Chapter 1

What Is Combinatorics?

1.1 THE THREE PROBLEMS OF COMBINATORICS

Perhaps the fastest-growing area of modern mathematics is combinatorics. Combinatorics is concerned with the study of arrangements, patterns, designs, assignments, schedules, connections, and configurations. In the modern world, people in almost every area of activity find it necessary to solve problems of a combinatorial nature. A computer scientist considers *patterns* of digits and switches to encode complicated statements. A shop supervisor prepares *assignments* of workers to tools or to work areas. An agronomist *assigns* test crops to different fields. An electrical engineer considers alternative *configurations* for a circuit. A banker studies alternative *patterns* for electronically transferring funds, and a space scientist studies such *patterns* for transferring messages to distant satellites. An industrial engineer considers alternative production *schedules* and workplace *configurations* to maximize efficient production. A university scheduling officer *arranges* class meeting times and students' *schedules*. A chemist considers possible *connections* between various atoms and molecules, and *arrangements* of atoms into molecules. A transportation officer *arranges* bus or plane *schedules*. A linguist considers *arrangements* of words in unknown alphabets. A geneticist considers *arrangements* of bases into chains of DNA, RNA, and so on. A statistician considers alternative *designs* for an experiment.

There are three basic problems of combinatorics. They are the *existence problem*, the *counting problem*, and the *optimization problem*. The existence problem deals with the question: Is there at least one arrangement of a particular kind? The counting problem asks: How many arrangements are there? The optimization problem is concerned with choosing, among all possible arrangements, that which is best according to some criteria. We shall illustrate these three problems with a number of examples.

Example 1.1 Design of Experiments Let us consider an experiment designed to test the effect on human beings of five different drugs. Let the drugs be labeled $1, 2, 3, 4, 5$. We could pick out five subjects and give each subject a different drug.

Table 1.1: A Design for a Drug Experiment[a]

		Day				
		M	Tu	W	Th	F
Subject	A	1	2	3	4	5
	B	1	2	3	4	5
	C	1	2	3	4	5
	D	1	2	3	4	5
	E	1	2	3	4	5

[a] The entry in the row corresponding to a given subject and the column corresponding to a given day shows the drug taken by that subject on that day.

Unfortunately, certain subjects might be allergic to a particular drug, or immune to its effects. Thus, we could get very biased results. A more effective use of five subjects would be to give each subject each of the drugs, say on five consecutive days. Table 1.1 shows one possible arrangement of the experiment. What is wrong with this arrangement? For one thing, the day of the week a drug is taken may affect the result. (People with Monday morning hangovers may never respond well to a drug on Monday.) Also, drugs taken earlier might affect the performance of drugs taken later. Thus, giving each subject the drugs in the same order might lead to biased results. One way around these problems is simply to require that no two people get the same drug on the same day. Then the experimental design calls for a 5×5 table, with each entry being one of the integers $1, 2, 3, 4, 5$, and with each row having all its entries different and each column having all its entries different. This is a particular kind of pattern. The crucial question for the designer of the drug experiment is this: Does such a design exist? This is the existence problem of combinatorics. ■

Let us formulate the problem more generally. We define a *Latin square*[1] as an $n \times n$ table that uses the numbers $1, 2, \ldots, n$ as entries, and does so in such a way that no number appears more than once in the same row or column. Equivalently, it is required that each number appear exactly once in each row and column. A typical existence problem is the following: Is there a 2×2 Latin square? The answer is yes; Table 1.2 shows such a square. Similarly, one may ask if there is a 3×3 Latin square. Again, the answer is yes; Table 1.3 shows one.

Our specific question asks whether or not there is a 5×5 Latin square. Table 1.4 shows that the answer is yes. (Is there an $n \times n$ Latin square for every n? The answer is known and is left to the reader.)

[1] The term "Latin square" comes from the fact that the elements were usually represented by letters of the Latin alphabet.

Table 1.2: A 2×2 Latin Square

1	2
2	1

Table 1.3: A 3×3 Latin Square

1	2	3
2	3	1
3	1	2

Table 1.4: A 5×5 Latin Square

1	2	3	4	5
2	3	4	5	1
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

Note that the Latin square is still not a complete solution to the problem that order effects may take place. To avoid any possible order effects, we should ideally have enough subjects so that each possible ordering of the 5 drugs can be tested. How many such orderings are there? This is the *counting problem*, the second basic type of problem encountered in combinatorics. It turns out that there are $5! = 120$ such orderings, as will be clear from the methods of Section 2.3. Thus, we would need 120 subjects. If only 5 subjects are available, we could try to avoid order effects by choosing the Latin square we use at random. How many possible 5×5 Latin squares are there from which to choose? We address this counting problem in Section 6.1.3.

As this very brief discussion suggests, questions of experimental design have been a major stimulus to the development of combinatorics.[2] We return to experimental design in detail in Chapter 9.

Example 1.2 Bit Strings and Binary Codes A *bit* or *binary digit* is a zero or a one. A *bit string* is defined to be a sequence of bits, such as 0001, 1101, or 1010. Bit strings are the crucial carriers of information in modern computers. A bit string can be used to encode detailed instructions, and in turn is translated into a sequence of on-off instructions for switches in the computer. A *binary code (binary block code)* for a collection of symbols assigns a different bit string to each of the symbols. Let us consider a binary code for the 26 letters in the alphabet. A typical such code is the Morse code which, in its more traditional form, uses dots for zeros and dashes for ones. Some typical letters in Morse code are given as follows:

$$\text{O: } 111, \ \text{A: } 01, \ \text{K: } 101, \ \text{C: } 1010.$$

If we are restricted to bit strings consisting of either one or two bits, can we encode all 26 letters of the alphabet? The answer is no, for the only possible strings are the following:

$$0, 1, 00, 01, 10, 11.$$

There are only six such strings. Notice that to answer the question posed, we had to *count* the number of possible arrangements. This was an example of a solution to a counting problem. In this case we counted by *enumerating* or listing all possible

[2] See Herzberg and Stanton [1984].

arrangements. Usually, this will be too tedious or time consuming for us, and we will want to develop shortcuts for counting without enumerating. Let us ask if bit strings of three or fewer bits would do for encoding all 26 letters of the alphabet. The answer is again no. A simple enumeration shows that there are only 14 such strings. (Can you list them?) However, strings of four or fewer bits will suffice. (How many such strings are there?) The Morse code, indeed, uses only strings of four or fewer symbols. Not every possible string is used. (Why?) In Section 2.1 we shall encounter a very similar counting problem in studying the genetic code. DNA chains encode the basic genetic information required to determine long strings of amino acids called proteins. We shall try to explain how long a segment in a DNA chain is required to be to encode for an amino acid. Codes will arise in other parts of this book as well, not just in the context of genetics or of communication with modern computers. For instance, in Chapter 10 we study the error-correcting codes that are used to send and receive messages to and from distant space probes, to fire missiles, and so on. ■

Example 1.3 The Best Design for a Gas Pipeline The flow of natural gas through a pipe depends on the diameter of the pipe, its length, the pressures at the endpoints, the temperature, various properties of the gas, and so on. The problem of designing an offshore gas pipeline system involves, among other things, decisions about what sizes (diameters) of pipe to use at various junctions or links so as to minimize total cost of both construction and operation. A standard approach to this problem has been to use "engineering judgment" to pick reasonable sizes of pipe and then to hope for the best. Any chance of doing better seems, at first glance, to be hopeless. For example, a modest network of 40 links, with 7 possible pipe sizes for each link, would give rise to 7^{40} possible networks, as we show in Section 2.1. Now 7^{40}, as we shall see, is a very large number. Our problem is to find the least expensive network out of these 7^{40} possibilities. This is an example of the third kind of combinatorial problem, an *optimization problem*, a problem where we seek to find the optimum (best, maximum, minimum, etc.) design or pattern or arrangement.

It should be pointed out that progress in solving combinatorial optimization problems has gone hand in hand with the development of the computer. Today it is possible to solve on a machine problems whose solution would have seemed inconceivable only a few years ago. Thus, the development of the computer has been a major impetus behind the very rapid development of the field of combinatorial optimization. However, there are limitations to what a computing machine can accomplish. We shall see this next.

Any finite problem can be solved in principle by considering all possibilities. However, how long would this particular problem take to solve by enumerating all possible pipeline networks? To get some idea, note that 7^{40} is approximately 6×10^{33}, that is, 6 followed by 33 zeros. This is a huge number. Indeed, even a computer that could analyze 1 billion (10^9) different pipeline networks in 1 second (one each nanosecond) would take approximately $1.9 \times 10^{17} = 190{,}000{,}000{,}000{,}000$ years

to analyze all 7^{40} possible pipeline networks![3]

Much of modern combinatorics is concerned with developing procedures or *algorithms* for solving existence, counting, or optimization problems. From a practical point of view, it is a very important problem in computer science to analyze an algorithm for solving a problem in terms of how long it would take to solve or how much storage capacity would be required to solve it. Before embarking on a computation (such as trying all possibilities) on a machine, we would like to know that the computation can be carried out within a reasonable time or within the available storage capacity of the machine. We return to these points in our discussion of computational complexity in Sections 2.4 and 2.18.

The pipeline problem we have been discussing is a problem that, even with the use of today's high-speed computer tools, does not seem tractable by examining all cases. Any foreseeable improvements in computing speed would make a negligible change in this conclusion. However, a simple procedure gives rise to a method for finding the optimum network in only about $7 \times 40 = 280$ steps, rather than 7^{40} steps. The procedure was implemented in the Gulf of Mexico at a savings of millions of dollars. See Frank and Frisch [1970], Kleitman [1976], Rothfarb, *et al.* [1970], or Zadeh [1973] for references. This is an example of the power of techniques for combinatorial optimization. ■

Example 1.4 Scheduling Meetings of Legislative Committees Committees in a state legislature are to be scheduled for a regular meeting once each week. In assigning meeting times, the aide to the Speaker of the legislature must be careful not to schedule simultaneous meetings of two committees that have a member in common. Let us suppose that in a hypothetical situation, there are only three meeting times available: Tuesday, Wednesday, and Thursday mornings. The committees whose meetings must be scheduled are Finance, Environment, Health, Transportation, Education, and Housing. Let us suppose that Table 1.5 summarizes which committees have a common member. A convenient way to represent the information of Table 1.5 is to draw a picture in which the committees are represented by dots or points and two points are joined by an undirected line if and only if the corresponding committees have a common member. The resulting diagram is called a *graph*.

Figure 1.1 shows the graph obtained in this way for the data of Table 1.5. Graphs of this kind have a large number of applications, for instance in computer science, operations research, electrical engineering, ecology, policy and decision science, and in the social sciences. We discuss graphs and their applications in detail in Chapters 3 and 11 and elsewhere.

Our first question is this: Given the three available meeting times, can we find an assignment of committees to meeting times so that no member has to be at

[3]There are roughly 3.15×10^7 seconds per year, so $3.15 \times 10^7 \times 10^9$ or 3.15×10^{16} networks could be analyzed in a year. Then the number of years it takes to check 6×10^{33} networks is

$$\frac{6 \times 10^{33}}{3.15 \times 10^{16}} \approx 1.9 \times 10^{17}.$$

Table 1.5: Common Membership in Committees[a]

	Finance	Environment	Health	Transportation	Education	Housing
Finance	0	0	0	0	0	1
Environment	0	0	1	0	1	0
Health	0	1	0	1	1	1
Transportation	0	0	1	0	0	1
Education	0	1	1	0	0	1
Housing	1	0	1	1	1	0

[a] The i, j entry is 1 if committees i and j have a common member, and 0 otherwise. (The diagonal entries are taken to be 0 by convention.)

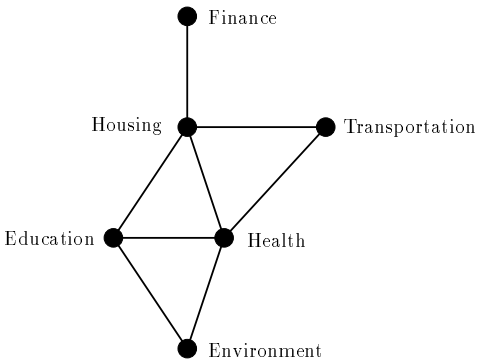

Figure 1.1: The graph obtained from the data of Table 1.5.

two meetings at once? This is an existence question. In terms of the graph we have drawn, we would like to assign a meeting time to each point so that if two points are joined by a line, they get different meeting times. Can we find such an assignment? The answer in our case, after some analysis, is yes. One assignment that works is this: Let the Housing and Environment committees meet on Tuesday, the Education and Transportation committees on Wednesday, and the Finance and Health committees on Thursday.

Problems analogous to the one we have been discussing arise in scheduling final exams or class meeting times in a university, in scheduling job assignments in a factory, and in many other scheduling situations. We shall return to such problems in Chapter 3 when we look at these questions as questions of *graph coloring* and think of the meeting times, for example, as corresponding to "colors."

The problem gets more realistic if each committee chair indicates a list of acceptable meeting times. We then ask if there is an assignment of committees to meeting times so that each committee is assigned an acceptable time and no member has to be at two meetings at once. For instance, suppose that the acceptable meeting times for Transportation are Tuesday and Thursday, for Education is Wednesday, and all other committees would accept any of the three days. It is not hard to show that there is no solution (see Exercise 13). We will then have solved the existence problem in the negative. This is an example of a scheduling problem known as a

Table 1.6: First Choice of Meeting Times

Committee	Finance	Environment	Health	Transportation	Education	Housing
Chair's first choice	Tuesday	Thursday	Thursday	Tuesday	Tuesday	Wednesday

list-coloring problem, a graph coloring problem where assigned colors (in this case representing "days of the week") are chosen from a list of acceptable ones. We return to this problem in Example 3.22. List colorings have been widely studied in recent years. See Alon [1993] and Kratochvíl, Tuza, and Voigt [1999] for recent surveys.

We might ask next: Suppose that each committee chair indicates his or her first choice for a meeting time. What is the assignment of meeting times that satisfies our original requirements (if there is such an assignment) and gives the largest number of committee chairs their first choice? This is an optimization question. Let us again take a hypothetical situation and analyze how we might answer this question. Suppose that Table 1.6 gives the first choice of each committee chair. One approach to the optimization question is simply to try to identify all possible satisfactory assignments of meeting times and for each to count how many committee chairs get their first choice. Before implementing any approach to a combinatorial problem, as we have observed before, we would like to get a feeling for how long the approach will take. How many possibilities will have to be analyzed? This is a counting problem. We shall solve this counting problem by enumeration. It is easy to see from the graph of Figure 1.1 that Housing, Education, and Health must get different times. (Each one has a line joining it to the other two.) Similarly, Transportation must get a different time from Housing and Health. (Why?) Hence, since only three meeting times are available, Transportation must meet at the same time as Education. Similarly, Environment must meet at the same time as Housing. Finally, Finance cannot meet at the same time as Housing, and therefore as Environment, but could meet simultaneously with any of the other committees. Thus, there are only two possible meeting patterns. They are as follows.

Pattern 1. Transportation and Education meet at one time, Environment and Housing at a second time, and Finance and Health meet at the third time.

Pattern 2. Transportation, Education, and Finance meet at one time, Environment and Housing meet at a second time, and Health meets at the third time.

It follows that Table 1.7 gives all possible assignments of meeting times. In all, there are 12 possible. Our counting problem has been solved by enumerating all possibilities. (In Section 3.4.1 we do this counting another way.) It should be clear from Example 1.3 that enumeration could not always suffice for solving combinatorial problems. Indeed, if there are more committees and more possible meeting times, the problem we have been discussing gets completely out of hand.

Having succeeded in enumerating in our example, we can easily solve the optimization problem. Table 1.7 shows the number of committee chairs getting their first choice under each assignment. Clearly, assignment number 7 is the best from this point of view. Here, only the chair of the Environment committee does not get his or her first choice. For further reference on assignment of meeting times for state legislative committees, see Bodin and Friedman [1971]. For work on other scheduling problems where the schedule is repeated periodically (e.g., every week), see, for instance, Ahuja, Magnanti, and Orlin [1993], Baker [1976], Bartholdi, Orlin, and Ratliff [1980], Chrétienne [2000], Crama, *et al.* [2000], Karp and Orlin [1981], Orlin [1982], or Tucker [1975]. For surveys of various workforce scheduling algorithms, see Brucker [1998], Kovalëv, *et al.* [1989], or Tien and Kamiyama [1982]. ■

This book is organized around the three basic problems of combinatorics that we have been discussing. It has four parts. After an introductory part consisting of Chapters 2 to 4, the remaining three parts deal with these three problems: the counting problem (Chapters 5 to 8), the existence problem (Chapters 9 to 11), and the optimization problem (Chapters 12 and 13).

1.2 THE HISTORY AND APPLICATIONS OF COMBINATORICS[4]

The four examples described in Section 1.1 illustrate some of the problems with which combinatorics is concerned. They were chosen from a variety of fields to illustrate the variety of applications of combinatorics in modern times.

Although combinatorics has achieved its greatest impetus in modern times, it is an old branch of mathematics. According to legend, the Chinese Emperor Yu (in approximately 2200 B.C.) observed a magic square on the back of a divine tortoise. (A *magic square* is a square array of numbers in which the sum of all rows, all columns, and both diagonals is the same. An example of such a square is shown in Table 1.8. The reader might wish to find a different 3×3 magic square.)

Permutations or arrangements in order were known in China before 1100 B.C. The binomial expansion [the expansion of $(a+b)^n$] was known to Euclid about 300 B.C. for the case $n = 2$. Applications of the formula for the number of permutations of an n-element set can be found in an anonymous Hebrew work, *Sefer Yetzirah*, written between A.D. 200 and 500. The formula itself was known at least 2500 years ago. In A.D. 1100, Rabbi Ibn Ezra knew the formula for the number of combinations of n things taken r at a time, the binomial coefficient. Shortly thereafter, Chinese, Hindu, and Arab works began mentioning binomial coefficients in a primitive way.

In more modern times, the seventeenth-century scholars Pascal and Fermat pursued studies of combinatorial problems in connection with gambling—among other things, they figured out odds. (Pascal's famous triangle was in fact known to Chu

[4]For a more detailed discussion of the history of combinatorics, see Biggs, Lloyd, and Wilson [1995] or David [1962]. For the history of graph theory, see Biggs, Lloyd, and Wilson [1976].

Table 1.7: Possible Assignments of Meeting Times

Assignment number	Tuesday	Wednesday	Thursday	Number of committee chairs getting their first choice
1	Transportation-Education	Environment-Housing	Finance-Health	4
2	Transportation-Education	Finance-Health	Environment-Housing	3
3	Environment-Housing	Transportation-Education	Finance-Health	1
4	Environment-Housing	Finance-Health	Transportation-Education	0
5	Finance-Health	Transportation-Education	Environment-Housing	2
6	Finance-Health	Environment-Housing	Transportation-Education	2
7	Transportation-Education-Finance	Environment-Housing	Health	5
8	Transportation-Education-Finance	Health	Environment-Housing	4
9	Environment-Housing	Transportation-Education-Finance	Health	1
10	Environment-Housing	Health	Transportation-Education-Finance	0
11	Health	Transportation-Education-Finance	Environment-Housing	1
12	Health	Environment-Housing	Transportation-Education-Finance	1

Table 1.8: A Magic Square

4	9	2
3	5	7
8	1	6

Shih-Chieh in China in 1303.) The work of Pascal and Fermat laid the groundwork for probability theory; in the eighteenth century, Laplace defined probability in terms of number of favorable cases. Also in the eighteenth century, Euler invented graph theory in connection with the famous Königsberg bridge problem and Bernoulli published the first book presenting combinatorial methods, *Ars Conjectandi*. In the eighteenth and nineteenth centuries, combinatorial techniques were applied to study puzzles and games, by Hamilton and others. In the nineteenth century, Kirchhoff developed a graph-theoretical approach to electrical networks and Cayley developed techniques of enumeration to study organic chemistry. In modern times, the techniques of combinatorics have come to have far-reaching, significant applications in computer science, transportation, information processing, industrial planning, electrical engineering, experimental design, sampling, coding, genetics, political science, and a variety of other important fields. In this book we always keep the applications close at hand, remembering that they are not only a significant benefit derived from the development of the mathematical techniques, but they are also a stimulus to the continuing development of these techniques.

EXERCISES FOR CHAPTER 1

1. Find a 4×4 Latin square.
2. Find all possible 3×3 Latin squares.
3. Describe how to create an $n \times n$ Latin square.
4. (Liu [1972]) Suppose that we have two types of drugs to test simultaneously, such as headache remedies and fever remedies. In this situation, we might try to design an experiment in which each type of drug is tested using a Latin square design. However, we also want to make sure that, if at all possible, all combinations of headache and fever remedies are tested. For example, Table 1.9 shows two Latin square designs if we have 3 headache remedies and 3 fever remedies. Also shown in Table 1.9 is a third square, which lists as its i, j entry the i, j entries from both of the first two squares. We demand that each entry of this third square be different. This is not true in Table 1.9.

   (a) Find an example with 3 headache and 3 fever drugs where the combined square has the desired property.

   (b) Find another example with 4 headache and 4 fever drugs. (In Chapter 9 we observe that with 6 headache and 6 fever drugs, this is impossible. The existence problem has a negative solution.) *Note*: If you start with one Latin square design for the headache drugs and cannot find one for the fever drugs so that the combined square has the desired property, you should start with a different design for the headache drugs.

5. Show by enumeration that there are 14 bit strings of length at most 3.
6. Use enumeration to find the number of bit strings of length at most 4.
7. Suppose that we want to build a *trinary code* for the 26 letters of the alphabet, using strings in which each symbol is 0, 1, or -1.

Table 1.9: A Latin Square Design for Testing Headache Drugs 1, 2, and 3, a Latin Square Design for Testing Fever Drugs a, b, and c, and a Combination of the Two.[a]

Subject:	Day 1	Day 2	Day 3
1	1	2	3
2	2	3	1
3	3	1	2

Headache Drugs

Subject:	Day 1	Day 2	Day 3
1	a	b	c
2	b	c	a
3	c	a	b

Fever Drugs

Subject:	Day 1	Day 2	Day 3
1	$1, a$	$2, b$	$3, c$
2	$2, b$	$3, c$	$1, a$
3	$3, c$	$1, a$	$2, b$

Combination

[a] The third square has as its i, j entry the headache drug and the fever drug shown in the i, j entries of the first two squares, respectively.

Table 1.10: Overlap Data[a]

	English	Calculus	History	Physics
English	0	1	0	0
Calculus	1	0	1	1
History	0	1	0	1
Physics	0	1	1	0

[a] The i, j entry is 1 if the ith and jth courses have a common member, and 0 otherwise.

(a) Could we encode all 26 letters using strings of length at most 2? Answer this question by enumeration.

(b) What about using strings of length exactly 3?

8. The genetic code embodied in the DNA molecule, a code we describe in Section 2.1, consists of strings of symbols, each of which is one of the four letters T, C, A, or G. Find by enumeration the number of different codewords or strings using these letters and having length 3 or less.

9. Suppose that in designing a gas pipeline network, we have 2 possible pipe sizes, small (S) and large (L). If there are 4 possible links, enumerate all possible pipeline networks. (A typical one could be abbreviated $LSLL$, where the ith letter indicates the size of the ith pipe.)

10. In Example 1.3, suppose that a computer could analyze as many as 100 billion different pipeline networks in a second, a 100-fold improvement over the speed we assumed in the text. Would this make a significant difference in our conclusions? Why? (Do a computation in giving your answer.)

11. Tables 1.10 and 1.11 give data of overlap in class rosters for several courses in a university.

(a) Translate Table 1.10 into a graph as in Example 1.4.

Table 1.11: More Overlap Data[a]

	English	Calculus	History	Physics	Economics
English	0	1	0	0	0
Calculus	1	0	1	1	1
History	0	1	0	1	1
Physics	0	1	1	0	1
Economics	0	1	1	1	0

[a] The i, j entry is 1 if the ith and jth courses have a common member, and 0 otherwise.

Table 1.12: Acceptable Exam Times

Course	English	Calculus	History	Physics
Acceptable exam times	Thur. AM	Wed. AM Thur. AM	Tues. AM Wed. AM	Tues. AM Wed. AM

(b) Repeat part (a) for Table 1.11.

12. (a) Suppose that there are only two possible final examination times for the courses considered in Table 1.10. Is there an assignment of final exam times so that any two classes having a common member get a different exam time? If so, find such an assignment. If not, why not?

    (b) Repeat part (a) for Table 1.10 if there are three possible final exam times.

    (c) Repeat part (a) for Table 1.11 if there are three possible final exam times.

    (d) Repeat part (a) for Table 1.11 if there are four possible final exam times.

13. Suppose that in the situation of Table 1.5, the acceptable meeting times for Transportation are Tuesday and Thursday, for Education is Wednesday, and for all others are Tuesday, Wednesday, and Thursday. Show that no assignment of meeting times is possible.

14. (a) Suppose that in the situation of Table 1.10, the acceptable exam time schedules for each course are given in Table 1.12. Answer Exercise 12(b) if, in addition, each exam must be scheduled at an acceptable time.

    (b) Suppose that in the situation of Table 1.11, the acceptable exam time schedules for each course are given in Table 1.13. Answer Exercise 12(d) if, in addition, each exam must be scheduled at an acceptable time.

15. Suppose that there are three possible final exam times, Tuesday, Wednesday, and Thursday mornings. Suppose that each instructor of the courses listed in Table 1.10 requests Tuesday morning as a first choice for final exam time. What assignment (assignments) of exam times, if any exist, gives the largest number of instructors their first choice?

Table 1.13: More Acceptable Exam Times

Course	English	Calculus	History	Physics	Economics
Acceptable	Wed. AM	Tues. AM	Tues. AM	Tues. AM	Mon. AM
exam times		Wed. AM	Wed. AM	Thur. AM	Wed. AM

REFERENCES FOR CHAPTER 1

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

Alon, N., "Restricted Colorings of Graphs," in K. Walker (ed.), *Surveys in Combinatorics*, Proceedings 14th British Combinatorial Conference, London Math. Soc. Lecture Note Series, Vol. 187, Cambridge University Press, Cambridge, 1993, 1–33.

Baker, K. R., "Workforce Allocation in Cyclical Scheduling Problems," *Oper. Res. Quart., 27* (1976), 155–167.

Bartholdi, J. J., III, Orlin, J. B., and Ratliff, H. D., "Cyclic Scheduling via Integer Programs with Circular Ones," *Oper. Res., 28* (1980), 1074–1085.

Biggs, N. L., Lloyd, E. K., and Wilson, R. J., *Graph Theory 1736–1936*, Oxford University Press, London, 1976.

Biggs, N. L., Lloyd, E. K., and Wilson, R. J., "The History of Combinatorics," in R. L. Graham, M. Grötschel, and L. Lovász (eds.), *Handbook of Combinatorics*, Elsevier, Amsterdam, 1995, 2163–2198.

Bodin, L. D., and Friedman, A. J., "Scheduling of Committees for the New York State Assembly," Tech. Report USE No. 71–9, Urban Science and Engineering, State University of New York, Stony Brook, NY, 1971.

Brucker, P., *Scheduling Algorithms*, Springer-Verlag, Berlin, 1998.

Chrétienne, P., "On Graham's Bound for Cyclic Scheduling," *Parallel Comput., 26* (2000), 1163–1174.

Crama, Y., Kats, V., van de Klundert, J., and Levner, E., "Cyclic Scheduling in Robotic Flowshops," *Ann. Oper. Res., 96* (2000), 97–124.

David, F. N., *Games, Gods, and Gambling*, Hafner Press, New York, 1962. (Reprinted by Dover, New York, 1998.)

Frank, H., and Frisch, I. T., "Network Analysis," *Sci. Amer., 223* (1970), 94–103.

Herzberg, A. M., and Stanton, R. G., "The Relation Between Combinatorics and the Statistical Design of Experiments," *J. Combin. Inform. System Sci., 9* (1984), 217–232.

Karp, R. M., and Orlin, J. B., "Parametric Shortest Path Algorithms with an Application to Cyclic Staffing," *Discrete Appl. Math., 3* (1981), 37–45.

Kleitman, D. J., "Comments on the First Two Days' Sessions and a Brief Description of a Gas Pipeline Network Construction Problem," in F. S. Roberts (ed.), *Energy: Mathematics and Models*, SIAM, Philadelphia, 1976, 239–252.

Kovalëv, M. Ya., Shafranskij, Ya. M., Strusevich, V. A., Tanaev, V. S., and Tuzikov, A. V., "Approximation Scheduling Algorithms: A Survey," *Optimization, 20* (1989), 859–878.

Kratochvíl, J., Tuza, Z., and Voigt, M., "New Trends in the Theory of Graph Colorings: Choosability and List Coloring," in R. L. Graham, J. Kratochvíl, J.

Nešetřil, and F. S. Roberts (eds.), *Contemporary Trends in Discrete Mathematics*, DIMACS Series, Vol. 49, American Mathematical Society, Providence, RI, 1999, 183–197.

Liu, C. L., *Topics in Combinatorial Mathematics*, Mathematical Association of America, Washington, DC, 1972.

Orlin, J. B., "Minimizing the Number of Vehicles to Meet a Fixed Periodic Schedule: An Application of Periodic Posets," *Oper. Res., 30* (1982), 760–776.

Rothfarb, B., Frank, H., Rosenbaum, D. M., Steiglitz, K., and Kleitman, D. J., "Optimal Design of Offshore Natural-Gas Pipeline Systems," *Oper. Res., 18* (1970), 992–1020.

Tien, J. M., and Kamiyama, A., "On Manpower Scheduling Algorithms," SIAM *Rev., 24* (1982), 275–287.

Tucker, A. C., "Coloring a Family of Circular Arcs," SIAM *J. Appl. Math., 29* (1975), 493-502.

Zadeh, N., "Construction of Efficient Tree Networks: The Pipeline Problem," *Networks, 3* (1973), 1–32.