, n)$, where equality holds if and only if $s = m$ and $t = n$.*

Unfortunately, the bounds in Theorems 17.5 and 17.6 cannot be used *per se* to determine the values of Ramsey numbers, because these bounds diverge very rapidly, as shown in Figure 17.3. These bounds may nevertheless be used as starting points in the search for exact Ramsey number values, although better bounds for small Ramsey numbers are generally attainable via other methods, as shown in Table 17.1.

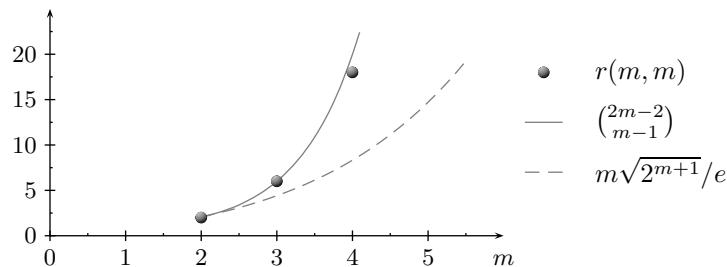


Figure 17.3: The bounds in Theorems 17.5 and 17.6. Exact values of (diagonal) Ramsey numbers are indicated by means of dots.

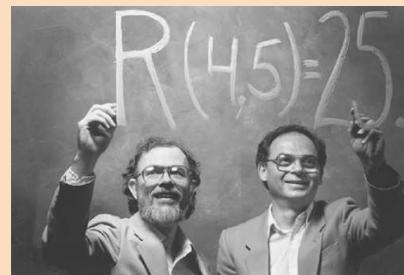
$\begin{matrix} n \rightarrow \\ \downarrow m \end{matrix}$	2	3	4	5	6	7	8	9
2	2	3	4	5	6	7	8	9
3		6	9	14	18	23	28	36
4			18	25	36 : 41	49 : 61	58 : 84	73 : 115
5				43 : 49	58 : 87	80 : 143	101 : 216	126 : 316
6					102 : 165	113 : 298	132 : 495	169 : 780
7						205 : 540	217 : 1031	241 : 1713
8							282 : 1870	317 : 3583
9								565 : 6588

Table 17.1: Best known bounds for the Ramsey number $r(m, n)$, for all $2 \leq n \leq m \leq 9$. Exact values are shown in bold face. Bounds are given in the form lower bound : upper bound, where the bounds are inclusive.

Apart from the result of Theorem 17.2 there are no general formulae for large classes of (classical) Ramsey numbers. In fact, only a handful of other Ramsey numbers $r(m, n)$ are known (for $m, n \geq 3$). These numbers are listed in Table 17.1. We encountered the Ramsey number $r(3, 3) = 6$ in the introduction, and the Ramsey numbers $r(3, 4) = 9$ and $r(3, 5) = 14$ in Theorem 17.4. These numbers,

as well as the Ramsey number $r(4, 4) = 18$ (which is the topic of [Project 17.1](#)), are due to [Greenwood and Gleason](#) [36]. The value of $r(3, 6) = 18$ is due to [Graver and Yackel](#) [35] and was established in 1968, while the value of $r(3, 7) = 23$ dates back to 1966 and is due to [Kalbfleisch](#) [49]. [McKay and Min](#) [51] established the Ramsey number $r(3, 8) = 28$ in 1992, whilst the Ramsey number $r(3, 9) = 36$ is due to [Grinstead and Roberts](#) [37] and its discovery dates back to 1982. Finally, the Ramsey number $r(4, 5) = 25$ is due to [McKay and Radziszowski](#) [52]; [Biographical Note 94](#) contains a picture of these two mathematicians celebrating their famous 1995 result.

Brendan Damien McKay (below, left) was born in Melbourne, Australia on 26 October 1951. He obtained a doctorate in mathematics from the University of Melbourne in 1980, with *Topics in computational graph theory* as thesis topic. He is currently professor of computer science at the Australian National University and is best known for his computational work in combinatorial designs, posets and Latin squares, as well as for his algorithm for graph isomorphism testing and its software implementation **NAUTY** (acronym for “No AUTomorphisms, Yes?”). He has also earned notoriety for his collaborative work with a group of Israeli mathematicians that criticises the *Bible code hypothesis* by arguing that the patterns in the Bible that supposedly have some predictive power may just as easily be found in other works, such as Leo Tolstoy’s *War and Peace*.



Stanisław P. Radziszowski (above, right) was born in Gdańsk, Poland on 7 June 1953. He obtained a doctorate in computer science from the Institute of Informatics at the University of Warsaw in 1980, with *Logic and complexity of synchronous parallel computations* as thesis topic. During the period 1976–1980 he was visiting professor at various universities in Mexico City. In 1984, he moved to the United States, where he has held a position in the Department of Computer Science at the Rochester Institute of Technology. He has published numerous results in graph theory, Ramsey theory, block designs, number theory and computational complexity. Perhaps his most cited contribution is the dynamic survey of currently known Ramsey numbers of various kinds, and best available bounds for such numbers [56], which has been maintained in eleven versions since 1994.

Biographic note 94: Brendan McKay (1951–present) and Stanisław Radziszowski (1953–present)

The most infamous Ramsey number that has thus far eluded evaluation is $r(5, 5)$, which is at least 43 and at most 49 according to [Table 17.1](#). It has, in fact, been conjectured by Exoo, [McKay](#) and [Radziszowski](#) that $r(4, 4) = 43$. Only time will tell whether or not this conjecture is true. [Paul Erdős](#) wrote about the problem of determining the value of $r(5, 5)$: “It must seem incredible to the uninitiated that

in the age of supercomputers $r(5, 5)$ is unknown. This, of course, is caused by the so-called combinatorial explosion: there are just too many cases to be checked” [26]. He even joked as follows about the expected difference between the difficulties of evaluating $r(5, 5)$ and $r(6, 6)$:

Suppose aliens were to invade the earth and threaten to destroy it within a year if human beings do not find the value of $r(5, 5)$. Then it would perhaps be possible to save the Earth by bringing together the world’s best minds and computers. But if the aliens were to demand the value of $r(6, 6)$, the human beings might as well attempt a pre-emptive strike without trying to solve the problem.

- ❖ The reader should now be able to attempt Exercises 17.3–17.9 and Project 17.1.

17.3 Two-colour extensions

The notion of a Ramsey number has been generalised in many ways, and we describe one such generalisation in this section. For any two graphs G and H , the **generalised Ramsey number** $r = r(G, H)$ is the smallest integer r for which a red subgraph isomorphic to G or a blue subgraph isomorphic to H appears in *any* red-blue edge colouring of the complete graph K_r . By “red subgraph” we again mean a subgraph whose edges are all coloured red, and similarly for the phrase “blue subgraph.” Clearly, the special case $r(K_m, K_n)$ is the classical Ramsey number $r(m, n)$ in the notation of the previous section.

Gerencsér and Gyárfás [32] established the class of path Ramsey numbers

$$r(P_m, P_n) = n + \lfloor \frac{m}{2} \rfloor - 1 \quad (17.3)$$

for all $2 \leq m \leq n$ in 1969, where P_m denotes a path of order m , whilst Rosta [58] and Faudree and Schelp [31] independently showed that

$$r(C_m, C_n) = \begin{cases} 2n - 1, & 3 \leq m \leq n, m \text{ odd}, (m, n) \neq (3, 3), \\ n - 1 + \frac{m}{2}, & 4 \leq m \leq n, m \text{ and } n \text{ even}, (m, n) \neq (4, 4), \\ \max\{n - 1 + \frac{m}{2}, 2m - 1\}, & 4 \leq m < n, m \text{ even}, n \text{ odd}, \end{cases} \quad (17.4)$$

where C_m denotes a cycle of order m . Furthermore, Burr *et al.* [12] established the class of star Ramsey numbers

$$r(K_{1,m}, K_{1,n}) = \begin{cases} m + n - 1, & \text{if both } m \text{ and } n \text{ are even,} \\ m + n, & \text{otherwise,} \end{cases} \quad (17.5)$$

in 1983, while Bondy and Erdős [3] showed that

$$r(K_m, C_n) = (n - 1)(m - 1) + 1 \quad (17.6)$$

for all $n \geq m^2 - 2$ in 1973. Faudree and Schelp [31] noted that the result in (17.6) also holds for $n > 3 = m$, and it is conjectured that the formula in (17.6) holds for all $3 < m < n$.

We proceed with a beautiful theorem of Chvátal [18] dating back to 1977.

Theorem 17.8 *For any tree T_m of order m and any complete graph K_n of order n , $r(T_m, K_n) = (m-1)(n-1) + 1$.*

Proof To establish the lower bound $r(T_m, K_n) > (m-1)(n-1)$, consider a red-blue edge colouring of $K_{(m-1)(n-1)}$ where the red subgraph is $(n-1)K_{m-1}$, i.e. $n-1$ disjoint copies of K_{m-1} . Then each component in the red subgraph has order $m-1$, and so contains no tree of order m . On the other hand, the blue subgraph is an $(n-1)$ -partite graph, and so cannot contain K_n as subgraph.

To prove that $r(T_m, K_n) \leq (m-1)(n-1) + 1$, we proceed by induction on n . Clearly, $r(T_m, K_2) = m$ since, if all the edges of K_m are coloured red, then certainly this graph contains a red tree of order m as subgraph; otherwise at least one edge of K_m is coloured blue, in which case the graph contains a blue K_2 as subgraph. This establishes the base case for the induction process. Suppose that $n \geq 3$ and assume as induction hypothesis that $r(T_m, K_{n-1}) = (m-1)(n-2) + 1$. Consider a red-blue edge colouring of $K_{(m-1)(n-1)+1}$ and let v be a vertex of this complete graph.

Suppose v is incident with at least $(m-1)(n-2) + 1$ blue edges. Consider the red-blue edge subcolouring of the graph $K_{n'}$ induced by the vertices joined to v by means of blue edges. If this subcolouring contains a red tree of order m , then so too does the original edge colouring. If not, then since $n' \geq r(T_m, K_{n-1})$, the subcolouring contains a blue K_{n-1} by the induction hypothesis. This blue K_{n-1} , together with the vertex v and the blue edges joining v in $K_{n'}$, forms a blue K_n .

Now assume that every vertex is incident with at most $(m-1)(n-2)$ blue edges. Therefore, every vertex is incident with at least $(m-1)(n-1) - (m-1)(n-2) = m-1$ red edges and the subgraph induced by the red edges has minimum degree at least $m-1$. But then this red subgraph contains any tree of order m as a subgraph (see Exercise 17.10). In particular, there is a red T_m . ■

Apart from the five infinite classes of known, generalised Ramsey numbers mentioned above, quite a number of Ramsey numbers are known in two further classes, namely where the sought-after subgraphs are either bipartite or $K_m - e$ (a complete graph from which a single edge has been deleted). For bipartite graphs the diagonal upper bound

$$r(K_{2,m}, K_{2,m}) \leq 2m - 2 \quad (17.7)$$

was established for all $m \geq 2$ in 1994 by Exoo [29]. Some known diagonal Ramsey numbers for bipartite graphs appear in Table 17.2. Bounds for and exact values of some Ramsey numbers of the form $r(K_m - e, K_n)$ or $r(K_m - e, K_n - e)$ appear in Table 17.3.

m	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$r(K_{2,m}, K_{2,m})$	6	10	14	18	21	26	30	33	38	42	46	50	54	57	62

Table 17.2: Ramsey numbers of the form $r(K_{2,m}, K_{2,m})$ for all $m \in \{2, \dots, 16\}$.

Finally, generalised Ramsey numbers have also been established for a host of other subgraph classes, including various combinations of paths, cycles, stars,

$r(\downarrow, \rightarrow)$	$K_3 - e$	$K_4 - e$	$K_5 - e$	$K_6 - e$	$K_7 - e$
$K_3 - e$	3	5	7	9	11
K_3	5	7	11	17	21
$K_4 - e$	5	10	13	17	28
K_4	7	11	19	$30 : 33$	$37 : 52$
$K_5 - e$	7	13	22	$31 : 39$	$40 : 66$
K_5	9	16	$30 : 34$	$43 : 67$	≤ 112
$K_6 - e$	9	17	$31 : 39$	$45 : 70$	$59 : 135$
K_6	11	21	$37 : 53$	≤ 110	≤ 205
$K_7 - e$	11	28	$40 : 66$	$59 : 135$	≤ 251
K_7	13	$28 : 30$	$51 : 83$	≤ 193	≤ 392

Table 17.3: Bounds for and exact values of Ramsey numbers of the form $r(K_m - e, K_n)$ or $r(K_m - e, K_n - e)$ for $3 \leq m, n \leq 7$. Exact values are shown in bold face. Bounds are given in the form $\leq a$ or in the form lower bound : upper bound, where the latter bounds are inclusive.

trees, complete graphs, wheel graphs, book graphs, and many more. The interested reader may consult the dynamic survey paper by Radziszowski [56] in this regard.

- ❖ The reader should now be able to attempt [Exercise 17.10](#) and [Project 17.2](#).

17.4 The multi-colour Ramsey problem

The notion of a Ramsey number may also be generalised from the two-colour problem described in Sections 17.2–17.3 to a problem involving an arbitrary number of colours. We refer to an arbitrary edge colouring of the edges of a complete graph, using $k \in \mathbb{N}$ colours, in which the subgraph induced by colour i is H_i ($i \in [k]$) as a **k -edge colouring** (H_1, \dots, H_k) of the complete graph in question.

The (classical) **k -colour Ramsey number** $r = r(m_1, \dots, m_k)$ is defined as the smallest integer r such that $K_{m_i} \subset H_i$ for some $i \in [k]$ in an arbitrary k -edge colouring (H_1, \dots, H_k) of the complete graph K_r . If $k = 2$, then we recover the two-colour classical Ramsey number $r(m, n)$ of [Section 17.2](#) as special case.

More general multi-colour Ramsey numbers are also of interest. The **generalised k -colour Ramsey number** $r = r(G_1, \dots, G_k)$ is defined as the smallest integer r such that $G_i \subset H_i$ for some $i \in [k]$ in an arbitrary k -edge colouring (H_1, \dots, H_k) of the complete graph K_r . If $k = 2$, then we recover the generalised two-colour Ramsey number $r(G, H)$ of [Section 17.3](#) as special case.

Very little is known about k -colour Ramsey numbers when $k > 2$. For instance, the only classical multi-colour Ramsey number known exactly, is $r(3, 3, 3) = 17$, which was established by Greenwood and Gleason [36] in 1955. The 3-edge colouring of K_{16} in [Figure 17.4](#) (due to E Pegg) contains no monochromatic triangle, showing that $r(3, 3, 3) > 16$. We leave it as an exercise to show (analytically) that $r(3, 3, 3) \leq 17$.

Bounds on a number of other small multi-colour classical Ramsey numbers are shown in [Table 17.4](#).

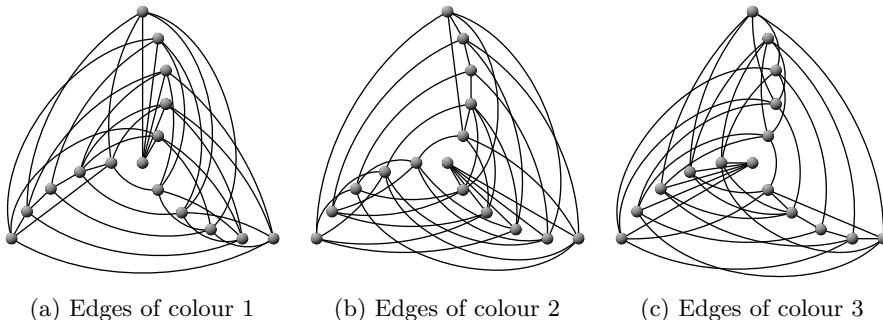


Figure 17.4: A 3-edge colouring of K_{16} , due to E Pegg, containing no monochromatic triangle, therefore showing that $r(3, 3, 3) > 16$.

$r(3, 3, 3)$	$r(3, 3, 4)$	$r(3, 3, 5)$	$r(3, 4, 4)$	$r(3, 4, 5)$	$r(4, 4, 4)$
17^a	30 : 31 ^b	45 : 57 ^{c,d}	55 : 79 ^d	89 : 161 ^e	128 : 236 ^f
$r(3, 3, 3, 3)$	$r(3, 3, 3, 4)$	$r(4, 4, 4, 4)$	$r(3, 3, 3, 3, 3)$	$r(3, 3, 3, 3, 3, 3)$	
52 : 64 ^{g,h}	87 : 155 ^e	$\geq 458^i$	162 : 317 ^{j,k}	$\geq 500^j$	

Table 17.4: Bounds on multi-colour Ramsey numbers, given in the form $\geq b$ or lower bound : upper bound, where the latter bounds are inclusive. Exact values are given in bold face. ^aDue to Greenwood and Gleason [36], ^bDue to Piwakowski and Radziszowski [55], ^cDue to Exoo [28], ^dDue to Kreher *et al.* [50], ^eDue to Exoo [30], ^fDue to Hill and Irving [47], ^gDue to Chung [17], ^hDue to Sanchez-Flores [60], ⁱDue to Wenlong [64], ^jDue to Exoo [29], ^kDue to Whitehead [65].

In addition, the four 3-colour cycle Ramsey numbers in Table 17.5 have also been found.

$r(C_4, C_4, C_4)$	$r(C_5, C_5, C_5)$	$r(C_6, C_6, C_6)$	$r(C_7, C_7, C_7)$
11^a	17^b	12^b	25^c

Table 17.5: Four 3-colour Ramsey numbers for cycles. ^aDue to Bialostocki and Schönhem [2], ^bDue to Yuansheng and Rowlinson [66, 67], ^cDue to Schelten *et al.* [61].

- ❖ The reader should now be able to attempt Exercise 17.11.

17.5 Multipartite Ramsey theory

Although indeed the pursuit in the classical Ramsey problem, there is no reason why we should seek monochromatic subgraphs in specifically *complete* graphs. In fact, there exists an extension to complete, balanced multipartite graphs of the classical Ramsey theory described in Sections 17.1–17.2 where one seeks monochro-

matic complete balanced multipartite graphs of specified dimensions in a complete balanced multipartite graph of specified structure. Since a complete balanced multipartite graph $K_{k \times j}$ has two dimensions, namely the number of partite sets k and the number of vertices j per partite set, two different kinds of multipartite Ramsey numbers arise, called the set multipartite Ramsey number and the size multipartite Ramsey number. In the former case we fix the number of vertices j per partite set and seek the smallest number of partite sets k of that size such that a red $K_{n \times \ell}$ or blue $K_{s \times t}$ necessarily appears as a subgraph in any red-blue colouring of $K_{k \times j}$. In the latter case we do the opposite, *i.e.* fix the number of partite sets k and seek the smallest number j of vertices per partite set such that a red $K_{n \times \ell}$ or blue $K_{s \times t}$ necessarily appears as a subgraph in any red-blue colouring of $K_{k \times j}$.

More precisely, if j, ℓ, n, s, t are natural numbers with $n, s \geq 2$, then the **set multipartite Ramsey number** $M_j(K_{n \times \ell}, K_{s \times t})$ is the smallest natural number k such that a red $K_{n \times \ell}$ or a blue $K_{s \times t}$ necessarily appears as a subgraph in an arbitrary red-blue edge colouring of $K_{k \times j}$. Similarly, for natural numbers k, ℓ, n, s, t with $n, s \geq 2$, the **size multipartite Ramsey number** $m_j(K_{n \times \ell}, K_{s \times t})$ is the smallest natural number j such that a red $K_{n \times \ell}$ or a blue $K_{s \times t}$ necessarily appears as a subgraph in an arbitrary red-blue edge colouring of $K_{k \times j}$. The following result ties the notions of set and size multipartite Ramsey numbers closely together.

Theorem 17.9 *If ℓ, t, n and s are natural numbers with $n, s \geq 2$, then*

- (i) $m_k(K_{n \times \ell}, K_{s \times t}) > j$ if and only if $M_j(K_{n \times \ell}, K_{s \times t}) > k$,
- (ii) $m_k(K_{n \times \ell}, K_{s \times t}) \leq j$ if and only if $M_j(K_{n \times \ell}, K_{s \times t}) \leq k$.

Proof (i) The inequality $M_j(K_{n \times \ell}, K_{s \times t}) > k$ holds if and only if there exists a red-blue edge colouring of $K_{k \times j}$ that contains neither a red $K_{n \times \ell}$ nor a blue $K_{s \times t}$ as subgraph, which may be restated as $m_k(K_{n \times \ell}, K_{s \times t}) > j$.

(ii) The result follows from (i) by a double contra-positive argument. ■

It is interesting that the set multipartite number $M_j(K_{n \times \ell}, K_{s \times t})$ exists for all $j, \ell, t \geq 1$ and all $n, s \geq 2$, but that this is not the case for the size multipartite number $m_j(K_{n \times \ell}, K_{s \times t})$, as stated in the following existence theorem, which is due to Burger and Van Vuuren [8, 9].

Theorem 17.10

- (i) *The set multipartite number $M_j(K_{n \times \ell}, K_{s \times t})$ exists and, in fact, $r(n, s) \leq M_j(K_{n \times \ell}, K_{s \times t}) \leq \binom{n\ell+st-2}{n\ell-1}$ for all $j, \ell, t \geq 1$ and all $n, s \geq 2$.*
- (ii) *The size multipartite number $m_j(K_{n \times \ell}, K_{s \times t})$ exists for all $\ell, t \geq 1$ and all $n, s \geq 2$ if and only if $j \geq r(n, s)$.*

In order to prove part (i) of this theorem we require the notion of an expansive red-blue edge colouring. A red-blue edge colouring of $K_{k \times j}$ is called **expansive** if, for every pair of partite sets of $K_{k \times j}$, the edges between all vertices in these partite sets have the same colour. Therefore, every expansive red-blue edge colouring of $K_{k \times j}$ corresponds to exactly one red-blue edge colouring of K_k (this may be seen by contracting each partite set of $K_{k \times j}$ to a single vertex), and we say that the expansive red-blue edge colouring of $K_{k \times j}$ is **induced** by the corresponding red-blue edge colouring of K_k (this definition is due to Day *et al.* [24]).

Proof of Theorem 17.10(i) Let $w = r(n, s) \geq 2$. By the definition of w , there exists a red-blue edge colouring of K_{w-1} that contains neither a red K_n nor a blue K_s as subgraph. The expansive colouring of $K_{(w-1) \times j}$ induced by this red-blue edge colouring therefore contains neither a red K_n nor a blue K_s as subgraph. But $K_n = K_{n \times 1} \subset K_{n \times \ell}$ and $K_s = K_{s \times 1} \subset K_{s \times t}$ for any $\ell, t \geq 1$, and so the expansive colouring of $K_{(w-1) \times j}$ also contains neither a red $K_{n \times \ell}$ nor a blue $K_{s \times t}$ as subgraph. We conclude that $M_j(K_{n \times \ell}, K_{s \times t}) > w - 1$.

Finally, consider the upper bound. It follows from [Theorem 17.5](#) that $r(n\ell, st) \leq \binom{n\ell+st-2}{n\ell-1} = u$, say. Hence, in an arbitrary red-blue edge colouring of K_u , a red $K_{n\ell}$ or a blue K_{st} is forced as subgraph. But since $K_{n \times \ell} \subset K_{n\ell}$, $K_{s \times t} \subset K_{st}$ and $K_u = K_{u \times 1} \subset K_{u \times j}$ for any $j \geq 1$, it follows that $K_{u \times j}$ necessarily contains a red $K_{n \times \ell}$ or a blue $K_{s \times t}$ as subgraph. ■

We refrain from the proof of [Theorem 17.10\(ii\)](#), because the proof is rather technical, involving an algorithm. The interested reader may, however, find the proof in [9]. The following growth properties hold for multipartite Ramsey numbers. We leave the proof of the theorem as an exercise.

Theorem 17.11 (Growth properties) *If $n, s, a, c \geq 2$ and j, k, ℓ, t, b, d are natural numbers, then*

- (i) $M_j(K_{n \times \ell}, K_{s \times t}) \leq M_j(K_{a \times b}, K_{c \times d})$ if $n \leq a$, $\ell \leq b$, $s \leq c$ and $t \leq d$. Strict inequality holds if at least one of the inequalities $n < a$ or $s < c$ holds,
- (ii) $m_j(K_{n \times \ell}, K_{s \times t}) \leq m_j(K_{a \times b}, K_{c \times d})$ if $n \leq a$, $\ell \leq b$, $s \leq c$ and $t \leq d$ (when both size multipartite Ramsey numbers exist),
- (iii) $M_j(K_{n \times \ell}, K_{s \times t}) \leq M_k(K_{n \times \ell}, K_{s \times t})$ if $k \leq j$,
- (iv) $m_j(K_{n \times \ell}, K_{s \times t}) \leq m_k(K_{n \times \ell}, K_{s \times t})$ if $k \leq j$ (when both size multipartite Ramsey numbers exist).

As in the case of the classical Ramsey numbers, only a handful of set multipartite Ramsey numbers are known. These are $M_1(K_{2 \times 2}, K_{3 \times 3}) = 7$ due to [Chartrand and Schuster](#) [14], $M_1(K_{2 \times 2}, K_{4 \times 1}) = 10$ due to [Chvátal and Harary](#) [19], $M_2(K_{2 \times 2}, K_{3 \times 1}) = 4$ and $M_2(K_{2 \times 2}, K_{4 \times 1}) = 7$ due to [Harborth and Mengersen](#) [41, 42], $M_1(K_{2 \times 2}, K_{5 \times 1}) = 14$ due to [Greenwood and Gleason](#) [36], $M_1(K_{2 \times 2}, K_{6 \times 1}) = 18$ due to [Exoo](#) [28], $M_1(K_{2 \times 3}, K_{2 \times 3}) = 18$ due to [Harborth and Mengersen](#) [40] and the complete classes of $(K_{2 \times 2}, K_{3 \times 1})$ and $(K_{2 \times 2}, K_{2 \times 2})$ multipartite Ramsey numbers listed in [Table 17.6](#). In addition, the only size multipartite Ramsey numbers known exactly are $m_2(K_{2 \times 2}, K_{2 \times 3}) = 9$ and $m_2(K_{2 \times 2}, K_{2 \times 4}) = 14$ due to [Hattingh and Henning](#) [44], $m_2(K_{2 \times 3}, K_{2 \times 3}) = 17$ due to [Beineke and Schwenk](#) [1] and $m_2(K_{2 \times 4}, K_{2 \times 4}) = 48$ due to [Irving](#) [48].

It may be observed in [Table 17.6](#) that the sequence $(M_j(K_{2 \times 2}, K_{s \times t}))_{j=1,2,3,\dots}$ of set multipartite Ramsey converges to the Ramsey number $r(2, s)$, whilst the sequence $(m_j(K_{2 \times 2}, K_{s \times t}))_{j=1,2,3,\dots}$ of size multipartite Ramsey converges to 1 for $(s, t) = (3, 1)$ or $(s, t) = (2, 2)$. This is a manifestation of a more general phenomenon, as stated in the following 2004 asymptotic result due to [Burger and Van Vuuren](#) [8, 9].

Theorem 17.12 (Asymptotic limits as $j \rightarrow \infty$) *For any $\ell, t \geq 1$ and $n, s \geq 2$,*

- (i) $m_j(K_{n \times \ell}, K_{s \times t}) \rightarrow 1$ as $j \rightarrow \infty$,
- (ii) $M_j(K_{n \times \ell}, K_{s \times t}) \rightarrow r(n, s)$ as $j \rightarrow \infty$.

Frank Harary was born in New York on 11 March 1921. He obtained bachelor's and master's degrees from Brooklyn College in 1941 and 1945, respectively, and a doctorate from University of California at Berkeley in 1948. Thereafter, he held a teaching and research position at the University of Michigan, first as assistant professor (from 1953), then as associate professor (from 1959) and finally as professor of mathematics (from 1964 until his retirement in 1986). After his retirement he joined the Department of Computer Science at New Mexico State University in Las Cruces, where he remained until his death on 4 January 2005. Together with his many doctoral students, he helped to standardise the terminology of graph theory by writing more than half a dozen books on the subject. One of these books was John Wiley's first eBook, *Graph Theory and Geography*. His research interests within graph theory were wide, and included graph enumeration, signed graphs (a branch of graph theory that he had invented) and Ramsey theory. He was one of the founding editors of the influential *Journal of Combinatorial Theory* and *Journal of Graph Theory*. He co-authored more than 700 journal papers together with more than 300 collaborators. He also travelled and lectured extensively — and was particularly proud that he had lectured in cities around the world beginning with every letter of the alphabet.



Biographic note 95: Frank Harary (1921–2005)

Proof (i) It follows from [Theorem 17.11\(iv\)](#) that the sequence $m_j(K_{n \times \ell}, K_{s \times t})$ is nonincreasing for increasing j and any fixed values of $n, s \geq 2$ and $\ell, t \geq 1$. We therefore only need to show that there exists a natural number k for which $m_k(K_{n \times \ell}, K_{s \times t}) = 1$. It is clear that $k = r(n\ell, st)$ is such a number, since every red-blue edge colouring of $K_k = K_{k \times 1}$ contains a red $K_{n\ell}$ as subgraph (in which case it also contains a red $K_{n \times \ell}$ as subgraph) or a blue K_{st} as subgraph (in which case it also contains a blue $K_{s \times t}$ as subgraph).

(ii) Proof of this part of the theorem is left as an exercise. ■

The multipartite size Ramsey number $m_2(K_{2 \times m}, K_{2 \times n})$ has been studied so extensively in the literature that we close this section by making special mention of this kind of Ramsey number, adopting the abbreviated notation $b(m, n)$ for this number and calling it a **bipartite Ramsey number**. Since $r(2, 2) = 2$ by [Theorem 17.2](#), it follows from [Theorem 17.10\(ii\)](#) that the bipartite Ramsey number $b(m, n)$ exists for all $m, n \in \mathbb{N}$. The following recursive upper bound may be shown to hold for bipartite Ramsey numbers (a step-by-step guide towards proving this result may be found in [Project 17.3](#)). This bound is similar to the recursive bound in [Theorem 17.3](#) for classical Ramsey numbers.

Theorem 17.13 *For all integers $m, n \geq 2$, $b(m, n) \leq b(m - 1, n) + b(m, n - 1)$.*

The bound in [Theorem 17.13](#) may, in turn, be used to establish the following closed-form upper bound on $b(m, n)$ for all $m, n \in \mathbb{N}$.

$j \rightarrow$	1	2	3	4	5	6	≥ 7
$M_j(K_{2 \times 2}, K_{3 \times 1})$	7^a	4^b	3^c	3^c	3^c	3^c	3^c
$m_j(K_{2 \times 2}, K_{3 \times 1})$	∞^d	∞^d	3^c	2^c	2^c	2^c	1^c
$M_j(K_{2 \times 2}, K_{2 \times 2})$	6^e	4^f	4^f	3^f	2^f	2^f	2^f
$m_j(K_{2 \times 2}, K_{2 \times 2})$	∞^d	5^g	3^g	2^g	2^g	1^g	1^g

Table 17.6: Two small classes of multipartite Ramsey numbers. ^aDue to Chartrand and Schuster [14]. ^bDue to Harborth and Mengersen [41, 42]. ^cDue to Burger and Van Vuuren [8]. ^dBy Theorem 17.10. ^eDue to Chvátal and Harary [19]. ^fDue to Burger *et al.* [6]. ^gDue to Day *et al.* [24].

Theorem 17.14 For all integers $m, n \geq 2$,

$$b(m, n) \leq \binom{m+n}{m} - 1.$$

The proof of [Theorem 17.14](#) is similar to that of [Theorem 17.10](#), and is also considered in [Project 17.3](#). Using the bounds in [Theorems 17.13](#) and [17.14](#), the following small bipartite Ramsey numbers may be established.

Theorem 17.15

- (i) $b(2, 2) = 5$,
- (ii) $b(2, 3) = 9$,
- (iii) $b(2, 4) = 14$.

Proof We prove part (ii) of the theorem, leaving the proofs of parts (i) and (iii) as exercises. It follows from [Theorem 17.14](#) that $b(2, 3) \leq \binom{2+3}{2} - 1 = 9$. Now consider the red-blue edge colouring of $K_{8,8}$ with red edges as shown in [Figure 17.5\(a\)](#) and blue edges as shown in [Figure 17.5\(b\)](#). This graph contains neither a red $K_{2,2}$ nor a blue $K_{3,3}$ as subgraph, showing that $b(2, 3) > 8$. ■

Apart from the bipartite Ramsey numbers in [Theorem 17.15](#), only two other bipartite Ramsey numbers are known exactly. These numbers are shown in [Table 17.7](#) together with upper bounds on a number of other bipartite Ramsey numbers.

$n \rightarrow$ $\downarrow m$	2	3	4	5	6
2	5^a	9^b	14^b	$\leq 19^c$	$\leq 25^c$
3		17^a	$\leq 29^c$	$\leq 41^c$	$\leq 56^c$
4			48^d	$\leq 72^c$	$\leq 101^c$
5				$\leq 115^c$	$\leq 168^c$

Table 17.7: Bounds for and exact values of bipartite Ramsey numbers $b(m, n)$. Exact values are shown in bold face. ^aDue to Beineke and Schwenk [1], ^bDue to Hattingh and Henning [43], ^cDue to Goddard *et al.* [33], ^dDue to Irving [48].

Finally, asymptotic lower and upper bounds have been established for bipartite Ramsey numbers.

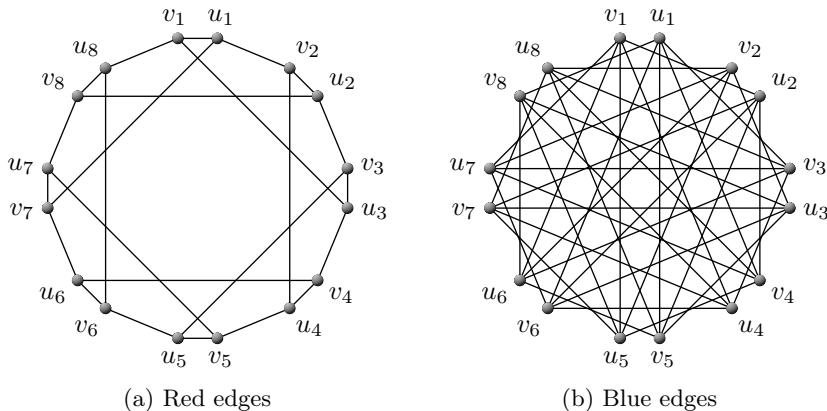


Figure 17.5: A red-blue edge colouring of $K_{8,8}$ showing that $b(2,3) > 8$. The vertices labelled u_1, \dots, u_8 form one partite set and the vertices labelled v_1, \dots, v_8 form the other partite set.

Theorem 17.16

- (i) $b(m, m) < 2^{m-1}(m-1)$ for all $m \geq 21$.
(ii) $b(m, m) > m\sqrt{2^{m+1}}/e$ for sufficiently large values of m .

Part (i) of Theorem 17.16 is due to Irving [48], whilst part (ii) is due to Hattingh and Henning [43].

- ❖ The reader should now be able to attempt Exercises 17.12–17.13 and Project 17.3.

17.6 Requirements other than that of a subgraph

Recall, from Sections 17.1–17.2, that the classical Ramsey number $r = r(m, n)$ was defined as the smallest natural number r such that in any red-blue edge colouring (R, B) of the complete graph K_r of order r , it holds that $K_m \subset R$ or $K_n \subset B$ (or perhaps both). If $K_m \subset R$, then it is clear that there is an independent set of order m in B , and hence that $\alpha(B) \geq m$. Similarly, $K_n \subset B$ implies that $\alpha(R) \geq n$. The classical Ramsey number $r(m, n)$ may therefore be redefined as the smallest natural number r such that in any red-blue edge colouring (R, B) of the complete graph K_r of order r , it holds that $\alpha(B) \geq m$ or $\alpha(R) \geq n$ (or perhaps both). For this reason the classical Ramsey number $r(m, n)$ is often also referred to as an **independent Ramsey number**. Recalling from Chapter 16 the upper domination-related parameters

$$\alpha(G) < \Gamma(G) < \text{IR}(G) \quad (17.8)$$

for any graph G , it is natural to generalise the definition of an independent Ramsey number in the following five ways:

The **irredundant Ramsey number** $s = s(m, n)$ is the smallest natural number s such that in any red-blue edge colouring (R, B) of the complete graph K_s of order s , it holds that $\text{IR}(B) \geq m$ or $\text{IR}(R) \geq n$ (or perhaps both). This definition dates from 1989 and is due to Brewster *et al.* [4].

Johannes Hendrik Hattingh was born in Johannesburg, South Africa on 27 May 1962. He obtained a doctorate in mathematics from the University of Johannesburg and he also holds an honours degree in computer science from the same institution. He accepted a lectureship position in the Mathematics Department of the University of Pretoria in 1988 where he remained for a year, after which he relocated to the University of Johannesburg in 1989. After being promoted to associate professor, he left that institution in 1998 to join the faculty of the Department of Mathematics at Georgia State University in the United States of America. There he remained until 2010, after being promoted to professor and having served a term as head of the department. He is currently professor of mathematics and head of the Mathematics Department at East Carolina University. His research interests include total domination, restrained domination, Ramsey theory and various topics in graph colouring. He has supervised a large number of master's and doctoral students.



Biographic note 96: Johannes Hattingh (1962–present)

The **mixed irredundant Ramsey number** $t = t(m, n)$ is the smallest natural number t such that in any red-blue edge colouring (R, B) of the complete graph K_t of order t , it holds that $\text{IR}(B) \geq m$ or $\alpha(R) \geq n$ (or perhaps both). This definition dates from 1990 and is due to [Cockayne et al.](#) [21].

The **upper domination Ramsey number** $u = u(m, n)$ is the smallest natural number u such that in any red-blue edge colouring (R, B) of the complete graph K_u of order u , it holds that $\Gamma(B) \geq m$ or $\Gamma(R) \geq n$ (or perhaps both). This definition dates from the early 1990s and is due to [Oellermann and Shreve](#) [54].

The **mixed domination Ramsey number** $v = v(m, n)$ is the smallest natural number v such that in any red-blue edge colouring (R, B) of the complete graph K_v of order v , it holds that $\Gamma(B) \geq m$ or $\alpha(R) \geq n$ (or perhaps both). This definition is also due to [Oellermann and Shreve](#) [54].

The **irredundant-domination Ramsey number** $w = w(m, n)$ is the smallest natural number w such that in any red-blue edge colouring (R, B) of the complete graph K_w of order w , it holds that $\text{IR}(B) \geq m$ or $\Gamma(R) \geq n$ (or perhaps both). This definition dates from 2014 and is due to [Burger et al.](#) [7].

It is, of course, also possible to generalise the definitions of the above five non-classical Ramsey numbers in a manner similar to that described in [Section 17.5](#) so as to accommodate more than two colours. The Ramsey numbers above are related to each other and to the classical Ramsey number $r(m, n)$ as described in the inequality chains

$$s(m, n) \leq w(m, n) \leq \begin{cases} u(m, n) \\ t(m, n) \end{cases} \leq v(m, n) \leq r(m, n), \quad (17.9)$$

which are easily shown to hold for all $m, n \geq 2$; this is left as an exercise. It is unknown in what way (if any) the Ramsey numbers $u(m, n)$ and $t(m, n)$ are related. The recursive upper bounds

$$s(m, n) \leq s(m - 1, n) + s(m, n - 1), \quad (17.10)$$

$$t(m, n) \leq t(m - 1, n) + t(m, n - 1), \quad (17.11)$$

$$u(m, n) \leq u(m - 1, n) + u(m, n - 1), \quad (17.12)$$

$$v(m, n) \leq v(m - 1, n) + v(m, n - 1), \text{ and} \quad (17.13)$$

$$w(m, n) \leq w(m - 1, n) + w(m, n - 1) \quad (17.14)$$

hold similar to the result of [Theorem 17.3](#); proof of these bounds are again left as exercise. Note, however, that in the case of the mixed Ramsey numbers $t(m, n)$, $v(m, n)$ and $w(m, n)$, the arguments m and n do not necessarily commute, i.e. $t(m, n) \neq t(n, m)$, $v(m, n) \neq v(n, m)$ and $w(m, n) \neq w(n, m)$ in general.

Exact values are known for only a handful of the five nonclassical Ramsey numbers considered in this section. These numbers appear in [Table 17.8](#).

$s(m, n)$	$w(m, n)$	$u(m, n)$	$t(m, n)$	$v(m, n)$	$r(m, n)$
$s(3, 3) = 6^a$	$w(3, 3) = 6^b$	$u(3, 3) = 6^e$	$t(3, 3) = 6^d$	$v(3, 3) = 6^e$	$r(3, 3) = 6^f$
$s(3, 4) = 8^a$	$w(3, 4) = 8^b$	$u(3, 4) = 8^e$	$t(3, 4) = 9^d$	$v(3, 4) = 9^e$	$r(3, 4) = 9^f$
$s(3, 5) = 12^a$	$w(4, 3) = 8^b$	$u(3, 5) = 12^e$	$t(4, 3) = 8^d$	$v(4, 3) = 8^b$	$r(3, 5) = 14^f$
$s(3, 6) = 15^g$	$w(3, 5) = 12^b$	$u(3, 6) = 15^e$	$t(3, 5) = 12^d$	$v(3, 5) = 12^e$	$r(3, 6) = 18^i$
$s(3, 7) = 18^j$	$w(5, 3) = 12^b$	$u(4, 4) = 13^c$	$t(5, 3) = 13^d$	$v(5, 3) = 13^d$	$r(3, 7) = 23^k$
$s(3, 8) = 21^\ell$	$w(3, 6) = 15^b$		$t(3, 6) = 15^h$	$v(3, 6) = 15^e$	$r(3, 8) = 28^m$
$s(4, 4) = 13^n$	$w(6, 3) = 15^b$		$t(6, 3) = 15^b$	$v(6, 3) = 15^b$	$r(3, 9) = 36^o$
	$w(3, 7) = 18^p$		$t(3, 7) = 18^p$	$v(4, 4) = 15^b$	$r(4, 4) = 18^f$
	$w(3, 8) = 21^e$		$t(3, 8) = 22^p$		$r(4, 5) = 25^q$

Table 17.8: Known nonclassical Ramsey numbers. ^aDue to [Brewster et al.](#) [4], ^bDue to [Burger and Van Vuuren](#) [10], ^cDue to [Dzido and Zakrzewska](#) [25], ^dDue to [Cockayne et al.](#) [21], ^eDue to [Henning and Oellermann](#) [46], ^fDue to [Greenwood and Gleason](#) [36], ^gDue to [Brewster et al.](#) [5], ^hDue to [Grobler](#) [38], ⁱDue to [Graver and Yackel](#) [35], ^jDue to [Cockayne et al.](#) [22], ^kDue to [Kalbfleisch](#) [49], ^lDue to [Burger and Van Vuuren](#) [11], ^mDue to [McKay and Min](#) [51], ⁿDue to [Cockayne et al.](#) [20], ^oDue to [Grinstead and Roberts](#) [37], ^pDue to [Burger et al.](#) [7], ^qDue to [McKay and Radziszowski](#) [52].

It is possible to use the probabilistic method to establish various bounds on the Ramsey numbers described in this section. For example, [Chen et al.](#) [15] showed in 1993 that $s(n, n) > \sqrt{n2^n/3}$ for sufficiently large values of n , while [Rousseau and Speed](#) [59] proved in 2003 that $t(3, n) \leq 5\sqrt{n^3/\log n}$ for sufficiently large values of n .

We close this section by remarking that various Ramsey numbers, other than the five nonclassical parameters considered in this section, have also been defined and studied in the literature.

- ❖ The reader should now be able to attempt Exercises [17.15–17.18](#).

Exercises

- 17.1 Is there a graph of order 5 which is not a 5-cycle such that neither the graph itself nor its complement contains K_3 as subgraph (*i.e.* is the counter example showing that $r(3, 3) > 5$ in Figures 17.1(b)–(c) unique)? If so, explain why. If not, produce an alternative counter example.
- 17.2 (a) Use a coin to produce ten random graphs of order 6, and then add together, for each case, the number of triangles in the graph and the number of triangles in its complement. What do you notice about these sums?
 (b) Because of the Ramsey number $r(3, 3) = 6$, your sum in (a) should in each case be at least 1. Verify that, in fact, your sum in (a) is at least 2 in each case — a result established by Goodman [34] in 1959!
- 17.3 Use a coin to produce a random red-blue edge colouring of the complete graph K_9 . Verify the result of Theorem 17.4(i) for this special case, namely that there is a red K_3 or a blue K_4 as subgraph in your edge colouring.
- 17.4 Complete the proof of Theorem 17.4(ii) by using the result of Theorem 17.3 to show that $r(3, 5) \leq 14$, and by producing a red-blue edge colouring of K_{13} in which there is no red K_3 as subgraph and no blue K_5 as subgraph, thereby showing that $r(3, 5) > 13$. (Hint: Consider colouring the edges red according to a circulant graph structure, and similarly for the blue edges, as was done in the proof of Theorem 17.4(i).)
- 17.5 Establish the validity of Pascal's identity (17.2).
- 17.6 (a) Which of the two bounds in Theorems 17.3 and 17.10 is the better bound?
 (b) For which of the known Ramsey numbers in Table 17.1 is the bound in Theorem 17.3 sharp?
 (c) For which of the known Ramsey numbers in Table 17.1 is the bound in Theorem 17.10 sharp?
- 17.7 Prove that $r(3, n) \leq n(n + 1)/2$ for every integer $n \geq 2$.
- 17.8 Prove Theorem 17.7.
- 17.9 Use the result of Theorem 17.7 to generalise the result of Theorem 17.6 so as to obtain a lower bound on the Ramsey number $r(m, n)$ for $m \neq n$.
- 17.10 Prove that if T is a tree of order m and G is a graph with minimum degree $\delta(G) \geq m - 1$, then T is a subgraph of G .
- 17.11 The 3-edge colouring in Figure 17.4 shows that $r(3, 3, 3) > 16$. Use the fact that $r(3, 3) = 6$ to prove that $r(3, 3, 3) \leq 17$.
- 17.12 Prove Theorem 17.11.
- 17.13 Prove Theorem 17.12(ii).
- 17.14 Use a coin to produce a random red-blue edge colouring of the complete bipartite graph $K_{5,5}$. Verify the result of Theorem 17.15(i) for this special case, namely that there is at least one monochromatic $K_{2,2}$ (cycle of length 4) as subgraph in your edge colouring.

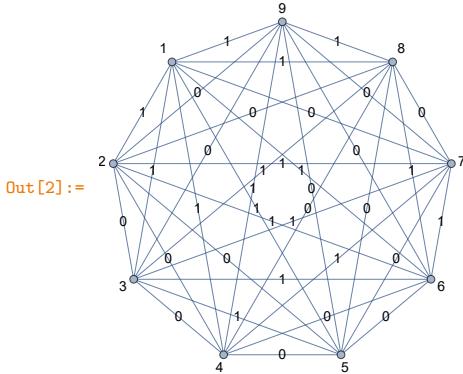
- 17.15 Use the inequality chain (17.8) to establish the validity of the inequality chains (17.9).
- 17.16 Prove the recursive upper bounds (17.10)–(17.14).
- 17.17 Prove that $s(2, n) = t(2, n) = t(n, 2) = u(2, n) = v(2, n) = v(n, 2) = n$ for all $n \geq 2$.
- 17.18 Prove that $s(3, 3) = t(3, 3) = u(3, 3) = v(3, 3) = 6$.

Computer exercises

In this section we show how to produce random red-blue edge colourings of complete graphs and to verify the property of a Ramsey number by seeking out one of the two guaranteed monochromatic subgraphs. Recall, from Chapter 4, that the **MATHEMATICA** commands

```
In[1]:= r = 9;
In[2]:= K = CompleteGraph[r,
  EdgeWeight -> RandomInteger[{0, 1}, r (r - 1)/2],
  VertexLabels -> "Name",
  EdgeLabels -> "EdgeWeight"]
```

may be used to produce a graphical representation of the labelled graph K_9 in the edges have randomly been weighted zero or one such as in the following case:



Furthermore, the commands

```
In[3]:= edges0 = EdgeList[K, _?(PropertyValue[{K, #}, EdgeWeight] == 0 &)]
In[4]:= edges1 = EdgeList[K, _?(PropertyValue[{K, #}, EdgeWeight] == 1 &)]
```

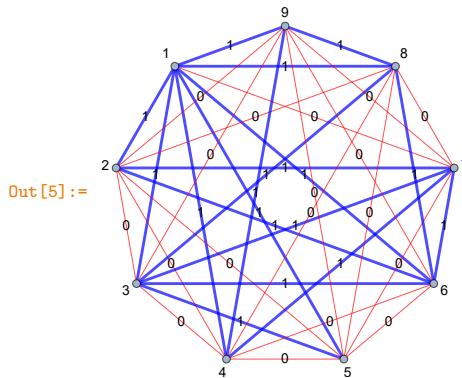
yield the sets of edges

```
Out[3]:= {1 ↔ 7, 2 ↔ 3, 2 ↔ 4, 2 ↔ 5, 2 ↔ 8, 2 ↔ 9, 3 ↔ 4, 3 ↔ 9, 4 ↔ 5, 4 ↔ 6, 4 ↔ 8, 5 ↔ 6, 5 ↔ 7, 5 ↔ 8, 5 ↔ 9, 6 ↔ 9, 7 ↔ 8, 7 ↔ 9}
Out[4]:= {1 ↔ 2, 1 ↔ 3, 1 ↔ 4, 1 ↔ 5, 1 ↔ 6, 1 ↔ 8, 1 ↔ 9, 2 ↔ 6, 2 ↔ 7, 3 ↔ 5, 3 ↔ 6, 3 ↔ 7, 3 ↔ 8, 4 ↔ 7, 4 ↔ 9, 6 ↔ 7, 6 ↔ 8, 8 ↔ 9}
```

weighted zero and one, respectively. These edge subsets may be coloured red and blue, respectively, in a graphical representation of the labelled graph by executing the command

```
In[5]:= HighlightGraph[K, {Style[edges0, Red], Style[edges1, Blue]}]
```

to yield the output:



The subgraphs induced by respectively the red and blue edges above may be captured by means of the commands

```
In[6]:= redgraph = Graph[edges0];
In[7]:= bluegraph = Graph[edges1];
```

Since the Ramsey number $r(4, 3)$ is 9, it follows that the random red-blue edge colouring above necessarily contains a red clique of order 4 or a blue triangle (or perhaps both). Instances of the largest cliques present in the red and blue subgraphs captured above may be found by executing the commands

```
In[8]:= FindClique[redgraph]
In[9]:= FindClique[bluegraph]
```

which yield the output

```
Out[8]:= {{2, 4, 5, 8}}
Out[9]:= {{1, 3, 6, 8}}
```

confirming that more than the above requirement is indeed met because there is clearly both a red clique of order 4 on the verices $\{2, 4, 5, 8\}$ and a blue triangle on any vertex triple in the set $\{1, 3, 6, 8\}$. Instances of red and blue cliques of the required orders can alternatively be found by means of the commands

```
In[10]:= n = 4; (* red *)
In[11]:= m = 3; (* blue *)
In[12]:= FindClique[redgraph, {n}]
In[12]:= FindClique[bluegraph, {m}]
```

which yield

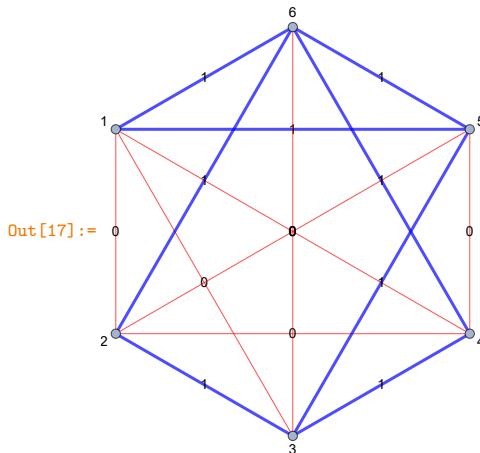
```
Out[11]:= {{2, 4, 5, 8}}
Out[12]:= {{1, 3, 5}}}
```

as output. The following commands may therefore be executed to return a red clique of order n (if it exists) and a blue clique of order m (if it exists) in a random bi-colouring of the edges of the complete graph K_r :

```
In[13]:= r = 6; (* large graph *)
In[14]:= n = 4; (* red *)
In[14]:= m = 3; (* blue *)
In[14]:= K = CompleteGraph[r,
  EdgeWeight -> RandomInteger[{0, 1}, r (r - 1)/2],
  VertexLabels -> "Name",
  EdgeLabels -> "EdgeWeight"];
In[15]:= edges0 = EdgeList[K, _?(PropertyValue[{K, #}, EdgeWeight] == 0 &)];
In[16]:= edges1 = EdgeList[K, _?(PropertyValue[{K, #}, EdgeWeight] == 1 &)];
In[17]:= HighlightGraph[K, {Style[edges0, Red], Style[edges1, Blue]}]
```

```
In[18]:= redgraph = Graph[edges0];
In[19]:= bluegraph = Graph[edges1];
In[20]:= Print["red clique of size ", n, " : ", FindClique[redgraph, {n}]];
In[21]:= Print["blue clique of size ", m, " : ", FindClique[bluegraph, {m}]];
```

These commands yield



```
Out[17]:= 0
Out[20]:= red clique of size 4 : {}
Out[21]:= blue clique of size 3 : {{1,5,6}}
```

as output.

Projects

This section contains three projects in which the reader is provided with step-by-step guidance on how to establish exact values for or bounds on known Ramsey numbers.

Project 17.1: The Ramsey number $r(4,4)$

The value of the Ramsey number $r(4,4)$ is listed as 18 in Table 17.1 — a 1955 result due to Greenwood and Gleason [36]. The objective in this project is to guide the reader towards proving this result.

Tasks

1. In order to show that $r(4,4) > 17$, it is necessary to produce a red-blue edge colouring of K_{17} in which no monochromatic subgraph is isomorphic to K_4 . Launch a search for such a red-blue edge colouring, bearing in mind that lower bounds on Ramsey numbers are traditionally achieved by means of symmetric edge colourings according to circulant graph structures. (Hint: Because of the symmetry of the arguments (4 and 4) of the Ramsey number in question, it is expected that the densities of the red and blue subgraphs in a suitable lower bound red-blue edge colouring should be approximately equal, *i.e.* the number of red edges and the number of blue edges should be approximately the same.)

2. The recursive upper bound of [Theorem 17.3](#) yields $r(4, 4) \leq 2r(3, 4) = 2 \times 9 = 18$, which establishes the Ramsey number $r(4, 4) = 18$ in view of your answer in [Task 1](#) above. Let us see, however, whether we can establish this upper bound directly via an analytical argument, without use of [Theorem 17.3](#). Construct such an argument (by contradiction) according to the following steps, by supposing that there exists a red-blue edge colouring Ξ of K_{18} containing neither a red nor a blue K_4 as subgraph:
- Show that the number of blue edges incident with every vertex in Ξ is at most 8. (Hint: Assume, to the contrary, that there is a vertex v of Ξ which is incident with 9 blue edges and apply the fact that $r(3, 4) = 9$ to the red-blue edge subcolouring induced by the neighbourhood of v .)
 - Repeat the argument in [Task 2\(a\)](#) to show that the number of red edges incident with every vertex in Ξ is also at most 8.
 - Do the results of Tasks [2\(a\)](#) and [\(b\)](#) lead to a contradiction? Explain.

Project 17.2: The star Ramsey number $r(K_{1,m}, K_{1,n})$

The value of the generalised Ramsey number $r(K_{1,m}, K_{1,n})$ is listed in [\(17.5\)](#) as $m+n-1$ if both m and n are even, or $m+n$ otherwise — a 1983 result due to [Burr et al.](#) [12]. The objective in this project is to guide the reader towards proving this result.

Tasks

- Consider first the case where both m and n are even. Let Ω be an arbitrary red-blue edge colouring of K_{m+n-1} .
 - Show, by contradiction, that every vertex in Ω is incident with at least $m-1$ red edges or with at least $n-1$ blue edges.
 - Use the result of [Corollary 1.2](#) to show that it is impossible for *every* vertex in Ω to be incident with exactly $m-1$ red edges and with exactly $n-1$ blue edges.
 - Explain how the results in Tasks [1\(a\)](#) and [\(b\)](#) above lead us to the conclusion that $r(K_{1,m}, K_{1,n}) \leq m+n-1$ if both m and n are even.
 - Use the properties of circulant graphs in Exercises [1.12](#) and [1.13](#) to find a red-blue edge colouring of K_{m+n-2} in which every vertex is incident with at most $m-1$ red edges and with at most $n-1$ blue edges, showing that $r(K_{1,m}, K_{1,n}) > m+n-2$ if both m and n are even.
- Now consider the case where at least one of m or n is odd.
 - Let Ω' be an arbitrary red-blue edge colouring of K_{m+n} . Show, by contradiction, that every vertex in Ω' is incident with at least m red edges or with at least n blue edges, yielding the bound $r(K_{1,m}, K_{1,n}) \leq m+n$ if at least one of m or n are odd.
 - Use the properties of circulant graphs in Exercises [1.12](#) and [1.13](#) to find a red-blue edge colouring of K_{m+n-1} in which every vertex is incident with exactly $m-1$ red edges and with exactly $n-1$ blue edges, showing that $r(K_{1,m}, K_{1,n}) > m+n-1$ if at least one of m and n are odd.

- (c) Why is a red-blue edge colouring of K_{m+n-1} as described in [Task 2\(b\)](#) above not possible if both m and n are even?

Project 17.3: Upper bounds for bipartite Ramsey numbers

The objective in this final project is to guide the reader towards proving Theorems [17.13](#) and [17.14](#) which give respectively recursive and closed-form upper bounds on the bipartite Ramsey number $b(m, n)$.

Tasks

1. (a) Let $b = b(m - 1, n) + b(m, n - 1) + 1$. Prove that if every vertex in a red-blue edge colouring of the bipartite graph $K_{b,b}$ is incident with at least $b(m - 1, n) + 1$ red edges, then a red $K_{m,m}$ or a blue $K_{n,n}$ occurs as subgraph in the edge colouring.
- (b) Use a similar argument as in [Task 1\(a\)](#) above to prove that if every vertex in a red-blue edge colouring of the bipartite graph $K_{b,b}$ is incident with at least $b(m, n - 1) + 1$ blue edges, where again $b = b(m - 1, n) + b(m, n - 1) + 1$, then a red $K_{m,m}$ or a blue $K_{n,n}$ occurs as subgraph in the edge colouring.
- (c) Now suppose that every vertex in a red-blue edge colouring of the bipartite graph $K_{b,b}$ is incident with at most $b(m - 1, n)$ red edges and at most $b(m, n - 1)$ blue edges. Let v be a vertex that is incident with exactly $b(m - 1, n)$ red edges. Suppose further that there are at least $b(m, n - 1) + 1$ vertices joined by blue edges to some vertex x , which, in turn, is joined by a blue edge to v . Prove that a red $K_{m,m}$ or a blue $K_{n,n}$ occurs as subgraph in the edge colouring.
- (d) Combine the results in Tasks 1(a)–(c) to prove that a red $K_{m,m}$ or a blue $K_{n,n}$ occurs as subgraph in *any* red-blue edge colouring of $K_{b,b}$, where $b = b(m - 1, n) + b(m, n - 1) + 1$, thereby showing that $b(m, n) \leq b(m - 1, n) + b(m, n - 1)$.
2. Use an induction argument (similar to that in [Theorem 17.10](#)), the result of [Theorem 17.13](#) and [Pascal's identity \(17.2\)](#) to prove that $b(m, n) \leq \binom{m+n}{m} - 1$.

Further reading

- [1] LW Beineke and AJ Schwenk, 1975. *On a bipartite form of the Ramsey problem*, Congressus Numerantium, **15**, pp. 17–22.
- [2] A Bialostocki and J Schönheim, 1984. *On Some Turán and Ramsey Numbers for C_4* , Academic Press, London, pp. 29–33.
- [3] JA Bondy and P Erdős, 1973. *Ramsey numbers for cycles in graphs*, Journal of Combinatorial Theory, **14**, pp. 46–54.
- [4] RC Brewster, EJ Cockayne and CM Mynhardt, 1989. *Irredundant Ramsey numbers for graphs*, Journal of Graph Theory, **13**, pp. 283–290.
- [5] RC Brewster, EJ Cockayne and CM Mynhardt, 1990. *The irredundant Ramsey number $s(3, 6)$* , Quaestiones Mathematicae, **13**, pp. 141–157.

- [6] AP Burger, PJP Grobler, EH Stipp and JH van Vuuren, 2004. *Diagonal Ramsey numbers in multipartite graphs*, Utilitas Mathematica, **66**, pp. 137–163.
- [7] AP Burger, JH Hattingh and JH van Vuuren, 2014. *The mixed irredundant Ramsey numbers $t(3, 7) = 18$ and $t(3, 8) = 22$* , Quaestiones Mathematicae, **37**, pp. 571–589.
- [8] AP Burger and JH van Vuuren, 2004. *Ramsey numbers in complete balanced multipartite graphs. Part I: Set numbers*, Discrete Mathematics, **283**, pp. 37–43.
- [9] AP Burger and JH van Vuuren, 2004. *Ramsey numbers in complete balanced multipartite graphs. Part II: Size numbers*, Discrete Mathematics, **283**, pp. 45–49.
- [10] AP Burger and JH van Vuuren, 2011. *Avoidance colourings for small non-classical Ramsey numbers*, Discrete Mathematics and Theoretical Computer Science, **13(2)**, pp. 81–96.
- [11] AP Burger and JH van Vuuren, *The irredundant Ramsey numbers $s(3, 8) = 21$ and $w(3, 8) = 21$* , Utilitas Mathematica, To appear.
- [12] SA Burr, P Erdős, RJ Faudree, CC Rousseau and RH Schelp, 1983. *On Ramsey numbers involving star-like multipartite graphs*, Journal of Graph Theory, **7**, pp. 395–409.
- [13] G Chartrand and OR Oellermann, 1993. *Applied and Algorithmic Graph Theory*, McGraw-Hill, New York (NY), §12.2.
- [14] G Chartrand and S Schuster, 1971. *On the existence of specified cycles in complementary graphs*, Bulletin of the American Mathematical Society, **77**, pp. 995–998.
- [15] G Chen, JH Hattingh and CC Rousseau, 1993. *Asymptotic bounds for irredundant and mixed Ramsey numbers*, Journal of Graph Theory, **17**, pp. 193–206.
- [16] G Chen and CC Rousseau, 1995. *The irredundant Ramsey number $s(3, 7)$* , Journal of Graph Theory, **19**, pp. 263–270.
- [17] FRK Chung, 1973. *On the Ramsey numbers $N(3, 3, \dots, 3; 2)$* , Discrete Mathematics, **5**, pp. 317–321.
- [18] V Chvátal, 1977. *Tree-complete graph Ramsey numbers*, Journal of Graph Theory, **1**, p. 93.
- [19] V Chvátal and F Harary, 1972. *Generalised Ramsey theory for graphs. II: Small diagonal numbers*, Proceedings of the American Mathematical Society, **32**, pp. 389–394.
- [20] EJ Cockayne, G Exoo, JH Hattingh and CM Mynhardt, 1992. *The irredundant Ramsey number $s(4, 4)$* , Utilitas Mathematica, **41**, pp. 119–128.
- [21] EJ Cockayne, JH Hattingh, J Kok and CM Mynhardt, 1990. *Mixed Ramsey numbers and irredundant Turán numbers for graphs*, Ars Combinatoria, **29C**, pp. 57–68.
- [22] EJ Cockayne, JH Hattingh and CM Mynhardt, 1991. *The irredundant Ramsey number $s(3, 7)$* , Utilitas Mathematica, **39**, pp. 145–160.

- [23] EJ Cockayne and CM Mynhardt, 1994. *The irredundant Ramsey number $s(3, 3, 3)$* , Journal of Graph Theory, **18**, pp. 595–604.
- [24] D Day, W Goddard, MA Henning and HC Swart, 2001. *Multipartite Ramsey numbers*, Ars Combinatoria, **58**, pp. 23–31.
- [25] T Dzido and R Zakrzewska, 2006. *The upper domination Ramsey number $u(4, 4)$* , Discussiones Mathematicae Graph Theory, **26**, pp. 419–430.
- [26] P Erdős, 1994. *Problems and results in discrete mathematics: Trends in discrete mathematics*, Discrete Mathematics, **136(1–3)**, pp. 53–73.
- [27] P Erdős and G Szekeres, 1935. *A combinatorial problem in geometry*, Compositio Mathematica, **2**, pp. 464–470.
- [28] G Exoo, 1987. *Constructing Ramsey graphs with a computer*, Congressus Numerantium, **59**, pp. 31–36.
- [29] G Exoo, 1994. *A lower bound for Schur numbers and multicolour Ramsey numbers of K_3* , Electronic Journal of Combinatorics, **1**, R8.
- [30] G Exoo, 1998. *Some new Ramsey colourings*, Electronic Journal of Combinatorics, **5**, R29.
- [31] RJ Faudree and RH Schelp, 1974. *All Ramsey numbers for cycles in graphs*, Discrete Mathematics, **8**, pp. 313–329.
- [32] L Gerencsér and A Gyárfás, 1969. *On Ramsey-type problems*, Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Mathematica, **10**, pp. 167–170.
- [33] W Goddard, MA Henning and OR Oellermann, 2000. *Bipartite Ramsey numbers and Zarankiewics numbers*, Discrete Mathematics, **219**, pp. 85–95.
- [34] AW Goodman, 1959. *On sets of acquaintances and strangers at any party*, American Mathematical Monthly, **66(9)**, pp. 778–783.
- [35] JE Graver and J Yackel, 1968. *Some graph theoretic results associated with Ramsey's theorem*, Journal of Combinatorial Theory, **4**, pp. 125–175.
- [36] RE Greenwood and AM Gleason, 1955. *Combinatorial relations and chromatic graphs*, Canadian Journal of Mathematics, **7**, pp. 1–7.
- [37] CM Grinstead and SM Roberts, 1982. *On the Ramsey numbers $r(3, 8)$ and $r(3, 9)$* , Journal of Combinatorial Theory, **33**, pp. 27–51.
- [38] PJP Grobler, 1992. *Irredundant Ramsey numbers*, Technical report, University of South Africa, Pretoria.
- [39] F Harary, 1983. *A tribute to Frank P Ramsey, 1903–1930*, Journal of Graph Theory, **7**, pp. 1–7.
- [40] H Harborth and I Mengersen, 1991. *The Ramsey number of $K_{3,3}$* , pp. 639–644 in Y Alavi, FRK Chung, RL Graham and DF Hsu (Eds), *Combinatorics, Graph Theory and Applications: Proceedings of the 6th Quadrennial International Conference on the Theory and Application of Graphs*, John Wiley & Sons, New York (NY).
- [41] H Harborth and I Mengersen, 1996. *Some Ramsey numbers for complete bipartite graphs*, Australasian Journal of Combinatorics, **13**, pp. 119–128.

- [42] H Harborth and I Mengersen, 2001. *Ramsey numbers in octahedron graphs*, Discrete Mathematics, **231**, pp. 241–246.
- [43] JH Hattingh and MA Henning, 1998. *Bipartite Ramsey theory*, Utilitas Mathematica, **53**, pp. 217–230.
- [44] JH Hattingh and MA Henning, 1998. *Star-path bipartite Ramsey numbers*, Discrete Mathematics, **185**, pp. 255–258.
- [45] MA Henning and OR Oellermann, 2002. *The upper domination Ramsey number $u(3, 3, 3)$* , Discrete Mathematics, **242**, pp. 103–113.
- [46] MA Henning and OR Oellermann, 2004. *On upper domination Ramsey numbers of graphs*, Discrete Mathematics, **274**, pp. 125–135.
- [47] R Hill and RW Irving, 1982. *On group partitions associated with lower bounds for symmetric Ramsey numbers*, European Journal of Combinatorics, **3**, pp. 35–50.
- [48] RW Irving, 1978. *A bipartite Ramsey problem and the Zarankiewics numbers*, Glasgow Mathematics, **19**, pp. 13–26.
- [49] JG Kalbfleisch, 1966. *Chromatic graphs and Ramsey's theorem*, PhD thesis, University of Waterloo, Waterloo.
- [50] DL Kreher, L Wei and SP Radziszowski, 1988. *Lower bounds for multi-coloured Ramsey numbers from group orbits*, Journal of Combinatorial Mathematics and Combinatorial Computing, **4**, pp. 87–95.
- [51] BD McKay and ZK Min, 1992. *The value of the Ramsey number $r(3, 8)$* , Journal of Graph Theory, **16**, pp. 99–105.
- [52] BD McKay and SP Radziszowski, 1995. $r(4, 5) = 25$, Journal of Graph Theory, **19**, pp. 309–322.
- [53] DH Mellor, 1983. *The eponymous FP Ramsey*, Journal of Graph Theory, **7(1)**, pp. 9–13.
- [54] OR Oellermann and W Shreve, Unpublished manuscript.
- [55] K Piwakowski and SP Radziszowski, 1998. $30 \leq r(3, 3, 4) \leq 31$, Journal of Combinatorial Mathematics and Combinatorial Computing, **27**, pp. 135–141.
- [56] SP Radziszowski, *Small Ramsey numbers*, Electronic Journal of Combinatorics, **DS1**.
- [57] FP Ramsey, 1930. *On a problem of formal logic*, Proceedings of the London Mathematical Society, **30**, pp. 264–286.
- [58] V Rosta, 1973. *On a Ramsey-type problem of JA Bondy and P Erdős*, Journal of Combinatorial Theory, **15**, pp. 94–120.
- [59] CC Rousseau and SE Speed, 2003. *Mixed Ramsey numbers revisited*, Combinatorics, Probability and Computing, **12**, pp. 653–660.
- [60] A Sanchez-Flores, 1995. *An improved bound for Ramsey number $N(3, 3, 3, 3; 2)$* , Discrete Mathematics, **140**, pp. 281–286.
- [61] A Schelten, I Schiermeyer and R Faudree, 1999. *3-colored Ramsey numbers of odd cycles*, Electronic Notes in Discrete Mathematics, **3**, pp. 176–178.

- [62] JH Spencer, 1975. *Ramsey's theorem — A new lower bound*, Journal of Combinatorial Theory, **18**, pp. 108–115.
- [63] JH Spencer, 1983. *Ramsey theory and Ramsey theoreticians*, Journal of Graph Theory, **7**, pp. 15–23.
- [64] S Wenlong, 1996. *The estimation of lower bounds about some Ramsey numbers $R_3(n)$ and $R_4(n)$* , Guangxi Sciences, (in Chinese).
- [65] EG Whitehead, 1973. *The Ramsey number $N(3, 3, 3, 3; 2)$* , Discrete Mathematics, **4**, pp. 389–396.
- [66] Y Yuansheng and P Rowlinson, 1992. *On the third Ramsey numbers of graphs with five edges*, Journal of Combinatorial Mathematics and Combinatorial Computing, **11**, pp. 213–222.
- [67] Y Yuansheng and P Rowlinson, 1993. *On graphs without 6-cycles and related Ramsey numbers*, Utilitas Mathematica, **44**, pp. 192–196.



Extremal graph theory

Contents

18.1	Introduction	655
18.2	Cycles in graphs	656
18.3	Turán's theorem	657
18.4	Zarankiewicz numbers	659
18.5	Framing numbers	662
18.6	Diameter two graphs of minimum size	667
18.7	Diameter two critical graphs of maximum size	673
	Exercises	678
	Further reading	678

18.1 Introduction

A n important topic in extremal graph theory is Ramsey Theory which was covered in some detail in Chapter 17. In this chapter, we investigate questions such as the following: *What is the maximum possible number of edges in a graph of order n that does not contain a complete graph K_r for some given r ?*, or *What is the maximum number of edges in a complete bipartite graph $K_{m,n}$ that does not contain a biclique $K_{r,r}$ for some given r ?*, or *What is the minimum number of edges in a graph of diameter 2 with no dominating vertex?* Questions of this nature are examples of problems typically considered in extremal graph theory.

Let F be a graph and let n be an integer such that $n \geq |V(F)|$. The **extremal number** $\text{ex}(n; F)$ of F is the maximum number of edges in a graph of order n that does not contain F as a subgraph. Graphs of order n and size $\text{ex}(n; F)$ not containing F as a subgraph are called **extremal** for n and F . Recall, furthermore, that we say that a graph is **F -free** if it does not contain F as an induced subgraph.

By definition, every graph of order n and size at least $\text{ex}(n; F) + 1$ contains F as a subgraph. Clearly, $\text{ex}(n; P_2) = 0$ for $n \geq 2$, while $\text{ex}(n; P_3) = \lfloor \frac{n}{2} \rfloor$ for $n \geq 3$ (see Exercise 18.1).

18.2 Cycles in graphs

For $n \geq 3$, we have $\text{ex}(n; C_n) = \binom{n-1}{2} + 1$ (see [Exercise 18.2](#)). [Mantel \[20\]](#) determined the maximum number of edges in a graph of order n that contains no triangle C_3 .

Theorem 18.1 *For $n \geq 3$, $\text{ex}(n; C_3) = \lfloor n^2/4 \rfloor$.*

Proof We show first that every graph of order n and size at least $\lfloor n^2/4 \rfloor + 1$ contains a triangle. We proceed by induction on n . When $n = 3$, $\lfloor n^2/4 \rfloor + 1 = 3$ and the only graph of order 3 and size 3 is K_3 , which is itself a triangle. This establishes the base case. Assume, as induction hypothesis, that $n \geq 4$ and that every graph of order $k \in \{3, 4, \dots, n-1\}$, and size at least $\lfloor k^2/4 \rfloor + 1$ contains a triangle. Let G be a graph of order n and size $m \geq \lfloor n^2/4 \rfloor + 1$. Let u and v be two adjacent vertices of G , and let $F = G - u - v$. If u and v have a common neighbour, then G contains a triangle, as desired. Hence, we may assume that u and v have no common neighbour, and so each vertex of F is adjacent to at most one of u and v . Thus, $m \leq |E(F)| + n - 1$, and so

$$|E(F)| \geq \left\lfloor \frac{n^2}{4} \right\rfloor + 1 - (n - 1) = \left\lfloor \frac{n^2 - 4n + 4}{4} \right\rfloor + 1 = \left\lfloor \frac{(n-2)^2}{4} \right\rfloor + 1.$$

Applying the induction hypothesis to the graph F , which has order $n-2$, we have that F contains a triangle. Since F is a subgraph of G , the graph G therefore contains a triangle, as desired. Hence, by induction, we have shown that every graph of order n and size at least $\lfloor n^2/4 \rfloor + 1$ contains a triangle, *i.e.* $\text{ex}(n; C_3) \leq \lfloor n^2/4 \rfloor$. That this bound is best possible may be seen by considering the graph $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$, where $n \geq 3$, which is a triangle-free graph of order n and size $\lfloor n^2/4 \rfloor$. Consequently, $\text{ex}(n; C_3) = \lfloor n^2/4 \rfloor$. ■

As we shall see later in this chapter, the graph $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$ is the unique extremal graph for n and C_3 . To determine the maximum number of edges in a graph of order n that contains no quadrilateral C_4 , we need the following inequality called [Jensen's inequality](#) (see [Exercise 18.3](#)): If a_1, \dots, a_n and b_1, \dots, b_n are nonincreasing sequences of integers, then

$$\sum_{i=1}^n a_i b_i \geq n \left(\frac{1}{n} \sum_{i=1}^n a_i \right) \left(\frac{1}{n} \sum_{i=1}^n b_i \right).$$

Theorem 18.2 *For $n \geq 4$, $\text{ex}(n; C_4) \leq \frac{n}{4} (1 + \sqrt{4n-3})$.*

Proof Let $G = (V, E)$ be an (n, m) graph with $n \geq 4$ that contains no 4-cycle. For a fixed vertex v of G , there are $\binom{d(v)}{2}$ distinct pairs of vertices that have v as their common neighbour. Since G contains no 4-cycles, however, each pair of vertices has at most one common neighbour and is therefore counted at most once in the sum $\sum_{v \in V} \binom{d(v)}{2}$. Hence,

$$\sum_{v \in V} \binom{d(v)}{2} \leq \binom{n}{2},$$

since there are $\binom{n}{2}$ distinct pairs of vertices in a graph of order n . Let d_1, \dots, d_n be the degree sequence of G in nonincreasing order. Then, by Jensen's inequality,

$$\begin{aligned}\sum_{v \in V} \binom{d(v)}{2} &= \sum_{i=1}^n \binom{d_i}{2} = \frac{1}{2} \sum_{i=1}^n d_i(d_i - 1) \geq \frac{n}{2} \left(\frac{1}{n} \sum_{i=1}^n d_i \right) \left(\frac{1}{n} \sum_{i=1}^n (d_i - 1) \right) \\ &= \frac{n}{2} \left(\frac{2m}{n} \right) \left(\frac{2m - n}{n} \right).\end{aligned}$$

Hence,

$$\frac{n}{2} \left(\frac{2m}{n} \right) \left(\frac{2m - n}{n} \right) \leq \binom{n}{2},$$

or, equivalently,

$$\frac{m(2m - n)}{n} \leq \frac{n(n - 1)}{2},$$

and so $4m^2 - 2nm + (n^2 - n^3) \leq 0$. Solving this inequality for m yields

$$m \leq \frac{n}{4} (1 + \sqrt{4n - 3}),$$

and the desired result follows. ■

❖ The reader should now be able to attempt Exercises 18.1–18.4.

18.3 Turán's theorem

In this section we present a classic result of Turán dating back to 1941 which characterises those graphs of order n with the largest possible number of edges that do not contain a complete graph K_r for some given integer r . Turán [24] significantly extended the result of Theorem 18.1.

Before stating the result of Turán, we define the Turán graph and present two useful lemmas. For $r \geq 1$, the **Turán graph**, denoted by $T_{n,r}$, is the (unique) complete r -partite graph of order $n \geq r$ whose partite sets differ in size by at most 1. For $n \leq r - 1$, we define $T_{n,r} = K_n$ and denote the number of edges in $T_{n,r}$ by $t_{n,r}$, which we call the **Turán number**.

No r -partite graph has a subgraph K_{r+1} , since each partite set contributes at most one vertex to any complete subgraph. Hence all complete r -partite graphs are obvious candidates as extremal graphs for n and K_{r+1} . We show first that the Turán graph $T_{n,r}$ is unique in terms of size among r -partite graphs.

Lemma 18.3 *Among all r -partite graphs, the Turán graph $T_{n,r}$ has the maximum number of edges.*

Proof Let $G = (V, E)$ be an r -partite graph of order n with the maximum number of edges. Necessarily, G is a complete r -partite graph. If there are two partite sets V_1 and V_2 in G with $|V_1| - |V_2| \geq 2$, then by moving a vertex v from the larger set V_1 to the smaller set V_2 we produce a new partition of V that defines a complete r -partite graph G' in which the edges of G not incident with v remain unchanged, but with v now adjacent to the vertices in $V_1 \setminus \{v\}$ and not adjacent to the vertices in V_2 . Hence, the size of the graph G' is $|E| + |V_1| - 1 - |V_2| \geq |E| + 1$, and so G' has size greater than that of G , contrary to our choice of G . Therefore,

no two partite sets of G differ by more than one, *i.e.* the partition sets of G differ in size by at most 1, whence $G = T_{n,r}$. \blacksquare

Next we present a recursive relation involving the Turán numbers.

Lemma 18.4 *For any integers $n > r \geq 2$, $t_{n,r} = t_{n-r,r} + (r-1)(n-r) + \binom{r}{2}$.*

Proof Let $G = T_{n,r}$. Then G is a complete r -partite graph of order $n > r$ whose partite sets differ in size by at most 1. Let V_1, \dots, V_r denote the partite sets of G . For $i \in [r]$, let $v_i \in V_i$ and define $S = \{v_1, \dots, v_r\}$. Then either $G - S$ is the complete graph K_{n-r} if $n-r < r$ or $G - S$ is a complete r -partite graph of order $n-r \geq r$ whose partite sets differ in size by at most 1. Hence, $G - S = T_{n-r,r}$. Furthermore, since G is a complete r -partite graph, each of the $n-r$ vertices in the set $V(G) \setminus S$ is adjacent in G to exactly $r-1$ vertices from the set S . It, therefore, follows that $t_{n,r} = |E(G)| = |E(G - S)| + (r-1)(n-r) + |E(K_r)| = t_{n-r,r} + (r-1)(n-r) + \binom{r}{2}$. \blacksquare

We are now in a position to prove Turán's result that the Turán graph $T_{n,r}$ is the unique extremal graph for n and K_{r+1} .

Theorem 18.5 *Let n and r be natural numbers. If G is a graph of order n and size $\text{ex}(n; K_{r+1})$ that contains no subgraph K_{r+1} , then $G = T_{n,r}$.*

Proof We proceed by induction on n . For $n \leq r$, we have that $\text{ex}(n; K_{r+1}) = \binom{n}{2} = t_{n,r}$ and that the unique extremal graph for n and K_{r+1} is the complete graph $K_n = T_{n,r}$. This establishes the base case. Assume, as induction hypothesis, that $n \geq r+1$ and that the result is true for all positive integers less than n .

Let G be a graph of order n and size $\text{ex}(n; K_{r+1})$ that is K_{r+1} -free. Since G is edge-maximal with respect to being K_{r+1} -free, the graph G contains a subgraph $F = K_r$. By the induction hypothesis, $T_{n-r,r}$ is the unique extremal graph for $n-r$ and K_{r+1} , and so $\text{ex}(n-r; K_{r+1}) = t_{n-r,r}$. Hence, the graph $G - V(F)$ of order $n-r$ that is K_{r+1} -free has size at most $t_{n-r,r}$. Since G is K_{r+1} -free, each of the $n-r$ vertices in the graph $G - V(F)$ is adjacent in G to at most $r-1$ vertices in $V(F)$. Hence,

$$\begin{aligned} \text{ex}(n; K_{r+1}) &= |E(G)| \leq |E(G - V(F))| + (r-1)(n-r) + |E(F)| \\ &\leq t_{n-r,r} + (r-1)(n-r) + \binom{r}{2} \\ &= t_{n,r} \quad (\text{by Lemma 18.4}) \\ &\leq \text{ex}(n; K_{r+1}). \end{aligned}$$

We must, therefore, have equality throughout the above inequality chain. In particular, every vertex in $G - V(F)$ has exactly $r-1$ neighbours in F . Furthermore, the graph $G - V(F)$ of order $n-r$ that is K_{r+1} -free has size $t_{n-r,r}$, and so, by the induction hypothesis, $G - V(F) = T_{n-r,r}$. Let $V(F) = \{v_1, \dots, v_r\}$. For $i \in [r]$, let V_i be the set of all vertices in G that are not adjacent to v_i , and so each vertex in V_i is adjacent to every vertex in $V(F) \setminus \{v_i\}$. Since G is K_{r+1} -free, each of the sets V_i is independent, and they partition $V(G)$. Hence, G is an r -partite graph. Since $G = T_{n-r,r}$, it follows readily that $G = T_{n,r}$. \blacksquare

For any integers $n \geq r \geq 1$, it holds that

$$t_{n,r} \leq \frac{1}{2} \left(\frac{r-1}{r} \right) n^2$$

(see [Exercise 18.5](#)). Hence, we have the following result as an immediate consequence of [Theorem 18.5](#).

Theorem 18.6 (Turán's Theorem) *For any integers $n \geq r \geq 1$,*

$$\text{ex}(n; K_{r+1}) \leq \left\lfloor \frac{1}{2} \left(\frac{r-1}{r} \right) n^2 \right\rfloor.$$

Furthermore, if G is a graph of order n and size $\text{ex}(n; K_{r+1})$ that contains no subgraph K_{r+1} , then $G = T_{n,r}$.

As a special case of [Theorem 18.6](#), we have the following bound on the maximum number of edges in a triangle-free graph.

Theorem 18.7 *If G is a triangle-free graph of order n and size m , then $m \leq \lfloor n^2/4 \rfloor$, with equality if and only if G is the complete bipartite graph $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$.*

❖ The reader should now be able to attempt [Exercise 18.5](#).

18.4 Zarankiewicz numbers

In this section, we investigate bounds on the maximum number of edges in a subgraph of the complete bipartite graph without a particular biclique.

We denote by $z(s, t; m, n)$ the least positive integer z such that any subgraph of $K_{s,t}$ with z edges contains $K_{m,n}$ as a subgraph, with the m vertices a subset of the s vertices and the n vertices a subset of the t vertices, and we put $z(s; m) = z(s, s; m, m)$. These are called the **Zarankiewicz numbers** [25]. The **Zarankiewicz problem** is to find the maximum number of edges in a bipartite graph not containing a given bipartite graph F as a subgraph.

In this section we take F to be a quadrilateral C_4 . We begin with an estimate for the maximum number $z(n; 2)$ of edges that a subgraph G of $K_{n,n}$ can have if G contains no quadrilateral, i.e. G contains no $K_{2,2}$ as a subgraph. The proof follows that of [Chung and Graham](#) [5].

Theorem 18.8 *For all integers $n \geq 2$,*

$$z(n; 2) \leq (n + n\sqrt{4n - 3})/2.$$

Proof Let $K_{n,n}$ have partite sets $\mathcal{L} = [n]$ and $\mathcal{R} = [n]$, and let $\mathbf{A} = (a_{ij})$ denote the $n \times n$ adjacency matrix of G , where $a_{ij} = 1$ if $i \in \mathcal{L}$ and $j \in \mathcal{R}$ are adjacent in G , or $a_{ij} = 0$ otherwise. Since $K_{2,2} \not\subseteq G$,

$$\sum_{j=1}^n a_{ij} a_{i'j} \leq 1 \tag{18.1}$$

for any choice of $1 \leq i < i' \leq n$. If c_j denotes $\sum_{i=1}^n a_{ij}$ then, summing (18.1) over all choices of i and i' , we obtain

$$\sum_{j=1}^n c_j (c_j - 1) \leq n(n - 1). \tag{18.2}$$

Applying the Schwarz inequality, $n \sum c_j^2 \geq (\sum c_j)^2$, to (18.2), we have

$$z(n; 2) = \sum_{j=1}^n c_j \leq \frac{n}{2} + n \sqrt{n - \frac{3}{4}},$$

which is the desired bound. ■

Equality in [Theorem 18.8](#) is known to hold for $n = q^2 + q + 1$, where q is a prime power (see [2]). For other large values of n , the upper bounds for $z(n; 2)$ given in [Theorem 18.8](#), or by other known results (see [2, 5], for example), are not always attainable.

We call a quadrilateral-free bipartite graph on n vertices with the largest possible number of edges an **extremal graph** for n ; its number of edges we call the **Zarankiewicz value** for n and C_4 , denoted by $z(n; C_4)$.

[Table 18.1](#) contains exact values of $z(n; C_4)$ for $n \leq 20$. Note that a quadrilateral-free bipartite graph has girth at least 6. Thus, in the case where the graph with maximum size and girth at least 6 is bipartite, the Zarankiewicz value will coincide.

n	≤ 5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$z(n; C_4)$	$n - 1$	6*	7*	9*	10	12	14	16	18	21*	22	24	26	29*	31*	34*

Table 18.1: Zarankiewicz values $z(n; C_4)$ for n and C_4 . An asterisk means that the extremal graph is unique.

First we determine the extremal graphs, which establishes the fact that the values in [Table 18.1](#) are lower bounds on Zarankiewicz values. Let $G_{18.1}$ be obtained from $K_{2,3}$ by subdividing the three edges incident with a vertex of degree 3 and let $G_{18.2}$ be the Heawood graph in [Figure 18.1](#) (which is the unique cubic graph of girth 6).

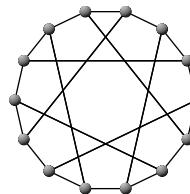


Figure 18.1: The Heawood graph, $G_{18.2}$.

Let $G^{(20)}$ be constructed as follows: Take the cartesian product $K_4 \square K_2$. In each copy of K_4 , subdivide each edge with a new vertex. Finally, join each new vertex in the first copy of K_4 to the new vertex in the second copy which is at distance 5 from it. For $n \leq 19$, $n \notin \{8, 14\}$, let $G^{(n)}$ be obtained from $G^{(n+1)}$ by deleting a vertex of minimum degree. For $n \leq 20$, the graphs $G^{(n)}$ constructed above are extremal graphs; that is, $G^{(n)}$ is a quadrilateral-free bipartite graph of order n and size $z(n; C_4)$.

In order to establish upper bounds on the Zarankiewicz values, we begin with the following lemma.

Lemma 18.9 *Let G be a quadrilateral-free bipartite graph of order n with minimum degree $\delta \geq 3$. Then each partite set has at least seven vertices, so that $n \geq 14$. Furthermore, a partite set with maximum degree Δ' has at least $2\Delta' + 1$ vertices. In particular, if there is a vertex of degree four, then each partite set has at least nine vertices, and if there is an edge joining two vertices of degree four, then each partite set has at least ten vertices.*

Proof A vertex of degree d has at least $2d$ vertices at distance 2. So each partite set has at least seven members. Furthermore, if $\Delta \geq 4$, then there must be a vertex in the smaller partite set with degree at least 4, and so each partite set has at least nine members. A vertex of degree 4 that is adjacent to a vertex of degree 4 has at least nine vertices at distance 2. ■

We are now in a position to prove that the lower bound values in Table 18.1 are, in fact, also upper bounds on the Zarankiewicz values for $n \leq 20$.

Theorem 18.10 *The entries in Table 18.1 are upper bounds on $z(n; C_4)$ for $n \leq 20$. Furthermore, for the entries with asterisks the extremal graphs are unique.*

Proof Let $H^{(n)}$ be a quadrilateral-free bipartite graph of order n and size $m_n = z(n; C_4)$. If $n \leq 5$, then $H^{(n)}$ is a tree, and so $m_n = n - 1$. If $m_6 \geq 6$, then $H^{(6)}$ contains a cycle. This must be an (induced) 6-cycle, and so $m_6 = 6$. If $m_7 \geq 7$, then $H^{(7)}$ contains a cycle. This must be an (induced) 6-cycle. Then the remaining vertex can have degree at most 1, and so $m_7 \leq 7$. Thus the unique extremal graph for $n = 7$ consists of a 6-cycle with an additional vertex of degree 1.

Suppose $n \in \{8, 9, \dots, 13\}$. By Lemma 18.9, $H^{(n)}$ contains a vertex v of degree at most 2. Hence $m_n \leq m(H^{(n)} - v) + 2 \leq m_{n-1} + 2$. If $m(H^{(8)} - v) = m_7 = 7$, then $H^{(8)} - v$ is the unique extremal graph for $n = 7$. Furthermore, if v has degree 2, then it must be adjacent to the two vertices at distance 4 in $H^{(8)} - v$. Thus, $m_8 = 9$ and the unique extremal graph is obtained from $K_{2,3}$ by subdividing the three edges incident with a vertex of degree 3. If $m(H^{(9)} - v) = m_8 = 9$, then it is the unique extremal graph for $n = 8$. Since this extremal graph has diameter 3, the vertex v cannot have degree 2, whence $m_9 \leq 10$. The upper bounds for $n \in \{10, 11, 12, 13\}$ now follow.

Suppose $n = 14$. If $\delta(H^{(14)}) = 2$, then $m_{14} \leq m_{13} + 2 \leq 20$. On the other hand, if $\delta(H^{(14)}) > 2$, then, by Lemma 18.9, $H^{(14)}$ must be a cubic graph. Hence, $m_{14} = 21$ and the Heawood graph $G_{18,2}$ — the unique cubic graph of girth 6 — is the unique extremal graph for $n = 8$.

Suppose $n \in \{15, 16, 17\}$. Then, by Lemma 18.9, $\Delta(H^{(n)}) \leq 3$. If G_n has a vertex v of degree 2, then $m_n \leq m(H^{(n)} - v) \leq m_{n-1} + 2$. If, however, $m(H^{(15)} - v) = m_{14} = 21$, then $H^{(15)} - v$ is the Heawood graph $G_{18,2}$ which has diameter 3, and so v cannot have degree 2. Thus, $m_{15} \leq 22$. Now either $H^{(16)}$ is a cubic graph or it has minimum degree 2. In both cases, $m_{16} \leq 24$. The upper bound for $n = 17$ now follows.

Suppose $n = 18$. If $\delta(H^{(n)}) = 2$, then $m_n \leq m_{n-1} + 2 = 28$. Suppose, then, that $\delta(H^{(n)}) \geq 3$. If G_n is a cubic graph, then $m_n \leq 27$, and so we may assume $\Delta(H^{(n)}) \geq 4$. Let $H^{(n)}$ have partite sets \mathcal{L} and \mathcal{R} . By Lemma 18.9, $|\mathcal{L}| = |\mathcal{R}| = 9$ and $\Delta(H^{(n)}) \leq 4$ (and so $\Delta(H^{(n)}) = 4$). If there are only two vertices of degree 4, then $m_n \leq 28$, and so we may assume there are at least two vertices of degree 4 in each partite set. Let v and w be two vertices of degree 4 in \mathcal{L} . By Lemma 18.9,

there is no edge joining two vertices of degree 4, and so every neighbour of v and w has degree 3. Furthermore, since there are no 4-cycles, $|N(v) \cup N(w)| = 7$. Let x be the common neighbour of v and w . Similarly, if v' and w' are two vertices of degree 4 in \mathcal{R} , then they are adjacent only to vertices of degree 3 and they have precisely one common neighbour, x' say. If x and x' are not adjacent, then x (x') belongs to a 4-cycle. So x and x' are adjacent. The remaining edges must be added so that the vertices of degree 3 different from x and x' induce a 2-regular graph. Since there are no 4-cycles, it is easily verified that these vertices induce a 12-cycle in such a way that each vertex of degree 4 is adjacent to every 4-th vertex on this cycle. Thus $m_{18} = 29$ and the extremal graph for $n = 18$ is unique.

Suppose $n = 19$. If $\delta(H^{(n)}) \geq 3$ and $m_n \geq 31$, then, by Lemma 18.9, $\Delta(H^{(n)}) = 4$ and each partite set has at least nine vertices. A partite set with nine vertices has at least four vertices of degree 4. Since there are no 4-cycles, there are exactly four such vertices and the union of their neighbourhoods contains all ten vertices of the larger partite set. In particular, there is an edge joining two vertices of degree 4. Thus, by Lemma 18.9, each partite set contains at least ten vertices, a contradiction. So if $\delta(H^{(n)}) \geq 3$, then $m_n \leq 30$. On the other hand, if $H^{(n)}$ has a vertex v of degree 2, then $m_n \leq m(H^{(n)} - v) + 2 \leq m_{n-1} + 2 = 31$. Moreover, if $m(H^{(n)} - v) = m_{n-1}$, then $H^{(n)} - v$ is the unique extremal graph for $n = 18$. The two neighbours of v must be at distance at least 4 apart in $H^{(n)} - v$. So there is a unique choice up to symmetry, namely, antipodal vertices of the 12-cycle. Thus, $m_{19} = 31$ and the extremal graph for $n = 19$ is unique.

Finally, suppose $n = 20$. If $\delta(H^{(n)}) = 2$, then $m_n \leq m_{n-1} + 2 = 33$. On the other hand, if $\delta(H^{(n)}) = 3$, then the deletion of a vertex v of degree 3 yields the unique extremal graph for $n = 19$. A simple calculation shows that the neighbours of v are uniquely identified. Thus, $m_{20} = 34$ and the extremal graph for $n = 20$ is unique. ■

18.5 Framing numbers

Chartrand *et al.* [4] introduced the notion of a framing number of a graph. A graph G can be **homogeneously embedded** in a graph H if, for every vertex x of G and every vertex y of H , there exists an embedding of G in H as an induced subgraph with x at y . A graph F of minimum order in which G can be homogeneously embedded is called a **frame** of G , and the order of F is called the **framing number** of G , denoted by $\text{fr}(G)$. In [4], it was shown that a frame exists for every graph, although such a frame need not be unique.

Chartrand *et al.* [4] extended the concept of framing numbers to more than one graph. For any two graphs G_1 and G_2 , the framing number of G_1 and G_2 , denoted by $\text{fr}(G_1, G_2)$, is defined as the minimum order of a graph F such that G_i ($i = 1, 2$) can be homogeneously embedded in F . The graph F is called a frame of G_1 and G_2 . It is known that $\text{fr}(G_1, G_2)$ exists and, in fact, $\text{fr}(G_1, G_2) \leq \text{fr}(G_1 \cup G_2)$. The following result was established in [4], the proof of which is left as an exercise (see Exercise 18.6).

Theorem 18.11 *For any graphs G and H , $\text{fr}(G, H) = \text{fr}(\bar{G}, \bar{H})$.*

A special case of this concept is when the graphs are vertex-transitive. If G is vertex-transitive, then G can be homogeneously embedded in H if every vertex

of H lies in an induced subgraph of H which is a copy of G . The framing number of a single vertex-transitive graph is trivially its order. If G and F are vertex-transitive graphs, then the framing number $\text{fr}(G, F)$ of G and F is the minimum order of a graph, every vertex of which belongs to an induced subgraph which is a copy of G and to an induced subgraph which is a copy of F . A result of this kind is presented later in this chapter, *viz.* the framing number of a complete graph and an independent set.

18.5.1 On cliques and independent sets

In this section, we determine the framing number $f(m, n) = \text{fr}(K_m, \bar{K}_n)$ of a complete graph and an independent set. The parameter $f(m, n)$ can also be defined as follows: For positive integers m and n , let $f(m, n)$ be the minimum order of a graph in which every vertex is in an m -clique, and every vertex is in an independent set of cardinality n .

If v is a vertex of a graph G , then we denote the maximum cardinality of an independent set of vertices of G that contains v by $\alpha(G, v)$. For any real number x , $\lfloor x \rfloor$ denotes the largest integer not exceeding x , $\lceil x \rceil$ the smallest integer not less than x , and $\llbracket x \rrbracket$ the nearest integer to x .

We establish the following result (see [6]).

Theorem 18.12 *For any integers $n, m \geq 2$,*

$$f(m, n) = \lceil (\sqrt{m-1} + \sqrt{n-1})^2 \rceil = m + n - 2 + \lceil 2\sqrt{(m-1)(n-1)} \rceil.$$

First, we consider a proof of the lower bound of [Theorem 18.12](#). For this purpose, we assume $G = (V, E)$ to be a *minimal* graph such that every vertex is in an m -clique ($m \geq 2$). That is, every vertex is in an m -clique, but upon the removal of any edge e there is then a vertex v not in an m -clique. We say that such an edge e is **critical** to the vertex v .

Lemma 18.13 *For every edge e of G there is a vertex u that is in a unique m -clique, and that m -clique contains e .*

Proof The edge e is critical to some vertex. Suppose first that $e = vw$ is critical to one of its endpoints, say v . Let F be an m -clique containing e . If F is unique, the desired result holds: set $u = v$. Otherwise, there is an edge vx_1 not in F . The edge vx_1 is critical to some vertex, call it y_1 . (Perhaps $y_1 = x_1$.) Note that $y_1 \notin \{v, w\}$. (Indeed, $y_1 \notin F$.) But every m -clique containing vx_1 contains v , and hence contains vw (since vw is critical to v). Thus, e is critical to y_1 . So, irrespective of whether $e = vw$ is critical to one of its endpoints or not, there is some vertex y_1 not incident with e such that e is critical to y_1 .

Let F_1 be an m -clique containing y_1 . Necessarily F_1 contains e , and hence v and w . If F_1 is unique, the desired result holds: set $u = y_1$. Otherwise, there is an edge y_1x_2 such that $x_2 \notin F_1$. The edge y_1x_2 is critical to some vertex, say y_2 . (Of course, perhaps $y_2 = x_2$.) Note that $y_2 \notin \{v, w, y_1\}$. But every m -clique that contains y_1x_2 contains y_1 , and so contains e . Hence e is critical to y_2 . Indeed, every m -clique containing y_2 contains y_1, v and w as well.

Let F_2 be an m -clique containing y_2 . If F_2 is not unique, we find an edge y_2x_3 with $x_3 \notin F_2$, and, by similar reasoning to the above, a vertex y_3 such that every

m -clique containing y_3 contains y_2, y_1, v and w also. If F_i is ever unique, the desired result holds: set $u = y_i$. Otherwise we produce an infinite sequence y_1, y_2, y_3, \dots such that every m -clique containing y_i contains $y_{i-1}, y_{i-2}, \dots, y_1, v$ and w . But this is absurd, since y_{m-1} would be in an m -clique with m other vertices. ■

Now, let L be the set of vertices of G of degree $m - 1$ — equivalently, those vertices in a unique m -clique. Let S be a maximum independent subset of L , and let $X = V \setminus S$. Let G have order p and let $|X| = x$. Note that $x \geq m - 1$.

Lemma 18.14 *If v is a vertex of G , then $\{v\} \cup (S \setminus N(v))$ is an independent set of maximum cardinality containing v .*

Proof Certainly $\{v\} \cup (S \setminus N(v))$ is independent and contains v . Let T be an independent set of maximum cardinality containing v . We show that there is an independent set T' that contains v such that $|T'| = |T|$, but for which $T' \cap (X \setminus \{v\}) = \emptyset$. Suppose T contains a vertex w of $X \setminus \{v\}$. By Lemma 18.13, there is a vertex $u \in L$ such that w is in the (unique) m -clique containing u . (Perhaps $u = w$.) If our first choice for u is not in S , then, in view of the maximality of S , u has a neighbour u' in S ; and w is in the m -clique containing u' . We may, therefore, assume that $u \in S$. If we replace w by u in T , we obtain an independent set that contains v still, but whose intersection with $X \setminus \{v\}$ contains one vertex fewer. Repeating this process sufficiently many times yields the desired set T' . By maximality it then follows that $T' = \{v\} \cup (S \setminus N(v))$. ■

The following result will also prove useful.

Lemma 18.15 $x + (x - (m - 1))(p - x) \geq nx$.

Proof The hypothesis of this lemma implies that $\sum_{v \in X} \alpha(G, v) \geq nx$. It follows from Lemma 18.14 that

$$\sum_{v \in X} \alpha(G, v) = \sum_{v \in X} |\{v\}| + \sum_{v \in X} |S \setminus N(v)|.$$

But every vertex w of S is adjacent to exactly $m - 1$ of the vertices in X . Thus, $\sum_{v \in X} |S \setminus N(v)| = |S|(|X| - (m - 1)) = (p - x)(x - (m - 1))$. ■

Rearranged, the inequality of Lemma 18.15 yields

$$p \geq (n - 1) + (m - 1) + \frac{(m - 1)(n - 1)}{\alpha} + \alpha,$$

where $\alpha = x - (m - 1)$. Since the function $f(\alpha) = \alpha + (m - 1)(n - 1)/\alpha$ assumes its minimum value at $\alpha = \sqrt{(m - 1)(n - 1)}$, it follows that a graph G which is minimal with respect to every vertex being in an m -clique, but is also such that every vertex in an independent set of cardinality n , has at least

$$n + m - 2 + \lceil 2\sqrt{(m - 1)(n - 1)} \rceil$$

vertices. This completes the proof of the lower bound of Theorem 18.12.

Next we present a proof of the upper bound of [Theorem 18.12](#). For positive integers m , x and s where $s(m-1) \geq x$ and $x \geq m-1$, we define a graph $G(m, x, s)$ as follows. The graph has vertex set $U \cup V$ where $U = \{u_0, u_1, \dots, u_{x-1}\}$ and $V = \{v_1, v_2, \dots, v_s\}$. The vertices in U form a clique and the vertices in V an independent set. Then for $i \in [s]$, v_i is joined to the $m-1$ vertices u_j for $(m-1)(i-1) \leq j \leq (m-1)i-1$, where subscripts are read modulo x . Hence there are $s(m-1)$ edges with one endpoint in U and the other endpoint in V , and these are distributed evenly amongst U . (The above definition is merely one way to ensure an even distribution.) The graph $G(m, x, s)$ has the following property.

Lemma 18.16 *Every vertex of $G(m, x, s)$ is in an m -clique, and every vertex is in an independent set of cardinality $s - \lceil s(m-1)/x \rceil + 1$.*

Now let $\alpha = \sqrt{(m-1)(n-1)}$ and consider the graph G isomorphic to $G(m, x, s)$ where $x = m-1 + \lceil \lceil \alpha \rceil \rceil$ and $s = n-1 + \lceil \alpha \rceil$. Certainly $x \geq m-1$ and it is easily verified that $s(m-1) \geq x$ (since $m, n \geq 2$).

Lemma 18.17 *The graph $G(m, x, s)$ has order $n+m-2+\lceil 2\sqrt{(m-1)(n-1)} \rceil$, every vertex is in an m -clique, and every vertex is in an independent set of cardinality n .*

Proof Since $\lceil \lceil \beta \rceil \rceil + \lceil \beta \rceil = \lceil 2\beta \rceil$ for any positive real number β , the graph G has order $(n-1) + (m-1) + \lceil 2\alpha \rceil$. [Lemma 18.16](#) implies that every vertex of G is in an m -clique. It thus remains to be verified that $s - \lceil s(m-1)/x \rceil + 1 \geq n$ or, equivalently, that

$$\lceil \alpha \rceil \geq \left\lceil \frac{(n-1 + \lceil \lceil \alpha \rceil \rceil)(m-1)}{m-1 + \lceil \lceil \alpha \rceil \rceil} \right\rceil. \quad (18.3)$$

Call $y = \lceil \lceil \alpha \rceil \rceil$, $a = m-1$, and $b = n-1$. If $\lceil \alpha \rceil = y$, then (18.3) holds if $y \geq (b+y)a/(a+y)$ (since $y \in \mathbb{N}$), which is equivalent to $y^2 \geq ab$ and hence to $y \geq \alpha$, which certainly holds. On the other hand, if $\lceil \alpha \rceil = y+1$, then (18.3) holds if $y+1 \geq (b+y+1)a/(a+y)$ (since $y+1 \in \mathbb{N}$), which is equivalent to $y^2 + y \geq ab$. But $y \geq \alpha - \frac{1}{2}$, and so $y^2 + y \geq \alpha^2 - \frac{1}{4} = ab - \frac{1}{4}$. Since $y, a, b \in \mathbb{N}$, however, it follows that $y^2 + y \geq ab$, as required. ■

This completes the proof of the upper bound of [Theorem 18.12](#).

18.5.2 On cliques and bicliques

In this section, we consider the framing number $g(m, n) = \text{fr}(K_m, K_{n,n})$ of a complete graph and a complete bipartite graph. The parameter $g(m, n)$ can also be defined as follows: For positive integers m and n , let $g(m, n)$ be the minimum order of a graph in which every vertex is in an m -clique, and every vertex is in an n -biclique.

For $n \geq 1$, we have $g(1, n) = g(2, n) = 2n$ (see [Exercise 18.7](#)), while for $m \geq 2$ we have $g(m, 1) = m$ (see [Exercise 18.8](#)). Our aim in this section is to show that if m is small, then we can determine the exact value of $g(m, n)$.

We define an **n - n -clique** to be the (disjoint) union of two complete subgraphs each of order n ; that is, an n - n -clique is an induced subgraph isomorphic to $2K_n$. Furthermore, we define a graph to be **n - n -clique-minimal** if every vertex is in an n - n -clique, but the removal of any edge leaves some vertex not in an n - n -clique. We shall need the following property of n - n -clique-minimal graphs. The proof is left as an exercise (see [Exercise 18.9](#)).

Lemma 18.18 *If G is n - n -clique-minimal, then for every edge e in G there is a vertex of degree $n-1$ in G such that every n - n -clique of G containing this vertex, also contains e .*

Using Lemma 18.18, we can readily establish the following result (see Exercise 18.10).

Lemma 18.19 *If G is n - n -clique-minimal, then G contains two vertices of degree $n-1$ whose closed neighbourhoods are disjoint.*

We are now in a position to establish a recursive bound on the framing number $g(m, n)$.

Lemma 18.20 *For $m \geq 2$ and $n \geq 2$, $g(m, n) \geq g(m+1, n) - 2$.*

Proof Let $I^{(m)}$ be a frame for K_m and $K_{n,n}$; that is, $I^{(m)}$ is a graph of order $g(m, n)$ in which every vertex is in an m -clique and every vertex is in an n -biclique. Then $J^{(m)} = \bar{I}^{(m)}$ is a graph of order $g(m, n)$ in which every vertex is in an independent set of cardinality m and every vertex is in an n - n -clique. We may assume that $J^{(m)}$ is n - n -clique-minimal. By Lemma 18.19, H_m contains two vertices v_1 and v_2 of degree $n-1$ whose closed neighbourhoods are disjoint. Let $J^{(m+1)}$ be the graph of order $g(m, n) + 2$ obtained from H_m by introducing two new vertices w_1 and w_2 and joining w_i to every vertex of $N(v_i)$ for $i = 1, 2$.

Since $J^{(m)}$ is an induced subgraph of $J^{(m+1)}$, every vertex of $J^{(m)}$ is still in an independent set of cardinality m in $J^{(m+1)}$ and every vertex of $J^{(m)}$ is in an n - n -clique in $J^{(m+1)}$. Since v_i belongs to an n - n -clique, so too does w_i (simply replace v_i by w_i in an n - n -clique of $J^{(m)}$ containing v_i). Hence, every vertex of $J^{(m+1)}$ is in an n - n -clique.

We show next that every vertex of $J^{(m+1)}$ is in an independent set of cardinality $m+1$. Let v be a vertex of $J^{(m)}$, and let S_v be an independent set of cardinality m containing v in $J^{(m)}$. If $v = v_i$ ($i = 1, 2$), then $S_v \cup \{w_i\}$ is an independent set of cardinality $m+1$. If $v \in N(v_i)$ ($i = 1, 2$), then we may choose S_v to contain v_{3-i} (if S_v contains a vertex of $N(v_{3-i})$, then simply replace this vertex by v_{3-i}). Then $S_v \cup \{v_{3-i}\}$ is an independent set of cardinality $m+1$. Finally, if $v \notin N[v_1] \cup N[v_2]$, then we may choose S_v to contain both v_1 and v_2 , in which case $S_v \cup \{w_1, w_2\}$ is an independent set of cardinality $m+2$. Hence, every vertex of $J^{(m+1)}$ is in an independent set of cardinality $m+1$.

Let $I^{(m+1)} = \bar{J}^{(m+1)}$. Then $I^{(m+1)}$ is a graph of order $g(m, n) + 2$ in which every vertex is in an $(m+1)$ -clique and every vertex is in an n -biclique. Hence, $g(m+1, n) \leq g(m, n) + 2$. \blacksquare

We next establish an upper bound on the framing number $g(m, n)$ in closed form.

Lemma 18.21 *For all $m, n \geq 2$, $g(m, n) \leq 2(m+n-2)$.*

Proof Let $G_1 \cong K_{n-1, n-1}$ have partite sets A and B , and let G_2 and G_3 be two (disjoint) copies of K_{m-1} . Let F be the graph of order $2(m+n-2)$ obtained from the disjoint union of G_1 , G_2 and G_3 by adding all edges between A and $V(G_2)$, all edges between B and $V(G_3)$, and the edges of a perfect matching between $V(G_2)$ and $V(G_3)$. Then every vertex of F is in an m -clique, and every vertex is in an n -biclique. Hence, $g(m, n) \leq |V(F)| = 2(m+n-2)$. \blacksquare

It is, in fact, possible to establish the following exact result.

Lemma 18.22 *For $m \geq 3$, $g(m, m + 2) = 4m$.*

Proof For $m \geq 3$, $\sqrt{m^2 - 1} = m - \epsilon$ where $0 < \epsilon < \frac{1}{4}$, and so $2m - \frac{1}{2} < 2\sqrt{m^2 - 1} < 2m$. Since $g(m, n) \geq f(m, n)$ for all integers $m \geq 2$ and $n \geq 2$, it follows from Theorem 18.12 that $g(m, m + 2) \geq \lceil (\sqrt{m-1} + \sqrt{m+1})^2 \rceil = 2m + \lceil 2\sqrt{m^2 - 1} \rceil = 4m$. By Lemma 18.21, however, $g(m, m + 2) \leq 4m$. Thus, $g(m, m + 2) = 4m$. ■

We are now in a position to state our main result of this section.

Theorem 18.23 *For $m \in \{2, 3, \dots, n - 2\}$, $g(m, n) = 2(m + n - 2)$.*

Proof By Lemma 18.22, $g(m, n) = 2(m + n - 2)$ for $m = n - 2$. The desired result now follows readily by induction on m by applying Lemmas 18.20 and 18.21. ■

For all integers $m, n \geq 2$ with $m \geq n - 1$, the value of $g(m, n)$ was investigated in [18]. We close this section by mentioning, without proof, two results from [18].

Theorem 18.24 *If $m \geq (n-1)^3$, then $g(m, n) \leq f(m, n) + 1$.*

Theorem 18.25 *Let m and n be integers with $m \geq n-1$. If $\lceil \sqrt{(m-1)(n-1)} \rceil \equiv 0 \pmod{n-1}$ or if m is sufficiently large and if $\sqrt{(m-1)(n-1)}$ is not an integer, then $g(m, n) = f(m, n)$.*

❖ The reader should now be able to attempt Exercises 18.6–18.10.

18.6 Diameter two graphs of minimum size

In this section we consider the following extremal problem due to Erdős and Rényi [7]: *What is the minimum size of a graph with diameter 2 such that no vertex is adjacent to every other vertex?*

We define a graph G to be a **diameter-2 graph** if $\text{diam}(G) = 2$. Let G be a diameter-2 graph of order n and size m . Necessarily, $n \geq 3$. Since G is connected, we have $m \geq n - 1$. Furthermore, if $m = n - 1$, then G is a tree. The only tree of order $n \geq 3$ and diameter 2 is, however, the star graph $K_{1,n-1}$. Hence we have the following trivial observation.

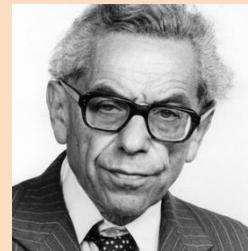
Observation 18.26 *If G is a diameter-2 graph of order n and size m , then $m \geq n - 1$, with equality if and only if G is the star graph $K_{1,n-1}$.*

Recall that a vertex of a graph G adjacent to every other vertex in G is called a **universal vertex** or a **dominating vertex** of G . If we forbid graphs with a dominating vertex, the situation becomes more interesting and significantly more edges are required. Erdős and Rényi [7] proved the following classical result on the minimum size of a diameter-2 graph with no dominating vertex.

Theorem 18.27 (Erdős-Rényi Theorem) *If G is a diameter-2 graph of order n and size m with no dominating vertex, then $m \geq 2n - 5$.*

In this section our aim is two-fold: First to prove the Erdős-Rényi Theorem, and secondly to characterise those graphs that achieve equality in the lower bound established in the Erdős-Rényi Theorem. The characterisation we present and its proof can be found in [19].

Paul Erdős was born in Budapest, Hungary on 26 March 1913. He was one of the most prolific mathematicians of the 20th century. Both of Erdős' parents were high school mathematics teachers. He was a mathematics prodigy who, at the age of three, could multiply three-digit numbers in his head. By the age of four, given a person's age, he could calculate in his head how many seconds they had lived. At age 18, he published a much-simplified proof of a theorem of Chebyshev stating that, if $n \geq 2$, then there is a prime between n and $2n$. He obtained a doctorate in mathematics in 1934 from the University of Budapest (today Péter Pázmány University) under the supervision of Lipót Fejér, before embarking on a four-year postdoctoral fellowship at the University of Manchester in England. In September 1938, he accepted a one-year appointment at the Institute for Advanced Study in Princeton in the United States. Thereafter, he became a nomadic academic for half a century, spurning many full-time job offers and wandering from one university to the next to pursue his research endeavours. He co-authored around 1525 journal papers with more than 500 collaborators, which prompted the creation of the so-called Erdős number, the number of steps in the shortest path between a mathematician and Erdős in terms of co-authorships. In 1984, Erdős won the most lucrative award in mathematics, the Wolf Prize, for his outstanding contributions to number theory, combinatorics, probability, set theory and mathematical analysis. He died in Warsaw, Poland on 20 September 1996.



Biographic note 97: Paul Erdős (1913–1996)

For this purpose, we introduce some additional terminology. A vertex of degree k is called a **degree- k vertex**. By **identifying** two distinct vertices u and v in G we mean removing these two vertices from G and replacing them with a new vertex joined to every vertex in $N_G(\{u, v\}) \setminus \{u, v\}$. By **duplicating** the vertex v in G we mean adding a new vertex to the graph G and joining it to every vertex in $N_G(v)$. If G contains a vertex of degree 2, then we define **degree-2 vertex duplication on G** as the operation that produces a new graph from G by duplicating any vertex of degree 2. By **destructively duplicating** the vertex v in G we mean duplicating the vertex with a new vertex v' and then deleting one or more of the edges incident with vertices in $\{v, v'\}$.

We now define two special families of graphs. Let $G_{18.3}$ (depicted in Figure 18.2) be the graph obtained from a 3-cycle by adding a pendent edge to each vertex of the cycle and then introducing a new vertex and joining it to the three resulting vertices of degree 1. Let \mathcal{G} be the family of graphs that: (i) contains C_5 , $G_{18.3}$ and the Petersen graph; and (ii) is closed under degree-2 vertex duplication. (See Figure 18.2.)

A **dominating edge** of G is an edge of G such that every vertex is adjacent to at least one of its endpoints. By the construction of graphs in the family \mathcal{G} , we note that no graph in \mathcal{G} has a dominating edge. Consequently we have the following observation.

Alfréd Rényi was born in Budapest, Hungary on 20 March 1921. He obtained a doctorate in mathematics in 1947 from the University of Szeged, Hungary under the supervision of Frigyes Riesz. In 1949, he was appointed Professor Extraordinary at the University of Debrecen, and in 1950, he founded the Mathematics Research Institute of the Hungarian Academy of Sciences, now known as the Alfréd Rényi Institute of Mathematics. He served as head of the Department of Probability and Mathematical Statistics at the Eötvös Loránd University from 1952 and made significant contributions in combinatorics, graph theory, number theory and probability theory. He was a prolific researcher, having co-authored 224 journal papers. Rényi is the source of two famous quotations. The first is “a mathematician is a device for turning coffee into theorems,” which is generally ascribed to Erdős with whom Rényi wrote 32 joint papers, the best-known of which are his papers introducing the Erdős-Rényi model of random graphs. The second quote is “if I feel unhappy, I do mathematics to become happy. If I am happy, I do mathematics to keep happy.” He died in Budapest, Hungary on 1 February 1970.



Biographic note 98: Alfréd Rényi (1921–1970)

Observation 18.28 *If $G \in \mathcal{G}$ and u and v are two adjacent vertices in G , then the graph obtained from G by introducing a new vertex, w , and adding the edges uw and vw , has diameter 3.*

Since identifying two vertices into one new vertex cannot increase the distance between any two vertices, we also have the following observation.

Observation 18.29 *The diameter of a graph obtained by identifying two vertices into one new vertex is at most the diameter of the original graph.*

The following lemma shows that destructively duplicating a vertex of a graph in the family \mathcal{G} increases its diameter. The proof is left as an exercise (see [Exercise 18.11](#)).

Lemma 18.30 *If $G \in \mathcal{G}$ and G' is a graph obtained from G by destructively duplicating a vertex, then $\text{diam}(G') \geq 3$.*

Recall, from [Chapter 1](#), that the **open neighbourhood** of a set X in a graph G is the set $N_G(X) = \bigcup_{u \in X} N(u)$, and its **closed neighbourhood** is the set $N_G[X] = N(X) \cup X$. Let $N_G^X(v)$ denote the set of neighbours of v in G that belong to the set X and let $d_G^X(v)$ denote the number of vertices in X adjacent to v in G . We say that X **dominates** G if $N[X] = V(G)$. We denote the set of edges that join a vertex of X and a vertex of a set Y in G by $G[X, Y]$, or simply $[X, Y]$ if the graph G is clear from context. For vertices u and v and a subset $X \subseteq V$, we also write $d(v)$, $d(u, v)$, $N(v)$, $N[v]$, $N_X(v)$ and $d_X(v)$ to denote $d_G(v)$, $d_G(u, v)$, $N_G(v)$, $N_G[v]$, $N_G^X(v)$, and $d_G^X(v)$, respectively, if the graph G is clear from the context.

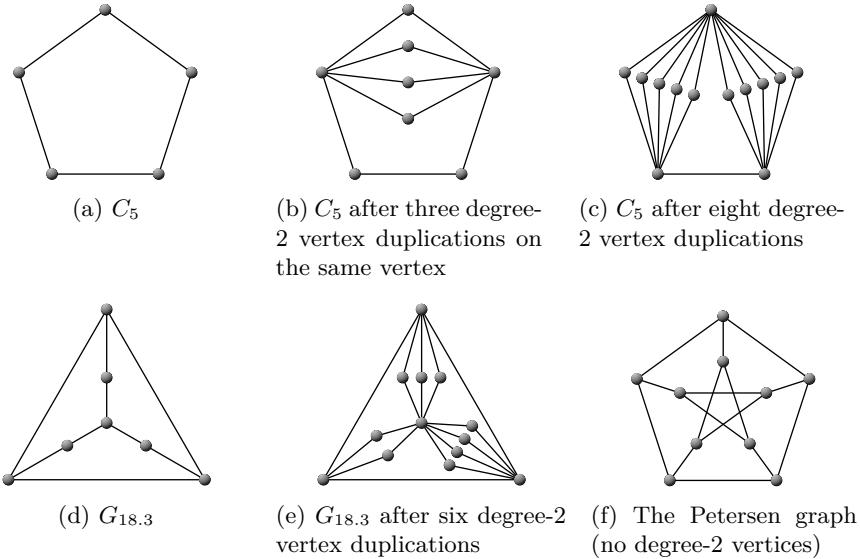


Figure 18.2: Some examples of graphs in the family \mathcal{G} .

We are now in a position to present a proof of the Erdős-Rényi Theorem and to characterise the extremal graphs. In particular, we prove the following result, following the approach in [19].

Theorem 18.31 *If G is a diameter-2 graph of order n and size m with no dominating vertex, then $m \geq 2n - 5$, with equality if and only if $G \in \mathcal{G}$.*

Proof Suppose, to the contrary, that the result of the theorem is false. Among all possible counter examples, let G be one of smallest possible order. Let $V = V(G)$ and $E = E(G)$. Since G is a counter example to our theorem, we have either $m < 2n - 5$ or $m = 2n - 5$ and $G \notin \mathcal{G}$. We proceed further with a series of claims that culminate in a contradiction.

Claim 18.32 $\Delta(G) \leq n - 3$.

Proof Since there is no dominating vertex in G , we have $\Delta(G) \leq n - 2$. Suppose $\Delta(G) = n - 2$. Let $v \in V$ be a vertex of maximum degree in G , let $W = N(v) = \{w_1, \dots, w_{n-2}\}$ and let x be the vertex not adjacent to v . For each $i \in [n-2]$, we let $e_i = vw_i$ and define f_i as follows: If $w_i x \in E$, then $f_i = w_i x$; otherwise $f_i = w_i w$, where w is a common neighbour of w_i and x . We note that each f_i is unique. Now, $m \geq |\{e_1, \dots, e_{n-2}, f_1, \dots, f_{n-2}\}| = 2n - 4$, a contradiction. Hence, $\Delta(G) \leq n - 3$. \blacksquare

Claim 18.33 *Suppose there exists a vertex $x \in V$ such that $d(x) = 2$. Let $N(x) = \{x_1, x_2\}$ and $G' = G - x$. Then, $d_{G'}(x_1, x_2) \geq 3$.*

Proof Suppose, to the contrary, that $d_{G'}(x_1, x_2) \leq 2$. Let $n' = n(G')$ and $m' = m(G')$. If G' has a dominating vertex, then such a vertex would have degree at least $n - 2$ in G , contradicting Claim 18.32. Hence, G' has no dominating vertex. Furthermore, since $d_{G'}(x_1, x_2) \leq 2$, we have $\text{diam}(G') \leq \text{diam}(G) = 2$. But, since G' has no dominating vertex, it follows that G' is not a complete graph, and

so $\text{diam}(G') \geq 2$. Consequently, $\text{diam}(G') = 2$. Now, since G is a minimum counter example to our theorem and $n' = n - 1$, it follows that $m' \geq 2n' - 5$, with equality if and only if $G' \in \mathcal{G}$. By our choice of G , we recall that $m \leq 2n - 5$, and so

$$m' = m - 2 \leq (2n - 5) - 2 = 2n' - 5.$$

Hence, $m' = 2n' - 5$ and $G' \in \mathcal{G}$. If $d_{G'}(x_1, x_2) = 1$, then $x_1x_2 \in E'$ and G can be obtained from G' by adding the vertex x as well as the edges xx_1 and xx_2 . But then we have, by [Observation 18.28](#), that $\text{diam}(G) = 3$, a contradiction. Therefore, $d_{G'}(x_1, x_2) = 2$. Let $x' \in N_{G'}(x_1) \cap N_{G'}(x_2)$. If $d_{G'}(x') = 2$, then G can be obtained from G' by duplicating x' , and so $G \in \mathcal{G}$, a contradiction. Hence $d_{G'}(x') \geq 3$. But now G can be obtained from G' by destructively duplicating x' and by [Lemma 18.30](#) we have that $\text{diam}(G) \geq 3$, a contradiction. ■

Claim 18.34 $\delta(G) = 3$.

Proof We note first that if $\delta(G) \geq 4$, then the sum of the degrees of the vertices in G is at least $4n$ and so $m \geq 2n$, a contradiction. Hence, $\delta(G) \leq 3$. Let x be a vertex of minimum degree in G . Since $\text{diam}(G) = 2$, certainly G is connected, and so $d(x) \in \{1, 2, 3\}$. If $d(x) = 1$, then the diameter two constraint implies that the neighbour of x in G is necessarily a dominating vertex in G , a contradiction. Hence, $d(x) \in \{2, 3\}$.

Suppose that $d(x) = 2$ and let $N(x) = \{x_1, x_2\}$. By [Claim 18.33](#), we have $x_1x_2 \notin E$ and $N(x_1) \cap N(x_2) = \{x\}$. Now, for each $u \in V \setminus \{x, x_1, x_2\}$ it follows that $d(u, x) \leq 2$, and so u is adjacent to either x_1 or x_2 (but not both). For $i \in \{1, 2\}$, we let $X_i = N(x_i) \setminus \{x\}$ and observe that $V = X_1 \cup X_2 \cup \{x, x_1, x_2\}$ and $|X_1| + |X_2| = n - 3$. If $d(u) \geq 3$ for all $u \in X_1 \cup X_2$, then

$$\begin{aligned} m &= \frac{1}{2} \left(\overbrace{d(x) + d(x_1) + d(x_2)}^{\sum_{u \in V} d(u)} + \sum_{u \in X_1} d(u) + \sum_{u \in X_2} d(u) \right) \\ &\geq \frac{1}{2} \left(2 + (|X_1| + 1) + (|X_2| + 1) + 3|X_1| + 3|X_2| \right) \\ &= 2(|X_1| + |X_2| + 1) = 2n - 4, \end{aligned}$$

a contradiction. Hence, some vertex in $X_1 \cup X_2$ has degree 2 in G . Relabelling vertices if necessary, we may assume there exists a $y_1 \in X_1$ such that $d(y_1) = 2$. Let $N(y_1) = \{x_1, y_2\}$ and note that since $d(x_2, y_1) \leq 2$, we must have $y_2 \in X_2$.

Suppose $d(y_2) = 2$. If there exists a $z \in (X_1 \cup X_2) \setminus \{y_1, y_2\}$, then necessarily $d(y_1, z) = d(y_2, z) = 2$ and consequently $z \in N(x_1) \cap N(x_2) = \{x\}$, a contradiction. Hence, $|X_1| = |X_2| = 1$ and $G = C_5 \in \mathcal{G}$, another contradiction. We conclude that $d(y_2) \geq 3$. Let $z_2 \in N(y_2) \setminus \{x_2, y_1\}$. Since $d(y_1) = 2$, we have, by [Claim 18.33](#), that $d_{G-y_1}(x_1, y_2) \geq 3$, implying that $z_2 \notin X_1$. Consequently, $z_2 \in X_2$. Now, $d(x_1, z_2) \leq 2$ and $x_1z_2 \notin E$. Therefore, there exists a $z_1 \in X_1 \setminus \{y_1\}$ such that $z_1 \in N(x_1) \cap N(z_2)$. Since $d(y_1) = 2$, we have, by [Claim 18.33](#), that $y_2z_1 \notin E$. For $i \in \{1, 2\}$, we now let $X'_i = X_i \setminus \{y_i, z_i\}$. We observe that $n = |X'_1| + |X'_2| + 7$. Let $m' = m(G[\{x, x_1, x_2, y_1, y_2, z_1, z_2\}])$ and note that $m' = 9$.

Suppose $d(z_1) = 2$. Then, since $d(x) = d(y_1) = d(z_1) = 2$, we have, by [Claim 18.33](#), that no vertex in X'_1 is adjacent to any vertex in $\{x_2, y_2, z_2\}$. Furthermore, since $\text{diam}(G) = 2$ it follows that $\{x_1, x_2\}$, $\{x_1, y_2\}$ and $\{x_1, z_2\}$ each

dominates V . Consequently, if $u \in X'_2$, then $ux_1 \notin E$ and so $\{ux_2, uy_2, uz_2\} \subseteq E$. Hence, $|[X'_2, \{x_2, y_2, z_2\}]| = 3|X'_2|$. Also, if $u \in X'_1$, then $ux_1 \in E$ and since $d(u, x_2) \leq 2$, we have that $uu' \in E$ for some $u' \in X'_2$. Hence, $|[X'_1, X'_2 \cup \{x_1\}]| \geq 2|X'_1|$ and we note that if $X'_1 \neq \emptyset$, then necessarily $X'_2 \neq \emptyset$. Equivalently, if $X'_2 = \emptyset$, then $X'_1 = \emptyset$. Counting edges in G , it therefore follows that

$$\begin{aligned} m &\geq |[X'_1, X'_2 \cup \{x_1\}]| + |[X'_2, \{x_2, y_2, z_2\}]| + m' \\ &\geq 2|X'_1| + 3|X'_2| + 9 \\ &= 2(|X'_1| + |X'_2| + 7) - 5 + |X'_2| = 2n - 5 + |X'_2|. \end{aligned}$$

However, $m \leq 2n - 5$, implying that $m = 2n - 5$ and $X'_2 = \emptyset$. This, in turn, implies that $X'_1 = \emptyset$. But now $G = G_{18.3} \in \mathcal{G}$, a contradiction. Hence, $d(z_1) \geq 3$.

Let e be an edge incident with z_1 different from x_1z_1 and z_1z_2 . Now, since $d(x) = d(y_1) = 2$ we have, by [Claim 18.33](#), that no vertex in X'_1 is adjacent to either x_2 or y_2 . Furthermore, since $\text{diam}(G) = 2$ we note that $\{x_1, x_2\}$ and $\{x_1, y_2\}$ both dominate V . Consequently, if $u \in X'_2$, then $ux_1 \notin E$, and so $\{ux_2, uy_2\} \subseteq E$. Hence, $|[X'_2, \{x_2, y_2\}]| = 2|X'_2|$. Also, if $u \in X'_1$, then $ux_1 \in E$ and since $d(u, x_2) \leq 2$ we have that $uu' \in E$ for some $u' \in X'_2 \cup \{z_2\}$. Hence, $|[X'_1, X'_2 \cup \{x_1, z_2\}]| \geq 2|X'_1|$. Counting edges in G , it now follows that

$$\begin{aligned} m &\geq |[X'_1, X'_2 \cup \{x_1, z_2\}]| + |[X'_2, \{x_2, y_2\}]| + m' + |\{e\}| \\ &\geq 2|X'_1| + 2|X'_2| + 10 = 2(|X'_1| + |X'_2| + 7) - 4 = 2n - 4, \end{aligned}$$

a contradiction. The desired result therefore follows. \blacksquare

It follows from [Claim 18.34](#) that $\delta(G) = 3$. Let u be a vertex of minimum degree in G , let $U = N(u) = \{u_1, u_2, u_3\}$ and let $W = V \setminus N[u]$.

Claim 18.35 *If $w \in W$, then $d(w) = 3$ and $d_U(w) = 1$. Furthermore, U is an independent set.*

Proof We note that for each $w \in W$ we have $d(u, w) = 2$ and so $d_U(w) \geq 1$. Consequently, $|[U, W]| \geq |W| = n - 4$, with equality if and only if $d_U(w) = 1$ for each $w \in W$. Let m' be the number of edges in the subgraph induced by U . Summing the degrees of vertices in U by considering first the three edges incident with u , then the m' edges in $G[U]$ and finally the edges between U and W , it follows that

$$\sum_{v \in U} d(v) = d_U(u) + 2m' + |[U, W]| \geq 3 + 2m' + (n - 4) = n - 1 + 2m', \quad (18.4)$$

with equality if and only if $|[U, W]| = n - 4$, which holds, as already noted, if and only if $d_U(w) = 1$ for all $w \in W$. Summing the degrees of vertices in W , it follows that

$$\sum_{v \in W} d(v) \geq \delta(G)|W| = 3(n - 4) = 3n - 12, \quad (18.5)$$

with equality if and only if $d(w) = 3$ for each $w \in W$. Recall that $m \leq 2n - 5$ and hence

$$\begin{aligned} 4n - 10 &\geq 2m = \sum_{v \in V} d(v) = d(u) + \sum_{v \in U} d(v) + \sum_{v \in W} d(v) \\ &\geq 3 + (n - 1 + 2m') + (3n - 12) = 4n - 10 + 2m'. \end{aligned}$$

Since $m' \geq 0$, we have $m' = 0$ and equality throughout the above inequality chain. Consequently, equality is achieved throughout (18.4) and (18.5) and so $d(w) = 3$ and $d_U(w) = 1$ for all $w \in W$, as desired. Furthermore, since $m' = 0$, it follows that U is an independent set. ■

For $i \in [3]$ let $W_i = N_W(u_i)$. By Claim 18.35 it follows that W_1 , W_2 and W_3 are pairwise disjoint. Furthermore, since U is independent and since $\delta(G) = 3$, it follows that $|W_i| \geq 2$ for each $i \in \{1, 2, 3\}$. Let $w_1 \in W_1$. For $i \in \{2, 3\}$ we have $u_i w_1 \notin E$ and so w_1 necessarily has a neighbour, x_i say, in W_i . Note that $N(w_1) = \{u_1, x_2, x_3\}$. Let $w_2 \in W_2 \setminus \{x_2\}$. Now, $w_2 \notin N(w_1)$ and hence $d(w_1, w_2) = 2$. Consequently, w_2 is adjacent to a vertex in $N(w_1) = \{u_1, x_2, x_3\}$. Clearly, $u_1 w_2 \notin E$. If $w_2 x_2 \in E$, then $N(x_2) = \{u_2, w_1, w_2\}$ and we have $d(u_3, x_2) = 3$, a contradiction. Hence, $w_2 x_3 \in E$. We note now that $N(x_3) = \{u_3, w_1, w_2\}$. Since $u_3 x_2 \notin E$, we have $d(u_3, x_2) = 2$. Let $w_3 \in N(u_3) \cap N(x_2)$ and note that $w_3 \notin \{u, x_3\}$. Similarly, $u_1 w_2 \notin E$ and we let $x_1 \in N(u_1) \cap N(w_2)$, noting that $x_1 \notin \{u, w_1\}$. We now have $N(x_2) = \{u_2, w_1, w_3\}$ and $N(w_2) = \{u_2, x_1, x_3\}$. Since $x_1 x_2 \notin E$ it follows that $d(x_1, x_2) = 2$ and so necessarily $x_1 w_3 \in E$.

Suppose $d(u_i) \geq 4$ for some $i \in [3]$. Relabelling vertices if necessary, we may assume $d(u_1) \geq 4$. Let $y_1 \in W_1 \setminus \{w_1, x_1\}$. Now $y_1 x_2 \notin E$ and y_1 is not adjacent to a vertex in $N(x_2) = \{u_2, w_1, w_3\}$ since $y_1 u_2 \notin E$, $N(w_1) = \{u_1, x_2, x_3\}$ and $N(w_3) = \{u_3, x_1, x_2\}$. But then $d(y_1, x_2) > 2$, a contradiction. Therefore, $d(v) = 3$ for all $v \in V$ and G is the Petersen graph. But then $G \in \mathcal{G}$, another a contradiction. We conclude that G does not exist, which completes the proof of Theorem 18.31. ■

❖ The reader should now be able to attempt Exercise 18.11.

18.7 Diameter two critical graphs of maximum size

In this section, we again consider diameter-2 graphs. We focus our attention on a long-standing conjecture that proposes an upper bound on the number of edges in a diameter-2 graph having the property that the removal of an arbitrary edge increases its diameter. Such graphs are called **diameter-2-critical graphs**. Conventionally the diameter of a disconnected graph is considered to be either undefined or defined as infinity. Here we adopt the former convention and hence require that a diameter-2-critical graph have minimum degree two (since removing an edge incident with a vertex of degree one results in a disconnected graph). We note, however, that if we relaxed the condition and defined the diameter of disconnected graphs to be infinity, the only additional diameter-2-critical graphs are stars of order at least 3.

Murty and Simon independently posed the following conjecture [3]:

Conjecture 18.36 (Murty-Simon Conjecture) *If G is a diameter-2-critical graph with n vertices and m edges, then $m \leq \lfloor n^2/4 \rfloor$, with equality if and only if G is the complete bipartite graph $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$.*

The **Murty-Simon Conjecture** has been in existence for a half century and has attracted the attention of many leading mathematicians in the field of extremal graph theory. According to Füredi [9], Erdős claimed that this conjecture goes back to the work of Ore during the 1960s. Although there have been many attempts to solve it, a complete solution has not been found to date. Fan [8] proved

that the inequality in the conjecture holds for $n \leq 24$ and for $n = 26$. Füredi [9] gave an asymptotic result proving that the inequality in the conjecture holds for large n , that is, for $n > n_0$ where n_0 is a tower of 2's of height about 10^{14} . The conjecture has, however, yet to be proven for all other values of n .

Zoltán Füredi was born in Budapest, Hungary on 21 May 1954. In 1972, he won the International Mathematical Olympiad with a perfect score. Since 1978 he has held an appointment at the Rényi Mathematical Institute of the Hungarian Academy of Sciences. He obtained a Candidate of Sciences degree (the Hungarian equivalent of a doctorate) in mathematics in 1981 from the Hungarian Academy of Sciences under the supervision of Gyula OH Katona based on a dissertation entitled *Extremal hypergraphs and finite geometries*. He was appointed as an associate professor at the Massachusetts Institute of Technology in 1990, before accepting a professorship in mathematics at the University of Illinois at Urbana-Champaign in 1991. Füredi has made outstanding contributions in combinatorics, mainly in discrete geometry and extremal combinatorics. He is currently a research professor of the Rényi Mathematical Institute of the Hungarian Academy of Sciences, and also a professor at the University of Illinois Urbana-Champaign. He is a prolific researcher.



Biographic note 99: Zoltán Füredi (1954–present)

We discuss in this section a new way to consider the conjecture from the point of view of total domination in graphs, a concept that we considered in Section 16.5. Although this concept is seemingly unrelated to that of diameter-2-criticality in graphs, we show that the Murty-Simon Conjecture is, in fact, equivalent to a conjecture involving total domination.

Recall from Section 16.5.9 that a graph G is **total domination edge critical** if $\gamma_t(G - e) < \gamma_t(G)$ for every edge $e \in E(\bar{G}) \neq \emptyset$. Furthermore, if $\gamma_t(G) = k$, then G is a **k_t -critical graph**. A k_t -critical graph with the property that $\gamma_t(G - e) = k - 2$ for every edge $e \in E(\bar{G})$ is a **k_t -supercritical graph**. Hanson and Wang [10] were the first to observe the following key relationship between diameter-2-critical graphs and total domination edge critical graphs. Note that this relationship is contingent on total domination being defined in the complement \bar{G} of the diameter-2-critical graph G , that is, \bar{G} has no isolated vertices. Hence, our elimination of stars as diameter-2-graphs is necessary for this association. We leave a proof of Theorem 18.37 as an exercise (see Exercise 18.12).

Theorem 18.37 (Hanson-Wang Theorem) *A graph is diameter-2-critical if and only if its complement is 3_t -critical or 4_t -supercritical.*

A characterisation of 4_t -supercritical graphs is a special case of the characterisation of k_t -supercritical graphs given in Theorem 16.118 of Section 16.5. This characterisation was first proven in [17].

Theorem 18.38 A graph G is 4_t -supercritical if and only if G is the disjoint union of two nontrivial complete graphs.

To illustrate Theorem 18.37, consider the graph $G_{18.4}$ that is presented in Figure 18.3(a). The graph is diameter-2-critical. Therefore, by Theorems 18.37 and 18.38, its complement, $\bar{G}_{18.4}$, is 3_t -critical.



Figure 18.3: A graph $G_{18.4}$ and its complement $\bar{G}_{18.4}$.

By Theorem 18.38, the complement of a 4_t -supercritical graph is a complete bipartite graph with both partite sets of cardinality at least two. A simple calculus argument shows that the size of a complete bipartite graph G is maximised when the partite sets differ in cardinality by at most one; that is, when the partite sets have orders $\lfloor \frac{n}{2} \rfloor$ and $\lceil \frac{n}{2} \rceil$, respectively, where n is the order of G . Hence, the Murty-Simon Conjecture holds for this case and a subset of the complements of 4_t -supercritical graphs yield the extremal graphs of the conjecture.

Suppose, therefore, that G is a diameter-2-critical graph which is not a complete bipartite graph. Then G is not the complement of a 4_t -supercritical graph, implying by Theorem 18.37 that G is the complement of a 3_t -critical graph. Therefore, by Theorems 18.37, 18.38 and the above observations, the Murty-Simon Conjecture is equivalent to the following conjecture.

Conjecture 18.39 If G is a 3_t -critical graph of order n and size m , then

$$m > \binom{n}{2} - \left\lfloor \frac{n^2}{4} \right\rfloor = \left\lceil \frac{n(n-2)}{4} \right\rceil.$$

Proving Conjecture 18.39, and hence the Murty-Simon Conjecture, is equivalent to determining the minimum number of edges in 3_t -critical graphs. This remains a challenging problem in extremal graph theory.

Recall that in Theorem 16.119, we showed that the diameter of a 3_t -critical graph is either two or three. Conjecture 18.39 has been proven for several special classes of graphs. We summarise the graph classes for which Conjecture 18.39 is known to hold in Table 18.2.

We prove that Conjecture 18.39 holds for bull-free graphs, but we omit the proofs that Conjecture 18.39 holds for the other special classes of graphs listed in Table 18.2.

A **bull graph** consists of a triangle with two disjoint pendant edges, as illustrated in Figure 18.4. A graph is **bull-free** if it has no bull as an induced subgraph. We observe that the bull is a self-complementary graph.

There are numerous infinite families of bull-free diameter-2-critical graphs, as noted in [1]. The simplest such family consists of expansions of a 5-cycle, where an **expansion** of a graph G is the graph obtained from G by replacing each vertex x with an independent set X and adding all edges between the sets X and Y that

Conjecture 18.39 holds for the following graph classes	
Graphs with diameter three	[13]
Claw-free graphs	[15]
Connectivity-1 graphs	[13]
Connectivity-2 graphs	[16]
Connectivity-3 graphs	[16]
Bull-free graphs	[1]
Diamond-free graphs	[11]
C_4 -free graphs	[12]
House-free graphs	[1]

Table 18.2: Graph classes for which Conjecture 18.39 holds.

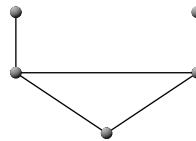


Figure 18.4: The bull graph, $G_{18.5}$.

correspond to adjacent vertices x and y in G . A 5-cycle and an expansion of a 5-cycle are illustrated in Figures 18.5(a) and (b), respectively.

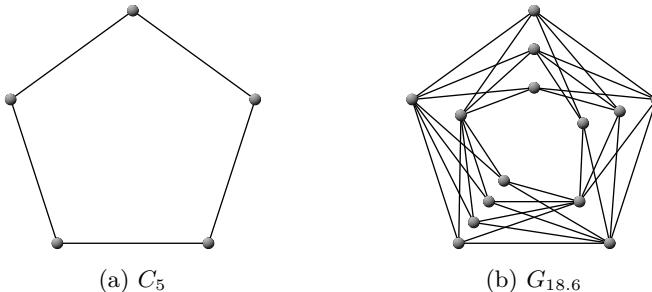


Figure 18.5: A 5-cycle and an expansion $G_{18.6}$ of a 5-cycle.

Before proving that Conjecture 18.39 holds for bull-free graphs, we first present a useful property of 3_t -critical graphs. Let G be a 3_t -critical graph. Then $\gamma_t(G) = 3$, implying that G is not complete and, therefore, $E(\bar{G}) \neq \emptyset$. Furthermore, $\gamma_t(G + e) = 2$ for every edge $e \in E(\bar{G})$. Therefore, we have the following observation, first observed in [13].

Observation 18.40 *For every 3_t -critical graph G and nonadjacent vertices u and v in G , either the set $\{u, v\}$ is a dominating set of G or, without loss of generality, the set $\{u, w\}$ is a dominating set of $G - v$ for some vertex w adjacent to u but not to v in G .*

An independent set of size three is called a **triad**. A graph is **triad-free** if it contains no triad. In order to prove that Conjecture 18.39 holds for bull-free graphs, it suffices to prove the following result [1].

Theorem 18.41 *Let G be a 3_t -critical graph or a 4_t -supercritical graph. Then G is bull-free if and only if G is triad-free.*

Proof Let G be a 3_t -critical or 4_t -supercritical. If G is 4_t -supercritical, then by [Theorem 18.38](#), G is the disjoint union of two nontrivial complete graphs. Therefore, in this case, G is both bull-free and triad-free. Hence, we may assume that G is a 3_t -critical graph, for otherwise the desired result is immediate. If G is triad-free, then G is bull-free since every induced bull contains a triad. Hence, it suffices to prove that if G is bull-free, then G is triad-free.

Let G be a bull-free, 3_t -critical graph. Suppose, to the contrary, that G has a triad, say $\{x, y, z\}$. The 3_t -criticality of G implies that $\gamma_t(G + e) = 2$ for every edge $e \in E(\bar{G})$. Taking $e = xy \in E(\bar{G})$, we have that $\gamma_t(G + xy) = 2$. Since the set $\{x, y\}$ is not a dominating set in G , we may assume by [Observation 18.40](#), and renaming the vertices x and y if necessary, that the set $\{w, x\}$ is a dominating set of $G - y$ for some vertex w adjacent to x but not to y in G . In order to dominate the vertex z , we note that w is adjacent to z in G .

Next we consider adding the edge $yz \in E(\bar{G})$ to G . By the 3_t -criticality of G , we have that $\gamma_t(G + yz) = 2$. If $\{a, y\}$ is a dominating set of $G - z$ for some vertex a adjacent to y but not to z in G , then such a vertex a is adjacent to both x and w . But then the set $\{a, x, y, w, z\}$ induces a bull in G , contradicting our assumption that G is bull-free. Moreover, the set $\{y, z\}$ is not a dominating set in G . Hence, by [Observation 18.40](#) we deduce that the set $\{a, z\}$ is a dominating set of $G - y$ for some vertex a adjacent to z but not to y in G . In order to dominate the vertex x , we have that the vertices a and x are adjacent in G . We remark that $a \notin \{x, y, z\}$, although possibly $a = w$.

By the 3_t -criticality of G , we have that $\gamma_t(G + xz) = 2$. Since the set $\{x, z\}$ is not a dominating set of G , we have, by [Observation 18.40](#), that the set $\{b, x\}$ is a dominating set of $G - z$ for some vertex b adjacent to x but not to z in G or that the set $\{b, z\}$ is a dominating set of $G - x$ for some vertex b adjacent to z but not to x in G . By symmetry, we may assume, without loss of generality, that $\{b, x\}$ is a dominating set of $G - z$ for some vertex b adjacent to x but not to z in G . Since the vertex w is adjacent to the vertex z , we note that $b \neq w$. Analogously, we note that $a \neq b$. In order to dominate the vertex y , we have that the vertices b and y are adjacent in G . Since $\{a, z\}$ is a dominating set in $G - y$, the vertices a and b are adjacent in G . But then the set $\{a, b, x, y, z\}$ induces a bull in G , contradicting our assumption that G is bull-free. Therefore, G is triad-free, as desired. ■

As observed earlier, the bull is a self-complementary graph. Recall that a clique of size three is called a **triangle** and a graph is **triangle-free** if it contains no triangle. Hence a graph G is triad-free if and only if its complement \bar{G} is triangle-free. As an immediate consequence of Theorems [18.37](#) and [18.41](#) we, therefore, have the following result.

Theorem 18.42 *Let G be a diameter-2-critical graph. Then G is bull-free if and only if G is triangle-free.*

As an immediate consequence of Theorems [18.7](#) and [18.42](#), we have the following result.

Theorem 18.43 *The Murty-Simon Conjecture is true for bull-free graphs.*

The [Murty-Simon Conjecture](#) is equivalent to [Conjecture 18.39](#), as observed earlier. Therefore, [Theorem 18.43](#) implies that [Conjecture 18.39](#) also holds for bull-free graphs.

❖ The reader should now be able to attempt [Exercise 18.12](#).

Exercises

18.1 Prove that $\text{ex}(n; P_3) = \lfloor \frac{n}{2} \rfloor$ for all $n \geq 3$ and describe the graphs that are extremal for n and P_3 .

18.2 Prove that $\text{ex}(n; C_n) = \binom{n-1}{2} + 1$ for all $n \geq 3$.

18.3 Prove [Jensen's inequality](#).

18.4 Determine $\text{ex}(n; K_{1,3})$ for any $n \geq 4$ and describe the graphs that are extremal for n and $K_{1,3}$.

18.5 Prove that

$$t_{n,r} \leq \frac{1}{2} \left(\frac{r-1}{r} \right) n^2$$

for all integers $n \geq r \geq 1$.

18.6 Prove [Theorem 18.11](#).

18.7 Show that $g(1, n) = g(2, n) = 2n$ for all $n \geq 1$.

18.8 Show that $g(m, 1) = m$ for all $m \geq 2$.

18.9 Prove [Lemma 18.18](#).

18.10 Use [Lemma 18.18](#) to prove [Lemma 18.19](#).

18.11 Prove [Lemma 18.30](#).

18.12 Prove [Theorem 18.37](#).

Further reading

- [1] C Balbuena, A Hansberg, TW Haynes and MA Henning, 2015. *On total domination edge critical graphs with total domination number three and with many dominating pairs*, Graphs and Combinatorics, **31**(5), pp. 1163–1176.
- [2] B Bollobás, 1995. *Extremal graph theory*, pp. 1231–1292 in RL Graham, M Grötschel and L Lovász (Eds), *Handbook of Combinatorics*, Elsevier Science, Amsterdam.
- [3] L Caccetta and R Häggkvist, 1979. *On diameter critical graphs*, Discrete Mathematics, **28**(3), pp. 223–229.
- [4] G Chartrand, H Gavlas and M Schultz, 1992. *FRAMED! A graph embedding problem*, Bulletin of the Institute of Combinatorics and its Applications, **4**, pp. 35–50.
- [5] FRK Chung and RL Graham, 1975. *On multicolor Ramsey numbers for complete bipartite graphs*, Journal of Combinatorial Theory, **18**, pp. 164–169.

- [6] RC Entringer, W Goddard and MA Henning, 1997. *A note on cliques and independent sets*, Journal of Graph Theory, **24**, pp. 21–23.
- [7] P Erdős and A Rényi, 1962. *On a problem in the theory of graphs (Hungarian. Russian, English summaries)*, Magyar Tud. Akad. Mat. Kutató Int. Közl, **7**, pp. 623–641.
- [8] G Fan, 1987. *On diameter 2-critical graphs*, Discrete Mathematics, **67**, pp. 235–240.
- [9] Z Füredi, 1992. *The maximum number of edges in a minimal graph of diameter 2*, Journal of Graph Theory, **16**, pp. 81–98.
- [10] D Hanson and P Wang, 2003. *A note on extremal total domination edge critical graphs*, Utilitas Mathematica, **63**, pp. 89–96.
- [11] TW Haynes and MA Henning, 2012. *A characterization of diameter-2-critical graphs whose complements are diamond-free*, Discrete Applied Mathematics, **160**, pp. 1979–1985.
- [12] TW Haynes and MA Henning, 2012. *A characterization of diameter-2-critical graphs with no antihole of length four*, Central European Journal of Mathematics, **10(3)**, pp. 1125–1132.
- [13] TW Haynes, MA Henning, LC van der Merwe and A Yeo, 2011. *On a conjecture of Murty and Simon on diameter two critical graphs*, Discrete Mathematics, **311**, pp. 1918–1924.
- [14] TW Haynes, MA Henning, LC van der Merwe and A Yeo, 2011. *On the existence of k -partite or K_p -free total domination edge-critical graphs*, Discrete Mathematics, **311**, pp. 1142–1149.
- [15] TW Haynes, MA Henning and A Yeo, 2011. *A proof of a conjecture on diameter two critical graphs whose complements are claw-free*, Discrete Optimization, **8**, pp. 495–501.
- [16] TW Haynes, MA Henning and A Yeo, 2012. *On a conjecture of Murty and Simon on diameter two critical graphs II*, Discrete Mathematics, **312**, pp. 315–323.
- [17] TW Haynes, CM Mynhardt and LC van der Merwe, 1998. *Criticality index of total domination*, Congressus Numerantium, **131**, pp. 67–73.
- [18] MA Henning, 2000. *On cliques and bicliques*, Journal of Graph Theory, **34**, pp. 60–66.
- [19] MA Henning and J Souley, 2015. *A characterisation of non-trivial diameter two graphs of minimum size*, Discrete Applied Mathematics, **187**, pp. 91–95.
- [20] W Mantel, 1907. *Problem 28, solution by H Gouwentak, A Mantel, J Teixeira de Mattes, F Schuh and WA Wythoff*, Wiskundige Opgaven, **10**, pp. 60–61.
- [21] LC van der Merwe, 1998. *Total domination edge critical graphs*, PhD thesis, University of South Africa, Pretoria.
- [22] LC van der Merwe, TW Haynes and CM Mynhardt, 1998. *Total domination edge critical graphs*, Utilitas Mathematica, **54**, pp. 229–240.

- [23] J Plesník, 1975. *Critical graphs of given diameter*, Acta F.R.N. Univ Comen Math. **30**, pp. 71–93.
- [24] P Turán, 1941. *Eine Extremalaufgabe aus der Graphentheorie*, Mat. Fiz. Lapok, **48**, pp. 436–452.
- [25] K Zarankiewicz, 1951. *Problem P101*, Colloquium Mathematicum, **2**, p. 301.



Graph enumeration

Contents

19.1	Counting labelled graphs	681
19.2	Counting unlabelled trees	692
19.3	Pólya's enumeration theorem	700
19.4	Using Pólya's theorem to enumerate graphs	708
	Exercises	714
	Computer exercises	717
	Projects	720
	Further reading	722

The subject of graph enumeration involves counting the number of distinct graphs satisfying some property (such as determining the number of graphs of order n , the number of trees of order n , the number of multigraphs of order n , or the number of digraphs of order n), without having to identify or draw all these graphs explicitly — which would be a tedious process indeed for even moderately large values of n . In this chapter, we develop the necessary combinatorial tools for carrying out such enumerative analyses. These tools include recurrence relations, generating functions and results from basic group theory.

19.1 Counting labelled graphs

In this section we are interested in counting or enumerating so-called *labelled graphs* of various structures, that is, graphs in which all the vertices are distinguishable and which satisfy some property. We may think of a labelled graph of order n as a graph in which the vertices have been labelled or named v_1, v_2, \dots, v_n — the graph vertices are therefore distinguishable by their names or labels. The graphs $G_{19.1}$ and $G_{19.2}$ in Figure 19.1, for example, are distinct when viewed as labelled graphs, but of course not when viewed as unlabelled graphs, because they are isomorphic. It is, in general, much harder to enumerate unlabelled graphs satisfying some property than to enumerate labelled graphs satisfying the same property, because of the isomorphisms involved in the former case.



Figure 19.1: Two graphs $G_{19.1}$ and $G_{19.2}$ which are distinct when viewed as labelled graphs, but which are isomorphic when viewed as unlabelled graphs.

19.1.1 Labelled simple graphs of specified order and size

Let us start by considering the following variations on perhaps the most basic graph enumeration question: How many distinct labelled graphs of order n (and size m) are there?

Throughout this chapter the binomial coefficient $\binom{a}{b}$ should be interpreted as $a!/(b!(a-b)!)$ in the usual manner for any nonnegative integers a, b satisfying $a \geq b$, but following the convention that $\binom{a}{b} = 0$ if $a < b$.

Theorem 19.1 *There are $\binom{n}{m}$ distinct labelled graphs of order $n \in \mathbb{N}$ and size $m \in \mathbb{N}$.*

Proof Pairs of vertices may be chosen in $\binom{n}{2}$ different ways from a set of n distinguishable vertices. Of these $\binom{n}{2}$ distinct potential “positions” for edges in an (m, n) labelled graph, exactly m “positions” have to be chosen to draw in edges, and this can be done in $\binom{\binom{n}{2}}{m}$ different ways. ■

The number of labelled graphs of order n and size m grows very rapidly as m and n increase, as shown in [Table 19.1](#). It follows from the table that the total number of distinct labelled graphs of orders 1, 2, 3 and 4 are 1, 2, 8 and 64, respectively. The next theorem extends this sequence¹ beyond $n = 4$.

Theorem 19.2 *There are $2^{\binom{n}{2}}$ distinct labelled graphs of order $n \in \mathbb{N}$.*

Proof Pairs of vertices may be chosen in $\binom{n}{2}$ different ways from a set of n distinguishable vertices. For each of these potential “positions” for edges, the choice must be made as to whether or not an edge is drawn in the particular “position.” Since these choices are independent,

$$\underbrace{2 \times 2 \times \cdots \times 2}_{\binom{n}{2} \text{ factors}}$$

labelled graphs can result. ■

The result of [Theorem 19.2](#) may also be deduced from [Theorem 19.1](#) by invoking the well-known binomial theorem. By considering all possible values of m in [Theorem 19.1](#), it follows that the number of distinct labelled graphs of order n is

$$\sum_{m=0}^{\binom{n}{2}} \binom{\binom{n}{2}}{m} = \sum_{m=0}^{\binom{n}{2}} \binom{\binom{n}{2}}{m} 1^m 1^{\binom{n}{2}-m} = (1+1)^{\binom{n}{2}}.$$

The number of labelled graphs of order n therefore grows exponentially as a function of n . This number is shown in the second line of [Table 19.2](#) for $n \in [9]$.

¹Sequence [A006125](#) of Sloane [10].

n	m	$\binom{\binom{n}{2}}{m}$	Labelled graphs	
1	0	1	•	{ 1 }
2	0	1	• •	{ 2 }
2	1	1	•—•	
3	0	1	•	{ }
3	1	3	•—• •—•—• •—•—•	{ 8 }
3	2	3	•—•—• •—•—• •—•—•	
3	3	1	•—•—•	
4	0	1	•	{ }
4	1	6	•—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—•	
4	2	15	•—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—•	{ 64 }
4	3	20	•—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—•	
4	4	15	•—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—•	
4	5	6	•—•—•—• •—•—•—• •—•—•—• •—•—•—• •—•—•—•	
4	6	1	•—•—•—•	

Table 19.1: The number of labelled graphs of order $n \in [4]$ and size $m \in \{0, 1, \dots, \binom{n}{2}\}$. Although the graph labels are omitted in the table to avoid clutter, vertices in fixed positions relative to other vertices should be interpreted as corresponding to fixed labels. The same drawing convention is adopted in the remainder of Section 19.1.1.

n	1	2	3	4	5	6	7	8	9
$2^{\binom{n}{2}}$	1	2	8	64	1 024	32 768	2 097 152	268 435 456	68 719 476 736
Θ_n	1	1	4	38	—	Exercise	—	251 548 592	66 296 291 072
$2^{\binom{n}{2}} - \Theta_n$	0	1	4	26	—	Exercise	—	16 886 864	2 423 185 664

Table 19.2: The number $2^{\binom{n}{2}}$ of labelled graphs of order $n \in [9]$, the number Θ_n of connected labelled graphs of order $n \in [9]$, and the number $2^{\binom{n}{2}} - \Theta_n$ of disconnected labelled graphs of order $n \in [9]$.

It may be seen in [Table 19.1](#) that the number of distinct connected, labelled graphs of orders 1, 2, 3 and 4 are 1, 1, 4 and 38, respectively. It is natural to attempt to extend this sequence² beyond $n = 4$. Unfortunately, no closed-form formula for the i -th term in this sequence is known. It is, however, possible to compute this sequence by means of a *recurrence relation* relating the number of connected labelled graphs of order n to those of orders less than n , as stated in the following theorem.

Theorem 19.3 *Let Θ_n denote the number of distinct connected, labelled graphs of order $n \in \mathbb{N}$. Then Θ_n satisfies the recurrence relation*

$$\Theta_n = 2^{\binom{n}{2}} - \frac{1}{n} \sum_{k=1}^{n-1} k \binom{n}{k} 2^{\binom{n-k}{2}} \Theta_k, \quad n = 2, 3, 4, \dots \quad (19.1)$$

together with the initial condition $\Theta_1 = 1$.

Proof Any of the $2^{\binom{n}{2}}$ labelled graphs of order n may be rooted in n different ways (since any vertex may be chosen as root), and hence there are $n 2^{\binom{n}{2}}$ rooted, labelled graphs of order n .

Suppose the root occurs in a component of order $k \in [n]$. There are $\binom{n}{k}$ different ways in which the vertices of this component may be labelled, and for each such labelling, there are Θ_k different ways in which to form the component in question. The component can be rooted in k different ways, and the remaining components of the graph may be formed in $2^{\binom{n-k}{2}}$ different ways according to [Theorem 19.2](#). By letting k vary over the set $[n]$, it therefore follows that

$$n 2^{\binom{n}{2}} = \sum_{k=1}^n k \binom{n}{k} 2^{\binom{n-k}{2}} \Theta_k,$$

from which we deduce that

$$n 2^{\binom{n}{2}} = n \Theta_n + \sum_{k=1}^{n-1} k \binom{n}{k} 2^{\binom{n-k}{2}} \Theta_k.$$

Solving for Θ_n produces the recurrence relation in (19.1) subject to the trivial initial condition $\Theta_1 = 1$. ■

One may utilise the recurrence relation in [Theorem 19.3](#) to build up the sequence $\Theta_2, \Theta_3, \Theta_4, \dots$ iteratively, observing the initial condition $\Theta_1 = 1$, as was done to populate [Table 19.2](#).

²Sequence [A001187](#) of Sloane [10].

Since $k \neq n$ in (19.1), the second term (involving the summation) on the right hand side of (19.1) represents the number of disconnected, labelled graphs of order n , which are subtracted from the total number of labelled graphs of order n to yield the number of connected, labelled graphs of order n . The first terms of the sequence³ giving the number of disconnected labelled graphs of order n are listed in [Table 19.2](#).

Notice in [Table 19.2](#) that the proportions of graphs of orders $3, 4, \dots, 8, 9, \dots$ that are connected are

$$50.00\%, 59.38\%, \dots, 93.71\%, 96.47\%, \dots \quad (19.2)$$

respectively, which is an increasing sequence. In fact, this sequence of proportions tends to 100% as $n \rightarrow \infty$, a result often stated as “almost all labelled graphs are connected.”

Theorem 19.4 (Almost all labelled graphs are connected)

$$\lim_{n \rightarrow \infty} \frac{\Theta_n}{2^{\binom{n}{2}}} = 1.$$

❖ The reader should now be able to attempt [Exercise 19.1](#) and [Project 19.1](#).

19.1.2 Labelled trees and rooted labelled trees

The Cambridge mathematician [Arthur Cayley](#) was arguably one of the fathers of graph theory. He became interested in graph enumeration when he set out counting the number of distinct labelled trees of order n . He proved the following theorem, now often called *Cayley's tree formula*. Many proofs of this beautiful theorem have been put forward. The proof we give here, dates back to 1918, is due to [Prüfer](#) [9] and proceeds by defining a bijection between the set of all labelled trees of order n and the set of $(n - 2)$ -tuples from the set $[n]$ (allowing repetition) — now known as Prüfer codes.

Theorem 19.5 *There are n^{n-2} distinct labelled trees of order $n \geq 2$.*

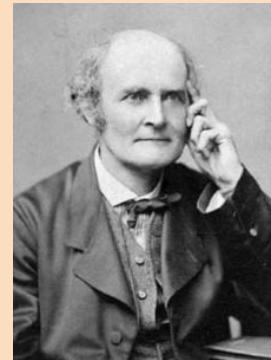
Proof Consider a tree T of order n whose vertices are labelled $1, 2, \dots, n$. A so-called unique **Prüfer code** may be constructed as follows by pruning the tree T . Remove the leaf of T with smallest label, a_1 say. Suppose the vertex labelled b_1 was the support vertex adjacent to the leaf that was removed. Suppose, furthermore, that among the remaining $n - 1$ vertices of T , the leaf labelled a_2 has the smallest label, and let b_2 be the label of the support vertex adjacent to this leaf. Remove the edge (a_2, b_2) , and repeat this process on the remaining $n - 2$ vertices of T , then on the remaining $n - 3$ vertices of T , and so on. The process terminates when only two vertices of the original tree remain. In the process a unique sequence

$$(b_1, b_2, \dots, b_{n-2}) \quad (19.3)$$

of support vertex labels is constructed iteratively as illustrated in [Figure 19.2\(a\)](#). This Prüfer code is an $(n - 2)$ -tuple from the set $[n]$ in which repetition is allowed.

³Sequence [A054592](#) of Sloane [10].

Arthur Cayley was born in Richmond, London on 16 August 1821. He is renowned for his uncanny memory, and was an avid (novel) reader and amateur mountaineer. At the age of 17 he was admitted to the University of Cambridge where he read mathematics. He had some difficulty obtaining a job after his graduation, and was admitted to the bar in 1849. Although very skilled as a lawyer, he always considered his 14 years of legal activities a means to make money so that he could pursue mathematics. In 1863, he was appointed first Sadlerian professor of mathematics at Cambridge. His research interests were in the theory of matrices, the theory of invariants, and analytic geometry of n -dimensional spaces. In algebra he was the first to define the concept of a group in the modern way — as a set together with a binary operation satisfying certain laws. His collected papers were published in thirteen volumes (1889–1898). He died in Cambridge on 26 January 1895 at the age of 73.



Biographic note 100: Arthur Cayley (1821–1895)

Conversely, given a Prüfer code as described above, a tree may be constructed uniquely as follows: Find the first number in the sequence

$$1, 2, \dots, n \tag{19.4}$$

that does not occur in the Prüfer code in (19.3). This number must be a_1 . Start the construction of a labelled tree by means of the edge (a_1, b_1) . Remove a_1 from the sequence in (19.4) and b_1 from the Prüfer code in (19.3). Find, amongst the remaining $n - 1$ numbers in the sequence (19.4), the first number that does not occur in the remaining part of the Prüfer code. This number must be a_2 . Continue the construction of the labelled tree by inserting the edge (a_2, b_2) . Remove a_2 from the remaining part of the sequence in (19.4) and b_2 from the remaining part of the Prüfer code in (19.3). Repeat this process until no labels remain in the Prüfer code. Finally insert an edge into the constructed labelled tree that joins the two vertices with remaining labels in (19.4). This iterative construction process is illustrated in Figure 19.2(b).

We have described a bijection between the set of all labelled trees of order n and the set of $(n - 2)$ -tuple Prüfer codes. It therefore follows that the number of distinct labelled trees equals the number of distinct $(n - 2)$ -tuple Prüfer codes, and all that remains is to count how many of these Prüfer codes there are. Since repetition is allowed, we may choose any one of the n elements in (19.4) to fill the i -th position of the Prüfer code, for all $i \in [n - 2]$. Since these choices are independent, there are

$$\underbrace{n \times n \times \cdots \times n}_{n-2 \text{ factors}}$$

distinct Prüfer codes, from which the desired result follows. ■

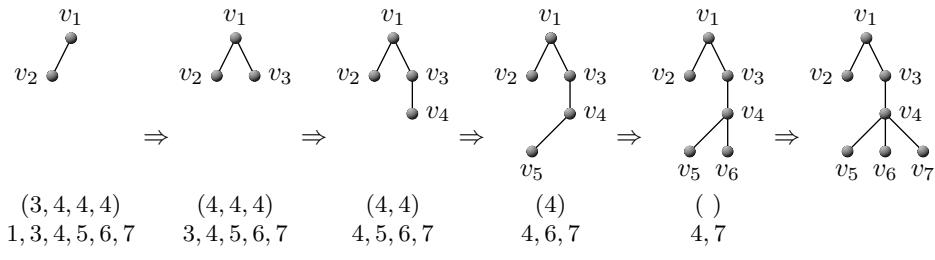
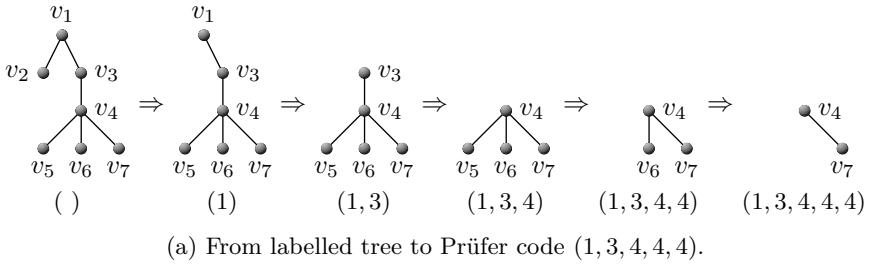


Figure 19.2: A bijection between the set of all labelled trees of order n and all Prüfer codes.

The sequence⁴ enumerating labelled trees of order n grows super-exponentially (faster than any exponential sequence) according to [Theorem 19.5](#). The first eight terms of the sequence are listed in [Table 19.3](#), whilst all labelled trees of orders 2, 3 and 4 are shown graphically in [Table 19.4](#).

From [Theorem 19.5](#) we immediately have the following corollary for the number of labelled, rooted trees.

Corollary 19.6 *There are n^{n-1} distinct labelled, rooted trees of order $n \in \mathbb{N}$.*

We leave the proof of this corollary as an exercise. The first nine terms of the sequence⁵ enumerating labelled, rooted trees of order n are listed in [Table 19.5](#), whilst all labelled, rooted trees of orders 1, 2 and 3 are shown in [Table 19.6](#).

❖ The reader should now be able to attempt Exercises [19.2–19.4](#).

19.1.3 Genealogical registers

In the discipline of genealogical ancestry, parents are used as points of reference to study their descendants. Usually only male descendants carry the name of the progenitor (original ancestor), and for this reason only males are traditionally recorded in family trees. A family tree may be modelled as a special kind of graph called a **genealogical register** or a **register** for short. In such a register, each male is represented by a vertex and two vertices are adjacent if and only if they are father and son. The resulting graph is a special kind of rooted tree in which the root represents the progenitor. The same convention as described for rooted trees in [Chapter 5](#) is adopted when drawing registers, namely placing the progenitor at

⁴Sequence [A000272](#) of Sloane [10].

⁵Sequence [A000169](#) of Sloane [10].

n	2	3	4	5	6	7	8	9
n^{n-2}	1	3	16	125	1 296	16 807	262 144	4 782 969

Table 19.3: The number of distinct labelled trees of orders $n \in \{2, \dots, 9\}$.

n	Labelled trees															
2																
3																
4																

Table 19.4: All labelled trees of order $n \in \{2, 3, 4\}$.

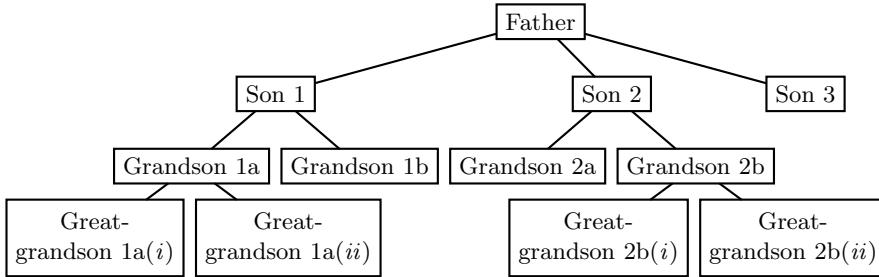
n	1	2	3	4	5	6	7	8	9
n^{n-1}	1	2	9	64	625	7 776	117 649	2 097 152	43 046 721

Table 19.5: The number of distinct labelled, rooted trees of order $n \in [9]$.

n	Labelled, rooted trees								
1									
2									
3									

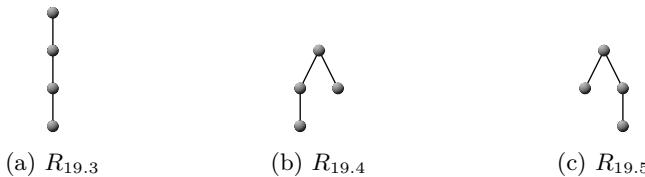
Table 19.6: All labelled, rooted trees of order $n \in [3]$. Root vertices are circled.

the top, with his sons on level 1, their sons on level 2, and so forth, as shown in [Figure 19.3](#). Thus, if a father is on level i of the register, then all his sons are on level $i + 1$ and adjacent to him.



[Figure 19.3](#): A genealogical register of order 12.

In genealogical ancestry, the chronological order (or relative ages) of siblings is, however, typically important; this may play some role in terms of inheritance. Therefore, the chronological order of siblings needs to be recorded in a register. In order to accomplish this, the convention is followed that siblings are arranged from left to right in chronological order of their birth dates on a level. For example, in the register in [Figure 19.3](#), “Son 1” is the eldest, followed by “Son 2,” and finally “Son 3” is the youngest sibling of the “Father.” Similarly, “Grandson 1a” is the elder of the two sons of “Son 1,” but “Grandson 1a” is not necessarily older than “Grandson 2a” (who is not his sibling). This additional, secondary or horizontal ordering distinguishes registers from ordinary root trees in which there is only a primary or vertical ordering.



[Figure 19.4](#): Three distinct genealogical registers of order 4.

For example, when viewed as ordinary graphs, the three graphs $R_{19.3}$, $R_{19.4}$ and $R_{19.5}$ in [Figure 19.4](#) are isomorphic. When viewed as rooted trees, however, the trees $R_{19.4}$ and $R_{19.5}$ are isomorphic, but they are not isomorphic to $R_{19.3}$. Yet, when viewed as genealogical registers, no two of the three registers are isomorphic.

Let \mathcal{C}_n denote the number of distinct genealogical registers of order n . Our objective in this section is to determine \mathcal{C}_n as a function of n , *i.e.* to enumerate the genealogical registers of order n . In [Figure 19.5](#) it may be seen that $\mathcal{C}_1 = 1$, $\mathcal{C}_2 = 1$, $\mathcal{C}_3 = 2$, $\mathcal{C}_4 = 5$ and $\mathcal{C}_5 = 14$.

Our approach towards evaluating \mathcal{C}_n as a function of n is to derive a recurrence relation to which \mathcal{C}_n is the solution, *i.e.* to find a relationship between \mathcal{C}_n and $\mathcal{C}_1, \dots, \mathcal{C}_{n-1}$ rather than to attempt evaluating \mathcal{C}_n directly.

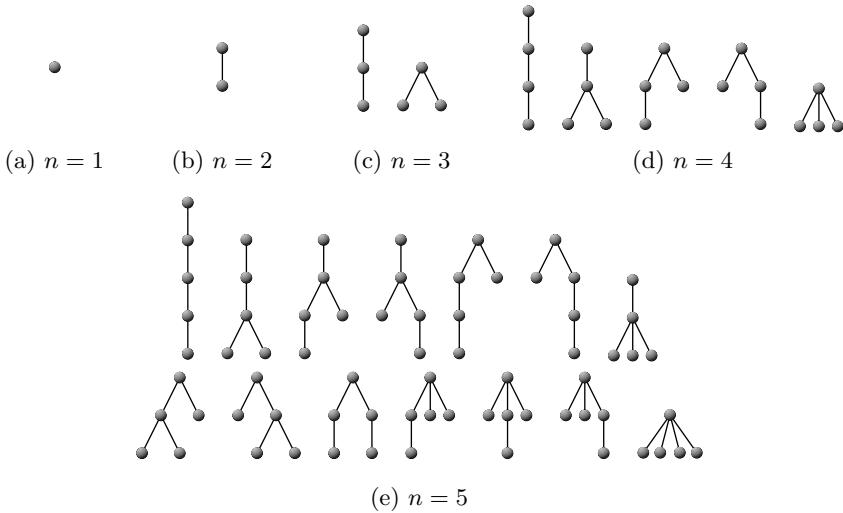


Figure 19.5: Genealogical registers of order $n \in [5]$.

Observe that if we remove the edge in a register of order $n \geq 2$ joining the progenitor and his eldest son, then we disconnect the register into exactly two components (which are again genealogical registers) — one of order $i \in [n - 1]$, which we call the **left subregister**, and the other of order $n - i$, which we call the **right subregister**. Furthermore, each different combination of left and right subregisters of orders respectively i and $n - i$ results in a different register of order n . Hence the number of distinct genealogical registers of order n equals the number of distinct ways in which we can form combinations of left and right subregisters of orders respectively i and $n - i$, for all $i \in [n - 1]$. For any fixed value of i the choices of left and right subregister structures are independent and hence there are $\mathcal{C}_i \times \mathcal{C}_{n-i}$ different combinations of left and right subregisters of orders i and $n - i$ respectively. By letting i vary over the set $[n - 1]$ it, therefore, follows that

$$\mathcal{C}_n = \sum_{i=1}^{n-1} \mathcal{C}_i \mathcal{C}_{n-i}, \quad n = 2, 3, 4, \dots \quad (19.5)$$

subject to the initial condition $\mathcal{C}_1 = 1$. We require the notion of a generating function before we can proceed to solve for \mathcal{C}_n in this nonlinear recurrence relation.

A **generating function** is a power series of the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots \quad (19.6)$$

in some dummy variable x . The coefficient a_k of x^k in this powerful enumeration tool represents the number of distinct combinatorial objects of interest satisfying some specified condition involving the integer k . The variable x serves no purpose other than being a symbol to which exponents may be attached. For example, it follows by the binomial theorem that

$$(1 + x)^n = \sum_{k=0}^n \binom{n}{k} x^k, \quad (19.7)$$

which is a power series of the form (19.6) in which

$$a_k = \begin{cases} \binom{n}{k} & \text{if } 0 \leq k \leq n \\ 0 & \text{if } k > n \end{cases}$$

counts the number of distinct combinations from n objects, taken k at a time. The advantage of considering the generating function $(1+x)^n$ in (19.7) rather than its coefficients during enumeration procedures, is that algebraic manipulation of the generating function is typically much easier (and often significantly less tedious) than manipulating its coefficients directly.

Returning to the recurrence relation (19.5), we examine the generating function

$$C(x) = \sum_{n=1}^{\infty} \mathcal{C}_n x^n. \quad (19.8)$$

Combining (19.5) and (19.8) we find that

$$C(x) - x = \sum_{n=2}^{\infty} [\mathcal{C}_1 \mathcal{C}_{n-1} + \mathcal{C}_2 \mathcal{C}_{n-2} + \cdots + \mathcal{C}_{n-1} \mathcal{C}_1] x^n.$$

Note that $\mathcal{C}_1 \mathcal{C}_{n-1} + \mathcal{C}_2 \mathcal{C}_{n-2} + \cdots + \mathcal{C}_{n-1} \mathcal{C}_1$ is also the coefficient of x^n in the product $C(x) C(x)$. Hence we may also write $C(x) - x = [C(x)]^2$, from which it follows that

$$C(x) = \frac{1 - \sqrt{1 - 4x}}{2}.$$

(The other solution is disregarded, since $C(x)$ does not contain a constant term.) The so-called **generalised binomial theorem** states that

$$(1+b)^a = 1 + \sum_{n=1}^{\infty} \frac{a(a-1)(a-2)\cdots(a-n+1)}{n!} b^n \quad (19.9)$$

for all $a, b \in \mathbb{R}$ satisfying $|b| < 1$; here a need not be integral, as was the case in (19.7). Hence

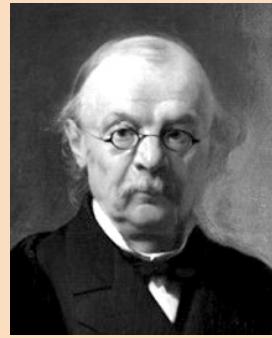
$$\begin{aligned} C(x) &= \frac{1 - \sqrt{1 - 4x}}{2} \\ &= \frac{1}{2} - \frac{1}{2} \left[1 + \sum_{n=1}^{\infty} \frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2)\cdots(\frac{1}{2}-n+1)}{n!} (-4x)^n \right] \\ &= \sum_{n=1}^{\infty} \frac{1 \cdot 2(1) \cdot 3 \cdot 2(2) \cdots (2n-5) \cdot 2(n-2) \cdot (2n-3) \cdot 2(n-1)}{n!(n-1)!} x^n \\ &= \sum_{n=1}^{\infty} \frac{(2n-2)!}{n!(n-1)!} x^n. \end{aligned} \quad (19.10)$$

Comparing (19.8) and (19.10) yields the so-called **Catalan numbers**

$$\mathcal{C}_n = \frac{(2n-2)!}{n!(n-1)!} = \frac{1}{n} \binom{2n-2}{n-1}, \quad n = 1, 2, 3, \dots \quad (19.11)$$

named after the Belgian-born French mathematician [Eugène Charles Catalan](#). The first thirteen terms of the Catalan sequence⁶ are shown in [Table 19.7](#). The n -th Catalan number therefore enumerates (amongst many other applications⁷) the genealogical registers of order n .

Eugène Charles Catalan was born in Bruges (now in Belgium, but then part of the First French Empire) on 30 May 1814. He entered the renowned École Polytechnique in 1833 to study mathematics, but was expelled the following year as a result of his left-wing political activities. Allowed to resume his studies in 1835, he graduated in sixteenth place and obtained a doctorate in mathematics in 1841. After teaching mathematics at various institutions across France he lost his job for refusing to take the required oath of allegiance to Napoleon Bonaparte after the latter had named himself emperor. He published extensively on various topics in descriptive geometry, continued fractions and number theory. He is most famous for the Catalan numbers featured in this section and for a conjecture that no two consecutive integers, other than 8 and 9, can be consecutive powers, *i.e.* that the equation $x^m - y^n = 1$ has only one solution — which was proved correct only in 2003! Catalan was elected to the Royal Belgium Academy of Science in 1865 and died in Liège, Belgium on 14 February 1894 at the age of 79.



Biographic note 101: Eugène Catalan (1814–1894)

n	1	2	3	4	5	6	7	8	9	10	11	12	13
C_n	1	1	2	5	14	42	132	429	1430	4862	16796	58786	208012

Table 19.7: The famous Catalan numbers C_1, \dots, C_{13} , enumerating the genealogical registers of order $n \in [13]$.

- ❖ The reader should now be able to attempt [Exercise 19.5](#) as well as Projects [19.2–19.3](#).

19.2 Counting unlabelled trees

In the previous section we confined our attention to counting various labelled graph types and structures, *i.e.* graphs in which all vertices are considered distinguishable. In this section we consider the more complex problem of counting various unlabelled graph types and structures, *i.e.* graphs in which vertices are considered indistinguishable.

⁶Sequence [A000108](#) of Sloane [10].

⁷See, for example, [Stanley](#) [11] for a large number of applications of the Catalan numbers.

19.2.1 Rooted trees

In order to count the number of distinct, rooted unlabelled trees we require the notion of a partition of a positive integer. Recall that a partition of an n -set into k nonempty sets X_1, \dots, X_k of cardinalities n_1, \dots, n_k respectively, satisfies the relationship

$$X = X_1 \cup X_2 \cup \dots \cup X_k.$$

When considering the cardinalities of the partition sets and the large set above, we arrive at the associated equation

$$n = n_1 + n_2 + \dots + n_k, \quad (19.12)$$

which is known as a **partition of the integer n** into k nonzero **parts of sizes n_1, n_2, \dots, n_k** , respectively. The order in which the parts of a partition are written is, of course, unimportant, and hence there are, for example, seven distinct partitions

$$5 = \begin{cases} 1 + 1 + 1 + 1 + 1 \\ 1 + 1 + 1 + 2 \\ 1 + 1 + 3 \\ 1 + 2 + 2 \\ 1 + 4 \\ 2 + 3 \quad \text{and} \\ 5 \end{cases}$$

of the integer 5. The only possible sizes of parts in a partition of the integer n are $1, 2, \dots, n$. If there are a_i parts of size i in such a partition, then we denote the partition by the standard notation $[1^{a_1} 2^{a_2} \dots n^{a_n}]$ (see, for example, [1]). If $a_i = 0$ for some $i \in [n]$, then it is standard practice to omit the part i^{a_i} in this notation. Therefore the above seven partitions of the integer 5 may be abbreviated as $[1^5]$, $[1^3 2^1]$, $[1^2 3^1]$, $[1^1 2^2]$, $[1^1 4^1]$, $[2^1 3^1]$ and $[5^1]$, respectively.

Let us now return to the problem of counting the number of distinct rooted unlabelled trees. Observe that a rooted unlabelled tree is a tree in which all vertices, except the root, are considered indistinguishable. Let $r_{n,m}$ denote the number of distinct rooted trees of order n in which the degree of the root is exactly m . When omitting the subscript m , *i.e.* when writing r_n , we refer to the number of distinct rooted trees of order n irrespective of the degrees of their roots, in other words, *all* rooted trees of order n . Clearly,

$$r_n = \sum_{m=1}^{n-1} r_{n,m}.$$

Any rooted tree T of order n with a root v_r of degree m may be decomposed into m rooted subtrees, each attached to v_r by means of a single edge joining the root of the particular subtree and the root v_r of the larger tree, by deleting the root of the tree (see, for example, Figure 19.6). Since $n - 1$ of the vertices of T (all the vertices except the root) are distributed amongst the m subtrees, we may associate the partition

$$s_1 + 2s_2 + 3s_3 + \dots + (n - 1)s_{n-1} = n - 1 \quad (19.13)$$

of the integer $n - 1$ satisfying

$$s_1 + s_2 + s_3 + \cdots + s_{n-1} = m$$

with the tree T , where s_i denotes the number of subtrees of order i in the decomposition.

For example, for the tree $T_{19.6}$ in Figure 19.6, $n = 13$, $m = 4$, $s_1 = 1$, $s_2 = 1$, $s_3 = 1$, $s_6 = 1$ and $s_4 = s_5 = s_7 = s_8 = \cdots = s_{12} = 0$. Furthermore, $\sum_{i=1}^{12} s_i = 4$ and $\sum_{i=1}^{12} i s_i = 1 + 3 + 2 + 6 = 12$.

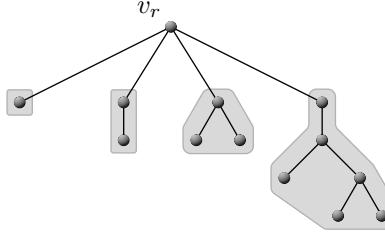


Figure 19.6: Decomposition of a rooted tree $T_{19.6}$ of order $n = 13$ with root v_r of degree $m = 4$ into four rooted subtrees.

We may select s_i trees from the r_i distinct unlabelled rooted trees of order i to form subtrees in the decomposition of our rooted tree T . Since the same tree may be selected more than once as subtree, we are interested in the number of different ways of selecting s_i objects from r_i (distinct) objects if unlimited repetition is allowed in the selection. It is left as an exercise to show that this number of selections is

$$\binom{r_i + s_i - 1}{s_i}$$

(see Exercise 19.7). Because every such selection may be made independently, the number of distinct rooted unlabelled trees corresponding to the partition $p_{n-1} = [1^{s_1} 2^{s_2} \cdots (n-1)^{s_{n-1}}]$ of $n - 1$ in (19.13) is

$$r_n[s_1, \dots, s_{n-1}] = \binom{r_1 + s_1 - 1}{s_1} \binom{r_2 + s_2 - 1}{s_2} \cdots \binom{r_{n-1} + s_{n-1} - 1}{s_{n-1}}. \quad (19.14)$$

It follows, by summing over all possible partitions of the integer $n - 1$, that

$$r_n = \sum_{p_{n-1}} \prod_{i=1}^{n-1} \binom{r_i + s_i - 1}{s_i}. \quad (19.15)$$

The recurrence relation (19.15) yields r_n , the numbers of distinct rooted unlabelled trees of order n , in terms of r_1, \dots, r_{n-1} , the numbers of distinct rooted unlabelled trees of smaller orders. One may utilise this recurrence relation to build up the sequence r_1, r_2, r_3, \dots iteratively, observing the trivial initial condition $r_1 = 1$ (the single rooted unlabelled tree of order $n = 1$ is shown in Figure 19.7(a)). Since there is only the single partition $[1^1]$ of the integer $n - 1 = 1$, it follows from (19.15) with $n = 2$ that

$$r_2 = \binom{r_1 + 1 - 1}{1} = \binom{1}{1} = 1. \quad (19.16)$$

This single rooted unlabelled tree of order $n = 2$ is shown in [Figure 19.7\(b\)](#). Since there are the two partitions $[1^2]$ and $[2^1]$ of the integer $n - 1 = 2$, it follows from [\(19.15\)](#) with $n = 3$ that

$$r_3 = \underbrace{\binom{r_1 + 2 - 1}{2}}_{\text{partition } [1^2]} + \underbrace{\binom{r_2 + 1 - 1}{1}}_{\text{partition } [2^1]} = \binom{2}{2} + \binom{1}{1} = 2.$$

These two rooted unlabelled trees of order $n = 3$ are shown in [Figure 19.7\(c\)](#). Similarly, since there are the three partitions $[1^3]$, $[1^12^1]$ and $[3^1]$ of the integer $n - 1 = 3$, it follows from [\(19.15\)](#) with $n = 4$ that

$$r_4 = \underbrace{\binom{r_1 + 3 - 1}{3}}_{\text{partition } [1^3]} + \underbrace{\binom{r_1 + 1 - 1}{1} \binom{r_2 + 1 - 1}{1}}_{\text{partition } [1^12^1]} + \underbrace{\binom{r_3 + 1 - 1}{1}}_{\text{partition } [3^1]} = 4.$$

These four rooted unlabelled trees of order $n = 4$ are shown in [Figure 19.7\(d\)](#). For $n - 1 = 4$ there are the five partitions $[1^4]$, $[1^22^1]$, $[2^2]$, $[1^13^1]$ and $[4^1]$. Hence it follows from [\(19.15\)](#) with $n = 5$ that

$$\begin{aligned} r_5 &= \underbrace{\binom{r_1 + 4 - 1}{4}}_{\text{partition } [1^4]} + \underbrace{\binom{r_1 + 2 - 1}{2} \binom{r_2 + 1 - 1}{1}}_{\text{partition } [1^22^1]} + \underbrace{\binom{r_2 + 2 - 1}{2}}_{\text{partition } [2^2]} \\ &\quad + \underbrace{\binom{r_1 + 1 - 1}{1} r_3 + 1 - 11}_{\text{partition } [1^13^1]} + \underbrace{\binom{r_4 + 1 - 1}{1}}_{\text{partition } [4^1]} \\ &= 1 + 1 + 1 + 2 + 4 = 9. \end{aligned} \tag{19.17}$$

These nine rooted trees of order $n = 5$ are shown in [Figure 19.7\(e\)](#). The process above may be continued to yield the sequence⁸ shown in [Table 19.8](#).

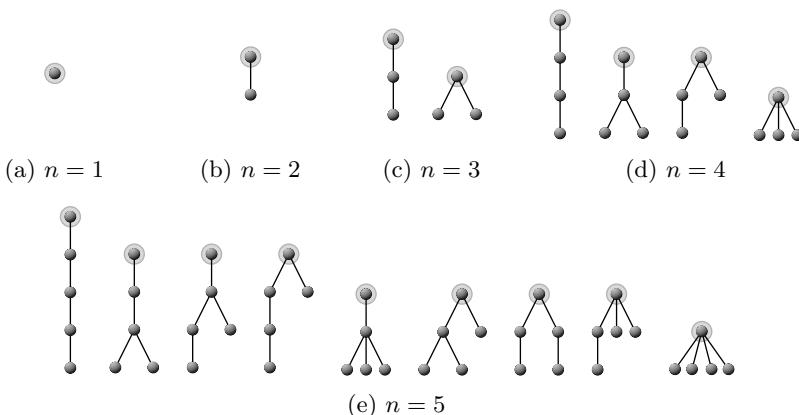


Figure 19.7: Rooted, unlabelled trees of order $n \in [5]$.

⁸Sequence [A000081](#) of Sloane [10].

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
r_n	1	1	2	4	9	20	48	115	286	719	1842	4766	12486	32973

Table 19.8: The number of (unlabelled) rooted trees of order $n \in [14]$.

The process described above is, however, clearly very tedious. If we were to seek the number of distinct rooted unlabelled trees of order $n = 25$, for example — there are 2 067 174 645 — then we would have to find all the partitions of the integer 24 (there are 1 575 such partitions!), and so evaluating the resulting sum in (19.15) would certainly be a nontrivial exercise.

Fortunately, there is a more efficient way of evaluating the sequence r_1, r_2, r_3, \dots , namely by considering its generating function

$$r(x) = \sum_{n=1}^{\infty} r_n x^n = x \sum_{n=1}^{\infty} r_n x^{n-1}. \quad (19.18)$$

Upon substitution of (19.15) into (19.18), it follows that

$$r(x) = x \sum_{n=1}^{\infty} \sum_{p_{n-1}} \prod_{i=1}^{n-1} \binom{r_i + s_i - 1}{s_i} x^{n-1},$$

which may be rewritten as

$$\begin{aligned} r(x) = x \sum_{n=1}^{\infty} & \sum_{p_{n-1}} \left[\binom{r_1 + s_1 - 1}{s_1} x^{s_1} \binom{r_2 + s_2 - 1}{s_2} x^{2s_2} \dots \right. \\ & \left. \dots \binom{r_{n-1} + s_{n-1} - 1}{s_{n-1}} x^{(n-1)s_{n-1}} \right]. \end{aligned} \quad (19.19)$$

By noting that every sequence of positive integers forms a partition of some integer, (19.19) may, in turn, be rewritten as

$$\begin{aligned} r(x) = x & \left[\sum_{s_1=0}^{\infty} \binom{r_1 + s_1 - 1}{s_1} x^{s_1} \right] \left[\sum_{s_2=0}^{\infty} \binom{r_2 + s_2 - 1}{s_2} x^{s_2} \right] \dots \\ & \dots \left[\sum_{s_{n-1}=0}^{\infty} \binom{r_{n-1} + s_{n-1} - 1}{s_{n-1}} x^{s_{n-1}} \right] \dots . \end{aligned} \quad (19.20)$$

It is left as an exercise to verify the identity

$$(1 - x^b)^{-a} = \sum_{i=0}^{\infty} \binom{a + i - 1}{i} x^{bi} \quad (19.21)$$

for any $a \in \mathbb{N}$ (see Exercise 19.9). Substitution of (19.21) into (19.20) yields the desired generating function

$$\begin{aligned} r(x) &= x(1-x)^{-r_1}(1-x^2)^{-r_2}(1-x^3)^{-r_3}\dots \\ &= x \prod_{i=1}^{\infty} (1-x^i)^{-r_i} \end{aligned} \quad (19.22)$$

for the number of rooted unlabelled trees of orders $n = 1, 2, 3, \dots$. This generating function may be expanded by iteratively building up the sequence

$$r^{(1)}(x), r^{(2)}(x), r^{(3)}(x), \dots$$

of power series, as shown in [Figure 19.8](#), where

$$r^{(j)}(x) = x \prod_{i=1}^j (1 - x^i)^{-r_i}, \quad j \in \mathbb{N}.$$

$$\begin{aligned} r_1 &= 1 \\ r^{(1)}(x) &= x(1-x)^{-1} \\ &= [x + 1x^2] + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + \dots \\ r^{(2)}(x) &= x(1-x)^{-1}(1-x^2)^{-1} \\ &= [x + x^2 + 2x^3] + 2x^4 + 3x^5 + 3x^6 + 4x^7 + 4x^8 + 5x^9 + 5x^{10} + \dots \\ r^{(3)}(x) &= x(1-x)^{-1}(1-x^2)^{-1}(1-x^3)^{-2} \\ &= [x + x^2 + 2x^3 + 4x^4] + 5x^5 + 7x^6 + 11x^7 + 13x^8 + 17x^9 + 23x^{10} + \dots \\ r^{(4)}(x) &= x(1-x)^{-1}(1-x^2)^{-1}(1-x^3)^{-2}(1-x^4)^{-4} \\ &= [x + x^2 + 2x^3 + 4x^4 + 9x^5] + 11x^6 + 19x^7 + 29x^8 + 47x^9 + 61x^{10} + \dots \\ r^{(5)}(x) &= x(1-x)^{-1}(1-x^2)^{-1}(1-x^3)^{-2}(1-x^4)^{-4}(1-x^5)^{-9} \\ &= [x + x^2 + 2x^3 + 4x^4 + 9x^5 + 20x^6] + 28x^7 + 47x^8 + 83x^9 + 142x^{10} + \dots \\ r^{(6)}(x) &= x(1-x)^{-1}(1-x^2)^{-1}(1-x^3)^{-2}(1-x^4)^{-4}(1-x^5)^{-9}(1-x^6)^{-20} \\ &= [x + x^2 + 2x^3 + 4x^4 + 9x^5 + 20x^6 + 48x^7] + 67x^8 + 123x^9 + 222x^{10} + \dots \end{aligned}$$

Figure 19.8: The process of building up the generating function [\(19.22\)](#).

In view of [\(19.21\)](#), the coefficients of x^k in the sequence of power series $(r^{(j)}(x))_{j=1,2,3,\dots}$ is nondecreasing for any fixed $k \in \mathbb{N}$, and clearly the limit of this sequence of integers is the coefficient of x^k in the generating function $r(x)$. Moreover, since

$$(1 - x^j)^{-r_j} = 1 + r_j x^j + \mathcal{O}(x^{2j}), \quad j \in \mathbb{N},$$

it follows that the first $j+1$ terms of $r^{(j)}(x)$ coincide exactly with the first $j+1$ terms of the generating function $r(x)$ (see the highlighted parts of the power series in [Figure 19.8](#)). Therefore, the coefficient of x^{j+1} in $r^{(j)}(x)$ may be taken as the value of r_{j+1} in the computation of the power series $r^{(j+1)}(x)$, as indicated by the arrows in [Figure 19.8](#).

It follows that if the first k terms of the generating function $r(x)$ are required exactly, then these terms may be read off from the power series $r^{(k-1)}(x)$. For example, by iteratively computing the sequence of power series $r^{(1)}(x), \dots, r^{(9)}(x)$, the first ten terms of the generating function (19.22) for the number of rooted unlabelled trees are found to be

$$r(x) = x + x^2 + 2x^3 + 4x^4 + 9x^5 + 20x^6 + 48x^7 + 115x^8 + 286x^9 + 719x^{10} + \dots, \quad (19.23)$$

from which we may populate a table such as Table 19.8.

At first glance the method described above for computing the sequence r_1, r_2, r_3, \dots enumerating the number of rooted unlabelled trees may not seem significantly less tedious than the method previously described of utilising the recurrence relation (19.15) directly, as was demonstrated in (19.16)–(19.17). The method involving the use of a generating function, however, involves mere multiplications of power series — something a computer is easily programmed to do. Programming a computer to generate all partitions of an integer and performing the computations illustrated in (19.16)–(19.17) for all of these partitions, on the other hand, is a little trickier (but possible, of course).

❖ The reader should now be able to attempt Exercises 19.6–19.10.

19.2.2 Trees that are not rooted

In this section we consider the problem of enumerating unlabelled trees of order n that are not rooted.

At any vertex v of degree d in a tree T , there are d maximal subtrees sharing only the vertex v . The **weight** of v is defined as the largest of the sizes of these subtrees and any vertex of minimum weight in T is called a **centroid** of T . It is left as an exercise to show that every tree has either exactly one centroid or exactly two centroids (see Project 19.4) — very reminiscent of the situation concerning the centre of a tree (see Theorem 6.4). A tree with exactly one centroid is called a **centroidal tree**, whilst a tree with exactly two centroids is called a **bicentroidal tree**. The tree in Figure 19.9(a) is an example of a centroidal tree, whilst the tree in Figure 19.9(b) is an example of a bicentroidal tree.

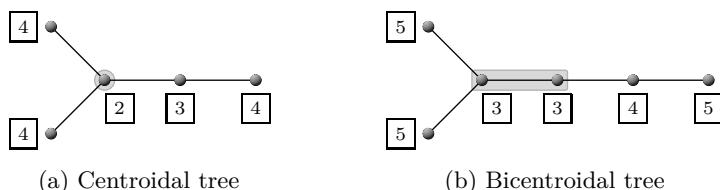


Figure 19.9: (a) A centroidal tree. (b) A bicentroidal tree. The boxed numbers shown next to the vertices are their weights. Centroids are denoted by means of circled vertices.

Our approach will be to enumerate the (unlabelled) centroidal trees of order n and the (unlabelled) bicentroidal trees of order n separately. Suppose

$$t(x) = \sum_{n=1}^{\infty} t_n x^n \quad (19.24)$$

is the generating function enumerating bicentroidal trees (*i.e.* the coefficient t_n of x^n in (19.24) is the number of distinct bicentroidal trees of order n). It may be shown that the order of a bicentroidal tree is necessarily even (see Project 19.4) and hence a bicentroidal tree T of order $n = 2\ell$ may be viewed as two rooted trees, T_L and T_R , each of order ℓ , and joined at their roots by means of an edge. There are clearly r_ℓ such trees T in which $T_L \cong T_R$, and $r_\ell(r_\ell - 1)/2$ such trees T in which $T_L \not\cong T_R$, where r_ℓ is the coefficient of x^ℓ in the generating function (19.18). Therefore, $t_n = r_\ell + r_\ell(r_\ell - 1)/2 = r_\ell(r_\ell + 1)/2$ and hence

$$\begin{aligned} t(x) &= \frac{1}{2} \sum_{\ell=1}^{\infty} r_\ell(r_\ell + 1)x^{2\ell} = \frac{1}{2} \sum_{\ell=1}^{\infty} r_\ell x^{2\ell} + \frac{1}{2} \sum_{\ell=1}^{\infty} (r_\ell x^\ell)^2 \\ &= \frac{1}{2} r(x^2) + \frac{1}{2} \sum_{\ell=1}^{\infty} (r_\ell x^\ell)^2 \end{aligned} \quad (19.25)$$

$$\begin{aligned} &= \frac{1}{2}(x^2 + x^4 + 2x^6 + 4x^8 + 9x^{10} + 20x^{12} + 48x^{14} + \dots) \\ &\quad + \frac{1}{2}(x^2 + x^4 + 4x^6 + 16x^8 + 81x^{10} + 400x^{12} + 2304x^{14} + \dots) \\ &= x^2 + x^4 + 3x^6 + 10x^8 + 45x^{10} + 210x^{12} + 1176x^{14} + \dots \end{aligned} \quad (19.26)$$

The first ten terms of the sequence⁹ enumerating bicentroidal trees brought forth by this generating function appear in Table 19.9.

n	2	4	6	8	10	12	14	16	18	20
Bicentroidal trees	1	1	3	10	45	210	1176	6670	41041	258840

Table 19.9: The number of (unlabelled) bicentroidal trees of orders n not exceeding 20.

Now suppose

$$\tau(x) = \sum_{n=1}^{\infty} \tau_n x^n \quad (19.27)$$

is the generating function enumerating centroidal trees (*i.e.* the coefficient τ_n of x^n in (19.27) is the number of distinct centroidal trees of order n). It may be shown, by means of a complicated case-wise decomposition analysis of the possible structures of centroidal trees that

$$\tau(x) = r(x) - \frac{1}{2} r^2(x) - \frac{1}{2} \sum_{\ell=1}^{\infty} (r_\ell x^\ell)^2 \quad (19.28)$$

$$\begin{aligned} &= x + x^2 + 2x^3 + 4x^4 + 9x^5 + 20x^6 + 48x^7 + 115x^8 + 286x^9 + \dots \\ &\quad - \frac{1}{2}(x^2 + 2x^3 + 5x^4 + 12x^5 + 30x^6 + 74x^7 + 188x^8 + 478x^9 + \dots) \\ &\quad - \frac{1}{2}(x^2 + x^4 + 4x^6 + 16x^8 + \dots) \\ &= x + x^3 + x^4 + 3x^5 + 3x^6 + 11x^7 + 13x^8 + 47x^9 + \dots \end{aligned} \quad (19.29)$$

⁹Sequence A102911 of Sloane [10].

The details of this analysis are not particularly illuminating, and are omitted in view of their involved nature. The coefficients of the first fifteen terms of the sequence¹⁰ enumerating centroidal trees brought forth by this generating function (19.29) appear in Table 19.10.

n	1	2	3	4	5	6	7	8	9	10	11	12
Centroidal trees	1	0	1	1	3	3	11	13	47	61	235	341

Table 19.10: The number of (unlabelled) centroidal trees of order $n \in [12]$.

By adding together (19.25) and (19.28), we obtain the generating function

$$T(x) = \overbrace{r(x)}^{t(x)} - \overbrace{\frac{1}{2}[r^2(x) - r(x^2)]}^{\tau(x)} \quad (19.30)$$

$$\begin{aligned} &= x + x^2 + 2x^3 + 4x^4 + 9x^5 + 20x^6 + 48x^7 + 115x^8 + 286x^9 + \dots \\ &\quad - \frac{1}{2}(x^2 + 2x^3 + 5x^4 + 12x^5 + 30x^6 + 74x^7 + 188x^8 + 478x^9 + \dots) \\ &\quad + \frac{1}{2}(x^2 + x^4 + 2x^6 + 4x^8 + \dots) \\ &= x + x^2 + x^3 + 2x^4 + 3x^5 + 6x^6 + 11x^7 + 23x^8 + 47x^9 + \dots \end{aligned} \quad (19.31)$$

for unlabelled trees. The generating function for unlabelled trees in terms of that for rooted unlabelled trees (19.30) dates back to 1948, and is due to [Otter](#) [7]. It has subsequently become known as **Otter's tree formula**. The expanded form (19.31) of this formula gives rise to a sequence¹¹ which may be used to populate a table such as Table 19.11. All unlabelled trees of order at most $n = 6$ are shown in Table 19.12.

n	1	2	3	4	5	6	7	8	9	10	11	12
t_n	1	1	1	2	3	6	11	23	47	106	235	551

Table 19.11: The number of (unlabelled) trees of order $n \in [12]$.

❖ The reader should now be able to attempt [Project 19.4](#).

19.3 Pólya's enumeration theorem

In order to enumerate more general (unlabelled) graph structures than trees (such as the number of graphs, multigraphs or digraphs of a specified order), we require a more powerful enumeration tool, known as [Pólya's Enumeration Theorem](#). In this section we therefore digress temporarily from the topic of graph enumeration in order to present the reader with the mathematical background necessary to understand this theorem. [Pólya's Enumeration Theorem](#) is very general indeed — it may be used to enumerate far more general combinatorial structures than graphs. We shall, however, not consider the theorem in its full generality, preferring to

¹⁰Sequence [A027416](#) of Sloane [10].

¹¹Sequence [A000055](#) of Sloane [10].

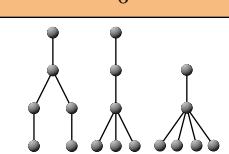
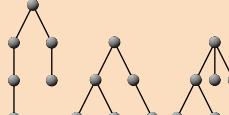
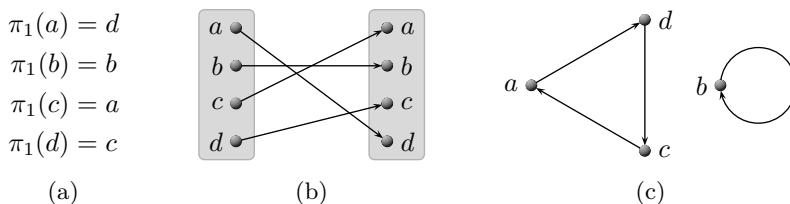
n	1	2	3	4	5	6
Centroidal trees	•					
Bicentroidal trees						
Total						

Table 19.12: Unlabelled trees of order $n \in [6]$.

present the theorem in the confined context required to be able to apply it to graph enumeration problems specifically. We apologise if, in the process, we do not do justice to the power and beauty of the theorem.

19.3.1 Permutations

A **permutation** is an injection $\pi : A \mapsto A$ from a finite set A to itself, and the cardinality $|A|$ of the set A is called the **degree** of the permutation. For example, the permutation π_1 of degree 4 on the set $A_1 = \{a, b, c, d\}$ defined in [Figure 19.10\(a\)](#) and illustrated graphically in [Figure 19.10\(b\)](#) is often denoted by $\pi_1 = \begin{pmatrix} a & b & c & d \\ d & b & a & c \end{pmatrix}$, which is read vertically in pairs as “ a maps to d ,” “ b maps to b ,” “ c maps to a ” and “ d maps to c ” under the permutation π_1 .

**Figure 19.10:** A permutation π_1 of degree 4 on the set $A_1 = \{a, b, c, d\}$.

Any permutation may be represented as a directed pseudograph, in which the indegree and outdegree of each vertex is 1. In such a directed pseudograph, every component is either a cycle or a loop, representing a so-called **permutation cycle** or **cycle** for short, and each arc describes a single element mapping under the permutation. For example, the directed pseudograph representation of the permutation π_1 is shown in [Figure 19.10\(c\)](#). The directed cycle on the left of the figure may be interpreted as the (permutation) cycle “ a maps to d ,” “ d maps to c ” and “ c maps to a ” under π_1 . The loop on the right may be interpreted as the (permutation) cycle “ b maps to b ” under π_1 . This directed pseudograph representation gives rise to the compact, standard notation $\pi_1 = (adc)(b)$, known as the **cyclic representation** of π_1 .

The **length** of a permutation cycle is the number of arcs in the corresponding component of the directed pseudograph representation. The length of the cycle (adc) in the permutation π_1 is therefore 3, whilst that of the cycle (b) is 1. Often one is only interested in information about the *numbers* of cycles of various lengths in a permutation and not in the cycles themselves. A permutation π of degree k is said to be of **type** $(\alpha_1, \alpha_2, \dots, \alpha_k)$ if π has α_i cycles of length $i \in [k]$. The permutation π_1 of degree 4 is therefore of type $(1, 0, 1, 0)$, whilst the permutation $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 1 & 2 & 5 & 6 \end{pmatrix} = (4231)(5)(6)$ of degree 6 is of type $(2, 0, 0, 1, 0, 0)$. If π is a permutation of type $(\alpha_1, \alpha_2, \dots, \alpha_k)$, then clearly the vector $(\alpha_1, \alpha_2, \dots, \alpha_k)$ induces a partition of the permutation degree k in the sense that

$$1\alpha_1 + 2\alpha_2 + \cdots + k\alpha_k = k.$$

A standard method for indicating the type of a permutation of degree k is to introduce k dummy variables y_1, \dots, y_k and then to indicate the permutation type as exponents in a product of these variables; that is, by writing

$$y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_k^{\alpha_k}. \quad (19.32)$$

The variables y_1, \dots, y_k have no significance, other than serving the purpose of symbols to which exponents may be attached. The expression (19.32) is known as the **cycle structure** of the permutation in question. It is common practice to omit a factor $y_i^{\alpha_i}$ from (19.32) if $\alpha_i = 0$ or to write y_i instead of y_i^1 if $\alpha_i = 1$ for some $i \in [k]$. The cycle structures of the permutations π_1 and π_2 are therefore $y_1 y_3$ and $y_1^2 y_4$, respectively. Similarly, the cycle structures of the set of all $3! = 6$ permutations of degree 3,

$$\left. \begin{array}{lll} \pi_3 = (a)(b)(c), & \pi_4 = (ab)(c), & \pi_5 = (ac)(b), \\ \pi_6 = (a)(bc), & \pi_7 = (abc), & \pi_8 = (acb), \end{array} \right\} \quad (19.33)$$

are

$$y_1^3, \quad y_1 y_2, \quad y_1 y_2, \quad y_1 y_2, \quad y_3 \quad \text{and} \quad y_3, \quad (19.34)$$

respectively.

❖ The reader should now be able to attempt Exercises 19.11–19.12.

19.3.2 Permutation groups and (ordered) pair groups

New permutations may be composed by combining two permutations via the binary operation of function composition, denoted by “ \circ .” Consider, for example, the permutations π_4 and π_5 in (19.33). Since

$$\begin{aligned} (\pi_4 \circ \pi_5)(a) &= \pi_4(\pi_5(a)) = \pi_4(c) = c \\ (\pi_4 \circ \pi_5)(b) &= \pi_4(\pi_5(b)) = \pi_4(b) = a \\ (\pi_4 \circ \pi_5)(c) &= \pi_4(\pi_5(c)) = \pi_4(a) = b \end{aligned}$$

it follows that the **composition** of the permutations π_4 and π_5 is

$$\pi_4 \circ \pi_5 = \begin{pmatrix} a & b & c \\ c & a & b \end{pmatrix} = (acb) = \pi_8.$$

A set of permutations P of degree k forms a **group** under the binary operation “ \circ ,” called a **permutation group** and denoted by (P, \circ) , if

- (1) $\pi \circ \pi' \in P$ for any permutations $\pi, \pi' \in P$;
- (2) $\pi \circ (\pi' \circ \pi'') = (\pi \circ \pi') \circ \pi''$ for any permutations $\pi, \pi', \pi'' \in P$;
- (3) there exists a permutation $e \in P$, called the **identity permutation**, such that $e \circ \pi = \pi \circ e = \pi$ for all permutations $\pi \in P$; and
- (4) there exists, for each permutation $\pi \in P$, a permutation $\pi^{-1} \in P$ called an **inverse permutation** of π , such that $\pi \circ \pi^{-1} = \pi^{-1} \circ \pi = e$.

The cardinality $|P|$ of the set P is called the **order** of the permutation group (P, \circ) , and the degree k of the permutations is also the **degree** of the permutation group. It may be shown that the six permutations in (19.33) form a group of degree 3 and order 6. This is a well-known group called the **full symmetric permutation group of degree 3**, denoted by (S_3, \circ) or S_3 for short, and its composition table is shown in Table 19.13. The identity permutation in any permutation group is the permutation that maps every element to itself — in this case π_3 . In general, the identity permutation of degree k therefore comprises k cycles of length 1, i.e. is of type $(k, 0, \dots, 0)$, has cyclic representation $(1)(2)(3)\cdots(k)$ and has cycle structure y_1^k . It follows from Table 19.13 that π_7 and π_8 are inverses of one another in S_3 , and that π_3, π_4, π_5 and π_6 are each their own inverses.

\circ	π_3	π_4	π_5	π_6	π_7	π_8
π_3	π_3	π_4	π_5	π_6	π_7	π_8
π_4	π_4	π_3	π_8	π_7	π_6	π_5
π_5	π_5	π_7	π_3	π_8	π_4	π_6
π_6	π_6	π_8	π_7	π_3	π_5	π_4
π_7	π_7	π_5	π_6	π_4	π_8	π_3
π_8	π_8	π_6	π_4	π_5	π_3	π_7

Table 19.13: Composition table for the full symmetric permutation group S_3 of degree 3, with elements as shown in (19.33).

It holds more generally that the set of all $k!$ permutations of degree k forms a group under function composition. This group, which has degree k and order $k!$, is known as the **full symmetric permutation group of degree k** and is denoted by (S_k, \circ) or S_k for short.

A permutation group (P', \circ) is called a **permutation subgroup** of another permutation group (P, \circ) if $P' \subseteq P$. The set of permutations $S'_3 = \{\pi_3, \pi_7, \pi_8\}$ in (19.33) forms a permutation subgroup (S'_3, \circ) of the full symmetric permutation group (S_3, \circ) of degree 3 under function composition; its composition table is shown in Table 19.14 (an extract from Table 19.13).

Since edges join *pairs* of vertices in graphs, another group, called the pair group, will be of interest to us. Consider, as example, again the permutation $\pi_1 = (adc)(b)$ of degree 4 on the set $A_1 = \{a, b, c, d\}$ shown in Figure 19.10. This permutation induces the **pair permutation**

$$\pi_1^* = \begin{pmatrix} ab & ac & ad & bc & bd & cd \\ bd & ad & cd & ab & bc & ac \end{pmatrix} = (abbd)(acadcd)$$

○	π_3	π_7	π_8
π_3	π_3	π_7	π_8
π_7	π_7	π_8	π_3
π_8	π_8	π_3	π_7

Table 19.14: Composition table for a subgroup of the full symmetric permutation group S_3 of degree 3, with elements as shown in (19.33).

of degree $\binom{4}{2} = 6$ on the set $\{ab, ac, ad, bc, bd, cd\}$ of six *unordered* pairs from A_1 , whose directed pseudograph representation is shown in Figure 19.11.

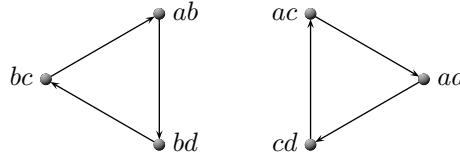


Figure 19.11: The directed pseudograph representation of the pair permutation $\pi_1^* = (ab\ bd\ bc)(ac\ ad\ cd)$ induced by the permutation $\pi_1 = (adc)(b)$ of degree 4 in Figure 19.10.

It may be shown that the set of pair permutations induced by a permutation group under composition again forms a group under composition (see Exercise 19.16). In particular, the set of pair permutations induced by the full symmetric permutation group S_k of degree k forms a group known as the **pair group** R_k , which has degree $\binom{k}{2}$ and order $k!$. For example, the elements of the pair group R_3 , induced by the six permutations in S_3 (see (19.33)), are

$$\left. \begin{aligned} \pi_3^* &= (ab)(ac)(bc), & \pi_4^* &= (ab)(ac\ bc), & \pi_5^* &= (ab\ bc)(ac), \\ \pi_6^* &= (ab\ ac)(bc), & \pi_7^* &= (ab\ bc\ ac), & \pi_8^* &= (ab\ ac\ bc), \end{aligned} \right\} \quad (19.35)$$

and the corresponding cycle structures are again

$$y_1^3, \quad y_1y_2, \quad y_1y_2, \quad y_1y_2, \quad y_3 \quad \text{and} \quad y_3, \quad (19.36)$$

respectively. Note that the cycle structures of permutations in S_k and those in R_k are not always the same, as happened in this small, special case where $k = 3$.

Finally, since arcs in a directed graph join *ordered pairs* of vertices, one final group is of concern to us, namely the ordered pair group. The permutation $\pi_1 = (adc)(b)$ in Figure 19.10 induces the **ordered pair permutation**

$$\begin{aligned} \hat{\pi}_1 &= (ab\ ac\ ad\ ba\ bc\ bd\ ca\ cb\ cd\ da\ db\ dc) \\ &= (ab\ db\ cb)(ac\ da\ cd)(ad\ dc\ ca)(ba\ bd\ bc) \end{aligned}$$

of degree $4 \times 3 = 12$ on the set $\{ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc\}$ of twelve *ordered* pairs from A_1 , whose directed pseudograph representation is shown in Figure 19.12.

The set of ordered pair permutations induced by the full symmetric permutation group S_k of degree k again forms a group known as the **ordered pair group** M_k ,

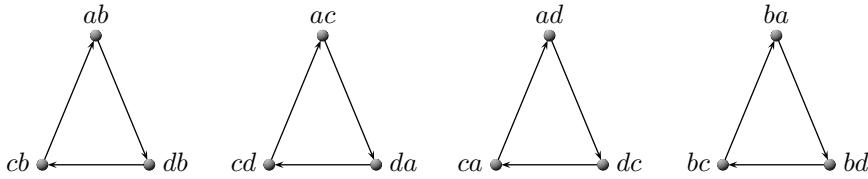


Figure 19.12: The directed pseudograph representation of the ordered pair permutation $\hat{\pi}_1 = (ab\ db\ cb)(ac\ da\ cd)(ad\ dc\ ca)(ba\ bd\ bc)$ induced by the permutation $\pi_1 = (adc)(b)$ of degree 4 in [Figure 19.10](#).

which has degree $k(k - 1)$ and order $k!$. For example, the elements of the ordered pair group M_3 , induced by the six permutations of S_3 in [\(19.33\)](#), are

$$\left. \begin{array}{ll} \hat{\pi}_3 = (ab)(ac)(ba)(bc)(ca)(cd), & \hat{\pi}_4 = (ab\ ba)(ac\ bc)(ca\ cb), \\ \hat{\pi}_5 = (ab\ cb)(ac\ ca)(ba\ bc), & \hat{\pi}_6 = (ab\ ac)(ba\ ca)(bc\ cb), \\ \hat{\pi}_7 = (ab\ bc\ ca)(ac\ ba\ cb), & \hat{\pi}_8 = (ab\ ca\ bc)(ac\ cb\ ba), \end{array} \right\} \quad (19.37)$$

and the corresponding cycle structures are

$$y_1^6, \quad y_2^3, \quad y_2^3, \quad y_2^3, \quad y_3^2 \quad \text{and} \quad y_3^2, \quad (19.38)$$

respectively.

❖ The reader should now be able to attempt Exercises [19.13–19.16](#).

19.3.3 The cycle index of a permutation group

The **cycle index** of a permutation group is merely the sum total of all the cycle structures of permutations appearing in the group, normalised by dividing by the group order. The cycle index of such a group G of degree k is denoted by $Z(G; y_1, \dots, y_k)$ and captures in coded form all the information about the number of cycles of various lengths in the permutations of the group. For example, it follows from [\(19.33\)–\(19.38\)](#) that we may populate [Table 19.15](#) with respect to the permutation groups S_3 , R_3 and M_3 .

Permutation type in S_3	Number of such permutations	Cycle structure	Induced term in R_3	Induced term in M_3
(3, 0, 0)	1	y_1^3	y_1^3	y_1^6
(1, 1, 0)	3	$y_1 y_2$	$y_1 y_2$	y_2^3
(0, 0, 1)	2	y_3	y_3	y_3^2

Table 19.15: Cycle types and structures of elements of the full symmetric permutation group S_3 , with corresponding terms for the pair group R_3 and the ordered pair group M_3 .

From [Table 19.15](#) we have that

$$Z(S_3; y_1, y_2, y_3) = Z(R_3; y_1, y_2, y_3) = \frac{1}{6}(y_1^3 + 3y_1 y_2 + 2y_3), \quad (19.39)$$

whilst

$$Z(M_3; y_1, y_2, y_3) = \frac{1}{6}(y_1^6 + 3y_2^3 + 2y_3^2). \quad (19.40)$$

It is left as an exercise to show that the cycle indices of the full symmetric permutation group S_4 , the pair group R_4 and the ordered pair group M_4 are

$$Z(S_4; y_1, y_2, y_3, y_4) = \frac{1}{24}(y_1^4 + 6y_1^2y_2 + 8y_1y_3 + 3y_2^2 + 6y_4), \quad (19.41)$$

$$Z(R_4; y_1, y_2, y_3, y_4) = \frac{1}{24}(y_1^6 + 9y_1^2y_2^2 + 8y_3^2 + 6y_2y_4) \quad \text{and} \quad (19.42)$$

$$Z(M_4; y_1, y_2, y_3, y_4) = \frac{1}{24}(y_1^{12} + 6y_1^2y_2^5 + 8y_3^4 + 3y_2^6 + 6y_4^3), \quad (19.43)$$

respectively (see [Exercise 19.18](#)). With some perseverance (or the help of a computer!) it may similarly be shown that the cycle indices of the full symmetric permutation group S_5 , the pair group R_5 and the ordered pair group M_5 are

$$\begin{aligned} Z(S_5; y_1, \dots, y_5) \\ = \frac{1}{120}(y_1^5 + 10y_1^3y_2 + 20y_1^2y_3 + 15y_1y_2^2 + 30y_1y_4 + 20y_2y_3 + 24y_5), \end{aligned} \quad (19.44)$$

$$\begin{aligned} Z(R_5; y_1, \dots, y_6) \\ = \frac{1}{120}(y_1^{10} + 10y_1^4y_2^3 + 20y_1y_3^2 + 15y_1^2y_2^4 + 30y_2y_4^2 + 20y_1y_3y_6 + 24y_5^2), \end{aligned} \quad (19.45)$$

$$\begin{aligned} Z(M_5; y_1, \dots, y_6) \\ = \frac{1}{120}(y_1^{20} + 10y_1^6y_2^7 + 15y_2^{10} + 20y_1^2y_3^6 + 30y_4^5 + 24y_5^4 + 20y_2y_3^2y_6^2), \end{aligned} \quad (19.46)$$

respectively.

- ❖ The reader should now be able to attempt Exercises [19.17–19.18](#).

19.3.4 The theorem in the context of graph enumeration

We are now in the position of having gathered enough background information on permutation groups in order to present [Pólya's Enumeration Theorem](#).

In order to state the theorem we require two sets which we denote by \mathcal{D} (called the **domain**) and \mathcal{R} (called the **range**). The elements of \mathcal{R} are called **states**. Because we are aiming towards working with generating functions, we shall associate with each state $r \in \mathcal{R}$ a power of a dummy variable x , called the **weight** of r and denoted by $w(r)$. Define a so-called **configuration** as a function $g : \mathcal{D} \mapsto \mathcal{R}$ and let the **configuration weight** be

$$w(g) = \prod_{v \in \mathcal{D}} w(g(v)).$$

Since every element of the domain \mathcal{D} can be mapped onto any one of the $|\mathcal{R}|$ elements in the range, and since the choices of these mappings are independent for different domain elements, there are $|\mathcal{R}|^{|\mathcal{D}|}$ distinct configurations. For example, there are $2^3 = 8$ distinct configurations g_1, \dots, g_8 from the three-element domain $\mathcal{D}_1 = \{v_1, v_2, v_3\}$ to the two-element range $\mathcal{R}_1 = \{u_1, u_2\}$, and the weights of these configurations are shown in [Table 19.16](#), assuming that $w(u_1) = x^s$ and $w(u_2) = x^t$ for some integers s and t .

If Π is a permutation group on the domain \mathcal{D} , then we say that two configurations f and f' are **Π -equivalent** if there exists some permutation $\pi \in \Pi$ such that

$$f(v) = f'(\pi(v)) \quad (19.47)$$

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
$w(g_i(v_1))$	x^s	x^s	x^s	x^t	x^s	x^t	x^t	x^t
$w(g_i(v_2))$	x^s	x^s	x^t	x^s	x^t	x^s	x^t	x^t
$w(g_i(v_3))$	x^s	x^t	x^s	x^s	x^t	x^t	x^s	x^t
$w(g_i)$	x^{3s}	x^{2s+t}	x^{2s+t}	x^{2s+t}	x^{s+2t}	x^{s+2t}	x^{s+2t}	x^{3t}

Table 19.16: Weights of the $2^3 = 8$ possible mappings g_1, \dots, g_8 from $\mathcal{D} = \{v_1, v_2, v_3\}$ to $\mathcal{R} = \{u_1, u_2\}$, where $w(u_1) = x^s$ and $w(u_2) = x^t$.

for every $v \in \mathcal{D}$ (that is, if the states of f are exactly the states of f' , but in permuted order under an element of Π). It may easily be shown that the relationship defined by (19.47) is an equivalence relation (see Exercise 19.19) and hence that it partitions the set of all possible configurations into equivalence classes, where every member of an equivalence class has the same configuration weight. We define the **equivalence class weight** as the weight of any configuration in the class. For the example in Table 19.16 under the permutation group $\Pi = (\{(v_1)(v_2)(v_3), (v_1 v_2 v_3), (v_1 v_3 v_2)\}, \circ)$, the equivalence classes are: $\{g_1\}$, $\{g_2, g_3, g_4\}$, $\{g_5, g_6, g_7\}$, and $\{g_8\}$.

The following theorem is due to Pólya [8], dates back to 1937 and counts the number of equivalence classes of various weights.

Theorem 19.7 (Pólya's Enumeration Theorem) *Suppose Π is a permutation group of degree k acting on the domain \mathcal{D} and that*

$$f(x) = \sum_{i=0}^{\infty} f_i x^i$$

*is the **state generating function** describing the weight assignment to elements of the range \mathcal{R} (where f_i denotes the number of distinct elements in \mathcal{R} which receive the weight x^i). Furthermore, suppose*

$$F(x) = \sum_{j=0}^{\infty} F_j x^j$$

*is the **configuration generating function** describing the weight assignments to configurations (in the sense that F_j denotes the number of distinct configurations with weight x^j). Then*

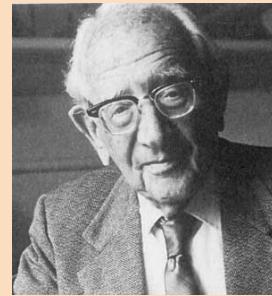
$$F(x) = Z\left(\Pi; \sum_{i=0}^{\infty} f_i x^i, \sum_{i=0}^{\infty} f_i x^{2i}, \sum_{i=0}^{\infty} f_i x^{3i}, \dots, \sum_{i=0}^{\infty} f_i x^{ki}\right).$$

That is, $F(x)$ is obtained by substituting $f(x^\ell)$ for y_ℓ in the cycle index $Z(\Pi; y_1, \dots, y_k)$ of the permutation group Π , for all $\ell \in [k]$.

In the section that follows we shall apply Theorem 19.7 to the problem of enumerating graphs. In this setting, the elements of the domain \mathcal{D} will be the vertices of a graph, the configurations will be graphs, and the states in the range \mathcal{R} will describe adjacency between vertices, assigning, for example, the weights x^0 for nonadjacency, x^1 for adjacency, x^2 for two edges between a pair of vertices (in the case of a multigraph), and so on.

❖ The reader should now be able to attempt Exercise 19.19.

George Pólya was born in Budapest, Hungary on 13 December 1887. He enrolled at the University of Budapest in 1905 where he first studied languages and later physics and mathematics. He obtained a doctorate in mathematics at Budapest in 1912. After spending some time in Göttingen and Paris, he took Swiss citizenship during World War I and was appointed professor at ETH, Zürich in 1928. In 1933, Pólya was awarded a (second) Rockefeller Fellowship to visit Princeton. While he was in the United States, he was invited to Stanford and greatly enjoyed his stay there. He returned to Zürich, but in 1940 the political landscape in Europe forced him to move to the United States where, after working at Brown University and Smith College, he took up an appointment at Stanford. In 1953, Pólya retired from Stanford, but continued a very active life in mathematics, particularly concerning himself with mathematical education. He is best known for his work in geometric symmetry and the enumeration of symmetry classes of objects — he added to the understanding of the seventeen plane crystallographic groups in 1924 by illustrating each with tilings of the plane, inspiring MC Escher to produce his well-known work on periodic drawings. He died in Palo Alto, California on 7 September 1985 at the age of 97.



Biographic note 102: George Pólya (1887–1985)

19.4 Using Pólya's theorem to enumerate graphs

In this section, we apply [Pólya's Enumeration Theorem](#), as described in the previous section, to the problems of enumerating three general classes of graphs, namely (i) unlabelled graphs of order n and size m , (ii) unlabelled multigraphs of order n , size m and edge multiplicity at most ϵ , and (iii) unlabelled digraphs of order n and size m .

19.4.1 Unlabelled graphs of specified order and size

Consider the problem of enumerating unlabelled graphs of order n and size m . Since any graph may be considered a mapping or configuration of the set of all $\binom{n}{2}$ *unordered* vertex pairs to a state of *nonadjacent* or *adjacent*, the domain \mathcal{D} in [Pólya's theorem](#) is the set of all unordered pairs of an n -set, whilst its range \mathcal{R} comprises two elements with weights x^0 (representing *nonadjacent*) and x^1 (representing *adjacent*). The state generating function is therefore

$$f_{\text{gr}}(x) = \sum_{i=1}^{\infty} f_{\text{gr},i} x^i = 1 + x.$$

Since the edges of a graph are *unordered* two-element subsets of its vertex set, the relevant permutation group in this case is the pair group R_n . According to [Theorem 19.7](#), the configuration generating function $F_{\text{gr}}^{(n)}(x)$ for unlabelled graphs of order n is obtained by substituting $y_1 = 1 + x$, $y_2 = 1 + x^2$, $y_3 = 1 + x^3$, and so on,

into the cycle index of R_n , where y_1, y_2, y_3, \dots are the dummy variables appearing in the cycle index. For example, it follows from (19.39), (19.42) and (19.45) that

$$\begin{aligned} F_{\text{gr}}^{(3)}(x) &= \frac{1}{6}((1+x)^3 + 3(1+x)(1+x^2) + 2(1+x^3)) \\ &= 1 + x + x^2 + x^3, \end{aligned} \quad (19.48)$$

$$\begin{aligned} F_{\text{gr}}^{(4)}(x) &= \frac{1}{24}((1+x)^6 + 9(1+x)^2(1+x^2)^2 + 8(1+x^3)^2 \\ &\quad + 6(1+x^2)(1+x^4)) \\ &= 1 + x + 2x^2 + 3x^3 + 2x^4 + x^5 + x^6, \text{ and} \end{aligned} \quad (19.49)$$

$$\begin{aligned} F_{\text{gr}}^{(5)}(x) &= \frac{1}{120}((1+x)^{10} + 10(1+x)^4(1+x^2)^3 + 20(1+x)(1+x^3)^3 \\ &\quad + 15(1+x)^2(1+x^2)^4 + 30(1+x^2)(1+x^4)^2 \\ &\quad + 20(1+x)(1+x^3)(1+x^6) + 24(1+x^5)^2) \\ &= 1 + x + 2x^2 + 4x^3 + 6x^4 + 6x^5 + 6x^6 + 4x^7 + 2x^8 + x^9 + x^{10}. \end{aligned} \quad (19.50)$$

The coefficient of x^m in $F_{\text{gr}}^{(n)}(x)$ represents the number of distinct unlabelled graphs of order n and size m . The graph generating functions $F_{\text{gr}}^{(3)}(x)$, $F_{\text{gr}}^{(4)}(x)$ and $F_{\text{gr}}^{(5)}(x)$ appear in Figure 19.13 with the graphs enumerated shown above each term of these generating functions. It may be seen from the figure (or by adding together the coefficients of the various graph generating functions) that there are 4, 11 and 34 distinct unlabelled graphs of orders 3, 4 and 5, respectively. These numbers form part of the sequence¹² shown in Table 19.17.

❖ The reader should now be able to attempt Exercise 19.20.

19.4.2 Multigraphs of specified order, size and multiplicity

Let us consider next the problem of enumerating unlabelled multigraphs of order n , size m and edge multiplicity at most ϵ (*i.e.* multigraphs that have at most ϵ edges between any pair of vertices). Since any such multigraph may be considered a mapping or configuration from the set of all $\binom{n}{2}$ *unordered* vertex pairs to a state of multiplicity 0 (or nonadjacent), multiplicity 1, multiplicity 2, …, multiplicity ϵ , the domain \mathcal{D} in Pólya's theorem is the set of all ordered pairs of an n -set, whilst its range \mathcal{R} comprises $\epsilon + 1$ elements with weights x^0 (representing multiplicity 0), x^1 (representing multiplicity 1), x^2 (representing multiplicity 2), …, x^ϵ (representing multiplicity ϵ). The state generating function is therefore

$$f_{\text{mg}}^{(\epsilon)}(x) = \sum_{i=1}^{\infty} f_{\text{mg},i}^{(\epsilon)} x^i = 1 + x + x^2 + \cdots + x^\epsilon$$

and the relevant permutation group in this case is again the pair group R_n . According to Theorem 19.7 the configuration generating function $F_{\text{mg}}^{(n,\epsilon)}(x)$ for multigraphs of order n and edge multiplicity at most ϵ is obtained by substituting $y_1 = f_{\text{mg}}^{(\epsilon)}(x)$, $y_2 = f_{\text{mg}}^{(\epsilon)}(x^2)$, $y_3 = f_{\text{mg}}^{(\epsilon)}(x^3)$, and so on, into the cycle index of R_n , where y_1, y_2, y_3, \dots are the dummy variables appearing in the cycle index. For

¹²Sequence A000088 of Sloane [10].

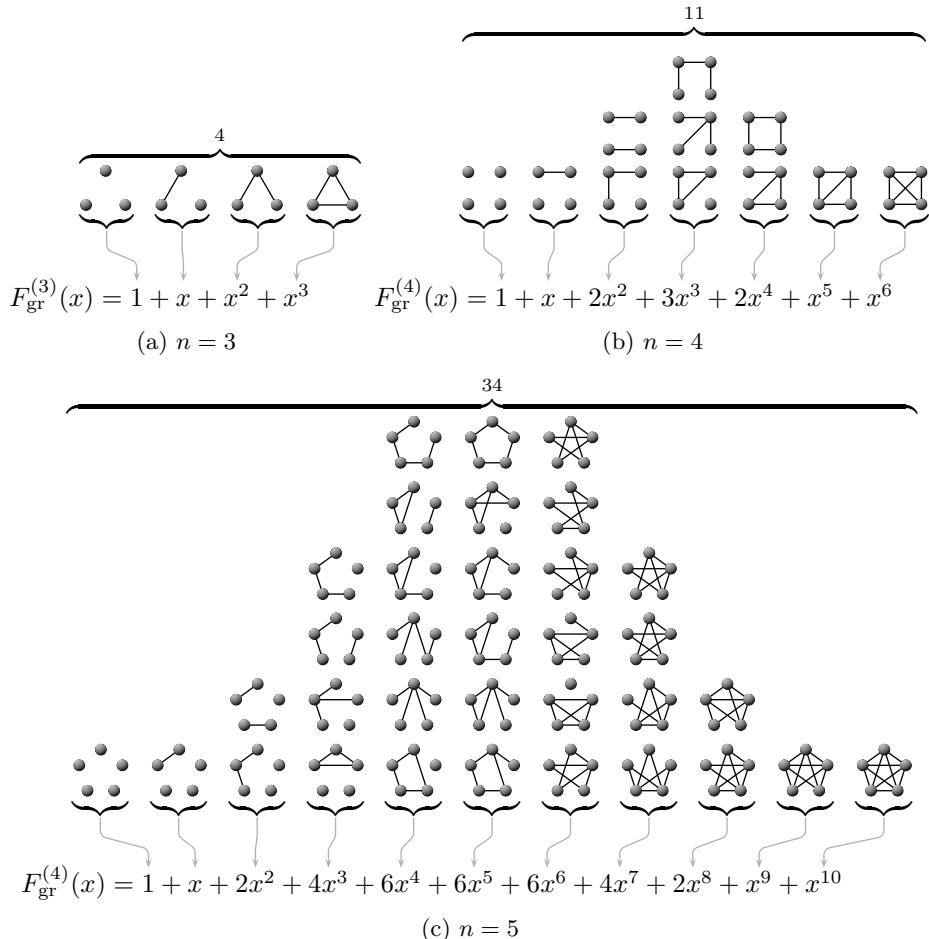


Figure 19.13: Unlabelled graphs of order $n \in \{3, 4, 5\}$.

Order	1	2	3	4	5	6	7	8	9	10
Graphs	1	2	4	11	34	156	1 044	12 346	274 668	12 005 168

Table 19.17: The number of unlabelled graphs of order $n \in [10]$.

example, it follows from (19.39) and (19.42) that

$$\begin{aligned} F_{\text{mg}}^{(3,2)}(x) &= \frac{1}{6}((1+x+x^2)^3 + 3(1+x+x^2)(1+x^2+x^4) + 2(1+x^3+x^6)) \\ &= 1+x+2x^2+2x^3+2x^4+x^5+x^6, \end{aligned} \quad (19.51)$$

$$\begin{aligned} F_{\text{mg}}^{(3,3)}(x) &= \frac{1}{6}((1+x+x^2+x^3)^3 + 2(1+x^3+x^6+x^9) \\ &\quad + 3(1+x+x^2+x^3)(1+x^2+x^4+x^6)) \\ &= 1+x+2x^2+3x^3+3x^4+3x^5+3x^6+2x^7+x^8+x^9, \end{aligned} \quad (19.52)$$

$$\begin{aligned} F_{\text{mg}}^{(4,2)}(x) &= \frac{1}{24}((1+x+x^2)^6 + 9(1+x+x^2)^2(1+x^2+x^4)^2 \\ &\quad + 8(1+x^3+x^6)^2 + 6(1+x^2+x^4)(1+x^4+x^8)) \\ &= 1+x+3x^2+5x^3+8x^4+9x^5+12x^6+9x^7+8x^8 \\ &\quad + 5x^9+3x^{10}+x^{11}+x^{12}, \text{ and} \end{aligned} \quad (19.53)$$

$$\begin{aligned} F_{\text{mg}}^{(4,3)}(x) &= \frac{1}{24}((1+x+x^2+x^3)^6 + 8(1+x^3+x^6+x^9)^2 \\ &\quad + 9(1+x+x^2+x^3)^2(1+x^2+x^4+x^6)^2 \\ &\quad + 6(1+x^2+x^4+x^6)(1+x^4+x^8+x^{12})) \\ &= 1+x+3x^2+6x^3+10x^4+15x^5+23x^6+28x^7+33x^8 \\ &\quad + 36x^9+33x^{10}+28x^{11}+23x^{12}+15x^{13}+10x^{14}+6x^{15} \\ &\quad + 3x^{16}+x^{17}+x^{18}. \end{aligned} \quad (19.54)$$

The coefficient of x^m in $F_{\text{mg}}^{(n,\epsilon)}(x)$ represents the number of distinct unlabelled multigraphs of order n , size m and edge multiplicity at most ϵ . The multigraph generating functions $F_{\text{mg}}^{(3,2)}(x)$ and $F_{\text{mg}}^{(4,2)}(x)$ appear in Figure 19.14 with the multigraphs enumerated shown above each term of these generating functions. By adding together the coefficients of the various graph generating functions in (19.51)–(19.52) we find that there are 10 and 66 distinct unlabelled multigraphs of orders 3 and 4 respectively, with edge multiplicity at most 2. These numbers form part of the sequence¹³ shown in the second row of Table 19.18. Similarly, there are 20 and 276 distinct unlabelled multigraphs of orders 3 and 4 respectively, with edge multiplicity at most 3, and these numbers form part of the sequence¹⁴ shown in the third row of Table 19.18. The first row of the table is, of course, the number of unlabelled graphs, enumerated in Section 19.4.1.

❖ The reader should now be able to attempt Exercise 19.21.

19.4.3 Unlabelled digraphs of specified order and size

In this final section we consider the problem of enumerating unlabelled digraphs of order n and size m . Since any digraph may be considered a mapping of the set of all $n(n-1)$ ordered vertex pairs to a state of nonadjacent or adjacent, the domain \mathcal{D} in Pólya's theorem is the set of all ordered pairs of an n -set, whilst its range \mathcal{R} comprises two elements with weights x^0 (representing nonadjacent) and x^1

¹³Sequence A004102 of Sloane [10].

¹⁴Sequence A053400 of Sloane [10].

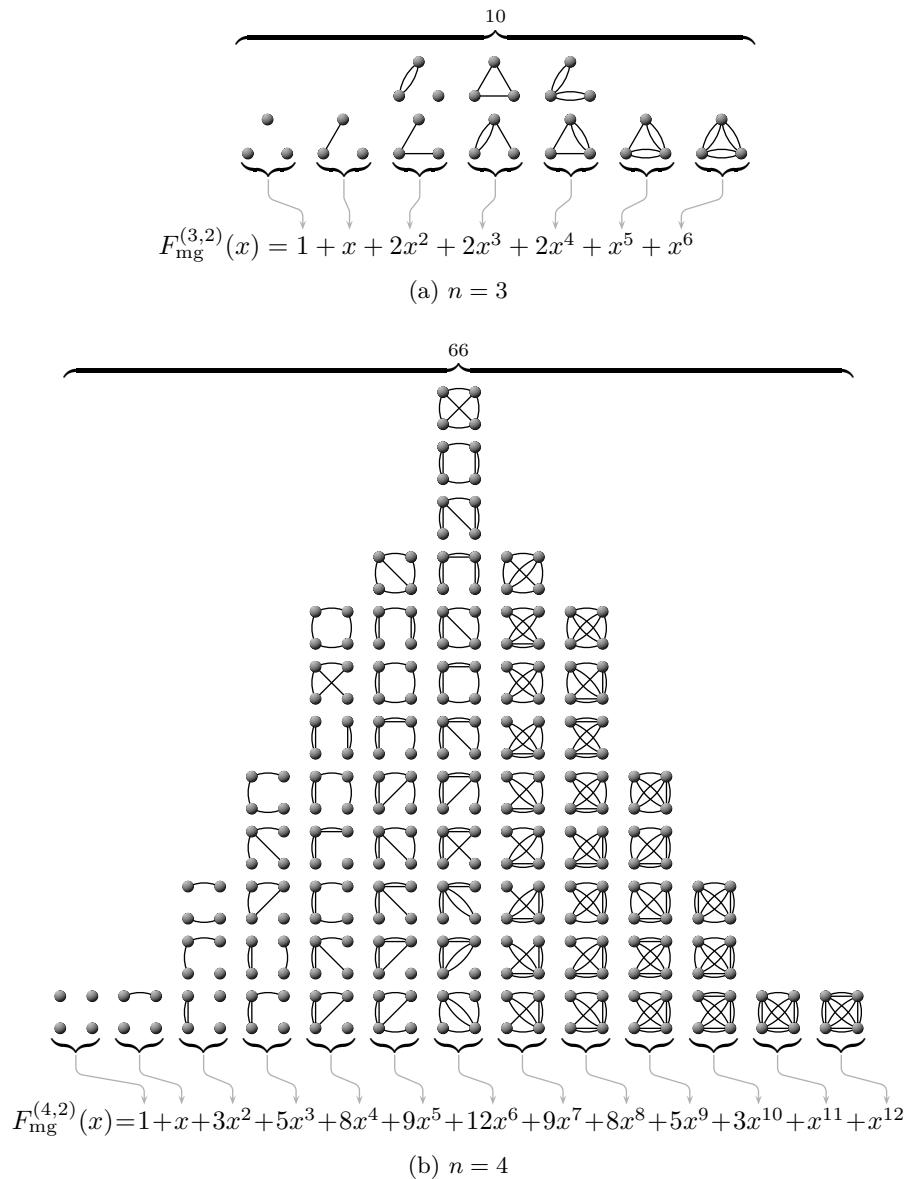


Figure 19.14: Unlabelled multigraphs of order $n \in \{3, 4\}$ and edge multiplicity at most $\epsilon = 2$.

Order	1	2	3	4	5	6	7	8
$\epsilon = 1$	1	2	4	11	34	156	1 044	12 346
$\epsilon = 2$	1	3	10	66	792	25 506	2 302 938	591 901 884
$\epsilon = 3$	1	4	20	276	10 688	1 601 952	892 341 888	1 799 786 093 088
$\epsilon = 4$	1	5	35	900	90 005	43 571 400	95 277 592 625	925 609 100 039 625
$\epsilon = 5$	1	6	56	2 451	533 358	661 452 084	4 364 646 955 812	152 397 092 027 960 154

Table 19.18: The number of unlabelled multigraphs of order $n \in [8]$ with edge multiplicity at most $\epsilon \in [5]$. For $\epsilon = 4$ and $\epsilon = 5$, see sequences [A053420](#) and [A053421](#), respectively of [Sloane](#) [10].

(representing adjacent) as was the case when enumerating unlabelled graphs in [Section 19.4.1](#). The state generating function is therefore again

$$f_{\text{di}}(x) = \sum_{i=1}^{\infty} f_{\text{di},i} x^i = 1 + x,$$

but the relevant permutation group in this case is the ordered pair group M_n . According to [Theorem 19.7](#), the configuration generating function $F_{\text{di}}^{(n)}(x)$ for digraphs of order n is obtained by substituting $y_1 = 1 + x$, $y_2 = 1 + x^2$, $y_3 = 1 + x^3$, and so on, into the cycle index of M_n , where y_1, y_2, y_3, \dots are the dummy variables appearing in the cycle index. For example, it follows from [\(19.40\)](#), [\(19.43\)](#) and [\(19.46\)](#) that

$$\begin{aligned} F_{\text{di}}^{(3)}(x) &= \frac{1}{6}((1+x)^6 + 3(1+x^2)^3 + 2(1+x^3)^2) \\ &= 1 + x + 4x^2 + 4x^3 + 4x^4 + x^5 + x^6, \end{aligned} \tag{19.55}$$

$$\begin{aligned} F_{\text{di}}^{(4)}(x) &= \frac{1}{24}((1+x)^{12} + 6(1+x)^2(1+x^2)^5 + 8(1+x^3)^4 \\ &\quad + 3(1+x^2)^6 + 6(1+x^4)^3) \\ &= 1 + x + 5x^2 + 13x^3 + 27x^4 + 38x^5 + 48x^6 + 38x^7 + 27x^8 \\ &\quad + 13x^9 + 5x^{10} + x^{11} + x^{12}, \text{ and} \end{aligned} \tag{19.56}$$

$$\begin{aligned} F_{\text{di}}^{(5)}(x) &= \frac{1}{120}((1+x)^{20} + 10(1+x)^6(1+x^2)^7 + 15(1+x^2)^{10} + 20(1+x)^2(1+x^3)^6 \\ &\quad + 30(1+x^4)^5 + 24(1+x^5)^4 + 20(1+x^2)(1+x^3)^2(1+x^6)^2) \\ &= 1 + x + 5x^2 + 16x^3 + 61x^4 + 154x^5 + 379x^6 + 707x^7 + 1155x^8 \\ &\quad + 1490x^9 + 1670x^{10} + 1490x^{11} + 1155x^{12} + 707x^{13} + 379x^{14} \\ &\quad + 154x^{15} + 61x^{16} + 16x^{17} + 5x^{18} + x^{19} + x^{20}. \end{aligned} \tag{19.57}$$

The coefficient of x^m in $F_{\text{di}}^{(n)}(x)$ represents the number of distinct unlabelled digraphs of order n and size m . The digraph generating function $F_{\text{di}}^{(3)}(x)$ appears in [Figure 19.15](#) with the digraphs enumerated shown above each term of the generating function. By adding together the coefficients of the various graph generating functions in [\(19.55\)–\(19.57\)](#) we find that there are 16, 218 and 9 608 distinct unla-

belled digraphs of orders 3, 4 and 5 respectively. These numbers form part of the sequence¹⁵ shown in [Table 19.19](#).

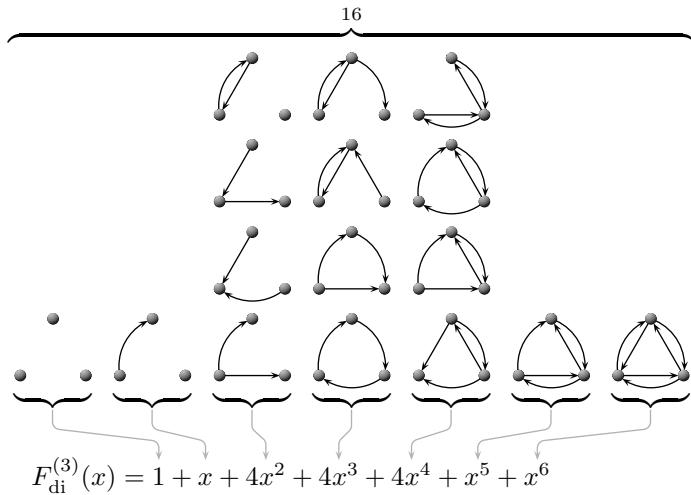


Figure 19.15: Unlabelled digraphs of order $n = 3$.

Order	1	2	3	4	5	6	7	8
Digraphs	1	3	16	218	9 608	1 540 944	882 033 440	1 793 359 192 848

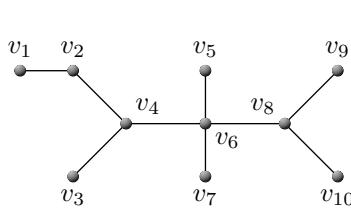
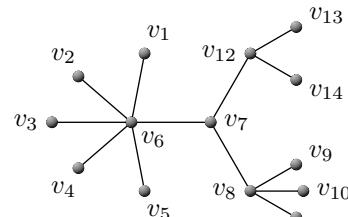
Table 19.19: The number of unlabelled digraphs of order $n \in [8]$.

- ❖ The reader should now be able to attempt [Exercise 19.22](#).

Exercises

- 19.1 Complete [Table 19.2](#).
- 19.2 Prove [Corollary 19.6](#).
- 19.3 Write down the Prüfer codes of the labelled trees $T_{19.7}$ and $T_{19.8}$ in [Figure 19.16](#).
- 19.4 Draw the labelled trees corresponding to the following Prüfer codes:
 - (a) (1, 2, 3, 4, 5); and
 - (b) (1, 2, 2, 3, 3, 3).
- 19.5 Research five different applications of the well-known Catalan numbers in addition to enumerating genealogical registers. (Hint: An internet search using the search phrase “Catalan numbers” should yield many more than five applications.)

¹⁵Sequence [A000273](#) of Sloane [10].

T_{19.7}T_{19.8}**Figure 19.16:** Example graphs associated with Exercise 19.3.

- 19.6 Write down all the partitions of the integer 8 using the standard notation $[1^{a_1} 2^{a_2} \dots n^{a_n}]$ described in Section 19.2.1. (Hint: There are 22 such partitions.)
- 19.7 Show that there are $\binom{n+k-1}{k}$ different ways of selecting k objects from n distinct objects, when repetitions are allowed.
- 19.8 (a) Compute r_6 , the number of distinct, rooted unlabelled trees of order 6 using the recurrence relation (19.15) directly.
(b) Draw all the trees enumerated in (a).
- 19.9 Use the generalised binomial theorem in (19.9) to verify the correctness of the identity in (19.21).
- 19.10 Extend the computations in Figure 19.8 in order to obtain the power series $r^{(10)}(x)$. Use this power series to verify that the number of rooted unlabelled trees of order 11 is $r_{11} = 1842$.
- 19.11 Write each of the following permutations in cyclic representation:
(a) $(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 1 & 2 & 6 & 4 & 3 \end{smallmatrix})$;
(b) $(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 4 & 5 & 3 & 2 & 1 \end{smallmatrix})$; and
(c) $(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 2 & 6 & 5 & 1 \end{smallmatrix})$.
- 19.12 Write down the cycle structure of each of the permutations in Exercise 19.11.
- 19.13 Which of the following sets of permutations form groups under the operation of permutation composition? Motivate.
(a) $\{(12345), (124)(35)\}$;
(b) $\{(1)(2)(3)(4)(5), (12345), (13524), (14253), (15432)\}$; and
(c) $\{(1)(2)(3)(4)(5), (12)(34), (13)(24), (14)(23)\}$.
- 19.14 (a) Verify that the permutation set $\{(a)(b)(c)(d), (ac)(bd), (abcd), (adcb)\}$ forms a group under the binary operation of permutation composition.
(b) What is the degree of the group? Motivate.
(c) What is the order of the group? Motivate.
(d) Write down the cycle structure of each element of the permutation group.
- 19.15 Write down all the permutation subgroups of the full symmetric permutation group S_3 of degree 3. (Hint: There are six such subgroups.)

19.16 Prove that the set of pair permutations induced by a permutation group under composition again forms a group under composition.

19.17 Write down the cycle index $Z(G; y_1, y_2, y_3, y_4)$ of the group G in [Exercise 19.14](#).

- 19.18 (a) Write down, in cycle form, all $4! = 24$ permutations of degree 4.
- (b) Write down the cycle structures of all the permutations in (a).
- (c) Use your answer in (b) above to construct a table similar to [Table 19.15](#), but for the groups S_4 , R_4 and M_4 .
- (d) Use your table in (c) above to verify the correctness of the expressions for the cycle indices $Z(S_4; y_1, y_2, y_3, y_4)$, $Z(R_4; y_1, y_2, y_3, y_4)$ and $Z(M_4; y_1, y_2, y_3, y_4)$ in [\(19.41\)–\(19.43\)](#).

19.19 Prove that the relation defined in [\(19.47\)](#) is an equivalence relation.

19.20 (a) Assume (without proving it!) that the cycle index of the pair group R_6 is

$$\begin{aligned} Z(R_6; y_1, \dots, y_6) = & \frac{1}{720} (y_1^{15} + 15y_1^7y_2^4 + 60y_1^3y_2^6 + 40y_1^3y_3^4 \\ & + 40y_3^5 + 180y_1y_2y_4^3 + 144y_5^3 \\ & + 120y_1y_2y_3^2y_6 + 120y_3y_6^2) \end{aligned}$$

and use [Pólya's Enumeration Theorem \(Theorem 19.7\)](#) to prove that there are 156 distinct graphs of order 6.

- (b) Draw all the graphs enumerated in (a). (Hint: There are 1, 1, 2, 5, 9, 15, 21, 24, 24, 21, 15, 9, 5, 2, 1, 1 unlabelled graphs of order 6 and of sizes 0, ..., 15 respectively.)
- 19.21 (a) Use [Pólya's Enumeration Theorem \(Theorem 19.7\)](#) to count the number of multigraphs of order 3 and size m if there is no restriction on edge multiplicity. Work accurately to $\mathcal{O}(x^{11})$ and tabulate the number of multigraphs of order 3 and sizes $m \in \{0, \dots, 10\}$.
- (b) Draw all the multigraphs tabulated in (a) above.
- (c) Repeat (a), but for multigraphs of order 4 instead.

19.22 Assume (without proving it!) that the cycle index of the ordered pair group M_6 is given by

$$\begin{aligned} Z(M_6; y_1, \dots, y_6) = & \frac{1}{720} (y_1^{30} + 15y_1^{12}y_2^9 + 45y_1^2y_2^{14} + 15y_2^{15} + 40y_1^6y_3^8 \\ & + 40y_3^{10} + 90y_1^2y_4^7 + 90y_2y_4^7 + 144y_5^6 \\ & + 120y_1^3y_3^4y_6^2 + 120y_6^5) \end{aligned}$$

and use [Pólya's Enumeration Theorem \(Theorem 19.7\)](#) to prove that there are 1540944 distinct digraphs of order 6 (without drawing them!).

Computer exercises

The **MATHEMATICA** command **PartitionsP**[n] produces the number of partitions of the integer n , while the command **IntegerPartitions**[n] produces a list of all these partitions. The latter command may also be supplied with a number of optional arguments. For instance, **IntegerPartitions**[n, k] yields the partitions of the integer n into at most k nonzero parts, while the slightly different **IntegerPartitions**[$n, \{k\}$] yields the partitions of n into exactly k nonzero parts. Moreover, **IntegerPartitions**[$n, \{k_{\min}, k_{\max}\}$] produces a list of partitions of n into between k_{\min} and k_{\max} nonzero parts (inclusive). For example, the commands

```
In[1]:= PartitionsP[5]
In[2]:= IntegerPartitions[5]
In[3]:= IntegerPartitions[5, 3]
In[4]:= IntegerPartitions[5, {3}]
In[5]:= IntegerPartitions[5, {2, 3}]
```

produce the output:

```
Out[1]:= 7
Out[2]:= {{5}, {4, 1}, {3, 2}, {3, 1, 1}, {2, 2, 1}, {2, 1, 1, 1}, {1, 1, 1, 1, 1}}
Out[3]:= {{5}, {4, 1}, {3, 2}, {3, 1, 1}, {2, 2, 1}}
Out[4]:= {{3, 1, 1}, {2, 2, 1}}
Out[5]:= {{4, 1}, {3, 2}, {3, 1, 1}, {2, 2, 1}}
```

The command **CatalanNumber**[n] returns the n -th Catalan number. For example, the command

```
In[6]:= CatalanNumber[5]
```

yields as output:

```
Out[6]:= 42
```

The command **Series**[$f(x)$, $\{x, x_0, n\}$] generates a power series expansion for the function $f(x)$ about the point $x = x_0$ up to order $(x - x_0)^n$. For instance, the command

```
In[7]:= Series[(1 - 2 x^2)^(-2), {x, 0, 20}]
```

returns

```
Out[7]:= 1 + 4 x^2 + 12 x^4 + 32 x^6 + 80 x^8 + 192 x^10 + 448 x^12 + 1024 x^14 +
2304 x^16 + 5120 x^18 + 11264 x^20 + 0[x]^21
```

as output. The command **Simplify**[$f(x)$] performs a sequence of algebraic and other transformations on an expression $f(x)$ and returns the simplest form it finds, while the command **Expand**[$f(x)$] expands out products and positive integer powers in the expression $f(x)$. For example, the commands

```
In[8]:= Simplify[x^2 + 2 x + 1]
In[9]:= Expand[(1 + x^2)^3 (1 + x^4)^4]
```

return the output:

```
Out[8]:= (1 + x)^2
Out[9]:= 1 + 3 x^2 + 7 x^4 + 13 x^6 + 18 x^8 + 22 x^10 + 22 x^12 + 18 x^14 + 13 x^16 +
7 x^18 + 3 x^20 + x^22
```

The command **Permutations**[list] produces a list of all possible permutations of the elements in list. The command **Range**[n] may, furthermore, be used to generate the list $\{1, \dots, n\}$. For instance, the commands

```
In[10]:= Permutations[{1, 2, 3}]
In[11]:= Permutations[Range[4]]
```

produce the output:

```
Out[10]:= {{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}}
Out[11]:= {{1, 2, 3, 4}, {1, 2, 4, 3}, {1, 3, 2, 4}, {1, 3, 4, 2}, {1, 4, 2, 3}, {1,
4, 3, 2}, {2, 1, 3, 4}, {2, 1, 4, 3}, {2, 3, 1, 4}, {2, 3, 4, 1}, {2,
4, 1, 3}, {2, 4, 3, 1}, {3, 1, 2, 4}, {3, 1, 4, 2}, {3, 2, 1, 4}, {3,
2, 4, 1}, {3, 4, 1, 2}, {3, 4, 2, 1}, {4, 1, 2, 3}, {4, 1, 3, 2}, {4,
2, 1, 3}, {4, 2, 3, 1}, {4, 3, 1, 2}, {4, 3, 2, 1}}
```

The command **PermutationCycles**[perm] returns as output a disjoint cycle representation of the permutation perm, while the command **PermutationList**[perm] produces a permutation list representation of perm. For example, the commands

```
In[12]:= PermutationCycles[{2, 1, 3}]
In[13]:= PermutationList[Cycles[{3, 1}]]
```

return as output:

```
Out[12]:= Cycles[{{1, 2}}]
Out[13]:= {3, 2, 1}
```

The command **SymmetricGroup**[n] // **GroupElements** returns a cycle representation of the elements of the symmetric permutation group S_n . For example, the command

```
In[14]:= SymmetricGroup[3] // GroupElements
```

returns as output:

```
Out[14]:= {Cycles[{}], Cycles[{{2, 3}}], Cycles[{{1, 2}}], Cycles[{{1, 2, 3}}],
Cycles[{{1, 3, 2}}], Cycles[{{1, 3}}]}
```

Lists of pre-generated graphs may be imported into **MATHEMATICA** in so-called *graph6* format — a format for storing undirected graphs in a compact manner, using only printable ASCII characters. More information on this format may be obtained from [Brendan Mckay's website](#) [6]. The command **Import** may be used to import graphs into **MATHEMATICA** directly from this website in *graph6* format. For example, the commands

```
In[15]:= glist5 = Import["https://users.cecs.anu.edu.au/~bdm/data/graph5c.g6"];
In[16]:= Length[glist5]
```

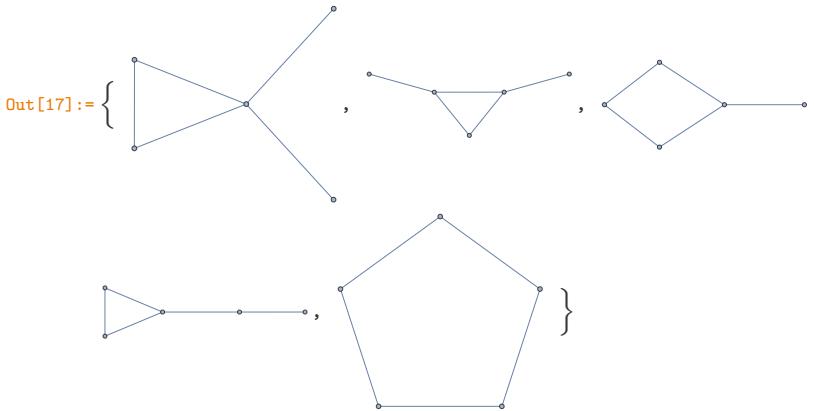
may be used to import a list of all connected graphs of order 5 into **MATHEMATICA** and count them (via the command **Length**). The commands return as output the value:

```
Out[16]:= 21
```

A selection or all of these graphs may then be displayed graphically by means of the command **Select**. For example, the command

```
In[17]:= Select[glist5, EdgeCount[#] == 5 &]
```

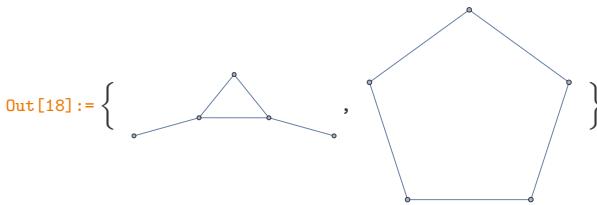
produces the following list of graphical representations of all connected graphs of order 5 and size 5:



Many other pre-generated lists of graphs are also available on [Brendan Mckay's website](#) for importation into **MATHEMATICA**. For example, the command

```
In[18]:= selfcompl = Import["https://users.cecs.anu.edu.au/~bdm/data/selfcomp5.g6"]
```

produces the following list of all self-complementary graphs of order 5:



Data can also be imported into **MATHEMATICA** in a tabular format from the aforementioned website by including the extension "Table" in the **Import** command. For example, the command

```
In[19]:= datalist = Import["https://users.cecs.anu.edu.au/~bdm/data/tree8.5.txt", "Table"]
```

produces as output the following tabular edge list representation of all trees of order 8 with diameter 5:

```
Out[19]:= {{0, 5, 0, 6, 1, 6, 1, 7, 2, 7, 3, 7, 4, 7}, {0, 5, 0, 7, 1, 6, 1, 7, 2, 6, 3, 7, 4, 7}, {0, 5, 0, 6, 1, 6, 2, 6, 3, 7, 4, 7, 5, 7}, {0, 5, 0, 7, 1, 6, 1, 7, 2, 6, 3, 6, 4, 7}, {0, 5, 1, 5, 2, 6, 2, 7, 3, 6, 4, 7, 5, 7}, {0, 4, 0, 6, 0, 7, 1, 5, 1, 7, 2, 6, 3, 7}, {0, 4, 0, 7, 1, 5, 1, 7, 2, 6, 3, 6}}
```

The tabular format above may be converted to actual edge lists by means of the commands

```
In[20]:= list2edges[edgelist_] := Map[ #[[1]] \[Rightarrow] #[[2]] &, Partition[edgelist, 2] ]  
In[21]:= edgelists = Map[list2edges, datalist]
```

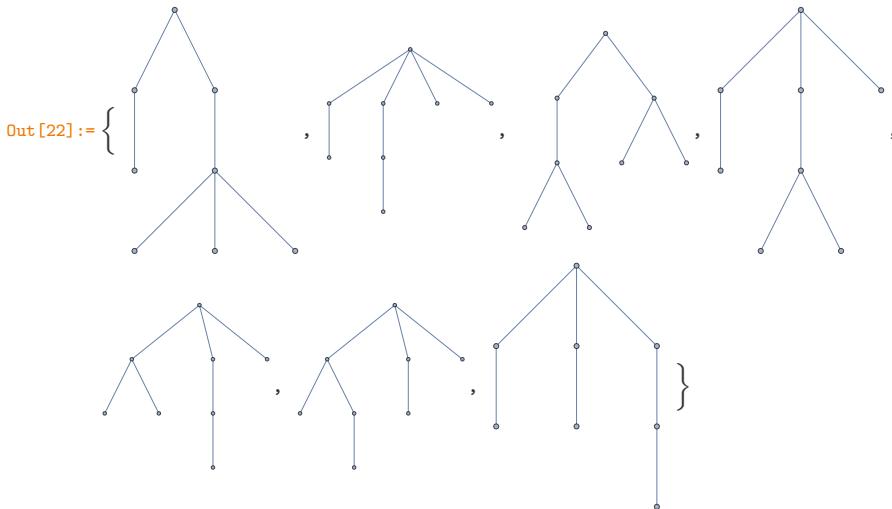
which return as output:

```
Out[21]:= {{0 \[Rightarrow] 5, 0 \[Rightarrow] 6, 1 \[Rightarrow] 6, 1 \[Rightarrow] 7, 2 \[Rightarrow] 7, 3 \[Rightarrow] 7, 4 \[Rightarrow] 7}, {0 \[Rightarrow] 5, 0 \[Rightarrow] 7, 1 \[Rightarrow] 6, 1 \[Rightarrow] 7, 2 \[Rightarrow] 6, 3 \[Rightarrow] 7, 4 \[Rightarrow] 7}, {0 \[Rightarrow] 5, 0 \[Rightarrow] 6, 1 \[Rightarrow] 6, 2 \[Rightarrow] 6, 3 \[Rightarrow] 7, 4 \[Rightarrow] 7, 5 \[Rightarrow] 7}, {0 \[Rightarrow] 5, 0 \[Rightarrow] 7, 1 \[Rightarrow] 6, 1 \[Rightarrow] 7, 2 \[Rightarrow] 6, 3 \[Rightarrow] 6, 4 \[Rightarrow] 7}, {0 \[Rightarrow] 5, 1 \[Rightarrow] 5, 2 \[Rightarrow] 6, 2 \[Rightarrow] 7, 3 \[Rightarrow] 6, 4 \[Rightarrow] 7}, {0 \[Rightarrow] 4, 0 \[Rightarrow] 6, 0 \[Rightarrow] 7, 1 \[Rightarrow] 5, 1 \[Rightarrow] 7, 2 \[Rightarrow] 6, 3 \[Rightarrow] 7}, {0 \[Rightarrow] 4, 0 \[Rightarrow] 7, 1 \[Rightarrow] 5, 1 \[Rightarrow] 7, 2 \[Rightarrow] 6, 3 \[Rightarrow] 6}, {0 \[Rightarrow] 4, 0 \[Rightarrow] 7, 1 \[Rightarrow] 5, 1 \[Rightarrow] 7, 2 \[Rightarrow] 6, 2 \[Rightarrow] 7, 3 \[Rightarrow] 6}}
```

The edge lists above may then finally be converted into a list of graphical representations by means of the command

```
In[22]:= Map[Graph, edgelists]
```

which returns as output:



- ❖ The reader should now be able to repeat Exercises 19.6, 19.11, 19.12, 19.18, 19.20(b) and 19.21(b) without a pen and paper.

Projects

This section contains four projects. In the first project, the reader is required to extend the entries of Tables 19.1 and 19.2, as well as the sequence (19.2). The second project requires drawings of distinct labelled trees, labelled rooted trees and genealogical registers of given orders, while the third project is devoted to the enumeration of the possible triangulations of labelled, convex polygons. The last project finally involves a closer look at centroidal and bicentroidal trees.

Project 19.1: Enumerating connected and disconnected graphs

The purpose of this project is to extend Tables 19.1 and 19.2 as well as the sequence (19.2).

Tasks

1. Notice, from Table 19.2, that there are 1024 labelled graphs of order 5. Extend Table 19.1 to $n = 5$. (Hint: For $m = 0, 1, 2, \dots, 10$ edges there are 1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1 labelled graphs respectively.)
2. Use your answer to Exercise 19.1 to fill in the missing proportions corresponding to $n = 5, 6, 7$ in the sequence (19.2).
3. Use the results of Theorems 19.2 and 19.3 to fill in the next two proportions corresponding to $n = 10, 11$ in the sequence (19.2).

4. Verify empirically that the sequence (19.2) together with your additions in Tasks 2 and 3 above is, in fact, strictly increasing (but bounded from above by 1).

Project 19.2: Drawing labelled trees and genealogical registers

Exhaustive lists of labelled trees, labelled rooted trees and genealogical registers are shown for small orders in Tables 19.4, 19.6 and Figure 19.5, respectively. The objective in this project is to extend these lists by one order each.

Tasks

1. Draw all 125 distinct labelled trees of order 5.
2. Draw all 64 distinct labelled, rooted trees of order 4.
3. Draw all 42 distinct genealogical registers of order 6.

Project 19.3: Convex polygon triangulation

A labelled convex polygon with $n + 1$ sides is said to be **triangulated** if $n - 1$ non-intersecting chords are drawn such that the interior of the polygon is partitioned into $n - 1$ triangular regions. Suppose there are \mathcal{C}_n distinct ways to triangulate a labelled convex polygon with $n + 1$ sides. Then, clearly, $\mathcal{C}_2 = 1$ and $\mathcal{C}_3 = 2$, as shown in Figure 19.17(a) and (b), respectively. We adopt the convention that $\mathcal{C}_1 = 1$.

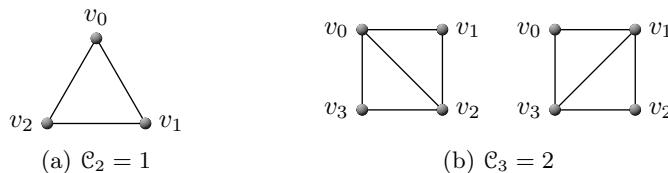


Figure 19.17: All triangulations of labelled convex polygons with (a) three or (b) four sides.

Tasks

1. There are $\mathcal{C}_4 = 5$ ways to triangulate a labelled convex polygon with five sides. One such triangulation is shown in Figure 19.18(a). Draw the other four triangulations.
2. There are $\mathcal{C}_5 = 14$ ways to triangulate a labelled convex polygon with six sides. Two such triangulations are shown in Figure 19.18(b). Draw the other twelve triangulations.
3. By now you should have recognised the first five terms $1, 1, 2, 5, 14, \dots$ of the sequence $(\mathcal{C}_n)_{n=1}^{\infty}$. Indeed, it is the well-known sequence of *Catalan numbers* in (19.11) and Table 19.7! Prove this observation, by showing that the number of distinct triangulations \mathcal{C}_n of a labelled convex polygon with $n + 1$ sides

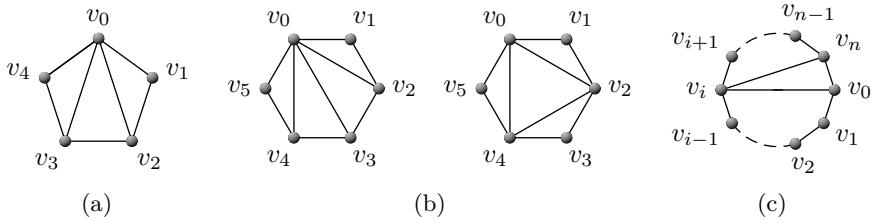


Figure 19.18: Some triangulations of labelled convex polygons with (a) five, (b) six or (c) n sides.

satisfies the recurrence relation

$$\mathcal{C}_n = \sum_{i=1}^{n-1} \mathcal{C}_i \mathcal{C}_{n-i}, \quad n = 2, 3, 4, \dots$$

in (19.5). (Hint: Suppose v_0, v_1, \dots, v_n are the vertex labels of the polygon. Consider the side v_0v_n of the polygon. This side must be an edge of some triangular region — suppose v_i is the third vertex of the triangular region in question, as shown in Figure 19.18(c).)

4. Draw all $\mathcal{C}_6 = 42$ triangulations of a labelled convex polygon with seven sides.

Project 19.4: On the enumeration of unlabelled trees

Recall that the **weight** of a vertex v of degree d in a tree T is defined as the largest of the sizes of the subtrees $T_1^{(v)}, \dots, T_d^{(v)}$ of T which meet at v , and that any vertex of minimum weight in T is called a **centroid** of T .

Tasks

1. Prove that every tree has either exactly one centroid or exactly two centroids.

Recall that a tree with one centroid is called a **centroidal tree** and that a tree with two centroids is called a **bicentroidal tree**.

2. Prove that the order of a bicentroidal tree is necessarily even.

Observe, from Table 19.12, that the order of a centroidal tree may be odd or even.

3. Indicate, in each case, the weights of the vertices of the trees in Table 19.12. Hence indicate the centroid(s) of each tree in the figure so as to verify that every tree has correctly been classified as centroidal or bicentroidal.
4. Extend Table 19.12 to $n = 8$ by drawing all eleven (centroidal) trees of order 7 as well as all twenty three trees of order 8 (of which 13 are centroidal and 10 are bicentroidal).

Further reading

- [1] NL Biggs, 1985. *Discrete Mathematics*, Oxford Science Publications Series, Clarendon Press, Oxford.

- [2] A Cayley, 1889. *A theorem on trees*, Quarterly Journal on Mathematics, **23**, pp. 376–378.
- [3] N Deo, 1974. *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, Englewood Cliffs (NJ), Chapter 10.
- [4] F Harary, 1956. *Note on the Pólya and Otter formulas for enumerating trees*, Michigan Mathematics Journal, **3**, pp. 109–112.
- [5] F Harary and EM Palmer, 1973. *Graphical Enumeration*, Addison-Wesley, Reading.
- [6] B McKay, *Combinatorial Data*, 2021, URL: <https://users.cecs.anu.edu.au/~bdm/data/>.
- [7] R Otter, 1948. *The number of trees*, Annals of Mathematics, **49**, pp. 583–599.
- [8] G Pólya, 1937. *Kombinatorische Anzahl Bestimmungen für Gruppen, Graphen, und chemische Verbindungen*, Acta Mathematica, **68**, pp. 145–254.
- [9] H Prüfer, 1918. *Neuer Beweis eines Satzes über Permutationen*, Archiv der Mathematik und Physik, **27**, pp. 742–744.
- [10] NJA Sloane, *The on-line encyclopedia of integer sequences*, URL: <http://www.research.att.com/~njas/sequences/>.
- [11] RP Stanley, 1999. *Enumerative Combinatorics*, Volume 2, Cambridge University Press, Cambridge.



The probabilistic method

Contents

20.1	Introduction	725
20.2	Ramsey theory	726
20.3	Linearity of expectation	727
20.4	Random graphs	734
	Exercises	746
	Projects	746
	Further reading	747

20.1 Introduction

In this chapter, we describe and illustrate the use of the well-known probabilistic method in graph theory. Application of the probabilistic method in graph theory is a relatively recent development, but it is a powerful proof technique for obtaining structural results in graphs. The first use of the probabilistic method to prove a graph theoretical result is generally accepted to be Erdős' proof of a lower bound on Ramsey numbers in 1947.

The idea behind the probabilistic method in graph theory is to consider all possible instances of a structure in a specified graph (such as all possible vertex-colourings of the graph, for example) and to show that there exists at least one structure having a desired property. This approach has been very successful in providing lemmas needed to prove major results.

Let Ω be a finite set (called the sample space). An element of Ω is denoted by G (think graph!). If A is a subset of Ω , we denote the complement of A in Ω by $\bar{A} = \Omega \setminus A$. We define a function \Pr mapping the subsets of Ω to the interval $[0, 1]$ which satisfies:

- (i) $\Pr(\Omega) = 1$,
- (ii) If A is a subset of Ω , then $\Pr(\bar{A}) = 1 - \Pr(A)$, and
- (iii) If A and B are disjoint subsets of Ω , then $\Pr(A \cup B) = \Pr(A) + \Pr(B)$.

The set Ω together with the function \Pr is called a **probability space**. We call Ω the **domain** or **sample space** of the probability space and \Pr we call a **probability distribution**. An **event** A in the probability space is a subset of Ω . For an event

$A \subseteq \Omega$, we define the probability of A by

$$\Pr(A) = \sum_{G \in A} \Pr(G).$$

20.2 Ramsey theory

Recall, from Chapter 17, that the **Ramsey number** $r = r(s, s)$ is the smallest integer r such that, if G is any graph of order r , then G or \bar{G} contains K_s as a subgraph. We present perhaps the first use of the probabilistic method in graph theory.

Theorem 20.1 (Erdős, 1947) *If*

$${n \choose s} 2^{1 - {s \choose 2}} < 1,$$

then $r(s, s) > n$.

Proof Consider a probability space whose elements are the red-blue edge colourings of K_n . Suppose the probabilities of colouring any edge of K_n red is $\frac{1}{2}$, independently of the choice of colour for any other edge.

Let S be a set of s vertices, and let A_S be the event that the subgraph $G[S]$ induced by S is a monochromatic red K_s or a monochromatic blue K_s . Thus, A_S is the subset of all red-blue edge colourings of K_n for which $G[S]$ is a red K_s or a blue K_s . Then

$$\Pr(A_S) = \left(\frac{1}{2}\right)^{{s \choose 2}} + \left(\frac{1}{2}\right)^{{s \choose 2}} = 2 \cdot \left(\frac{1}{2}\right)^{{s \choose 2}} = 2^{1 - {s \choose 2}}.$$

Consider the event $\cup A_S$, the union over all s -element subsets S of K_n . Since there are ${n \choose s}$ such subsets, and since the probability of a union of events is at most the sum of the probabilities of the events,

$$\Pr(\cup A_S) \leq \sum \Pr(A_S) = {n \choose s} 2^{1 - {s \choose 2}} < 1$$

by assumption. Thus, the probability that at least one of the complete subgraphs on s vertices is monochromatic is less than one. It follows that

$$\Pr(\cap \bar{A}_S) = \Pr(\cup \bar{A}_S) = 1 - \Pr(\cup A_S) > 0.$$

Thus $\cap \bar{A}_S$ is not the null event, implying that there is a point in the probability space for which $\cap \bar{A}_S$ holds. Such a point, however, is a red-blue edge colouring of K_n containing no monochromatic K_s . Hence there is at least one red-blue edge colouring of K_n containing no monochromatic K_s , and so $r(s, s) > n$. ■

As a consequence of Theorem 20.1, we have the following lower bound on the Ramsey number $r(s, s)$.

Corollary 20.2 *For every integer $s \geq 3$, $r(s, s) > \lfloor 2^{s/2} \rfloor$.*

Proof Let $n = \lfloor 2^{s/2} \rfloor$. Since $n \leq 2^{s/2}$,

$${n \choose s} \leq \frac{n^s}{s!} \leq \frac{2^{s^2/2}}{s!}.$$

Since $s \geq 3$, a simple induction argument on s shows that

$$2^{s^2/2} < \frac{1}{2}s! 2^{\binom{s}{2}}.$$

Thus,

$$\binom{n}{s} 2^{1-\binom{s}{2}} \leq \frac{2^{s^2/2}}{s!} \cdot 2^{1-\binom{s}{2}} < \frac{1}{2} 2^{\binom{s}{2}} 2^{1-\binom{s}{2}} = 1.$$

Hence, by [Theorem 20.1](#), $r(s, s) > n = \lfloor 2^{s/2} \rfloor$. ■

We remark that the bound of [Corollary 20.2](#) is trivial for $s = 3$ but false for $s = 1, 2$. With more work, we can use Stirling's formula $s! > \sqrt{2\pi s} \left(\frac{s}{e}\right)^s$, to obtain the following stronger result. We leave the proof of this result as an exercise.

Corollary 20.3 *For every integer $s \geq 3$,*

$$r(s, s) > \frac{s}{e\sqrt{2}} \cdot 2^{s/2}.$$

❖ The reader should now be able to attempt [Exercise 20.1](#).

20.3 Linearity of expectation

Given a probability space Ω , a **random variable X on Ω** is a real-valued function on Ω ; that is, $X : \Omega \mapsto \mathbb{R}$. If X is a random variable defined on a probability space $\Omega = \{v_1, \dots, v_r\}$ where $\Pr(v_i) = p_i$, then the **expected value** of X is

$$E(X) = \sum_{i=1}^r p_i X(v_i).$$

Furthermore, suppose $E(X) = t$. If $X(v_i) < t$ for all elements v_i of Ω , then

$$t = E(X) = \sum_{i=1}^r p_i X(v_i) < t \sum_{i=1}^r p_i = t \Pr(\Omega) = t \times 1 = t,$$

which is absurd. Hence, there exists an element v_i of Ω such that $X(v_i) \geq t$. Similarly, there exists an element v_j of Ω such that $X(v_j) \leq t$. This simple observation that expectation is a weighted average value proves very powerful in application of the probabilistic method as we shall see shortly.

If X_1, X_2, \dots, X_s are random variables on a probability space Ω and

$$X = c_1 X_1 + c_2 X_2 + \cdots + c_s X_s,$$

then **linearity of expectation** states that

$$E(X) = c_1 E(X_1) + c_2 E(X_2) + \cdots + c_s E(X_s).$$

This observation also proves very powerful in application of the probabilistic method since there is no requirement that the X_i should be independent in this equality.

An **indicator random variable** is a special kind of random binary variable associated with the occurrence of an event. The indicator random variable I_A associated with an event A assumes the value 1 if A occurs, or the value 0 otherwise. We decompose the random variable $X = X_1 + \dots + X_s$ into a sum of its indicator random variables, and so $E(X) = E(X_1) + \dots + E(X_s)$. Thus, if X counts how many items of a set S belongs to another set T , then an indicator random variable X_i will indicate whether item i of S belongs to T , and therefore $X = X_1 + \dots + X_{|S|}$.

In graph theoretic applications, **linearity of expectation** is usually applied when we are interested in a random variable X which counts the number of structures of a particular type in a class of graphs. For this purpose, we define an indicator random variable for each possible substructure in the graph.

To illustrate the idea of **linearity of expectation**, we present several examples in the remainder of this section.

20.3.1 Bipartite subgraphs

For a graph $G = (V, E)$ and subsets A and B of V , recall that $G[A, B]$ denotes the set of edges of G that join a vertex of A and a vertex of B . The simplified notation $[A, B]$ is used when G is clear from the context.

Theorem 20.4 *Every graph G contains a bipartite subgraph of size at least $m(G)/2$.*

Proof Let $G = (V, E)$ be a graph and consider a probability space Ω whose elements are all possible subsets of V . Let A be a (random) subset of V where a vertex is chosen to be in A with probability $\frac{1}{2}$, independently of the choice for any other vertex, and let $B = V \setminus A$. Let $X : \Omega \mapsto \mathbb{R}$ be the random variable defined by $X(A) = |[A, B]|$ for every $A \in \Omega$; that is, X assigns to A the number of edges in $[A, B]$. For each edge $e \in E$, let X_e be the indicator random variable for the edge e , and so $X_e(A) = 1$ if $e \in [A, B]$, or $X_e(A) = 0$ otherwise. Then

$$X = \sum_{e \in E} X_e.$$

For each edge $e \in E$, we have that $E(X_e) = \frac{1}{2}$ (see [Exercise 20.2](#)). Thus, by **linearity of expectation**,

$$E(X) = \sum_{e \in E} E(X_e) = \sum_{e \in E} \frac{1}{2} = \frac{m(G)}{2}.$$

Hence, $X(A^*) \geq m(G)/2$ for some vertex set $A^* \subseteq V$, and the graph induced by the edge set $[A^*, B^*]$ where $B^* = V \setminus A^*$ is the desired bipartite subgraph. ■

❖ The reader should now be able to attempt [Exercise 20.2](#).

20.3.2 Hamiltonian paths

We next harness the notion of linearity of expectation to determine a lower bound on the number of hamiltonian paths in a tournament. Recall, from [Chapter 15](#), that a **tournament** T is an orientation of the edges of a complete graph with vertex set $V(T)$. Hence, for every two vertices u and v in T , either (u, v) or (v, u) is an arc of T , but not both. The following result was first established by [Szele](#) [20].

Theorem 20.5 ([20]) *For every positive integer n , there is a tournament of order n containing at least $n!/2^{n-1}$ (directed) hamiltonian paths.*

Proof Consider a probability space Ω whose elements are the $2^{\binom{n}{2}}$ different labelled tournaments T with vertex set $V(T) = \{v_1, \dots, v_n\}$, where there is an arc from v_i to v_j or an arc from v_j to v_i with equal probability $\frac{1}{2}$ (that is, we flip a coin to determine the direction of each arc) and let these events be mutually independent. Let $X : \Omega \mapsto \mathbb{R}$ be the random variable that assigns to each tournament $T \in \Omega$ the number of (directed) hamiltonian paths in T . For each of the $n!$ permutations π of $V(T)$, let X_π be the indicator random variable for the event that π is a hamiltonian path in T (that is, $X_\pi = 1$ if π is a hamiltonian path in T , or $X_\pi = 0$ otherwise). Then

$$X = \sum_{\pi} X_\pi,$$

where the summation is taken over all $n!$ permutations π of $V(T)$. For each given permutation π of $V(T)$, we have that

$$E(X_\pi) = \left(\frac{1}{2}\right)^{n-1}$$

since the probability that each of the $n-1$ arcs corresponding to the permutation π has the desired orientation is $\frac{1}{2}$. Thus, by [linearity of expectation](#),

$$E(X) = \sum_{\pi} E(X_\pi) = \sum_{\pi} \left(\frac{1}{2}\right)^{n-1} = n! \left(\frac{1}{2}\right)^{n-1}.$$

Hence, $X(T) \geq n!/2^{n-1}$ for some tournament $T \in \Omega$. Such a tournament T has n vertices and at least $n!/2^{n-1}$ hamiltonian paths. ■

20.3.3 Independent sets

Recall, from [Section 16.3](#), that the *independence number* of G , denoted $\alpha(G)$, is the maximum cardinality of an independent set in G . In the following result, we apply the linearity of expectation to determine a lower bound on the independence number of a graph.

Theorem 20.6 *If G is a graph of order n and size $nk/2$, for some integer $k \geq 1$, then $\alpha(G) \geq n/2k$.*

Proof Let $G = (V, E)$ be a graph of order n and size $nk/2$. Consider a probability space Ω whose elements are all 2^n subsets of V . For $0 \leq p \leq 1$, let S be a (random) subset of V where a vertex is chosen to be in S with probability p , independently of the choice for any other vertex. Let $X : \Omega \mapsto \mathbb{R}$ and $Y : \Omega \mapsto \mathbb{R}$ be the random variables defined by $X(A) = |A|$ and $Y(A) = m(G[A])$, respectively, for each $A \in \Omega$. In particular, we note that $Y(A)$ is the number of edges in the subgraph induced by the set A in G . Then

$$E(X(S)) = E(|S|) = \sum_{v \in V} \Pr(v \in S) = \sum_{v \in V} p = np.$$

For each edge $e \in E$, let Y_e be the indicator random variable for the edge e , and

so for each $A \in \Omega$, we have $Y_e(A) = 1$ if $e \in E(G[A])$, or $Y_e(A) = 0$ otherwise. Therefore,

$$Y(A) = \sum_{e \in E} Y_e(A).$$

For each edge $e \in E$, we have that

$$E(Y_e(S)) = \Pr(e \in E(G[A])) = p^2,$$

and so, by [linearity of expectation](#),

$$E(Y(S)) = \sum_{e \in E} E(Y_e(S)) = |E| \cdot p^2 = \frac{1}{2} n k p^2.$$

Again, by [linearity of expectation](#), it therefore follows that

$$E(X(S) - Y(S)) = E(X(S)) - E(Y(S)) = np - \frac{1}{2} n k p^2.$$

The quantity $np - \frac{1}{2} n k p^2$ is maximised when $p = \frac{1}{k}$. Setting $p = \frac{1}{k}$, we have that

$$E(X(S) - Y(S)) = np - \frac{1}{2} n k p^2 = n/2k.$$

Since expectation is a weighted average value, there exists a subset S of vertices in G such that $|S| - |E(G[S])| \geq E(X(S) - Y(S)) = n/2k$. Thus, the number of vertices in S less the number of edges in the subgraph $G[S]$ induced by S is at least $n/2k$. For each edge e in $G[S]$, we select one of its endpoints v_e , and we let

$$S' = \bigcup_{e \in E(G[S])} \{v_e\}$$

and $I = S \setminus S'$. By construction, the set I is an independent set. Furthermore, since we delete at most $m(G[S])$ vertices from S , our choice of the set S implies that $|I| \geq n/2k$. Consequently, $\alpha(G) \geq |I| \geq n/2k$. ■

We next present a well-known result established independently by [Caro \[4\]](#) and [Wei \[23\]](#) that presents a lower bound on the independence number in terms of the degree sequence of the graph.

Theorem 20.7 (Caro-Wei Theorem) *For every graph G ,*

$$\alpha(G) \geq \sum_{v \in V(G)} \frac{1}{d_G(v) + 1}.$$

Proof Let $G = (V, E)$ be a graph of order n and consider a probability space Ω whose elements are the $n!$ distinct permutations $\pi : V(G) \mapsto [n]$ of $V(G)$, where each permutation is chosen with equal probability $\frac{1}{n!}$. Each permutation π corresponds to an ordering of the vertices of V from least to greatest. We say that a vertex $v \in V$ is a π -least vertex with respect to a permutation π if $\pi(v) < \pi(w)$ for all neighbours w of v in G . For each $A \in \Omega$, let π_A be the permutation associated with A and let I_A be the set of all π_A -least vertices. Let $X : \Omega \mapsto \mathbb{R}$ be the random variable defined by $X(A) = |I_A|$. For each vertex $v \in V$, let X_v be the indicator random variable for the vertex v , and so for every $A \in \Omega$, we have $X_v(A) = 1$ if $v \in I_A$, or $X_v(A) = 0$ otherwise. Then

$$X(A) = \sum_{v \in V} X_v(A).$$

For each vertex $v \in V$, we denote $d_G(v)$ simply by $d(v)$. We now consider an arbitrary vertex $v \in V$. The vertex $v \in I_A$ if and only if it is the least element among v and all $d(v)$ neighbours of v . Each vertex in $N[v]$ has an equal chance of being first in the ordering. Thus, the probability that the vertex v is the least element is $1/(d(v) + 1)$. Hence,

$$E(X_v(A)) = \Pr(v \in I_A) = \frac{1}{d(v) + 1}.$$

By linearity of expectation,

$$E(X(A)) = \sum_{v \in V} E(X_v(A)) = \sum_{v \in V} \frac{1}{d(v) + 1}.$$

Since expectation is an average value, there exists a permutation A of the vertices in G such that $|I_A| = X(A) \geq \sum_{v \in V} 1/(d(v) + 1)$. It remains to be shown that the set I_A is independent. Suppose, to the contrary, that there is an edge uv in $G[I_A]$. Since $u \in I_A$ and v is a neighbour of u , we have that $\pi_A(u) < \pi_A(v)$. Analogously, since $v \in I_A$ and u is a neighbour of v , we have that $\pi_A(v) < \pi_A(u)$. This is clearly a contradiction. Hence, I_A is an independent set in G , implying that

$$\alpha(G) \geq |I_A| = X(A) \geq \sum_{v \in V} \frac{1}{d(v) + 1}. \quad \blacksquare$$

Recall that a **k -clique** is a complete subgraph on k vertices of a given graph and that in Chapter 18 we proved Turán's Theorem (see Theorem 18.6) which states that if G is a graph of order n and size m that has no $(r+1)$ -clique, then $m \leq \lfloor \frac{1}{2} \left(\frac{r-1}{r} \right) n^2 \rfloor$, with equality if and only if $G = T_{n,r}$, where $T_{n,r}$ is the Turán graph defined in Section 18.3. As a consequence of the Caro-Wei Theorem (Theorem 20.7), we have the following lower bound on the independence number of a graph. Recall, for the proof of this bound, the Cauchy-Schwarz inequality which states that

$$\left(\sum_{i=1}^n x_i^2 \right) \left(\sum_{i=1}^n y_i^2 \right) \geq \left(\sum_{i=1}^n x_i y_i \right)^2 \quad (20.1)$$

for any real numbers x_i and y_i , $i \in [n]$.

Theorem 20.8 ([22]) *If G is a graph of order n and size $nk/2$, then $\alpha(G) \geq n/(k+1)$.*

Proof Let G be a graph of order n and size $m = nk/2$, and let d_1, \dots, d_n denote the degree sequence of G . Applying inequality (20.1) with $x_i = (d_i + 1)^{1/2}$ and $y_i = 1/x_i$ for $i \in [n]$, we have that

$$(2m + n) \left(\sum_{i=1}^n \frac{1}{d_i + 1} \right) = \left(\sum_{i=1}^n (d_i + 1) \right) \left(\sum_{i=1}^n \frac{1}{d_i + 1} \right) \geq \left(\sum_{i=1}^n 1 \right)^2 = n^2.$$

Hence, it follows from Theorem 20.7 that

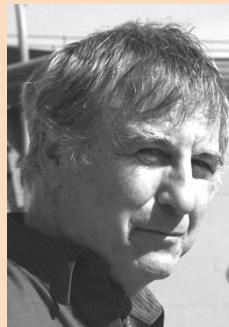
$$\alpha(G) \geq \sum_{i=1}^n \frac{1}{d_i + 1} \geq \frac{n^2}{2m + n} = \frac{n}{k+1},$$

as desired. ■

20.3.4 Dominating sets

Recall, from [Chapter 16](#), that a **dominating set** of a graph G is a subset D of vertices of G such that every vertex not in D has at least one neighbour in D . In the following result, we apply the linearity of expectation to determine an upper bound on the **domination number**, the smallest cardinality of a dominating set of a graph in terms of its order and minimum degree. The proof employs the so-called *deletion method*, also called the *alteration method* or the *two-step method*. The idea is that, given a graph G , we form a random set of vertices where each vertex is chosen with a certain probability. Since the random choice of our set is, however, not necessarily a dominating set, the set has to be altered by adding to it a set of vertices not yet dominated. The resulting set is a dominating set. By choosing carefully the probability that each vertex is included in the initial random set, we obtain an upper bound on the domination number. The formal details are presented in the proof of the following result which can be found in [Alon and Spencer \[2\]](#), [Arnaudov \[3\]](#), [Haynes et al. \[13\]](#), [Payan \[18\]](#), and elsewhere.

Joel Spencer was born in the United States of America on 20 April 1946. He obtained a bachelor of science degree at the Massachusetts Institute of Technology in 1965 and received a doctorate in mathematics from Harvard University in 1970 under the supervision of Andrew Gleason. He later became a Silver Professor of Mathematics and Computer Science at the Courant Institute of New York University. In 1984, Spencer received the Lester R Ford Award and in 2012 became a fellow of the American Mathematical Society. He is the co-founder and co-editor of the journal *Random Structures and Algorithms* and is also a Sloane Foundation Fellow. He has published extensively and is the co-author of *Ramsey Theory* (a second edition) and *The Probabilistic Method* (a fourth edition), both published by Wiley. He is known for his significant contributions to discrete mathematics and the theory of computing, particularly in random graphs and networks, Ramsey theory, logic, and randomised algorithms.



Biographic note 103: Joel Spencer (1946–present)

Theorem 20.9 *Let G be a graph of order n with minimum degree $\delta \geq 2$. Then*

$$\gamma(G) \leq \left(\frac{1 + \ln(\delta + 1)}{\delta + 1} \right) n.$$

Proof Let $G = (V, E)$ be a graph of order n and consider a probability space Ω whose elements are all 2^n subsets of V . For $0 \leq p \leq 1$, let S be a (random) subset of V where a vertex is chosen to be a member of S with probability p , independently of the choice for any other vertex. Let $X : \Omega \mapsto \mathbb{R}$ be the random

variable defined by $X(A) = |A|$ for each $A \in \Omega$. Then

$$E(X(S)) = E(|S|) = \sum_{v \in V} \Pr(v \in S) = \sum_{v \in V} p = np.$$

For each vertex $v \in V$, let X_v be the indicator random variable for the vertex v , and so for each $A \in \Omega$, we have $X_v(A) = 1$ if $v \in A$, or $X_v(A) = 0$ otherwise. Therefore,

$$X(A) = \sum_{v \in V} X_v(A).$$

For the random set $S \in \Omega$, let $T = S_T$ be the set of vertices of G not in S that have no neighbour in S ; that is, $T = V(G) \setminus N[S]$. The resulting set $S \cup T$ is a dominating set of G . For each vertex $v \in V$, we have that

$$E(X_v(T)) = \Pr(v \text{ and all its neighbours are not in } S) = (1 - p)^{d_G(v)+1}.$$

Recall that $0 \leq 1 - p \leq 1$ and that $d_G(v) \geq \delta$ for each vertex $v \in V$. Using the important inequality $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$, we therefore have that

$$E(X_v(T)) = (1 - p)^{d_G(v)+1} \leq (1 - p)^{\delta+1} \leq e^{-p(\delta+1)}.$$

Thus, by [linearity of expectation](#),

$$E(X(T)) = E(|T|) = \sum_{v \in V} E(X_v(T)) \leq ne^{-p(\delta+1)}.$$

As observed earlier, $E(X(S)) = E(|S|) = np$. Hence, by [linearity of expectation](#),

$$E(X(S) \cup X(T)) = E(X(S)) + E(X(T)) \leq np + ne^{-p(\delta+1)}.$$

The quantity $np + ne^{-p(\delta+1)}$ is minimised when $p = \ln(\delta+1)/(\delta+1)$. Setting $p = \ln(\delta+1)/(\delta+1)$, we have that

$$E(X(S) \cup X(T)) \leq np + ne^{-p(\delta+1)} = \left(\frac{1 + \ln(\delta+1)}{\delta+1} \right) n.$$

Since expectation is a weighted average, $|S| + |T| = X(S) \cup X(T) \leq (1 + \ln(\delta+1))/(\delta+1) n$ for some event $S \in \Omega$. For such an event S , it follows that $S \cup T$ is a dominating set in G of cardinality at most the claimed bound. ■

We remark that several of the details in the probabilistic proofs presented in this section are often omitted in the literature to make the proof easier to read. As an illustration, we present next a shortened proof of [Theorem 20.9](#) omitting some of the details presented in the proof above.

Noga Alon was born in Israel on 17 February 1956. He is a Baumritter Professor of Mathematics and Computer Science at Tel Aviv University, Israel. He graduated from the Hebrew Reali School in 1974, received a doctorate in mathematics from the Hebrew University of Jerusalem in 1983 and has held visiting positions in various research institutes, including the Massachusetts Institute of Technology, the Institute for Advanced Study at Princeton, IBM's Almaden Research Center, Bell Laboratories, Bellcore and Microsoft Research. He serves on the editorial boards of more than a dozen international journals and is a prolific researcher, having published well over 500 papers. His research interests are mainly in combinatorics, graph theory and their applications in theoretical computer science. Among his many contributions is the development and application of algebraic and probabilistic methods in discrete mathematics. He is the recipient of many prestigious prizes and awards, including the Erdős Prize in 1989, the Feher Prize in 1991, the Pólya Prize in 2000, the Bruno Memorial Award in 2001, the Landau Prize in 2005, the Gödel Prize in 2005, the Israel Prize in 2008, the EMET Prize in 2011, and the Dijkstra Prize in 2016.



Biographic note 104: Noga Alon (1956–present)

Alternative proof of Theorem 20.9 Let $p = \ln(\delta + 1)/(\delta + 1)$ and form a random subset $S \subseteq V(G)$ where a vertex is chosen to be a member of S with probability p , independently of the choice for any other vertex. Let T be the set of vertices of G not in S that have no neighbour in S . The set $S \cup T$ is a dominating set of G . The expected value of $|S|$ is $E(|S|) = np$. The random variable $|T|$ can furthermore be written as the sum of n indicator random variables X_v when $v \in V(G)$ and where $X_v(T) = 1$ if $v \in T$, or $X_v(T) = 0$ otherwise. For each vertex $v \in V(G)$, we have that $E(X_v(T)) = \Pr(v \text{ and all its neighbours are not in } S) = (1 - p)^{\deg(v)+1} \leq (1 - p)^{\delta+1}$ since the minimum degree is at least δ and $0 \leq 1 - p \leq 1$. Since $1 - p \leq e^{-p}$ for all $p \in \mathbb{R}$, it therefore follows that $E(X_v(T)) \leq e^{-p(\delta+1)}$ for each vertex v . By linearity of expectation, $E(|S|) + E(|T|) \leq np + \sum_{v \in V} E(X_v(T)) \leq np + ne^{-p(\delta+1)} = n(1 + \ln(\delta + 1))/(\delta + 1)$. ■

20.4 Random graphs

In this section, we briefly consider the theory of random graphs which was introduced by Erdős and Rényi [10] in 1960. We adopt their model $\mathcal{G}(n, p)$ of a random graph, where n denotes the number of vertices and where each edge is chosen independently with probability p . More formally, for a positive integer n and a real number p with $p \in (0, 1)$, we denote by $\mathcal{G}(n, p)$ the probability space whose elements G are all possible graphs of order n with vertex set $V_n = \{v_1, \dots, v_n\}$ and in which each edge is chosen to be in G with probability p , independently of the choice for any other edge. We call an element $G \in \mathcal{G}(n, p)$ a **random graph**. In this section, we require an extremely useful result called [Markov's inequality](#).

Lemma 20.10 (Markov's Inequality) *Let X be a random variable assuming only nonnegative values. Then, for all positive real numbers t ,*

$$\Pr(X \geq t) \leq \frac{E(X)}{t}.$$

Proof Let X be a random variable on a probability space Ω , and let $A = \{G \in \Omega \mid X(G) \geq t\}$. Then

$$\begin{aligned} E(X) &= \sum_{G \in \Omega} X(G) \Pr(G) \geq \sum_{G \in A} X(G) \Pr(G) \geq t \sum_{G \in A} \Pr(G) \\ &= t \Pr(A) = t \Pr(X \geq t), \end{aligned}$$

and so $\Pr(X \geq t) \leq E(X)/t$. ■

Setting $t = 1$ in Lemma 20.10, we have the following immediate consequence which is used frequently in graph theoretic applications.

Corollary 20.11 *Let X be a random variable assuming only nonnegative integer values. Then*

$$\Pr(X \geq 1) \leq E(X)$$

or, equivalently,

$$\Pr(X = 0) = 1 - \Pr(X \geq 1) \geq 1 - E(X).$$

In particular, if $E(X) \rightarrow 0$, then $\Pr(X = 0) \rightarrow 1$.

A **graph property** or **graph invariant** is a property of graphs that is preserved under all possible isomorphisms of a graph and, therefore, depends only on the abstract structure of graphs as opposed to the way the graph is drawn or represented. Some examples of graph properties include connectedness, planarity, diameter 2, girth 6, and chromatic number 4. Let \mathcal{P} be a graph property and let $G \in \mathcal{G}(n, p)$ be a random graph, where $0 < p < 1$ and where p may depend on n . If

$$\lim_{n \rightarrow \infty} \Pr(G \in \mathcal{P}) = 1,$$

then we say that **almost every** random graph $G \in \mathcal{G}(n, p)$ has property \mathcal{P} with respect to p , while if

$$\lim_{n \rightarrow \infty} \Pr(G \in \mathcal{P}) = 0,$$

then we say that **almost no** random graph G has property \mathcal{P} with respect to p .

20.4.1 Colourings

We present here a beautiful and surprising result of Erdős [8] that shows that, by choosing n sufficiently large and by choosing the probability p carefully, there exists a random graph in $\mathcal{G}(n, p)$ that contains an induced subgraph with arbitrarily large girth and arbitrarily large chromatic number, where recall that the **girth** of a graph is the length of its shortest cycle. We first prove the following lemma.

Lemma 20.12 Let $p \in (0, 1)$ be a given constant and let n and t be integers with $n \geq t \geq 2$. Then the probability that a random graph $G \in \mathcal{G}(n, p)$ has independence number at least t is

$$\Pr(\alpha(G) \geq t) \leq \binom{n}{t} (1-p)^{\binom{t}{2}}.$$

Proof Let $G \in \mathcal{G}(n, p)$ have vertex set V_n . The probability that a fixed t -element subset of V_n is independent in G is

$$(1-p)^{\binom{t}{2}}.$$

For each t -element subset T of V_n , let A_T denote the event that T is independent. Then, since there are $\binom{n}{t}$ such sets T , it follows that the probability that G has an independent set of t vertices is

$$\Pr(\alpha(G) \geq t) = \Pr(\cup A_T) \leq \sum \Pr(A_T) = \binom{n}{t} (1-p)^{\binom{t}{2}},$$

where the union and the sum are over all t -element subsets T of V_n . ■

We are now in a position to prove Erdős' result.

Theorem 20.13 ([8]) For all positive integers k and ℓ , there exists a graph G with chromatic number larger than k and girth larger than ℓ .

Proof If $k = 1$, then we can take G to be an even cycle of length greater than ℓ . Hence, we may assume that $k \geq 2$. Initially, let n be a fixed positive integer. Fix θ with $0 < \theta < \frac{1}{\ell}$ and define $p = n^{\theta-1}$. Then $0 < p < 1$. Consider the probability space $\mathcal{G}(n, p)$ of random graphs. For $i = 3, \dots, \ell$, let $X_i : \mathcal{G}(n, p) \mapsto \mathbb{N} \cup \{0\}$ be the random variable that assigns to each $G \in \mathcal{G}(n, p)$ the number of i -cycles in G . For $i \geq 3$ an integer, let

$$(n)_i := n(n-1)(n-2) \cdots (n-i+1).$$

For $i = 3, \dots, \ell$, we have (see Exercise 20.3) that

$$E(X_i) = \frac{(n)_i}{2i} p^i.$$

Let $X : \mathcal{G}(n, p) \mapsto \mathbb{N} \cup \{0\}$ be the random variable that assigns to each $G \in \mathcal{G}(n, p)$ the number of cycles of length at most ℓ in G . Then

$$X = \sum_{i=3}^{\ell} X_i,$$

and so, by linearity of expectation,

$$E(X) = \sum_{i=3}^{\ell} E(X_i) = \sum_{i=3}^{\ell} \frac{(n)_i}{2i} p^i \leq \frac{1}{2} \sum_{i=3}^{\ell} \frac{n^i}{i} p^i \leq \frac{1}{2} \sum_{i=3}^{\ell} \frac{n^i}{i} n^{(\theta-1)i} = \frac{1}{2} \sum_{i=3}^{\ell} \frac{n^{\theta i}}{i}.$$

By our choice of θ , we have that $\theta\ell = 1 - \epsilon$ for some real number ϵ with $0 < \epsilon < 1$, and so

$$E(X) \leq \frac{1}{2} \sum_{i=3}^{\ell} \frac{n^{\theta i}}{i} \leq \frac{1}{2} \sum_{i=3}^{\ell} \frac{n^{\theta\ell}}{i} = \frac{1}{2} \sum_{i=3}^{\ell} \frac{n^{1-\epsilon}}{i} \leq \frac{n}{2} \cdot \frac{\ell-2}{n^{\epsilon}}.$$

Thus, by Lemma 20.10,

$$\Pr\left(X \geq \frac{n}{2}\right) \leq \frac{E(X)}{n/2} \leq \frac{\ell - 2}{n^\epsilon}.$$

This implies that

$$\lim_{n \rightarrow \infty} \Pr\left(X \geq \frac{n}{2}\right) = 0.$$

Thus, for n sufficiently large,

$$\Pr\left(X \geq \frac{n}{2}\right) < \frac{1}{2}.$$

Next we calculate the probability that a random graph G in $\mathcal{G}(n, p)$ has reasonably large independence number; notice that having a small vertex independence number $\alpha(G)$, implies that the graph has a large vertex chromatic number, since $\chi(G) \geq n(G)/\alpha(G)$. Using Lemma 20.12, one can show (see Exercise 20.4) that for n sufficiently large,

$$\Pr\left(\alpha(G) \geq \frac{n}{2k}\right) < \frac{1}{2}.$$

Let n be so large that both

$$\Pr\left(X \geq \frac{n}{2}\right) < \frac{1}{2} \quad \text{and} \quad \Pr\left(\alpha(G) \geq \frac{n}{2k}\right) < \frac{1}{2}$$

hold. Hence there exists a graph $G \in \mathcal{G}(n, p)$ having fewer than $\frac{n}{2}$ cycles of length at most ℓ and having vertex independence number less than $n/2k$. Let H be the subgraph obtained from G by deleting one vertex from each cycle of length at most ℓ . Then, $|V(H)| \geq \frac{n}{2}$ and every cycle in H has length exceeding ℓ , and so $g(H) > \ell$. Since H is an induced subgraph of G , $\alpha(H) \leq \alpha(G) < n/2k$, and so

$$\chi(H) \geq \frac{|V(H)|}{\alpha(H)} \geq \frac{n/2}{\alpha(G)} > k. \quad \blacksquare$$

Whereas it is natural to expect that a large value for the chromatic number of a graph is the result of a large clique being present in the graph, the significance of Theorem 20.13 is that it shows that this expectation is false in general. Because the girth of a complete graph is three, the theorem shows that a graph can have a large girth without containing a clique of large order.

We determine next a lower bound on the chromatic number of almost every random graph.

Theorem 20.14 *Let $\epsilon > 0$ be given, let $p \in (0, 1)$ be a constant, and set $b = 1/(1-p)$. Then almost every graph $G \in \mathcal{G}(n, p)$ has chromatic number satisfying*

$$\chi(G) > \left(\frac{1}{2 + \epsilon}\right) \frac{n}{\log_b n}.$$

Proof Let t be an integer satisfying $2 \leq t \leq n$ and recall the logarithmic rules $a^{\log_a b} = b$, $\log a^b = b \log a$ and $\log_a b = (\log b)/(\log a)$. We note that for $x \in (0, 1)$ a constant, we can write

$$n^t = x^{\log_x n^t} = x^{t \log_x n}.$$

Additionally, if we let $x = 1 - p$, then $\frac{1}{x} = 1/(1 - p) = b$ and we can write

$$n^t = (1 - p)^{-t \log_b n}.$$

By Lemma 20.12, we therefore have

$$\begin{aligned} \Pr(\alpha(G) \geq t) &\leq \binom{n}{t} (1 - p)^{\binom{t}{2}} \leq n^t (1 - p)^{\binom{t}{2}} \\ &= (1 - p)^{-t \log_b n} (1 - p)^{\frac{1}{2}t(t-1)} \\ &= (1 - p)^{\frac{t}{2}(-2(\log_b n) + t - 1)}. \end{aligned}$$

Setting $t = (2 + \epsilon) \log_b n$, we note that

$$\frac{t}{2}(-2(\log_b n) + t - 1) = \frac{t}{2}(\epsilon(\log_b n) - 1) \rightarrow \infty$$

as $n \rightarrow \infty$ (because $\epsilon > 0$ and $\log_b n \rightarrow \infty$). Hence, since $0 < 1 - p < 1$, we have that

$$\lim_{n \rightarrow \infty} \Pr(\alpha(G) \geq t) \rightarrow 0,$$

implying that almost every graph $G \in \mathcal{G}(n, p)$ has independence number less than t . This, in turn, implies that almost every graph $G \in \mathcal{G}(n, p)$ has chromatic number

$$\chi(G) \geq \frac{n}{\alpha(G)} > \frac{n}{t} = \left(\frac{1}{2 + \epsilon}\right) \frac{n}{\log_b n},$$

as claimed. ■

❖ The reader should now be able to attempt Exercises 20.3 and 20.4.

20.4.2 Lovász Local Lemma

In this section, we discuss a fundamental result in probabilistic methods, known as the **Lovász Local Lemma**. Consider n events A_1, \dots, A_n in a probability space. If these events are mutually independent and $\Pr(A_i) < 1$ for all $i \in [n]$, then the probability that no event occurs is

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) = \prod_{i=1}^n \Pr(\overline{A_i}) = \prod_{i=1}^n (1 - \Pr(A_i)) > 0;$$

that is, with positive probability none of these events occurs. We state this observation formally as follows.

Observation 20.15 *If A_1, \dots, A_n are mutually independent events in a probability space and $\Pr(A_i) < 1$ for all $i \in [n]$, then*

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) > 0.$$

What happens, however, if there is some dependency among the events A_1, \dots, A_n ? In this case, we note that

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) = \Pr\left(\overline{\bigcup_{i=1}^n A_i}\right) = 1 - \Pr\left(\bigcup_{i=1}^n A_i\right) \geq 1 - \sum_{i=1}^n \Pr(A_i).$$

Thus if $\sum_{i=1}^n \Pr(A_i) < 1$, then with positive probability none of these events occurs. We state this observation formally as follows.

Observation 20.16 *If $\sum_{i=1}^n \Pr(A_i) < 1$ for events A_1, \dots, A_n in a probability space, then*

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) > 0.$$

The probabilistic method discussed and illustrated so far in this chapter relies heavily on the independence of the events in a probability space. If there is some dependency among the events, however, then we invoke a profoundly important tool for handling the case when dependencies occur, called the **Lovász Local Lemma**. This lemma was first proved by Erdős and Lovász [9] in 1973.

Before we present the lemma, we introduce some additional notation. Consider n events A_1, \dots, A_n in a probability space and a graph G of order n with vertex set $[n]$, where the edge ij is included in G if the events A_i and A_j are dependent. Thus, for any $i \in [n]$, the event A_i is mutually independent of all events A_j for which ij is not an edge of G . The resulting graph G is called the *dependency graph* for the events A_1, \dots, A_n .

Recall that if A, B and C are events, then the conditional probability $\Pr(A | B)$ of A , given B , and the conditional probability $\Pr(A | B \cap C)$ of A , given $B \cap C$, are

$$\Pr(A | B) = \frac{\Pr(A \cap B)}{\Pr(B)} \quad \text{and} \quad \Pr(A | B \cap C) = \frac{\Pr(A \cap B \cap C)}{\Pr(B \cap C)},$$

respectively, where we assume here that $\Pr(B) > 0$ and $\Pr(B \cap C) > 0$. In particular,

$$\Pr(A \cap B | C) = \Pr(A | B \cap C) \Pr(B | C).$$

More generally, if A_1, \dots, A_k and B are events, then

$$\Pr\left(\bigcap_{i=1}^k A_i \mid B\right) = \prod_{i=1}^k \Pr\left(A_i \mid \left(\bigcap_{j=i+1}^k A_j\right) \cap B\right). \quad (20.2)$$

We are now in a position to state the so-called **symmetric case of the Lovász Local Lemma**.

Lemma 20.17 (Lovász Local Lemma: Symmetric case, version I) *Let A_1, \dots, A_n be events in a probability space with a dependency graph G of maximum degree Δ and*

$$\Pr(A_i) \leq p < 1 \quad \text{and} \quad 4p\Delta < 1$$

for all $i \in [n]$. Then

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) > 0.$$

Proof If $\Delta = 0$, then the events A_1, \dots, A_n are mutually independent, and the desired result follows from [Observation 20.16](#). Hence, we may assume that $\Delta \geq 1$. We first prove, by induction on s , that for any subset $S \subset [n]$ with $|S| = s < n$, and $i \notin S$,

$$\Pr\left(A_i \mid \bigcap_{j \in S} \overline{A_j}\right) \leq 2p. \quad (20.3)$$

If $s = 0$, then $S = \emptyset$, and (20.3) follows immediately from our supposition that $\Pr(A_i) \leq p$. This establishes the base case. Let $s \geq 1$ with $s < n$, and assume, for any subset $S' \subset [n]$ with $|S'| = s' < s$ and $i \notin S'$, that $\Pr(A_i \mid \bigcap_{j \in S'} \overline{A_j}) \leq 2p$. Let $S_1 = \{j \in S \mid ij \in E(G)\}$, and let $S_2 = S \setminus S_1$. From Bayes' Theorem on conditional probability it follows that

$$\Pr\left(A_i \mid \bigcap_{j \in S} \overline{A_j}\right) = \frac{\Pr\left(A_i \cap \left(\bigcap_{j \in S_1} \overline{A_j}\right) \mid \bigcap_{j \in S_2} \overline{A_j}\right)}{\Pr\left(\bigcap_{j \in S_1} \overline{A_j} \mid \bigcap_{k \in S_2} \overline{A_k}\right)}. \quad (20.4)$$

To bound the numerator of the term in (20.4), we note that A_i is mutually independent of the events A_j for $j \in S_2$, implying that

$$\Pr\left(A_i \cap \left(\bigcap_{j \in S_1} \overline{A_j}\right) \mid \bigcap_{j \in S_2} \overline{A_j}\right) \leq \Pr\left(A_i \mid \bigcap_{k \in S_2} \overline{A_k}\right) = \Pr(A_i) \leq p. \quad (20.5)$$

We next bound the denominator of the term in (20.4). If $S_1 = \emptyset$, then the denominator is 1, and (20.3) follows readily. We may therefore assume that $S_1 \neq \emptyset$. Since $|S_1| \leq \Delta$, $|S_2| = |S| - |S_1| < |S|$ and $4p\Delta < 1$, we have

$$\begin{aligned} \Pr\left(\bigcap_{j \in S_1} \overline{A_j} \mid \bigcap_{k \in S_2} \overline{A_k}\right) &= 1 - \Pr\left(\bigcup_{j \in S_1} A_j \mid \bigcap_{k \in S_2} \overline{A_k}\right) \\ &\geq 1 - \sum_{j \in S_1} \Pr\left(A_j \mid \bigcap_{k \in S_2} \overline{A_k}\right) \\ &\geq 1 - \sum_{j \in S_1} 2p \quad (\text{induction}) \\ &= 1 - 2p|S_1| \geq 1 - 2p\Delta \geq \frac{1}{2}. \end{aligned} \quad (20.6)$$

Thus, substituting (20.5) and (20.6) into (20.4), we have the quotient

$$\Pr\left(A_i \mid \bigcap_{j \in S} \overline{A_j}\right) \leq \frac{p}{1/2} = 2p,$$

which completes the proof of (20.3) by induction. Since $\Delta \geq 1$ and $4p\Delta < 1$, we observe that $p < \frac{1}{4\Delta} \leq \frac{1}{4} < \frac{1}{2}$, and so $1 - 2p > 0$. Using (20.3), it therefore follows that

$$\begin{aligned} \Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) &= \prod_{i=1}^n \Pr\left(\overline{A_i} \mid \bigcap_{j=1}^{i-1} \overline{A_j}\right) \\ &= \prod_{i=1}^n \left(1 - \Pr\left(A_i \mid \bigcap_{j=1}^{i-1} \overline{A_j}\right)\right) \\ &\geq \prod_{i=1}^n (1 - 2p) \quad \text{by (20.3)} \\ &> 0. \end{aligned}$$
■

In many applications of the probabilistic method, we wish to determine whether a given graph contains some particular substructure. In a graph theoretic sense, “bad events” are those events that guarantee the nonexistence of our desired substructure, while “good events” are those that guarantee existence of the substructure. We next apply [Lemma 20.17](#) to determine a lower bound on the Ramsey number $r(s, s)$.

Theorem 20.18 *If*

$$4 \binom{s}{2} \binom{n}{s-2} 2^{1-\binom{s}{2}} < 1,$$

then $r(s, s) > n$.

Proof Consider a probability space whose elements are the red-blue edge colourings of K_n . We define the probabilities by colouring each edge red with probability $1/2$, independently of the choice for any other edge. As in the proof of [Theorem 20.1](#), for each subset S of s vertices in K_n , let A_S be the event that the subgraph induced by S is a monochromatic red K_s or a monochromatic blue K_s . Then

$$\Pr(A_S) = 2^{1-\binom{s}{2}}.$$

Let G be the dependency graph of order N , where $N = \binom{n}{s}$ corresponds to the total number of distinct subsets of s vertices in K_n and where we add an edge in G between S and T if the corresponding sets S and T in K_n intersect in at least two vertices (and therefore induce subgraphs that share at least one edge). Thus, the event A_S is mutually independent of all events A_T where ST is not an edge of G (that is, $|S \cap T| \leq 1$). Given any two vertices in S , the number of subsets of s vertices in K_n that contain these two vertices is $\binom{n-2}{s-2}$. Thus, the maximum degree of the dependency graph G is

$$\Delta(G) \leq \binom{s}{2} \binom{n-2}{s-2} < \binom{s}{2} \binom{n}{s-2}.$$

We now let

$$p = 2^{1-\binom{s}{2}} \quad \text{and} \quad \Delta = \binom{s}{2} \binom{n}{s-2}.$$

We note that $\Pr(A_S) = p$ and, by supposition, $4p\Delta < 1$. Thus, if \mathcal{S} denotes the set of all s -element subsets of vertices of K_n , then applying the [Lovász Local Lemma](#) ([Lemma 20.17](#)) we deduce that

$$\Pr\left(\bigcap_{S \in \mathcal{S}} \overline{A_S}\right) > 0.$$

Consequently, there is a point in the probability space for which $\bigcap_{S \in \mathcal{S}} \overline{A_S}$ is not the null event. Such a point is a red-blue edge colouring of K_n containing no monochromatic K_s . Therefore, $r(s, s) > n$. ■

Using [Stirling’s formula](#) and some computation, we obtain the following result which improves the lower bound in [Corollary 20.2](#) on the Ramsey number $r(s, s)$ by a factor of 2.

Corollary 20.19 For every integer $s \geq 3$,

$$r(s, s) > \frac{\sqrt{2}}{e} s 2^{s/2} (1 + \mathcal{O}(1)).$$

We prove next the general case of the Lovász Local Lemma.

Lemma 20.20 (Lovász Local Lemma: General case) Let A_1, \dots, A_n be events in a probability space with dependency graph G . If there exist real numbers x_1, \dots, x_n such that $x_i \in [0, 1)$ and

$$\Pr(A_i) \leq x_i \prod_{ij \in E(G)} (1 - x_j)$$

for all $i \in [n]$, then

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) \geq \prod_{i=1}^n (1 - x_i) > 0.$$

Proof We first prove, by induction on s , that for any subset $S \subset [n]$ with $|S| = s < n$, and $i \notin S$,

$$\Pr\left(A_i \mid \bigcap_{j \in S} \overline{A_j}\right) \leq x_i. \quad (20.7)$$

If $s = 0$, then $S = \emptyset$, and (20.7) follows immediately from our supposition that $\Pr(A_i) \leq x_i \prod_{ij \in E(G)} (1 - x_j) \leq x_i$, noting that $1 - x_j \leq 1$. This establishes the base case. Let $s \geq 1$ with $s < n$, and assume, for any subset $S' \subset [n]$ with $|S'| = s' < s$ and $i \notin S'$, that $\Pr(A_i \mid \bigcap_{j \in S'} \overline{A_j}) \leq x_i$. Let $S_1 = \{j \in S \mid ij \in E(G)\}$, and let $S_2 = S \setminus S_1$. From Bayes' Theorem on conditional probability, we note that

$$\Pr\left(A_i \mid \bigcap_{j \in S} \overline{A_j}\right) = \frac{\Pr\left(A_i \cap \left(\bigcap_{j \in S_1} \overline{A_j}\right) \mid \bigcap_{j \in S_2} \overline{A_j}\right)}{\Pr\left(\bigcap_{j \in S_1} \overline{A_j} \mid \bigcap_{k \in S_2} \overline{A_k}\right)}. \quad (20.8)$$

To bound the numerator of the term in (20.8), we note that A_i is mutually independent of the events A_j for $j \in S_2$, implying that

$$\begin{aligned} \Pr\left(A_i \cap \left(\bigcap_{j \in S_1} \overline{A_j}\right) \mid \bigcap_{j \in S_2} \overline{A_j}\right) &\leq \Pr\left(A_i \mid \bigcap_{k \in S_2} \overline{A_k}\right) \\ &= \Pr(A_i) \leq x_i \prod_{ij \in E(G)} (1 - x_j). \end{aligned} \quad (20.9)$$

We next bound the denominator of the term in (20.8). If $S_1 = \emptyset$, then the denominator is 1, and (20.7) follows readily. Thus, we may assume that $S_1 \neq \emptyset$. Let $S_1 = \{j_1, \dots, j_k\}$, where $k \geq 1$ by assumption. We note that $|S_1| + |S_2| = k + |S_2| = |S|$, and so $|S_2| + k - 1 < |S|$. By the expression in (20.2),

$$\begin{aligned} \Pr\left(\bigcap_{i=1}^k \overline{A_{j_i}} \mid \bigcap_{k \in S_2} \overline{A_k}\right) &= \prod_{i=1}^k \Pr\left(\overline{A_{j_i}} \mid \left(\bigcap_{\ell=i+1}^k \overline{A_{j_\ell}}\right) \cap \left(\bigcap_{k \in S_2} \overline{A_k}\right)\right) \\ &= \prod_{i=1}^k \left(1 - \Pr\left(A_{j_i} \mid \left(\bigcap_{\ell=i+1}^k \overline{A_{j_\ell}}\right) \cap \left(\bigcap_{k \in S_2} \overline{A_k}\right)\right)\right) \\ &\geq \prod_{i=1}^k (1 - x_i), \quad (\text{induction}) \end{aligned} \quad (20.10)$$

where we note that the inductive assumption can be applied here since each event is conditioned on a subset of events of cardinality at most $|S_2| + k - 1 < |S|$. Thus, substituting (20.9) and (20.10) into (20.8), we have the quotient

$$\Pr\left(A_i \mid \bigcap_{j \in S} \overline{A_j}\right) \leq \frac{x_i \prod_{ij \in E(G)} (1 - x_j)}{\prod_{i=1}^k (1 - x_i)} \leq x_i,$$

which completes the proof of (20.7) by induction. Finally, using (20.7) and applying Bayes' Theorem on conditional probability repeatedly, it follows that

$$\begin{aligned} \Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) &= \prod_{i=1}^n \Pr\left(\overline{A_i} \mid \bigcap_{j=1}^{i-1} \overline{A_j}\right) \\ &= \prod_{i=1}^n \left(1 - \Pr\left(A_i \mid \bigcap_{j=1}^{i-1} \overline{A_j}\right)\right) \\ &\geq \prod_{i=1}^n (1 - x_i) > 0. \end{aligned}$$
■

As a consequence of the [general case of the Lovász Local Lemma](#), we have the following version of the symmetric case, where $e = 2.718\dots$ is the base of the natural logarithm.

Lemma 20.21 (Lovász Local Lemma: Symmetric case, version II) *Let A_1, \dots, A_n be events in a probability space with a dependency graph G of maximum degree Δ . If*

$$\Pr(A_i) \leq p \quad \text{and} \quad ep(\Delta + 1) \leq 1$$

for all $i \in [n]$, then

$$\Pr\left(\bigcap_{i=1}^n \overline{A_i}\right) > 0.$$

Proof If $\Delta = 0$, then $p \leq 1/e < 1$. Furthermore, the events A_1, \dots, A_n are mutually independent. The desired result therefore follows from [Observation 20.15](#). Hence, we may assume that $\Delta \geq 1$. If, in the statement of [Lemma 20.20](#), we let $x_i = x$ for all $i \in [n]$, then the condition

$$\Pr(A_i) \leq x_i \prod_{ij \in E(G)} (1 - x_j) \tag{20.11}$$

simplifies to $\Pr(A_i) \leq x(1 - x)^\Delta$, noting that event A_i is mutually independent of a set of all the other events A_j , except for at most Δ such events. The function $x(1 - x)^\Delta$ is maximised when $x = 1/(\Delta + 1)$ and attains the optimum value of $\Delta^\Delta / (\Delta + 1)^{\Delta+1}$. Thus, condition (20.11) becomes

$$\Pr(A_i) \leq \frac{\Delta^\Delta}{(\Delta + 1)^{\Delta+1}}$$

for all $i \in [n]$. Since $ep(\Delta + 1) \leq 1$, however, we note that

$$\Pr(A_i) \leq p \leq \frac{1}{\Delta + 1} \cdot \frac{1}{e} < \frac{\Delta^\Delta}{(\Delta + 1)^{\Delta+1}},$$

and using the well-known fact that for all $\Delta \geq 1$, we have

$$\frac{1}{e} < \left(1 - \frac{1}{\Delta + 1}\right)^\Delta.$$

Thus, condition (20.11) is indeed satisfied and so our desired result follows from Lemma 20.20. \blacksquare

The **Lovász Local Lemma** was devised by Erdős and Lovász in 1973 to tackle the problem of hypergraph 2-colourings. Recall, from Chapter 16, that a **hypergraph** H consists of a finite vertex set $V(H)$ and a finite multiset $E(H)$ of edges, where each edge is a subset of $V(H)$. Moreover, a hypergraph is **2-colourable** if there is a 2-colouring of the vertices containing no monochromatic hyperedge. Recall that a hypergraph H is **k -uniform** if every edge in H has size k , while the **degree** of a vertex v in H is the number of edges of H which contain v . The hypergraph H is **k -regular** if every vertex has degree k in H .

Theorem 20.22 ([9]) *Every k -uniform, k -regular hypergraph is 2-colourable, provided $k \geq 9$.*

Proof Let H be a k -uniform, k -regular hypergraph. Consider a probability space whose elements are the red-blue vertex colourings of the vertices of H . Define the probabilities by colouring each vertex red with probability $\frac{1}{2}$, independently of the choice for any other vertex. Thus, we colour each vertex of H randomly and independently either red or blue with equal probability. For each edge f of H , let A_f be the event that the edge f is monochromatic. Thus, either every vertex in f is coloured red or every vertex in f is coloured blue. The probability of A_f is

$$\Pr(A_f) = 2 \times \left(\frac{1}{2}\right)^k = \frac{1}{2^{k-1}}.$$

Furthermore, the event A_f only depends on an event A_ℓ if the edges f and ℓ intersect. Since every edge in H intersects at most $k(k-1)$ other edges, the dependency graph G for the events A_1, \dots, A_n has maximum degree at most $\Delta = k(k-1)$. For each $k \geq 9$, we note that

$$e(k(k-1)+1) \leq 2^{k-1}.$$

Thus, setting $p = 1/2^{k-1}$, we have

$$\Pr(A_f) \leq p \quad \text{and} \quad ep(\Delta+1) \leq 1$$

for every edge f in H . Thus, by the Lovász Local Lemma (Lemmas 20.20 and 20.21), there is a positive probability of all events not occurring; that is, there exists a red-blue vertex colourings of the vertices of H with no monochromatic edge, implying that H is 2-colourable. \blacksquare

In 1988, Alon and Bregman [1] improved the Erdős-Lovász result in Theorem 20.22. We state their result without proof.

Theorem 20.23 ([1]) *Every k -uniform, k -regular hypergraph is 2-colourable, provided $k \geq 8$.*

In 1992, Thomassen [21], in turn, improved the Alon-Bregman result of Theorem 20.23.

Theorem 20.24 ([21]) *Every k -uniform, k -regular hypergraph is 2-colourable, provided $k \geq 4$.*

As remarked by Alon and Bregman [1], not every 3-uniform, 3-regular hypergraph is 2-colourable, as may be seen by considering the Fano plane, shown in Figure 20.1.

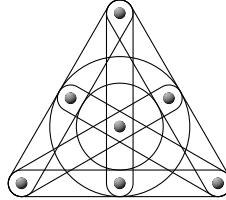


Figure 20.1: The Fano plane.

For $m \geq 2$, a hypergraph is **m -colourable** if there is an m -colouring of the vertices with no monochromatic hyperedge; that is, every edge in H contains vertices of all m possible colours. Henning and Yeo [14] proved the following result.

Theorem 20.25 ([14]) *If m is an integer such that $2 \leq m \leq k/(3\ln(k))$, then every k -uniform, k -regular hypergraph is m -colourable.*

Proof Let H be a k -uniform, k -regular hypergraph, and let m and k be defined as in the statement of the theorem. Let $E(H) = \{e_1, \dots, e_{|E(H)|}\}$. Consider a random colouring of H , such that every vertex is independently assigned a random colour in the set $[m]$ and let A_j^i be the event that colour i is missing from e_j . The probability of A_j^i is

$$\Pr(A_j^i) = \left(1 - \frac{1}{m}\right)^k.$$

Furthermore, the event A_j^i only depends on A_a^b if the edge e_a intersects e_j . Since every edge of H intersects at most $k(k-1)$ other edges, the event A_j^i is independent of the collection of events $\{A_a^b : e_a \cap e_j = \emptyset\}$, implying that A_j^i depends on at most

$$\Delta = (k(k-1) + 1)m - 1$$

other events. By the Lovász Local Lemma, there is therefore a positive probability of all events not occurring if

$$e \left[\left(1 - \frac{1}{m}\right)^k \right] \left[[(k(k-1) + 1)m - 1] + 1 \right] < 1.$$

In order to establish the inequality above, let $m = \Theta k / (\ln(k))$ and by assumption $\Theta \leq 1/3$. As $1 + \ln(k)[3 - \frac{1}{\Theta}] + \ln(\Theta) < 0$ and $(1 - 1/x)^x < 1/e$ for all $x > 1$, it follows that

$$1 + \ln(k)[3 - \frac{1}{\Theta}] + \ln(\Theta) < 0.$$

The inequality above may be rewritten as

$$1 - \frac{\ln(k)}{\Theta} + 2\ln(k) + \ln(k\Theta) < 0,$$

from which it follows that

$$1 - \frac{k}{m} + 2\ln(k) + \ln\left(\frac{k\Theta}{\ln(k)}\right) < 0. \quad (20.12)$$

But (20.12) is equivalent to

$$e^{\left[\left(\frac{1}{e}\right)^{k/m}\right]} k^2 m < 1,$$

from which it follows that

$$e^{\left[\left(1 - \frac{1}{m}\right)^m\right]^{k/m}} k^2 m < 1,$$

in turn implying that

$$e^{\left[\left(1 - \frac{1}{m}\right)^k\right] [(k(k-1)+1)m]} < 1. \quad \blacksquare$$

Exercises

- 20.1 Use Stirling's formula to prove Corollary 20.3.
- 20.2 In the proof of Theorem 20.4, prove that for each edge $e \in E$, $E(X_e) = \frac{1}{2}$.
- 20.3 In the proof of Theorem 20.13, prove that for $i = 3, \dots, \ell$, $E(X_i) = p^i(n)_i / 2i$.
- 20.4 In the proof of Theorem 20.13, prove that $\Pr(\alpha(G) \geq \frac{n}{2k}) < \frac{1}{2}$, for n sufficiently large.

Projects

This section contains a single project in which the reader is led in a step-by-step fashion to derive a bound on the total domination number of a graph by exploiting the relationship between this parameter and the transversal number of a hypergraph.

Project 20.1: Transversals in hypergraphs

A subset T of vertices in a hypergraph H is a **transversal** if T intersects every edge of H and the **transversal number** $\tau(H)$ of H is the minimum size of a transversal in H .

In this project, the reader is led to apply the linearity of expectation to determine an upper bound on the transversal number of a uniform hypergraph in terms of the number of vertices and number of edges, and the average degree, where we recall that the **average degree** of an r -uniform hypergraph H with n vertices and m edges is rm/n .

For $r \geq 2$, let H be an r -uniform hypergraph with n vertices, m edges and with average degree $d = rm/n$, and such that $\delta(H) \geq 1$. For $0 \leq p \leq 1$, choose each vertex in H independently with probability p . Let X be the set of chosen vertices and let Y be the set of edges from which no vertex was chosen. Then, $E(|X|) = np$ and $E(|Y|) = m(1-p)^r$. By linearity of expectation, we have that $E(|X| + |Y|) = E(|X|) + E(|Y|) = np + m(1-p)^r$. Hence, if we add to X one vertex from each edge in Y , we obtain a transversal T of H such that $E(|T|) \leq np + m(1-p)^r$, implying that $\tau(H) \leq np + m(1-p)^r$.

Tasks

1. Let $f(p) = np + m(1-p)^r$. Show that this function is minimised when

$$p^* = 1 - \left(\frac{1}{d}\right)^{\frac{1}{r-1}},$$

which is a legitimate value for p as $d \geq \delta(H) \geq 1$.

2. Show that

$$f(p^*) = \left(1 - \left(\frac{r-1}{r}\right)\left(\frac{1}{d}\right)^{\frac{1}{r-1}}\right)n.$$

3. Taking $p = (\ln d)/r$, show that $E(|T|) \leq n(\ln d + 1)/r$.
4. Use the result of [Task 3](#) above to show that

$$\tau(H) \leq \left(1 - \left(\frac{r-1}{r}\right)\left(\frac{1}{d}\right)^{\frac{1}{r-1}}\right)n \leq n(\ln d + 1)/r.$$

We remark that the above result was established in [\[5\]](#).

5. Recall, by [Observation 16.106](#), that the transversal number of the open neighbourhood hypergraph of a graph is precisely the total domination number of the graph. As a consequence of the result in [Task 4](#) above, prove that

$$\gamma_t(G) \leq \left(\frac{1 + \ln \delta}{\delta}\right)n,$$

where G is a graph of order n with minimum degree $\delta \geq 2$.

Further reading

The book by [Alon and Spencer \[2\]](#) is an outstanding reference work on the probabilistic method.

- [1] N Alon and Z Bregman, 1988. *Every 8-uniform 8-regular hypergraph is 2-colorable*, Graphs and Combinatorics, **4**, pp. 303–306.
- [2] N Alon and J Spencer, 2000. *The Probabilistic Method*, Second edition, Wiley-Interscience, New York (NY).
- [3] VI Arnautov, 1974. *Estimation of the exterior stability number of a graph by means of the minimal degree of the vertices*, Prikl. Mat. i Programmirovaniye Vyp., **11**, pp. 3–8.

- [4] Y Caro, 1979. *New results on the independence number*, Technical Report, Tel-Aviv University, Tel-Aviv.
- [5] Y Caro and MA Henning, 2014. *Simultaneous domination in graphs*, Graphs and Combinatorics, **30**, pp. 1399–1416.
- [6] P Dankelmann, W Goddard, MA Henning and R Laskar, 2006. *Simultaneous graph parameters: Factor domination and factor total domination*, Discrete Mathematics, **306**, pp. 2229–2233.
- [7] R Diestel, 1997. *Graph Theory*, Springer, New York (NY).
- [8] P Erdős, 1959. *Graph theory and probability*, Canadian Journal of Mathematics, **13**, pp. 34–38.
- [9] P Erdős and L Lovász, 1973. *Problems and results on 3-chromatic hypergraphs and some related questions*, pp. 609–627 in A Hajnal, R Rado and VT Sós (Eds), *Infinite and Finite Sets*, North-Holland, Amsterdam.
- [10] P Erdős and A Rényi, 1960. *On the evolution of random graphs*, Publ. Math. Inst. Hungar. Acad. Sci. **5**, pp. 17–61.
- [11] J Harant and MA Henning, 2005. *On double domination in graphs*, Discussiones Mathematicae Graph Theory, **25**, pp. 29–34.
- [12] J Harant, A Pruchnewski and M Voigt, 1998. *On dominating sets and independent sets of graphs*, Journal on Combinatorial Probability and Computing, **8**, pp. 547–553.
- [13] TW Haynes, ST Hedetniemi and PJ Slater, 1998. *Fundamentals of Domination in Graphs*, Marcel Dekker, New York (NY).
- [14] MA Henning and A Yeo, 2013. *2-Colorings in k -regular k -uniform hypergraphs*, European Journal of Combinatorics, **34**, pp. 1192–1202.
- [15] MA Henning and A Yeo, 2014. *The domination number of a random graph*, Utilitas Mathematica, **94**, pp. 315–328.
- [16] S Jukna, 2001. *Extremal Combinatorics with Applications in Computer Science*, Springer, Berlin.
- [17] L Lovász, 1973. *Coverings and colorings of hypergraphs*, Congressus Numerantium, **8**, pp. 3–12.
- [18] C Payan, 1975. *Sur le nombre d'absorption d'un graphe simple*, Cahiers Centre Études Rech. Opér. **17**, pp. 307–317.
- [19] J Spencer, 1994. *Ten Lectures on the Probabilistic Method*, Second edition, SIAM, Philadelphia (PA).
- [20] T Szele, 1943. *Kombinatorikai vizsgálatok az irányított teljes gráffal kapcsolatban*, Math. Phys. Lapok, **50**, pp. 223–256.
- [21] C Thomassen, 1992. *The even cycle problem for directed graphs*, Journal of the American Mathematical Society, **5**, pp. 217–229.
- [22] P Turán, 1941. *Eine Extremalaufgabe aus der Graphentheorie*, Mat. Fiz. Lapok, **48**, pp. 436–452.
- [23] VK Wei, 1981. *A lower bound on the stability number of a simple graph*, Technical Memo, Bell Laboratories.

- [24] B Wieland and A Godbole, 2001. *On the domination number of a random graph*, Electronic Journal of Combinatorics, **8**, #R37 (13).

Index

Subject index

Symbols

- 2-cell, 427
- embedding, 427
- $\langle\gamma, i\rangle$ -graphs, 557
- $\langle i, \alpha\rangle$ -graphs, 559
- $\langle r, s\rangle$ -dominating set, 613
- $\langle r, s\rangle$ -domination number, 613

A

- absolute
 - centre, 147, 155
 - median, 147, 155
- acquaintance graph, 29, 625
- activity digraph, 88
- acyclic graph, 37
- adjacency
 - list, 19
 - matrix, 18
 - table, 19
- adjacent
 - edge, 2
 - from, 17
 - to, 17
 - vertex, 2
- admissible face, 409
- algorithm, 55
 - complexity of, 55
 - for computing
 - blocks of a connected graph, 364
 - the centre of an unweighted, undirected tree, 152
 - the chromatic number of a graph, 474
- the chromatic number of a graph, 467
- clique numbers, 71
- the connectivity of a graph, 378
- cut-vertices, 143
- degree sequences, 56
- the domination number of a tree, 549
- eulerian circuits or trails in (multi)digraphs, 310
- eulerian circuits or trails in (multi)graphs, 298
- Fibonacci numbers, 74, 83
- greatest common divisors, 83
- longest paths in digraphs, 100
- maximum matchings in bipartite graphs, 252
- maximum matchings in general graphs, 258
- maximum weight matchings in general graphs, 267
- optimal Chinese postman tours, 304
- shortest paths in graphs (Dijkstra), 93
- shortest paths in graphs (Floyd), 98
- shortest spanning trees (Kruskal), 121
- shortest spanning trees (Prim), 125
- for constructing breadth-first search trees, 143
- depth-first search trees, 131

- for deciding
 - bipartiteness of a graph, 488
 - planarity, 411
 - set membership (binary search), 79
 - for sorting sets
 - bubble sort, 81
 - merge sort, 81, 82
 - sequential sort, 81
 - almost
 - hamiltonicity, 344
 - perfect matching, 226
 - traceability, 344
 - alternating
 - forest subroutine, 254
 - path, 224
 - analytic hierarchy process, 521
 - ancestor, 128
 - Anderson's Theorem, 247
 - antisymmetric, 430
 - arc, 16
 - f -point of, 153
 - set, 16
 - art gallery problem, 442
 - artificial variable, 200
 - assignment problem, 225, 285
 - asymmetric digraph, 17, 501
 - asymptotic upper bound, 60
 - augmenting path, 224
 - technique, 177
 - automorphism, 16
- B**
- back edge, 131
 - backward arc, 42
 - balanced
 - complete
 - k -partite graph, 7
 - multipartite graph, 7
 - rooted tree, 128
 - basic
 - feasible solution, 200
 - operation, 56
 - variable, 202
 - Berge's Theorem, 224
 - bicentroidal tree, 698, 722
 - biclique, 7
 - “big \mathcal{O} ” order notation, 60
 - properties of, 61
- binary
 - relation, 430
 - search membership algorithm, 79
 - tree, 129
 - binding
 - number, 246
 - set, 246
 - bipartite
 - closure, 349
 - complement, 588
 - graph, 6
 - characterisation of, 38, 451
 - maximum matching algorithm for, 252
 - Ramsey number, 639
 - bishop's graph, 604
 - block
 - graph, 361, 381
 - of a graph, 361, 408
 - blocks
 - algorithm for computing of, 364
 - characterisation of, 363
 - Bondy and Chvátal's Theorem, 329
 - boolean
 - clause, 66
 - conjunction, 66
 - literal, 66
 - operation
 - and, 66
 - or, 66
 - variable, 66
 - negation, 66
 - boundary, 395, 594
 - branch
 - set, 399
 - vertex, 400
 - breadth-first tree search, 133, 143
 - Brélaz's colouring heuristic, 469
 - brickfactory minimisation problem, 417
 - bridge, 40, 361
 - characterisation of, 41
 - computation of, 142
 - bridges of Königsberg, 294
 - Brooks' Theorem, 463
 - Brown's algorithm, 474
 - bubble sort algorithm, 81

C

- capacitated
 - postman problem, 307
 - vehicle routing problem, 353
 - Clarke and Wright savings method for, 355
- capacity
 - constraint, 196
 - function on a network, 174
 - of a cut, 175
- Caro-Wei Theorem, 730
- cartesian product graph, 10
- Catalan number, 691, 721
- Cauchy-Schwarz inequality, 731
- Cayley's (tree) formula, 685
- central vertex, 148
- centre, 147, 148
 - of a graph, 148
 - of an unweighted tree, 150, 151
- centroid, 698, 722
- centroidal tree, 698, 722
- certificate, 63
- chess graph, 604
- child, 128
- Chinese postman problem, 301
 - variants of, 306
- tour, 301
 - algorithm for computing, 304
 - optimal weight of, 302
- chordal graph, 544
- Christofides' algorithm, 341
- chromatic
 - index, 477
 - number, 459
- Chvátal's
 - Theorem, 332
 - Watchman Theorem, 442
- circuit
 - in a digraph, 42
 - in a graph, 37
- circulant, 8
- Clarke and Wright savings method, 355
- class
 - 1 graphs, 481
 - 2 graphs, 481
 - timetabling, 450, 495
- classical
 - centre, 149
 - median, 152
- clause, 66
- clique, 6
 - chromatic number, 495
 - colouring, 495
 - number, 71, 463
- closed
 - neighbourhood, 4
 - trail or walk, 36
 - under the minor ordering of graphs, 431
 - minimal element of, 431
- closure of a graph, 329
- co-NP
 - certificate, 63
 - class of algorithms, 63
- coarseness, 423
- colour
 - class, 451, 477
 - degree, 469
 - print scheduling problem, 355
- complement
 - of a graph, 10
 - of a hypergraph, 587
 - of a set of graphs, 431
- Complementary Slackness Theorem, 197, 263
- complete
 - bipartite graph, 7
 - graph, 6
 - k -partite graph, 7
 - ℓ -ary tree, 129
 - order of, 129
 - multipartite graph, 7
- complexity
 - classes, 70
 - of the domination problem, 548
 - of the edge colouring problem, 481
 - of the vertex colouring problem, 451
- component, 38
- composite circulant, 9
- computation problem, 70
- conjunction, 66
- conjunctive normal form, 66

- connected
 - graph, 38
 - vertices, 38
- connection set, 8
- connectivity, 368
 - algorithm for computing, 378
- contraction
 - graph, 382
 - of an edge, 398
- convex
 - embedding of a planar graph, 402
 - polygon triangulation, 721
- corona, 534
- course graph, 450
- cover, 227
- critical
 - path, 88
 - with respect to the chromatic index, 480
 - with respect to the chromatic number, 459
- crossing number of a graph
 - in the plane, 415
 - on a torus, 427
 - on an orientable surface, 427
- cubic graph, 6
- cut-edge, 361
- cut-vertex, 40, 360
 - characterisation of, 40
 - computation of, 142
- cycle
 - edge, 41
 - graph, 37
 - in a digraph, 42
 - in a graph, 41
 - index, 705
 - length, 37
- D**
 - de Bruijn
 - pseudodigraph, 319
 - sequence, 318
 - decision
 - problem, 63
 - theory, 63
 - degree, 4
 - sequence, 11
 - algorithm for, 56
- characterisation of, 12
- realisation, 12
- demand vertex, 195
- dense graph, 19
- depth-first
 - index, 131
 - tree search, 130
 - algorithm for, 131
 - backtracking in, 131
 - time complexity of, 133
- descendant, 128
- diameter, 148
 - bounds in terms of radius, 149
- digraph, 16
 - order, 16
 - size, 16
- Dijkstra's algorithm, 93
 - time complexity of, 95
- Dirac's Theorem, 326
- directed
 - edge, 16
 - graph, *see* digraph
 - tree, 127
- disconnected graph, 38
- distance
 - domination, 610
 - in a digraph, 42
 - in a weighted graph, 90
 - in an unweighted graph, 38
 - k dominating set, 610
 - k domination number, 611
 - preserving spanning tree, 119
- domain constraint, 196
- dominating set, 529
- domination
 - inequality chain, 556, 571
 - number, 529
 - perfect graph, 557
 - sequence, 572
 - characterisation of, 572
- dual graph, 443
- E**
 - eccentricity, 148
 - edge, 2
 - colouring, 477
 - connectivity, 368
 - cover, 585
 - cut, 361

- disjoint, 297, 375
- f*-point of, 153
- independence, 223
 - number, 227
- induced subgraph, 4
- separating set, 361, 375
- set, 2
- union, 10
- weight, 90
 - function, 553
- Edmonds' Blossom-Shrinking Lemma, 254
- elementary
 - circulant, 9
 - subdivision, 399
- embedded, 427
- embedding
 - of a graph in the plane, 394
 - of a graph on an orientable surface, 427
- empty graph, 6
- end
 - block, 362, 408
 - vertex, 4
- enumeration
 - of connected labelled graphs, 684
 - of genealogical registers, 687
 - of labelled graphs, 682
 - of labelled rooted trees, 687
 - of labelled trees, 685
 - of unlabelled digraphs, 711
 - of unlabelled graphs, 708
 - of unlabelled multigraphs, 709
 - of unlabelled rooted trees, 693
- Erdős' Theorem, 466
- Erdős-Rényi Theorem, 667
- Euclidean algorithm, 83
- eulerian
 - circuits
 - counting of, 317
 - in a (multi)digraph, 308
 - in a (multi)graph, 294
 - (multi)digraph, 308
 - characterisation of, 308
 - (multi)graph, 294
 - characterisation of, 294
 - trails
- in a (multi)digraph, 308
- in a (multi)graph, 294
- Euler's formula, 396
- even
 - cycle, 37
 - vertex, 4
- examination timetabling, 489
- expected value, 727
- exponential-time algorithm, 63
- external private neighbourhood, 531
- extremal, 655
 - graph, 656
 - number, 655
 - of cycles, 656
 - path, 409
- F**
- face, 395
 - boundary of, 395
- facility location problems
 - classification of, 146
 - objectives in, 146
- factor, 10
- factorable, 10
- factorisation, 11
- fan, 373
- Fano plane, 588, 745
- Fáry embedding, 421
- Fáry's Theorem, 421
- f*-augmenting path, 177
 - algorithm, 181
- Fibonacci numbers, 73
- Five-colour Theorem, 452
- Fleury's algorithm
 - for (multi)digraphs, 310
 - for (multi)graphs, 298
- flow
 - bound in terms of cut, 176
 - conservation constraint, 174, 196
 - in a network, 174
 - value of, 174
 - into a vertex, 174
 - out of a vertex, 174
- Floyd's algorithm, 98
 - time complexity of, 99
- forest, 116
- forward arc, 42
- Four-colour Theorem, 453

- Kempe's attempted proof of, 489
- f*-point, 153
- fragments
 - competition of, 410
 - of a graph, 409
- frame, 662
- framing number, 662
- Friendship Theorem, 625
- full symmetric permutation group, 703
- f*-unsaturated semipath, 177
- G**
 - genealogical register, 687
 - general
 - absolute
 - centre, 147, 159
 - median, 147, 160
 - centre, 147, 153
 - median, 147, 153
 - generalised vertex-to-edge/arc distance matrix, 154
 - generating function, 690
 - genus
 - of a graph, 429
 - of a surface, 426
 - girth, 466, 735
 - graph, 2
 - automorphism, 16
 - (cartesian) product, 10
 - circulant, 8
 - complement, 10
 - component, 38
 - embedded in a plane, 394
 - factor, 10
 - factorisation, 11
 - invariant, 735
 - isomorphism, 15
 - join, 10
 - order, 2
 - size, 2
 - union, 9
 - graphical
 - degree sequence, 12
 - representation of a graph, 12
 - greatest common divisor, 83
 - algorithm for, 83
 - greedy algorithm, 120
- group, 703
- guard, 442
- H**
 - Hall's Marriage Theorem, 227
 - Hall's Theorem, 225
 - hamiltonian, 506
 - cycle, 323, 506
 - graph, 323
 - characterisation in terms of closure, 330
 - necessary conditions for, 325
 - sufficient conditions for, 326
 - path, 323, 505
 - Handshake Lemma, 5
 - Hanson-Wang Theorem, 674
 - Heesch's method of discharging, 457
 - height
 - of a complete ℓ -ary tree, 130
 - of a rooted tree, 128
 - of an ℓ -ary tree, 129
 - hereditary property, 570
 - heuristic
 - for the capacitated vehicle routing problem, 355
 - for the travelling salesman problem, 336, 340, 341
 - hierarchical postman problem, 307
 - hitting set, 585
 - Hungarian
 - forest, 255
 - method, 286
 - tree, 255, 266
 - hypergraph, 585, 744
 - average degree of, 746
 - traversal in, 585
 - traversal number of, 585, 746
 - hypo
 - hamiltonian
 - digraph, 346
 - graph, 346
 - oriented graph, 513
 - traceable
 - digraph, 346
 - graph, 344
 - oriented graph, 513
 - I**
 - identical graphs, 16

- in-neighbourhood, 17
 - incidence bipartite graph, 588
 - incident, 2
 - from, 17
 - to, 17
 - indegree, 17
 - independence number, 555
 - independent, 535
 - dominating set, 555
 - domination number, 555
 - bounds on, 555
 - edges, 223
 - Ramsey number, 641
 - indicator random variable, 728
 - induced subgraph, 4
 - inner face, 395
 - integer partitions, 693
 - internal
 - f -point, 153
 - private neighbourhood, 531
 - vertex, 128
 - internally disjoint, 371
 - intractable problem, 64
 - irredundance
 - number, 570
 - bounds on, 575
 - perfect graph, 573
 - irredundant
 - domination Ramsey number, 642
 - Ramsey number, 641
 - set, 569
 - vertex, 569
 - isolated vertex, 4
 - isomorphic
 - digraphs, 18
 - graphs, 15
 - isomorphism, 15, 18
 - iterative algorithm, 56
- J**
- join, 10
- K**
- k -connected graph, 368
 - Whitney's characterisation of, 373
 - k -chromatic, 459
 - k -colour Ramsey number, 635
- k -colourable, 451
 - k -colouring, 451
 - k -critical
 - with respect to the chromatic index, 480
 - with respect to the chromatic number, 460
 - k -edge
 - chromatic, 477
 - colourable, 477
 - colouring, 477, 635
 - k -edge cut, 361
 - Kempe chain, 454, 489
 - Kempe's attempted proof of the Four-colour Theorem, 489
 - king (of a tournament), 504
 - king's graph, 604
 - knight's
 - graph, 604
 - tour problem, 324
 - knock-out tournament scheduling, 140
 - König-Egerváry theorem, 228
 - Königsberg bridges, 294
 - Kruskal's algorithm, 121
 - time complexity of, 123
 - Kuratowski's Theorem, 402
 - k -vertex cut, 360
- L**
- ℓ -ary tree, 129
 - largest-first colouring heuristic, 468
 - leaf, 116, 128
 - leeway, 178
 - length, 42
 - of a circuit, cycle, path, trail or walk, 36, 90
 - of a semipath or semiwalk, 42
 - line, 282
 - graph, 557
 - linear forest, 325
 - linearity of expectation, 727
 - literal, 66
 - locations
 - of facilities in a facility location problem, 146
 - of users of a facility in a facility location problem, 146
 - longest (directed) path, 88, 99

- algorithm for, 100
 - time complexity of, 101
- loop, 19
- lottery, 528, 599
 - graph, 528, 599
 - number, 527, 599
- Lovász Local Lemma, 738
 - general case, 742
 - symmetric case, version I, 739
 - symmetric case, version II, 743
- lowpoint, 142

- M**
- map colouring, 453, 483
- Markov's inequality, 735
- matched, 225
 - characterisation in bipartite graphs, 225
 - under, 225
 - vertex, 224
- matching, 223
 - alternating cycle, 252
 - forest, 254
 - path, 224
 - augmenting path, 266
 - blossom, 252
 - shrinking, 253
 - flower, 252
 - base of, 252
 - root of, 252
 - number, 227
 - stem, 252
 - tree, 229
 - component, 229
- max-flow min-cut, 177
 - theorem, 178, 377
- maximal
 - independent set, 556
 - irredundant set, 570
 - characterisation of, 571
- maximum
 - benefit postman problem, 307
 - degree, 4
 - chromatic number, 493
 - colouring, 492
 - flow, 174, 176
 - matching, 223

- algorithm for bipartite graphs, 252
- algorithm for general graphs, 258
- characterisation of, 224
- weight matching, 263
 - algorithm, 267
- median, 147, 152
- Menger's Theorem
 - edge form of, 375
 - vertex form of, 371
- merge sort algorithm, 81
- metric, 38
- minimal
 - dominating set, 530
 - characterisation of, 531
 - domination imperfect graph, 559
 - irredundance imperfect graph, 575
 - nonplanar graph, 400
 - total dominating set, 576
 - characterisation of, 577
- minimum
 - connector problem, 115
 - cost flow problem, 193, 197
 - dual of, 197
 - cut, 175, 176
 - degree, 4
 - dominating set, 529
 - spanning tree, 119, 120
- minor
 - characterisation of, 399
 - of a graph, 398, 399
 - ordering, 431
- mixed
 - domination Ramsey number, 642
 - irredundant Ramsey number, 642
- Möbius strip, 425
- multigraph, 19
- multipartite graph, 7
- multiple traversal postman problem, 307
- Murty-Simon Conjecture, 673

- N**
- nearest neighbour heuristic, 336

- negation, 66
- neighbourhood, 4
- network, 174
 - associated with a bipartite graph, 250
 - simplex algorithm, 197, 204, 207, 208
- nonbasic variable, 202
- nondecreasing function, 62
- nondeterministic
 - polynomial-time class of algorithms, 63
- nonisomorphic
 - digraphs, 18
 - graphs, 15
- nonplanar, 394
- nontrivial graph, 2
- Nordhaus-Gaddum bounds
 - for the domination number, 544
 - for the independent domination number, 568
 - for the vertex chromatic number, 466
- NP
 - certificate, 63
 - complete, 64
 - hard, 64
 - class of algorithms, 63
- O**
- obstruction set, 431
- odd
 - component, 232
 - cycle, 37, 266
 - pair partition, 302
 - number, 302
 - weight of, 302
 - vertex, 4
- open
 - neighbourhood, 4
 - hypergraph, 585
 - packing, 595
 - number, 595
 - trail or walk, 36
- order, 2, 16
- ordered pair
 - group, 704
 - permutation, 704
- Ore's Theorem, 331
- orientable surface, 425
- orientation of a graph, 501
- oriented
 - graph, 501
 - postman problem, 306
- Otter's (tree) formula, 700
- outdegree, 17
- outer face, 395
- out-neighbourhood, 17
- P**
- P
 - certificate, 63
 - class of algorithms, 63
 - versus co-NP and NP, 64
- packing, 594, 614
 - number, 594, 614
- pair
 - group, 704
 - permutation, 703
- pairwise
 - edge-disjoint, 297
- parallel edges, 19
- parent, 128
- partially ordered
 - binary relation, 431
 - set, 431
- partite set, 6, 7
- path
 - chromatic number, 494
 - colouring, 494
 - graph, 37
 - in a digraph, 42
 - in a graph, 36
- perfect matching, 226
 - characterisation in a bipartite graph, 232
 - in bipartite graphs, 226
- peripheral vertex, 148
- periphery, 149
- permutation, 701
 - cycle, 701
 - structure of, 702
 - cyclic representation of, 701
 - degree of, 701
 - group, 703
 - degree of, 703
 - order of, 703
 - length of, 702

- subgroup, 703
 type of, 702
 Perron-Frobenius Theorem, 508
 Petersen's Theorem, 233
 planar graph, 346, 393
 Kuratowski's characterisation of, 402
 Wagner's characterisation of, 406
 planarity testing algorithm, 411
 complexity of, 415
 plane
 embedding of a graph, 394
 graph, 394
 point-to
 -edge/arc distance, 160
 -vertex distance, 155
 Pólya's Enumeration Theorem, 707
 polynomial-time
 algorithm, 63
 class of algorithms, 63
 equivalent, 64
 reducible, 64
 Pósa's Theorem, 332
 Prüfer code, 685
 predecessor task, 88
 Prim's algorithm, 125
 time complexity of, 126
 primitive tournament, 508
 private neighbourhood, 531
 probabilistic method, 725
 probability
 distribution, 725
 space, 725
 project scheduling, 111
 proper
 edge colouring, 477
 k -colouring, 451
 k -edge colouring, 477
 vertex colouring, 451
 pseudo
 -graph, 19
 -vertex, 266
- Q**
- queen's
 domination problem, 605
 graph, 530, 555, 604
- R**
 radius, 148
 Ramsey number
 in the classical sense, 626, 628, 726
 multi-colour extensions, 635
 two-colour extensions, 633
 values of, 631, 636, 640, 643
 random
 graph, 734
 variable, 727
 ranking (in a tournament), 506, 520
 rectilinear crossing number, 421
 recursive
 algorithm, 56
 call, 56
 red-blue edge colouring, 628
 reduced cost, 265
 matrix, 286
 reducible configuration, 457
 redundant
 set, 569
 vertex, 569
 reflexive, 430
 regular
 digraph, 18
 graph, 6
 of degree, 6, 18
 Robertson-Seymour theorems, 430
 Roman
 dominating set, 609
 domination number, 609
 rook's graph, 604
 root
 arc (network simplex algorithm), 200
 of a tree, 127
 uniqueness of, 128
 vertex (network simplex algorithm), 200
 rooted
 digraph, 200
 spanning
 subdigraph, 200
 tree, 200
 tree, 127
 characterisation of, 127
 height of, 128

round-robin tournament, 30
rural postman problem, 306

S

satisfiability problem, 67
NP-completeness, 67
satisfiable, 67
savings method of Clarke and Wright, 355
scheduling digraph, 88
score
 of a tournament vertex, 503
 sequence of a tournament, 504
self
 -centred, 167
 -complementary graph, 43
semi
 -path, 42
 -walk, 42
sequential
 colouring heuristic, 467
 membership algorithm, 79
 sort algorithm, 81
set multipartite Ramsey number,
 637
shortest path, 87, 91
simple drawing, 415
simplex algorithm, 197, 203
singular circulant, 9
sink, 89, 174, 195
size
 multipartite Ramsey number,
 637
 of a digraph, 16
 of a graph, 2
 of a tree, 117
smallest
 -last colouring heuristic, 471
 member algorithm, 79
sorting algorithms, 80
source, 89, 174, 195
space complexity, 55
spanning
 forest, 119
 subgraph, 3
 tree, 119
sparse graph, 19
sphere, 425
star graph, 7

stereographic projection, 396

Stirling's formula, 727

strong

 digraph, 42, 501

 orientation, 501

strongly connected, 42

sub

 -digraph, 18

 -dividing vertices, 400

 -division, 399

 -graph, 3

subjective preference selection, 521

successive augmentation, 471

superhereditary property, 570

supply vertex, 195

support vertex, 4

symmetric digraph, 17, 501

T

ternary tree, 129

thickness, 421

Thomassen's Lemma, 402

three houses and utilities problem,
 394

tight edge, 265

time complexity, 55

 of alternative forest subroutine,
 257

 of computing

 bridges, 143

 cut-vertices, 143

 optimal Chinese postman
 tours, 304

 the blocks of a connected
 graph, 367

 the connectivity of a
 connected graph, 380

 of depth-first search tree
 construction, 133

 of Dijkstra's algorithm, 95

 of Fleury's algorithm for
 (multi)digraphs, 309

 of Fleury's algorithm for
 (multi)graphs, 298

 of Floyd's algorithm, 99

 of Kruskal's algorithm, 123

 of the longest (directed) path
 algorithm, 101

 of planarity testing, 415

- of Prim's algorithm, 126
- of the travelling salesman problem, 335
- timetabling
 - of classes, 451, 495
 - of examinations, 489
- topological minor, 400
- toroidal crossing number, 427
- torus, 425
- total
 - dominating set, 576
 - domination
 - edge critical graph, 593
 - number, 576
- tournament, 503
 - graph, 30
 - king, 504
 - ranking, 506, 520
 - score sequence, 504
- Towers of Hanoi
 - puzzle, 56
 - recurrence relation, 57
 - recursive algorithm, 58
- traceable, 323
- tractable problem, 64
- trail
 - in a digraph, 42
 - in a graph, 36
- transitive, 430
 - tournament, 504
- transportation problem, 193
- transversal number, 585
- travelling salesman problem, 334
 - heuristic for, 340
 - lower bound on solutions of, 338
 - nearest neighbour heuristic for, 336
- tree, 116
 - characterisation of, 116
 - properties of, 118
 - search, backtracking in, 131
 - size, 117
- triangle, 37
 - inequality, 38, 339
- triangulated, 433
- triangulation, 443
- trivial
 - block, 361
- graph, 2
- path, walk or trail, 36
- Turán
 - graph, 657
 - number, 657
- Turán's Theorem, 659
- Tutte's Theorem, 232, 402
- Tutte-Berge formula, 235
- U**
- unavoidable set, 457
- underlying
 - decision problem, 71
 - digraph, 174, 200
 - graph, 17
- union, 9
- universal vertex, 167, 380
- unmatched vertex, 224
- upper
 - domination
 - number, 552
 - Ramsey number, 642
 - irredundance number, 570
 - total domination number, 576, 590
- V**
- vehicle routing problem, 353
- Clarke and Wright savings method for, 355
- vertex, 2
 - adjacency, 2
 - arc incidence matrix, 196
 - of attachment, 409
 - chromatic number, 459
 - colouring, 451
 - cover, 85, 227
 - covering number, 227
 - cut, 360
 - induced subgraph, 4
 - pancyclic, 505
 - separating set, 360, 371
 - set, 2, 16
 - to-edge/arc distance, 153
 - transitive, 16
 - weight function, 553
- Vizing's Conjecture, 542

- edge bound on the domination number, 539
 Theorem, 478
 Voyage Round the World, 323
- W**
 Wagner's Conjecture, 431
 Wagner's Theorem, 406
 walk
 in a digraph, 41
 in a graph, 36
 weakly connected, 42, 198
 weight, 87, 90
 matrix, 91
 of a subgraph, 120
 weighted
 digraph, 90
 graph, 87, 90
 well-covered graph, 559
- characterisation of, 559
 Welsh-Powell colouring heuristic, 468
 Whitney's
 characterisation of k -connected graphs, 373
 Theorem, 369
 windy postman problem, 306
 Woodall's Theorem, 248
 worst-case complexity
 of an algorithm, 60
- Z**
 Zarankiewicz
 number, 659
 problem, 659
 value, 660
 Zarankiewicz' Conjecture, 417
 zero flow, 174

Author index

A

- Acharya, BD, 539
 Aharoni, R, 544
 Allan, RB, 557
 Alon, N, 734
 Anderson, I, 246
 Appel, KI, 459
 Archdeacon, D, 583
 Arnautov, VI, 732
 Auslander, L, 407

B

- Bachmann, PGH, 60
 Beineke, LW, 423
 Berge, C, 5, 224
 Biedl, T, 236
 Bollobás, B, 532
 Bondy, JA, 330
 Boole, G, 66
 Booth, KS, 407
 Bregman, Z, 744
 Brélaz, D, 469
 Brešar, B, 555
 Brigham, RC, 580
 Brooks, RL, 463

Brown, JR, 473

- Bujtás, C, 532
 Burger, AP, 517, 637
 Burr, SA, 633

C

- Caro, Y, 730
 Carrington, JR, 580
 Catalan, EC, 692
 Cayley, A, 454, 686
 Champagna, L, 307
 Chartrand, GT, 370, 424
 Chen, G, 643
 Chen, XG, 562
 Christofides, N, 342
 Chung, FRK, 659
 Chvátal, V, 331, 586, 634
 Clark, WE, 542
 Clarke, G, 355
 Cockayne, EJ, 532
 Cook, SA, 65

D

- Dankelmann, P, 613
 Dantzig, GB, 198, 354

- Daskin, MS, 307
 Dawes, RM, 580
 de Jaenisch, CF, 530
 de Morgan, A, 456
 Demaine, ED, 236
 Dijkstra, EW, 92
 Dirac, PAM, 328
 Dorbec, P, 591
 Duncan, CA, 236
- E**
 Edmonds, JR, 253, 306
 Egervary, E, 228
 Ellis-Monaghan, J, 583
 Entringer, RC, 663
 Erdős, P, 668
 Erwin, D, 610
 Euclid of Alexandria, 84
 Euler, L, 295
 Even, S, 407
 Exoo, G, 634
- F**
 Fary, I, 421
 Faudree, RJ, 633
 Favaron, O, 562
 Fibonacci, L, 73
 Fischer, D, 583
 Fisk, S, 442
 Fleischer, R, 236
 Fleury, M, 297
 Floyd, RW, 97
 Frederickson, GN, 306
 Frick, M, 515
 Frobenius, FG, 510
 Froncek, D, 583
 Fulkerson, DR, 178
 Fulman, J, 558
 Furedi, Z, 674
- G**
 Gaddum, JW, 466
 Gallai, T, 344
 Garey, MR, 419
 Gavlas, H, 662
 Gerencser, L, 633
 Gillet, B, 289
 Gimbel, J, 561
 Gleason, AM, 629
- Goddard, W, 247, 547
 Golden, BL, 307
 Grotsch, H, 467
 Graham, RL, 659
 Graver, JE, 632
 Greenwood, RE, 629
 Grinstead, CM, 632
 Grotschel, M, 347
 Groves, GW, 307
 Grundlingh, WR, 601
 Guan, M, 306
 Gupta, RP, 478
 Guthrie, F, 455
 Guy, RK, 428
 Gyarfas, A, 633
- H**
 Haken, W, 459
 Hakimi, S, 12
 Hales, RS, 246
 Hall, P, 226
 Hamilton, WR, 326
 Hanson, D, 674
 Harary, F, 370, 639
 Hartnell, B, 544
 Hattingh, JH, 642
 Haviland, J, 565
 Haynes, TW, 615
 Heawood, PJ, 458
 Hedetniemi, ST, 150
 Henning, MA, 229
 Hierholzer, C, 294, 296
 Hopcroft, J, 407
 Horton, JD, 345
- J**
 Jaeger, F, 544
 Johnson, DS, 419
 Johnson, EL, 262, 306
- K**
 Kalbfleisch, JG, 632
 Kane, VG, 246
 Karp, RM, 180, 451
 Katreni, P, 516
 Kelly, JB, 465
 Kelly, LM, 465
 Kelmans, A, 538
 Kemnitz, A, 513

Kempe, AB, 457
Klavžar, S, 546
Kleitman, DJ, 420
Kobourov, SG, 236
König, D, 39
Kostochka, A, 538
Kruskal, JB, 122, 338
Kuhn, H, 286
Kuratowski, K, 401

L

Löwenstein, C, 565
Lam, PCB, 563
Landau, EGH, 62
Laskar, R, 557
le Roux, J, 307
Lemieux, PF, 307
Lenstra, JK, 306
Lester, RF, 177
Leucker, GS, 407
Levin, LA, 70
Lovász, L, 738
Lovász, L, 232
Lucas, FÉA, 57
Lyle, J, 568

M

Ma, DX, 562
Malandraki, C, 307
Mantel, W, 656
McCuaig, W, 535
McDiarmid, C, 585
McKay, BD, 632
Menger, K, 372
Min, ZK, 632
Minieka, E, 306
Mitchell, SL, 549
Mitchem, J, 466
Mohanty, SP, 246
Moon, JW, 505
Moore, EF, 135
Moore, JL, 558
Murty, USR, 513
Mycielski, J, 465
Mynhardt, CM, 573

N

Nielsen, M, 516
Nordhaus, EA, 466

O

Oellermann, OR, 642
Orden, A, 285
Ore, Ø, 8
Orloff, CS, 306
Otter, R, 700

P

Payan, C, 535, 544, 732
Pegg, E, 635
Pentico, DW, 285
Perron, O, 509
Perter, S, 407
Petersen, PCJ, 233, 460
Plummer, M, 559
Pólya, G, 708
Pósa, L, 333
Powell, MB, 466
Prüfer, H, 685
Prim, RC, 124, 338

R

Rédei, L, 505
Radziszowski, SP, 632
Rall, D, 545
Ramser, JH, 354
Ramsey, FP, 626
Rautenbach, D, 541
Ravindra, G, 559
Reed, B, 537
Rényi, A, 669
Rinnooy Kan, AHG, 306
Robbins, HE, 502
Roberts, SM, 632
Robertson, GN, 430
Rosta, V, 633
Rousseau, CC, 633
Roux, A, 614

S

Sampathkumar, E, 539
Schelp, RH, 633
Schiermeyer, I, 514
Schultz, M, 662
Seager, S, 583
Seymour, PD, 432
Shepherd, B, 535
Shiu, WC, 563
Shreve, WE, 642

Slater, PJ, 615
 Southey, J, 553
 Spencer, JH, 630, 732
 Stocker, C, 538
 Stodolsky, BY, 538
 Strauss, EG, 246
 Suen, S, 542
 Sumner, DP, 558
 Sun, L, 563
 Swart, HC, 546
 Szabó, T, 544
 Szekeres, G, 630
 Szele, T, 728

T

Tarjan, RE, 407
 Tarr, J, 544
 Thapa, MN, 263
 Thomas, R, 461
 Thomassé, S, 587
 Thomassen, C, 404
 Topp, J, 557
 Turán, P, 418
 Tutte, WT, 232, 403
 Tuza, Z, 532

V

van Aardt, S, 516
 van der Merwe, LC, 593
 van Vuuren, JH, 307, 637
 Verstraëte, J, 564
 Vestergaard, PD, 561
 Vitray, RP, 580

Vizing, VG, 479
 Volkmann, L, 559
 von Neumann, J, 80
 Votaw, DF, 285

W

Wagner, K, 406
 Wakabayashi, Y, 348
 Walikar, HB, 539
 Wallis, WD, 480
 Wang, P, 674
 Wei, B, 583
 Wei, VK, 730
 Welsh, DJA, 466
 Whitney, H, 369
 Wilson, RJ, 481
 Win, Z, 306
 Wong, RT, 307
 Woodall, DR, 246
 Wright, JW, 355

X

Xuong, NH, 535, 546

Y

Yackel, J, 632
 Yeo, A, 229, 583
 Yuster, R, 583

Z

Zarankiewicz, K, 419
 Zverovich, IE, 558
 Zverovich, WE, 558
 Zykov, AA, 465