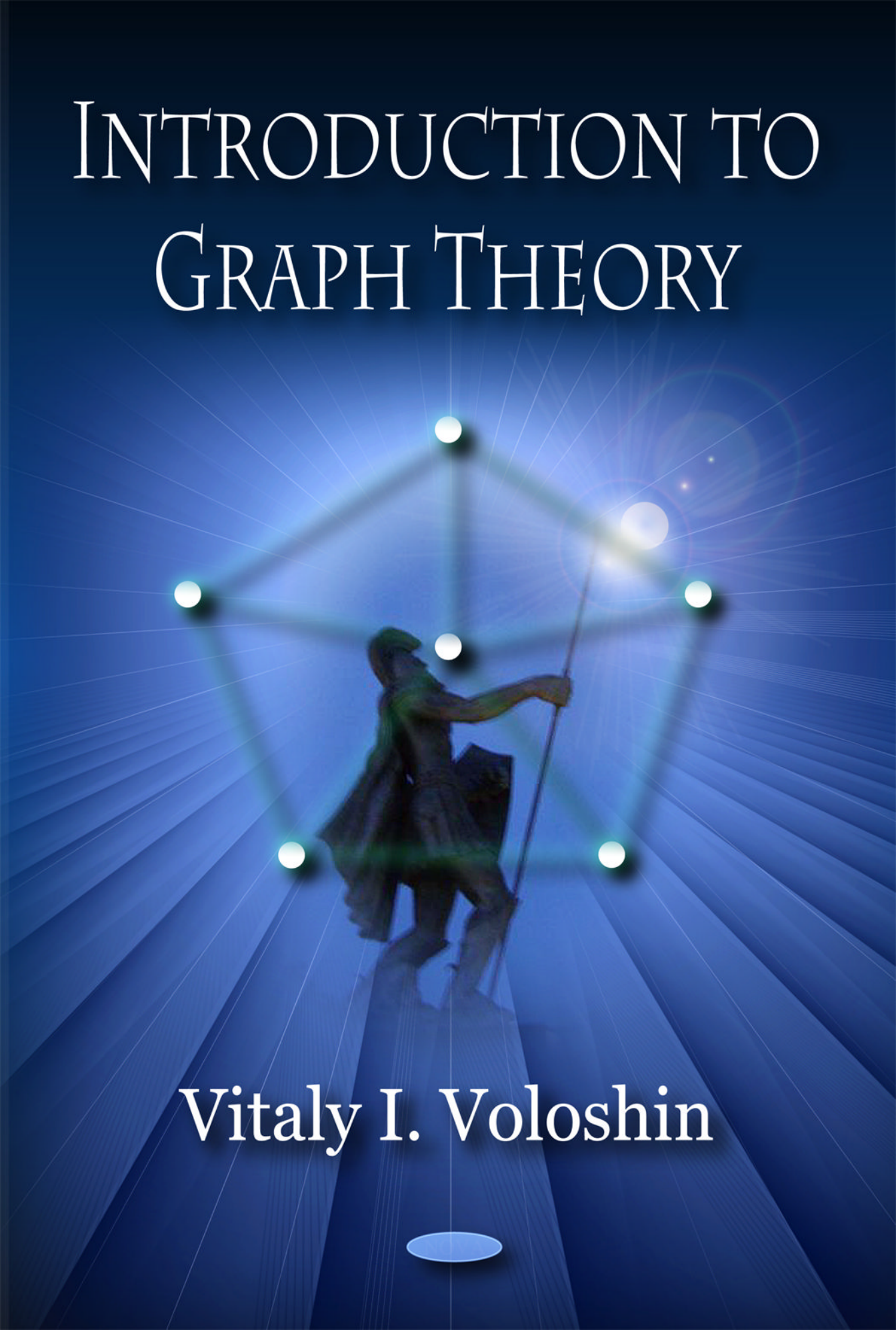
Introduction to Graph Theory



Vitaly I. Voloshin

INTRODUCTION TO GRAPH THEORY

INTRODUCTION TO GRAPH THEORY

VITALY I. VOLOSHIN

Nova Science Publishers, Inc.
New York

Library of Congress Cataloging-in-Publication Data
Voloshin, Vitaly I. (Vitaly Ivanovich), 1954-
Introduction to graph theory / Vitaly I. Voloshin.
p. cm.
ISBN: 978-1-61470-113-2 (eBook)
1. Graph theory. I. Title.
QA166.V65 2009
511'.5–dc22
2008044446

Published by Nova Science Publishers, Inc. ✣ New York

To Julian, Olesea and Georgeta for unlimited love and support

Contents

Preface

Graph Theory is an important area of contemporary mathematics with many applications in computer science, genetics, chemistry, engineering, industry, business and in social sciences. It is a young science invented and developing for solving challenging problems of "computerized" society for which traditional areas of mathematics such as algebra or calculus are powerless.

This book is for math and computer science majors, for students and representatives of many other disciplines (like bioinformatics, for example) taking the courses in graph theory, discrete mathematics, data structures, algorithms. It is also for anyone who wants to understand the basics of graph theory, or just is curious. No previous knowledge in graph theory or any other significant mathematics is required. The very basic facts from set theory, proof techniques and algorithms are sufficient to understand it; but even those are explained in the text.

The book discusses the key concepts of graph theory with emphasis on trees, bipartite graphs, cycles, chordal graphs, planar graphs and graph coloring.

The reader is conducted from the simplest examples, definitions and concepts step by step towards an understanding of a few most fundamental facts in the field. When writing I pursued the following goals:

- to make it as readable as possible;
- to choose the most instructive (not complex!) theorems and algorithms;
- to exhibit sequential generalization of concepts and ideas;
- to show an interaction between the sections and chapters for the sake of integrity;
- clearly expose the essence and core of graph theory.

The book may be used on undergraduate level for one semester introductory course. It includes many examples, figures and algorithms; each section ends with a set of exercises and a set of computer projects. The answers and hints to selected exercises are provided at the end of the book. The material has been tested in class during more than 20-years of teaching experience of the author. Math majors will pay more attention to theorems and proofs, computer science majors will work more with the concepts, algorithms and computations, and representatives of other sciences will find models and ideas for solutions

of optimization problems in their fields.

On the contents, *four core areas of graph theory have been chosen*: bipartite graphs, chordal graphs, planar graphs and graph coloring. The text exhibits the survey of basic results in these areas. Bipartite graphs, planar graphs and graph colorings were the source, the origin of graph theory. Chordal graphs, discovered much later, have a very special place in the entire theory: because of their simplicity and very many nice properties, they are the best playground for introduction to graph theory.

There are several pedagogical methods consistently used throughout the text:

- when formulating a new definition or concept, a formulation and examples of the negation often follow;
- when formulating a new theorem, the cases and examples when it does not work often follow;
- the names are given to many special graphs; they are used instead of drawings thereafter;
- the same examples of graphs are used for many different computational problems; as the opposite, the same problems and algorithms are used for different examples of graphs;
- structurization of more complex proofs is made in order to ease the understanding of a few basic steps;
- detailed proofs of some long theorems are omitted and only ideas or sketch of the proofs are provided;
- contradictory facts or statements are used to call a surprise and make the reading simply interesting;
- an idea is explained first and then the details follow;
- since the comparison is crucial for understanding, opposite versions of some concepts are provided and the respective graphs are compared;
- since the visualization is the feature of graph theory, "look $\Rightarrow$ read $\Rightarrow$ look" - rule is implicitly applied;
- the main goal of exercises is to test understanding of concepts and theorems, and the main goal of computer projects is to train in programming for computations in graph theory;
- **bold type** is used for definitions and paragraph headings, *italic type* is used to call a special attention, and symbol $\square$ is used to indicate the end of proofs.

After all, the ultimate goal of the book is to popularize graphs and their applications.

Acknowledgements: I am grateful to Troy University for repeated support of this project. I also thank the students of Troy University who took the course Introduction to Graph Theory and helped in polishing the text.

Troy University
Troy, Alabama
vvoloshin@troy.edu
November 09, 2008

VITALY I. VOLOSHIN

”...Graph Theory begins with “Graph”

and

ends with “Theory”...

Chapter 1

Basic Definitions and Concepts

"Pictures speak louder than words..."

1.1. Fundamentals

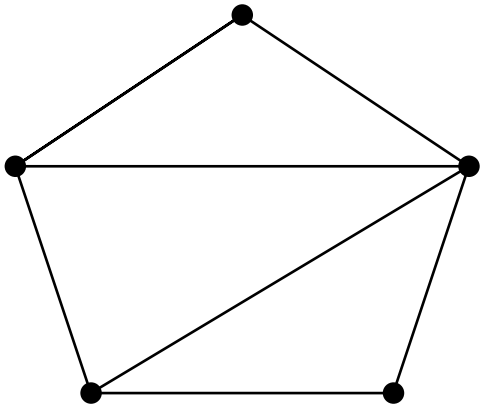


Figure 1.1. This is a graph.

An example of a graph is shown in Figure 1.1. The most simple and least strict **definition of a graph** is the following: a graph is a set of points and lines connecting some pairs of the points. Mathematicians name and number everything: in graph theory, points are called **vertices**, and lines are called **edges**. So, the graph in Figure 1.1 consists of five vertices and seven edges.

Throughout the book, we use the standard notation: upper case letters $A, B, \ldots, X, Y, Z$ for sets (all sets are finite), lower case letters $a, b, \ldots, e, \ldots, x, y, z$ for the elements of a set and curly braces $\{,\}$ for listing the elements of a set. It is convenient to assign indices if we have many elements of the same type. A finite set is a list of its elements; no element is repeated and the order of elements in the list does not matter (we read the lists from left to right). The number of elements in a set A is denoted by $|A|$, and the empty set is denoted

by $\emptyset$. If a set contains other sets as elements, then it is called a **family**; in a family, elements may be repeated but order still does not matter.

In a graph, the set of vertices is denoted by X and is written as $X = \{x_1, x_2, \ldots, x_n\}$ where x_i is the i-th vertex and n is the number of vertices. The set of edges is denoted by E and is written as $E = \{e_1, e_2, \ldots, e_m\}$ where e_i is the i-th edge and m is the number of edges. *Each edge e_i is identified by the pair of respective vertices which are connected by e_i.* It remains to "invent" the last letter to denote the entire graph: G. Now we are ready to present the strict definition of a graph:

Definition 1.1.1 *A* **graph** G *is a set X of vertices together with a set E of edges. It is written as*

$$G = (X, E).$$

Figure 1.2 presents the same graph shown in Figure 1.1 using the definition and agreements above. It has $n = 5$ vertices and $m = 7$ edges. We write $G = (X, E)$ where $X = \{x_1, x_2, x_3, x_4, x_5\}$ and $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$. Since each edge connects a pair of vertices we write $e_1 = \{x_1, x_2\}$, $e_2 = \{x_2, x_3\}$, $e_3 = \{x_3, x_4\}$, $e_4 = \{x_4, x_5\}$, $e_5 = \{x_5, x_1\}$, $e_6 = \{x_2, x_5\}$, $e_7 = \{x_2, x_4\}$, and therefore $E = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_4, x_5\}, \{x_5, x_1\}, \{x_2, x_5\}, \{x_2, x_4\}\}$. Since E is a set of sets it is a family.

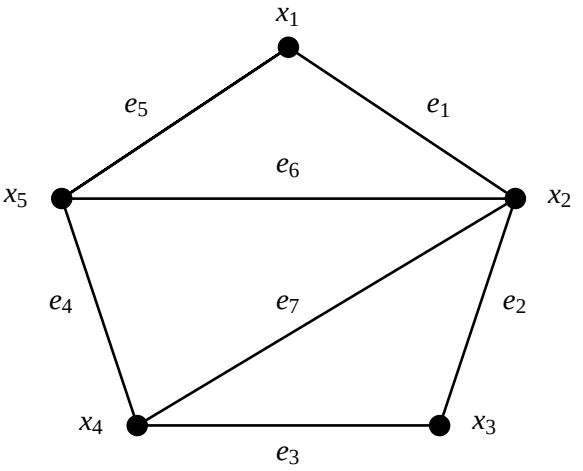


Figure 1.2. Graph $G = (X, E)$.

If two vertices are connected by an edge, then they are called **adjacent**, otherwise they are called **disjoint.** For example, vertices x_1 and x_2 are adjacent, but vertices x_1 and x_3 are disjoint. For a given vertex x, the number of all vertices adjacent to it is called **degree** of the vertex x, denoted by $d(x)$. In our example, $d(x_1) = 2$, $d(x_2) = 4$, and so on. The maximum degree over all vertices is called the **maximum degree of** G, denoted by $\Delta(G)$. For graph G, see Figure 1.2, $\Delta(G) = d(x_2) = 4$.

The adjacent vertices are sometimes called **neighbors** of each other, and all the neighbors of a given vertex x are called the **neighborhood** of x. The neighborhood of x is denoted by $N(x)$. In our graph, for example, $N(x_1) = \{x_2, x_5\}$. Evidently, the degree of a vertex is the cardinality (the number of elements) of its neighborhood: $d(x_1) = |N(x_1)| = 2$, $d(x_2) = |N(x_2)| = 4$, and so on.

For a graph G, if we count the degree of each vertex and arrange these degrees in non decreasing order, then we obtain a sequence called the **degree sequence** of G. The degree sequence for G, see Figure 1.2, is: (2,2,3,3,4).

Two edges are said to be **adjacent** if they have a vertex in common and **disjoint** otherwise. In Figure 1.2 edges e_1 and e_2 are adjacent, and edges e_1 and e_3 are disjoint. If a vertex x belongs to an edge e, then we say that they are **incident** to each other. In the example above, edge e_1 is incident to vertex x_1 and is not incident to vertex x_5 and so on. As one can see, adjacency is referred to the elements of the same type and incidence is referred to the elements of different types.

Sometimes the vertex set of a graph G is denoted by $V(G)$ and the edge set by $E(G)$. So, generally any graph $G = (V(G), E(G))$. The number of vertices is usually denoted by $n = n(G)$, and the number of edges by $m = m(G)$. The **set of edges incident to a vertex** x is denoted by $E(x)$. In our example, $V(G) = X = \{x_1, x_2, x_3, x_4, x_5\}$, $|X| = n = 5$, $E(G) = E = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_4, x_5\}, \{x_5, x_1\}, \{x_2, x_5\}, \{x_2, x_4\}\}$, $|E| = m = 7$, $E(x_1) = \{e_1, e_5\}$, and so on.

Proposition 1.1.1 (Degree equality) *For any graph $G = (X, E)$, the following equality holds:*

$$\Sigma_{i=1}^{n} d(x_i) = 2m. \tag{1.1}$$

Proof. Indeed, if we sum the degrees of all the vertices then each edge is counted twice because it has two ends. □

Applying formula (1.1) to graph G, see Figure 1.2, gives:

$$2 + 4 + 2 + 3 + 3 = 14 = 2 \times 7.$$

Proposition 1.1.2 *In any graph $G = (X, E)$, there are two vertices with the same degree.*

Proof. If a graph G has n vertices, then the degree sequence has n integer numbers. For any vertex x, the minimum number of neighbors is 0, and the maximum number of neighbors is $n-1$, therefore $0 \leq d(x) \leq n-1$.

Assume that all n numbers are different. Then they must take all values on the interval of integers from 0 to $n-1$. 0 means that there is a vertex with no neighbors, and $n-1$ means that there is a vertex adjacent to all the rest of vertices. This cannot occur simultaneously. Therefore, there must be two vertices with the same degree. □

In graph G, see Figure 1.2, for example, $d(x_1) = d(x_3) = 2$, and $d(x_4) = d(x_5) = 3$.

Exercises 1.1.

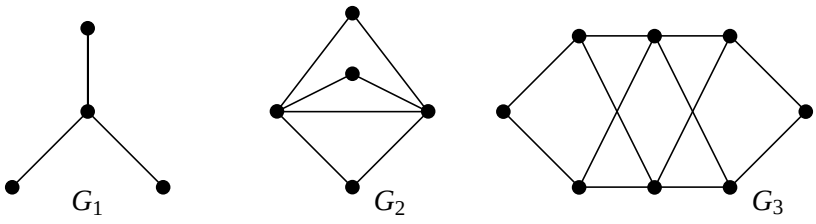


Figure 1.3.

1. For each of the graphs in Figure 1.3, find the number of vertices n and the number of edges m.
2. For each of the graphs in Figure 1.3, name or number the vertices and for each pair of vertices show if they are adjacent or not.
3. For each of the graphs in Figure 1.3, name the edges and for each pair of edges show if they are adjacent or not.
4. In each of the graphs in Figure 1.3, for each pair of vertices and edges show whether they are incident.
5. In each of the graphs in Figure 1.3, find degree and neighborhood of every vertex and degree sequence.
6. Apply degree equality (Proposition 1.1.1) to each of the graphs G_1, G_2 and G_3.
7. In each of the graphs in Figure 1.3, find the vertices of the same degree.
8. For each of the graphs in Figure 1.3, find the maximum degree $\Delta(G)$.

1.2. Graph Modeling Applications

Consider a simple instructive problem. Suppose we have a chemical plant that produces five chemical compounds A, B, C, D, and E which must be stored in storage areas. It is known however, that chemical A combined with chemical B might explode, so they must not be stored in the same storage area. The same occurs if chemical A is combined with E, chemical B is combined with C or E, chemical C is combined with D, and chemical D is combined with E. What is the minimum number of storage areas and how the chemical compounds should be stored to avoid any explosion hazards?

Let us "translate" the problem from the wording above into the language of Graph Theory. Denote chemicals A, B, C, D and E respectively by letters x_1, x_2, x_3, x_4 and x_5 and draw five vertices with the names x_1, x_2, x_3, x_4 and x_5 in the plane (put them, for example, in some imaginary circle clockwise with x_1 on the top). Now draw the edges: read the problem again and connect by an edge every pair of vertices corresponding to the chemicals

which are explosive if combined. Thus, since A and B might explode if combined, connect corresponding vertices x_1 and x_2 with an edge; in the same way, connect x_1 with x_5, connect x_2 with x_3 and x_5, connect x_3 with x_4 and connect x_4 with x_5. The graph obtained is shown in Figure 1.4. It is visually reflecting the relations between the chemicals. Now the problem can be mathematically formulated in the following way: how can we partition the vertex set $X = \{x_1, x_2, x_3, x_4, x_5\}$ into the smallest number of parts, i.e. the subsets, in such a way that no subset contains adjacent vertices? Each such subset will be considered as the subset of chemicals that can be safely stored in the same storage area.

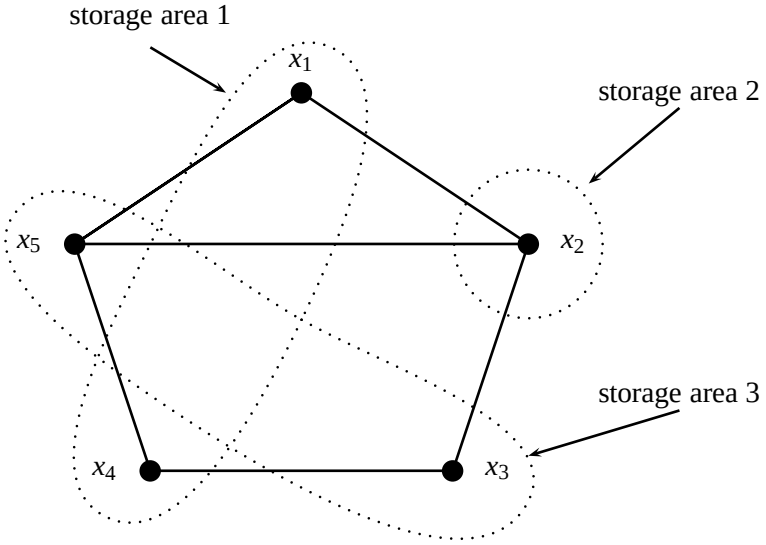


Figure 1.4. Graph $G = (X, E)$.

Looking at Figure 1.4, we can evidently partition X into five subsets: $\{x_1\}$, $\{x_2\}$, $\{x_3\}$, $\{x_4\}$, $\{x_5\}$; this partition is feasible, i.e. good, but not optimal. We can partition X into four subsets: $\{x_1, x_3\}$, $\{x_2\}$, $\{x_4\}$, $\{x_5\}$, what is better but still not optimal. At last, we can partition X into three subsets: $\{x_1, x_4\}$, $\{x_2\}$, $\{x_3, x_5\}$, see Figure 1.4 (the partition is shown by dotted closed curves), and show that it is optimal, i. e. minimal. Indeed, visually one can see that any partition of X into two subsets leaves edges in one of the subsets and therefore is not good. Strict mathematical proof would be the following: consider triangle formed by vertices $x_1, x_2,$ and x_5; any partition of it into two parts (even if one part is empty) leaves an edge in one of the parts. So, two storage areas is not a possible solution, and three storage areas are sufficient and represent the optimal solution of the problem. Looking at the picture, the reader can easily find at least one another optimal solution, say $\{x_1, x_3\}$, $\{x_2, x_4\}$, and $\{x_5\}$. How many and which optimal solutions do exist?

We will see in Chapter 5 and other chapters how such and many other problems which ask not only for the optimal but for all possible solutions, can be efficiently solved. In all applications, like in the example above, visualization is the key feature. Using vertices, edges and their meanings, one apply graphs to depict situations in different sciences. Different problems, first formulated in ordinary language, as in the example above, are then

translated into the language of Graph Theory and solved mathematically. Mathematicians then provide algorithms for finding optimal solutions which are implemented in a software by computer engineers and passed back to the respective businesses for use in the industry.

We next provide a series of situations from different sciences which can successfully be modeled by graphs. *In the next chapters we will consider many mathematical problems which have a respective meaning if applied to graphs that model these and other situations.* In each of the examples below, the reader can easily draw a respective graph with a few vertices and edges keeping in mind their meaning.

Mathematics:

- the vertices are natural numbers from 1 to 100; two vertices are adjacent if the respective numbers have a common divisor different from 1;
- the vertices are intervals on real line; two vertices are adjacent if the respective intervals intersect;
- the vertices are all n-dimensional vectors with binary coordinates (each component is either 0 or 1); two vertices are adjacent if the respective vectors differ in precisely one component.

Computer science:

- vertices are computers in a network; two vertices are adjacent if the respective computers are linked together by telecommunications circuits;
- vertices are processors in parallel architectures; two vertices are adjacent if the respective processors have a direct link;
- vertices are files in a data base; two vertices are adjacent if the respective files cannot be opened simultaneously;
- vertices are all web pages in the world; two vertices are adjacent if the respective web pages are connected by any hypertext link;
- vertices are the image fragments in image segmentation in computer vision; two vertices are adjacent if the respective fragments are related.

Genetics:

- the vertices are fragments of a DNA sequence; two vertices are adjacent if the respective fragments overlap;
- the vertices are species; two vertices are adjacent if the respective species have a common hereditary property.

Chemistry:

- the vertices are atoms in a molecule; two vertices are adjacent if the respective atoms have a bond;

- the vertices are chemical compounds produced by a chemical factory; two vertices are adjacent if the respective compounds are explosive when combined.

Engineering:

- the vertices are junction points of an electric circuit; two vertices are adjacent if the respective junction points are connected by a wire.

Economics:

- the vertices are all companies in the world; two vertices are adjacent if the respective companies are the suppliers for each other.

Healthcare:

- the vertices are drugs; two vertices are adjacent if the combination of the respective drugs is lethal.

Sociology:

- the vertices are employees in a company; two vertices are adjacent if the respective people are in conflict;
- the vertices are the people in a town; two vertices are adjacent if the respective people are friends.

Broadcasting:

- the vertices are radio transmitters in a region; two vertices are adjacent if the respective transmitters interfere.

Geographical maps:

- the vertices are cities; two vertices are adjacent if the respective cities are connected by a highway.

Generally, graphs represent the simplest visual models of systems: any system is a set of elements together with a set of relations between the elements. In the most general setting however, the relations are expressed by statements about any subsets rather than just pairs of elements.

Exercises 1.2.

1. Suppose there are the following intervals on a real line: $[0,3]$, $[4,9]$, $[2.7,5]$, $[5,7]$, $[2,4.3]$, $[1,4]$, $[10,11]$ and $[0,12]$. Draw a graph where vertices represent the intervals and two vertices are adjacent if and only if the respective intervals intersect (the "intersection", or "interval" graph).

2. There are following eight possible sequences of length three consisting of 0 and 1: 000, 001, 010, 011, 100, 101, 110, 111. Draw a graph where the vertices represent the sequences and two vertices are adjacent if and only if the respective sequences differ in precisely one digit. Why is this graph called "cube"?

3. Make the list of all your friends. Draw a graph where the vertices are your friends and two vertices are adjacent if and only if the respective friends are friends themselves (the "friendship" graph).

4. For Florida, Alabama, Georgia, Mississippi, South Carolina, Tennessee, Kentucky, Virginia and California construct a graph where the vertices are these states and two vertices are adjacent if and only if the respective states have a common border.

5. There are four workers A, B, C and D and five jobs 1, 2, 3, 4, and 5. Worker A can do jobs 1 and 2, worker B can do jobs 1, 4 and 5, worker C can do jobs 2, 3, and 4, and worker D can do job 5. Draw a graph where the vertices are the workers and jobs and two vertices are adjacent if and only if they correspond to a worker and a job that the worker can do.

6. Think about five different ways you can drive from home to the school. Draw a graph where the vertices are your home, the school and all street crossings on your way; two vertices are adjacent if and only if the respective crossings, home and the school are consecutive on your way.

7. There are three houses and three wells. Draw a graph where the vertices are houses and wells and connect each house with each well by a curve representing an edge. Is it possible to draw the graph in such a way that the curves do not intersect in the plane at points other than a house or a well?

1.3. Graph Representations

For the last 50 years, Computer Science became a major provider of problems to Graph Theory and, simultaneously, it became a major consumer of the solutions to such problems. In practical applications, graphs involved have not five, and not even ten, or twenty, but hundreds and thousands of vertices and edges. It is not possible for human being to solve any problem with such a huge number of elements by just using the description or even drawing a graph. Computers are used for such tasks, and there are several special ways to store graphs in computer memory.

Adjacency lists. Let us consider graph G shown in Figure 1.5. For every vertex x, form a list of all of its neighbors. The set of all such lists is called the **adjacency list**. In G, the neighbors are:

- For $x_1 : x_2, x_5$,
- For $x_2 : x_1, x_3, x_4, x_5$,
- For $x_3 : x_2, x_4$,

- For $x_4 : x_2, x_3, x_5$,
- For $x_5 : x_1, x_2, x_4$.

The adjacency list, denoted by $L(G)$, is:

$$L(G) = \{\{x_2, x_5\}, \{x_1, x_3, x_4, x_5\}, \{x_2, x_4\},$$
$$\{x_2, x_3, x_5\}, \{x_1, x_2, x_4\}\}.$$

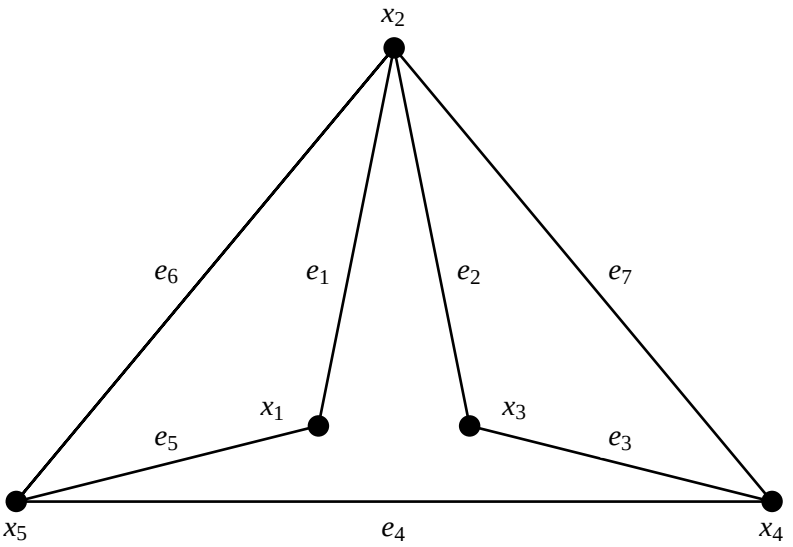


Figure 1.5. Graph $G = (X, E)$.

Adjacency matrix. It is a matrix (rectangular table from letters and/or numbers) which has one row and one column for each vertex. If vertex x_i is adjacent to vertex x_j, then (i, j)-entry (element at the intersection of ith row and jth column) in the matrix is 1, otherwise it is 0. In fact, adjacency matrix is a square matrix.

For graph G, see Figure 1.5, the adjacency matrix denoted by $A(G)$ is:

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

One can compare A with the adjacency lists: in every row, the 1's indicate the respective neighbors from the lists and vice versa.

If we write (from left to right) the rows as columns (1st row as 1st column, 2nd row as 2nd columns and so on), then the columns become rows, and we obtain another matrix, A' which in this particular case is the same as A. Such operation is called **transposition** of the

matrix. Since $A = A'$, this matrix is called **symmetric**. One can think about transposition as of a rotation of the matrix about its diagonal, imaginary line in space connecting upper left and lower right entries.

Incidence matrix. It is a matrix which has one row for each vertex and one column for each edge of a graph. If vertex x_i is incident to edge e_j, then the (i, j)-entry in the matrix is 1, otherwise it is 0. For our graph G, the incidence matrix denoted by $I(G)$ has 5 rows corresponding to the vertices and 7 columns corresponding to the edges:

$$I(G) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

As one can see, every column has precisely two 1's; their rows point out on the vertices which are connected by the respective edge. If we transpose this matrix (write rows sequentially as columns), we obtain another matrix which is different from I.

Edge lists. One can describe graph by giving just the list of all of its edges. For graph G, the edge list, denoted by $J(G)$ is the following:

$$J(G) = \{\{x_1,x_2\},\{x_2,x_3\},\{x_3,x_4\},$$
$$\{x_4,x_5\},\{x_1,x_5\},\{x_2,x_5\},\{x_2,x_4\}\}.$$

Important comment. If we compare Figure 1.2 and Figure 1.5, then we observe that they represent the same graph G; the only difference is in the positions of vertices in the plane. It is the feature of graph theory that the same graph may be drawn in many different ways. Not only the vertices may have different positions, the edges may be drawn as curves connecting the same pairs. As far as the names of vertices and edges remain the same, we accept the agreement that it is the same graph because it has the same mathematical model.

At this moment it is important to see that all three drawings Figure 1.1, Figure 1.2 and Figure 1.5, the description of pair $G = (X,E)$, adjacency list $L(G)$, adjacency matrix $A(G)$, incidence matrix $I(G)$, and edge list $J(G)$, all are different representations of the same concept called graph and denoted by just one letter G. Inverse, having any of these descriptions one can draw a graph and/or construct any other representation as well.

In practice, depending on the problem, some representations are more suitable than the others.

Exercises 1.3.

1. For graphs G_1, G_2 and G_3, see Figure 1.6, construct an adjacency list.
2. For graphs G_1, G_2 and G_3, construct an adjacency matrix.
3. For graphs G_1, G_2 and G_3, construct an incidence matrix.
4. For graphs G_1, G_2 and G_3, construct an edge list.

5. Write down an arbitrary adjacency list, adjacency matrix, incidence matrix, edge list and draw a respective graph.

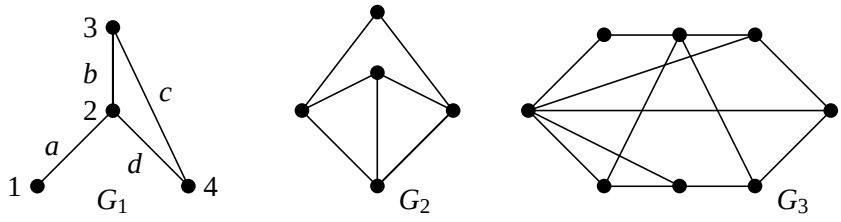


Figure 1.6.

Computer Projects 1.3. Write a program with the following input and output.

1. Given an adjacency list, find the adjacency matrix.
2. Given an adjacency list, find the incidence matrix.
3. Given an adjacency list, find the edge list.
4. Given an adjacency matrix, find the adjacency list.
5. Given an adjacency matrix, find the incidence matrix.
6. Given an adjacency matrix, find the edge list.
7. Given an incidence matrix, find the adjacency list.
8. Given an incidence matrix, find the adjacency matrix.
9. Given an incidence matrix, find the edge list.
10. Given an edge list, find the adjacency list.
11. Given an edge list, find the adjacency matrix.
12. Given an edge list, find the incidence matrix.

1.4. Generalizations

When degenerated cases may occur. In some cases, especially from theoretical view point, it is convenient to consider an edge connecting a vertex to itself. Such edges are called **loops.** It may also happen that some vertex has no neighbors, i.e. its degree is 0. These vertices are said to be **isolated**. Loops and isolated vertices are shown in Figure 1.7.

When repeated edges occur. In some areas of applications, there are many connections between some of the points (for example, in geographic maps), so in graph models, there

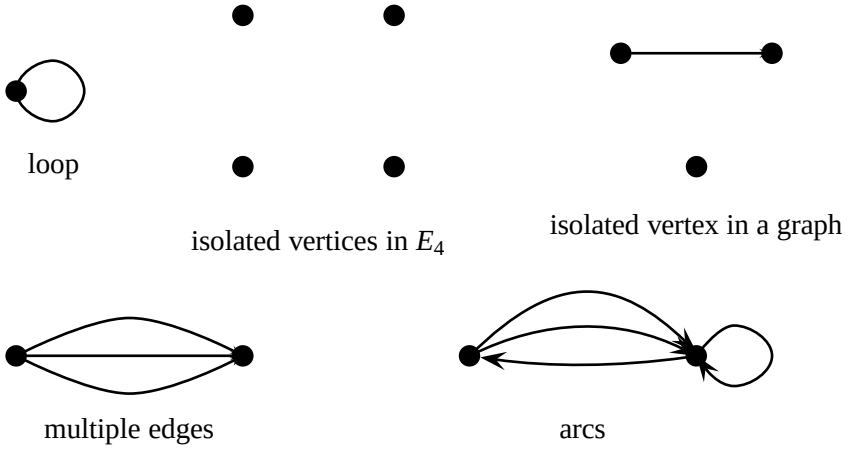


Figure 1.7. Generalizations.

may be many edges connecting the same pair of vertices. Such edges are called **parallel** or **multiple**. The number of repetitions of an edge is its **multiplicity**. Graphs which admit multiple edges are called **multigraphs**. Multiple edges are shown in Figure 1.7.

When direction/order is important. Until now we made no distinction in writing the same sets in different ways. Usually, we read from left to right, and the following two writings for the same set are equivalent: $\{a,b\} = \{b,a\}$. However, there are situations in real life when an order is crucially important. For example, a road (no direction specified) is not the same as a one-way street (direction should be strictly observed). To reflect such situations, one introduce an order which becomes important. In writing ordered sets instead of curly braces $\{,\}$, one use parentheses $(,)$. So now $(a,b) \neq (b,a)$. In a graph, an ordered pair of vertices is called an **arc**. If (x,y) is an arc, then x is called the **initial vertex** and y is called the **terminal** vertex. A graph in which all edges are ordered pairs is called the **directed graph**, or **digraph**. Even loops may be ordered, see Figure 1.7.

Comparing edges and arcs one can accept the following point of view: an edge of a graph is equivalent to two arcs going in opposite directions, see Figure 1.8.

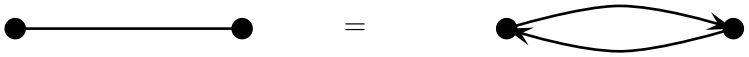


Figure 1.8.

Graphs in which order is not important are called **undirected** graphs. Figure 1.8 implies that directed graphs represent more general structures than undirected graphs. Or, equivalently, the undirected graphs represent a special case of directed graphs, namely, when each edge is replaced by a pair of arcs going in opposite directions. Undirected graphs without loops and multiple edges are called **simple** graphs or simply **graphs**. Usually, unless otherwise stated, one consider simple graphs.

Graph representations can accordingly be adjusted. For undirected multigraphs, adjacency list L may contain multiple elements and empty sets; adjacency matrix A may have

zero rows, zero columns and integer numbers different from 1 (=multiplicity); incidence matrix I may have zero rows and repeated columns; edge list J may have repeated elements. For directed graphs, adjacency list L will not change, adjacency matrix A may not be symmetric (the order may be "row $\rightarrow$ column"), incidence matrix I will have -1' for initial vertices, and edge list J will become ordered.

An example of a directed multigraph G with a loop is shown in Figure 1.9.

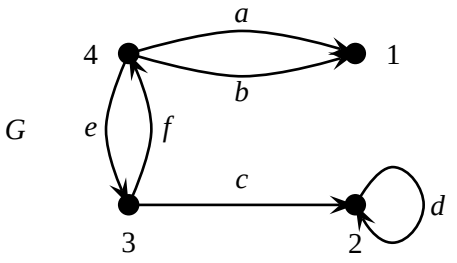


Figure 1.9.

The adjacency list of G is:

$$L(G) = \{\emptyset, \{2\}, \{2,4\}, \{1,1,3\}\};$$

the adjacency matrix A is:

$$A(G) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{pmatrix};$$

the incidence matrix I is:

$$I(G) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & l & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & -1 \\ -1 & -1 & 0 & 0 & -1 & 1 \end{pmatrix};$$

and, at last, the edge list J is:

$$J(G) = \{(4,1),(4,1),(3,2),(2,2),(4,3),(3,4)\}.$$

Notice that the columns in $I(G)$ are in the order of arcs a, b, c, d, e, f, and "l" in fourth column and second row indicates that arc d is the loop at vertex 2.

Some graph modeling examples when order is important are:

- the vertices are street crossings in the city map; crossings x_i and x_j form an arc (x_i, x_j) if the traffic is allowed in the direction from x_i to x_j;
- the vertices are all companies in the world; companies x_i and x_j form an arc (x_i, x_j) if company x_i is a supplier for the company x_j;

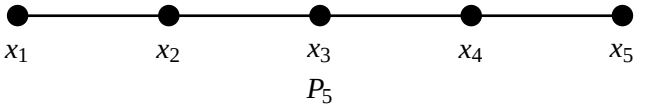
- the vertices are all web pages in the Internet; the web pages x_i and x_j form an arc (x_i, x_j) if there is a hypertext link from page x_i to page x_j;
- the vertices are folders of a folder system in a computer; folders x_i and x_j form an arc (x_i, x_j) if folder x_i contains folder x_j as a subfolder;
- the vertices are states of a discrete system; states x_i and x_j form an arc (x_i, x_j) if the probability of transition from state x_i to state x_j is positive;
- the vertices are all your predecessors; predecessors x_i and x_j form an arc (x_i, x_j) if predecessor x_i is the parent of predecessor x_j;
- the vertices are students in a classroom; students x_i and x_j form an arc (x_i, x_j) if student x_i likes student x_j.

Exercises 1.4. For graphs in Figures 1.3 and 1.6, replace each edge with an arc and construct the adjacency list L, adjacency matrix A, incidence matrix I, and edge list (i.e. arc list) J.

Computer Projects 1.4.
Repeat Computer Projects 1.3. for: a) undirected multigraphs; b) directed multigraphs.

1.5. Basic Graph Classes

Empty graphs. Any graph must have at least one vertex. In other words, we do not accept graphs without vertices. A graph may have no edges at all; such graphs are called **empty**, denoted by E_n where n is the number of vertices. Graph E_4 is depicted in Figure 1.7. If a graph is not empty, then it has at least one edge. Notice that an empty graph is not the same as the empty set which is always denoted by $\emptyset$.



P_5

Figure 1.10. Path P_5.

Paths. A graph in which all vertices can be numbered (ordered from left to right) $x_1, x_2, \ldots, x_n$ in such a way that there is precisely one edge connecting every two consecutive vertices and there are no other edges, is called a **path**. The number of edges in a path is its **length.** A path on n vertices is denoted by P_n. Evidently, in any P_n, the number of edges $m = n - 1$. Any edge itself is a path P_2. The vertices x_1 and x_n both have degree 1 in a path P_n; we say that the path **connects** vertices x_1 and x_n. Generally, any path connecting vertices x and y is called (x,y)-**path.** In Figure 1.10, path P_5 of length 4 connects vertices x_1 and x_5, or is (x_1, x_5)-path.

Connected graphs. A graph is called **connected** if in it any two vertices are connected by some path; otherwise it is called **disconnected.** It means that in a disconnected graph there always exists a pair of vertices having no path connecting them. Any disconnected graph is a union of two or more connected graphs; each such connected graph is then called a **connected component** of the original graph.

Figure 1.11. Connected graph G_1 and disconnected graph G_2.

For example, see Figure 1.11, G_1 is a connected graph, but G_2 is a disconnected graph having two connected components. Any isolated vertex is a connected component. Generally, an empty graph E_n has n components. If each component of a graph G represents the same graph, say G', and if G has k connected components, then we write: $G = kG'$.

Cycles. A connected graph in which every vertex has degree 2 is called a **cycle** (sometimes "simple cycle"). It is denoted by C_n where n is the number of vertices, see Figure 1.12. If n is an even number, then C_n is called **even cycle**. If n is odd, then C_n is **odd cycle.**

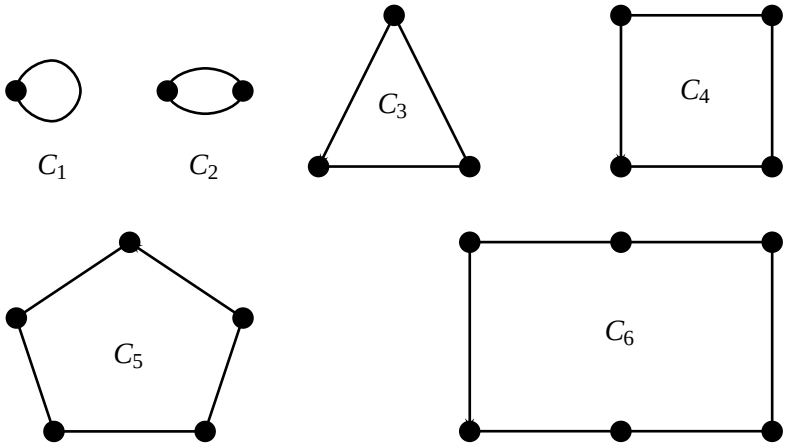


Figure 1.12. Cycles C_1, C_2, C_3, C_4, C_5, and C_6.

In C_n, the number of edges coincides with the number of vertices and it is called the **length** of the cycle. C_1 represents an edge connecting the vertex with itself, it is a loop. C_2 represents two parallel edges connecting the same pair of vertices. The cycle C_3 is called **triangle**. In simple graphs, if there are cycles, then they must have length at least 3. If a graph is not a cycle, then either it has a vertex of degree other than 2, or it is disconnected.

Wheels. If for any cycle $C_k, k \geq 3$, we add a new vertex and connect it to each of the vertices of C_k, then the graph obtained is called a **wheel**, denoted by W_{k+1}. The wheel W_6 is shown in Figure 1.13.

Complete graphs. A graph in which *every* pair of vertices is an edge, is called **com-**

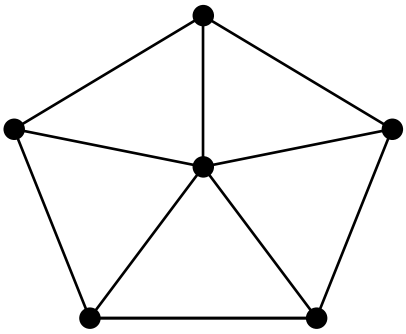


Figure 1.13. The wheel W_6.

plete, denoted by K_n where as usually, n is the number of vertices. It is called complete because we cannot add any new edge to it and obtain a simple graph. For every $n \geq 1$, the degree of each vertex in K_n is $n-1$ and the number of edges is:

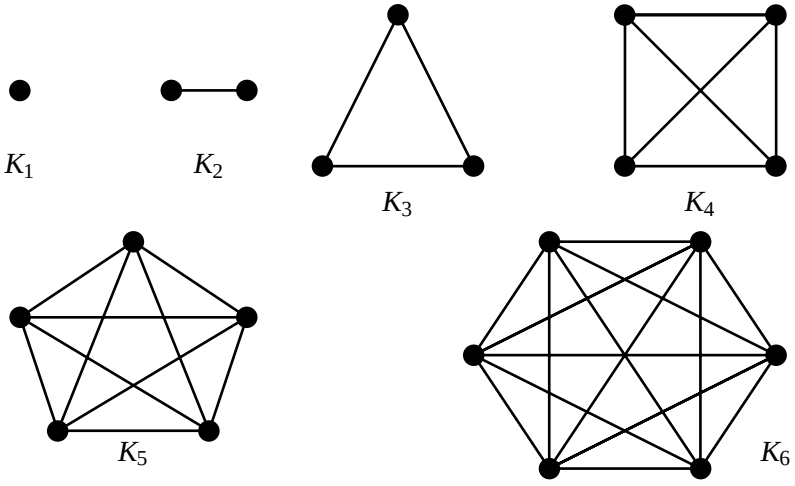


Figure 1.14. Complete graphs K_1, K_2, K_3, K_4, K_5, and K_6.

$$m = \binom{n}{2} = \frac{n(n-1)}{2}.$$

This formula can easily be obtained by counting and adding degrees of every vertex, i.e. applying Proposition 1.1.1: having n vertices of degree $n-1$ each, we obtain the number $n(n-1) = 2m$. For example, the number of edges in K_6 is: $m = 6(6-1)/2 = 15$.

If a graph is not complete, then it has at least two vertices which are not adjacent.

Complete graphs K_1, K_2, K_3, K_4, K_5, and K_6 are shown in Figure 1.14. Notice that $K_2 = P_2$, $K_3 = C_3$, and $K_4 = W_4$.

Trees. A connected graph which has no cycles is called a **tree.** Usually any tree on n vertices is denoted by T_n. In contrast to P_n, K_n, C_n and W_n, there are many distinct trees on n vertices. P_n is a special (simplest) case of a tree. We will have the concept of isomorphism to distinguish graphs. Figure 1.15 shows three examples of trees, among which the 1st and the 3rd "are the same" just because they both are P_4, and the 2nd is "different" from

them. There are no other trees on 4 vertices. In any tree on n vertices, the number of edges $m = n-1$. Disconnected graph without cycles is called a **forest**. Evidently, in a forest every component is a tree.

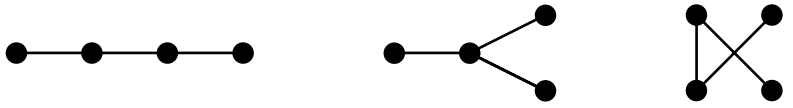


Figure 1.15. Trees.

The last example shows that a graph may be drawn in many different ways; different drawings are like different views of the same object. It is not important how a graph is drawn; it is important which vertices are adjacent and which are not.

As we have seen, not every intersection of edges in a drawing is a vertex. But every vertex lies on the intersection of the respective edges.

Bipartite graphs. A graph $G = (X,E)$ is called **bipartite** if its vertex set X *can be* partitioned into two disjoint sets X_1 and X_2, called parts, in such a way that every edge connects vertices from different sets, see Figure 1.16 . It means that there are no edges inside X_1 and there are no edges inside X_2. In other words, for a bipartite graph, in its drawing, it is possible to color the vertices using just two colors (say blue and red) in such a way that adjacent vertices have different colors. If a graph is not bipartite, then in any partition of its vertex set into two subsets, at least one of the subsets contains at least one edge. Trees and even cycles are bipartite graphs; odd cycles are not. The smallest simple graph which is not bipartite is triangle C_3. Notice that any graph which contains triangle cannot be bipartite. If $G = (X,E)$ is a bipartite graph, it is convenient to write it as $G = (X_1,X_2;E)$ where X_1 is its left and X_2 is its right part (as is the first graph in Figure 1.16).

A **complete bipartite graph** is a bipartite graph in which *every vertex* from part X_1 is adjacent to *every vertex* from part X_2. It is called complete because it is not possible to add a new edge to it and obtain another bipartite graph. If in a complete bipartite graph $|X_1| = r$ and $|X_2| = s$, then the graph itself is denoted by $K_{r,s}$. The number of edges in $K_{r,s}$ clearly equals rs.

Among the examples of bipartite graphs shown in Figure 1.16 (on the top), the first graph G is not complete, the second is $K_{1,3}$, and the third is $K_{2,2} = C_4$. Observe that bipartition X_1,X_2 is explicitly shown for graph G and it is not shown for $K_{1,3}$ and C_4; this fact exhibits the difference between "can be partitioned" (as in definition) and "is partitioned".

Regular graphs. A graph in which every vertex has the same degree k is called **regular of degree** k or k**-regular**. Empty graph E_n is 0-regular, cycle C_n is 2-regular, and K_n is $(n-1)$-regular graph. The 3-regular graphs are called the **cubic graphs**.

It is easy to construct cubic graphs. For example, one can take two cycles of the same length, draw one inside another and connect the respective vertices by edges called "spokes". If we do that with C_3 we obtain a graph called **prism**; if we do that with C_4 we obtain a graph called **cube**. If we do that with C_5 by drawing inner cycle differently, we obtain a famous graph called the **Petersen graph**, see Figure 1.16.

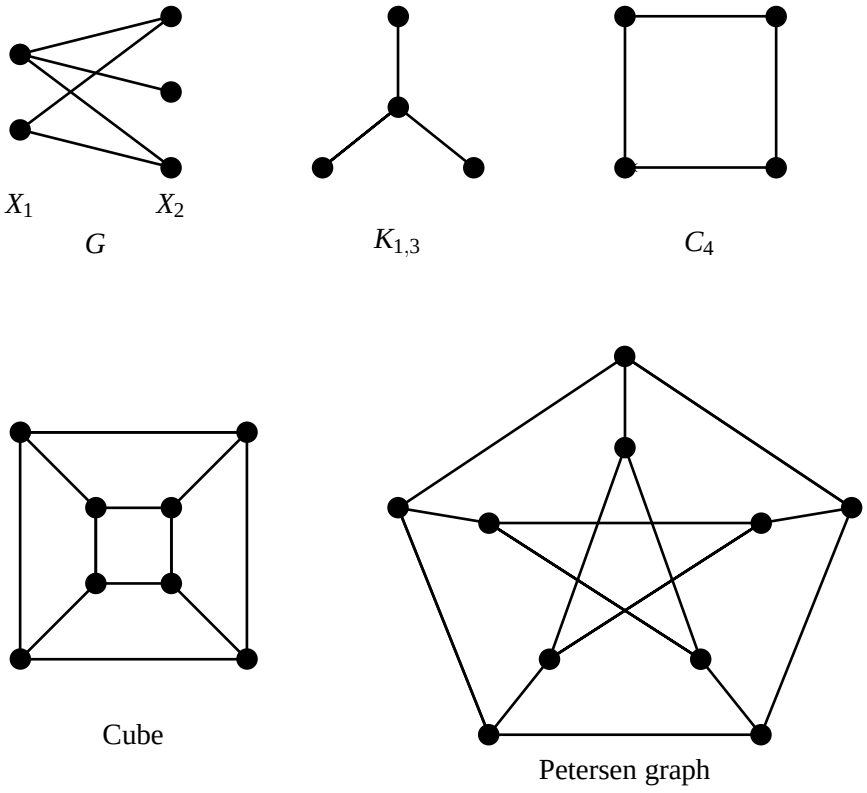


Figure 1.16. Bipartite and regular graphs.

Isomorphic graphs. How can we compare different graphs? Consider graph G_1 in Figure 1.17. Let us re-draw it aside by placing vertex 4 inside triangle formed by vertices 1, 2 and 3, and preserving the edges. We obtain graph G_2. Is G_2 different from G_1, or that is the same graph? Let us re-draw G_1 again by placing the vertices on a square as in G_1 but now replacing the segments of straight lines (representing the edges) by arbitrary curves. We obtain graph G_3. Is G_3 different from G_1, or that is the same graph? On one hand, of course, three graphs are pairwise different because the vertices are different points in the plane. On the other hand, they all have the same adjacency matrix, i.e., the same mathematical model. We can say that graphs G_1, G_2 and G_3 are "essentially the same". But here is an important point: graphs G_1, G_2 and G_3 have the same adjacency matrix because the vertices have the same names; if we choose a different numbering of vertices in any of the graphs, we obtain a different adjacency matrix. In practical applications, the names of vertices are not given at all; so, how to recognize if two graphs are "essentially the same"?

Mathematical definition for two graphs "to be essentially the same graph" is expressed in the concept of "isomorphism" (Greek: "iso" = equal, and "morphe" = shape). Two simple graphs $G_1 = (X_1, E_1)$ and $G_2 = (X_2, E_2)$ are called **isomorphic** if there exists a one-to-one correspondence between vertex sets X_1 and X_2 *such that* any two vertices are adjacent in G_1 if and only if their images in the correspondence are adjacent in G_2. Any such one-to-one

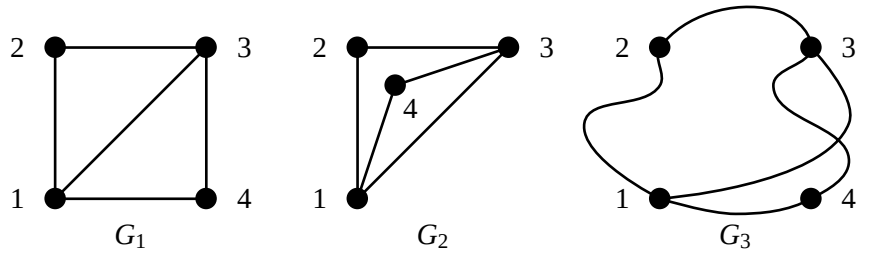


Figure 1.17.

correspondence is called an **isomorphism**. If G_1 and G_2 are isomorphic, then we say that G_1 is isomorphic to G_2, and G_2 is isomorphic to G_1, and we write $G_1 \cong G_2$.

Suppose $|X_1| = |X_2| = n$. Generally, how many one-to-one correspondences do exist? The first vertex of X_1 can be mapped into any of n vertices of X_2, so we have n possibilities for it. Once the first vertex is mapped, the second vertex has $n-1$ possibilities, then the third vertex has $n-2$ possibilities and so on. Total number of possibilities is $n(n-1)(n-2)\cdots 3\cdot 2\cdot 1 = n!$. If in at least one of these $n!$ cases there is a complete "coincidence" of G_1 and G_2, then they are isomorphic. Otherwise they are not. So, two graphs are not isomorphic, if no matter how we map the vertices of one into the vertices of another, there will always be a pair of vertices which are adjacent in one graph and disjoint in another. In such case the graphs are really different because we can never match them.

We can think about the number of isomorphisms to be 0 (when graphs are not isomorphic), and 1, 2, $\ldots n!$ when they are isomorphic. In Figure 1.15, for example, the first and the third trees are isomorphic, they both represent P_4. There are two different choices to map end vertices, then the rest of mapping is determined univocal. So, we can observe that there are two isomorphisms between any two paths P_n. On the other hand, the second tree is not isomorphic to P_4. Graphs E_n and K_n have $n!$ isomorphisms, graphs C_n have $2n$ and so on.

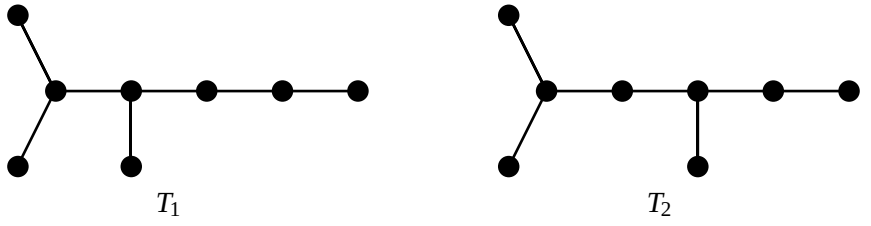


Figure 1.18. Non isomorphic trees.

To be isomorphic, graphs must have the same number of vertices, the same number of edges, the same number of vertices of each degree. However, graphs may have the same number of vertices and edges and even degrees and still not to be isomorphic, see example in Figure 1.18. Both trees T_1 and T_2 have eight vertices, seven edges, four vertices of degree 1, two vertices of degree 2 and two vertices of degree 3. However, in T_1 the vertices of degree 3 are adjacent but in T_2 the vertices of degree 3 are disjoint. This observation and common sense tells us that there will be no match in all 8!=40320 one-to-one correspon-

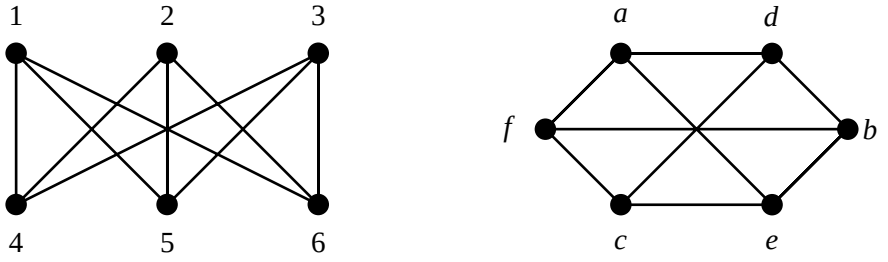


Figure 1.19. Isomorphic graphs.

dences. Generally, to prove that graphs are not isomorphic, instead of considering all $n!$ mappings, it is sufficient to find some property in one of them and show that it is missing in another. To show that graphs are isomorphic, it is sufficient to exhibit that very same one-to-one correspondence from the definition.

Next example, see Figure 1.19, shows two graphs G_1 and G_2 which appear to be isomorphic. In fact, the figure represents two "very different" drawings of the same graph. The mapping (one-to-one correspondence) of the vertices denoted by σ is the following:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ a & b & c & d & e & f \end{pmatrix}$$

One can check manually that every two vertices are adjacent in G_1 if and only if the respective vertices are adjacent in G_2. For example, vertices 1 and 2 are disjoint in G_1, so are the corresponding vertices a and b in G_2. Vertices 1 and 4 are adjacent in G_1, so are the corresponding vertices a and d in G_2, and so on for each pair of the vertices. The number of all such comparisons is $\binom{6}{2} = 15$ which is much less than $6! = 720$, the number of all one-to-one correspondences.

In determining a graph class, one proceed in the following way: first, observe some graph property, then investigate all graphs having that property. There are hundreds of graph classes that have been investigated. In the simplest classes such as E_n, K_n, C_n, W_n, etc, the lower index always shows the number of vertices.

Exercises 1.5.

1. Construct $L(G)$ for each graph G drawn in this section.
2. Construct $A(G)$ for each graph G drawn in this section.
3. Construct $I(G)$ for each graph G drawn in this section.
4. Construct $J(G)$ for each graph G drawn in this section.
5. Find out and explain which graphs in Figure 1.20 are bipartite.
6. For each pair of integer numbers r and s, $1 \leq r, s \leq 5$, draw a complete bipartite graph $K_{r,s}$.
7. Explain why $K_{r,s}$ and $K_{s,r}$ are isomorphic.

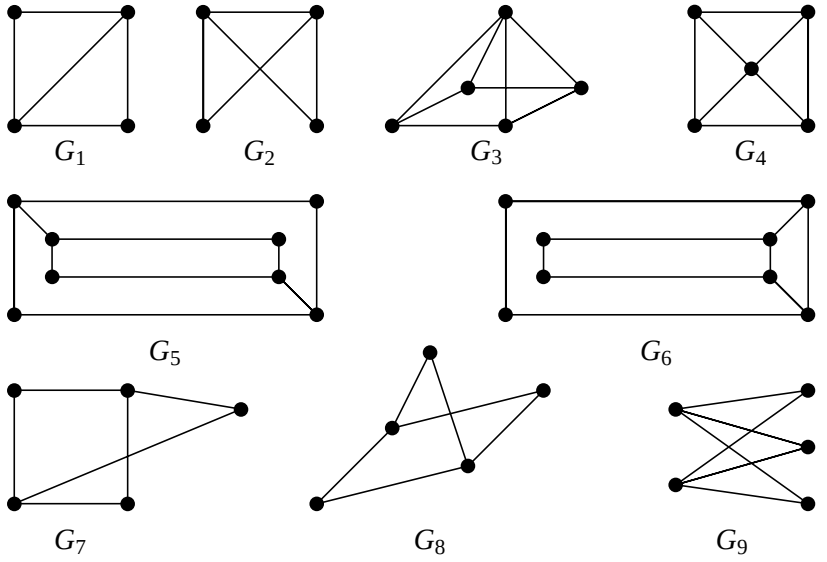


Figure 1.20.

8. Find out and explain which pairs of graphs in Figure 1.20 are isomorphic and which are not.

9. Compare graph representations for two isomorphic graphs G_1 and G_2 in Figure 1.19. When are they identical?

10. Explain when two isomorphic graphs have the same adjacency matrix and the same incidence matrix.

Computer Projects 1.5. Write a program with the following input and output.

1. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a path.
2. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is connected.
3. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a cycle.
4. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a complete graph.
5. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a tree.
6. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a wheel.
7. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a complete bipartite graph.
8. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is a bipartite graph.
9. Given any of $L(G), A(G), I(G), J(G)$, recognize if G is regular.

10. Given any of $L(G_1), A(G_1)$, $I(G_1)$, or $J(G_1)$, any of $L(G_2)$, $A(G_2)$, $I(G_2)$, or $J(G_2)$ and a one-to-one correspondence σ between the vertices of G_1 and G_2. Recognize if σ realizes an isomorphism between G_1 and G_2.

1.6. Basic Graph Operations

In many proofs and algorithms, one often apply graph operations that allow to obtain one graph from another. We now consider some of them.

Deletion of a vertex. Let us have a graph $G = (X, E)$ and a vertex $x \in X$. A **deletion** of x from G is the removing of x from set X and removing from E all edges of G that contain x. Recall that $E(x)$ denotes the set of edges containing vertex x in graph G. If $X_1 = X - \{x\}$, and $E_1 = E - E(x)$, then deletion of x from G results in obtaining the graph $G_1 = (X_1, E_1)$, see Figure 1.21. We write this operation as $G_1 = G - x$. In G_1, we can choose and delete another vertex to obtain a graph G_2 and so on; sequential deletion of vertices results in a sequence of graphs.

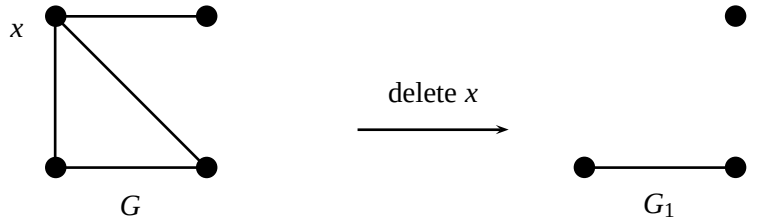


Figure 1.21. Deletion of x from G.

We may want to delete an entire subset of vertices; it is equivalent to a sequential deletion of the respective vertices in any order.

Deletion of an edge. It is the simplest operation of deletion: we just remove an edge from the list of edges. All the rest remains unchanged, see Figure 1.22.

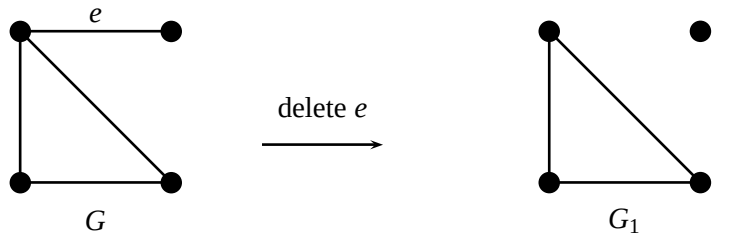


Figure 1.22. Deletion of e from G.

Contraction of an edge. Let us have a graph $G = (X, E)$ and an edge $e = \{x, y\} \in E$. **Contraction of edge** e consists in the following two steps, see Figure 1.23:

1. Identifying vertices x and y in a new vertex called xy and removing e from E;

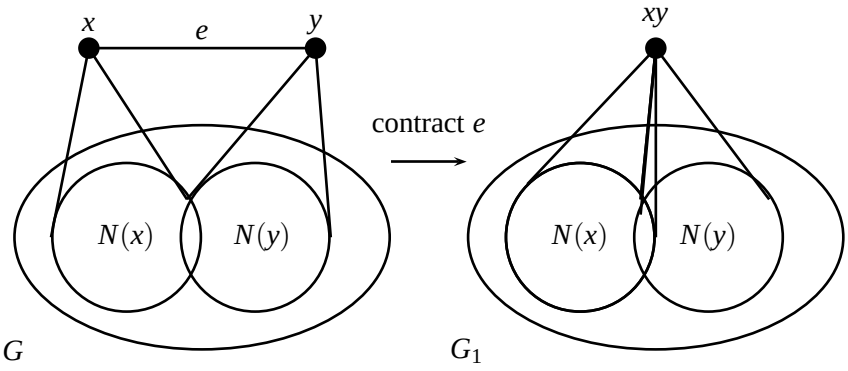


Figure 1.23. Contraction of e in G.

2. All edges of G having one end at x or y will have this end at xy with other end unchanged.

If G is a simple graph and $N(x) \cap N(y) \neq \emptyset$, then each vertex from $N(x) \cap N(y)$ will be connected with vertex xy by two edges, i.e. graph G_1 will have multiple edges. If G is not simple and has loops at x or y, the loops remain in G_1 at vertex xy.

At last, if G is not simple and had another edge e' connecting x and y, then e' becomes a loop at vertex xy. So, contraction of an edge indeed means contraction of that edge up to one point. An example of contraction is shown in Figure 1.24.

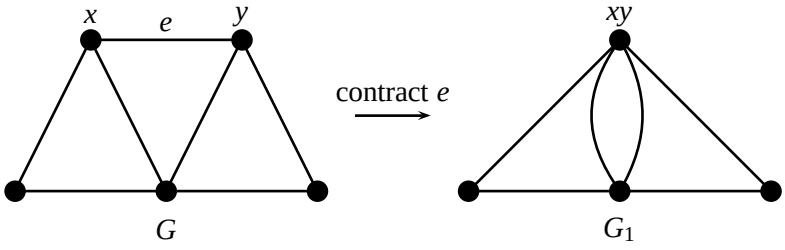


Figure 1.24. $G_1 = G \cdot e$.

A graph G_1 is **contractible** to a graph G_2 if G_2 may be obtained from G_1 by a sequence of contractions of the edges. For example, any $C_k, k \geq 4$ is contractible to $C_3 = K_3$, any tree with at least two vertices is contractible to K_2, any K_n is contractible to K_m if $n \geq m$ (disregard multiple edges), and so on. On the other hand, no C_k can be contracted to K_4, no tree can be contracted to K_3. The maximum value of n such that a graph G is contractible to K_n is called the **Hadwiger number** of the graph G, denoted by $\eta(G)$.

Taking the complement. Consider a simple graph $G = (X, E)$ with $|X| = n$. The **complement of** G denoted by $\overline{G}$ is the graph on the same vertex set X in which two vertices are adjacent if and only if they are disjoint in the original graph G. In other words, $\overline{G} = (X, E')$ where E' is such an edge set that $E \cup E'$ forms the edge set of K_n. This is why $\overline{G}$ is called the complement of G. Evidently, $\overline{\overline{G}} = G$, $\overline{E}_n = K_n$ and $\overline{K}_n = E_n$. Notice that P_4 is isomorphic to $\overline{P_4}$, C_5 is isomorphic to $\overline{C_5}$, and $\overline{K_{r,s}} = \{K_r, K_s\}$. It may happen that G is connected graph but $\overline{G}$ is not, see Figure 1.25.

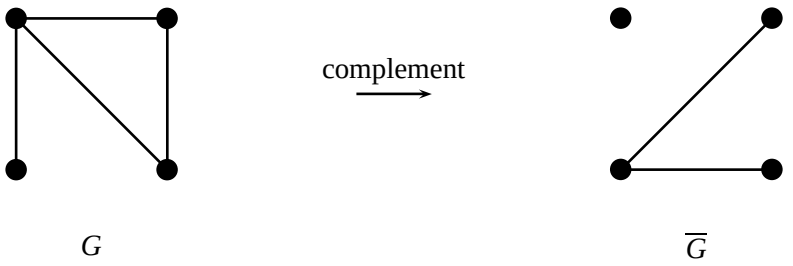


Figure 1.25. G and $\overline{G}$.

Sequential application of operations. Any of the basic operations above (except taking complement) may be applied sequentially many times what results in a sequence of graphs. Since we consider finite graphs, there will always be the last graph in this sequence. We then can consider the sequence of inverse operations and reconstruct the original graph from the last graph. This method is very common in graph theory and is used in proofs by mathematical induction.

Exercises 1.6.

1. Implement sequential (in any order) deletion of vertices of C_4, W_5, K_5, $K_{3,3}$ and Petersen graph.
2. Implement sequential (in any order) deletion of edges of C_4, W_5, K_5, $K_{3,3}$ and Petersen graph.
3. Implement a sequential (in any order) contraction of edges of C_4, W_5, K_5, $K_{3,3}$ and Petersen graph; at each step, remove multiple edges.
4. Find the Hadwiger number of any tree, C_5, W_5, $K_{2,3}$.
5. Construct the complement of C_3, C_4, C_5, C_6, K_5, $K_{3,5}$, Petersen graph, cube, P_7, any tree on 6 vertices, E_5, $2C_3$.
6. Show that $\overline{C}_5 \cong C_5$.
7. Show that $\overline{C}_6$ is isomorphic to a prism.
8. How are the degree sequences of G and $\overline{G}$ related ?

Computer Projects 1.6. Write a program for the following operations.

1. Given any graph representation of a graph, delete a vertex.
2. Given any graph representation of a graph, delete an edge.

3. Given any graph representation of a graph, construct the same representation for the complement.

4. Given any graph representation, implement projects 1-3 with drawing on the screen a graph obtained at each intermediate step.

1.7. Basic Subgraphs

Graphs as combinatorial structures may have many different properties. The properties are very often determined by the presence or absence of some specific substructures called subgraphs. We next consider some of them.

Subgraphs. Let us have a graph $G = (X, E)$. Any graph $G' = (X', E')$ is called a **subgraph** of G if and only if $X' \subseteq X$, and $E' \subseteq E$. In such case, we write $G' \subseteq G$. Since E' contains only the elements of E, both ends of any edge from E' must be in X'. Therefore, G' can be obtained from G by deletion of vertices $X - X'$ (sequentially in any order or at once) and further deletion of remaining edges $E - E'$ (sequentially in any order or at once). In Figure 1.26, both G_1 and G_2 are subgraphs of G: G_1 is obtained by deletion of vertex x_1 and deletion of edge $\{x_3, x_5\}$, and G_2 is obtained by deletion of x_2 and x_4. It is evident that the order in which the vertices and edges are deleted is not important.

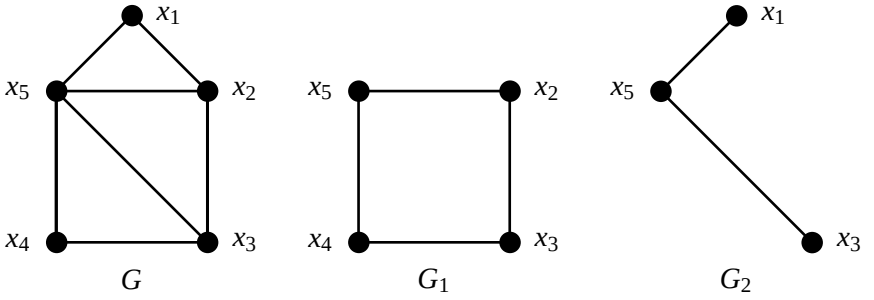


Figure 1.26. Graph G, subgraph G_1 and induced subgraph G_2.

Induced subgraphs. A graph $G' = (X', E')$ is called an **induced subgraph** of a graph $G = (X, E)$ if $X' \subseteq X$ and all edges of G having both ends in X' form edge set E'. Sometimes we say that G' is a subgraph **induced by** X'**.** Induced subgraph G' may be obtained from G by just deletion of vertices $X - X'$ (sequentially in any order or at once). Induced subgraph is a special case of subgraph. A subgraph is not induced if at least one edge of G with both ends in X', is missing. In a graph G, it is convenient to denote the subgraph induced by a set $Y \subseteq X$ by G_Y.

In Figure 1.26, G_2 is an induced subgraph of G but G_1 is not. One can see that G contains three subgraphs isomorphic to K_3, two subgraphs isomorphic to C_4 and one subgraph isomorphic to C_5. Among these subgraphs, only K_3's are induced.

Cycles. In a graph, any subgraph representing a sequence of vertices such that every two consecutive vertices are connected by an edge, and, the first and the last vertices coincide, is called the **cycle.** The number of edges in a cycle is called its **length**. A cycle is called

odd or **even** if its length is respectively odd or even. Cycles may be induced or not; not every induced cycle is isomorphic to the simple cycle C_k for some $k \geq 3$. For example, the sequence of vertices x_2, x_3, x_4, x_5, x_2, see Figure 1.26, forms the cycle of length 4, which is isomorphic to C_4 (represented by graph G_1), however, if we consider an induced subgraph, then we need to add the edge $\{x_3, x_5\}$ to G_1. A cycle which has all the vertices different (except the first and the last) is called **simple**. For example, the cycle $x_1, x_2, x_3, x_4, x_5, x_1$ is simple, but the cycle $x_1, x_2, x_5, x_3, x_4, x_5, x_1$ is not simple because the vertex x_5 is used twice. Though the length of the last cycle is 6, it is not isomorphic to C_6 by the same reason. Notice that any odd cycle, if it is not simple, then it can be split into two cycles one of which is odd and another is even. This implies that if a graph has an odd cycle, then it has a simple odd cycle.

Cliques. Since graphs have vertices, they always contain subgraphs isomorphic to some of K_1, K_2, K_3,..., K_n. They are called **cliques.** If we have a subgraph isomorphic to K_r which is not contained in a subgraph isomorphic to K_{r+1}, then we say that K_r is the **maximal by inclusion complete subgraph**, or, equivalently, the **maximal clique.** Being maximal by inclusion, different cliques may have different size, i.e. the number of vertices. The largest size of a clique among all the cliques of a graph G is called the **clique number of** G denoted by $\omega(G)$. Simply, $\omega(G)$ is the maximum number of pairwise adjacent vertices. For any graph G, $1 \leq \omega(G) \leq n$.

For example, see Figure 1.26, vertices x_2 and x_3 induce K_2 in graph G but it is not a maximal clique since it is not maximal; it is contained in K_3 induced by vertices x_2, x_3, x_5 which is maximal by inclusion. On the other hand, the same vertices x_2 and x_3 in graph G_1 form a maximal clique. We conclude that $\omega(G) = 3$ and $\omega(G_1) = \omega(G_2) = 2$.

Independent sets. Similarly to complete subgraphs, any graph contains subgraphs isomorphic to some of E_1, E_2, ..., E_n. In a graph G, a subset of vertices which induces a subgraph E_k is called the **stable set,** or **independent set.** There are also maximal by inclusion and not maximal by inclusion stable sets. The largest size of a stable in a graph G is called the **stability number**, denoted by $\alpha(G)$. Simply, $\alpha(G)$ is the maximum number of pairwise disjoint vertices. For any graph G, $1 \leq \alpha(G) \leq n$.

Observe that $\omega(G) = \alpha(\overline{G})$ and $\alpha(G) = \omega(\overline{G})$ because when taking the complement every complete subgraph becomes a stable set and vice versa.

For graph G, see Figure 1.26, vertices x_1 and x_3 form a stable set, vertices x_2 and x_4 form another stable set. Both are maximal by inclusion, and $\alpha(G) = 2$.

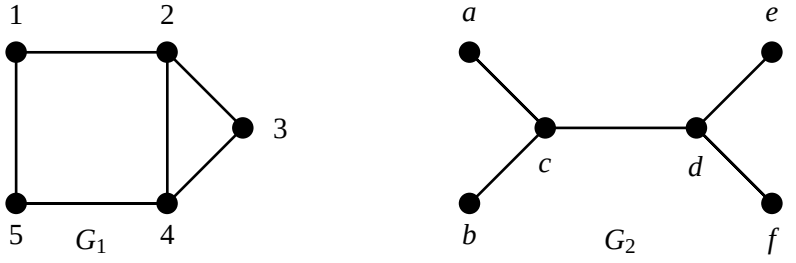


Figure 1.27. Maximal cliques and stable sets.

One more example is shown in Figure 1.27. In G_1, vertices 1 and 2 form a clique, and

vertices 2, 3, and 4 form a clique, $\omega(G_1) = 3$. In G_2, vertices a, b, and d form a maximal by inclusion stable set, and vertices a, b, e, and f also form a maximal by inclusion stable set; the last being maximal is maximum, so $\alpha(G_2) = 4$.

Important comment. In graph theory, given any property, it is common to use term **maximal (minimal)** in the sense of **maximal (minimal) by inclusion** and term **maximum (minimum)** in the sense of **largest (smallest) over all maximal (minimal)**. The difference is similar to the difference between the local and global maximum (minimum) of a function.

Transversals (vertex covers). In a graph $G = (X, E)$, a subset of vertices $T \subseteq X$ is called a **transversal (vertex cover)** if its complement $X \setminus T$ is an independent set. It means that every edge of G has at least one end in T. The minimum cardinality of a transversal of a graph G is called the **transversal number** and denoted by $\tau(G)$. It follows from the definitions above that for any graph G,

$$\alpha(G) + \tau(G) = |X|.$$

Spanning subgraphs. Let us have a graph $G = (X, E)$, $|X| = n$. Any subgraph $G' \subseteq G$ such that $G' = (X, E')$ is called a **spanning subgraph.** Thus spanning subgraphs have the same vertex set as the graph itself. Spanning subgraph which is a tree is called **spanning tree.** If a graph has a spanning tree, then it is connected because any tree is a connected graph.

In Figure 1.27, if we delete edges $\{1,2\}$ and $\{2,3\}$, then we obtain a spanning tree of G_1.

Matchings. In a simple graph G, a subgraph in which every vertex has degree 1, is called a **matching**. Every matching simply represents a collection of edges which have no common vertices, i.e., which are pairwise disjoint. A **perfect matching** is a matching which is a spanning subgraph. The **maximum size of a matching** (over all matchings) is denoted by $\nu(G)$. If a graph G has a perfect matching, then clearly, n is an even number and $\nu = n/2$.

In graph G_2, see Figure 1.27, edge $\{c, d\}$ forms a maximal by inclusion matching. However, a maximum matching is formed for example by edges $\{a, c\}$ and $\{d, e\}$, and $\nu(G_2) = 2$.

Since at least one vertex from every edge of any matching must belong to any transversal, the cardinality of any transversal is at least the cardinality of any matching; i.e, for any graph G,

$$\tau(G) \geq \nu(G).$$

Therefore, if we have a matching and a transversal of the same cardinality, then both are optimal.

Factors. Let G be a simple graph. A k-**factor** of G is a *spanning subgraph* in which every vertex has degree k. In this way, 1-factor represents a perfect matching, 2-factor is a cycle or a collection of disjoint cycles, and so on. Factors not always exist, sometimes there are many distinct k-factors. Complete graphs have the largest number of factors. Examples of factors in a graph G (which is a prism) are shown in Figure 1.28. Regular edges show a 1-factor which is a perfect matching; dashed edges show a 2-factor consisting of two connected components being C_3 each. There is one more 2-factor which is connected and represented by cycle C_6. The graph itself is a 3-factor.

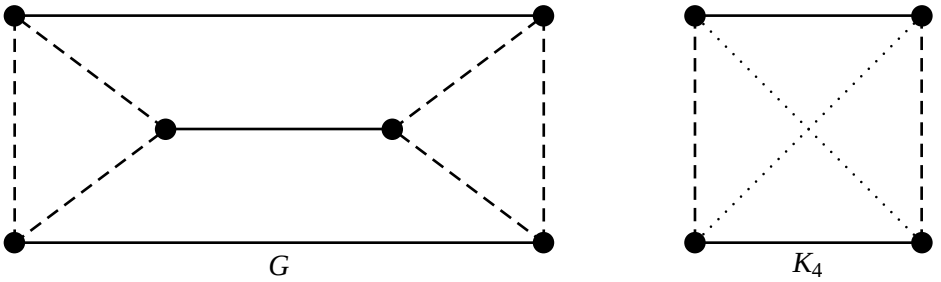


Figure 1.28. Factors.

Partition of the edges of a graph into k-factors is called k-**factorization.** The 1-factorization of K_4 is shown in Figure 1.28.

Graph minors. A graph G' is a **minor** of a graph G, if G' can be obtained from G by a sequence of any vertex deletions, edge deletions and edge contractions.

Exercises 1.7.

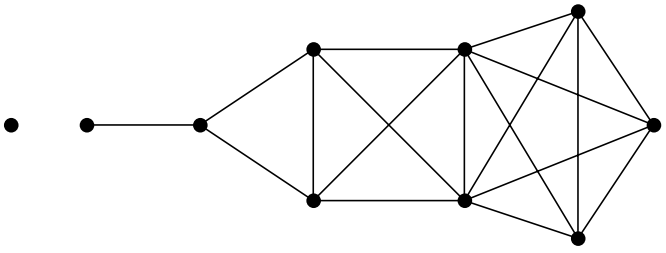


Figure 1.29.

1. For graph in Figure 1.29, write down the list of all cliques, the list of all maximal cliques, and the list of maximum cliques. Begin with all 1-vertex cliques, then all 2-vertex cliques, and so on.

2. Write down the list of all cliques of P_6, C_3, C_4, C_5, W_4, W_5, $K_{2,3}$, cube and Petersen graph. Begin with all 1-vertex cliques, then all 2-vertex cliques, and so on.

3. Write down the list of all maximal cliques of P_6, C_3, C_4, C_5, $W_4, W_5, K_{2,3}$, cube and Petersen graph.

4. Write down the list of all maximum cliques of P_6, C_3, C_4, C_5, W_4, W_5, $K_{2,3}$, cube and Petersen graph.

5. Using the concept of a tree, suggest a way to construct the list of all maximal independent sets in a graph G.

6. Write down the list of all independent sets of $P_6, C_3, C_4, C_5, W_4, W_5, K_{2,3}$, cube and Petersen graph. Begin with all 1-vertex independent sets, then all 2-vertex independent sets, and so on.

7. Write down the list of all maximal independent sets of P_6, C_3, C_4, C_5, W_4, W_5, $K_{2,3}$, cube and Petersen graph.

8. Write down the list of all maximum independent sets of $P_6, C_3, C_4, C_5, W_4, W_5, K_{2,3}$, cube and Petersen graph.

9. Implement exercises 2-6 for the complements of the same graphs.

10. Find the longest path in C_7, cube, W_9, Petersen graph.

11. Find the shortest and the longest cycles in $K_{4,4}$, cube, Petersen graph.

12. Find a spanning tree in cube, $K_{6,9}$, Petersen graph.

13. Find $\omega(G)$ for every graph drawn in this section and in Figure 1.20.

14. Find $\alpha(G)$ for every graph drawn in this section and in Figure 1.20.

15. Find $\tau(G)$ for every graph drawn in this section and in Figure 1.20.

16. Find $\nu(G)$ for every graph drawn in this section and in Figure 1.20.

17. Find α, ω, τ, and ν of cube, Petersen graph, K_{999}, K_{1000}, C_{20}, C_{21}, W_{99}, W_{100}, P_n, K_n, C_n, W_n, for all integers $n \geq 2$.

18. Find all non isomorphic minors of W_5.

Computer Projects 1.7. Using a convenient graph representation, write a program for the following algorithmic problems:

1. Given a graph G, a subset of vertices and a subset of edges. Check if the subsets form a subgraph of G.

2. Given a graph G and a subset of vertices. Output the subgraph induced by the subset.

3. Given a graph G and a subset of vertices. Check if the subset induces a clique.

4. Given a graph G and a subset of vertices. Check if the subset is an independent set of vertices.

5. Given a graph G and a clique. Check if the clique is maximal.

6. Given graph G and an independent set. Check if the set is maximal.

7. Input: Petersen graph. Output: the collection of all maximal independent subsets.

8. Given a graph G and a subset of vertices. Check if the subset is a transversal.

9. Given a graph G and a list of edges. Check if the edges form a matching in G.

10. Given a graph G and a list of edges. Check if the edges form a k-factor in G.

1.8. Separation and Connectivity

Let $G=(X,E)$ be a simple connected graph and $x,y \in X$. If vertices x and y are not adjacent, deleting the set $X-\{x,y\}$ from G leaves only vertices x an y, i.e. a disconnected graph. Any set $S \subseteq X$ of vertices which after deletion from G leaves a disconnected graph is called a **separator** or **vertex cut**. A **subgraph** induced by some separator is also called **separator**. Among connected graphs, only complete graphs do not have separators.

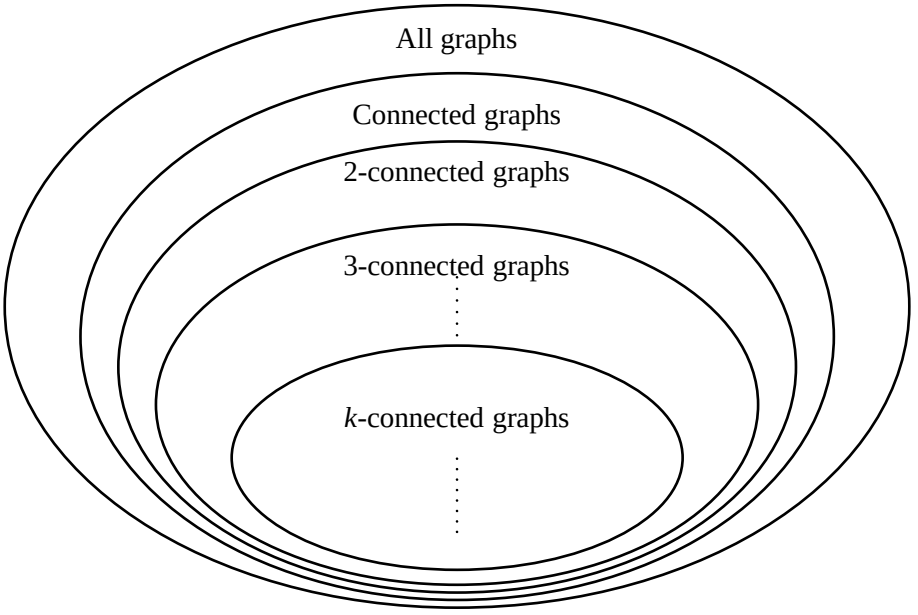


Figure 1.30. Connectivity classes.

For disconnected graphs, a separator is any set of vertices deleting of which increases the number of connected components. So, if S is a separator in G, then there are at least two vertices, say x and y which are separated by S; in this case, S is called (x,y)-**separator.** The meaning of a separator is that all (x,y)-paths pass through it.

Separators may contain other subsets which are also separators. A separator which does not contain any other separator as a proper subset, is called a **minimal separator.** As usually, minimality is meant by inclusion; different minimal separators may have different size. The minimum over all sizes of all minimal separators is called the **connectivity** of G and denoted by $\kappa(G)$. There is only one exception here, namely graph K_n. Since it has no separators, it is convenient to put by definition $\kappa(K_n)=n-1$. So, for any incomplete connected graph G, $\kappa(G)$ is the size of the smallest separator.

There is one more important concept related to separation. A graph G is called k-**connected** if connectivity $\kappa(G) \geq k$. So, all graphs are 0-connected, connected graphs are 1-connected, connected graphs having $\kappa(G) \geq 2$ are 2-connected and so on. Generally, if a graph G is k-connected, then it is $(k-1)$-connected, $(k-2)$-connected, and so on. But this implication is not working in the opposite way: if a graph is k-connected, it may be not $(k+1)$-connected. The inclusion of connectivity classes is shown in Figure 1.30.

Figure 1.31 shows a graph G that has many separators. Set $\{2,3,5,6\}$ is a (1,4)-

separator though it is not a minimal separator. Subsets $\{2,6\}$ and $\{2,5\}$ also represent (1,4)-separators both being minimal. It is seen from the figure that for this graph $\kappa(G) = 2$.

Graph G in Figure 1.31 is 2-connected. Therefore, it is 1-connected and 0-connected. But it is not 3-connected and it is not k-connected for any $k \geq 3$.

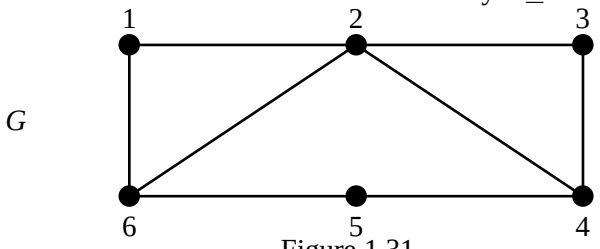


Figure 1.31.

Suppose we delete a separator S from a connected graph G and obtain two connected components induced by vertex sets X_1 and X_2. The two subgraphs induced by $X_1 \cup S$ and $X_2 \cup S$ are called **derived subgraphs** of the graph G with respect to separator S. We will denote them by $G_{X_1 \cup S}$ and $G_{X_2 \cup S}$, or simply by G_1 and G_2 respectively, see Figure 1.32. Notice that both X_1 and X_2 are not empty sets, and none of the vertices from X_1 is adjacent to any of the vertices from X_2.

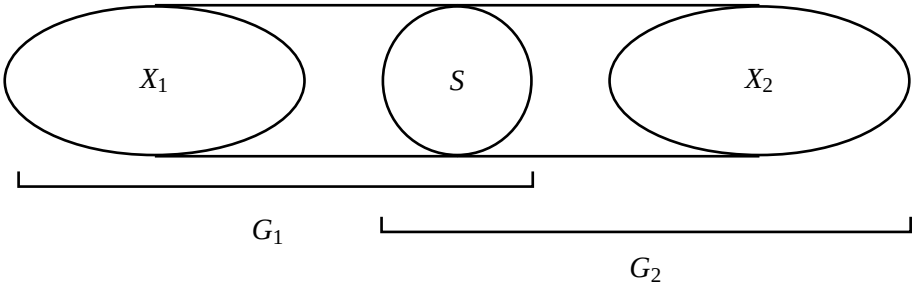


Figure 1.32. Derived subgraphs: general scheme.

Proposition 1.8.1 *In a connected graph G, if S is a minimal separator, then each vertex of S has neighbors in both X_1 and X_2.*

Proof. If some $x \in S$ has no vertex in X_1 adjacent to it, then $S - \{x\}$ is also a separator of G what contradicts the minimality of S. □

For example in Figure 1.33, vertices 3, 5 and 6 in separator $\{2,3,5,6\}$ do not have neighbors in both components because the separator is not minimal; however, vertices 2 and 5 form a minimal separator and both have neighbors in each of two components. The two connected components produced by this separator are induced by vertex sets $\{3,4\}$ and $\{1,6\}$. The two derived subgraphs produced by separator $\{2,5\}$ are induced by vertex sets $\{1,2,5,6\}$ and $\{2,3,4,5\}$. They are shown as G_1 and G_2.

The separation scheme above and Proposition 1.8.1 easily generalize for the case if deletion of S leaves any number $k \geq 2$ connected components induced by $X_1, X_2, \ldots, X_k$. The derived subgraphs then are: $G_1 = G_{X_1 \cup S}$, $G_2 = G_{X_2 \cup S}$, …, and $G_k = G_{X_k \cup S}$.

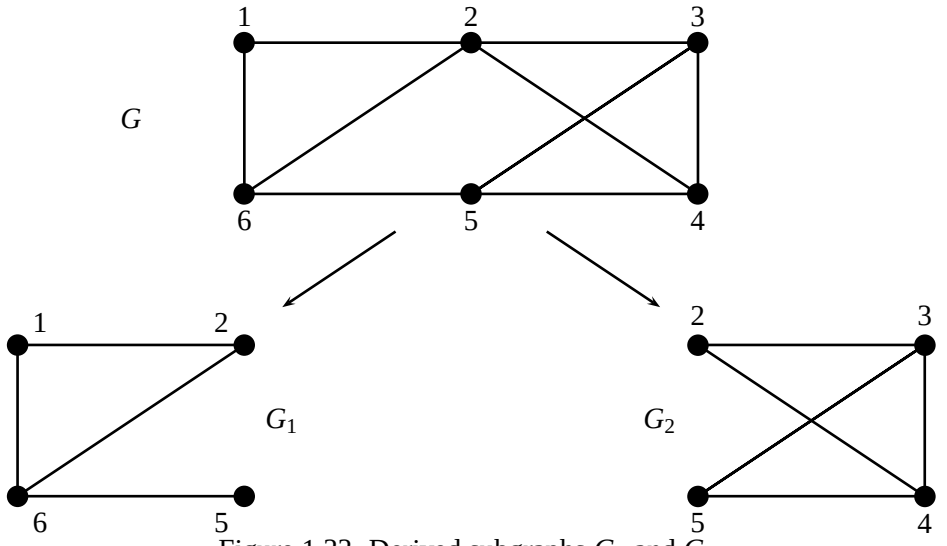


Figure 1.33. Derived subgraphs G_1 and G_2.

Another idea is that one can delete edges from a connected graph and obtain a disconnected graph. Such subsets of edges are called **edge-separators** or **edge-cuts**. If an edge itself is a separator, it is called a **bridge**.

What do we need this for? The answer is that many important properties of graphs can be successfully investigated by using derived subgraphs. Derived subgraphs have less vertices than original graph what opens the way for mathematical induction.

We end the section with formulation of the important Menger's Theorem:

Theorem 1.8.1 (Menger, 1927) *In a connected graph G, for two non adjacent vertices x and y, the minimum number of vertices in an (x,y)-separator equals the maximum number of (internally) vertex disjoint (x,y)-paths.*

Idea behind the proof: to disconnect vertices x and y, one need to destroy every (x,y)-path. □

Exercises 1.8.

1. For every pair of disjoint vertices of graph G, see Figure 1.34, find a separator.
2. For every pair of disjoint vertices of graph G, see Figure 1.34, find a minimal separator.
3. For graph G in Figure 1.34, find the connectivity $\kappa(G)$.
4. For which integer $k \geq 0$, graph G in Figure 1.34 is k-connected?
5. For separator $\{1,3,4,7,8\}$, construct the derived subgraphs.
6. For separator $\{1,8,3,4,6\}$, construct the derived subgraphs.

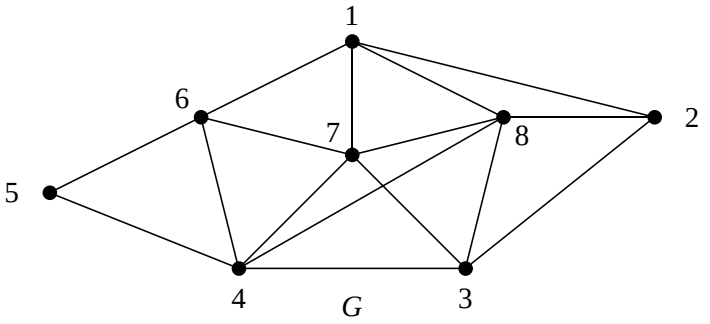


Figure 1.34.

7. For separator $\{5,6,7,8,2\}$, construct the derived subgraphs
8. Apply Proposition 1.8.1 for separator $\{1,8,3\}$.
9. Find an edge-separator for graph G in Figure 1.34.
10. For graph G in Figure 1.34 find a minimal (1,4)-separator.
11. For graph G in Figure 1.34 find a minimal (1,4)-edge-separator.
12. What is the smallest number of edges that disconnect graph G in Figure 1.34?
13. Apply Menger's Theorem to vertices 3 and 6 in graph G, see Figure 1.34.

Computer Projects 1.8. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. For a given subset of vertices in a graph G, find out if the subset is a separator.
2. For a separator of a graph G, determine if the separator is minimal.
3. Find the connectivity of a graph G.
4. For a given subset of edges in a graph G, find out if the subset is an edge-separator.

Chapter 2

Trees and Bipartite Graphs

"– Why are these graphs called "trees"?

2.1. Trees and Cyclomatic Number

Theorem 2.1.1 *For a simple graph G, the following statements are equivalent:*

1. *G is a tree;*
2. *G is connected and $m(G) = n(G) - 1$;*
3. *G has no cycles and $m(G) = n(G) - 1$;*
4. *There is a unique path connecting any two vertices of G;*
5. *G has no cycles and connecting any of its two nonadjacent vertices by an edge results in precisely one cycle.*

Proof. 1. $\Rightarrow$ 2. G is connected by the definition of tree. Prove the equality by induction on n. For $n = 1, 2$ the statement is trivial. Assume $n(G) > 2$. Since G is a tree it has a pendant vertex; delete it and obtain a tree G_1. For G_1 by the induction hypothesis, $m(G_1) = n(G_1) - 1$. Now return deleted vertex with an edge and reconstruct G. Evidently, $m(G) = m(G_1) + 1$, and $n(G) = n(G_1) + 1$, and the implication follows.

2. $\Rightarrow$ 3. Prove that G has no cycles. By contradiction, suppose that G has a cycle. Delete an edge $e = \{x, y\}$ of this cycle; $G - e$ is connected since (x, y)-path remains. If $G - e$ contains cycles, repeat the procedure until graph obtained contains no cycles. Since it is connected, it is a tree, and by 1. $\Rightarrow$ 2. $m = n - 1$, a contradiction to the fact that we deleted at least one edge.

3. $\Rightarrow$ 4. Let G have no cycles and $m = n - 1$. It may have many components. If it has $k > 1$ components $G_1, G_2, \ldots, G_k$, every G_i is a tree. By 1. $\Rightarrow$ 2. $m_i = n_i - 1$ for each G_i. Therefore $m(G) = m_1 + m_2 + \cdots + m_k = (n_1 - 1) + (n_2 - 1) + \cdots + (n_k - 1) = n - k$. So we obtain $m = n - 1 = n - k$ which implies $k = 1$ and G is connected. It means that there is a path between any pair of vertices. If there are two different paths connecting a pair of vertices, then there is a cycle what contradicts to condition 3. Therefore, 4. holds.

4. $\Rightarrow$ 5. G cannot have cycles because in any cycle any two vertices are connected by two different paths. Let x,y be two nonadjacent vertices. Hence there is a unique (x,y)-path. Therefore adding the edge $\{x,y\}$ to G produces a cycle. This cycle is unique because otherwise, if we obtain two cycles C_k and C_l, the subgraph $(C_k \cup C_l) - \{x,y\}$ forms the third cycle, a contradiction to 4.

5. $\Rightarrow$ 1. G has no cycles. If it is not a tree (i.e. it is disconnected), then connect two vertices from different components by an edge. We obtain no cycles, a contradiction. $\square$

Let G be a simple graph having k connected components. The number $\Lambda(G) = m(G) - n(G) + k$ is called the **cyclomatic number** of G. If G is connected, then $k = 1$ and $\Lambda = m - n + 1 = m - (n-1)$. Theorem 2.1.1 in fact states that $n-1$ is the minimum number of edges for a graph to be connected, or, equivalently, to be a tree. Therefore, the number $m-(n-1)$ shows how many extra edges graph G has. Starting from any spanning tree, one can sequentially add remaining Λ edges to reconstruct G. Every such edge forms precisely one cycle with spanning tree. These cycles are called **elementary** and edges are called **chords** with respect to the spanning tree.

If G is disconnected, we apply the same reasoning to each component and replace "tree" with "forest". In other words, the cyclomatic number indicates "how far" graph G is from the forest. That is why it is called the "cyclomatic number".

Indeed, the following holds:

Corollary 2.1.1 $\Lambda(G) = 0$ *if and only if G is a forest.*

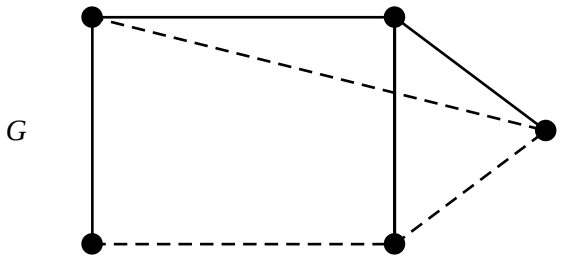


Figure 2.1. Spanning tree, chords and cycles.

An example of a graph and its cyclomatic number is shown in Figure 2.1. The solid edges form a spanning tree, the dashed edges show the chords. Correspondingly, $\Lambda(G) = m - n + 1 = 7 - 5 + 1 = 3$. One can see that every chord forms exactly one elementary cycle with the spanning tree. The total number of cycles however, is greater than Λ. There are additional cycles that can be expressed as combinations of elementary cycles.

Generally, a graph may have many spanning trees, and therefore many different sets of elementary cycles; important however is that any cycle can be expressed as a combination of elementary cycles and the number of elementary cycles is always the same, namely equal to the cyclomatic number.

Let us agree that two spanning trees of K_n are considered different if they are formed by different sets of edges; in fact, some of them may be isomorphic.

Theorem 2.1.2 (Cayley's Formula, 1889) *The number of spanning trees in graph* $K_n, n \geq 1$, *equals* n^{n-2}. $\square$

Since any tree is a connected graph, with the agreement above Cayley's formula implies that there are total n^{n-2} trees for every n. For example, if $n = 3$, then there are three "different" trees which in fact all are isomorphic to each other.

Exercises 2.1.

1. Find the cyclomatic number of E_n, C_n, K_n, W_n, prism, cube and the Petersen graph.
2. Find all spanning trees for graph G in Figure 2.1.
3. Describe the procedure for finding all spanning trees in C_6, K_4, W_6.
4. Describe the procedure for finding all spanning trees in cube and prism.

Computer Projects 2.1. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Generate all spanning trees in prism, cube and the Petersen graph.
2. Generate all spanning trees in K_n, W_n, $n \geq 5$.

2.2. Trees and Distance

Graph E_n is called **trivial**. Unless stated otherwise, we consider nontrivial graphs. It means that any graph has at least two vertices and at least one edge.

Let $G = (X, E)$ be a graph, $x, y \in X$. The **distance** from x to y denoted by $d(x,y)$ is the length of the shortest (x,y)-path. If there is no such path in G, then $d(x,y) = \infty$; evidently, in this case G is disconnected and x and y are in different components. Any segment of a shortest path is a shortest path itself. The distance between x and a set of vertices $Y \subseteq X$ is defined as $d(x,Y) = \min_{y \in Y} d(x,y)$. It is the shortest distance between x and any vertex of Y.

For all $x, y, z \in X$, the following properties of distances hold:

1. $d(x,y) \geq 0$, and $d(x,x) = 0$;
2. $d(x,y) = d(y,x)$;
3. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality).

The **diameter** of G denoted by $diam(G)$ is $\max_{x,y \in X} d(x,y)$; in other words it is the distance between the farthest vertices. For connected graphs, diameter is a positive integer number. An (x,y)-path for which $d(x,y) = diam(G)$ is called a **diametral path.** There may be many diametral paths in a graph. Let $N_\infty(x)$ denote the set of farthest vertices from vertex x. It means that if $y \in N_\infty(x)$ and $z \notin N_\infty(x)$, then $d(x,z) < d(x,y)$. The distance between vertex x and set $N_\infty(x)$ is called the **eccentricity of** x. The **center** of G is a set of vertices of minimum eccentricity. The **radius** of G is equal to eccentricity of any vertex from the center. At last, a vertex of degree 1 is called **pendant vertex** or **leaf.**

Theorem 2.2.1 *Let $x \in X$ be an arbitrary vertex in a tree T. Then every vertex $y \in N_\infty(x)$ is pendant.*

Proof. Let $y \in N_\infty(x)$. Consider (x,y)-path and a vertex z on this path adjacent to y, see Figure 2.2. By contradiction, assume there is another vertex z' adjacent to y. The shortest (x,z')-path cannot use y because it would be longer than $d(x,y)$. Therefore, it must use vertex z or any other vertex from (x,y)-path. In any case we obtain a cycle (at least triangle) what contradicts that T is a tree. □

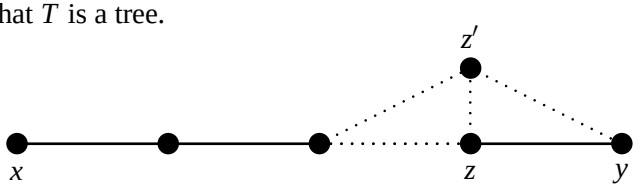


Figure 2.2.

Corollary 2.2.1 *Any tree has at least two pendant vertices.*

Proof. Consider x,y such that $d(x,y) = diam(T)$. Evidently, $x \in N_\infty(y)$ and $y \in N_\infty(x)$, hence by Theorem 2.2.1 both x and y are pendant. □

Theorem 2.2.2 (Jordan, 1869) *The center of a tree is either K_1 or K_2.*

Proof. Notice that deletion of any pendant vertex from a tree T leaves all distances between remaining vertices unchanged. Delete all pendant vertices from T, obtain tree T_1. Since we delete all pendant vertices, by Theorem 2.2.1, the eccentricity of each vertex in T_1 is less than the eccentricity of the same vertex in T by 1. Therefore vertices with minimum eccentricity in T and T_1 are the same. Repeat the procedure as many time as possible. We obtain a sequence of trees $T_1, T_2, T_3, \ldots$, the last being K_2 or K_1 what is the center. □

The theorem explicitly presents an algorithm how to find the center of a tree. One can prove that $diam(T)$ can be found in the following two steps: 1) start at any vertex z and find any $x \in N_\infty(z)$; 2) repeat the procedure for x and find any $y \in N_\infty(x)$: $diam(T) = d(x,y)$.

Exercises 2.2.

1. For the tree in Figure 2.3, find the diameter, radius and center.
2. For each vertex x in the tree in Figure 2.3, find $N_\infty(x)$ and the eccentricity.

Computer Projects 2.2. Using an appropriate graph representation, write a program for the following algorithmic problems:

1. Given any tree, find the diameter.
2. Given any tree, find the center.
3. Given any tree and a vertex x, find $N_\infty(x)$.

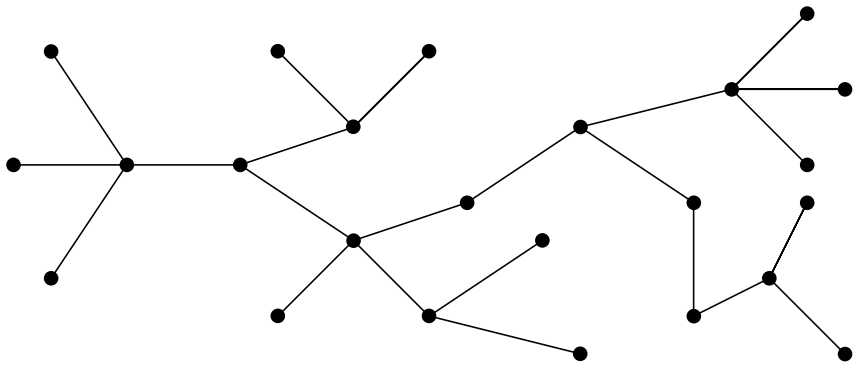


Figure 2.3.

2.3. Minimum Spanning Tree

In this section, we use letters $\mathcal{D}$ for an edge set and D for its elements because we need letter E for another purpose. A graph $G = (X, \mathcal{D})$ is called **weighted** if each edge $D \in \mathcal{D}$ is assigned a positive real number $w(D)$ called the **weight** of edge D. Usually, in many practical applications, the weight represents a distance, time, cost, capacity, resistance, probability, etc. Consider any spanning tree $T = (X, E)$ of G. The **weight** $w(T)$ of tree T is the sum of weights of all edges of T. Different spanning trees may have different weights. The problem of finding a spanning tree of minimum (maximum) weight is called the **minimum (maximum) spanning tree problem**. It is one of the few optimization problems that allows an efficient algorithm for any graph.

Algorithm 2.3.1 *Finding minimum spanning tree (Kruskal's algorithm)*
INPUT: A connected weighted graph G.
OUTPUT: A tree T of minimum weight.

1. *Order the edges of G in increasing (non-decreasing) order of their weights and set T to be an empty graph.*
2. *Add the first edge from the ordering to T.*
3. *Consider the next edge in the ordering. If it produces a cycle in T with already included edges, skip it. Otherwise, include it in T.*
4. *Repeat step 3. until T is connected.*
5. *Output T.*

Theorem 2.3.1 (Kruskal, 1956) *For any connected weighted graph G, Algorithm* 2.3.1 *constructs a spanning tree of minimum weight.*

Proof. First of all, Algorithm produces a tree because it does not create cycles and cannot end when T is disconnected. Since G is connected, some edges between connected components of T exist and at least one must be included in T.

Suppose T is not an optimal spanning tree, i.e. there exist a spanning tree $T^* \neq T$ of minimum weight such that $w(T^*) < w(T)$. Let D be the first edge in the ordering that was chosen for T and is missing in T^*. Adding D to T^* creates a unique cycle. Since T has no cycles, this cycle contains an edge D' which is not in T. Construct the spanning tree $T^* + D - D'$. Now notice that $w(D) \leq w(D')$, and therefore $w(T^* + D - D') = w(T^*) + w(D) - w(D') \leq w(T^*)$. We obtain a new spanning tree of the minimum weight which has one edge more in common with T. Repeating this procedure as many times as necessary, we eventually obtain T what proves that it is an optimal spanning tree. □

Algorithm and Theorem work if we need to find a spanning tree of maximum weight; we just proceed in inverse ordering of the edges.

Exercises 2.3.

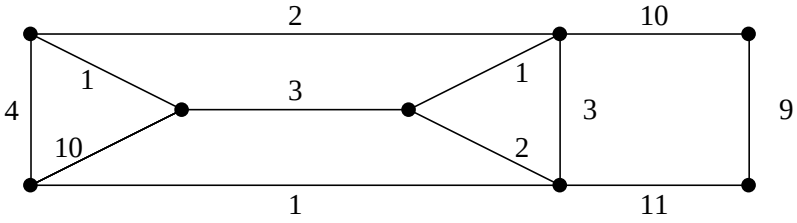


Figure 2.4. Weighted graph G with integer weights.

1. Choose any spanning tree in graph G, see Figure 2.4, and compute its weight.
2. In graph G in Figure 2.4, find the minimum spanning tree in two different ways: a) intuitively; b) by using Kruskal's algorithm.
3. In graph G in Figure 2.4, find the maximum spanning tree in two different ways: a) intuitively; b) by using Kruskal's algorithm.
4. Given weighted complete graph K_n with all weights equal $w > 0$, how many minimum weighted trees does it have?

Computer Projects 2.3. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a weighted graph G, find a spanning tree and compute its weight.
2. Given a weighted graph G and a spanning tree; check if the tree is of maximum (minimum) weight.

3. Given a weighted graph G, implement the Kruskal's algorithm.

4. Given a weighted graph G, find the maximum spanning tree.

2.4. Bipartite Graphs

Theorem 2.4.1 (König, 1936) *A graph G is bipartite if and only if it does not have odd cycles.*

Proof. $\Rightarrow$ Let G be a bipartite graph with parts A and B, and C_k be any cycle in it with a vertex $x \in A$. Traverse C_k starting at x in any direction. Since G is bipartite, each time we alternate parts A and B. Since we end at x, k is even.

$\Leftarrow$ Let $G = (X,E)$ be a connected nontrivial ($n \geq 2$) graph without odd cycles; choose any $x \in X$. Denote $\{x\}$ by N_0, neighborhood $N(x)$ by N_1, vertices at distance 2 from x by N_2, vertices at distance 3 from x by N_3 and so on, The last set in this sequence is $N_\infty(x)$ denoted by N_k. We obtain the following partition of X:

$$X = N_0 \cup N_1 \cup N_2 \cup N_3 \cup \ldots \cup N_k.$$

Observe that $N_i \cap N_j = \emptyset$, $i \neq j$. Moreover, any edge of G connects vertices either from the same N_i or from consecutive sets.

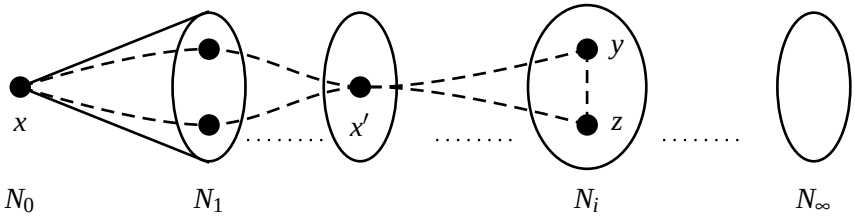


Figure 2.5.

Now form a bipartition of G by placing in A all sets with even indices and placing in B all sets with odd indices, so $X = A \cup B$. It remains to prove that there are no edges of G inside any of sets N_i. By contrary, assume there is an edge $e = \{y,z\}$, such that $y,z \in N_i$ for some $i > 0$, see Figure 2.5. Consider the shortest (x,y)-path and (x,z)-path. They may have common vertices other than x. Recall that every segment of a shortest path is also a shortest path. Let x' be the last common vertex counting from x. Then (x',y)-path and (x',z)-path have the same length, say l, and no common vertices other than x'. Therefore, these paths and edge e form a cycle C_{2l+1}, a contradiction. □

The idea of the proof of König's theorem allows to suggest a simple algorithm which not only gives the possibility to recognize bipartite graphs but is also used for finding solutions of many other problems.

Algorithm 2.4.1 (Breadth-first search) *INPUT: A graph $G = (X,E)$*
OUTPUT: A labeling of vertices of G by the numbers 0,1,2,...

1. *Start at arbitrary unmarked vertex and mark it with 0; set $i = 0$;*

2. *Mark by $i+1$ all vertices which are not marked and adjacent to vertices marked i;*
3. *If there are unmarked vertices adjacent to marked vertices, set $i := i+1$ and go to step 2;*
4. *End.*

Clearly, each marked vertex of G has a mark (label) i which equals the distance to or from the initial vertex. To check whether G is bipartite it is sufficient to add one more step: verify that no two vertices with equal labels are adjacent. If two different vertices with the same label induce an edge, then there is an odd cycle, and by Theorem 2.4.1, G is not a bipartite graph. If G is connected, then Algorithm 2.4.1 marks all the vertices. If G is not connected, then some vertices remain unmarked; one can run it again starting at any unmarked vertex. The number of re-runs will coincide with the number of connected components of G.

To find the distance, i.e. the length of the shortest path between any two vertices of G, it is sufficient to run the algorithm starting at any of these vertices, and the label of the second vertex will be the distance.

If G is a directed graph, then Algorithm 2.4.1 may be used to find the set of vertices that can be reached from a given vertex x of G.

As the opposite to the breadth-first algorithm, there is another way of search in a graph which is called the **depth-first search algorithm**. We next describe the idea of the algorithm.

Assume we need to find a spanning tree in a connected graph G. Choose a vertex and declare it visited. Choose any unvisited vertex adjacent to the last visited vertex, declare it "current" and add connecting edge to the spanning tree. If there are edges connecting current vertex with other visited vertices, declare them as **back edges**. Choose another unvisited vertex adjacent to the last visited vertex and repeat this procedure as long as possible. We get stuck at a vertex which has no unvisited neighbors. At this point, we return back to the vertex from which the current vertex was visited (this step is called **backtracking**) and look for another unvisited neighbor. If there are such, we visit one and repeat the procedure. If there are none, we return one more step back and repeat the procedure again. We add to the spanning tree one edge at a time when visiting a new vertex; the back edges are not added because they form cycles with the edges of the tree. Eventually algorithm stops at the very first visited vertex. That vertex is called the **root** of the spanning tree. As the breadth-first search, the depth-first search algorithm can be applied to solve a number of different search problems.

We now continue the discussion of bipartite graphs. Let $G = (X,E)$ be a graph. The **neighborhood** $N(S)$ **of a subset** $S \subseteq X$ is the union of all neighborhoods of the vertices from S minus S itself, i.e.,

$$N(S) = \cup_{x \in S} N(x) \setminus S.$$

If $G = (X,Y;E)$ is a bipartite graph, then the neighborhood of any subset of X is in Y, and the neighborhood of any subset of Y is in X. We will use the notation $N_G(S)$ to emphasize that the neighborhood is considered in a graph G.

Recall that a matching is a set of pairwise disjoint edges. We say that a **matching covers a set of vertices** if each vertex from the set is incident to an edge of the matching.

Theorem 2.4.2 (Hall, 1935) *A bipartite graph $G = (X,Y;E)$ has a matching that covers X if and only if for every subset $S \subseteq X$,*

$$|N_G(S)| \geq |S|.$$

Proof. $\Rightarrow$ If $G = (X,Y;E)$ has a matching that covers X, then evidently, any subset $S \subseteq X$ has at least $|S|$ neighbors in Y, i.e., $|N_G(S)| \geq |S|$.

$\Leftarrow$ Let $G = (X,Y;E)$ be a bipartite graph. We prove the sufficiency of the theorem by induction on $|X|$. If $|X| = 1$, then by the definition of bipartite graph, the single vertex x of X has at least one neighbor y in Y, thus the edge xy is the required matching.

Let now $|X| > 1$ and the theorem be true for all bipartite graphs with the first part on $< |X|$ vertices. There are two cases to consider.

Case 1: for any subset $S \subset X$, $S \neq X$,

$$|S| < |N_G(S)|. \tag{2.1}$$

Consider an arbitrary edge xy of G. Delete vertices x and y from G and denote the bipartite graph obtained by $G' = (X',Y';E')$. Evidently, $|X'| = |X| - 1 < |X|$. Since precisely one vertex is deleted from Y, the inequality (2.1) implies that in G' for any subset $S' \subseteq X'$, the following holds:

$$|S'| \leq |N_G(S')|.$$

By the induction hypothesis, there exists a matching in G' that covers X'. Add the edge xy to it and obtain a matching that covers X in graph G.

Case 2: there exists a subset $S_0 \subset X$, $S_0 \neq X$, such that

$$|S_0| = |N_G(S_0)|. \tag{2.2}$$

In Figure 2.6, which illustrates this case, the respective sets are shown by ellipses. We now split the vertex set of G into two subsets:

$$A = S_0 \cup N_G(S_0) \text{ and } B = (X \cup Y) \setminus A$$

and consider two induced bipartite subgraphs, G_A and G_B.

In G_A, for any subset $S \subseteq S_0$, we have that $N_{G_A}(S) = N_G(S)$. Therefore, $|S| \leq |N_{G_A}(S)|$. By the induction hypothesis, G_A has a matching that covers S_0.

In G_B, for any subset $S \subseteq X - S_0$, we have that

$$N_{G_B}(S) = N_G(S) \setminus N_G(S_0),$$

see Figure 2.6. Therefore,

$$|S_0| + |S| = |S_0 \cup S| \leq |N_G(S_0 \cup S)| = |N_G(S_0)| + |N_{G_B}(S)|.$$

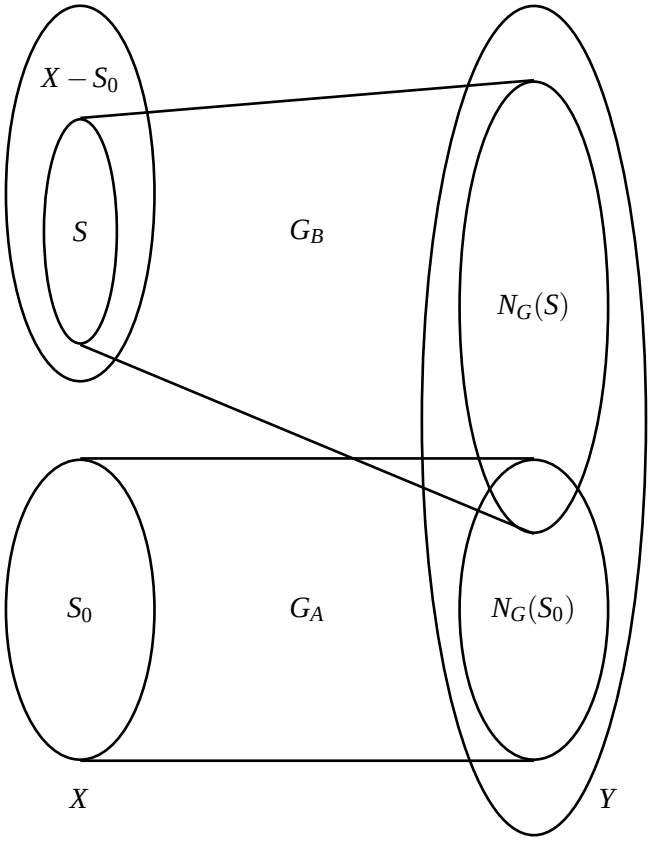


Figure 2.6.

Since $|S_0| = |N_G(S_0)|$ by (2.2), we obtain that $|S| \leq |N_{G_B}(S)|$. Hence by the induction hypothesis, graph G_B has a matching that covers $X - S_0$.

Combining matchings of G_A and G_B we obtain a matching of G that covers X. □

Corollary 2.4.1 *Every regular bipartite graph has a perfect matching.*

Proof. Let $G = (X, Y; E)$ be a k-regular ($k \geq 1$) bipartite graph. Counting the degrees of vertices in X and in Y leads to the equality $k|X| = k|Y|$ what implies $|X| = |Y|$. It means that every matching that covers X, also covers Y.

Let $S \subseteq X$. Since G is k-regular, the number i of edges from S to $N_G(S)$ is $i = k|S|$. Since each vertex from $N_G(S)$ has degree k, $i \leq k|N_G(S)|$; therefore, for any $S \subseteq X$, $|N_G(S)| \geq |S|$. By Theorem 2.4.2, G has a perfect matching. □

Recall that $\tau(G)$ is the cardinality of a minimum transversal of a graph G, i.e. the minimum number of vertices that "touch" all edges. As we mentioned in Section 1.7., for any graph G, $\tau(G) \geq \nu(G)$.

Theorem 2.4.3 (König, 1931) *For any bipartite graph G,*

$$\tau(G) = \nu(G).$$

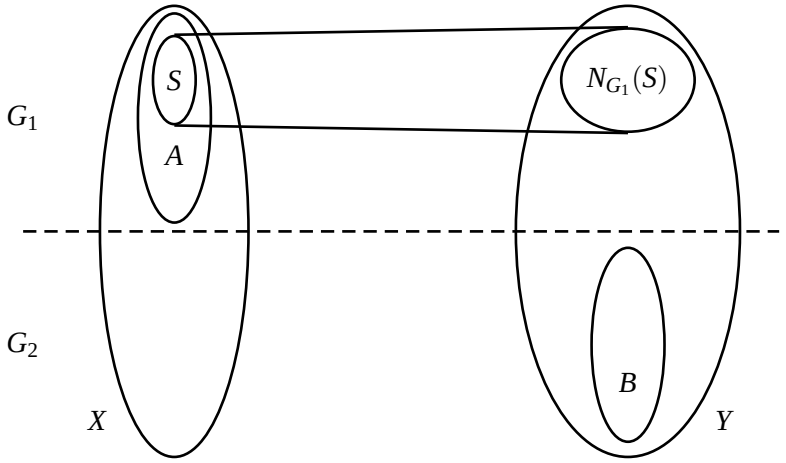


Figure 2.7.

Proof. Let $G = (X,Y;E)$ be a bipartite graph and $T \subseteq (X \cup Y)$ be a minimum transversal of G. We will construct a matching of size $\tau(G) = |T|$ what proves the theorem.

Let $X \cap T = A$ and $Y \cap T = B$, and $G_1 = G_{A\cup(Y\setminus B)}$ and $G_2 = G_{B\cup(X\setminus A)}$, see Figure 2.7.

Since $A \cup B$ is a transversal, there are no edges between $Y \setminus B$ and $X \setminus A$. For each $S \subseteq A$, consider $N_{G_1}(S)$. If $|N_{G_1}(S)| < |S|$, then, because $N_{G_1}(S)$ "touches" all edges incident to S that are not "touched" by B, we could replace S by $N_{G_1}(S)$ and obtain a smaller transversal of G than T. Since T is a minimum transversal, this is impossible, and therefore, $|N_{G_1}(S)| \geq |S|$ for any subset $S \subseteq A$. By Theorem 2.4.2, graph G_1 has a matching that covers A. Applying the same reasoning to graph G_2, we obtain a matching in G_2 that covers B. Since graphs G_1 and G_2 have disjoint vertex sets, we combine these two matchings in one matching of graph G that has $|A| + |B| = |T| = \tau(G)$ edges. Hence $\tau(G) = \nu(G)$. □

Exercises 2.4.

1. Run the breadth-first search algorithm for prism, cube and Petersen graph starting at an arbitrary vertex.
2. For integers $m,n \geq 1$, formulate the conditions when $K_{m,n}$ has a perfect matching.
3. Find all perfect matchings in the cube.
4. For integers $m,n \geq 1$, find $\tau(K_{m,n})$ and $\nu(K_{m,n})$.
5. Run the breadth-first search algorithm for graph in Figure 2.8 to determine if it is bipartite.
6. For graph in Figure 2.8, find the values of τ and ν and the respective transversal and matching.

Computer Projects 2.4. Using an appropriate graph representation, write a program for the following algorithmic problems.

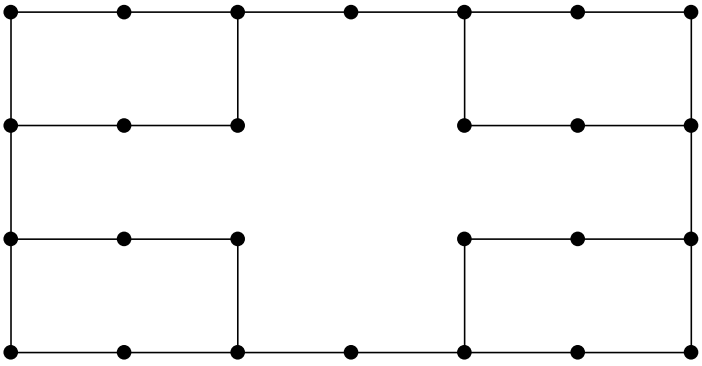


Figure 2.8.

1. Given an arbitrary graph G, run the breadth-first search algorithm to recognize if G is bipartite.
2. Given an arbitrary graph G, run the breadth-first search algorithm to recognize if G is connected.
3. Given an arbitrary graph G, run the breadth-first search algorithm to recognize if G is a tree.
4. Repeat projects 1-3 using the depth-first search algorithm.
5. Given a graph G and a subset of vertices, determine if the subset is a transversal.
6. Given a graph G and a subset of edges, determine if the subset is a matching.

Chapter 3

Chordal Graphs

"Smart people find shorter ways, i.e. the chords in common ways..."

3.1. Preliminary

If a connected graph G is not a tree, then it has cycles. Some cycles may have two non-consecutive vertices which are adjacent in G. The edge connecting them is a **chord**, or a **diagonal** of the cycle. Graph G is called **chordal** if *every* cycle of length ≥ 4 has a chord. Since cycles C_k, $k \geq 4$, have no chords as separate graphs, chordal graphs cannot contain them as induced subgraphs. In other words, if a chordal graph contains C_k, $k \geq 4$, then none of them is induced. Notice that cycles C_1, C_2, and C_3 do not have non-consecutive vertices and therefore cannot have chords; it explains why "chordality" begins with the cycles of length ≥ 4.

A chord splits a cycle into two smaller cycles; if graph is chordal and at least one of the cycles is not a triangle, then it has another chord, and so on. Eventually, every cycle is split into a number of triangles. That is why chordal graphs are also known as "triangulated" or "rigid circuit" graphs.

As follows from the definition, the smallest graph which is not chordal is C_4. Figure 3.1 shows three graphs among which G_1 and G_3 are chordal and G_2 is not. For G_1, one can manually check that all cycles of length ≥ 4 have chords; G_3 is a tree and has no cycle at all; G_2 contains an induced cycle C_4 shown by dashed edges.

If a graph is not chordal, then it contains an induced cycle $C_k, k \geq 4$. Since trees contain no cycles, they all are chordal graphs.

A vertex is called **simplicial** if all of its neighbors are pairwise adjacent, i.e., the neighbors induce a complete subgraph. It follows that a vertex is not simplicial if it has two disjoint neighbors.

In Figure 3.1, simplicial vertices are labeled by "s". We will see that chordal graphs always have simplicial vertices. Moreover, as trees are the special case of chordal graphs, in the same way pendant vertices represent the special case of simplicial vertices.

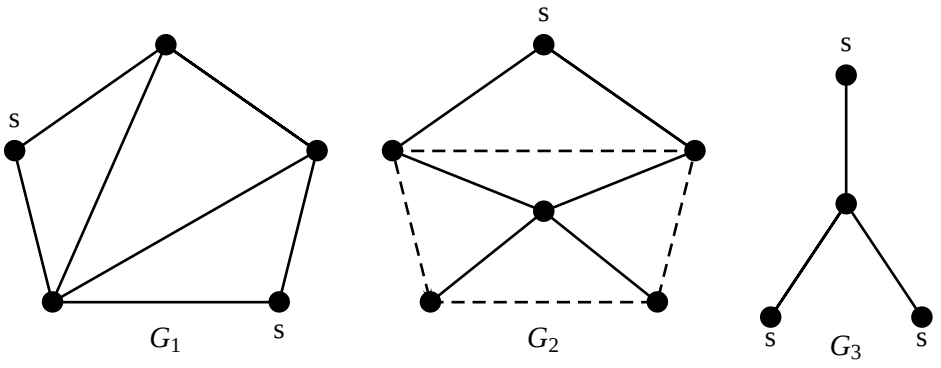


Figure 3.1.

Proposition 3.1.1 *In a chordal graph, every induced subgraph is chordal.*

Proof. If in a chordal graph an induced subgraph is not chordal, then it contains an induced cycle $C_k, k \geq 4$. Evidently, C_k is an induced subgraph in the original graph, what contradicts that it is chordal. □

The above property is important because it allows to prove many results about chordal graphs by induction: if we delete any vertex from a chordal graph, we obtain a chordal graph again.

3.2. Separators and Simplicial Vertices

"Separate and dominate – how old is that?"

Theorem 3.2.1 (Minimal separator theorem) *A graph is chordal if and only if every minimal separator is a clique.*

Proof. $\Rightarrow$ Let $G = (X, E)$ be a chordal graph with a minimal separator S. Assume we have two derived subgraphs $G_1 = G_{X_1 \cup S}$ and $G_2 = G_{X_2 \cup S}$, see Figure 3.2.

We need to prove that G_S is a clique. Consider any pair of vertices $x, y \in S$. Since S is a minimal (by inclusion) separator, each of x, y has neighbors in both X_1 and X_2. Choose a shortest (x, y)-path through X_1; its length is at least 2. Next choose a shortest (x, y)-path through X_2; its length is also at least 2. Combining these two (x, y)-paths we obtain a cycle $C_k, k \geq 4$. Since there are no edges between X_1 and X_2 and G is chordal, vertices x and y must be adjacent. Since x and y are arbitrary vertices of S, all vertices of S are pairwise adjacent, i.e., G_S is a clique.

In the case when there are more than two derived subgraphs, it is sufficient to consider any two of them.

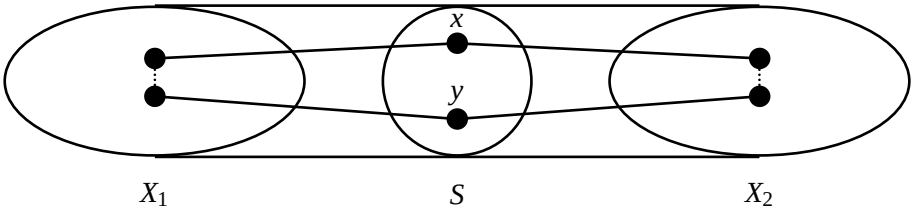


Figure 3.2. Minimal separator in a chordal graph.

$\Leftarrow$ Let $G = (X,E)$ be a graph in which every minimal separator is a clique. By contradiction, assume G is not chordal. Then it contains an induced subgraph C_k, $k \geq 4$. Consider two nonadjacent vertices of C_k, say, x and y. C_k is formed by two different (x,y)-paths. Since $k \geq 4$, each such path contains at least one internal vertex. Since C_k is induced, internal vertices of the first path are not adjacent to the internal vertices of the second path. Now notice that any minimal (x,y)-separator contains at least one internal vertex from each path, and therefore contains two nonadjacent vertices, a contradiction. □

Theorem 3.2.2 (Dirac theorem, 1961) *Every noncomplete connected chordal graph contains at least two simplicial vertices which are not adjacent.*

Proof. Let $G = (X,E)$ be a connected chordal graph which is not a clique; hence $|X| = n \geq 3$. We prove the statement by induction on n. The theorem is evident for $n = 3$. Assume the theorem is true for all noncomplete connected chordal graphs on less than n vertices and prove it for G.

Since G is not a complete graph, there are two vertices, say x and y, which are not adjacent. Hence vertex set $X - \{x,y\}$ is a separator. Choose any minimal separator from this set and denote it by S. By Theorem 3.2.1, G_S is a clique.

Suppose deleting of S from G leaves two connected components, G_{X_1} and G_{X_2}; let $G_1 = G_{X_1 \cup S}$ and $G_2 = G_{X_2 \cup S}$ be two derived subgraphs with respect to S, see Figure 3.2. Graphs G_1 and G_2 both are chordal as subgraphs of G.

Consider subgraph G_1. If it is a clique, then every vertex from X_1 being simplicial in G_1 is simplicial in G because there are no edges between X_1 and X_2. Suppose G_1 is not a clique. It is connected because G_{X_1} is connected and G_S is a clique. It has $< n$ vertices because $X_2 \neq \emptyset$. By the induction hypothesis, G_1 contains at least two simplicial vertices which are not adjacent. Since G_S is a clique, both simplicial vertices cannot be in S. Therefore, one of them is in X_1. Again, since there are no edges between X_1 and X_2, the simplicial vertex from X_1 is a simplicial vertex in G.

Consider subgraph G_2 and apply the same reasoning. We obtain that in G there are two simplicial vertices which are not adjacent.

If deletion of S from G leaves $k \geq 3$ connected components, apply the same reasoning to each component; evidently, in such case G will contain k simplicial vertices which are pairwise not adjacent. □

Observe that the statement of Dirac theorem may be extended to any complete graph K_n, $n \geq 2$, where all vertices are simplicial, with the exception that all they are pairwise adjacent.

Corollary 3.2.1 *Every nontrivial tree contains two pendant vertices.*

Proof. Any tree being a chordal graph by Theorem 3.2.2 must have two simplicial vertices which in this case are pendant. □

Let $G_1 = (X, E)$ be a chordal graph; it contains a simplicial vertex, denote it by x_1. Delete x_1 from G_1 and denote the graph obtained by G_2. Graph G_2 is a subgraph of G_1 and therefore is chordal. It contains a simplicial vertex, denote it by x_2. Delete x_2 from G_2, obtain another chordal graph G_3, which has a simplicial vertex x_3; delete it and continue this procedure on. At every step we obtain a chordal graph and apply Theorem 3.2.2. The last step in this procedure will occur when we delete the last vertex x_n and arrive to $G_{n+1} = \emptyset$. We obtain the ordering of vertices $x_1, x_2, \ldots, x_n$. Denote it by σ, so $\sigma = (x_1, x_2, \ldots, x_n)$. The ordering σ is called a **simplicial elimination ordering** or a *perfect* elimination ordering. The procedure of deleting the vertices in ordering σ is called the **simplicial decomposition**. Its main feature is that every vertex x_i is a simplicial vertex in graph G_i induced by vertices $x_i, x_{i+1}, \ldots, x_n$ and $G_{i+1} = G_i - x_i$, $i = 1, 2, \ldots, n$.

It follows from the observations above that any chordal graph has an ordering σ, or, equivalently, a simplicial decomposition. But if a graph has a simplicial elimination ordering, is it chordal then? Suppose it is not; then it contains an induced $C_k, k \geq 4$. If a simplicial decomposition exists, then sooner or later, the first vertex of C_k must appear in it. However, no vertex of C_k is simplicial in any induced subgraph, a contradiction.

Summarizing the observations above, we arrive to the following conclusion:

Theorem 3.2.3 (Simplicial decomposition theorem) *A graph G is chordal if and only if it has a simplicial elimination ordering.* □

Figure 3.3 shows graph G_1 and its simplicial elimination ordering. Respectively, one can construct a sequence of graphs $G_2, G_3, \ldots, G_8, G_9 = \emptyset$ by sequential deletion of simplicial vertices in order 1, 2, 3,..., 8, so $\sigma = (1, 2, 3, 4, 5, 6, 7, 8)$. Observe that there are only two simplicial vertices in G_1: 1 and 2. However, each of the remaining vertices becomes simplicial at some step.

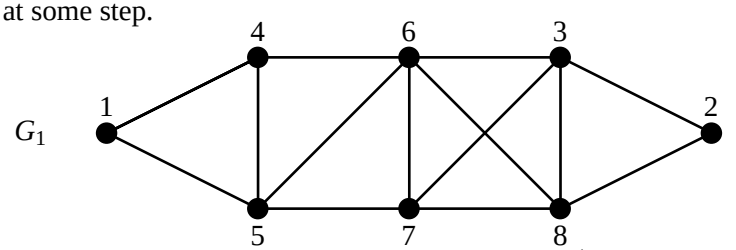


Figure 3.3. Simplicial elimination ordering $\sigma = (1, 2, 3, 4, 5, 6, 7, 8)$.

The power of simplicial elimination ordering is that in determining if a graph is chordal, it is not necessary to investigate all subsets of sizes $4, 5, 6, \ldots$, and check if every subset does not induce C_k; instead, it is sufficient to find at least one simplicial elimination ordering or show that none exist. In other words, searching for a simplicial elimination ordering replaces an exhaustive search for recognition of chordal graphs.

For example, consider graph G, see Figure 3.3. If we use only the definition of chordal graph, then we have to check $\binom{8}{4}$ subsets of size 4 for the case if any induces C_4. Then repeat the procedure for C_5, it would give us $\binom{8}{5}$ subsets. Then we would have to consider

$\binom{8}{6}$ subsets for C_6, $\binom{8}{7}$ subsets for C_8 and $\binom{8}{8}$ subsets for C_8. Visualization of G could save some steps but only in such small examples. However, using the last theorems, instead of that long procedure, we just look for the simplicial elimination ordering.

Non-chordal graphs may have simplicial vertices but if we start simplicial decomposition, sooner or later we get stuck because in obtained graph no vertex is simplicial. In Figure 3.1, graph G_2 is not chordal and simplicial decomposition gets stuck after removing of the unique simplicial vertex.

Theorem 3.2.4 *In a chordal graph, any vertex may be the last vertex in some simplicial elimination ordering.*

Proof. Indeed, let G be a chordal graph and x be any vertex. Since G has at least two simplicial vertices at any step of simplicial decomposition, we can avoid deleting x at every step, except the last one. □

For example, in Figure 3.3 vertex 1 being the first in ordering σ will be the last in ordering $\sigma' = (2,3,8,7,6,5,4,1)$.

In a chordal graph, if any vertex z may terminate some σ, then any vertex y adjacent to z may be the last but one. Then any vertex x forming a triangle with z and y may be the third from the end. If we develop this procedure further, then if graph is chordal, we could reconstruct an ordering inverse to simplicial elimination ordering. The main point is that we can start at any vertex. This idea is at the base of the most efficient algorithm for recognizing chordal graphs, the so called "Maximum Cardinality Search".

Theorem 3.2.5 *If x is a simplicial vertex in a graph $G = (X,E)$, then there exists a maximum independent set $S \subseteq X$ such that $x \in S$.*

Proof. Since x is simplicial, every maximum independent set S' contains precisely one vertex from set $\{x\} \cup N(x)$ because otherwise the set $S' \cup \{x\}$ would be independent and have greater cardinality. If there is a vertex $y \in N(x)$ such that $y \in S'$, then we replace it by x and put $S = S' \setminus \{y\} \cup \{x\}$. □

Notice that the theorem above holds for any graph, not necessarily chordal. However, the existence of a simplicial elimination ordering allows to suggest a simple algorithm for finding the independence number and the respective maximum stable set of a chordal graph:

Algorithm 3.2.1 (Finding maximum stable set of a chordal graph)
INPUT: A chordal graph $G = (X,E)$
OUTPUT: A maximum independent set $S \subseteq X$ with $|S| = \alpha(G)$

1. *Set $S = \emptyset$.*
2. *Find a simplicial vertex x, delete $\{x\} \cup N(x)$ and include x in S;*
3. *If at least one vertex remains, repeat step 2.*
4. *End.*

If we run the algorithm for graph G shown in Figure 3.3, then on the first step, vertex 1 is included in S, and set $\{1,4,5\}$ is deleted; on the second step, vertex 2 is included in S, and set $\{2,3,8\}$ is deleted; at last, on the third step, vertex 6 is included in S, and vertices 6 and 7 are deleted. There are no more vertices left; we end and conclude that one of the maximum stable sets of graph G is $S = \{1,2,6\}$, and $\alpha(G) = 3$. Respectively, vertices $\{3,4,5,7,8\}$ form a minimum transversal and $\tau(G) = 5$. Depending on the simplicial vertices chosen at each step, there are other maximum stable sets and minimum transversals. Notice that Algorithm 3.2.1 also works for graphs which are not chordal.

Further, observe that deleted sets induce cliques; one can prove that they form a minimum number of cliques that **cover** X, namely: the cliques induced by $\{1,4,5\}$, $\{2,3,8\}$ and $\{6,7\}$ have the property that each vertex of G belongs to one of them, and the number of such cliques is the minimum.

In general, the **clique cover number** $\theta(G)$ of a graph G is the minimum number of cliques in graph G such that each vertex belongs to precisely one clique. We say that the cliques **cover** the vertex set of G. As we will see, clique coverings play an important role in graph theory. So, for the graph in Figure 3.3, $\theta(G) = 3$.

Exercises 3.2.

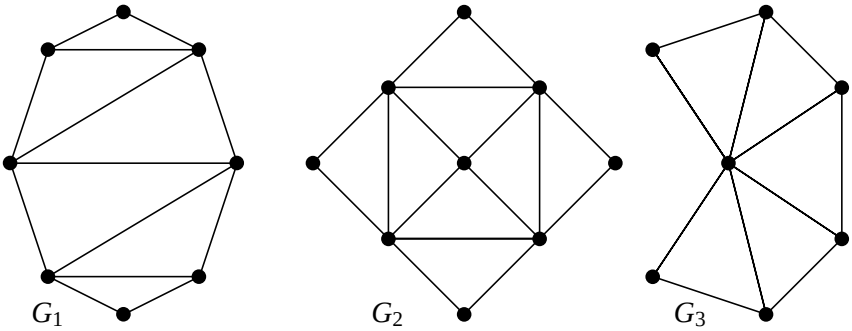


Figure 3.4.

1. In each of the graphs in Figure 3.4 determine which vertices are simplicial and which are not.
2. For each of the graphs in Figure 3.4 determine if it is chordal or not.
3. Which of the graphs in Figure 3.4 has a minimal separator which is not a clique?
4. For chordal graphs in Figure 3.4, find all simplicial elimination orderings.
5. In each chordal graph shown in Figure 3.4, choose a vertex at random and find a simplicial elimination ordering in which the vertex is the last.
6. Run Algorithm 3.2.1 for each of the graphs in Figure 3.4.

7. For each of the graphs in Figure 3.4 find the minimum clique cover; compare the number of cliques with the independence number.

8. What is the minimum number of edges to be added to the cube (prism, W_n, Petersen graph) to make it chordal?

9. What is the minimum number of edges to be deleted from the cube (prism, W_n, Petersen graph) to make it chordal?

Computer Projects 3.2. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a graph G and a vertex x, determine if x is a simplicial vertex.

2. Given a graph G, find out if it contains simplicial vertices.

3. Given a graph G, recognize if it is chordal.

4. Given a chordal graph G, find the maximum independence number and minimum clique cover.

3.3. Degrees

Degree of a vertex is the number of its neighbors; it is a very general term defined for all graphs. The vertices of minimum degree play an important role in many optimization problems. Sometimes one investigate the maximum possible value of minimum degree over all subgraphs in a given graph. When we say "the degree of a vertex in a subgraph" we mean the number of its neighbors in the subgraph only.

It appears that chordal graphs have a special place if we investigate the degrees. For a graph $G = (X, E)$, let us define the following parameter, known as **Szekeres-Wilf number:**

$$M(G) = max_{X' \subseteq X} min_{x \in G'} d(x).$$

As it follows from the definition, to find $M(G)$ one need to consider all induced subgraphs G' of the graph G, count the minimum degree in each of them and find the maximum value over all of them. However, there is a simple procedure for finding $M(G)$. It consists in decomposition of G by sequential elimination of vertices of minimum degree; maximum value t over all these minimums coincides with $M(G)$.

Indeed, on one hand, $M(G)$ cannot be less than t since t is the minimum degree in just one subgraph from the sequence of graphs. On the other hand, if $M(G) > t$, then there is an induced subgraph $G' \subseteq G$ which has minimum degree greater than t. But in the decomposition by minimum degrees, the first vertex of G' appears at some step i, see Figure 3.5; its degree cannot exceed t by definition of t. So, we conclude $M(G) = t$.

Generally, notice that the degree of any vertex in subgraph G' cannot exceed the degree of the same vertex in G_i, and moreover in the original graph G.

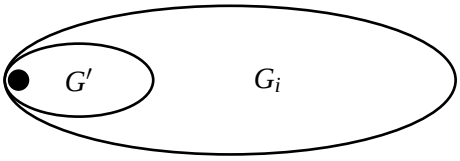


Figure 3.5.

Proposition 3.3.1 *For any graph G, $M(G) \geq \omega(G) - 1$.*

Proof. Indeed, by definition, $\omega(G)$ is the maximum size (number of vertices) of a clique in G. Therefore, there is at least one induced complete subgraph of G with minimum degree $\omega(G) - 1$. Since $M(G)$ is the maximum of minimum degrees over all induced subgraphs, the inequality follows. □

As we see, the value $\omega(G) - 1$ in fact is the lower bound for $M(G)$ for all graphs. The next theorem shows that chordal graphs realize the absolute minimum for $M(G)$:

Theorem 3.3.1 *The following statements are equivalent:*
1) $M(G') = \omega(G') - 1$ *for each induced subgraph* $G' \subseteq G$*;*
2) G *is chordal.*

Proof. 1) $\Rightarrow$ 2) Let $M(G') = \omega(G') - 1$ for each induced subgraph $G' \subseteq G$ and suppose G is not chordal. Then G contains an induced cycle C_k of length ≥ 4. But $M(C_k) = 2 = \omega(C_k)$, a contradiction.

2) $\Rightarrow$ 1) Let G be a chordal graph. Since every subgraph of a chordal graph is also chordal, without loss of generality, prove the equality for G. Note that $M(G) \geq \omega(G) - 1$. As chordal graph, G has a simplicial decomposition. Let the highest degree of a simplicial vertex in such decomposition be t. Then the size of maximum clique in G is $\omega(G) = t + 1$. The simplicial vertex of degree t is not necessarily a vertex of minimum degree, therefore we have $M(G) \leq t = \omega(G) - 1$. Hence $M(G) = \omega(G) - 1$. □

One can see in the conclusion that for chordal graphs, both simplicial decomposition and minimum degree decomposition give the same maximum value of degree equal to $\omega(G) - 1$. In special case, when G is a tree, both simplicial decomposition and minimum degree decomposition coincide. In general case, the situation is much less attractive. For example, for complete bipartite graph $K_{n,n}$ with $n \geq 2$, we obtain $M(K_{n,n}) = n$, $\omega(K_{n,n}) = 2$, and evidently, there are no simplicial vertices.

Exercises 3.3.

1. For each of the graphs in Figure 3.6, find the value of $M(G)$ and compare with $\omega(G) - 1$.

2. Determine which of the graphs in Figure 3.6 is chordal and which is not. For a non chordal graph, find a minimal induced subgraph G' such that $M(G') > \omega(G') - 1$.

3. Determine $M(C_n)$, $M(K_n)$, $M(W_n)$, and the value of Szekeres-Wilf number of the cube, prism and Petersen graph.

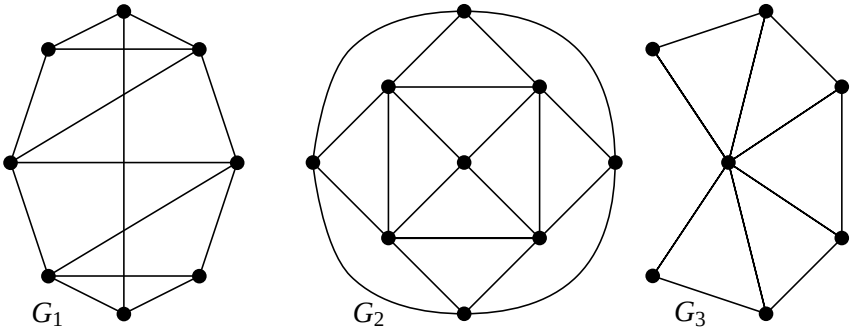


Figure 3.6.

4. Construct a graph G with arbitrarily large difference

$$M(G) - (\omega(G) - 1).$$

5. Prove that for any k-regular graph G, $M(G) = k$.

Computer Projects 3.3. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a graph G, find a vertex of minimum degree.
2. Given a graph G, find $M(G)$.
3. Given a graph G and a subset of vertices; find the minimum vertex degree in the subgraph induced by the subset.

3.4. Distances in Chordal Graphs

Next theorem is a generalization of Theorem 2.2.1:

Theorem 3.4.1 *If G is a chordal graph, then for every vertex x, the set $N_\infty(x)$ of farthest vertices contains a vertex which is simplicial in G.*

Proof. Without loss of generality, consider a connected nontrivial chordal graph $G = (X, E)$. Prove the theorem by induction on n. The statement is evident for $n = 2$. Assume it holds for all chordal graphs on $< n$ vertices, and $|X| = n$.

Let N_i be the set of vertices at distance i from x. We obtain the following partition of X:

$$X = N_0 \cup N_1 \cup N_2 \cup \ldots \cup N_{k-1} \cup N_k$$

where $N_k = N_\infty(x)$, see Figure 3.7.

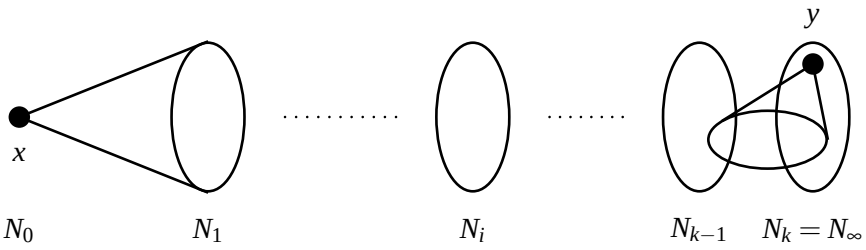


Figure 3.7.

If $k = 1$, then $N_\infty(x) = X - \{x\}$. Chordal graph $G - x$ has a simplicial vertex. Any simplicial vertex in $G - x$ is simplicial in G because x is adjacent to all other vertices. Therefore in this case, $N_\infty(x)$ has a vertex simplicial in G.

Let now $k \geq 2$. The set N_{k-1} evidently is a separator in G. It contains a minimal separator, and subgraph G_{N_k} has at least one connected component. Without loss of generality, suppose N_{k-1} is a minimal separator and, moreover, G_{N_k} has just one connected component.

By Theorem 3.2, $G_{N_{k-1}}$ is a clique. If $G_{N_{k-1} \cup N_k}$ is a clique, then every vertex from N_k is simplicial in G. If $G_{N_{k-1} \cup N_k}$ is not a clique, then as chordal graph, it contains at least two simplicial vertices which are not adjacent. They both cannot be in N_{k-1} because N_{k-1} induces a clique. Hence at least one of them is in N_k. It remains to observe, see Figure 3.7, that every vertex $y \in N_k$ which is simplicial in $G_{N_{k-1} \cup N_k}$, is simplicial in G. □

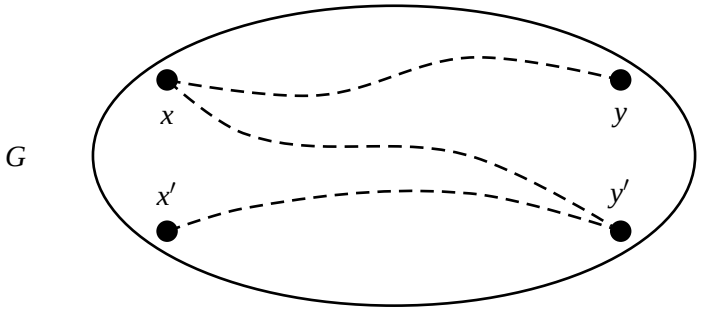


Figure 3.8.

Next corollary generalizes both Theorem 3.2.2 and Corollary 2.2.1.

Corollary 3.4.1 *Any connected nontrivial chordal graph has at least two simplicial vertices.*

Proof. Let $G = (X, E)$ be a connected nontrivial chordal graph. Consider vertices x and y such that $d(x,y) = diam(G)$, see Figure 3.8. By Theorem 3.4.1, there is a vertex $y' \in N_\infty(x)$ which is simplicial in G. Obviously, $d(x,y') = d(x,y) = diam(G)$. By the same reason, for vertex y', there is a simplicial vertex $x' \in N_\infty(y')$ such that $d(y',x') = d(y',x) = diam(G)$. □

Next example (Figure 3.9) shows how it looks in a graph G. It is easy to see that $diam(G) = 4$. Let us start at vertex 2. So, $N_0 = \{2\}$. Then $N_1(2) = \{1, 12, 10, 3\}$. Next we

find $N_2(2) = \{11,4,8,9\}$. At last, $N_3(2) = N_\infty(2) = \{5,6,7\}$. Observe that vertices 5 and 7 are simplicial.

We now run the same procedure starting at, say, 5. $N_0(5) = \{5\}$, $N_1(5) = \{4,6\}$, $N_2(5) = \{7,8,9,3\}$, $N_3(5) = \{2,10\}$, $N_4(5) = N_\infty(5) = \{1,12,11\}$. We find two simplicial vertices, 11 and 5 such that $d(11,5) = diam(G) = 4$. There are two diametral (11,5)-paths: 11-10-3-4-5 and 11-10-9-4-5.

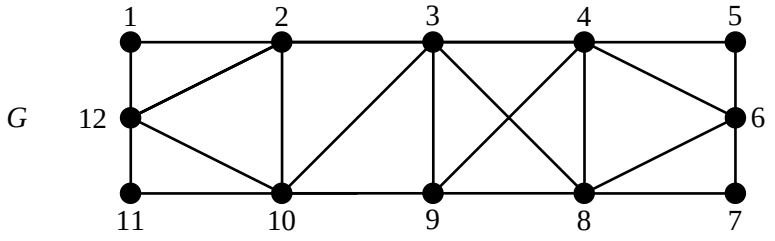


Figure 3.9.

Notice that simplicial decomposition of a chordal graph does not change any distance in graphs obtained at each step; this is true even if graph is not chordal but has simplicial vertices.

Last example shows that in chordal graphs, $N_\infty(x)$ may contain vertices which are not simplicial. In Figure 3.9, $N_3(2) = N_\infty(2) = \{5,6,7\}$ where vertex 6 is not simplicial. Therefore Theorem 2.2.2 cannot be generalized to chordal graphs. There are other methods for finding the center in chordal graphs.

Exercises 3.4.

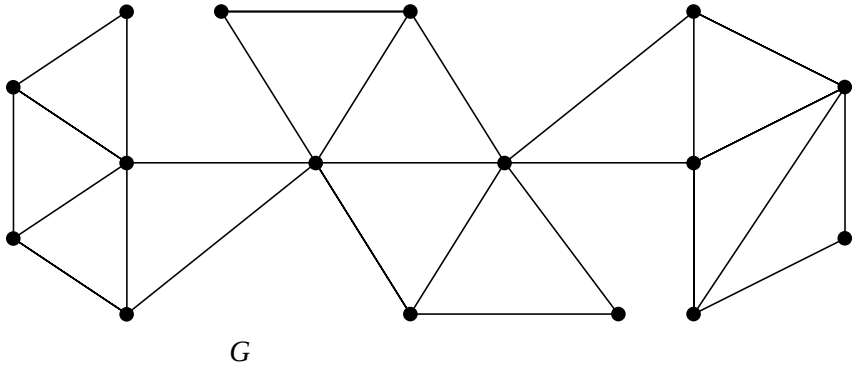


Figure 3.10.

1. In chordal graph G, see Figure 3.10, for each vertex x, find $N_\infty(x)$ and a simplicial vertex in $N_\infty(x)$.

2. Find the diameter, radius and center of the graph in Figure 3.10.

3. In chordal graph G, see Figure 3.10, find a pair of simplicial vertices which are the ends of a diametral path.

4. In chordal graph G, see Figure 3.10, find a diametral path with both ends not being simplicial vertices.

5. For chordal graph G in Figure 3.10, find a simplicial decomposition ordering; reconstruct G in inverse ordering and track the change of the diameter, radius and center.

Computer Projects 3.4. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a chordal graph G and a vertex x, find all simplicial vertices from $N_\infty(x)$.

2. Given a chordal graph G find a simplicial decomposition ordering; reconstruct G in inverse ordering and track the change of the diameter, radius and center.

3. Given a chordal graph G, find a diametral path.

4. Given a chordal graph G, find diameter, radius and center.

3.5. Quasi-triangulated Graphs

Not only chordal graphs have many interesting properties, but they serve as an important base for further generalizations in graph theory. This section represents an example of such generalization and its application to cyclic structure of graphs and their complements.

In a graph G, a vertex is called **weakly cyclic** if it belongs to no induced $C_k, k \geq 4$. Evidently, in any graph, simplicial vertices belong to no induced $C_k, k \geq 4$; so they are weakly cyclic. Not only simplicial vertices are weakly cyclic. In chordal graphs, all vertices are weakly cyclic. The concept of weakly cyclic vertex is more general than the concept of simplicial vertex.

A graph G is called **latticed** if each vertex belongs to some induced $C_k, k \geq 4$ *and* some induced $\overline{C_l}, l \geq 4$. Latticed graphs are invariant with respect to taking the complement in the sense that both do not contain weakly cyclic vertices. One can think about latticed graphs as of graphs with all vertices being "strongly" cyclic.

In a graph G, a vertex x is called **co-simplicial** if it is simplicial in the complement $\overline{G}$. That means its non-neighbors form an independent set of vertices, or, equivalently, no edge entirely lies outside the neighborhood $N(x)$.

Definition 3.5.1 *A graph G is called* **quasi-triangulated** *if it has a decomposition by sequential elimination of vertices that at each step, are simplicial or co-simplicial.*

Quasi-triangulated graphs are invariant when taking the complement: if G is quasi-triangulated, then its complement $\overline{G}$ is also quasi-triangulated.

For example, C_4 is a quasi-triangulated graph: every vertex is co-simplicial; when we delete any vertex, the remaining graph is chordal. Graph $\overline{C_4}$ is chordal, with all vertices weakly cyclic.

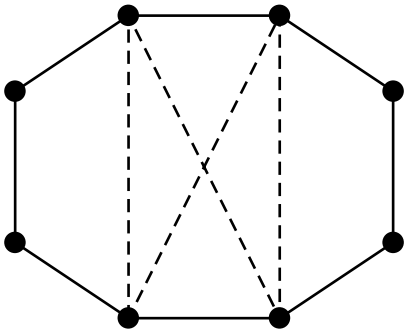


Figure 3.11.

Cycle C_5 is not quasi-triangulated because it has no simplicial or co-simplicial vertices. Moreover, it is latticed because evidently, C_5 and $\overline{C_5}$ are isomorphic.

One can observe that all cycles $C_k, k \geq 5$ are latticed: each vertex belongs to C_k and $\overline{C_4}$, see Figure 3.11.

Consider all graphs that can be decomposed by sequential elimination of vertices which at each step are weakly cyclic in graph or its complement. How they are related to quasi-triangulated graphs? Surprisingly, these two classes of graphs coincide. To prove this, we need the following

Lemma 3.5.1 *If G is a graph and x is a vertex that belongs to no induced $C_k, k \geq 4$, then every minimal separator S in the neighborhood $N(x)$ is a clique.*

Proof. Let G be a graph and x be a vertex that belongs to no induced $C_k, k \geq 4$. Let Y induce a component of $G - S$ that does not contain x, see Figure 3.12. Choose

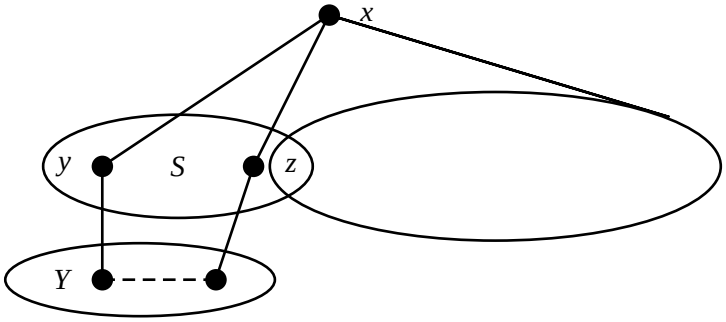


Figure 3.12.

any two vertices $y, z \in S$. Since S is a minimal separator, each of y and z has a neighbor in Y. Since Y induces a connected component, there is a shortest path of length at least two joining y to z through Y. This path together with x forms an induced $C_k, k \geq 4$, a contradiction to our assumption on x. Therefore, y and z must be adjacent, i.e. S induces a clique. □

Theorem 3.5.1 (Characterization theorem, Gorgos, 1984) *For a graph G, the following three statements are equivalent:*

(i) *G is quasi-triangulated.*

(ii) *G can be decomposed by sequential elimination of weakly cyclic vertices in graph or its complement.*

(iii) *G does not contain latticed subgraphs.*

Proof. If G is quasi-triangulated, then it can be decomposed by sequential elimination of weakly cyclic vertices in graph or its complement because every simplicial vertex is weakly cyclic in graph and every co-simplicial vertex is weakly cyclic in the complement. In turn, if G can be decomposed by sequential elimination of weakly cyclic vertices in graph or its complement, then it does not contain latticed subgraphs because they are not decomposable by definition. So, (i) implies (ii), and (ii) implies (iii). Therefore, we only need to prove that (iii) implies (i).

We prove (iii)$\Rightarrow$(i) by induction on the number of vertices $n(G)$. Let $G=(X,E)$ be a graph satisfying (iii). If G contains a simplicial or co-simplicial vertex, then we delete it, the obtained graph has $< n$ vertices, we use the induction hypothesis and prove the implication. Therefore, assume G contains no simplicial vertex and no co-simplicial vertex. The proof is split into the following steps.

Step 1. Proof that G is a connected graph. Suppose G is disconnected. If at least one connected component is a chordal graph, then it has a simplicial vertex what was already excluded. If no one component is chordal, then each contains an induced $C_k, k \geq 4$. Choose any two of them as induced subgraphs, say, $C_k, k \geq 4$ and $C_l, l \geq 4$. Considered together they form a latticed subgraph because each vertex belongs to some cycle ($C_k, k \geq 4$ or $C_l, l \geq 4$) and to the complement of C_4, see Figure 3.13. Therefore, G is a connected graph.

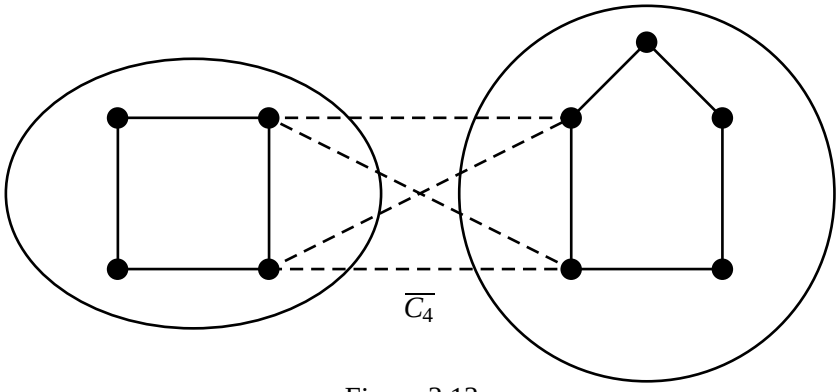


Figure 3.13.

Step 2. Finding subgraph G' and vertex y. Since G is not a latticed graph, one of G or $\overline{G}$ contains a weakly cyclic vertex. Without loss of generality, suppose G contains a weakly cyclic vertex. Denote the set of all weakly cyclic vertices by X'. Hence, $X' \neq \emptyset$.

Let $G' = G - X'$. Subgraph G' has at least one vertex, because otherwise $X' = X$, all vertices of G are weakly cyclic, i.e. G is chordal and contains a simplicial vertex.

Since $X' \neq \emptyset$ we conclude that $n(G') < n(G)$. Evidently, G' does not contain latticed subgraphs. By the induction hypothesis, G' contains a simplicial or co-simplicial vertex; denote it by y.

Since every vertex of G' lies in some $C_k, k \geq 4$, y is co-simplicial. We will prove that y is adjacent to all vertices of X': this will imply y is co-simplicial in the initial graph G, a contradiction.

Step 3. Proof that y is co-simplicial in G. Let x be any vertex in X'. Since G is connected and x is not co-simplicial, there is a nonempty set S of vertices in $N(x)$ that is a minimal separator of G, see Figure 3.14 (x is not shown). By Lemma 3.5.1, S is a clique. Let G_1, G_2 be induced subgraphs of G such that $G = G_1 \cup G_2, G_1 \cap G_2 = S$, and there is no edge between $G_1 - S$ and $G_2 - S$. If at least one of G_1 or G_2 is chordal, then there is a simplicial vertex in $G_1 - S$, or $G_2 - S$, and thus it is a simplicial vertex in G, a contradiction.

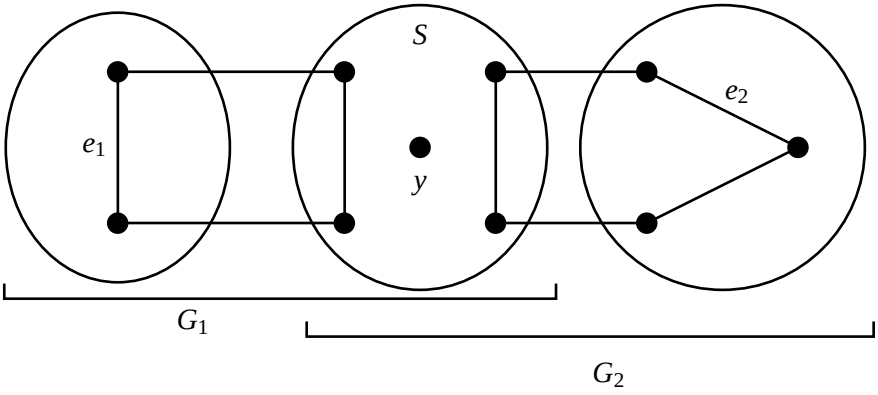


Figure 3.14.

Therefore, G_1 contains a $C_k, k \geq 4$ and G_2 contains a $C_l, l \geq 4$. Since S is a clique, one edge, say e_1, of the first cycle lies completely in $G_1 - S$. Similarly, there is an edge, say e_2, of the second cycle that lies completely in $G_2 - S$. All endpoints of e_1, e_2 are in G'. Since y is co-simplicial in G', all of its non-neighbors from G' are pairwise disjoint. If y lies in $G_1 - S$, then both ends of e_2 are adjacent non-neighbors. If y lies in $G_2 - S$, then both ends of e_1 are adjacent non-neighbors. Therefore, the only possibility for y is to be in S. But $S \subseteq N(x)$, and we obtain that y is adjacent to vertex x. Since x was chosen arbitrary from X', y is adjacent to all vertices from X' and therefore is co-simplicial for the entire graph G. This final contradiction proves the theorem. □

Exercises 3.5.

1. Which of the graphs in Figure 3.15 is quasi-triangulated and which is not?
2. In a graph which is not quasi-triangulated in Figure 3.15, find a latticed subgraph.
3. Construct a quasi-triangulated graph having only one vertex which is simplicial or co-simplicial.
4. For each of graphs G_1, G_2, and G_3 in Figure 3.15, find a minimum clique cover and a maximum independent set.

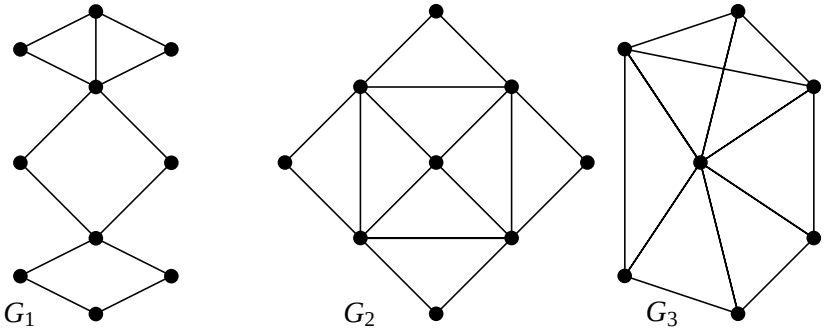


Figure 3.15.

5. Which of the graphs C_n, K_n, W_n, $n \geq 3$, cube, prism and Petersen graph is quasi-triangulated and which is not?

Computer Projects 3.5. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a graph G, recognize if it is quasi-triangulated.
2. Given a graph G, recognize if $\overline{G}$ is quasi-triangulated.
3. Given a quasi-triangulated graph G, find a maximum independent set and a minimum clique cover.
4. Given a quasi-triangulated graph G, find a maximum independent set and a minimum clique cover of the complement $\overline{G}$.

Chapter 4

Planar Graphs

"Planarity is the cause of car collisions..."

4.1. Plane and Planar Graphs

Omitting some topological details we say that a continuous curve in the plane which connects two points (called the first and the last) and has no intersection with itself is called a **Jordan curve**. A Jordan curve is **closed** if the first and the last points coincide. We will use the following

Theorem 4.1.1 (Jordan Curve Theorem) *A closed Jordan curve L partitions the plane into precisely two regions, bounded and unbounded, each having L as boundary.* □

It is clear that the unbounded region contains the infinite point. A region is **connected** if any pair of its points can be connected by a Jordan curve which lies inside the region. In drawing graphs, the vertices are represented by points in the plane, and the edges are represented by Jordan curves connecting the respective points. Evidently, a segment of a straight line is the simplest case of a Jordan curve. If two Jordan curves intersect at a point different from the first and the last for each of them, then we say that the respective edges **cross (intersect)**, or, alternatively, we have an **edge-intersection,** or **crossing**.

A **planar graph** is a graph that *can be drawn* in the plane without crossings of the edges. If a planar graph *is drawn* in the plane without intersections, then such drawing is called a **plane graph**.

Sometimes a plane graph is called **plane embedding** of a planar graph. Plane graph divides the plane into connected regions called **faces.** So each face is bounded by some cycle. The number of vertices of such cycle is called the **size of the face.** For a plane graph G, we will denote the number of faces by $f(G)$ or simply by f.

Any planar graph may have several plane embeddings. Figure 4.1 shows three drawings of the same graph K_4; only the first two of them are plane graphs. One can see that in the first drawing, face f_1 is bounded by the cycle 1-2-4-1, while in the second by the cycle 1-2-3-1 and so on. The number of faces in both embeddings is 4; there is always one face which is not bounded. It is called the **unbounded face**. The unbounded face in both embeddings of K_4 is denoted by f_4.

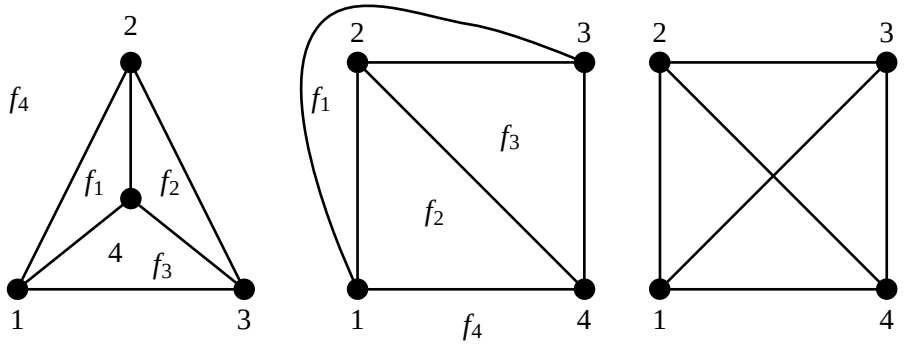


Figure 4.1. Three different pictures of K_4: only the first two are plane graphs.

Having the third drawing of K_4 with one crossing, we could re-draw it to obtain one of the first two plane embeddings. However, it is not possible to do that for every graph. Generally, graphs may have different plane embeddings with different number of crossings. The minimum number of crossings over all possible drawings is called the **crossing number** of a graph. Clearly, for planar graphs the crossing number is 0.

Exercises 4.1.

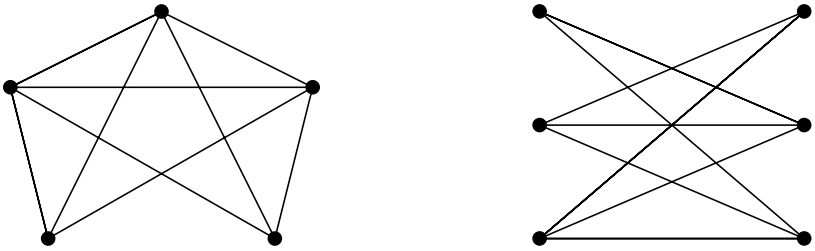


Figure 4.2.

1. For each of the graphs G_1 and G_2 in Figure 4.2, find the number of crossings. Are G_1 and G_2 plane graphs?

2. For each of the graphs G_1 and G_2 in Figure 4.2, find a plane embedding, denote the faces and find the size of each face; find the cycle which forms the unbounded face.

3. Are G_1 and G_2 planar graphs?

4. Connect the two lower vertices in graph G_1 (Figure 4.2) by an edge and, if possible, find a plane embedding of the obtained graph.

5. Connect the two upper vertices in graph G_2 (Figure 4.2) by an edge and, if possible, find a plane embedding of the obtained graph.

6. Find at least two different plane embeddings of the prism, cube, $K_{2,3}$ and W_n ($n \geq 4$).

7. Find a drawing of the Petersen graph with the smallest number of edge-intersections.

Computer Projects 4.1. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a drawing of a graph in the plane (vertices = points, edges = straight line segments) find out if it is a plane graph.

2. Given a drawing of a graph in the plane (vertices = points, edges = arc segments) find out if it is a plane graph.

3. Given a drawing of a graph in the plane (vertices = points, edges = straight line segments) find the number of edge-intersections.

4.2. Euler's Formula

Theorem 4.2.1 (Euler, 1750) *If G is a connected plane graph with n vertices, m edges and f faces, then*

$$n - m + f = 2.$$

Proof. Let T be a spanning tree of G. Evidently, $m(T) = n-1,\ f(T) = 1$, so $n(T) - m(T) + f(T) = n - (n-1) + 1 = 2$, and the formula holds for T.

Now add sequentially the remaining edges of G to T: each such adding increases the number of edges and the number of faces by 1. Since in the formula m and f are of the opposite signs, the equality holds for G itself. □

Corollary 4.2.1 *If a graph G is planar, then all of its plane embeddings have the same number of faces equal to $m - n + 2$.*

Proof. Indeed, since $n(G)$ and $m(G)$ are constant, Euler's formula implies that $f(G) = m - n + 2$.

□

Corollary 4.2.2 *If G is a connected planar graph without parallel edges, then*

$$m(G) \leq 3n(G) - 6.$$

Proof. Consider a plane embedding of G. Since there are no parallel edges, the size of each face is at least 3. Count the edges around each face. The minimum number that we can obtain is $3f$. In fact, each edge is counted twice, so we obtain $2m$. Therefore, $3f \leq 2m$. From Euler's formula, $f = 2 - n + m$. Hence, $3(2 - n + m) \leq 2m$ what gives $m \leq 3n - 6$. □

Corollary 4.2.3 *Every connected planar graph without parallel edges contains a vertex of degree at most* 5.

Proof. If the graph contains at most 6 vertices, then every vertex has degree at most 5. Therefore, it remains to consider the case when the graph has at least 7 vertices. Suppose each vertex has degree at least 6. Counting edges around each vertex results in $6n \leq 2m$, or, equivalently, $3n \leq m$. Combining this inequality with Corollary 4.2.2 we obtain the following: $3n \leq m \leq 3n - 6$, which leads to contradiction $0 \leq -6$. □

For any plane graph G and every its bounded face f, there exists such a plane embedding of G that face f becomes unbounded. This can be shown using the so called **stereographic projection**.

Suppose we have a plane embedding of G. Put a sphere tangent to the plane and map the plane onto the sphere "to the north pole" as shown in Figure 4.3. In this way we obtain an image of G on the surface of the sphere. We now rotate the sphere in such a way that the desired face contains the north pole. From the new north pole, we then project the sphere back onto the plane. A new plane embedding of G is obtained; the face that was bounded in the first embedding, becomes unbounded in this new plane embedding. Therefore, the following proposition holds.

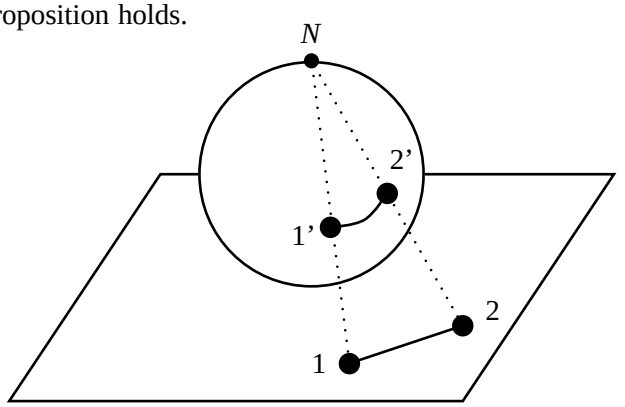


Figure 4.3. Stereographic projection.

Proposition 4.2.1 *A graph can be drawn on the sphere without intersections of edges if and only if it is planar.*

The statement above leads to the following observation about polyhedra (3-dimensional figures bounded by the intersections of planes):

Corollary 4.2.4 *If a convex polyhedron has n vertices, m edges and f faces, then $n-m+f=2$.*

Proof. Indeed, having such a polyhedron, place it into a sphere and project it out onto that sphere. The vertices and edges of the polyhedron form an embedding of a graph on the sphere. We then use stereographic projection to project it onto the plane and obtain a plane embedding of the respective graph. The vertices, edges and faces of such embedding correspond to the vertices, edges and faces of the initial polyhedron. □

For example, consider a cube: $n=8, m=12, f=6$ and evidently, the formula holds.

Exercises 4.2.

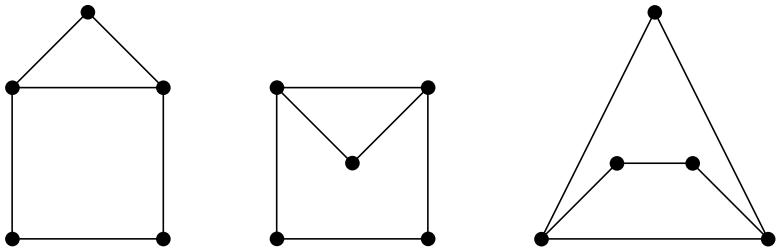


Figure 4.4.

1. Draw a plane embedding of the prism, cube, $K_{2,3}$, each of the graphs in Figure 4.2 and verify the Euler's formula.
2. For each of the drawings in 1., check the inequality of Corollary 4.2.2.
3. Using the stereographic projection, show how to obtain from each other all three embeddings of the same graph in Figure 4.4.
4. Check the inequality of Corollary 4.2.2 for prism, cube, K_n, W_n, $K_{m,n}$ ($m \geq 1, n \geq 3$), and the Petersen graph.

Computer Projects 4.2. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a plane graph G and a face f, construct a plane embedding of G with f being the unbounded face.
2. Given a planar graph G and its drawing in the plane having one crossing. Construct a plane embedding of G.
3. Given a planar graph G, a vertex x and a plane embedding of $G-x$. Construct a plane embedding of G.

4.3. K_5 and $K_{3,3}$ Are not Planar Graphs

Theorem 4.3.1 *K_5 and $K_{3,3}$ are not planar graphs.*

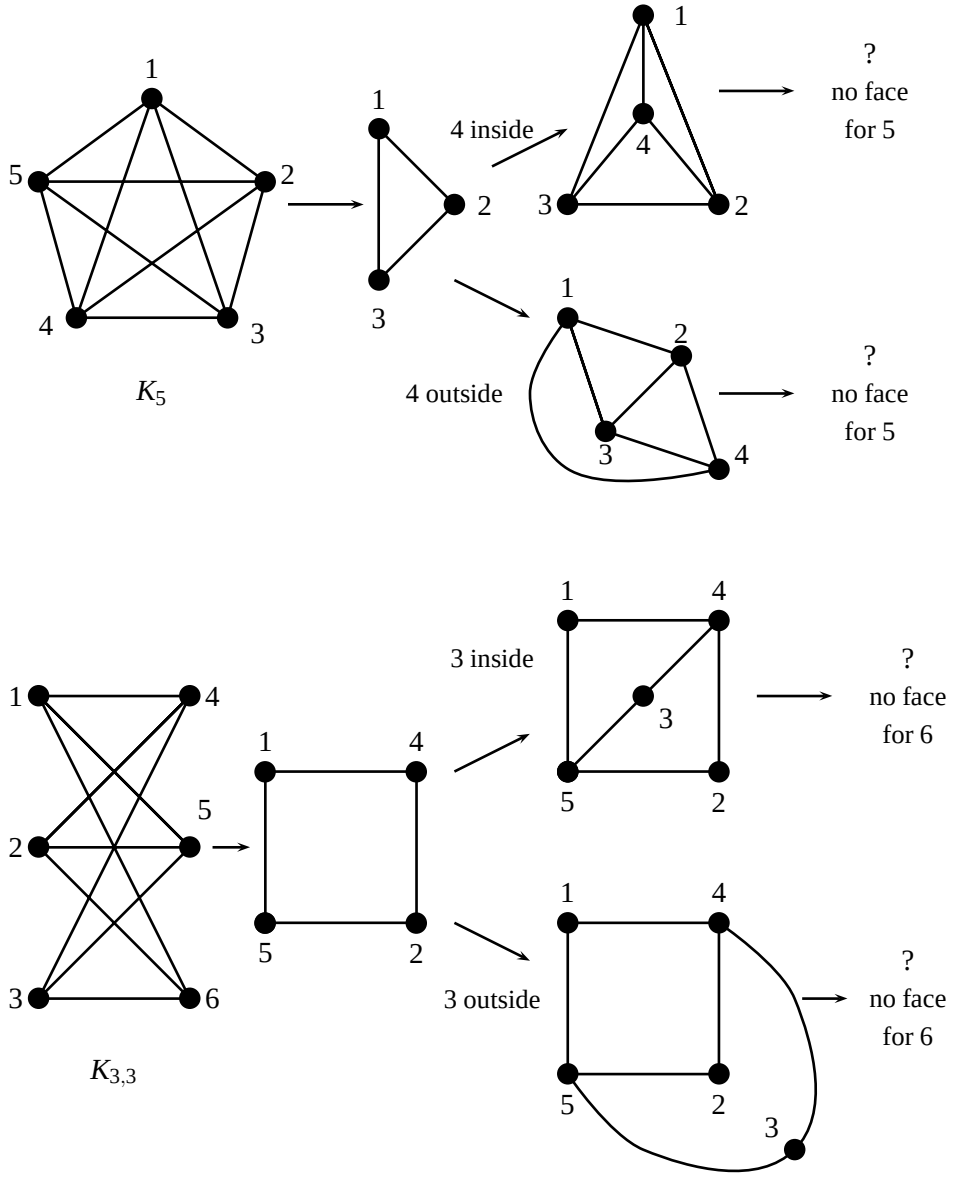


Figure 4.5. K_5 and $K_{3,3}$ are not planar.

Proof. Suppose K_5 is planar. Let us try to draw it in the plane, see Figure 4.5. Since it has cycle 1-2-3, any plane drawing must have this cycle. The cycle partitions the plane into two faces. Vertex 4 may be embedded inside or outside of this cycle. In both cases, since it is adjacent to vertices 1, 2, and 3, the inside or outside face is split into three faces of size

3 each. So, all the faces in the plane have size 3. It remains to find the face for vertex 5. However, since vertex 5 has degree 4, there is no face for it (see Figure 4.5).

If we take vertex 5 instead of 4, and embed it inside or outside of the cycle 1-2-3, we will not be able to find a face for vertex 4. All cases are exhausted; it implies that K_5 cannot be drawn in the plane without crossings of the edges, i.e., it is not planar.

Now suppose $K_{3,3}$ is planar, and vertex set $X = \{1,2,3,4,5,6\}$. Since $K_{3,3}$ is bipartite, without loss of generality assume that vertices 1, 2, and 3 form the first part, and vertices 4, 5, and 6 form the second part, see Figure 4.5.

Let us try to draw $K_{3,3}$ without crossings in the plane. Since it has cycle 1-4-2-5, any plane drawing must have this cycle, so start with it. The plane is now partitioned into two faces. Vertex 3 may be drawn in inside or outside face. Since it is adjacent to vertices 4 and 5, each case leads to three faces of size 4 each. Suppose we draw vertex 3 in inside face. So, there are three faces to draw vertex 6 which must be adjacent to 1, 2, and 3. If we place it in face 1-4-3-5, then it cannot be connected with 2. If we place it in face 2-4-3-5, then it cannot be connected with 1. If we place it in outer face 1-4-2-5, then it cannot be connected with vertex 3. So, there is no face to place vertex 6. The case when vertex 3 is drawn in outside face is considered similarly.

If, instead of vertex 3 we proceed with vertex 6 and embed it inside or outside the cycle 1-2-4-5, by similar reasoning we arrive to the conclusion that there is no face for vertex 3.

All possible cases are considered; it implies that $K_{3,3}$ cannot be drawn in the plane without crossings of the edges, i.e., it is not planar as well. □

Exercises 4.3.

1. Find the values of k for which K_5 and $K_{3,3}$ are k-connected.
2. Using graph $K_{3,3}$ disprove the following statement: let G be a graph and S be a minimal separator having at least three vertices with derived subgraphs G_1 and G_2. Then G is planar if and only if both G_1 and G_2 are planar.
3. Given a graph G and a 2-vertex separator K_2 producing planar derived subgraphs G_1 and G_2. Using plane embeddings of G_1 and G_2, construct a plane embedding of G.
4. Check the inequality of Corollary 4.2.2 for both K_5 and $K_{3,3}$.
5. Prove that any graph G containing K_5 or $K_{3,3}$ as any subgraph (induced or not) is not planar.
6. Find the crossing number of K_5 and $K_{3,3}$.

Computer Projects 4.3. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a 4-regular graph G, determine if it contains K_5 as a subgraph.
2. Given a 3-regular graph G, determine if it contains $K_{3,3}$ as a subgraph.

4.4. Kuratowski's Theorem and Planarity Testing

If a graph is planar, then every its subgraph is evidently planar. If a graph contains a non-planar subgraph, then the graph itself is not planar. Theorem 4.3.1 implies that any graph containing K_5 or $K_{3,3}$ as subgraphs is not planar. But if a graph is not planar, does it contain K_5 or $K_{3,3}$ as subgraphs? The answer is that it may not contain K_5 or $K_{3,3}$ as subgraphs, but it must contain subgraphs closely related to K_5 or $K_{3,3}$.

Let us define two graphs G_1 and G_2 to be **homeomorphic** if both can be obtained from some graph G_3 by replacing some edges with some paths. In other words, we simply draw ("put") additional vertices on some edges of G_3. It is clear that any such replacing does not change the planarity of G_3: if G_3 is planar, then both G_1 and G_2 are planar and vice versa. For example, any two cycles C_k and C_l, with $k,l \geq 3$ are homeomorphic because they both can be obtained from C_3 by such replacing. Observe that any graph homeomorphic to K_5 or $K_{3,3}$ is not planar even if it does not contain K_5 and $K_{3,3}$ as subgraphs of any type.

The next theorem is one of the fundamental theorems in graph theory. It shows that graphs that are homeomorphic to K_5 and $K_{3,3}$ represent the unique cause of non-planarity.

Theorem 4.4.1 (Kuratowski, 1930) *A graph is planar if and only if it does not contain subgraphs homeomorphic to K_5 or $K_{3,3}$.*

Proof. We omit the complete proof of this theorem because it is long and involves many additional results. It can be found in several extended texts, such as e.g. [1, 2, 5]. The idea of the proof can be described using the following steps.

1. Observe that if a graph G is planar, then for any edge of G, there exists an embedding such that the edge belongs to the unbounded face. This embedding can be found by the stereographic projection.

2. Suppose a graph G contains a separator with two vertices, say x,y, such that the derived subgraphs are G_1 and G_2. Let $G'_i = G_i \cup \{x,y\}$. Using step 1, one prove then that if G is not planar, then at least one of G'_1, G'_2 is not planar.

3. The last implies that any minimal non-planar graph must be 3-connected. In other words, it is sufficient to consider further 3-connected graphs only.

4. At this point one use the lemma stating that any 3-connected graph on ≥ 5 vertices contains an edge whose contraction does not change its 3-connectivity.

5. Next one prove the following lemma. In a graph G, contract an edge and obtain a graph G'. The lemma states that if G' contains subgraphs homeomorphic to K_5 or $K_{3,3}$, then the initial graph G also contains subgraphs homeomorphic to K_5 or $K_{3,3}$.

6. Final step: one prove that if G is a 3-connected graph without subgraphs homeomorphic to K_5 or $K_{3,3}$, then it has a plane embedding. The proof is by induction on the number of vertices. In G, contract an edge (guaranteed by step 4) to obtain a graph G' which has less vertices. By step 4, it is 3-connected. By step 5, G' does not contain subgraphs homeomorphic to K_5 or $K_{3,3}$. Therefore, by the induction hypothesis, there exist a plane embedding of G'. The final argument consists in considering a

few possible cases how the initial graph G can be reconstructed from G'. In each of these cases one show that it is possible to construct a plane embedding of G unless it contains subgraphs homeomorphic to K_5 or $K_{3,3}$. □

Planarity testing algorithm. There are many algorithms for determining if a graph is planar. Surprisingly, they do not use search for subgraphs homeomorphic to K_5 or $K_{3,3}$. Next we informally describe the basic idea of such an algorithm.

Consider a graph G and a subgraph $G' \subseteq G$ (not necessarily induced). If we delete the vertices of G' from G, we obtain a number (may be 0) of connected components. A **fragment** of G with respect to G' is one of the following:

1) an edge of G which is not in G' but connects two vertices of G';

2) a connected component of $G - V(G')$ together with edges connecting it to G' (vertices of attachment from G' included).

Now, let G' be a plane graph (algorithm usually starts with a cycle). Find all fragments of G with respect to G'. For each fragment A, determine a set of faces $F(A)$ that contain all vertices of attachment. If $F(A) = \emptyset$ for some A, then G is not planar. If $|F(A)| = 1$ for some A, then select A for the next step. If $|F(A)| > 1$ for all A, then select any fragment A.

Next, chose any path connecting two vertices of attachment of the selected fragment A. Embed the path inside a face from $F(A)$. Call the resulting plane graph G' and repeat the procedure., i.e., find a new set of fragments, for each of them find all admissible faces, and so on. If we arrive to the initial graph G, then G is planar. One can prove that the algorithm works correctly, i.e., if G is planar, then it finds a plane embedding, otherwise it stops at some subgraph $G' \neq G$.

Exercises 4.4.

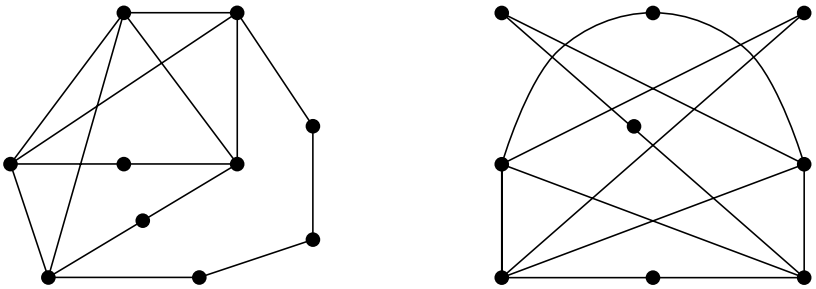


Figure 4.6.

1. For which values of $m \geq 1$ and $n \geq 3$, graphs K_n, $K_{m,n}$, W_n are planar?
2. Count the number of crossings in each of the graphs in Figure 4.6.
3. Which of the graphs in Figure 4.6 is planar and which is not?

4. In Petersen graph, find a subgraph homeomorphic to $K_{3,3}$.

5. What is the minimum number of vertices (edges) that must be deleted from Petersen graph to make it planar?

6. Prove that replacing an edge by multiple edges does not change the planarity of a graph.

Computer Projects 4.4. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a plane graph G and an edge, construct an embedding of G with the edge being on the unbounded face.

2. Given a graph G, determine if it is homeomorphic to K_5.

3. Given a graph G, determine if it is homeomorphic to $K_{3,3}$.

4. Given a graph G and $n \geq 4$, determine if G is homeomorphic to C_n.

5. Given a graph G and $n \geq 4$, determine if G is homeomorphic to K_n.

6. Given a graph G and $n \geq 4$, determine if G is homeomorphic to W_n.

7. Given a graph G, determine if it is homeomorphic to a cube.

8. Given a graph G, determine if it is homeomorphic to a prism.

9. Given a graph G, determine if it is planar.

4.5. Plane Triangulations and Dual Graphs

Plane triangulations. A simple connected plane graph is called **plane triangulation** if every its face, including unbounded, represents a triangle (i.e. has size 3). A simple planar (plane) graph is called **maximal planar (plane) graph** if adding any new edge to it results in a non-planar graph.

One can prove that these two concepts are equivalent, namely a graph is a plane triangulation if and only if it is a maximal plane graph. It is easy to determine the number of edges in a plane triangulation:

Theorem 4.5.1 *For any plane triangulation,* $m = 3n - 6$.

Proof. Indeed, if we count the edges around each face, we obtain the equality $3f = 2m$. Substitution of f from Euler's formula results in $m = 3n - 6$. □

Every plane graph is a spanning subgraph of some plane triangulation; the latter can be obtained by adding edges to a given plane graph. Plane triangulations are important because sometimes it is sufficient to prove results for them to conclude that the results hold for all planar graphs.

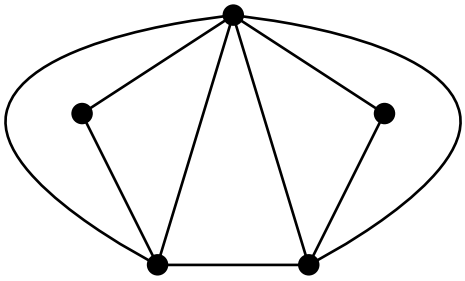


Figure 4.7.

An example of a graph G which is a plane triangulation is shown in Figure 4.7. One can see that all faces including unbounded face are triangles. G has $n = 5$ vertices and $m = 3n - 6 = 15 - 6 = 9$ edges. Thus any plane graph on five vertices is a subgraph of graph G.

Dual graphs. For any plane graph G one can construct another plane graph denoted by G^* and called the **dual** of G. The rules are the following:

1. In each face of G choose a point which becomes a vertex of G^*.
2. For each edge of G separating faces f_i and f_j construct an edge of G^* connecting vertices f_i and f_j.

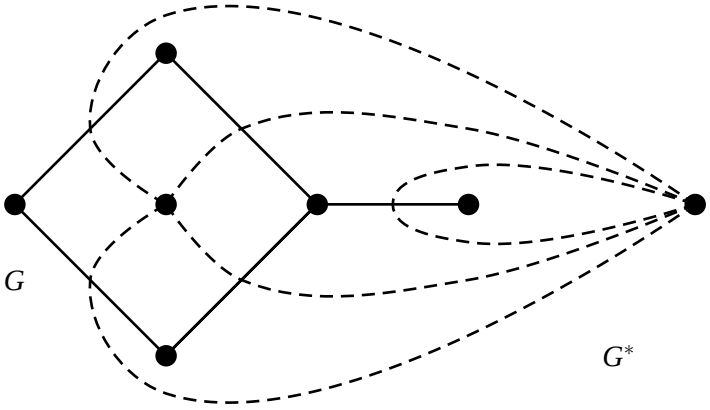


Figure 4.8. Plane graph G and its dual G^*.

An example of a graph G and its dual G^* is shown in Figure 4.8. The edges of G are drawn by solid lines while the edges of G^* are drawn by dashed curves. Since G had two faces, G^* has two vertices. Every edge of G is crossed by the corresponding edge of G^*. As one can see, the dual to simple graph G is not a simple graph, in particular, the separating edge of G corresponds to a loop of G^*.

It is evident that $n(G^*) = f$, $m(G^*) = m$ and $f(G^*) = n$. The last follows from the observation that all faces around each vertex in G are consecutively connected by the edges of G^* and thus produce a face of G^*.

Theorem 4.5.2 *If G is a plane connected graph, then $(G^*)^*$ is isomorphic to G.*

Proof. Evidently, one can reconstruct graph G from the plane embedding of G^* using the same rules. □

One can show however, that different embeddings of the same planar graph G may have non-isomorphic duals.

Exercises 4.5.

1. In K_5, delete an edge, draw a plane embedding which is a triangulation and construct the dual.
2. In $K_{3,3}$, delete an edge, draw a plane embedding which is a triangulation and construct the dual.
3. Construct the dual to a prism.
4. Construct the dual to a cube.
5. Construct the dual to $W_n, n \geq 4$.

Computer Projects 4.5. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a plane graph G, draw G^*.
2. Given a plane graph, complete it to a triangulation.

Chapter 5

Graph Coloring

"Warnings: do not use in painting; do not give any preference to any of the colors..."

5.1. Preliminary

Coloring theory started with the problem of coloring the countries of a map in such a way that no two countries that have a common border receive the same color. If we denote the countries by points in the plane and connect each pair of points that correspond to countries with a common border by a curve, we obtain a planar graph. The celebrated **Four Color Problem** asks if every planar graph can be colored with 4 colors. It seems to have been mentioned for the first time in writing in an 1852 letter from A. De Morgan to W.R. Hamilton. Nobody thought at that time that it was the beginning of a new theory. The first "proof" was given by Kempe in 1879. It stood for more than 10 years until Heawood in 1890 found a mistake. Heawood proved that five colors are enough to color any map. The Four Color Problem became one of the most famous problems in discrete mathematics of the 20th century. Besides colorings it stimulated many other areas of graph theory.

Generally, coloring theory is the theory about conflicts: adjacent vertices in a graph always must have distinct colors, i.e. they are in a permanent conflict. If we have a "good" coloring, then we respect all the conflicts. If we have a "bad" coloring, then we have a pair of adjacent vertices colored with the same color. This looks like having a geographic map where some two countries having common border are colored with the same color. Graphs are used to depict "what is in conflict with what", and colors are used to denote the state of a vertex. So, more precisely, coloring theory is the theory of "partitioning the sets having internal unreconcilable conflicts" because we will only count "good" colorings.

In day by day life, perhaps the most common and simple application of coloring is in traffic lights. People who invented traffic lights did not even realize how smart they were: they were first to observe that it was not important what sign (stop, or drive) to put; it was only important that whatever they put should be in different states at any moment of time for two given streets which intersect. Of course, as in many practical applications of math, they borrowed "red" and "green" from maritime rules, added "yellow" for an intermediate state, and put that on the most visible position. But mathematical model only represents the coloring of graph K_2.

It may look surprising but in graph coloring it does not matter which color is "blue" or which is "red" etc; it only matters how many different colors are available. No color has any preferences. Instead of really coloring the vertices of a graph, we just label them by numbers 1,2,..., λ where λ is the number of available colors. In this sense, the graph coloring is the most color-blind subject.

In the most general setting, a color of a vertex may be thought of as a "state of a point", or even more generally, a "statement about anything".

5.2. Definitions and Examples

Let $G=(X,E)$ be a simple graph and $\{1,2,\ldots,\lambda\}$ be the **set of available colors**. Any labeling of vertices of graph G by the numbers from $\{1,2,\ldots,\lambda\}$ is called a **coloring**. Each vertex is assigned precisely one color.

Definition 5.2.1 *A coloring is called* **proper** *if adjacent vertices have* **different** *colors.*

Not every coloring is proper. In a proper coloring, vertices of the same color induce an independent set. If we change the color for at least one vertex, we obtain another coloring. It may be proper or not. The number of all proper colorings of a graph G with at most λ colors is denoted by $P(G,\lambda)$**.** We will see that $P(G,\lambda)$ is a polynomial in λ and therefore we call it the **chromatic polynomial.** When we need to underline the number of available colors, we say that we consider a **proper λ-coloring.** Notice that the number of really used colors in a proper λ-coloring may be strictly less than λ; but it can never be greater than λ. Often, when we say just "coloring" we mean proper coloring. The minimum number of colors over all proper colorings is called the **chromatic number** of a graph G, denoted by $\chi(G)$. Evidently, the maximum number of colors that can be used in a proper λ-coloring is $\min\{n(G),\lambda\}$.

Since there are no colorings with less than $\chi(G)$ colors, $P(G,\lambda)=0$ for all integer values of λ such that $1\le\lambda\le\chi(G)-1$. Since any coloring with $\chi(G)$ colors is at the same time a proper λ-coloring for any $\lambda\ge\chi(G)$, we conclude that $P(G,\lambda)\ge 1$ for any integer $\lambda\ge\chi(G)$. Since λ is the number of colors, in our discussions it is always an integer variable.

Consider an example. Let $G=(X,E)$ where $X=\{x,y,z\}$, and $E=\{\{x,y\},\{y,z\}\}$, see Figure 5.1. There is no coloring with one color, so $\chi(G)>1$. On the other hand, it is easy to see a coloring with two colors, say 1-2-1, so $\chi(G)=2$. Suppose the set of available colors is $\{1,2,3\}$, consequently, $\lambda=3$. Having totally three colors and keeping in mind that $\chi(G)=2$, we obtain the following possibilities for the number i of really used colors: $i=2$ and $i=3$. If we use only 2 colors, then we have to chose them from $\{1,2,3\}$. Thus we have the choices: colors 1,2, colors 2,3, and colors 1,3. The total number of choices equals $\binom{\lambda}{i}=\binom{3}{2}=3$. For the first choice, the colorings are 1-2-1 and 2-1-2, see the first two colorings in Figure 5.1, column $i=2$. For the second choice, the colorings are 2-3-2 and 3-2-3. For the last choice, the colorings are 1-3-1 and 3-1-3. Notice that in this case the colorings come in pairs, and in each pair, we just permute the colors.

If $i=3$, we observe that any coloring will be proper because all the vertices are of different colors. To construct all colorings, fix color 1 for x and obtain two colorings by permuting colors 2 and 3 on y and z: 1-2-3 and 1-3-2. Do the same for color 2: we have

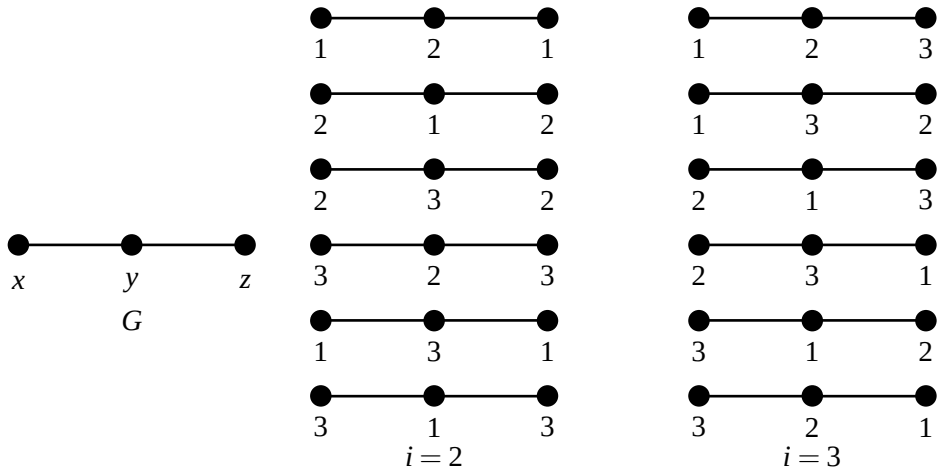


Figure 5.1. Graph G and 12 proper 3-colorings: 6 strict 2-colorings and 6 strict 3-colorings.

colorings 2-1-3, 2-3-1. At last, for color 3 we obtain the colorings 3-1-2 and 3-2-1, see Figure 5.1.

Summarizing all the cases we arrive to the conclusion that $P(G,3) = 12$. What about $P(G,\lambda)$ for an arbitrary value of λ? Should we consider all possibilities by exhaustive search? And what about $P(G,\lambda)$ for any graph? Is there any formula? We will show that there are general procedures for computation of $P(G,\lambda)$ for an arbitrary graph G.

Before proceeding to the next section, let us consider one more concept. Let $G = (X,E)$ be a graph, and i be the number of used colors. So, each of i colors is in use. Such proper colorings are called the **strict i-colorings.** Each strict i-coloring partitions the vertex set X into i nonempty subsets, called **cells** where each cell is the set of vertices of the same color. Such **partitions** are called **feasible** and cells are called **color classes**. In other words, in a feasible partition of X into i cells, adjacent vertices belong to different cells and each cell is an independent subset of vertices. Notice that proper colorings and feasible partitions are different concepts: colorings are labelings of vertices, partitions are divisions of X into nonempty subsets. However, they are closely related. They are even more close than it appears.

In contrast with λ which has no upper limit, the number i of really used colors satisfies the following inequality: $1 \le i \le n(G)$. Let $r_i(G)$ **be the number of feasible partitions of** G **into** i **cells**. The vector

$$R(G) = (r_1, r_2, \ldots, r_n)$$

is called the **chromatic spectrum** of G.

Since by definition $\chi(G)$ is the smallest number of colors in a proper coloring, the chromatic spectrum, in fact, always has the following form:

$$R(G) = (0,0,\ldots,0,r_\chi,r_{\chi+1},\ldots,r_n).$$

In our example, see Figure 5.1, when $i = 2$, we have six proper colorings. Each of these colorings is a proper 3-coloring and simultaneously, a strict 2-coloring. For all cases, there is only one feasible partition of X: $X_1 = \{x,z\}$(first cell) and $X_2 = \{y\}$ (second cell). Therefore, $r_2(G) = 1$.

When $i = 3$, we have other six proper colorings. Each of these colorings is a proper 3-coloring and simultaneously, a strict 3-coloring. All the colorings generate only one feasible partition with cells $X_1 = \{x\}$, $X_2 = \{y\}$, and $X_3 = \{z\}$. Hence $r_3(G) = 1$. Since evidently, $r_1(G) = 0$, we obtain that the chromatic spectrum of G is:

$$R(G) = (0,1,1).$$

The chromatic spectrum $R(G)$ is called **continuous (gap-free)** if it does not contain zeroes between positive components. Otherwise it is called **broken (has gaps)**.

Theorem 5.2.1 *For any graph G, the chromatic spectrum $R(G)$ is continuous.*

Proof. As we noticed, $R(G) = (0,0,\ldots,0,r_\chi,r_{\chi+1},\ldots,r_n)$. By definition, $r_\chi > 0$. Consider any strict coloring of G with χ colors and choose any color class which has > 1 vertices. Split it into two non-empty subsets; we obtain another feasible partition using $\chi+1$ cells what implies $r_{\chi+1} > 0$. Repeating this procedure further for any other cell having > 1 vertices, conclude that $r_{\chi+2} > 0$, $r_{\chi+3} > 0$, ... We cannot continue the splitting when all cells have precisely one vertex each, and that corresponds to $r_n = 1 > 0$. □

This fundamental property does not hold in a more general case of hypergraph coloring.

There are many results about the bounds on the chromatic number. We end the section with the following observations. The König theorem (Theorem 2.4.1) about bipartite graphs may now be reformulated in terms of colorings:

Theorem 5.2.2 *For a graph G, $\chi(G) \leq 2$ if and only if it has no odd cycles.*

An important theorem relating maximum degree $\Delta(G)$ and chromatic number $\chi(G)$ of a graph G is

Theorem 5.2.3 (Brooks, 1941) *If G is a connected graph different from a clique and an odd cycle, then $\chi(G) \leq \Delta(G)$.* □

Exercises 5.2.

1. Find the chromatic number of E_n, K_n, $K_{m,n}$, tree T_n, C_{2n}, C_{2n+1}, W_n, prism, cube and Petersen graph.

2. In graph G, see Figure 5.1, delete an edge and find all feasible partitions, $P(G',3)$ and $R(G')$.

3. For all graphs from 1. starting with any proper coloring by the minimum number of colors, show that the chromatic spectrum is gap free.

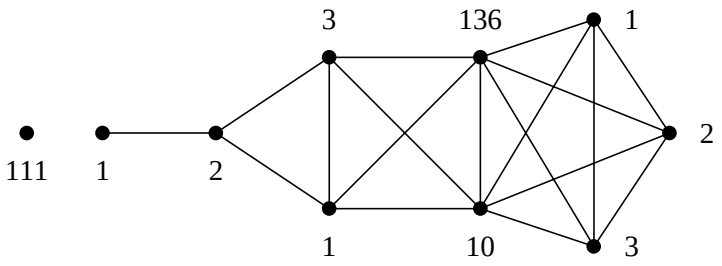


Figure 5.2.

4. Figure 5.2 exhibits a graph G and a proper coloring. Find a respective feasible partition and values of λ for which the coloring is a proper λ-coloring. Find a value of i for which the coloring is a strict i-coloring.

5. Beginning with the feasible partition into $i = 6$ cells for the graph in Figure 5.2, construct a sequence of feasible partitions into $i+1$, $i+2$, ..., $n = 10$ cells.

6. For the graph in Figure 5.2, find a bound on the chromatic number by applying Brooks Theorem.

7. For the graph in Figure 5.2, find the exact value of the chromatic number and the corresponding optimal coloring.

8. For the graph in Figure 5.2, starting with optimal χ-coloring, construct a sequence of feasible partitions into χ, $\chi+1$, $\chi+2$, ..., $n = 10$ cells.

Computer Projects 5.2. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. Given a graph G and a coloring, check if the coloring is proper.

2. Given a graph G and a proper coloring, output the respective feasible partition.

3. Given a graph G, using a generator of random numbers, generate a proper coloring.

5.3. Structure of Colorings

Let us have $\lambda \geq 1$ colors and consider graph K_n. Evidently, if $\lambda < n$, then there are no proper colorings of K_n because all the vertices must be colored pairwise differently. Suppose now $\lambda \geq n$. Any assignment of n different colors to the vertices of K_n results in a proper coloring, so $\chi(K_n) = n$. What about $P(K_n, \lambda)$?

Let us start coloring K_n. We have λ possibilities to color the first vertex. Then, for the second vertex, we have $(\lambda - 1)$ possibilities. For the third vertex there are $(\lambda - 2)$ possibilities. Any color used once, cannot be used again. Because of the symmetry of K_n, the order of such coloring procedure does not matter. If we continue it until the last vertex is colored, we arrive to the conclusion that

$$P(K_n, \lambda) = \lambda(\lambda-1)(\lambda-2)\dots(\lambda-n+1).$$

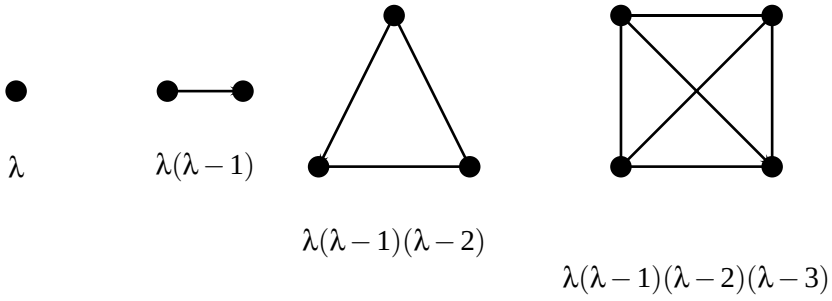


Figure 5.3. Complete graphs K_1, K_2, K_3, K_4, and their chromatic polynomials.

Figure 5.3 shows examples of the chromatic polynomials of the simplest complete graphs. The product $\lambda(\lambda-1)(\lambda-2)\dots(\lambda-n+1)$ is denoted by $\lambda^{(n)}$ and sometimes called the **falling factorial**.

Let us have now a graph $G=(X,E)$ which is not a K_n. How to compute $P(G,\lambda)$? Since G is not K_n, it has two nonadjacent vertices, say x and y. All proper colorings of G with λ colors are split into two classes: when x and y have different colors and when x and y have the same color. All proper λ-colorings of G when x and y have different colors are the proper λ-colorings of the graph $G_1 = G \cup \{x,y\}$. All proper λ-colorings of G when x and y have the same color are the proper λ-colorings of the graph $G_2 = G_1 \cdot \{x,y\}$ (recall that $G \cdot e$ denotes the graph obtained by contraction of an edge e, see Section 1.6., Figure 1.23). Therefore,

$$P(G,\lambda) = P(G_1,\lambda) + P(G_2,\lambda). \tag{5.1}$$

Notice that compared to G, graph G_1 has the same vertices and edges except a newly added edge $\{x,y\}$. In its turn, graph G_2 has the same edge set and the same vertex set except that x and y are replaced be the new vertex xy. Graph G_1 has more edges, and graph G_2 has less vertices than G. It is important to observe that graph G_2 may have multiple edges. Namely, if vertices x and y have common neighbors in G, then contraction of edge $\{x,y\}$ leads to multiple edges. Since in the definition of proper coloring multiple edges play no role (only adjacency is important), we replace every multiple edge in graph G_2 by a single edge. So, without any loss of generality for colorings, we can assume that graph G_2 is also simple.

Figure 5.4 shows how we can depict the equality above as the equality of graph drawings. In this way, graph G_1 stands for "connection" and graph G_2 stands for "contraction". The **connection-contraction algorithm** itself consists in recurrent application of this step to every graph obtained. That means if G_1 or G_2 is not a complete graph, then we find another pair of not adjacent vertices and proceed as we did it with x and y. If $P(G_1,\lambda) = P(G_3,\lambda) + P(G_4,\lambda)$ and $P(G_2,\lambda) = P(G_5,\lambda) + P(G_6,\lambda)$, then we obtain that $P(G,\lambda) = P(G_3,\lambda) + P(G_4,\lambda) + P(G_5,\lambda) + P(G_6,\lambda)$ and so on. When the procedure stops? It stops when we are not able to apply the connection, i.e. all obtained graphs are complete graphs. Therefore,

$$P(G,\lambda) = P(K_{i_1},\lambda) + P(K_{i_2},\lambda) + \dots + P(K_{i_s},\lambda)$$

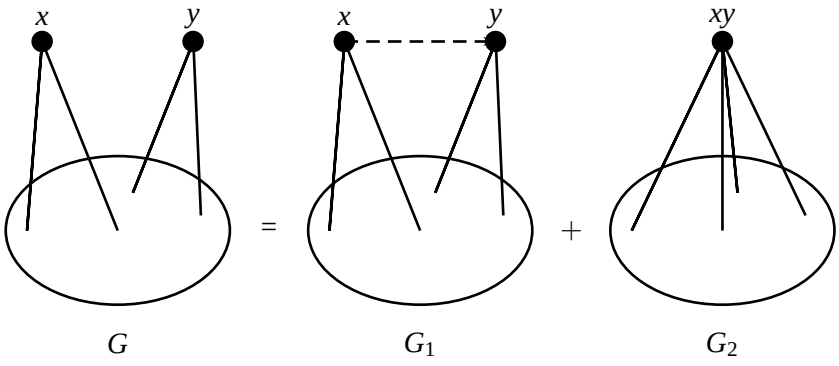


Figure 5.4. Connection-contraction.

for some integer $s \geq 1$.

Since $P(K_{i_j}, \lambda)$ is a polynomial in λ, and the sum of polynomials is a polynomial, we conclude immediately that $P(G, \lambda)$ **is a polynomial in** λ. Moreover, among all complete graphs K_{i_1}, K_{i_2}, ... , K_{i_s} only one is isomorphic to K_n; namely that which is obtained by connections only. All the other complete graphs have $< n$ vertices because they are obtained from G by at least one contraction. Therefore, keeping in mind that $P(K_n, \lambda) = \lambda(\lambda-1)(\lambda-2)\ldots(\lambda-n+1)$, our next conclusion is that the **degree of** $P(G, \lambda)$ **is** n **and the major coefficient is 1.**

However, deeper analysis leads to deeper conclusions. Recall that all vertices of G have their names, i.e. labels. When contracting the edges we concatenated the names and produced new names for new vertices. As the result, the vertices in every complete graph K_i have composite names. Only the names of vertices in the unique graph K_n are exactly the same as they are in G. Notice the following important fact. If the name of a vertex in some K_i is xyz, for example, then there is a strict i-coloring of G where vertices x, y, and z are colored with the same color. In other words, every complete graph K_i corresponds to some strict i-coloring of G, each vertex of K_i corresponds to some color, and the composite name corresponds to all vertices of G colored with that color.

There are no identical complete graphs with the same number of vertices if we compare the composite names of vertices. It is so because each K_i was obtained by a unique way from G. Therefore, the **number of all complete graphs having** i **vertices equals the number of all feasible partitions of** G **into** i **cells,** $r_i(G)$.

Collecting now all complete graphs on the same number of vertices and doing that for all possible values we obtain the following equality:

$$P(G,\lambda) = P(K_{i_1},\lambda) + P(K_{i_2},\lambda) + \cdots + P(K_{i_s},\lambda) =$$
$$r_1(G)P(K_1,\lambda) + r_2(G)P(K_2,\lambda) + \cdots + r_n(G)P(K_n,\lambda) =$$
$$\sum_{i=1}^{n} r_i(G)P(K_i,\lambda). \tag{5.2}$$

Since $r_1 = r_2 = \cdots = r_{\chi-1} = 0$ and $P(K_i, \lambda) = \lambda^{(i)}$, we arrive at last to the final **fundamental relation**:

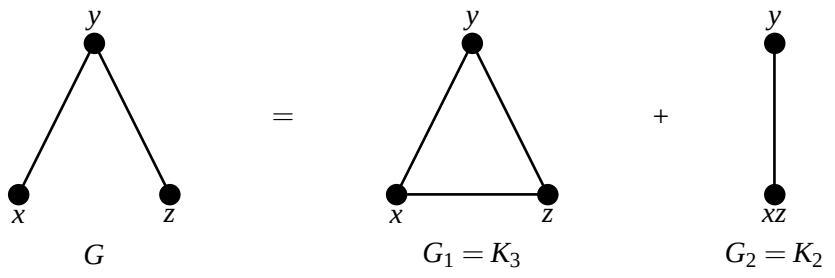


Figure 5.5.

$$P(G,\lambda) = \sum_{i=\chi}^{n} r_i(G)\lambda^{(i)}. \tag{5.3}$$

Equality (5.3) explicitly shows the structure of proper colorings and their relation to feasible partitions: fix any feasible partition into i cells, then count all the colorings that can be obtained from this partition by permutation of the colors ($\lambda^{(i)}$), then do that for all feasible partitions ($r_i(G)\lambda^{(i)}$), and at last count that for all i (obtain $\sum_{i=\chi}^{n} r_i(G)\lambda^{(i)}$). Connection-contraction algorithm not only shows this structure, it also shows the way how to obtain $P(G,\lambda)$ for any graph G.

Consider an example, see Figure 5.5. Let $G = (X,E)$ be the same graph as in Figure 5.1. Apply connection-contraction and immediately obtain

$$P(G,\lambda) = P(K_3,\lambda) + P(K_2,\lambda) = \lambda^{(3)} + \lambda^{(2)} =$$

$$\lambda(\lambda-1)(\lambda-2) + \lambda(\lambda-1) = \lambda(\lambda-1)^2.$$

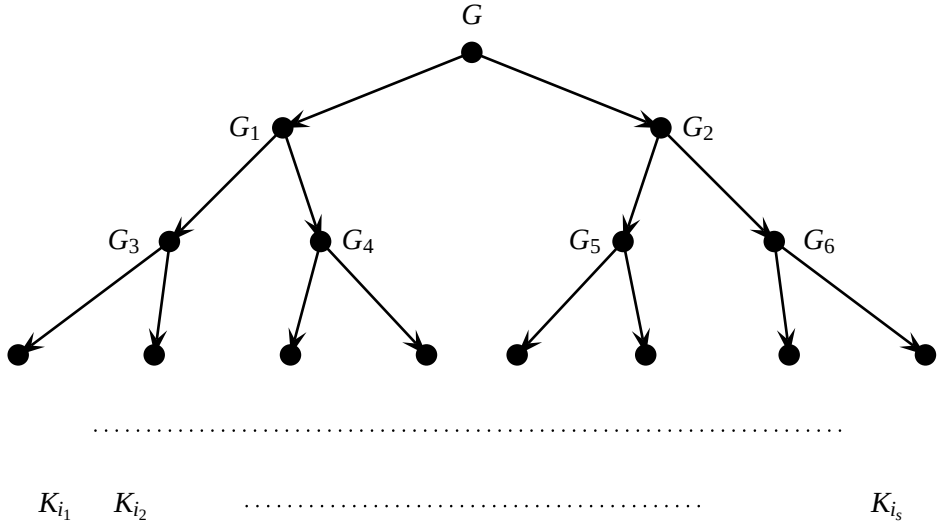


Figure 5.6. Connection-contraction tree.

Now without any exhaustive search for the colorings, we obtain, for example, that $P(G,3) = 3(3-1)^2 = 12$. Moreover, since $P(G,\lambda) = P(K_3,\lambda) + P(K_2,\lambda) = 0 \cdot P(K_1,\lambda) + 1 \cdot P(K_2,\lambda) + 1 \cdot P(K_3,\lambda)$, we conclude that the chromatic spectrum $R(G) = (0,1,1)$. The structure of the colorings can also be seen: when $i = 3$, all the colorings are obtained from permutations of the colors; when $i = 2$, all the colorings are obtained by permutations of two colors when vertex y is colored with one color and vertices x and z with the other color.

The connection-contraction algorithm is good for small graphs but it is not efficient for large n. Since every graph produces two new graphs, the number of graphs is doubling at each step and is power of 2. The whole procedure can be depicted itself as a graph, which is a directed tree, see Figure 5.6. Each arc shows which graph is obtained from which; left directed arcs denote connection, right directed arcs denote contraction. Graph G is located on the zero level, graphs G_1 and G_2 on the first level, graphs G_3, G_4, G_5, and G_6 on the second level and so on down. Not all of the graphs K_{i_j} are on the last level. Some of them may appear on higher levels.

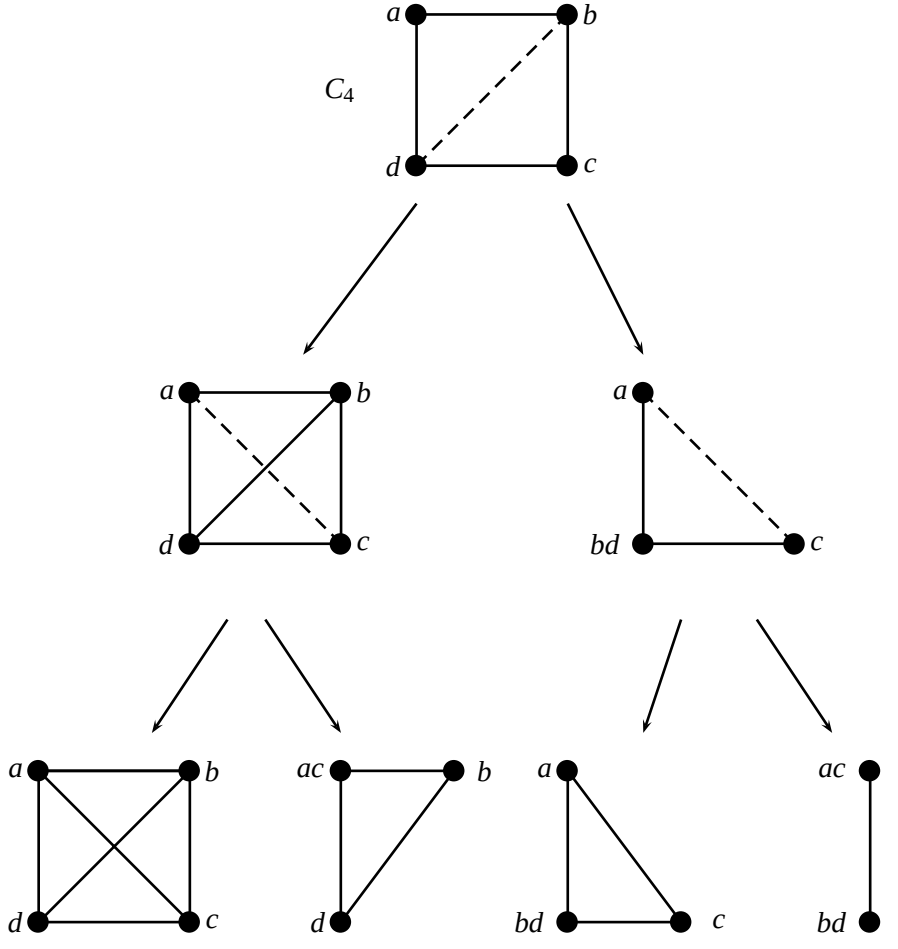


Figure 5.7.

Since the chromatic polynomial is unique for any graph G, there is a unique expansion of G into combination of complete graphs as described in equality (5.2). This observation leads us to the conclusion that the **order in which the connection-contraction is implemented is not important**.

An implementation of the connection-contraction algorithm for cycle C_4 is shown in Figure 5.7 in full. Edges to be added at the next level and then contracted are shown by dashed lines. Multiple edges which appear after contraction are not shown. All feasible partitions of C_4 can be found from the names of the vertices of complete graphs. One can conclude that

$$\begin{gathered} P(C_4,\lambda) = 0\cdot P(K_1,\lambda) + 1\cdot P(K_2,\lambda) + 2\cdot P(K_3,\lambda) + 1\cdot P(K_4,\lambda) = \\ \lambda^{(2)} + 2\lambda^{(3)} + \lambda^{(4)} = \lambda(\lambda-1) + 2\lambda(\lambda-1)(\lambda-2) + \lambda(\lambda-1)(\lambda-2)(\lambda-3) = \\ \lambda(\lambda-1)[1+2(\lambda-2)+(\lambda-2)(\lambda-3)] = \lambda(\lambda-1)(\lambda^2-3\lambda+3) = \\ \lambda^4 - 4\lambda^3 + 6\lambda^2 - 3\lambda. \end{gathered} \tag{5.4}$$

and

$$R(C_4) = (0,1,2,1).$$

If for example, 10 colors are available, then the number of all proper 10-colorings is $P(C_4,10) = 10\cdot 9\cdot(10^2 - 3\cdot 10 + 3) = 6570$.

Exercises 5.3.

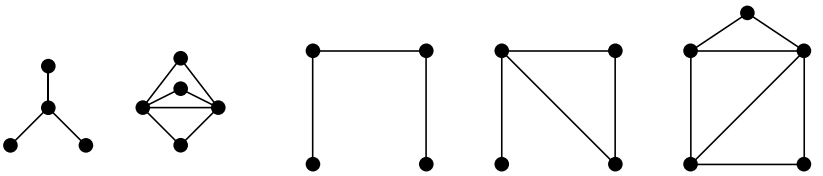


Figure 5.8.

1. Applying the connection-contraction algorithm find the chromatic polynomial of the following graphs: E_4, E_5, C_5, C_6, W_4, W_5, P_4, P_5, P_6 and P_n for every $n \geq 6$.
2. Applying the connection-contraction algorithm find all feasible partitions for P_4, P_5, P_6, C_5 and W_5.
3. Applying the connection-contraction algorithm find all feasible partitions and the chromatic spectrum for the graphs in Figure 5.8.
4. A manager needs to arrange five people A, B, C, D and E in the offices. Person A is in conflict with B and E; person B is in conflict with A and C; person C is in conflict with B and D; person D is in conflict with A, B, C, E; person E is in conflict with A and D. What is the minimum number of offices and in how many ways the people can be arranged in such a way that there is no conflict inside any office? Find all the assignments to the offices.

5. There are five cell phone towers A, B, C, D, and E. When transmitting a signal using the same frequency, tower A interferes with B and E; tower B interferes with A and C; tower C interferes with B and D; tower D interferes with A, B, C, E; tower E interferes with A and D. What is the minimum number of frequencies necessary to avoid interference at any moment of time? Find all possible optimal assignments of frequencies to the towers.

6. Which of the graphs in Figure 5.8 is the model for the problems 4 and 5 above?

Computer Projects 5.3. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. Given a graph G, using a generator of random numbers, find several feasible partitions.

2. Given a graph G, using a generator of random numbers, find the lower bound on the chromatic spectrum.

3. Apply the connection-contraction algorithm to find all feasible partitions, chromatic polynomial and the chromatic spectrum for: a) cube; b) Petersen graph.

5.4. Chromatic Polynomial

Proposition 5.4.1 *Suppose G is not a connected graph, and let $G_1, G_2, \ldots, G_k$ be the connected components, $k \geq 2$. Then*

$$P(G,\lambda) = P(G_1,\lambda)P(G_2,\lambda)\ldots P(G_k,\lambda). \tag{5.5}$$

Proof. Indeed, each component can be colored independently; since G_1 has $P(G_1, \lambda)$ proper colorings, G_2 has $P(G_2,\lambda)$ proper colorings, and so on, the total number of proper colorings of G is the product of these numbers. □

The equality (5.5) can be immediately applied to graph E_n: since $E_n = nK_1$, and $P(K_1,\lambda) = \lambda$, we obtain

$$P(E_n,\lambda) = \lambda \cdot \lambda \cdot \cdots \cdot \lambda = \lambda^n.$$

On the other hand, if we apply connection-contraction algorithm to E_n, we obtain the following equality:

$$\lambda^n = S(n,1)\lambda^{(1)} + S(n,2)\lambda^{(2)} + \ldots + S(n,n)\lambda^{(n)} \tag{5.6}$$

where $S(n,i) = r_i(E_n)$ are some numbers; in other words, $S(n,i)$ equals the number of partitions of a set of n elements into i subsets. These numbers are known as the so called **Stirling numbers of the second kind**. Therefore, the chromatic spectrum of E_n is nothing else than

$$R(E_n) = (S(n,1), S(n,2), \ldots, S(n,n)).$$

One can check, for example, that $R(E_1) = (1), R(E_2) = (1,1)$, $R(E_3) = (1,3,1)$, $R(E_4) = (1,7,6,1)$ and so on.

In turn, if we expand the expression for $\lambda^{(n)}$, then we obtain some polynomial with coefficients:

$$\lambda^{(n)} = s(n,1)\lambda + s(n,2)\lambda^2 + \ldots + s(n,n)\lambda^n$$

which are called the **Stirling numbers of the first kind**. For example, $\lambda^{(4)} = -6\lambda + 11\lambda^2 - 6\lambda^3 + \lambda^4$, and therefore $s(4,1) = -6$, $s(4,2) = 11$, $s(4,3) = -6$, and $s(4,4) = 1$.

Generally, Stirling numbers serve as the coefficients to express λ^n (or, equivalently, E_n) in terms of $\lambda^{(i)}$ (or, equivalently, K_i) and $\lambda^{(n)}$ in terms of λ^i, $i = 1,2,\ldots,n$. The latter in a more general setting can be expressed as the **disconnection-contraction algorithm**.

The idea of it consists in the following. Recall that in the connection-contraction we had the equality (5.1):

$$P(G,\lambda) = P(G_1,\lambda) + P(G_2,\lambda)$$

where G_1 is obtained from G by connection, and G_2 by contraction. We re-write this equality as

$$P(G_1,\lambda) = P(G,\lambda) - P(G_2,\lambda). \tag{5.7}$$

We now look as if G_1 is an original graph, and G is obtained from G_1 by deletion of an edge, i.e. disconnection, and G_2 is obtained from G_1 by the contraction of that edge. Applying this operation recurrently to each of the graphs obtained as many times as possible, we stop when all graphs on the right side of the equation above are the empty graphs. If the connection-contraction algorithm leads to a combination of complete graphs, the disconnection-contraction algorithm leads to a combination of empty graphs.

An example of disconnection-contraction algorithm applied to the very same graph G, see Figure 5.5, is shown in Figure 5.9. One can see that the same chromatic polynomial may be obtained by two different ways.

The next theorem describes the behavior of the coefficients of the chromatic polynomial.

Theorem 5.4.1 (Whitney, 1933) *The chromatic polynomial $P(G,\lambda)$ is of degree $n(G)$, with integer coefficients alternating in sign and beginning with* $1, -m(G), \ldots$.

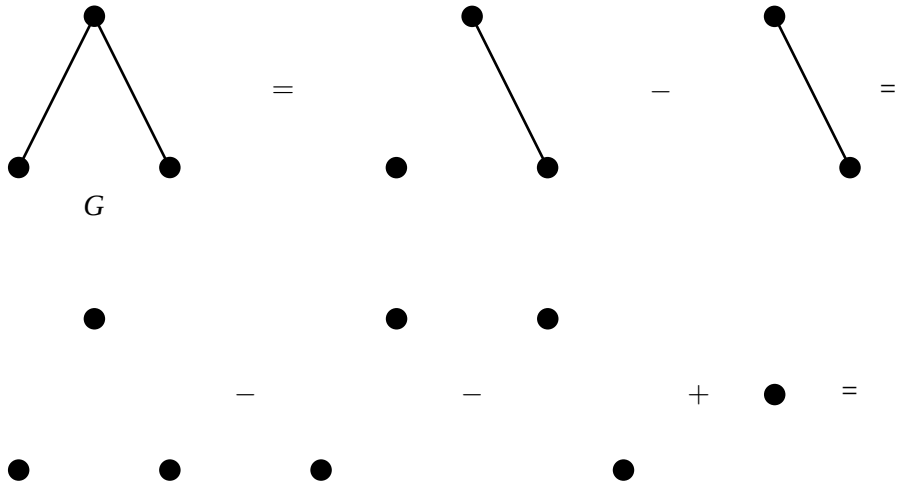
Proof. We prove by induction on the number of edges $m(G)$. The theorem holds for $m = 0$ because in this case $G = E_n$ and $P(G,\lambda) = \lambda^n$. Assume now that the theorem is true for all graphs with ¡ m edges and let G be an n-vertex graph with $m > 1$ edges. Consider arbitrary edge e of G. Graphs $G - e$ and $G \cdot e$ have fewer edges than G each. In addition, $G \cdot e$ has $n-1$ vertices. By the induction hypothesis, there exist nonnegative integer numbers $\{a_i\}$ and $\{b_i\}$ such that

$$P(G-e,\lambda) = \Sigma_{i=0}^{n}(-1)^i a_i \lambda^{n-i}$$

and

$$P(G\cdot e,\lambda) = \Sigma_{i=0}^{n-1}(-1)^i b_i \lambda^{n-1-i}.$$

Applying one step of disconnection-contraction to G we obtain the equality which proves the theorem:



$$E_3 - 2E_2 + E_1 \qquad \Rightarrow P(G,\lambda) = \lambda^3 - 2\lambda^2 + \lambda = \lambda(\lambda-1)^2$$

Figure 5.9. Example of disconnection-contraction.

$$\begin{aligned} P(G,\lambda) &= P(G-e,\lambda) - P(G\cdot e,\lambda) \\ &= \lambda^n - (m-1)\lambda^{n-1} + a_2\lambda^{n-2} - \cdots + (-1)^i a_i \lambda^{n-i} \cdots \\ &\quad -(\lambda^{n-1} - b_1\lambda^{n-2} + b_2\lambda^{n-2} - \cdots + (-1)^i b_{i-1}\lambda^{n-i}\cdots) \\ &= \lambda^n - m(G)\lambda^{n-1} + (a_2+b_1)\lambda^{n-2} \cdots (-1)^i (a_i + b_{i-1})\lambda^{n-i} \cdots . \end{aligned}$$ □

Lemma 5.4.1 *If a graph G contains a clique of size k, then the set of all proper colorings of G can be partitioned into $\lambda^{(k)}$ classes having $P(G,\lambda)/\lambda^{(k)}$ colorings each.*

Proof. Let $G = (X,E)$ be a connected graph with some $S \subseteq X$ inducing a clique of size k. Let us have λ available colors. Fix a proper coloring of S with λ colors and consider all proper λ-colorings of G that can be obtained by the extension of the coloring of S. Suppose that the number of such colorings is N_1. Fix now another proper coloring of S with λ colors and consider all proper λ-colorings of G that can be obtained by the extension of this new coloring of S. Denote the number of such colorings by N_2. We claim that $N_1 = N_2$. Indeed, every coloring from the second set of colorings can be obtained from a coloring from the first set (and vice versa) by a permutation of colors. Permutation of colors i and j means that all vertices colored i get color j and all vertices colored j get color i. Since S induces a clique, all colors are different, and any two colorings of S can be obtained from each other by a permutation of colors.

Let $t = P(G_S,\lambda) = P(K_k,\lambda) = \lambda^{(k)}$. Repeating the reasoning above for each of t colorings of S, we arrive to the conclusion that $N_1 = N_2 = N_3 = \cdots = N_t$. Hence the set of all $P(G,\lambda)$ colorings of G is partitioned into t equal classes. This implies that the number of colorings in each class is $P(G,\lambda)/\lambda^{(k)}$. □

Observe that the statement of the lemma does not hold if S does not induce a clique. If S has k vertices and is not a clique, then it has at least two nonadjacent vertices. The subgraph induced by S has two different colorings, one with k colors and another with $k-1$ colors. These colorings cannot be obtained from each other by a permutation of colors. When S induces a clique, it has the same unique feasible partition for every coloring of G.

Theorem 5.4.2 *Let $G=(X,E)$ be a connected graph having a separator S which is a clique of size k. Suppose $G_1 = G_{X_1\cup S}$ and $G_2 = G_{X_2\cup S}$ are the two derived subgraphs with respect to S. Then*

$$P(G,\lambda) = \frac{P(G_1,\lambda)P(G_2,\lambda)}{\lambda^{(k)}}. \tag{5.8}$$

Proof. Since G_S is a clique of size k in $G_{X_1\cup S}$, by Lemma 5.4.1 the set of all $P(G_{X_1\cup S},\lambda)$ colorings can be partitioned into $P(G_S,\lambda) = \lambda^{(k)}$ equal classes. Each class contains $P(G_{X_1\cup S},\lambda)/\lambda^{(k)}$ colorings. Similarly, the $P(G_{X_2\cup S},\lambda)$ colorings of $G_{X_2\cup S}$ can be partitioned into $\lambda^{(k)}$ equal classes, and each such class contains exactly $P(G_{X_2\cup S},\lambda)/\lambda^{(k)}$ colorings.

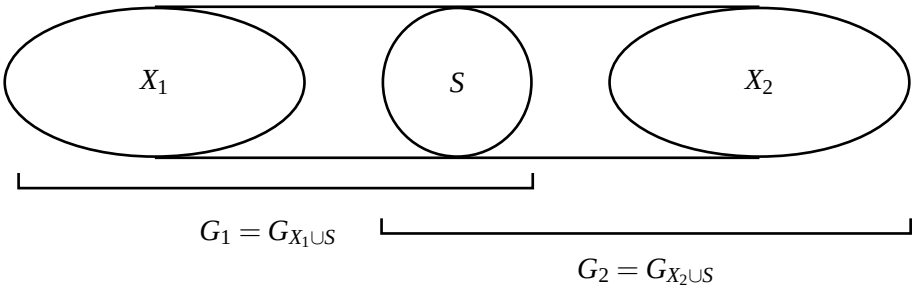


Figure 5.10. Separation and the chromatic polynomial.

Combining every coloring from each class of $G_{X_1\cup S}$ with every coloring from the corresponding class of $G_{X_2\cup S}$ gives a coloring of G, see Figure 5.10. Therefore, the total number of colorings of G is

$$P(G,\lambda) = \frac{P(G_{X_1\cup S},\lambda)}{\lambda^{(k)}}\frac{P(G_{X_2\cup S},\lambda)}{\lambda^{(k)}}\lambda^{(k)} = \frac{P(G_{X_1\cup S},\lambda)P(G_{X_2\cup S},\lambda)}{\lambda^{(k)}}.$$

□

Corollary 5.4.1 *Let $G=(X,E)$ be a connected graph having a separator S which is a clique of size k. Suppose $G_1, G_2, \ldots, G_l$ are the derived subgraphs with respect to the separator S. Then*

$$P(G,\lambda) = \frac{P(G_1,\lambda)P(G_2,\lambda)\ldots P(G_l,\lambda)}{[\lambda^{(k)}]^{l-1}}. \tag{5.9}$$

Proof. Indeed, since S belongs to every graph, we can repeat the same reasoning for each $G_i, i=1,2,\ldots,l$. The colorings of graphs all combine in every class of colorings generated by any single coloring of S. Since the number of classes is $\lambda^{(k)}$, the formula follows. □

Formula (5.5) may be regarded as a special case of the formula above if for a moment we accept the point of view that disconnected graph is like "connected" graph having a separator which is an empty set, and define $P(\emptyset,\lambda) = 1$.

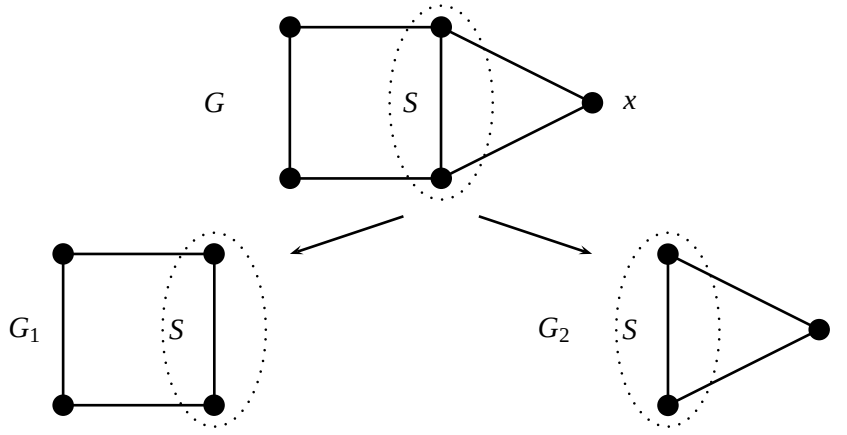


Figure 5.11.

An example to the theorem above is shown in Figure 5.11. Since $G_1 = C_4$, $G_2 = K_3$ and $|S| = 2$, we use the formula (5.4) for $P(C_4,\lambda)$ and $P(K_3,\lambda) = \lambda(\lambda-1)(\lambda-2)$ to compute the chromatic polynomial of the graph G:

$$P(G,\lambda) = \frac{P(G_1,\lambda)P(G_2,\lambda)}{\lambda^{(2)}} =$$

$$\frac{[\lambda(\lambda-1)(\lambda^2-3\lambda+3)][\lambda(\lambda-1)(\lambda-2)]}{\lambda(\lambda-1)} =$$

$$\lambda(\lambda-1)(\lambda-2)(\lambda^2-3\lambda+3).$$

Corollary 5.4.2 *If a connected graph G has a simplicial vertex x of degree k, then*

$$P(G,\lambda) = (\lambda-k)P(G-x,\lambda). \tag{5.10}$$

Proof. Suppose G is not a complete graph. Then the neighborhood $N(x)$ is a complete separator, i.e. a separator induced by a clique on k vertices. In this case $X_2 = \{x\}$, see Figure 5.10. Applying formula (5.8) with $G_1 = G-x$, $G_2 = K_{k+1}$, and $G_S = K_k$ we obtain:

$$P(G,\lambda) = \frac{P(G_1,\lambda)P(K_{k+1},\lambda)}{\lambda^{(k)}} =$$

$$\frac{P(G-x,\lambda)\lambda^{(k+1)}}{\lambda^{(k)}} = (\lambda-k)P(G-x,\lambda).$$

If G is a complete graph, then $G = K_{k+1}$, $G-x = K_k$ and the formula follows directly. □

We can apply formula (5.10) to graph G in Figure 5.11 and, since x is a simplicial vertex and $G_1 = G-x = C_4$, compute the chromatic polynomial of G directly:

$$P(G,\lambda) = (\lambda-2)P(G-x,\lambda) = \lambda(\lambda-1)(\lambda-2)(\lambda^2-3\lambda+3).$$

Theorem 5.4.3 *If a graph G has a simplicial vertex x of degree k, then*

$$r_i(G) = (i-k)r_i(G-x) + r_{i-1}(G-x).$$

Proof. Recall that $r_i(G)$ is the number of feasible partitions of graph G into i cells; it coincides with the number of strict i-colorings of G if we do not count the permutations of colors.

If vertex x is colored with one of the colors already used in $G-x$, then, because all colors in the neighborhood of x are different, we have $(i-k)r_i(G-x)$ such possibilities. If vertex x is colored with the color not used in $G-x$, then there are $r_{i-1}(G-x)$ possibilities. Hence the formula follows. □

Corollary 5.4.3 *If $S(n,i)$ is the Stirling number of the second kind, then*

$$S(n,i) = iS(n-1,i) + S(n-1,i-1).$$

Proof. Apply the theorem above to graph E_n: $r_i(E_n) = S(n,i)$ and each vertex may be regarded as a simplicial vertex of degree 0. □

Since trivially $r_1(E_1) = S(1,1) = 1$, the first five rows and columns of Stirling numbers are:

	1	2	3	4	5
1	1	0	0	0	0
2	1	1	0	0	0
3	1	3	1	0	0
4	1	7	6	1	0
5	1	15	25	10	1

One can see, for example, that

$$S(5,3) = 25 = 3 \cdot S(4,3) + S(4,2) = 3 \cdot 6 + 7 = 25.$$

Exercises 5.4.

1. Apply the disconnection-contraction algorithm to compute the chromatic polynomial of P_3, P_4, P_5, P_n for $n \geq 6$, for C_4, C_5, C_6, C_7, and for W_4, W_5, W_6.
2. Compute the Stirling numbers of the second kind corresponding to E_6, E_7, and E_8, and of the first kind corresponding to K_3, K_4, K_5, and K_6.
3. For each graph in Figure 5.12, by finding a simplicial vertex or a complete separator compute the chromatic polynomial.

Computer Projects 5.4. Write a program for the following algorithmic problem.

1. For any positive integers n and $i \leq n$, compute $S(n,i)$.
2. Find a complete separator in a graph.
3. Program the disconnection-contraction algorithm for: a) the cube; b) Petersen graph.

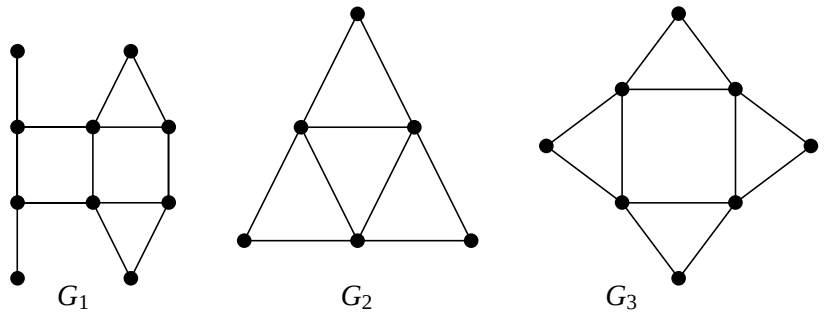


Figure 5.12.

5.5. Coloring Chordal Graphs

Chordal graphs have a very special place in graph coloring because the roots of their chromatic polynomials have nice properties. Recall that if G is a chordal graph, then it has a simplicial elimination ordering, i.e. it can be decomposed by sequential elimination of simplicial vertices.

Let $G_1 = (X,E)$, $|X| = n$, be a chordal graph, and $\sigma = (x_1, x_2, \ldots, x_n)$ be a simplicial elimination ordering. Let $G_2 = G_1 - x_1$, $G_3 = G_2 - x_2$, ..., $G_n = G_{n-1} - x_{n-1} = \{x_n\}$, such that $G_{n+1} = G_n - x_n = \emptyset$. It means that vertex x_i is a simplicial vertex of degree, say, k_i, in the graph G_i, $i = 1, \ldots, n$. Apply sequentially formula (5.10) to each of the graphs in this order:

$$P(G_1,\lambda) = (\lambda - k_1)P(G_2,\lambda) = (\lambda - k_1)(\lambda - k_2)P(G_3,\lambda)\cdots =$$
$$(\lambda - k_1)(\lambda - k_2)(\lambda - k_3)\ldots(\lambda - k_n).$$

We see that all the roots of $P(G_1,\lambda)$ are integer numbers equal to the degrees of the respective simplicial vertices. Since G_n consists only of one vertex x_n, conclude that $k_n = 0$ and we fix the first root $\lambda = 0$. If G_1 is disconnected, then in each component there will be a last vertex of degree 0; thus the multiplicity of the root $\lambda = 0$ equals the number of connected components of the graph G_1.

By the definition of clique number, G_1 contains a complete subgraph on $\omega(G_1)$ vertices, so maximum among all k_i's is $\omega(G_1) - 1$. We now fix the second root of the chromatic polynomial, namely, $\lambda = \omega(G_1) - 1$. Recall that $\omega(G_1) \le \chi(G_1)$ what implies that there is no coloring using less than ω colors. Consequently, every integer on the closed interval $[0, \omega - 1]$ is a root of $P(G_1,\lambda)$. Therefore, all roots k_i are the integer numbers from the interval $[0, \omega - 1]$, some of them may coincide (have multiplicity) but there are no gaps in these integers.

Finally, let us show that $\chi(G_1) = \omega(G_1)$ by coloring the vertices in the order inverse to σ, i.e. $x_n, x_{n-1}, x_{n-2}, \ldots, x_2, x_1$. In general case this procedure is called **online coloring** because the vertices of a graph can imaginably be put on the real line and colored from left to right (or from right to left). The main point is to color the vertices sequentially to obtain a proper coloring of a graph.

To do that, color vertex x_n with color 1. Then, if x_{n-1} is not possible to color with 1 (i.e., x_n and x_{n-1} are adjacent), color it with color 2. For x_{n-2} try to use color 1, if not possible,

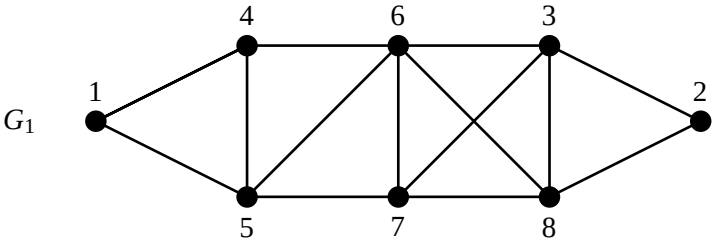


Figure 5.13.

try to use color 2, if not possible use the color 3. Continue this procedure each time trying to use the smallest possible color. If none of the used colors fits, use the new color. This is why sometimes the online coloring is called a **greedy coloring**. Since the maximum degree among k_i's is equal to $\omega - 1$, we eventually obtain a coloring of G_1 with ω colors. Hence, the conclusion is that $\omega(G) = \chi(G)$. All these facts can be summarized in

Theorem 5.5.1 *If G is a chordal graph, then the chromatic polynomial has the following form:*

$$P(G,\lambda) = \lambda^{s_0}(\lambda-1)^{s_1}(\lambda-2)^{s_2}\dots(\lambda-\chi(G)+1)^{s_{\chi-1}} \tag{5.11}$$

where $s_i \geq 1 (i = 0,1,\dots,\chi-1)$ is the number of simplicial vertices of degree i in the simplicial elimination ordering of G.

Consider now how this theorem works on the same example of a chordal graph shown in Figure 3.3, see Figure 5.13. As we have seen, G_1 has a simplicial elimination ordering $\sigma = (1,2,3,4,5,6,7,8)$. The degrees of simplicial vertices in elimination σ are: 2, 2, 3, 2, 2, 2, 1, 0. These are the roots of the chromatic polynomial, they all are from the interval $[0,3]$. Therefore, the chromatic polynomial

$$P(G_1,\lambda) = (\lambda-2)(\lambda-2)(\lambda-3)(\lambda-2)(\lambda-2)(\lambda-2)(\lambda-1)(\lambda-0) = \lambda(\lambda-1)(\lambda-2)^5(\lambda-3). \tag{5.12}$$

Vertices 3, 6, 7 and 8 form the unique maximum clique, so $\omega(G_1) = \chi(G_1) = 4$. Online coloring in the order 8, 7, 6, 5, 4, 3, 2, 1 (inverse to σ) produces a proper coloring using four colors, see Figure 5.14 (now the numbers are colors). There are other colorings with four colors; the total number of 4-colorings, for example, is:

$$P(G_1,4) = 4(4-1)(4-2)^5(4-3) = 384.$$

It would be difficult to find this number of colorings manually by exhaustive search, or even using connection-contraction algorithm developed for all graphs; structure of chordal graphs provides an efficient algorithm for computing the number of proper colorings for any number of available colors. But nice coloring properties of chordal graph go much beyond this. The rest of the section is devoted to such facts that are unimaginable for general graphs. We begin with

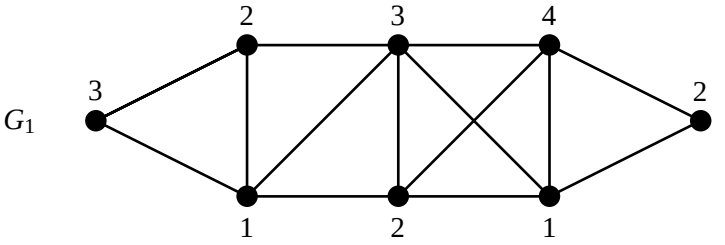


Figure 5.14. Online coloring of chordal graph.

Corollary 5.5.1 *If T_n is a tree on n vertices, then*

$$P(T_n,\lambda) = \lambda(\lambda-1)^{n-1}. \tag{5.13}$$

Proof. Trees are special case of chordal graphs, and pendant vertices are the special case of simplicial vertices. Decompose T_n by sequential elimination of pendant vertices and apply formula (5.11). We obtain $n-1$ vertices of degree 1 and the last vertex of degree 0, i.e. $k_1 = k_2 = \cdots = k_{n-1} = 1$, $k_n = 0$. □

Lemma 5.5.1 *For every cycle $C_n, n \geq 1$*

$$P(C_n,\lambda) = (\lambda-1)^n + (-1)^n(\lambda-1). \tag{5.14}$$

Proof. Induction on the number of vertices n. For $n = 1,2,3$ verify directly (though $C_1$and C_2 are not simple graphs, the formula holds). Assume the formula holds for any cycle on $< n$ vertices. Choose any edge of C_n and apply disconnection-contraction:

$$P(C_n,\lambda) = P(T_n,\lambda) - P(C_{n-1},\lambda) =$$
$$\{\text{apply formula (5.13) and the induction hypothesis}\} =$$
$$\lambda(\lambda-1)^{n-1} - [(\lambda-1)^{n-1} + (-1)^{n-1}(\lambda-1)] =$$
$$(\lambda-1)^n + (-1)^n(\lambda-1).$$ □

We now are able to state the following criterion for chordal graphs:

Theorem 5.5.2 *A graph G is chordal if and only if for any induced subgraph G' (including G itself) the chromatic polynomial has the following form:*

$$P(G',\lambda) = \lambda^{s'_0}(\lambda-1)^{s'_1}\ldots(\lambda-\chi'+1)^{s'_{\chi'-1}},$$

where $\chi' = \chi(G')$, and $s'_i \geq 1$ $(i = 0,1,\ldots,\chi'-1)$.

Proof. $\Rightarrow$ Every induced subgraph of a chordal graph is chordal as well. Apply Theorem 5.5.1.

$\Leftarrow$ Suppose the chromatic polynomial of any induced subgraph of G has the required form. If G is not a chordal graph, then it contains a cycle $C_k, k \geq 4$ as an induced subgraph. It is easy to see that $\chi(C_k) = 2$ if k is even and $\chi(C_k) = 3$ if k is odd. Hence all the roots of $P(C_k,\lambda)$ should be from the set $\{0,1,2\}$. However by Lemma 5.5.1 $P(C_k,\lambda) = (\lambda-1)^k +$

$(-1)^k(\lambda-1)$, and thus $P(C_k,\lambda)$ is a polynomial of degree $k\geq 4$. One can prove that this polynomial has at least one complex root; a contradiction. □

Let $G=(X,E)$ be a graph, $A\subset X$ be an arbitrary separator, $G_1^*=(X_1,E_1)$, $G_2^*=(X_2,E_2),\ldots,G_k^*=(X_k,E_k)$, $k\geq 2$, be the connected components obtained after removing vertex set A together with all incident edges from G. If G is a disconnected graph having l connected components, then we assume that $k>l$. So, we have $X_1\cup X_2\cup\ldots\cup X_k\cup A=X$, $X_i\cap X_j=\emptyset$, $i\neq j$. As usually, denote the derived induced subgraphs in the following way:

$$G_{X_1\cup A}=G_1,\ G_{X_2\cup A}=G_2,\ \ldots,\ G_{X_k\cup A}=G_k,\ G_A=G_0.$$

Theorem 5.5.3 *A graph $G=(X,E)$ is chordal if and only if, for any induced subgraph $G'=(X',E')$ (including G itself) and any separator $A'\subset X'$ of G',*

$$P(G',\lambda)=\frac{P(G_1',\lambda)P(G_2',\lambda)\ldots P(G_k',\lambda)}{P(G_0',\lambda)^{k-1}}. \tag{5.15}$$

Proof. $\Rightarrow$ Since every induced subgraph of a chordal graph is chordal too, we prove the statement for $G'=G$. We proceed by induction on $|X|=n$. The cases $n=2,3,4$, can be verified directly. Let the statement be true for all chordal graphs with fewer than n vertices where $n>4$. Let $n(G)=n$, and $x_0\in X$ be a simplicial vertex of degree p in G. There are two possible cases.

Case 1. $x_0\notin A$. Suppose that $x_0\in X_1$. Since A is a separator and x_0 is a simplicial vertex, the neighborhood $N(x_0)\subseteq X_1\cup A$. Therefore, x_0 is a simplicial vertex in G_1. Applying formula (5.10) to G_1, we obtain:

$$P(G_1,\lambda)=(\lambda-p)P(G_1-x_0,\lambda).$$

Applying the same formula to G, we have:

$$P(G,\lambda)=(\lambda-p)P(G-x_0,\lambda).$$

Notice that A is a separator in $G-x_0$ with the same number of derived subgraphs. Since $n(G-x_0)<n$ by the induction hypothesis,

$$P(G-x_0,\lambda)=\frac{P(G_1-x_0,\lambda)P(G_2,\lambda)\ldots P(G_k,\lambda)}{P(G_0,\lambda)^{k-1}}. \tag{5.16}$$

Multiplying both sides of (5.16) with the factor $(\lambda-p)$, we obtain

$$P(G,\lambda)=\frac{P(G_1,\lambda)P(G_2,\lambda)\ldots P(G_k,\lambda)}{P(G_0,\lambda)^{k-1}}.$$

Case 2. $x_0\in A$, see Figure 5.15. Let $|N(x_0)\cap A|=p_1$. Since $N(x_0)$ induces a complete graph, the remaining $p-p_1=p_2$ vertices from $N(x_0)$ belong to at most one, say X_1, of the sets $X_i, 1\leq i\leq n$. Then

$$\{x_0\}\cup N(x_0)\subseteq X_1\cup A,\ N(x_0)\cap A\subseteq X_i\cup A,\ 1\leq i\leq k.$$

One can see that the vertex x_0 is simplicial in G_1 with degree p and in all subgraphs G_i, $i=0,2,3,\ldots,k$ with degree p_1. Apply formula (5.10) to G:

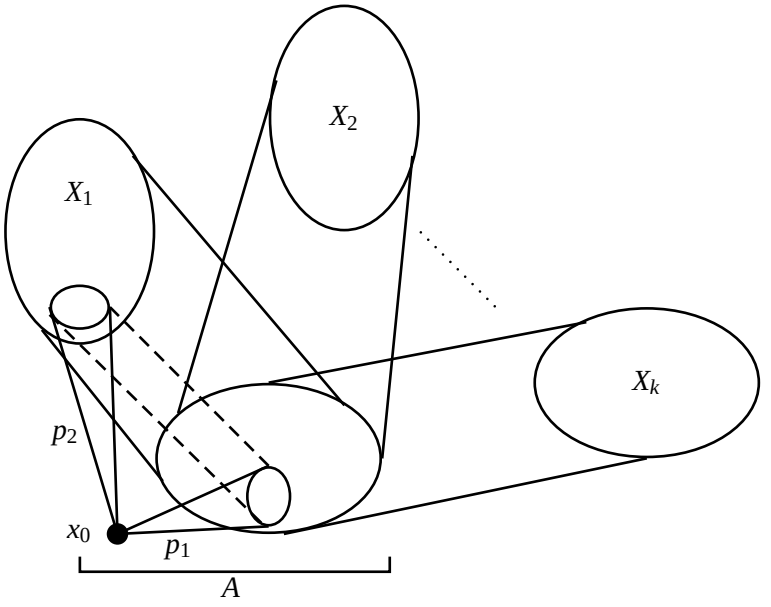


Figure 5.15.

$$P(G,\lambda) = (\lambda - p)P(G - x_0,\lambda).$$

Notice that set $A - x_0$ is a separator in $G - x_0$ with the same number of derived subgraphs. Since $n(G - x_0) < n$, by the induction hypothesis we have

$$P(G - x_0,\lambda) = \frac{P(G_1',\lambda)P(G_2',\lambda)\dots P(G_k',\lambda)}{P(G_0',\lambda)^{k-1}}, \tag{5.17}$$

where the graphs G_i' are obtained from G_i by deletion of vertex $x_0,\ i = 0,1,\dots,k$.
On the other hand,

$$P(G_1,\lambda) = (\lambda - p)P(G_1',\lambda),\ P(G_i,\lambda) = (\lambda - p_1)P(G_i',\lambda),$$
$$i = 0,2,\dots,k.$$

Again it follows immediately that

$$P(G,\lambda) = \frac{P(G_1,\lambda)P(G_2,\lambda)\dots P(G_k,\lambda)}{P(G_0,\lambda)^{k-1}}.$$

$\Leftarrow$ For a contradiction, suppose G is not chordal. Then it contains a cycle $C_k, k \geq 4$, as an induced subgraph. Let x be a vertex of C_k. Since $N(x)$ is a separator, by formula (5.15), we obtain

$$P(C_k,\lambda) = \lambda(\lambda-1)^2\lambda(\lambda-1)^{k-2}\lambda^{-2} = (\lambda-1)^k \neq P(C_k,\lambda),$$

a contradiction. □

As we have seen, formula (5.15) is known to be true for arbitrary graphs provided a separator induces a complete graph, see formula (5.9). The main feature of Theorem 5.5.3

is that in chordal graphs it holds for any separator. One can see that (5.15) holds for disconnected G and G_A (provided the number of connected components increases) or even for the empty separator. If we accept $P(\emptyset,\lambda)=1$, then (5.15) turns into the formula for the chromatic polynomial of a disconnected graph G with connected components $G_1,\ldots,G_k$ (case $A=\emptyset$).

From this point of view, chordal graphs have so nice structure of colorings that they mysteriously look like "complete graphs having separators". However, (5.15) has an additional final important consequence. Namely, it implies the universal formula for computing the chromatic polynomial of a chordal graph using an **arbitrary elimination ordering** what is impossible for general graphs.

For a graph G, define the function

$$W(G,\lambda)=\frac{\lambda P(G,\lambda-1)}{P(G,\lambda)}.$$

Theorem 5.5.4 (universal formula) *A graph $G=(X,E)$ is chordal if and only if in every connected induced subgraph $G'=(X',E')$ for any vertex $x\in X'$ the following equality holds:*

$$P(G',\lambda)=P(G'-x,\lambda)W(G'_{N(x)},\lambda). \tag{5.18}$$

Proof. $\Rightarrow$ Suppose G is a chordal graph. Since every induced subgraph of a chordal graph is chordal, we prove the necessity for $G'=G$. Hence, we may assume that $G=(X,E)$ is a connected chordal graph and $x\in X$ is an arbitrary vertex.

Case 1. $X=\{x\}\cup N(x)$. Since in any coloring of G with λ colors vertex x requires a separate color,

$$P(G,\lambda)=\lambda P(G-x,\lambda-1)=\frac{P(G-x,\lambda)}{P(G-x,\lambda)}\lambda P(G-x,\lambda-1)=$$
$$P(G-x,\lambda)W(G_{N(x)},\lambda).$$

Case 2. $X\neq\{x\}\cup N(x)$. Since G is connected, the subgraph $G_{N(x)}$ is a separator. Apply equality (5.15):

$$P(G,\lambda)=\frac{P(G-x,\lambda)P(G_{\{x\}\cup N(x)},\lambda)}{P(G_{N(x)},\lambda)}=$$

$$P(G-x,\lambda)\frac{\lambda P(G_{N(x)},\lambda-1)}{P(G_{N(x)},\lambda)}=P(G-x,\lambda)W(G_{N(x)},\lambda).$$

$\Leftarrow$ Suppose G is not chordal and the formula (5.18) holds for any connected induced subgraph. Then it contains an induced cycle $C_k, k\geq 4$. Let x be a vertex of C_k which is denoted by just C. Observe that $C-x$ is a tree on $k-1$ vertices, and $C_{N(x)}=E_2$. We see that $P(C-x,\lambda)=\lambda(\lambda-1)^{k-2}$, $W(C_{N(x)},\lambda)=\lambda^{-1}(\lambda-1)^2$, and by the formula

$$P(C,\lambda)=P(C-x,\lambda)W(C_{N(x)},\lambda)=(\lambda-1)^k\neq P(C,\lambda),$$

a contradiction. $\square$

Corollary 5.5.2 *If x is a simplicial vertex of degree $k \geq 0$ in a chordal graph G, then*

$$P(G,\lambda) = (\lambda - k)P(G - x,\lambda).$$

Proof. Indeed, $N(x) = K_k$, so

$$W(G_{N(x)},\lambda) = W(K_k,\lambda) = \frac{\lambda(\lambda-1)^{(k)}}{\lambda^{(k)}} = (\lambda - k).$$

Applying universal formula (5.18) we obtain

$$P(G,\lambda) = P(G-x,\lambda)W(G_{N(x)},\lambda) = (\lambda-k)P(G-x,\lambda). \qquad \square$$

So, decomposition of chordal graphs using a simplicial elimination ordering is a special case of a general procedure of decomposition by eliminating vertices in an arbitrary order and applying the universal formula (5.18).

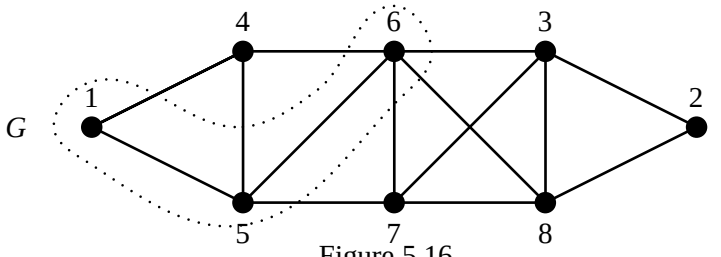


Figure 5.16.

Let us consider how the last theorems work on the example of graph G, see Figure 5.16. Suppose we delete non-simplicial vertex 4 and obtain graph G_1. To compute the chromatic polynomial $P(G,\lambda)$ from G_1, by the universal formula we need to multiply $P(G_1,\lambda)$ by the function $W(G_{N(4)},\lambda)$. Since $N(4) = \{1,5,6\}$, the induced subgraph is a tree on three vertices (encircled by a dotted curve in the figure). Hence

$$W(G_{N(4)},\lambda) = \frac{\lambda P(T_3,\lambda-1)}{P(T_3,\lambda)} = \frac{\lambda(\lambda-1)(\lambda-2)^2}{\lambda(\lambda-1)^2} = \frac{(\lambda-2)^2}{\lambda-1}.$$

If we decompose G_1 by simplicial elimination in ordering 1, 5, 6, 7, 8, 3, 2, the degrees of simplicial vertices respectively are: 1, 2, 3, 2, 2, 1, 0. We compute

$$P(G_1,\lambda) = \lambda(\lambda-1)^2(\lambda-2)^3(\lambda-3).$$

Now

$$P(G_1,\lambda)W(G_{N(4)},\lambda) = \lambda(\lambda-1)^2(\lambda-2)^3(\lambda-3)\frac{(\lambda-2)^2}{\lambda-1} =$$

$$\lambda(\lambda-1)(\lambda-2)^5(\lambda-3) = P(G,\lambda)$$

as it was found in formula (5.12).

Exercises 5.5.

1. For graph G in Figure 5.16, compute the chromatic polynomial using separator $\{5,6,7,8\}$ and respective derived subgraphs.
2. For graph G in Figure 5.16, apply the universal formula for vertex 7.
3. Construct an example of a chordal graph and an ordering of vertices such that online coloring in that order does not give the minimum number of colors.

Computer Projects 5.5. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. Given a graph G, recognize if G is chordal.
2. Given a graph G, recognize if G is chordal, and if yes, compute the chromatic polynomial.
3. Given a graph G, recognize if G is chordal, and if yes, construct an optimal coloring.

5.6. Coloring Planar Graphs

Recall that for a graph $G = (X,E)$, the Szekeres-Wilf number is

$$M(G) = max_{X' \subseteq X} min_{x \in G'} d(x)$$

where $d(x)$ is the degree of vertex x in a subgraph G' induced by the subset of vertices X'. As we mentioned at the beginning of section 3.3., this number can easily be found by sequential elimination of vertices of minimum degrees; the maximum degree obtained in this procedure is $M(G)$.

Theorem 5.6.1 *For any graph G, $\chi(G) \leq M(G)+1$.*

Proof. Decompose G by a sequential elimination of vertices of minimum degree. We obtain a sequence of graphs $G_1 = G$, G_2, G_3, ..., G_n and corresponding sequence of vertices $x_1, x_2, \ldots, x_n$ such that each vertex x_i is a vertex of minimum degree in a graph G_i, $i = 1, \ldots, n$. We now apply online coloring to G by reconstructing it in ordering $x_n, x_{n-1}, \ldots, x_2, x_1$. That means we color the vertices in order $x_n, x_{n-1}, \ldots, x_2, x_1$ each time using the smallest suitable color. In this sequence, the maximum number of colored neighbors for vertex x_i is $M(G)$, see Figure 5.17. The worst case occurs when all $M(G)$ neighbors of x_i have distinct colors (otherwise, we just assign to x_i the first missing color). We then assign $(M(G)+1)th$ color to x_i and continue online coloring. If even the worst case occurs several times, the total number of colors used in the obtained proper coloring does not exceed $M(G)+1$. □

Corollary 5.6.1 *If G is a planar graph, then $\chi(G) \leq 6$.*

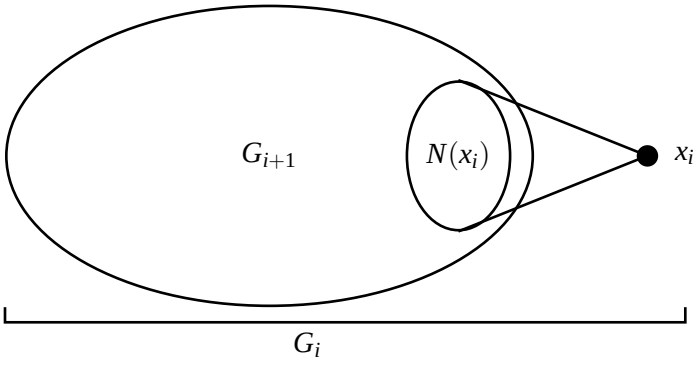


Figure 5.17.

Proof. Recall that by Corollary 4.2.3, G contains a vertex of degree at most 5. Any subgraph of G is a planar graph. Therefore, in any decomposition of G by vertices of minimum degrees, the degrees do not exceed 5. Hence $M(G) \leq 5$. Applying Theorem 5.6.1 results in $\chi(G) \leq 6$. □

Theorem 5.6.2 (Five Color Theorem, Heawood, 1890) *If G is a planar graph, then* $\chi(G) \leq 5$.

Proof. We prove the theorem by induction on $n(G) = n$. Evidently, all planar graphs having ≤ 5 vertices are 5-colorable (color all vertices differently).

Assume now that all planar graphs having $< n$ vertices are 5-colorable. We will prove that under this assumption, $\chi(G) \leq 5$. By Corollary 4.2.3, G contains a vertex x of degree $d(x) \leq 5$. By the induction hypothesis, $\chi(G-x) \leq 5$. Consider a proper 5-coloring of $G-x$. By this, all the vertices of $N(x)$ get some colors. We will show that there is always a way to assign a color to vertex x such that G is 5-colorable.

Case 1: the number of colors used in $N(x)$ is $\leq$4. We assign the 5*th* color to x and obtain a 5-coloring of G.

Case 2: the number of colors used in $N(x)$ is 5. It means that $d(x) = 5$. The main idea of the rest of the theorem is to show that we can construct another proper 5-coloring of $G-x$ which uses only 4 colors in the neighborhood $N(x)$.

The situation is equivalent to that shown in Figure 5.17 for $G_i = G$, but now it can be depicted more specifically in Figure 5.18. Without loss of generality, denote $N(x) = \{a,b,c,d,e\}$ and respective colors by 1, 2, 3, 4 and 5.

Subcase 2.1: in the 5-coloring of $G-x$, there exists an (a,c)-path P with vertices colored by alternating colors 1 and 3 as shown in Figure 5.18 (numbers are the colors). Path P along with edges ax and xc form a cycle C in the plane with all vertices colored 1 and 3, and vertex x uncolored. Switch the colors 2 and 4 on vertices located in the plane inside cycle C; such re-coloring does not affect vertices colored 2 and 4 which are outside C. Therefore, vertex d preserves color 4. We obtain another proper 5-coloring of $G-x$ where vertices d and b are colored with color 4. In this new coloring, color 2 is missing in $N(x)$; assign color 2 to vertex x and obtain a proper 5-coloring of G.

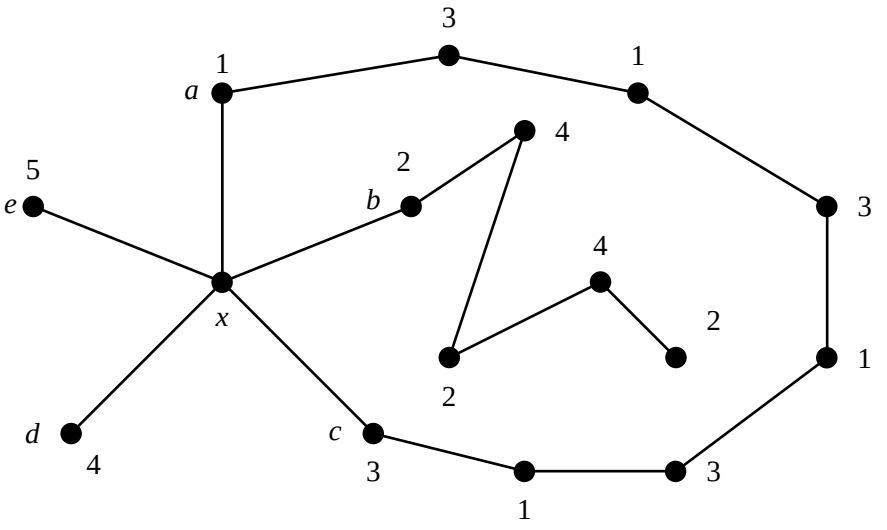


Figure 5.18.

Subcase 2.2: in the 5-coloring of $G-x$, there is no (a,c)-path with vertices colored by alternating colors 1 and 3. Let G_{13} be a subgraph of $G-x$ induced by the vertices colored with colors 1 and 3. G_{13} is a disconnected graph and vertices a and c are in different components. Switch the colors 1 and 3 in the component containing vertex a. We obtain a proper 5-coloring of $G-x$ where color 1 is missing in $N(x)$. Assign color 1 to vertex x and obtain a proper 5-coloring of G, what proves the theorem. □

Theorem 5.6.3 (Four Color Theorem, Appel, Haken, Koch, 1977) *If G is a planar graph, then $\chi(G) \leq 4$.*

Kempe's proof, 1879, the most famous error in the history of Graph Theory. We "prove" the theorem by induction on $n(G) = n$. Since K_5 is not a planar graph, all planar graphs having ≤ 5 vertices are 4-colorable.

Assume now that all planar graphs having $< n$ vertices are 4-colorable. We will "prove" that under this assumption, $\chi(G) \leq 4$. Observe that any plane graph can be complete to a triangulation, i.e. a plane graph with all faces being triangles by just adding some edges. If we prove that any triangulation is 4-colorable, it will imply that any plane graph is 4-colorable, too. Therefore, without loss of generality we assume that G is a triangulation.

By Corollary 4.2.3, G contains a vertex x of degree $d(x) \leq 5$. By the induction hypothesis, $\chi(G-x) \leq 4$. Consider a proper 4-coloring of $G-x$. By this, all the vertices of $N(x)$ get some colors. We "will show" that there is always a way to assign a color to vertex x such that G is 4-colorable.

Case 1: the number of colors used in $N(x)$ is ≤ 3. We assign the $4th$ color to x and obtain a 4-coloring of G.

Case 2: the number of colors used in $N(x)$ is 4. It means that $d(x) = 4$ or $d(x) = 5$.

Subcase 2.1: $d(x) = 4$. Apply the reasoning from Theorem 5.6.2 for $G' = G - e$ (in fact, vertex e and color 5 was never used in that proof).

Subcase 2.2: $d(x) = 5$. Since the number of colors used in $N(x)$ is 4, two vertices get the same color. Denote $N(x) = \{a,b,c,d,e\}$ and respective colors by 1, 2, 3, and 4. Since

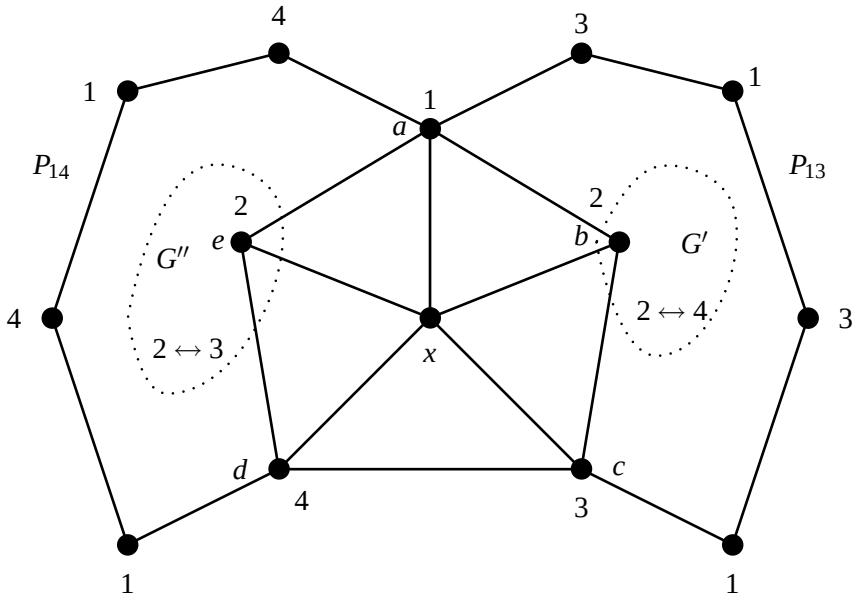


Figure 5.19.

we assumed that G is a triangulation, the vertices of the same color cannot be consecutive in the order a,b,c,d,e. Assume that the colors are distributed as shown in Figure 5.19. Let G_{ij} denote the subgraph induced by vertices colored i and j, and P_{ij} denote any path having all vertices colored with alternating (along the path) colors i and j.

Subcase 2.2.1: in the 4-coloring of $G-x$, there is no (a,c)-path P_{13} or there is no (a,d)-path P_{14}. For example, assume there is no (a,c)-path P_{13}. Then the subgraph G_{13} is disconnected, and vertices a and c are in different components. Switching the colors 1 and 3 in the component containing vertex a eliminates color 1 from $N(x)$; we then assign color 1 to x and obtain a proper 4-coloring of G. Similar reasoning applies if there is no (a,d)-path P_{14}.

Subcase 2.2.2: in the 4-coloring of $G-x$, both (a,c)-path P_{13} and (a,d)-path P_{14} exist, see Figure 5.19. Observe that the component G' of G_{24} containing vertex b is separated from vertices d and e by the cycle completed by (a,c)-path P_{13} and edges ax and xc. Similarly, the component G'' of G_{23} containing vertex e is separated from vertices b and c by the cycle completed by (a,d)-path P_{14} and edges ax and xd. We now permute color 2 with 4 in G' and color 2 with 3 in G'' . This recoloring eliminates color 2 from $N(x)$. Assign color 2 to vertex x and obtain a proper 4-coloring of G, what "proves" the theorem. □

Where is the trap in this reasoning by Kempe? An answer to this question is depicted in Figure 5.20. The problem is that in Subcase 2.2.2, (a,c)-path P_{13} and (a,d)-path P_{14} may intersect at a vertex of color 1. Then the permutation of color 2 with 4 in G' and of color 2 with 3 in G'' leads to a pair of adjacent vertices colored with color 2 (marked by "?"), i.e. to a non proper coloring of G.

The proof above was published by Kempe in 1879, and the error is very instructive. On one hand, it shows an excellent example how drawings may be used in Graph Theory proofs. On the other hand, it explicitly exhibits the limits of drawings. Since the result was

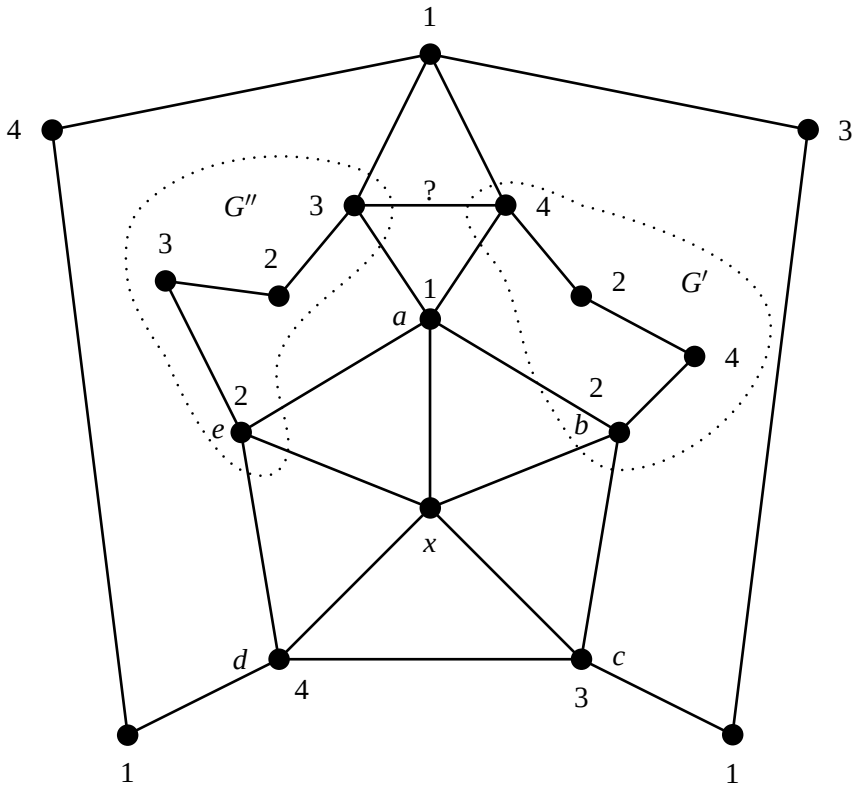


Figure 5.20. The most famous error in Graph Theory.

considered proven until 1890, it also demonstrated the fact that mathematicians of those times liked writing papers more than reading them. But on the top of all that, nobody could even imagine what a dramatic history was ahead.

Formulated first in 1852 by Francis Guthrie (a student (!) of de Morgan), "proved" in 1879 by Kempe, refuted in 1890 by Heawood, the Four Color Problem became one of the most famous problems in Discrete Mathematics in 20th century before in 1977 it became the Four Color Theorem by Appel, Haken, and Koch. Besides many erroneous proofs, it generated many new directions in Graph Theory. For example, only one sub-direction of chromatic polynomials introduced by Birkhoff in 1912 with the aim to solve the problem by algebraic methods counts more than five hundred research papers.

The main idea of the final proof is quite simple - by induction on the number of vertices; but the number of cases is huge. Though the Kempe's proof was erroneous, his idea of alternating paths and further re-coloring of the respective subgraphs was used in the final proof. A path on which two colors alternate is called a **Kempe chain.** In a plane triangulation, a **configuration** is a derived subgraph with respect to a separating cycle; it is a subgraph induced by the cycle and all the vertices which are inside the cycle in the plane. For example, in Figure 5.20, the subgraph induced by vertices a, b, c, d, e and x is a configuration. It is basically the same idea that was used in chordal graphs, or more generally, any graphs

having separators, see Figures 1.32, 5.8, 5.17. In chordal graphs, separators were cliques, in triangulations, separators are cycles.

In a plane triangulation, a configuration is **reducible** if any 4-coloring of the cycle can be extended to a 4-coloring of the entire triangulation. There were the following two basic steps in the proof:

1. Proof that any plane triangulation contains a configuration from a list of **unavoidable** configurations.

2. Proof that each unavoidable configuration is reducible.

In 1976, Appel, Haken, and Koch, using 1,200 hours of computer time, found 1936 (!) unavoidable configurations and proved that all they are reducible. Historically, it was the first time when a famous mathematical problem was solved by extensive use of computers. The final accord in this one-century drama was when the result was widely announced, the paper was published in 1977, and the University of Illinois even announced it by postage meter stamp of "four colors suffice", a few errors were found in the original proof. Fortunately for the authors, they have been fixed. For the first time, regardless the fact that there is no human being who would check the entire proof (because it contains steps that most likely can never be verified by humans), the problem is considered completely solved.

In 1996, Robertson, Sanders, Seymour and Thomas improved the proof by finding the set of only (!) 633 reducible configurations. Computers, Kempe chains, and some other techniques were used in both proofs.

The three consecutive theorems - Corollary 5.6.1 (six colors), Theorem 5.6.2 (five colors) and Theorem 5.6.3 (four colors) show the main feature of graph coloring: there is a very simple proof for six colors, a relatively simple proof for five colors and an incredible difficult and complex proof for four colors. The Four Color Problem, formulated by a student, was first "solved" by a lawyer, and really solved with many contributions of the very prominent mathematicians of the century.

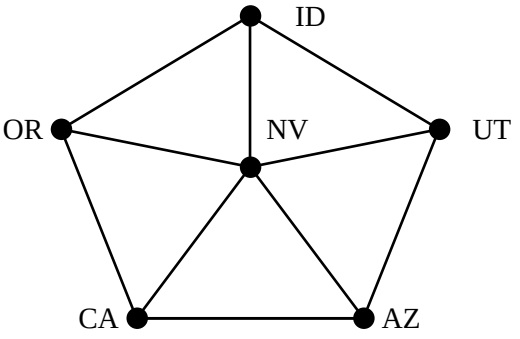


Figure 5.21.

So we can conclude that every geographic map can be colored with at most four colors. Sometimes one can use less colors. What about the map of USA? Four Color Theorem implies that it can be colored with four colors. But can it be colored with 3 colors? The answer is no. Indeed, take for example the state of Nevada and five its neighbors: Idaho, Utah, Arizona, California and Oregon. One can easily check, see Figure 5.21, that their

adjacency forms the wheel W_6 for which $\chi(W_6) = 4$. West Virginia and Kentucky are also in the center of W_6 and W_8 respectively. Hence a planar graph corresponding to the entire map of USA contains subgraphs which need four colors each anyway. It means that precisely four colors is the minimum for the entire map of USA.

Exercises 5.6.

1. Compute $M(G)$ where G is: P_n, K_n, $K_{m,n}$, C_n, W_n, $m,n \geq 4$, tree, Petersen graph, cube and prism, find the bound on the chromatic number and the respective coloring.

2. Arbitrarily draw any number of straight lines in the plane. Use mathematical induction to prove that the obtained regions can be colored with two colors in such a way that no two of them with a common segment of line have the same color.

3. Color the map of USA with four colors.

4. Construct a planar graph G with no subgraphs isomorphic to $W_n, n \geq 4$, such that $\chi(G) = 4$.

Computer Projects 5.6. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. Given a graph G, find $M(G)$, bound on the chromatic number, and the respective coloring.

2. Given a planar graph G, find an optimal coloring.

5.7. Perfect Graphs

Any graph G contains a maximum clique which by definition has $\omega(G)$ vertices. Since its vertices are pairwise adjacent, any proper coloring of G requires at least ω colors, i.e.

$$\chi(G) \geq \omega(G). \tag{5.19}$$

There are graphs with $\chi(G) = \omega(G)$, and there are graphs for which $\chi(G) > \omega(G)$. For example, $\chi(C_4) = \omega(C_4) = 2$ but $\chi(C_5) = 3 > \omega(C_5) = 2$. How large can be the difference $\chi(G) - \omega(G)$? The answer is: it can be arbitrary large.

Consider the so called **Mycielski's construction.** Two vertices are called **copies** of each other if they have the same neighborhoods and are not adjacent. Respectively, copying a vertex means adding a new vertex adjacent to all neighbors of a given vertex. A vertex is called **universal** if it is adjacent to all other vertices of a graph. In a complete graph all vertices are universal. Generally, adding a new vertex to a graph and making it universal increases the size of maximum clique by 1. It also increases the chromatic number by 1 because the universal vertex requires a new color.

Start with graph K_2. Copy the vertices, see Figure 5.22, encircled. Then add a new vertex adjacent to all the copies. The last vertex is "universal" to the copies but its addition

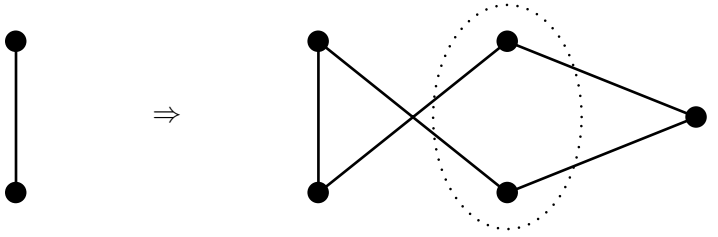


Figure 5.22.

does not create triangles and therefore does not increase the clique number. However, it increases the chromatic number. This is the main trick of the construction. Notice that the graph obtained is C_5.

Now repeat the procedure for C_5: copy each vertex and add a new vertex adjacent to all of the copies. It is convenient to draw the copies and the "universal" vertex inside the cycle, see Figure 5.23. This graph is called the **Grötzsch graph**. For it $\chi = 4$, and evidently, $\omega = 2$. One can prove that if we repeat the procedure k times, we obtain a graph with $\chi - \omega = k$. It means that from theoretical point of view, there exist graphs with arbitrary large difference $\chi - \omega$ which are triangle-free. If $\omega(G)$ is the lower bound for $\chi(G)$, then what are the graphs for which $\chi(G) = \omega(G)$?

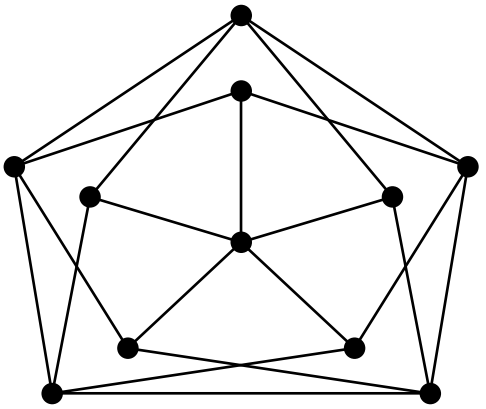


Figure 5.23.

In Graph Theory, when we have some property for a graph, it is convenient to require this property to hold for each induced subgraph. The main motivation is that in such case we can apply mathematical induction by the number of vertices and prove many results.

Definition 5.7.1 *A graph G is* **perfect** *if $\chi(G') = \omega(G')$ for any induced subgraph G' including G itself.*

It appears that perfection of graphs is related to the complements of graphs. Recall that the clique cover number $\theta(G)$ of a graph G is the minimum number of cliques in graph G such that each vertex belongs to precisely one clique. We say that the cliques "cover" the vertex set of G. Since every clique in G induces an independent set in complement $\overline{G}$, we conclude that every coloring of G is equivalent to a clique covering of $\overline{G}$, and thus

$\chi(G) = \theta(\overline{G})$. Therefore, if $\overline{G}$ is a perfect graph, then $\theta(G') = \alpha(G')$ holds for any induced subgraph G' of G including G itself.

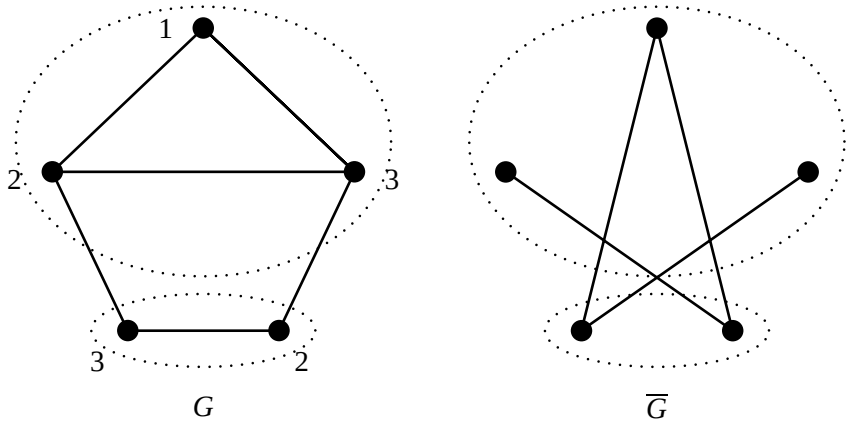


Figure 5.24.

For example, a graph G in Figure 5.24 has $\chi(G) = 3, \omega(G) = 3$, and $\alpha(G) = 2$ and $\theta(G) = 2$. A proper 3-coloring is shown by numbers, and clique covering is shown by dotted ellipses. Respectively, for its complement $\overline{G}$, we have $\chi(G) = 2, \omega(G) = 2, \alpha(G) = 3$ and $\theta(G) = 3$. One can easily see that clique covering of G by two cliques corresponds to a proper 2-coloring of $\overline{G}$ and so on. It is also easy to check that G and $\overline{G}$ both are perfect graphs. An example of an imperfect graph is C_5: $\chi(C_5) = 3, \omega(C_5) = 2$, $\alpha(C_5) = 2$ and $\theta(C_5) = 3$ (recall that C_5 is isomorphic to $\overline{C_5}$). In these examples graphs and their complements behave similarly with respect to graph perfection. Is this the rule?

May we have the equality $\chi(G') = \omega(G')$ for all induced subgraphs G' and inequality $\theta(G') \neq \alpha(G')$ for at least one induced subgraph? In other words, may we have a perfect graph G such that $\overline{G}$ is not perfect? The answer is no. This fact was first stated as the **Weak Perfect Graph Conjecture** by Berge in early 60th and proved by Lovasz in 1972 (the first proof used hypergraph approach).

Theorem 5.7.1 (Weak Perfect Graph Theorem, Lovasz, 1972) *A graph G is perfect if and only if its complement $\overline{G}$ is perfect.*

Chordal graphs were the first proved to be perfect:

Theorem 5.7.2 *If G is a chordal graph, then it is perfect.*

Proof. Let G be a chordal graph. Since any induced subgraph G' is chordal, it is sufficient to prove that $\chi(G) = \omega(G)$. Since G is chordal, it has a simplicial elimination ordering. Apply online (greedy) coloring to G by coloring vertices in inverse ordering as we did it in Section 5.5.. Recall that the maximum vertex degree is $\omega(G) - 1$. Hence, $\chi(G) = \omega(G)$ and G is perfect. □

Theorem 5.7.3 *If G is a quasi-triangulated graph, then it is perfect.*

Proof. Let G be a quasi-triangulated graph. Then by definition, see Section 3.5., it has a decomposition by sequential elimination of vertices that at each step, are simplicial or co-simplicial (simplicial in the complement). For any induced subgraph G', this ordering of vertices induces an ordering of vertices of G' with the same properties. Therefore G' has a decomposition by sequential elimination of vertices that at each step, are simplicial or co-simplicial. It means that G' is also a quasi-triangulated graph. Hence, as for chordal graphs, it is sufficient to prove that $\chi(G) = \omega(G)$.

Prove by induction on the number of vertices $n(G)$. The equality $\chi(G) = \omega(G)$ is true for $n = 3,4,5$. Let it be true for all quasi-triangulated graphs on $< n$ vertices and $n(G) = n$. By definition, G has a vertex x which is simplicial in G or in $\overline{G}$.

Case 1: x is a simplicial vertex in G. Consider a proper coloring of $G-x$ by $\chi(G-x) = \omega(G-x)$ colors. If $\omega(G) = \omega(G-x)$, then $|N(x)| \leq \omega(G-x)-1$ and we can use a color missing in $N(x)$ to color vertex x. If $\omega(G) = \omega(G-x)+1$, then $|N(x)| = \omega(G-x)$ and we must use a new color for x. In both cases we obtain a proper coloring of G with $\chi(G) = \omega(G)$ colors. Therefore, G is perfect.

Case 2: x is a simplicial vertex in $\overline{G}$. Again, graph $G-x$ is quasi-triangulated, has $< n$ vertices, and by the induction hypothesis is perfect. By Theorem 5.7.1, graph $\overline{G-x}$ is perfect quasi-triangulated graph. $\overline{G}$ is obtained from $\overline{G-x}$ by adding simplicial vertex x. By Case 1, $\overline{G}$ is perfect. By Theorem 5.7.1, G is perfect, what proves the theorem. □

There are many classes of perfect graphs. Among them are bipartite graphs, **weakly chordal** graphs (containing no induced cycles of length ≥ 5 in G and in $\overline{G}$), **strongly perfect** (every induced subgraph has an independent set intersecting all maximal cliques), **Meyniel graphs** (each odd cycle of length ≥ 5 has at least two chords), **Berge graphs** (no induced odd cycles of length ≥ 5 in graph and its complement). Examples and theorem above give rise to the idea that odd cycles of length ≥ 5 in graph and its complement prevent a graph from being perfect. Stated as the **Strong Perfect Graph Conjecture** by Berge in early 60th, this became

Theorem 5.7.4 (Strong Perfect Graph Theorem, Chudnovsky, Robertson, Seymour, Thomas, 2003) *A graph G is perfect if and only if it contains no induced C_{2k+1} and $\overline{C}_{2k+1}$, $k \geq 2$.* □

One of the most important fundamental properties of perfect graphs is that they, in contrast to general graphs, allow polynomial time algorithm (see Section 7.2.) for such optimization problem as finding the chromatic number, and consequently, the minimum clique covering, maximum independent set, and the maximum clique.

Exercises 5.7.

1. Determine if Petersen graph and its complement are perfect.
2. Determine which wheels are perfect and which are not.
3. Apply one step of Mycielski's construction starting with P_5, C_4, C_6, K_4, prism.
4. Prove that $\theta(C_7) > \alpha(C_7)$.

5. Are cube and prism perfect graphs?

Computer Projects 5.7. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. Given a graph G, determine if it contains induced C_5.
2. Given a quasi-triangulated graph G, find an optimal coloring.
3. Given a quasi-triangulated graph G, find the clique cover number $\theta(G)$ and the respective clique covering.

5.8. Edge Coloring and Vizing's Theorem

In Graph Coloring, we can also color the edges of a graph. In such colorings, parallel (multiple) edges will make a difference, but loops create inconveniences. Therefore the loops are ignored. Let $G = (X, E)$ be a graph with possible parallel edges but without loops, and $\lambda \geq 0$ be the number of available colors.

Definition 5.8.1 *A* **proper edge λ-coloring** *of G is an assignment of a color from set $\{1, 2, \ldots, \lambda\}$ to every edge of G is such a way that all edges incident to every vertex have distinct colors. A graph is* **edge k-colorable** *if it has a proper edge coloring with at most k colors.*

Since the maximum number of colors that can be used equals $|E|$, we are interested in the minimum number of colors. In order not to confuse with the chromatic number, the minimum number of colors over all proper edge colorings is called the **chromatic index** of G and denoted by $\chi'(G)$.

In any proper edge coloring of $G = (X, E)$, edges of the same color represent some matchings. Thus the coloring itself is the partition of the edge set E into a number of matchings. The chromatic index $\chi'(G)$ is the minimum number of distinct matchings over all such partitions.

Edge colorings of G can be expressed as vertex colorings of an auxiliary graph with vertices representing the edges of G. For graph $G = (X, E)$, construct a simple graph $G' = (X', E')$ such that $X' = E$ and E' is formed in the following way: two vertices in G' are adjacent if and only if the respective edges have a common vertex in G. Graph G' is called the **line graph** of G and is denoted by $L(G)$. Not every graph can be a line graph.

An example of a graph G, its line graph $L(G)$, proper edge 4-coloring of G and proper vertex 4-coloring of $G' = L(G)$ is shown in Figure 5.25. One can see that every proper edge coloring of G corresponds to a proper vertex coloring of G' and vice versa. For this example $L(G)$ is a chordal graph, so for edge colorings of G we can even compute the chromatic polynomial $P(L(G), \lambda) = \lambda(\lambda - 1)(\lambda - 2)^2(\lambda - 3)$ as it follows from the respective simplicial decomposition. Notice that every non pendant vertex of G forms a clique in $L(G)$ but not vice versa: clique $\{b, c, d, e\}$ in $L(G)$ does not have a corresponding vertex in G. At last, observe that $\chi'(G) = 4 = \chi(L(G))$, and moreover, the equality $\chi'(G) = \chi(L(G))$ holds for any graph G without loops.

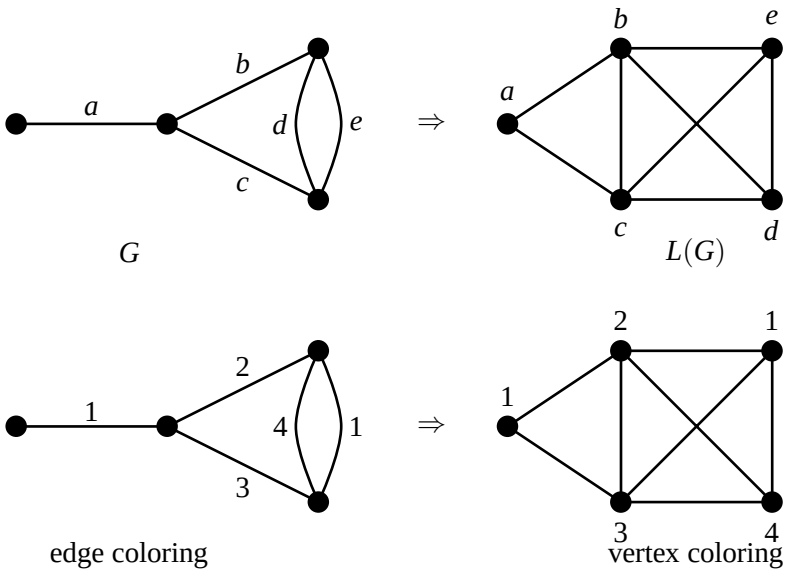


Figure 5.25.

As usually, let $\Delta(G)$ denote the maximum vertex degree of G. Evidently, $\chi'(G) \geq \Delta(G)$. Maximum degree Δ plays a similar role for the chromatic index χ' as the clique number ω plays for the chromatic number χ in vertex coloring where $\chi \geq \omega$. But how far may be the value $\chi'(G)$ from $\Delta(G)$? Surprisingly, it must be very close.

Theorem 5.8.1 (Vizing, 1964) *If G is a simple graph, then either*

$$\chi'(G) = \Delta(G), \text{ or } \chi'(G) = \Delta(G) + 1.$$

Proof. Simple examples (like C_{2k} and C_{2k+1}) show that there are graphs with both values of χ'. We will show that any graph can be edge colored with $\Delta + 1$ colors.

Let G be a simple graph. Take $\Delta + 1$ colors and properly color as many edges as possible. If all edges are colored, we are done. Otherwise, we obtain some partial edge coloring. Suppose an edge connecting vertices x and y_0 and denoted by just xy_0, is uncolored. We will recolor edges in such a way that xy_0 becomes colored and the number of colors remains the same. After repeating the procedure as many times as necessary, a complete proper edge coloring of G will be obtained. Since we use $\Delta + 1$ colors and Δ is the maximum degree, at each vertex at least one color is missing. A feature of this proof is that the reasoning is constructed in terms of such missing colors.

The best case: a color c is missing at both vertices x and y_0. Color edge xy_0 with c.

The good case: there is no color missing at both x and y_0 simultaneously. Let a color c_0 be missing at x and color c_1 be missing at y_0. If there is an edge xy_k colored c_1 with both ends missing a color c_i, then re-color xy_k with c_i and color xy_0 with c_1.

The bad case: there is no edge xy_k colored c_1 with both ends missing a color. Edge xy_1 is colored c_1; a color c_2 is missing at y_1. Edge xy_2 is colored c_2, a color c_3 is missing at y_2. Edge xy_3 is colored c_3, a color c_4 is missing at y_3, and so on, see Figure 5.26 (missing

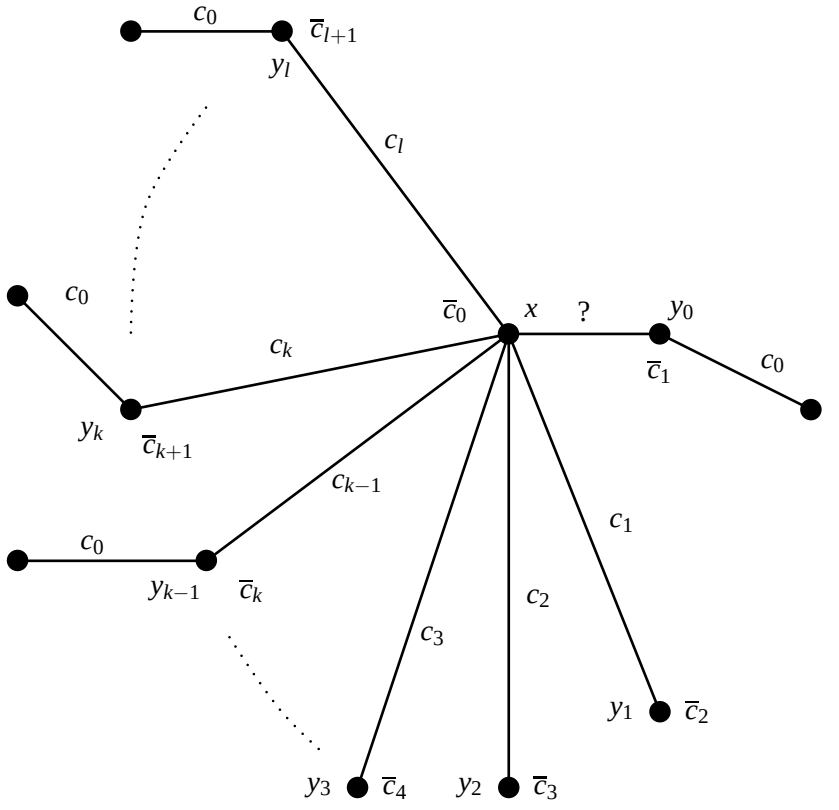


Figure 5.26.

colors are denoted by $\overline{c}_i$). Suppose this sequence continues until a color c_{l+1} is missing at both vertex x and y_l. Then we re-color each edge xy_i from c_i to c_{i+1}, $i = 1,2,\ldots,l$. Call this re-coloring **downshifting** from y_l. We arrive to a situation when color c_1 is missing at x. Since it is missing at y_0, too, color edge xy_0 with c_1.

The worst case: we never reach the situation when some color is missing at both ends of an edge. Then in the sequence of colors $c_1, c_2, c_3, \ldots$, sooner or later, some color c_{l+1} repeats (i.e., we have a loop) because the number of colors is finite. Let $c_{l+1} = c_k$ be the first such coincidence of the colors, $1 \le k \le l$. Observe that color c_0 is present at all colored neighbors of x and it is missing at x. Consider now the Kempe path P consisting of edges of two alternating colors c_0 and c_k and beginning at vertex y_l. Since we color the edges, each vertex may have at most two edges colored with c_0 and c_k; therefore, there is the only such path.

Subcase 1: P reaches vertex y_k and therefore ends at x. Switch colors c_0 and c_k along P. We obtain edge xy_k with color c_k missing at both ends. Downshifting from y_k results in release of color c_1 to be used for xy_0.

Subcase 2: P reaches vertex y_{k-1} and therefore ends at y_{k-1}. Again, switch colors c_0 and c_k along P. As a result, color c_0 is missing at y_{k-1}. Since it is also missing at x, we obtain edge xy_{k-1} with color c_0 missing at both ends. Perform downshifting from y_{k-1} and again, release color c_1 to color edge xy_0.

Subcase 3: P never reaches y_k or y_{k-1}. Since c_k is present at x, P never reaches vertex x. Switch the colors c_0 and c_k along P; color c_0 becomes missing at y_l. Perform downshifting from y_l and release color c_1 to color edge xy_0. □

The theorem above partitions all simple graphs into two classes: **Class 1** if $\chi'(G) = \Delta(G)$ and **Class 2** if $\chi'(G) = \Delta(G)+1$. Determining whether a simple graph is Class 1 or Class 2 is a hard problem.

If a graph G has multiple (parallel) edges (but not loops), the **multiplicity** of G, denoted by $\mu(G)$, is the maximum number of parallel edges connecting a pair of vertices. Theorem 5.8.1 allows then the following generalization:

Theorem 5.8.2 (Vizing, 1964) *If G is a loop-less multigraph with multiplicity $\mu(G)$, then* $\Delta(G) \leq \chi'(G) \leq \Delta(G)+\mu(G)$.

The "worst" example when the upper bound is achieved (i.e. $\chi'(G) = \Delta(G)+\mu(G)$) is represented by the so called "fat triangle", see Figure 5.27. Theorem 5.8.2 is a direct generalization of Theorem 5.8.1 because for simple graphs $\mu = 1$.

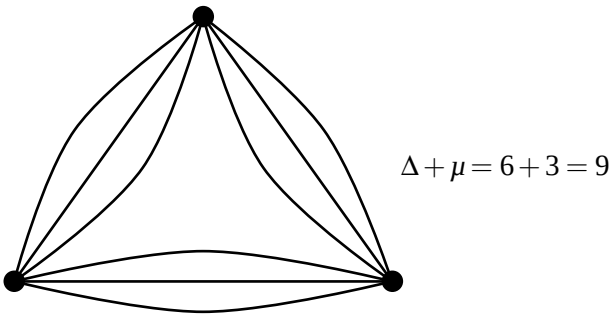


Figure 5.27. "Fat triangle" requires $\Delta+\mu$ colors.

We conclude with

Theorem 5.8.3 (König, 1916) *If G is a bipartite graph, then*

$$\chi'(G) = \Delta(G).$$ □

Exercises 5.8.

1. Find the chromatic index and an optimal edge coloring of: $K_n, C_n, W_n, K_{m,n}$, where $m,n \geq 3$.

2. Find the chromatic index, chromatic class and an optimal edge coloring of cube, prism and their complements.

3. Prove that for the Petersen graph $\chi' = 4$.

4. In the Petersen graph, replace each edge by three parallel edges and find the chromatic index and respective edge coloring.

Computer Projects 5.8. Using an appropriate graph representation, write a program for the following algorithmic problem.

1. Given a tree, find the chromatic index and an optimal edge coloring.

2. Given a graph G, at random generate an edge coloring and test if it is proper.

3. Given a multigraph G, generate a proper edge coloring using a vertex ordering.

5.9. Upper Chromatic Index

In this section we show how replacing the definition of proper edge coloring with the *opposite* version (the requirement "all of different colors" is replaced by "at least two of the same color") leads to the concept of the upper chromatic index and a formula for it.

Let $G = (X, E)$ be an arbitrary multigraph without loops and, as usually, $\{1, 2, \ldots, \lambda\}$ be the set of available colors.

Definition 5.9.1 *A* **proper edge λ-coloring** *of multigraph G is an assignment of a color from set $\{1, 2, \ldots, \lambda\}$ to every edge of G in such a way that every non-pendant vertex of G is incident to at least two edges of the same color.*

Following this new definition, we can color all edges with one color and it will be a proper coloring; however, we cannot use $|E|$ colors unless G is just a matching. Let us agree that if k colors are really used in a proper edge coloring, $k \leq \lambda$, then the coloring is called **strict edge k-coloring**, or just k-coloring.

Definition 5.9.2 *The maximum number k for which there exists a proper edge k-coloring of multigraph G is called the* **upper chromatic index** *and denoted by $\overline{\chi}'(G)$.*

An example of a multigraph G and its edge 3-coloring is shown in Figure 5.28. We will show that it uses the maximum number of colors, i.e., $\overline{\chi}'(G) = 3$.

Theorem 5.9.1 *For a connected multigraph G, the upper chromatic index $\overline{\chi}'(G) = 1$ if and only if the maximum degree $\Delta(G) \leq 2$.*

Proof. $\Rightarrow$ Let $\overline{\chi}'(G) = 1$. For a contradiction, suppose $\Delta(G) \geq 3$. If G contains a cycle, then color the edges of the cycle with color 1, and all the other edges with color 2. If G is a tree, color any maximal path with color 1 and all other edges with color 2. In both cases we obtain a proper edge 2-coloring, a contradiction.

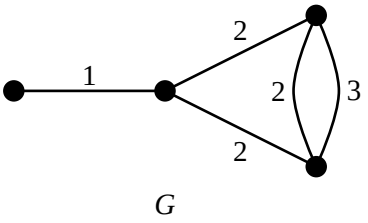


Figure 5.28.

$\Leftarrow$ The converse is evident since $\Delta(G) \leq 2$ implies that G is either a cycle or a path. $\square$

For a connected multigraph G, theorem 5.9.1 immediately implies that $\overline{\chi}'(G) \geq 2$ if and only if $\Delta(G) \geq 3$.

Consider a proper edge k-coloring of $G = (X, E)$; it partitions E into k *color classes* $\{C_1, C_2, \ldots, C_k\}$ where each C_i is a set of edges colored with color i. We denote such partition and the respective edge coloring by f.

Let $G_i = (X_i, C_i)$ be the subgraph of G with a set of vertices X_i determined by endpoints of the edges of C_i. Notice that subgraphs G_i are not necessarily connected multigraphs and may have common vertices with each other; however, they do not have common edges.

A vertex x of a subgraph G_i is said to be **satisfied** by subgraph G_i if it is not a pendant vertex in G_i.

Let us discuss some properties of edge colorings that use $\overline{\chi}'(G)$ colors.

Theorem 5.9.2 *If f is a coloring of G using $\overline{\chi}'(G)$ colors, then*

$$\overline{\chi}'(G_i) = 1,\ i = 1, \ldots, \overline{\chi}'(G).$$

Proof. In coloring f, every pendant vertex of G_i is satisfied by some other subgraph. Therefore any proper coloring of G_i with at least two colors leads to a proper $(\overline{\chi}'(G)+1)$-coloring of G, a contradiction. $\square$

Corollary 5.9.1 *Let f be a coloring of G using $\overline{\chi}'(G)$ colors. Then each G_i, where $1 \leq i \leq \overline{\chi}'(G)$, is either a cycle or a path.*

Proof. Indeed, in f, every G_i must be connected because otherwise we could increase the number of colors. Any connected multigraph with $\overline{\chi}'(G) = 1$ is either a cycle or a path (Theorem 5.9.1). $\square$

Theorem 5.9.3 *Let f be a coloring of G using $\overline{\chi}'(G)$ colors. Then for every vertex x of G the following implications hold:*

1) *if x is satisfied by a cycle, then x is satisfied by no path;*

2) *if x is satisfied by two cycles, say G_i and G_j, then x is the only common vertex for G_i and G_j.*

Proof. In 1), otherwise, we could break the path in two paths and increase the number of colors.

In 2), otherwise, we could split two cycles G_i and G_j into one cycle and two paths, see Figure 5.29, assign colors i and j to the paths and a new color to the new cycle, and again, increase the number of colors. □

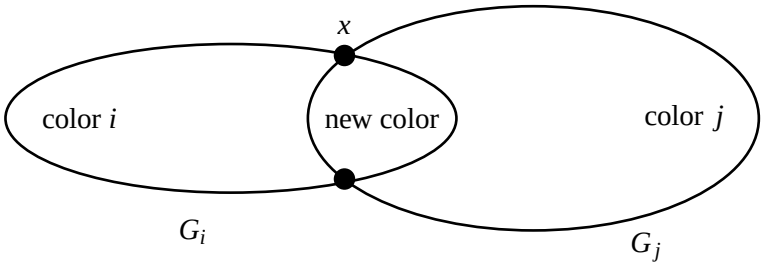


Figure 5.29.

Corollary 5.9.2 *Let f be a coloring of G using $\overline{\chi}'(G)$ colors. If a vertex x of G is satisfied by more than one G_k, then all G_ls which satisfy x, are all cycles with one vertex in common.*

□

Given a $\overline{\chi}'(G)$-coloring of an arbitrary multigraph G, we now partition the subgraphs G_i of G into the following three classes.

Class A: contains all G_i forming the maximum number of (vertex) disjoint cycles in f. Clearly, the number c' of subgraphs in A satisfies $c' \leq c$ where c is the maximum number of disjoint cycles in G.

Class B: contains the remaining cycles and paths with length at least two each.

Class C: contains G_i which represent a separate edge each.

Observe that all non-pendant vertices of G are satisfied by subgraphs from classes A and B, and none is satisfied by any edge from part C; in addition, no pendant vertex is satisfied.

If class B contains cycles, then given the maximum number of disjoint cycles in A, and by Theorem 5.9.2, each of these cycles has a single vertex in common with only one of the cycles in A. The cycles in B may be considered as paths whose endpoints coincide. Notice that the number of edges in each such cycle or path equals the number of internal vertices plus 1.

The next theorem determines the formula for the upper chromatic index. It may be seen as the *opposite to Vizing's theorem* in the sense that finding the minimum number of colors is replaced with finding the maximum number of colors.

Theorem 5.9.4 (Gionfriddo, Milazzo, Voloshin, 2001) *If $G = (X,E)$ is an arbitrary multigraph with $|X| = n$, $|E| = m$, the number of pendant vertices p, and the maximum number of disjoint cycles c, then*

$$\overline{\chi}'(G) = c + m - n + p.$$

Proof. We prove first that $\overline{\chi}'(G) \leq c+m-n+p$. By Corollary 5.9.1, if a coloring f uses $\overline{\chi}'(G)$ colors, then subgraphs G_i, $1 \leq i \leq \overline{\chi}'(G)$, are either cycles or paths, while Corollary 5.9.2 states that if a vertex x is satisfied by more than one graph G_i, then all these graphs have precisely one vertex in common.

Let c' be the number of subgraphs, and v' be the number of satisfied vertices in class A. Clearly, v' coincides with the number of edges in A because all cycles in A are vertex disjoint. Let x be a satisfied vertex in G. If x is satisfied by some G_l from class A, then it is possible to look at all the other G_ks which satisfy x, like paths with endpoints x.

If x is not satisfied by any subgraph from class A, then it is satisfied by a subgraph from class B; therefore it is possible to consider one such subgraph G_k as a cycle and all others as paths with endpoints x.

Let r be the number of G_is in class B. Since pendant vertices are not satisfied, the number of vertices that are satisfied by subgraphs from B equals $n - v' - p$. They are the internal vertices for the paths or the internal vertices for the cycles which are considered as paths with coinciding endpoints. Since the number of edges in each of them equals the number of internal vertices plus 1, and the number of all such subgraphs equals r, the total number of edges in B equals

$$n - v' - p + r.$$

Let m' be the number of edges in classes A and B combined. Then we obtain:

$$m' = v' + (n - v' - p + r) = n - p + r.$$

Since class C contains $m - m'$ edges, the number of colors in f is:

$$\overline{\chi}'(G) = c' + r + (m - m') = c' + m - n + p,$$

and therefore we obtain

$$\overline{\chi}'(G) = c' + m - n + p \leq c + m - n + p.$$

Next we prove that $\overline{\chi}'(G) \geq c+m-n+p$. Let us choose c vertex disjoint cycles $C_1, \ldots, C_c$ and denote the subgraph with vertex set X and edges of these cycles by A_0. Color the edges of the cycles properly with the colors $1, \ldots, c$. Since the number of edges $m(A_0) = n(A_0)$, and the number of pendant vertices $p(A_0) = 0$, we obtain a proper coloring of A_0 using $c + m(A_0) - n(A_0) + p(A_0)$ colors. Therefore, $\overline{\chi}'(A_0) \geq c + m(A_0) - n(A_0) + p(A_0)$, and the inequality holds.

Now we implement the following coloring augmenting procedure. Choose any satisfied vertex, say x, incident to an uncolored edge and construct a path along uncolored edges until the first satisfied vertex y is reached or we get stuck at a pendant vertex y. Since A_0 constitutes the maximum number of disjoint cycles, the new vertices and edges represent either a cycle or a path (if they form a cycle, then $x = y$).

Add these new vertices and edges to A_0 and denote the obtained subgraph by A_1. Color the added edges with a new color and declare respective vertices satisfied.

If y is pendant, then the numbers of newly added vertices and edges coincide. If y is not pendant, then the number of added edges equals the number of added vertices plus 1. So, in either case,

$$\overline{\chi}'(A_1) \geq \overline{\chi}'(A_0) + 1 \geq c + m(A_1) - n(A_1) + p(A_1),$$

and therefore the inequality holds. We repeat this coloring procedure by constructing subgraphs $A_2, A_3, \ldots$ until all the edges of G are colored and all non-pendant vertices satisfied. Since at each coloring step the inequality holds, we obtain that $\overline{\chi}'(G) \geq c + m - n + p$. Hence the theorem follows. □

For our example, see Figure 5.28, it is easy to see that $c = 1$, $m = 5$, $n = 4$ and $p = 1$, and the formula gives:

$$\overline{\chi}'(G) = c + m - n + p = 1 + 5 - 4 + 1 = 3,$$

so the coloring in the Figure is **maximal**, i.e., using the maximum number of colors. As there are several largest sets of vertex disjoint cycles, there are several maximal colorings. Notice that any two parallel edges form cycle C_2 and may contribute to the value of c.

Theorem 5.9.4 allows finding the upper chromatic index for particular classes of multigraphs.

Theorem 5.9.5 *If G is a tree, then*

$$\overline{\chi}'(G) \; = \; p \; - \; 1.$$

Proof. Indeed, for any tree $c = 0$ and $m \; = \; n \; - \; 1$. □

Exercises 5.9.

1. For graph G, see Figure 5.30, find a maximal edge coloring by hand.

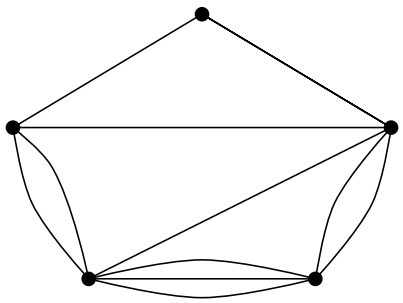


Figure 5.30.

2. For graph G, see Figure 5.30, find the largest set of vertex disjoint cycles and determine $\overline{\chi}'(G) = c + m - n + p$ and respective edge coloring.

3. For graph G, see Figure 5.30, find another largest set of vertex disjoint cycles, calculate $\overline{\chi}'(G)$ and show respective edge coloring.

4. For graph G, see Figure 5.30, show that coloring the edges of C_5 with one color does not produce maximal coloring.

5. Find the upper chromatic index and a maximal edge coloring of: C_n, W_n, $K_n, K_{m,n}$, where $m, n \geq 2$.

6. Find the upper chromatic index and a maximal edge coloring of Petersen graph, cube and prism and their complements.

7. In Petersen graph, replace each edge by three parallel edges and find the upper chromatic index and respective edge coloring.

8. In $C_n, W_n, K_n, K_{m,n}$, where $m, n \geq 2$, replace each edge by two parallel edges and find the upper chromatic index and respective edge coloring.

Computer Projects 5.9. Write a program for the following algorithmic problems.

1. Given a graph G, find a number of vertex disjoint cycles and estimate on $\overline{\chi}'(G)$.

2. Given a graph G and a number of properly colored vertex disjoint cycles, implement the edge coloring procedure as described in the proof of Theorem 5.9.4 to augment the coloring.

3. Generate an edge coloring at random and verify if it is proper.

4. Generate a maximal edge coloring of complete graph K_n, $n \geq 5$.

5. Given a tree T_n, construct a maximal edge coloring.

Chapter 6

Graph Traversals and Flows

6.1. Eulerian Graphs

In a graph G, a **walk** is an alternating sequence of vertices and edges where every edge connects preceding and succeeding vertices in the sequence. A walk starts at a vertex, end at a vertex and has the following form: $x_0, e_1, x_1, e_2, \ldots, e_k, x_k$. There are no restrictions on repetitions of vertices and edges. The number of edges in a walk is its **length.** So, a walk beginning at x_0 and ending at x_k has the length k; it is a (x_0, x_k)-walk. A walk with $x_0 = x_k$ is called **closed.** If no edge is repeating, then a walk is called a (x_0, x_k)-**trail**. A trail with no repeating vertices clearly is a path; a path with $x_0 = x_k$ clearly is a cycle. Thus we recognize the well known concepts as the special cases of a walk and trail.

A connected graph G is called **Eulerian** if it has a closed trail containing all edges of G; such a trail is then called an **Eulerian trail**. Since trails do not repeat edges, Eulerian trail passes through each edge once.

Lemma 6.1.1 *If in a graph G the degree of every vertex is at least 2, then G contains a cycle.*

Proof. If G contains loops or multiple edges, then the statement is evident. Therefore, assume G is a simple graph. Consider a path P of maximal length with an endpoint x. Observe that all neighbors of x line on P because otherwise we could extend P. Since G is simple and the degree of x is at leats 2, vertex x has at least two distinct neighbors on P. Thus vertex x, the two its neighbors on P and the segment of P between them form a cycle of length at least 3. □

Theorem 6.1.1 (Euler, 1736) *A connected graph G is Eulerian if and only if the degree of every vertex is even.*

Proof. $\Rightarrow$ Let T be an Eulerian trail in G.Each time T passes through a vertex, it uses two edges. Since T is closed and uses all edges, every vertex has an even degree.

$\Leftarrow$ Induction on the number of edges of G: assume the statement is true for all graphs with less than $m(G)$ edges. Since the degree of every vertex in G is even and G is connected, the degree each vertex is at least 2. Therefore, by Lemma 6.1.1, G contains a cycle C, see Figure 6.1 (take any cycle as C). Delete the edges of C from G and obtain a graph G'

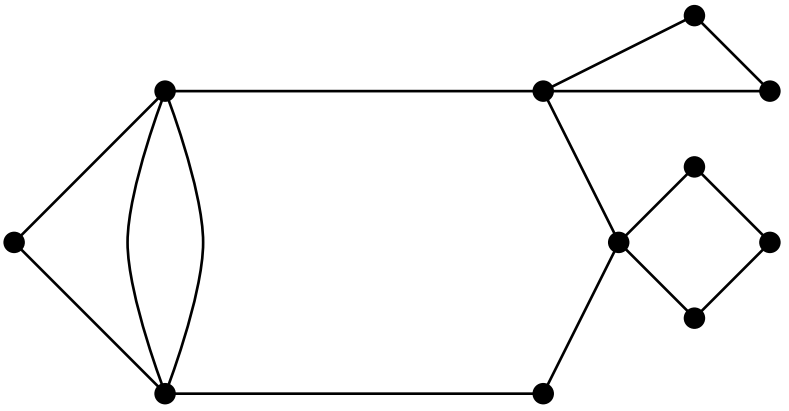


Figure 6.1.

in which all vertices have even degrees. Graph G' may be disconnected but since G is connected, cycle C passes through each component of G'. Since $m(G') < m(G)$, by the induction hypothesis, every connected component of G' has an Eulerian trail. Now we combine cycle C with an Eulerian trail of each connected component of G' in the following way: traverse C and make a detour at the very first vertex of each component. Since every detour ends at the same vertex where it started, an Eulerian trail of G is constructed. □

The theorem above implies that every graph $G = (X,E)$ having all vertices of even degree may be decomposed into cycles, i.e., edge set E may be partitioned into subsets where each subset forms a cycle.

The proof of the theorem above explicitly suggests the idea how to construct an Eulerian trail in a given Eulerian graph: choose a closed trail and then recurrently extend it until the initial graph is obtained.

Exercises 6.1.

1. Which of the graphs K_n, $K_{m,n}$, W_n, cube, prism, the Petersen graph, a tree are Eulerian?

2. The degree equality $\Sigma_{i=1}^{n} d(x_i) = 2m$ (See Proposition 1.1.1) implies that in any graph G, the number of vertices of odd degree is even. How this fact can be used to make any graph Eulerian?

3. Show that if a connected graph G contains k vertices of odd degree, then the minimum number of trails that partition the edges is $k/2$.

4. Show that if a connected graph G contains k vertices of odd degree, then $k/2$ continuous pen-strokes are sufficient to draw G in the plane.

5. What is the minimum number of trails that partition the edge set of the Petersen graph?

Computer Projects 6.1. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a graph G, recognize if it is Eulerian, and if yes, then construct an Eulerian trail.

6.2. Hamiltonian Graphs

In contrast to Eulerian graphs, one may ask if in a graph G, there exists a closed trail passing exactly one time through every vertex of G. Clearly, such a trail is a spanning cycle which is then called a **Hamiltonian cycle.** If G contains a Hamiltonian cycle, then it is called a **Hamiltonian graph**.

In contrast (one more) to Eulerian graphs, the problem of recognizing Hamiltonian graphs is very difficult. Until now there are no criteria for characterization of Hamiltonian graphs. Typical results about Hamiltonian graphs require significant number of edges in a graph. However, there are Hamiltonian graphs (like cycles etc.) with small number of edges. Recall that $d(x)$ is the degree of a vertex x.

Theorem 6.2.1 (Ore, 1960) *If in a simple graph $G = (X,E)$, with $|X| = n \geq 3$, for every pair x and y of disjoint vertices*

$$d(x) + d(y) \geq n,$$

then G is a Hamiltonian graph.

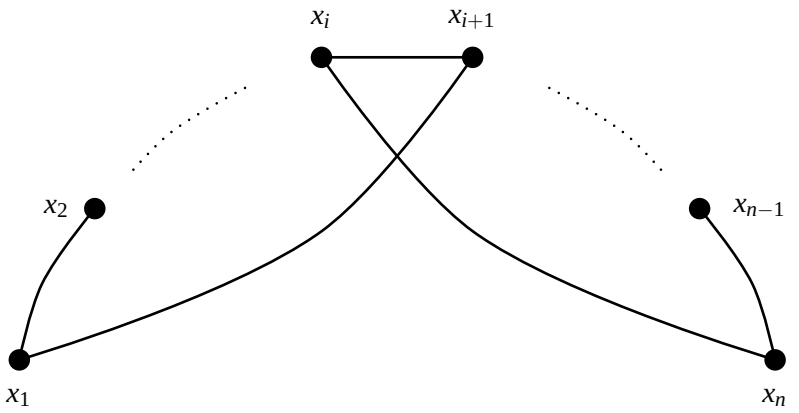


Figure 6.2.

Proof. By contradiction, assume that G is not Hamiltonian. Hence it is not a complete graph and we can sequentially add edges to G as long as possible before it becomes Hamiltonian. The addition of edges preserves the inequality on vertex degrees. Therefore, without loss of generality, we can assume that G is maximal "non-Hamiltonian" graph. Hence it contains a path P passing (in order from left to right) through all the vertices $x_1, x_2, \ldots, x_n$,

see Figure 6.2. Since G is not Hamiltonian, vertices x_1 and x_n are disjoint what implies $d(x_1)+d(x_n)\geq n$. On the other hand, since x_1 and x_n are disjoint, $d(x_1)\leq n-2$ and $d(x_n)\leq n-2$. Consider all $n-3$ internal edges of path P. If for each of them the left end is not adjacent to x_n *or* the right end is not adjacent to x_1, then in total sum of degrees for x_1 and x_n at least $n-3$ edges are missing. Therefore

$$d(x_1)+d(x_n)\leq (n-2)+(n-2)-(n-3)=n-1$$

what contradicts the condition of theorem. The last implies that among $n-3$ internal edges of P there exists one, say, (x_i,x_{i+1}) with x_i adjacent to x_n and x_{i+1} adjacent to x_1.

Thus the cycle

$$x_1,x_{i+1},x_{i+2},\ldots,x_n,x_i,x_{i-1},\ldots,x_1$$

is Hamiltonian what completes the proof. □

Corollary 6.2.1 (Dirac, 1952) *If in a simple graph $G=(X,E)$, with $|X|=n\geq 3$, the degree of each vertex is at least $n/2$, then G is Hamiltonian.*

Proof. The statement follows directly from the theorem above since $d(x)+d(y)\geq n$ holds for every pair of vertices in G. □

In both the theorem and corollary we include the condition $n\geq 3$ to avoid graph K_2 which is not Hamiltonian but satisfies the degree inequality.

Exercises 6.2.

1. For which values of n and m graphs $K_n, K_{m,n}$, W_n are Hamiltonian?
2. Are the prism, cube, and the Petersen graph Hamiltonian?
3. Is the Grözsch graph Hamiltonian (see Figure 5.23)?
4. Which of the complement of prism, cube, Petersen and Grözsch graphs are Hamiltonian?
5. Which of the graphs $\overline{C_7}$ and $\overline{C_8}$ are Hamiltonian?
6. Construct a list of all Hamiltonian cycles of K_4 and K_5.

Computer Projects 6.2. Using an appropriate graph representation, write a program for the following algorithmic problems.

1. Given a graph G with the degree condition as in Theorem 6.2.1, and a path with disjoint ends passing through all the vertices. For the path, find an edge extension to construct a cycle as in Theorem 6.2.1.
2. Given a graph G and a set of edges, verify if the set of edges forms a cycle.

3. Given a graph G and a cycle, find the procedure of extending the cycle as much as possible.

4. Given a complete graph K_n, generate all Hamiltonian cycles.

5. Given a graph G, generate n edges at random and check if they form a cycle.

6. Given a 3-regular graph on 10 vertices, check if it is Hamiltonian.

6.3. Network Flows

Networks are anywhere: in traffic, communications, internet, even in our body. It appears that graph theory provides an important mathematica model that allows to find optimal flows in networks.

Recall that a graph in which all edges are ordered pairs (and therefore are called arcs) is called a digraph. A digraph $N = (X,A)$ is called a **network**, if X is a set of vertices (sometimes called **nodes**), A is a set of arcs, and to each arc $a \in A$ a non-negative real number $c(a)$ is assigned which is called the **capacity** of arc a. For any vertex $y \in X$, any arc of type (x,y) is called **incoming**, and every arc of type (y,z) is called **outcoming**. In N, there are two special vertices: $u \in X$, which is not incident to any incoming arc and is called the **source**, and $v \in X$, which is not incident to any outcoming arc and is called the **sink**.

A **network flow** F is an assignment to *each* arc $a \in A$ a non-negative real number $f(a)$ (called the **flow in** a) such that:

1. $f(a) \leq c(a)$;

2. for any vertex $x \in X$, except u and v, the following **flow conservation law** holds: *the sum of flows of all incoming arcs equals the sum of flows of all outcoming arcs.*

An arc $a \in A$ for which $f(a) = c(a)$ is called **saturated**; if $f(a) < c(a)$, then arc a is called **unsaturated**. The **value of the network flow** is the sum of all flows in arcs of type (u,x); the flow conservation law implies that it equals the sum of all flows in arcs of type (x,v).

Given a network $N = (X,A)$, how can we find a maximum flow? Observe that there is no such concept as the conservation law for the capacities: otherwise we could run the maximum flow right from the source. The answer to this question is closely related to the concept of a **cut** which is a subset of arcs $S \subseteq A$ that separates the source from the sink. It means that every path from the source u to the sink v contains at least one arc from S. The **capacity of the cut** S is the sum of the capacities of all arcs from S. Different cuts have different capacities, and evidently, no flow can exceed the smallest capacity over all cuts. Any cut having the smallest capacity is called the **minimum cut**.

Theorem 6.3.1 (Max-flow min-cut theorem, Ford and Fulkerson, 1956)
In each network, the value of maximum flow equals the capacity of minimum cut.

Proof. Since the value of maximum flow does not exceed the capacity of minimum cut, we prove that for any given maximum flow there exists a minimum cut having the capacity equal to the value of the flow.

Let $N = (X, A)$ be a network with a maximum flow. Consider a sequence of vertices $u = x_0 \to x_1 \to x_2 \to \cdots \to x_k$ such that either (x_i, x_{i+1}) is an unsaturated arc, or (x_{i+1}, x_i) is an arc with a non-zero flow, $i = 0, 1, \ldots, k-1$. Denote the set of all such vertices in all such sequences by Y. Source u does not have incoming arcs; if all outcoming arcs are saturated, then we are done. If at least one outcoming arc from the source is unsaturated, then we have at least one such sequence and $u \in Y$. Let $Z = X \setminus Y$. We claim that the sink $v \in Z$.

By contradiction, assume $v \in Y$. It means that there exists a sequence $u = x_0 \to x_1 \to x_2 \to \cdots \to x_k = v$ with the property above. Choose a number $\delta > 0$ which does not exceed the value needed to saturate any unsaturated arc of type (x_i, x_{i+1}) and does not exceed the flow in any arc of the type (x_{i+1}, x_i). We now increase the flow in all arcs of the first type and decrease it in all arcs of the second type by the same value δ. It is clear that the flow conservation law holds, and no capacity is exceeded. Therefore we obtain a flow which is greater than the maximum flow, a contradiction.

Now, let S be the set of all arcs with initial vertex in Y and terminal vertex in Z. Evidently, S is a cut. Every arc from S is saturated because otherwise we could move a respective terminal vertex from Z to Y. Each arc from Z to Y has the zero flow because otherwise we could move a respective initial vertex from Z to Y. This implies that the capacity of cut S equals the value of the given maximum flow what completes the proof. □

The idea of the proof in the theorem above is used in the Ford-Fulkerson algorithm for finding maximum flow in a network. It is illustrated by the example of a network and flow shown in Figure 6.3 with continuation in Figure 6.4.

The pair of numbers (0,3) attached to arc (u, x_1) in the initial flow F_0 means that the flow in arc (u, x_1) is 0, and the capacity is 3. The same "(flow, capacity)" rule holds for every arc in each flow. As one can see, the initial flow F_0 is obtained by running one unit of flow along the path

$$u \to x_4 \to x_3 \to x_1 \to x_2 \to v$$

and assigning flow 0 in all remaining arcs. It immediately saturates the arc (x_3, x_1) as the pair (1,1) shows.

Flow F_1 is obtained from the flow F_0 by adding one unit in arc (u, x_1), subtracting it in backward arc (x_1, x_3), and adding it in arc the (x_3, v).

Flow F_2 is obtained from the flow F_1 by adding two units along the path

$$u \to x_1 \to x_2 \to v.$$

Flow F_3 (see the next figure) is obtained from the flow F_2 by adding one unit along the path

$$u \to x_4 \to x_3 \to v.$$

At last maximum flow F_4 is obtained from the flow F_3 by adding one unit along the path

$$u \to x_4 \to x_3 \to x_1 \to x_2 \to v.$$

We conclude that the flow value equals 6, it is maximum because the arcs (u, x_1), (x_3, x_1) and (x_3, v) form a cut with the capacity 6, so it is the minimum cut. At each step we

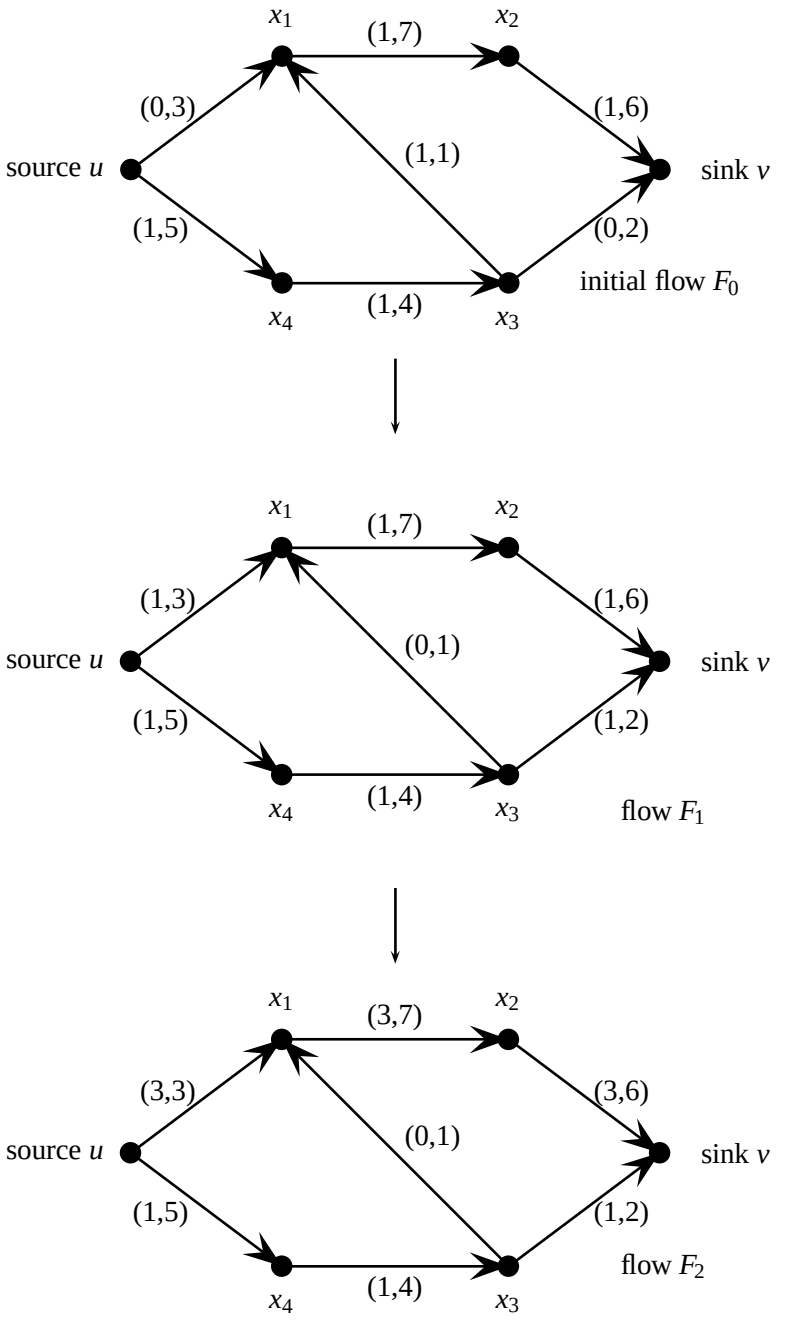


Figure 6.3.

augmented the flow. One can easily see that there is no further flow-augmenting path at this point.

Generally, there are many ways to search for the flow augmenting paths.

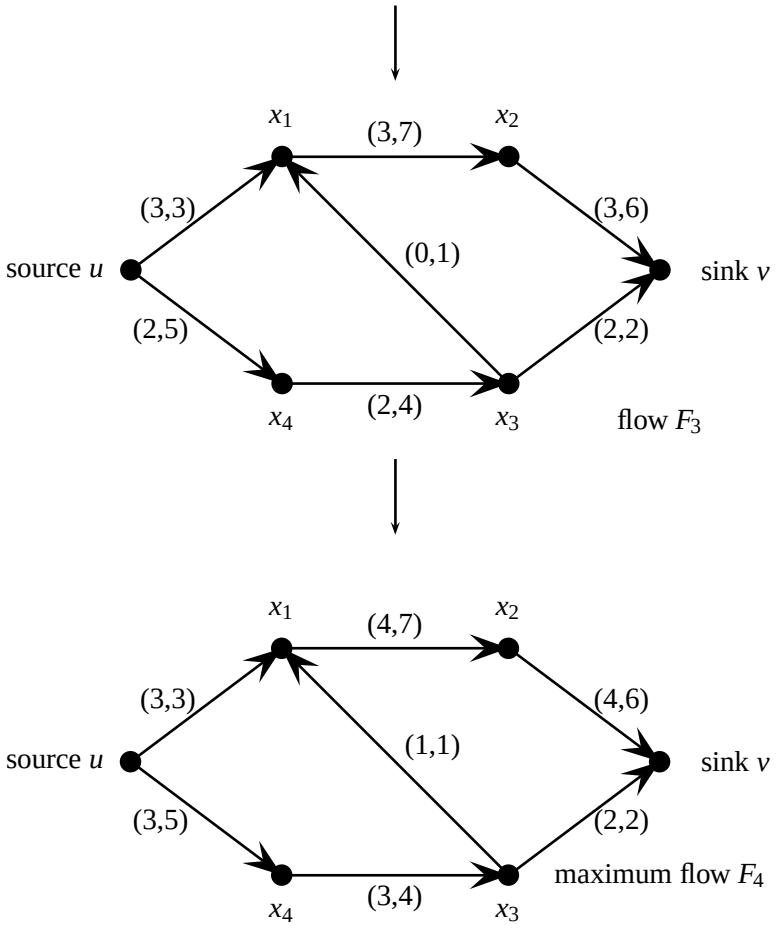


Figure 6.4.

Exercises 6.3.

1. In the example in Figure 6.3, add arc (x_4, x_2) with capacity 2 and find the maximum flow.

2. In the example in Figure 6.3, change the capacity of some arcs and find the maximum flow.

3. In the example in Figure 6.3, change the direction of an arc and find the maximum flow.

4. Given a network with several sources and sinks, suggest a way to reduce the problem to the one with one source and one sink.

5. Given a cube, prism, wheel W_7, assign direction to every edge (making them arcs) in a way to obtain a network with one source and one sink, assign the capacity to each arc, and find the maximum flow.

Computer Projects 6.3. Write a program for the following algorithmic problems.

1. Given a network, find a flow augmenting path.
2. Transform Petersen graph into a network with one source and one sink and find a maximum flow.
3. Given a network with a flow, determine if the flow is maximum.

Chapter 7

Appendix

"What did you like most during the study at the university?
- Great school, great professors...
What did you hate most?
- Mathematical induction..."

7.1. What Is Mathematical Induction

One of the most popular methods of proof in graph theory is the proof by mathematical induction. The idea is simple. Suppose we want to prove a statement P_n depending on n where $n = 1, 2, 3, \ldots$. This is equivalent to prove an infinite list of statements $P_1, P_2, \ldots$ to be true. Instead of proving the infinite list of theorems, we prove just the following two:

1. Prove P_1.

2. For any $k \geq 1$, prove that if P_k is true, then P_{k+1} is true.

It is easy to see that this is sufficient to conclude that P_n is true for all $n = 1, 2, \ldots$, or, equivalently, all statements $P_1, P_2, \ldots$ are true.

Step 1. is called the **basis of induction**, see Fig. 7.1. Step 2. is called the **induction step**. It can be abbreviated as $P_k \rightarrow P_{k+1}$ and called an **implication.** In step 2., the statement P_k is called the **induction hypothesis.** In notation P_n, n is called the **induction parameter**.

Mathematical induction is a bright example how generalizations work in mathematics: instead of proving $P_1 \rightarrow P_2$, then $P_2 \rightarrow P_3$, and so on (up to infinity!), one just prove the general case $P_k \rightarrow P_{k+1}$. In this way, the induction step replaces infinitely many proofs. To have a complete proof, we only need an initial condition which is the induction basis. Evidently, any positive integer greater than 1 may serve as the induction basis.

Summarizing these arguments, one can describe the following sequence of steps in writing the proofs by mathematical induction in Graph Theory:

1. Write down the statement P_n, i. e. the theorem to be proved.

2. **Induction basis:** write down and prove the statement P_{n_0}, where property P_{n_0} is directly observed; usually, n is the number of vertices, and $n_0 = 3$, or $n_0 = 4$.

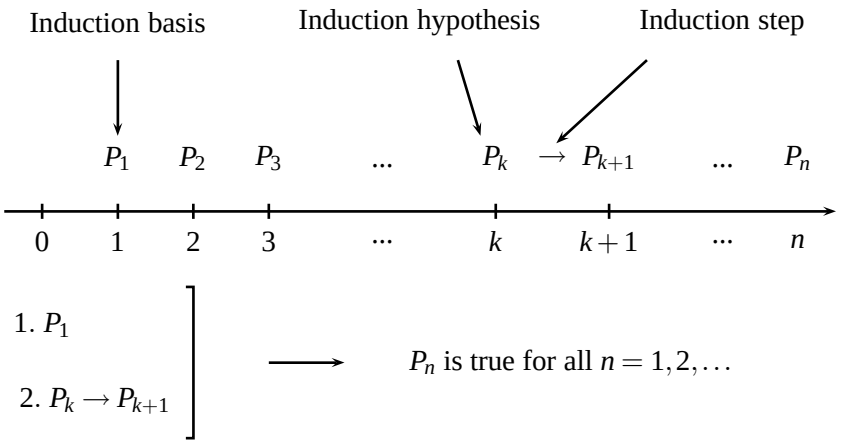


Figure 7.1. Mathematical induction.

3. Write down the statement P_k.
4. Write down the statement P_{k+1}.
5. **Induction step:** prove that if P_k is true, then P_{k+1} is true. In the proof, the assumption that P_k is true, **must be used**. To prove P_{k+1}, one consider a graph on $k+1$ vertices. Then apply to it an operation such as removing a vertex or contraction an edge to obtain a graph having k vertices. At this point it is very important to assure that property P_k (for a graph on k vertices) holds. Finally, considering the inverse operation one prove that property P_{k+1} (for a graph on $k+1$ vertices) fulfills.
6. Conclude that P_n is true for all $n \geq n_0$.

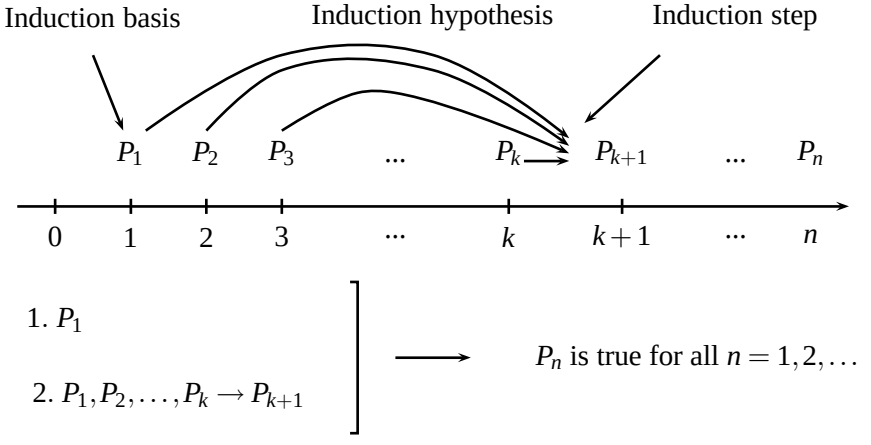


Figure 7.2. Strong mathematical induction.

There is another form of this method which is called the **strong mathematical induction**. The difference is in the induction step. Namely, instead of proving $P_k \to P_{k+1}$, the

strong mathematical induction requires the following: prove that if P_1 is true, and if P_2 is true, and if P_3 is true, and so on, and if P_k is true, then P_{k+1} is true. This can be abbreviated as $P_1, P_2, \ldots, P_k \to P_{k+1}$. Implicitly it is clear that if we started at P_1 and arrived to P_k, then all intermediate statements $P_2, P_3, \ldots, P_{k-1}$ must be true. In some complicated cases we need all these values to prove P_{k+1}, and that is exactly the motivation of applying the strong mathematical induction. The idea is depicted in Figure 7.2. As in simple mathematical induction, evidently, any positive integer greater than 1 may serve as the induction basis.

In Graph Theory proofs, sometimes index k is omitted or hidden; the assumption is "let P hold for all graphs on $< n$ vertices" and then one prove P for a graph on n vertices.

7.2. Graph Theory Algorithms and Their Complexity

Generally, an **algorithm** is a finite set of precise instructions for solving a problem. In graph theory, algorithms consist of sequences of numbered steps (with possible repetitions and checking logical conditions) describing what to do to solve a problem for a graph. In all cases, a graph is in the input, and a number or a special subgraph in many cases is in the output. An algorithm can be programmed and the program can be run on the computer. If we try the same program for different graphs, then we find that time for computations depends on the number of vertices n, i.e., it is some function $f(n)$. If we have another algorithm for the same problem, then the running time is another function of n, say, $g(n)$. How to compare the algorithms? The first algorithm is better if $f(n) \leq g(n)$ beginning with some fixed number of vertices, say $n \geq N_1$. If the first algorithm is run on a computer which is c_1 times faster, and the second algorithm is run on a computer which is c_2 times faster, then the first is better when $f(n)/c_1 \leq g(n)/c_2$ holds for all $n \geq N_2$ where N_2 is some other natural number. This is equivalent to $f(n) \leq Cg(n)$ for some constant $C = c_1/c_2$. Evidently, it holds for any other constant $C' \geq C$ and any other $N \geq N_2$. In other words, no matter how fast the new computers would be, if the inequality $f(n) \leq Cg(n)$ holds for some sufficiently large constant C and all $n \geq N$, then the first algorithm is better. This reasoning explains the meaning of the following notation in comparison of the complexity of algorithms.

We say that $f(n)$ **is** $O(g(n))$ (read "$f(n)$ **is big-oh of** $g(n)$", sometimes denoted by $f(n) = O(g(n))$) if there are constant C and a number N such that

$$f(n) \leq Cg(n)$$

for all $n \geq N$. The basic idea of this definition is that **time is the measure of the complexity** of algorithms, and that time should not depend on the speed of a computer. The comparison of algorithms is **asymptotical**, i.e., what occurs beginning with some N on, or, as we say, when n, the number of vertices, approaches infinity.

In practice, when estimating algorithms, one compute the number of elementary operations (addition, multiplication, comparison, etc.) as a function of n in the *worst case* at every step of the algorithm. That is an upper bound for the complexity. One then say that the complexity of the algorithm is $O(g(n))$ where $g(n)$ is that very same upper bound.

If $g(n) = n$, then the algorithm is called **linear-time** and its complexity is denoted by $O(n)$. Generally, if $g(n)$ is a polynomial of degree k on n, then the algorithm is called **polynomial-time** and complexity is denoted by $O(n^k)$. If $g(n)$ is an exponential function,

then the algorithm is called **exponential-time** and complexity is denoted by $O(a^n)$ where $a > 1$. There is also a **constant complexity** denoted by $O(1)$, **logarithmic complexity** denoted by $O(\log n)$ and even **factorial complexity** denoted by $O(n!)$.

For example, let a graph G be given by its adjacency matrix $A(G)$ of size $n \times n$ and the problem is to find the vertex degrees. Since the matrix is symmetric, it is sufficient to check every entry of the lower triangle if it is 0 or 1. The number of such checks is $(n^2 - n)/2$ which is the polynomial of n. Therefore the complexity of such procedure is $O(n^2)$.

However, many problems in graph theory are much more complex. The best known algorithms for finding the chromatic number of a general graph, or determine if two graphs are isomorphic, for example, are exponential-time.

In general, a problem that can be solved by polynomial-time algorithm is called **tractable**, otherwise it is called **intractable**. Tractable problems form the so called **class P** of problems. There are many problems that *no polynomial-time algorithm* can solve them, but a solution (if known) can be checked in polynomial time. Such problems form the so called **class NP**. At last, there is a class of problems with the property that if any of these problems can be solved in polynomial time, then all of them can be solved in polynomial time because there is a polynomial-time transformation from each other. They form the so called class of **NP-complete problems**.

7.3. Answers and Hints to Selected Exercises

CHAPTER 1

Section 1.1

1. $n(G_1) = 4, m(G_1) = 3, n(G_2) = 5, m(G_2) = 7, n(G_3) = 8, m(G_3) = 12$. **5.** Degree sequences: $G_1 : (1,1,1,3), G_2 : (2,2,2,4,4)$, $G_3 : (2,2,3,3,3,3,4,4)$.

Section 1.3

1. $L(G_1) = \{\{2\},\{1,3,4\},\{2,4\},\{2,3\}\}$. **2.**

$$A(G_1) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

3.

$$I(G_1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

4. $J(G_1) = \{\{1,2\},\{2,3\},\{3,4\},\{2,4\}\}$.

Section 1.5

5. G_5, G_6, G_7, G_8, G_9. **7.** Because $K_{s,r}$ is obtained from $K_{r,s}$ by just interchanging the parts. **8.** $G_1 \cong G_2$, $G_3 \cong G_4$, $G_7 \cong G_8 \cong G_9$. **9.** When the names and order of vertices correspond to the given isomorphism. **10.** When the names and order of vertices correspond to any isomorphism.

Section 1.6
6. $\eta(T) = 2, \eta(C_5) = 3, \eta(W_5) = 4, \eta(K_{2,3} = 3$. **10.** If we reverse the order in degree sequence of G and add it with the degree sequence of $\overline{G}$ as two vectors, then we obtain a vector with all components equal to $n-1$.

Section 1.7
11. In $K_{4,4}$: C_4, C_8, in cube: C_4, C_8, in Petersen graph: C_5, C_9. **13.** For graphs in Figure 1.20: $\omega(G_1) = \omega(G_2) = \omega(G_3) = \omega(G_4) = 3$, $\omega(G_5) = \omega(G_6) = \omega(G_7) = \omega(G_8) = \omega(G_9) = 2$. **14.** For graphs in Figure 1.20: $\alpha(G_1) = \alpha(G_2) = \alpha(G_3) = \alpha(G_4) = 2$, $\alpha(G_5) = \alpha(G_6) = 4$, $\alpha(G_7) = \alpha(G_8) = \alpha(G_9) = 3$. **15.** For graphs in Figure 1.20: $\tau(G_1) = \tau(G_2) = 2$, $\tau(G_3) = \tau(G_4) = 3$, $\tau(G_5) = \tau(G_6) = 4$, $\tau(G_7) = \tau(G_8) = \tau(G_9) = 2$. **16.** For graphs in Figure 1.20: $\nu(G_1) = \nu(G_2) = \nu(G_3) = \nu(G_4) = 2$, $\nu(G_5) = \nu(G_6) = 4$, $\nu(G_7) = \nu(G_8) = \nu(G_9) = 2$. **17.** If n is even: $\alpha(C_n) = \tau(C_n) = \nu(C_n) = n/2$, if n is odd: $\alpha(C_n) = \nu(C_n) = (n-1)/2$, $\tau(C_n) = (n+1)/2$, for all n: $\omega(C_n) = 2$.

Section 1.8
3. $\kappa(G) = 2$. **4.** $k \geq 2$. **12.** 2.

CHAPTER 2

Section 2.1
1. $\Lambda(E_n) = 0$, $\Lambda(C_n) = 1$, $\Lambda(K_n) = (n-1)(n-2)/2$, $\Lambda(W_n) = n-1$.

Section 2.2
1. $diam(T) = 9, radius = 5$.

Section 2.3
2. Minimum weight = 28. **3.** Maximum weight = 49. **4.** n^{n-2}.

Section 2.4
2. $m = n$. **4.** $\tau(K_{m,n}) = \nu(K_{m,n}) = \min\{m,n\}$. **6.** $\tau = \nu = 13$.

CHAPTER 3

Section 3.2
2. G_1 and G_3 are chordal; G_2 is not chordal. **3.** G_2. **7.** $\theta(G_1) = \alpha(G_1) = 3$, $\theta(G_2) = \alpha(G_2) = 5$, $\theta(G_3) = \alpha(G_3) = 3$. **8.** For cube: 6. **9.** From cube: 5.

Section 3.3
1. $M(G_1) = 3, \omega(G_1) - 1 = 2$; $M(G_2) = 4, \omega(G_2) - 1 = 2$; $M(G_3) = 2, \omega(G_3) - 1 = 2$. **2.** G_1, G_2 are not chordal; G_3 is chordal. **3.** $M(C_n) = 2$, $M(K_n) = n-1$, $M(W_n) = 3$; for cube, prism and Petersen graph: $M(G) = 3$. **4.** Take empty graph E_n with sufficient large n; add a vertex and make it adjacent to all the vertices of E_n; repeat the procedure. Observe that $\omega = 2$ for all obtained graphs, and $M(G)$ is increasing by 1 every step until it reaches n. Evidently, the graph obtained is $K_{n,n}$. **5.** Delete vertices by minimum degree; since graph is k-regular, at any step except the first, there is a vertex of degree $\leq k-1$.

Section 3.4
2. $diam(G) = 6$, radius = 3.

Section 3.5
1. G_1 and G_2 are quasi-triangulated, G_3 is not. **2.** C_5 in G_3. **5.** Quasi-triangulated graphs: C_n for $n = 3,4$, K_n for $n \geq 3$, W_n for $n = 4,5$.

CHAPTER 4

Section 4.1
1. G_1: 5; G_2: 7. **3.** Yes.

Section 4.2
1. See Figure 1.28 for prism and Figure 1.16 for cube.

Section 4.3
1. For $K_{3,3}$: 0,1,2,3; for K_5: 0,1,2,3,4. **6.** 1.

Section 4.4
1. K_n: $n = 3,4$; $K_{m,n}$: $m = 1, n \geq 3$, or $m = 2, n \geq 3$; W_n: $n \geq 4$. **3.** G_1 is not planar; G_2 is planar.

CHAPTER 5

Section 5.2
1. $\chi(E_n) = 1$, $\chi(K_n) = n$, $\chi(K_{m,n}) = 2$, $\chi(T_n) = 2$, $\chi(C_{2n}) = 2$, $\chi(C_{2n+1}) = 3$, $\chi(W_{2n}) = 4$, $\chi(W_{2n+1}) = 3$. **4.** $\lambda \geq 136$. **6.** $\chi(G) \leq 6$. **7.** $\chi(G) = 5$.

Section 5.3
1. $P(E_4,\lambda) = \lambda^4$; $P(C_5,\lambda) = (\lambda-1)^5 - (\lambda-1)$; $P(W_4,\lambda) = \lambda^{(4)}$; $P(P_n,\lambda) = \lambda(\lambda-1)^{n-1}$. **6.** The last one.

Section 5.4
2. $S(7,1) = 1$, $S(7,2) = 63$, $S(7,3) = 301$, $S(7,4) = 350$, $S(7,5) = 140$, $S(7,6) = 21$, $S(7,7) = 1$; $s(3,1) = 2$, $s(3,2) = -3$, $s(3,3) = 1$. **3.** $P(G_1,\lambda) = \lambda(\lambda-1)^3(\lambda-2)^2(\lambda^2 - 3\lambda+3)^2$; $P(G_2,\lambda) = \lambda(\lambda-1)(\lambda-2)^4$; $P(G_3,\lambda) = \lambda(\lambda-1)(\lambda-2)^4(\lambda^2-3\lambda+3)$.

Section 5.5
3. Graph $G = (X,E)$ with $X = \{1,2,3,4,5\}$ and adjacency list $L(G) = \{\{2,4\}, \{1,3,4,5\}, \{2,5\}, \{1,2,5\}, \{2,3,4\}\}$; the order of online coloring: 1,2,3,4,5.

Section 5.6
1. For Petersen graph: $M = 3$, therefore $\chi \leq 4$. **2.** Idea: switch the colors in the regions on one side of a newly added line.

Section 5.7
1. None of them is perfect because each contains induced C_5. **2.** W_{2k-1} are perfect, W_{2k} are not, $k \geq 3$.

Section 5.8
1. $\chi'(K_n) = n$ if $n \geq 3$ is odd, $\chi'(K_n) = n-1$ if $n \geq 2$ is even. $\chi'(C_n) = 3$ if $n \geq 3$ is odd, $\chi'(C_n) = 2$ if $n \geq 2$ is even. $\chi'(W_n) = n-1$ for all $n \geq 4$. $\chi'(K_{m,n}) = \max\{m,n\}$.

Section 5.9
2. $\overline{\chi}'(G) = 8$. **5.** $\overline{\chi}'(C_n) = 1$, $\overline{\chi}'(W_n) = n-1$. **6.** For Petersen graph $\overline{\chi}' = 7$, for cube $\overline{\chi}' = 6$, and for prism $\overline{\chi}' = 5$.

CHAPTER 6

Section 6.1
2. Connect each pair of such vertices by an edge.

Section 6.2
1. $K_n : n \geq 3$; $K_{m,n} : m = n$; $W_n : n \geq 4$. **2.** Prism and cube - yes; Petersen graph - not. **3.** Yes.

Section 6.3
1. 8. **4.** Add a new source and connect it with outcoming arcs to all sources; add a new sink and connect it with incoming arcs from all sinks.

7.4. Glossary of Additional Concepts

This glossary contains informal definitions of additional concepts that are most often used in the literature.

- Acyclic graph: graph without cycles
- Acyclic orientation: replacing edges by arcs which produces no directed cycles
- Almost always true: a property which has asymptotic (as $n \to \infty$) probability 1
- Antihole: a subgraph induced by $\overline{C_k}$
- Automorphism: a permutation of the vertices that keeps the adjacency
- Binary tree: a tree with a root in which every non-pendant vertex has at most two neighbors further from the root
- Bipartite Ramsey number: given a bipartite graph G, the minimum n such that any 2-coloring of the edges of $K_{n,n}$ produces a monochromatic copy of G
- Block: maximal 2-connected subgraph

- Cactus: graph in which no two cycles share an edge
- Chinese Postman Problem: to find a shortest closed walk passing through each edge of a weighted graph
- k-choosable graph: when for any lists of colors of length k assigned to vertices, there exists a proper list coloring
- Claw: graph $K_{1,3}$
- Cograph: graph with no induced path P_4
- Color-critical: a graph for which every proper subgraph has the smaller chromatic number (index)
- Density: ratio $m(G)/n(G)$
- Dominating set: a subset S of vertices in a graph such that every vertex not in S has a neighbor from S
- Domination number: the size of a smallest dominating set
- Eigenvalue of a graph: eigenvalue of the adjacency matrix
- Genus of a graph: the minimum genus of a surface on which the graph can be embedded without crossings of edges
- Genus of a surface: the number of handles added to the sphere
- Girth: the length of a shortest cycle
- Greedy algorithm: an algorithm for finding an optimal solution of a problem that takes the best possible choice at each step; it does not guarantee the optimal solution for the whole problem
- Hereditary class: a class of graphs such that any induced subgraph of any graph from the class is also in the class
- Hole: induced subgraph isomorphic to $C_k, k \geq 4$
- Homomorphism: a map $f : V(G_1) \to V(G_2)$ that keeps adjacency
- Hypergraph: a set of vertices with a collection of subsets of arbitrary cardinality called hyperedges; graph is a special case of hypergraph where all hyperedges have cardinality 2
- List coloring: proper vertex coloring of a graph in which every vertex has a list of admissible colors
- Matroid: discrete hereditary structure generalizing linear independence in vector spaces; it has many equivalent formulations; for example, graphic matroid for a graph G consists of edges as "elements" and subsets of edges not forming any cycle as "bases"

- Orientable surface: a surface with two different sides
- Outerplanar graph: a planar graph that can be embedded in the plane with all the vertices on the unbounded face
- Pigeonhole principle: if n items are put into m pigeonholes and $n > m$, then at least one pigeonhole contains more than one item; or, if n vertices are colored with m colors and $n > m$, then there are at least two vertices of the same color
- Ramsey number $R(p,q)$: the minimum n such that any 2-coloring of the edges of K_n produces either a monochromatic copy of K_p or a monochromatic copy of K_q; for example, $R(3,3) = 6$
- Random graph: a graph in which every pair of vertices forms an edge with probability p
- Satisfiability problem: the problem of finding truth values for logical variables such that some logical formula becomes true
- Spectrum of a graph: the set of eigenvalues with their multiplicity
- System of distinct representatives: for a collection of sets, a choice of one element from each set such that all chosen elements are distinct
- Thickness of a graph G: minimum number of planar graphs into which G can be split
- Topological graph theory: the study of graph drawings on different surfaces
- Torus: the orientable surface of genus 1; equivalently, a sphere with added handle
- Total coloring: coloring of both vertices and edges so that no adjacent and no incident elements have the same color
- Tournament: digraph obtained from K_n by replacing edges with arcs (orientation)
- Transitive digraph: if there are arcs (x,y) and (y,z), then there is arc (x,z)
- Traveling Salesman Problem: to find a shortest spanning cycle in a weighted graph
- Traversal: visiting all the vertices or edges of a graph in a special way
- Turán graph: the complete multipartite graph with all parts of almost the same size (different by at most 1)
- Turán's theorem: for a given n, the Tuŕan r-partite graph contains the maximum number of edges and does not contain K_{r+1}

References

[1] J. A. Bondy and U. S. R Murty. *Graph Theory.* Springer, 2008.

[2] R. Diestel. *Graph Theory.* Springer, 2006.

[3] J. Gross and J. Yellen (Editors). *Handbook of Graph Theory.* CRC Press, 2003.

[4] J. Gross and J. Yellen. *Graph Theory and Its Applications, Second Edition*. Chapman & Hall / CRC, 2006.

[5] D. B. West. *Introduction to Graph Theory.* Prentice Hall, 2001.

Index