Algorithms
and Computation
in Mathematics

Volume 5

Dieter Jungnickel

Graphs, Networks and Algorithms

Springer

Algorithms and Computation in Mathematics • Volume 5

Editors

E. Becker M. Bronstein H. Cohen
D. Eisenbud R. Gilman

Springer-Verlag Berlin Heidelberg GmbH

Dieter Jungnickel

Graphs, Networks and Algorithms

With 200 Figures
Translated from the German by Tilla Schade

Springer

Dieter Jungnickel
Universität Augsburg
Lehrstuhl für Diskrete Mathematik
Optimierung und Operations Research
Universitätsstr. 14
D-86135 Augsburg
e-mail: jungnickel@math.uni-augsburg.de

Mathematics Subject Classification (1991): 05-01, 68R10, 68Q25

The English edition is based on the third German edition published by Bibliographisches Institut Wissenschaftsverlag in 1994.

Library of Congress Cataloging - in - Publication Data applied for

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Jungnickel, Dieter:
Graphs, networks and algorithms / Dieter Jungnickel. - Berlin;
Heidelberg; New York; Barcelona; Budapest; Hong Kong;
London; Milan; Paris; Singapore; Tokyo:
Springer, 1999
(Algorithms and computation in mathematics; Vol.5)

ISSN 1431-1550

ISBN 978-3-662-03824-6 ISBN 978-3-662-03822-2 (eBook)
DOI 10.1007/978-3-662-03822-2

Originally published by Springer-Verlag Berlin Heidelberg New York in 1999.

Softcover reprint of the hardcover 1st edition 1999

Typesetting: Camera-ready copy from the author
Cover design: *design & production GmbH*, Heidelberg

SPIN: 10559386 46/3143 - 5 4 3 2 1 0 – Printed on acid-free paper

Preface

During the last few decades, Combinatorial Optimization and Graph Theory have – as the whole field of Combinatorics in general – experienced a particularly fast development. There are various reasons for this fact; one is, for example, that applying combinatorial ways of deduction has become more and more common. However, two developments on the 'outside' of mathematics may have been more important: First, a lot of problems in Combinatorial Optimization arose immediately from everyday practice in engineering and management, for example determining shortest or most reliable paths in traffic or communication networks, maximal or compatible flows or shortest tours, planning connections in traffic networks, coordinating projects or supply and demand problems. Second, practical instances of these tasks, which belong to Operations Research, have become accessible by the development of more and more efficient computer systems. Furthermore, Combinatorial Optimization problems are also important for Complexity Theory, an area in the common intersection of Mathematics and Theoretical Computer Science which deals with the analysis of algorithms. Combinatorial Optimization is a fascinating part of mathematics, and a lot of its fascination – at least for me – comes from its interdisciplinarity and its practical relevance.

The present book concerns mainly that part of Combinatorial Optimization which can be formulated and treated by graph theoretical methods; neither the theory of Linear Programming nor Polyhedral Combinatorics are considered. Simultaneously, the book gives an introduction into Graph Theory, where we restrict ourselves to finite graphs. We motivate the problems by practical interpretations wherever possible.[1] Also, we use an algorithmic point of view, that is, we are not content with knowing that an optimal solution exists (this is trivial to see in most cases anyway), but we are mainly interested in the problem of how to find an optimal (or at least almost optimal) solution as efficiently as possible. Most of the problems we treat have a 'good' algorithmic solution, but we also show how even difficult problems can be treated (for example by approximation algorithms or complete enumeration) using a particular 'hard' problem (namely the famous 'travelling

[1] Most of the subjects we treat here are of great importance for practical applications, for example for VLSI-Layout or for designing traffic or communication networks. We recommend the books by Bermond (1992), Korte, Lovász, Prömel and Schriver (1990) and Lengauer (1990).

salesman problem') as an example. Such techniques are interesting even for problems where it is possible to find an exact solution because they may decrease the amount of calculation work considerably. To be able to judge the quality of algorithms and the degree of difficulty of problems, we introduce the basic ideas of Complexity Theory (in an informal way) and explain one of the main open problems of modern mathematics (namely the question 'P=NP?'). In the first chapters of the book, we will present algorithms in a rather detailed manner but turn to a more concise presentation in later parts. We decided not to include any explicit programs in this book; it should not be too difficult for a reader who is used to writing programs to transfer the given algorithms. Giving programs in any fixed programming language would have meant that the book is likely to be obsolete within a short time; moreover, explicit programs would have obscured the mathematical background of the algorithms. However, we use a structured way of presentation for our algorithms, including special commands based on PASCAL (a rather usual approach). The book contains a lot of exercises and, in the appendix, the corresponding solutions or hints for finding the solution. As in any other discipline, Combinatorial Optimization can be learned best by really working with the material; this is true in particular for understanding the algorithms. Therefore, we urge the reader to work on the exercises seriously (and do the mere calculations as well).

The present book is a translation of a revised version of the third edition of my German text book 'Graphen, Netzwerke und Algorithmen'. The translation and the typesetting was done by Dr. Tilla Schade in collaboration with myself.

The text is based on two courses I gave in the winter term 1984/85 and in the summer term 1985 at the Justus-Liebig-University in Gießen. As the first edition of the book which appeared in 1987 was received quite well, a second edition became necessary in 1990. This second edition was only slightly changed (there were only a few corrections and some additions made, including a further appendix and a number of new references), because it appeared a relatively short time after the first edition. The third edition, however, was completely revised and newly typeset. Besides several corrections and rearrangements, some larger supplements were added and the references brought up to date. The lectures and seminars concerning Combinatorial Optimization and Graph Theory I continued to give regularly (first at the University of Gießen, since the summer term 1993 at the University of Augsburg) were very helpful here. I used the text presented here repeatedly; I also took it as the basis for a workshop for high school students organized by the 'Verein Bildung und Begabung'. This workshop showed that the subjects treated in this book are accessible even to high school students; if they are motivated sufficiently, they approach the problems presented with great interest. Moreover, the German edition has been used regularly at various other universities.

I thank my students and assistants and the students who attended the workshop mentioned above for their constant attention and steady interest. Thanks are due, in particular, to Priv.-Doz. Dr. Dirk Hachenberger and Prof. Dr. Alexander Pott who read the whole manuscript of the (German) third edition with critical accuracy; the remaining errors are my responsibility.

Augsburg, May 1998 Dieter Jungnickel

Table of Contents

1. Basic Graph Theory

The history of Graph Theory begins with a paper by Euler (1736)[1] where he solved the well-known 'Königsberger Brückenproblem'. This problem consists in finding a circular tour through Königsberg using each of the seven bridges over the river Pregel exactly once.

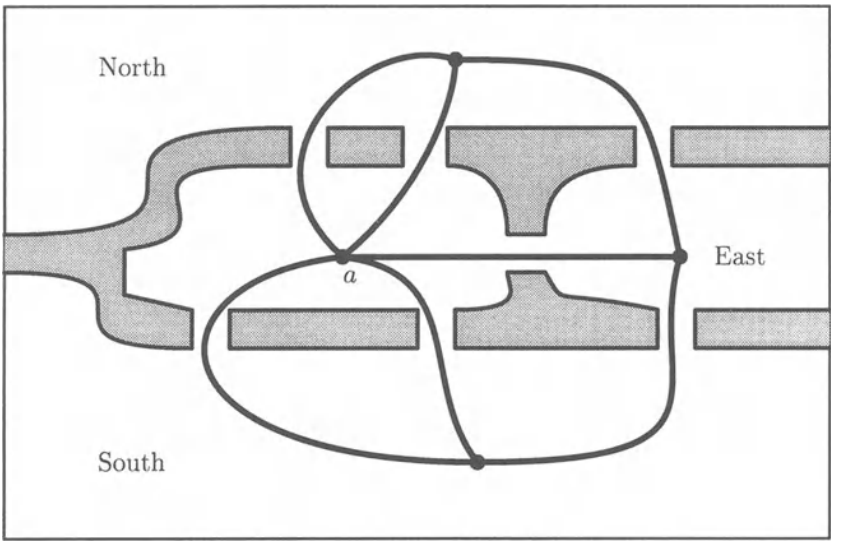

Fig. 1.1 Map of Königsberg and the corresponding graph

When trying to solve this problem one soon gets the feeling that there is no solution. But how can this be proved? Euler realized that the shapes of the islands and of the banks of the river are not important; the solvability depends only on the 'connection properties'. We represent the two islands and the two banks of the river by points (called vertices), and the bridges by lines

[1] see Wilson (1986) and Biggs, Lloyd and Wilson (1976).

between the respective points. Thus we get the 'graph' of the above picture. Trying to find a circular tour, we now begin a tour, say, at the vertex called a. When our tour returns to a for the first time, we have used two of the five bridges connected with a. At our next return to a we have used four bridges. Now we can leave a again using the fifth bridge, but there is no possibility to return to a without using one of the five bridges again. This shows that the problem is indeed unsolvable. Using a similar technique, it can be shown that it is impossible even to find any tour, not necessarily circular, using each bridge exactly once (that is, the tour might end at a different vertex than it begins). Euler proved even more in his paper and gave a necessary and sufficient condition for an arbitrary graph to have a circular tour of the above kind. We will treat his theorem in Section 1.3. But first, we have to introduce some basic notations.

The present chapter contains a lot of definitions. We recommend the reader to work on the exercises to get a better idea of what the terms really mean. Even though the present chapter has a more introductory nature, we will also prove a couple of nontrivial theorems and give two interesting applications. We warn the reader that the terminology in Graph Theory is rather unhomogeneous (although this improved a little after the book by Harary (1969) appeared).

1.1 Graphs, Subgraphs and Factors

A *graph* G is a pair $G = (V, E)$ consisting of a finite[2] set $V \neq \emptyset$ and a set E of two-element subsets of V. The elements of V are called *vertices*. An element $e = \{a, b\}$ of E is called an *edge* with *end vertices* a and b. We say that a and b are *incident* with e and that a and b are *adjacent* or *neighbours* of each other, and write $e = ab$ or $a \overset{e}{—} b$.

There are two important series of examples: The *complete graph* K_n has n vertices (that is, $|V| = n$) and all two-element subsets of V as edges. The *complete bipartite graph* $K_{m,n}$ has as vertex set the disjoint union of a set V_1 with m elements and a set V_2 with n elements; edges are all the sets $\{a, b\}$ with $a \in V_1$ and $b \in V_2$.

We will often illustrate graphs by pictures in the plane. The vertices of a graph $G = (V, E)$ are represented by (bold type) points and the edges by lines (preferably straight lines) connecting the end points. We give some examples:

[2] In Graph Theory, infinite graphs are studied as well. However, we restrict ourselves in this book – like Harary (1969) – to the finite case.

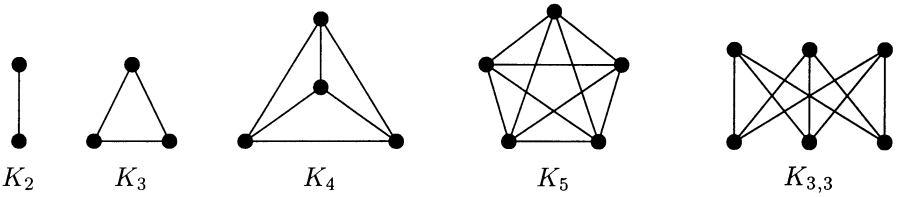

Fig. 1.2 Some graphs

We emphasize that in these pictures the lines merely show with which vertices the edges are incident. In particular, the 'inner points' of these lines as well as possible points of intersection of two edges (as in Figure 1.2 for the graphs K_5 and $K_{3,3}$) are not significant. In Section 1.5 we will study the question which graphs can be drawn without such additional points of intersection.

Let $G = (V, E)$ be a graph and V' be a subset of V. By $E|V'$ we denote the set of all edges e which have both their vertices in V'. The graph $(V', E|V')$ is called the *induced subgraph* on V' and is denoted by $G|V'$. Any graph $G' = (V', E')$ where $V' \subset V$ and $E' \subset E|V'$ is called a *subgraph* of G. A subgraph with $V' = V$ is called a *spanning subgraph*. Some examples are given in Figure 1.3.

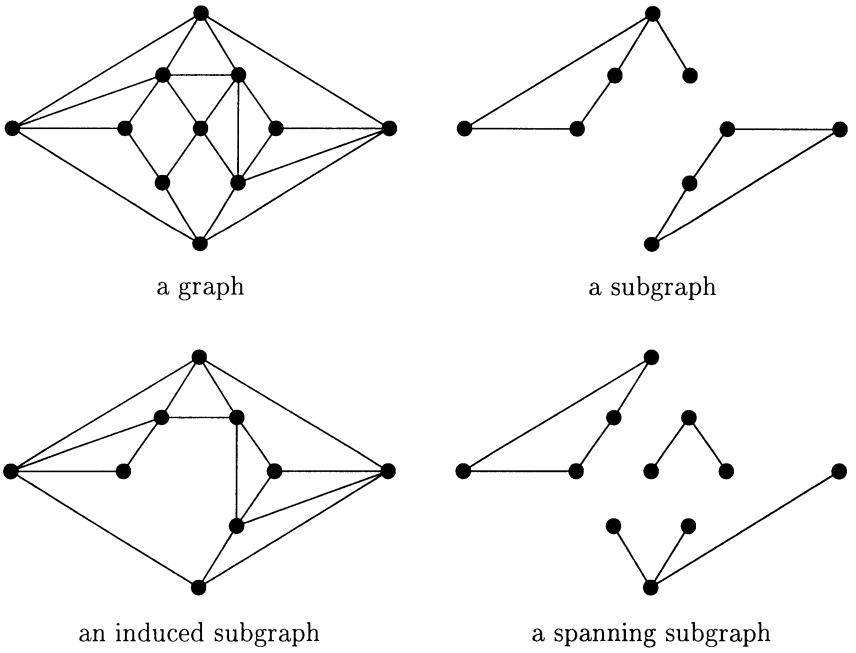

Fig. 1.3 Subgraphs

For any vertex v of a graph, the *degree* $\deg v$ of v is the number of edges incident with v. We have the first (almost trivial) result:

Lemma 1.1.1. *In any graph, the number of vertices of odd degree is even.*

Proof. Summing the degree $\deg v$ over all vertices v, each edge is counted exactly twice (once for each of its vertices). Thus we have $\sum_v \deg v = 2|E|$. As the right hand side is even, the number of odd terms $\deg v$ in the sum on the left hand side must be even. □

If all vertices of a graph G have the same degree r, G is called a *regular* graph, more precisely an *r-regular* graph. The graph K_n is $(n-1)$-regular, the graph $K_{m,n}$ is regular only if $m = n$ (in which case it is n-regular). A *k-factor* is a k-regular spanning subgraph. If the edge set of a graph can be divided into k-factors, such a decomposition is called a *k-factorization* of the graph. A 1-factorization is also called a *factorization* or a *resolution*. Obviously, a 1-factor can exist only if G has an even number of vertices. Factorizations of K_{2n} can be interpreted as schedules for a tournament of $2n$ teams (in soccer, basketball etc.). The following exercise shows that such a factorization exists for any n. The problem of setting up schedules for tournaments will be studied in Section 1.7 as an application.

Exercise 1.1.2. We use $\{a, b_1, \ldots, b_{2n-1}\}$ as the vertex set of the complete graph K_{2n} and divide the edge set into subsets E_i for $i = 1, \ldots, 2n-1$, where $E_i = \{ab_i\} \cup \{b_j b_k : j + k \equiv 2i \pmod{2n-1}\}$. Show that the E_i form a factorization of K_{2n}. (The case $n = 3$ is shown in Figure 1.4.) 1-factorizations were introduced by Kirkman (1847); interesting surveys are given by Mendelsohn and Rosa (1985) and Wallis (1992).

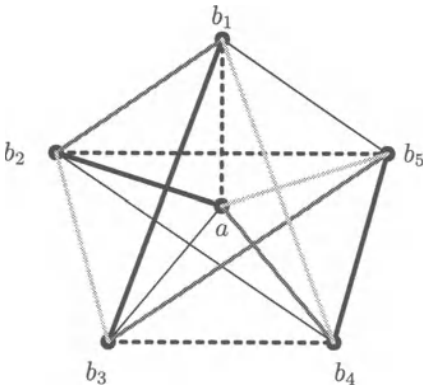

Fig. 1.4 A factorization of K_6

At the end of this section, we introduce one more series of graphs. The *triangular graph* T_n has as vertices the two-element subsets of a set with n elements. Two of these vertices are adjacent if and only if their intersection is not empty. Obviously, T_n is a $(2n-4)$-regular graph. But T_n has even stronger regularity properties: the number of vertices adjacent to two given vertices x, y depends only on the fact whether x and y themselves are adjacent or not. Such a graph is called a *strongly regular graph*, abbreviated by *SRG*. These graphs are of great interest in Finite Geometry; see the books by Cameron and van Lint (1991) and Beth, Jungnickel and Lenz (1998). We will not look at SRG's any further in this book and only give some exercises concerning this concept:

Exercise 1.1.3. Draw the graphs T_n for $n = 3, 4, 5$ and show that T_n has parameters $a = 2n-4$, $c = n-2$ and $d = 4$, where a is the degree of any vertex, c is the number of vertices adjacent to both x and y if x and y are adjacent, and d is the number of vertices adjacent to x and y if x and y are not adjacent.

For the next exercise, we need another definition. For a graph $G = (V, E)$, the graph $\overline{G} = (V, \binom{V}{2} \backslash E)$ is called the *complementary graph*. Two vertices of V are adjacent in $\overline{G}$ if and only if they are not adjacent in G.

Exercise 1.1.4. Let G be an SRG with parameters a, c and d having n vertices. Show that $\overline{G}$ is also an SRG and give its parameters. Moreover, prove the formula

$$a(a-c-1) = (n-a-1)d.$$

(Hint: Count the number of edges yz for which y is adjacent to a given vertex x, and z is not adjacent to x.)

1.2 Paths, Cycles, Connectedness, Trees

Before we can go on to the Theorem of Euler mentioned in Section 1.1, we have to give a formal definition of what a 'circular tour' really means. Let $(e_1, \ldots, e_n)$ be a sequence of edges in a graph G. If there are vertices $v_0, \ldots, v_n$ such that $e_i = v_{i-1}v_i$ for $i = 1, \ldots, n$, the sequence is called a *walk*, for $v_0 = v_n$ a *closed walk*. A walk for which the e_i are pairwise distinct is called a *trail*, a closed walk with that property is a *closed trail*. If, in addition, the v_j are pairwise distinct, the trail is a *path*. A closed trail with $n \geq 3$, for which the v_j are pairwise distinct (except, of course, $v_0 = v_n$), is called a *cycle*. In any of these cases we use the notation

$$W: \quad v_0 \overset{e_1}{\text{———}} v_1 \overset{e_2}{\text{———}} v_2 \text{———} \ldots \text{———} v_{n-1} \overset{e_n}{\text{———}} v_n$$

and call n the *length* of W. The vertices v_0 and v_n are called the *start vertex* and the *end vertex* of W, respectively. We will sometimes specify a walk

by the sequence of vertices $(v_0, \ldots, v_n)$, provided that $v_{i-1}v_i$ is an edge for $i = 1, \ldots, n$. In the graph of the following picture, (a, b, c, v, b, c) is a walk, but not a trail; and (a, b, c, v, b, u) is a trail, but not a path. Also, (a, b, c, v, b, u, a) is a closed trail, but not a cycle, whereas (a, b, c, w, v, u, a) is a cycle. We suggest the reader to consider some more examples.

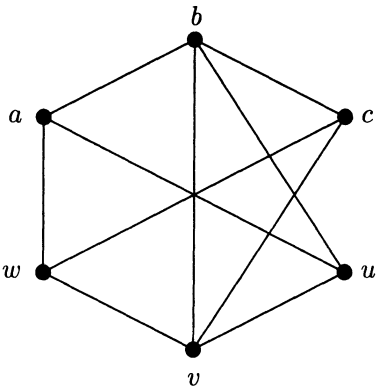

Fig. 1.5 An example for walks

Exercise 1.2.1. Show that a closed walk of odd length contains a cycle. What do closed walks not containing a cycle look like?

Two vertices a and b of a graph G are called *connected* if there exists a walk with start vertex a and end vertex b. If any two vertices of G are connected, G is called *connected*. For any vertex a, we consider (a) as a trivial walk (of length 0), so that any vertex is connected with itself. Then connectedness is an equivalence relation on the vertex set of G. The equivalence classes of this relation are called the *connected components* of G. Thus G is connected if and only if its vertex set V is a connected component. Components which contain only one vertex are also called *isolated vertices*. We give some exercises concerning the above definitions:

Exercise 1.2.2. Let G be a graph having n vertices and let any vertex of G have degree at least $\frac{n-1}{2}$. Show that G must be connected.

Exercise 1.2.3. A graph G is connected if and only if, for any decomposition $V = V_1 \dot{\cup} V_2$ (that is, $V_1 \cap V_2 = \emptyset$) of the vertex set of G, there exists an edge $e = vw$ such that $v \in V_1$ and $w \in V_2$.

Exercise 1.2.4. If G is not connected, the complementary graph $\overline{G}$ is connected.

If a and b are two vertices in the same connected component of a graph G, obviously there is a path of shortest length d between a and b. (Why?) Then a and b are said to have *distance* $d = d(a, b)$. The notion of distances in a graph is a very fundamental one in Graph Theory; we will investigate it (and a generalization) thoroughly in Chapter 3.

The rest of this section is devoted to characterizing and examining the minimal connected graphs. First, some more definitions and an exercise. A graph is called *acyclic* if it does not contain a cycle. For a subset T of the vertex set V of a graph G we denote by $G\backslash T$ the induced subgraph on $V\backslash T$. This graph arises from G by omitting all vertices in T and all edges incident with these vertices. For a one-element set $T = \{v\}$ we write $G\backslash v$ instead of $G\backslash\{v\}$.

Exercise 1.2.5. Let G be a graph without isolated vertices having n vertices and $n-1$ edges (where $n \geq 2$). Show that G contains at least two vertices of degree 1.

Lemma 1.2.6. *Any connected graph on n vertices contains at least $n-1$ edges.*

Proof. We use induction on n; the case $n = 1$ is trivial. So let G be a connected graph on $n \geq 2$ vertices. Choose an arbitrary vertex v of G and consider the graph $H = G\backslash v$. H is not necessarily connected, so suppose H has connected components Z_i having n_i vertices ($i = 1, \ldots, k$), that is, $n_1 + \ldots + n_k = n - 1$. By induction hypothesis, the subgraph of H induced on Z_i has at least $n_i - 1$ edges. Moreover, v must be connected in G with each of the components Z_i by at least one edge. Thus G contains at least

$$(n_1 - 1) + \ldots + (n_k - 1) + k = n - 1$$

edges. □

Lemma 1.2.7. *Any acyclic graph on n vertices has at most $n-1$ edges.*

Proof. If $n = 1$ or $E = \emptyset$, the statement is obvious. For the general case, choose any edge $e = ab$ in G. Then the graph $H = G\backslash e$ has exactly one more connected component than G. (Note that there cannot be a path in H from a to b, because such a path together with the edge e would give rise to a cycle in G.) Thus, H can be decomposed into connected, acyclic graphs $H_1, \ldots, H_k$ (where $k \geq 2$). By induction, we can assume that each graph H_i contains at most $n_i - 1$ edges, where n_i is the number of vertices of H_i. But then G has at most $(n_1 - 1) + \ldots + (n_k - 1) + 1 = (n_1 + \ldots + n_k) - (k - 1) \leq n - 1$ edges. □

Theorem 1.2.8. *Let G be a graph with n vertices. Then any two of the following conditions imply the third:*

(a) G is connected.
(b) G is acyclic.
(c) G has $n-1$ edges.

Proof.

(i) Let G be acyclic and connected. Then Lemmas 1.2.6 and 1.2.7 imply that G has exactly $n-1$ edges.

(ii) Now let G be a connected graph having $n-1$ edges. Suppose G contains a cycle C, then the graph $H = G\backslash e$ (where e is any edge of C) is a connected graph with n vertices and $n-2$ edges. This contradicts Lemma 1.2.6.

(iii) Suppose G is an acyclic graph having $n-1$ edges. Then Lemma 1.2.7 implies that G cannot contain an isolated vertex (note that omitting such a vertex would give an acyclic graph with $n-1$ vertices and $n-1$ edges). By Exercise 1.2.5, G then has a vertex of degree 1, so that $G\backslash v$ is an acyclic graph with $n-1$ vertices and $n-2$ edges. By induction it follows that $G\backslash v$ and hence G are connected.

□

Exercise 1.2.9. Give a different proof for Lemma 1.2.6 using the technique of omitting an edge e from G.

A graph T for which the conditions of Theorem 1.2.8 hold is called a *tree*. Any vertex of T having degree 1 is called a *leaf*. A *forest* is a graph whose connected components are trees. We will have a closer look at trees in Chapter 4.

In Section 4.2 we will prove the formula for the number of trees on n vertices due to Borchardt (1860) using rather sophisticated techniques from Linear Algebra. Now we give a much more elementary proof which furthermore has the advantage of proving a stronger result[3]. By $f(n,s)$ we denote the number of forests G having n vertices and exactly s connected components, for which s fixed vertices are in pairwise distinct components. Then the number of trees on n vertices is $f(n,1)$. The following theorem due to Cayley (1889) gives a formula for the numbers $f(n,s)$; we give a simple proof due to Takács (1990a).

Theorem 1.2.10. *We have $f(n,s) = sn^{n-s-1}$.*

Proof. We begin by proving the following recursion formula:

$$f(n,s) = \sum_{j=0}^{n-s} \binom{n-s}{j} f(n-1, s+j-1), \tag{1.1}$$

[3] The reader might skip the rest of this section during the first reading, but should come back after having read Section 4.2 for comparison.