Chapter 12

Matching and Covering

12.1 SOME MATCHING PROBLEMS

In this chapter we study a variety of problems that fall into two general categories called matching problems and covering problems. We look at these problems in two ways, first as existence problems and then as optimization problems. Thus, this chapter will serve as a transition from our emphasis on the second basic problem of combinatorics, the existence problem, to the third basic problem, the optimization problem. We begin with a variety of examples.

Example 12.1 Job Assignments (Example 5.10 Revisited) In Example 5.10 we discussed a job assignment problem. In general, one can formulate this problem as follows. There are n workers and m jobs. Each worker is suited for some of the jobs. Assign each worker to one job, making sure that it is a job for which he or she is suited, and making sure that no two workers get the same job. In Example 5.10 we were concerned with counting the number of ways to make such an assignment. We used rook polynomials to do the counting. Here we ask an existence question: Is there *any* assignment that assigns each worker to one job to which he or she is suited, making sure that no job has two workers? Later, we ask an optimization question: What is the best assignment?

It will be convenient to formulate the existence question graph-theoretically. Build a graph G as follows. There are $m+n$ vertices in G, one for each worker and one for each job. Join each worker by an edge to each job for which he or she is suited. There are no other edges. Figure 12.1 shows the resulting graph G for one specific job assignment problem. Graph G is a *bipartite graph* (X, Y, E), a graph whose vertex set is divided into two sets X and Y, and with an edge set E so that all edges in E are between sets X and Y (see Section 3.3.4).

A job assignment of the kind we are seeking can be represented by replacing an edge from worker x to job y by a wiggly edge if x is assigned job y. Figure 12.2

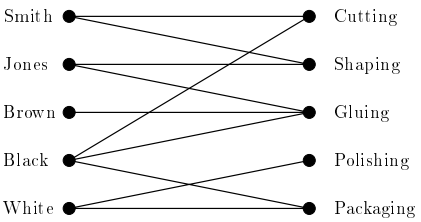

Figure 12.1: An edge from worker x to job y indicates that x is suitable for y.

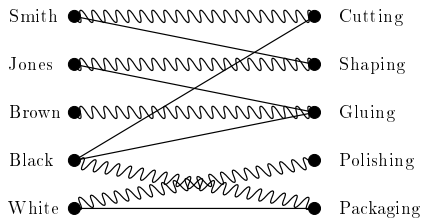

Figure 12.2: A job assignment for the graph of Figure 12.1 shown by wiggly edges.

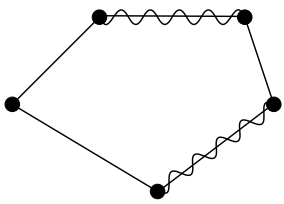

Figure 12.3: A matching M in a nonbipartite graph.

shows one such assignment. Note that in this assignment, each vertex is on at most one wiggly edge. This corresponds to each worker being assigned to at most one job and each job being assigned to at most one worker. A set M of edges in a graph G is called a *matching* if each vertex in G is on at most one edge of M. Thus, we seek a matching. In a matching M, a vertex is said to be *saturated* (M-*saturated*) if it is on some edge of M. We seek a matching that saturates every vertex corresponding to a worker. We first study matchings for bipartite graphs and then study them for arbitrary graphs. Figure 12.3 shows a matching in a nonbipartite graph. ■

Example 12.2 Storing Computer Programs (Example 5.10 Revisited) In Example 5.10 we also discussed a problem of assigning storage locations to computer programs. Formulating this in general terms, we can think of n programs and m storage locations making up the vertices of a graph. We put an edge between program x and location y if y has sufficient storage capacity for x. We seek an assignment of programs to storage locations such that each storage location gets at most one program and each program is assigned to exactly one location, a location that has sufficient storage capacity for the program. Thus, we seek a matching in the corresponding bipartite graph that saturates all vertices corresponding to programs. In contrast to the situation in Example 5.10, where we were interested in counting the number of such assignments, here we are interested in the question of whether or not there is such an assignment. ■

Example 12.3 Smallpox Vaccinations In preparing a plan for a possible outbreak of smallpox, a local health authority seeks to assign each person living in a city to a vaccination facility. They would like to find an assignment so that each

person gets assigned to a facility no more than 10 miles from his or her home. We can think of a bipartite graph with vertex set X being the residents and vertex set Y being the facilities. An edge joins a resident to any facility within 10 miles of his or her home. Note that this is *not* a matching problem, since we allow more than one person to be assigned to each facility. However, we can reformulate it as a matching problem if we realize that each facility has a maximum capacity for vaccinations and replace Y by a set consisting of available appointment times in all facilities. Then we include an edge from a resident to all the appointment times in all facilities within 10 miles of his or her home and we seek a matching that saturates every X vertex. ■

Example 12.4 Pilots for RAF Planes (Berge [1973], Minieka [1978]) During the Battle of Britain in World War II, the Royal Air Force (RAF) had many pilots from foreign countries. The RAF had to assign two pilots to each plane, and always wanted to assign two pilots to the same plane whose language and training were compatible. The RAF's problem can be translated as follows. Given a collection of pilots available for use on a mission, use these as the vertices of a graph, and join two vertices with an edge if and only if the two pilots can fly together. Then the RAF wanted a matching in this graph. (It is not necessarily bipartite.) Moreover, they were interested in flying as large a number of planes as possible. Thus, they were interested in a *maximum-cardinality matching*, a matching of the graph with the largest possible number of edges. In Sections 12.4 and 12.5 we will be interested in solving the combinatorial optimization problem that asks for a maximum-cardinality matching. ■

Example 12.5 Real Estate Transactions (Minieka [1978]) A real estate agent at a given time has a collection X of potential buyers and a collection Y of houses for sale. Let r_{xy} be the revenue to the agent if buyer x purchases house y. From a purely monetary standpoint, the agent wants to match up buyers to houses so as to maximize the sum of the corresponding r_{xy}. We can represent the problem of the agent by letting the vertices X and Y define a *complete bipartite graph* $G = (X, Y, E)$; that is, we take all possible edges between X and Y. We then seek a matching M in G that maximizes the sum of the weights. More generally, we can consider the following *maximum-weight matching problem*. Suppose that G is an arbitrary graph (not necessarily bipartite) with a weight (real number) r_{xy} on each edge $\{x, y\}$. If M is a matching of G, we define

$$r(M) = \sum \{r_{xy} : \{x, y\} \in M\}.$$

We seek a *maximum-weight matching* of G, a matching M such that for all the matchings M' of G, $r(M) \geq r(M')$. ■

Example 12.6 The Optimal Assignment Problem Let us return to the job assignment problem of Example 12.1, but add the simplifying assumption that every worker is suited for every job. Then the "is suited for" graph G is a complete

bipartite graph. Let us also assume that worker x is given a rating r_{xy} for his or her potential performance (output) on job y. We seek an assignment of workers to jobs that assigns every worker a job, no more than one per job, and maximizes the sum of the ratings. The problem of finding such an assignment is called the *optimal assignment problem*. If there are at least as many jobs as workers, the problem can be solved. It calls for a matching of the complete bipartite graph G that saturates the set of workers and has at least as large a sum of weights as any other matching that saturates the set of workers. But it is clear that every maximum-weight matching saturates the set of workers. Hence, the optimal assignment problem reduces to the maximum-weight matching problem for a complete bipartite graph. We return to the optimal assignment problem in Section 12.7, where we present an algorithm for solving it. (The algorithm can be understood from the material of Section 12.4, specifically Corollary 12.5.1.) ■

Example 12.7 Smallpox Vaccinations (Example 12.3 Revisited) In Example 12.3 we tried to assign people to vaccination facilities within 10 miles of their homes. Now, suppose that we wish to assign people to the closest possible vaccination facilities. We can let r_{xy} represent the distance from person x to the facility whose appointment time is represented by y. Then we seek an assignment that minimizes the sum of the r_{xy} over all x, y, where x is assigned to y. This is the minimum variant of the optimal assignment problem of Example 12.6. An alternative optimal assignment problem arises if people have preferences for different appointment times in different facilities. Let s_{xy} represent the rating by person x of appointment time y in its corresponding facility, independent of whether y is within 10 miles of x. Then we might seek an assignment of people to facilities so that the sum of the ratings is maximized, exactly as in the job assignment problem (Example 12.1). ■

Example 12.8 Pairing Speakers in Sound Systems (Ahuja, Magnanti, and Orlin [1993], Mason and Philpott [1988]) A manufacturer of sound systems seeks to pair up speakers before selling them as a set. How the two speakers will perform as a set depends on frequency response. The response f_{xy} between speakers x and y is measured (by comparing responses at a variety of frequencies), with low f_{xy} being better than high f_{xy}. The manufacturer might only be willing to pair up speakers if the measure f_{xy} is sufficiently small, e.g., less than some value T. Then, starting with a collection of speakers, the manufacturer might wish to create as many pairs of speakers as possible. This is a *maximum-cardinality matching problem*. The speakers are the vertices and an edge between x and y means that f_{xy} is less than T. A more sophisticated goal is to create pairs with $f_{xy} < T$ so that the sum of all of the f_{xy} over such pairs is as small as possible. This is the minimum-weight matching problem. In contrast to that of Example 12.5 (which has maximum instead of minimum), note that here the problem is on an arbitrary graph, not necessarily a bipartite graph. ■

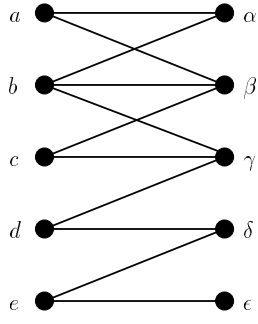

Figure 12.4: Graph for Exercise 1, Section 12.1.

Example 12.9 Oil Drilling (Ahuja, Magnanti, and Orlin [1993], Devine [1973]) A number of potential drilling sites have been identified by an oil company. It might be cheaper to drill one well for two sites rather than to drill separate wells. The company estimates the cost of drilling at each site, as well as the cost of drilling one well for each pair of sites. The problem is to identify which sites will involve a single well and which will be paired up so as to minimize total drilling costs. If we disregard the single-well possibility, we have a minimum-weight matching problem. If we allow an edge from a well to itself, we can also formulate the entire problem as a minimum-weight matching problem. Note that the graph in question is not bipartite. ■

There are many other applications of matching. Examples described by Ahuja, Magnanti, and Orlin [1993] include determining chemical bonds, rewiring special electrical typewriters in manufacturing, locating objects in space, matching moving objects, optimal depletion of inventory, and scheduling on parallel machines. We explore some of these applications in the exercises.

In Section 12.7 we shall see how the maximum-weight matching problem arises in connection with the "Chinese Postman" Problem of Section 11.4.1 and the related computer graph plotting problem of Section 11.4.2. For a discussion of the general maximum-weight matching problem as well as others described in the chapter, see, for example, Ahuja, Magnanti, and Orlin [1993], Christofides [1975], Cook, *et al.* [1998], Grötschel, Lovász, and Schrijver [1993], Lawler [1976], Lovász and Plummer [1986], Minieka [1978], or Papadimitriou and Steiglitz [1982].

EXERCISES FOR SECTION 12.1

1. In the graph of Figure 12.4:
   (a) Find a matching that has four edges.
   (b) Find a matching that saturates vertex b.
   (c) Find a matching that is not maximum but is maximal.
2. Repeat Exercise 1 for the graph of Figure 12.5.
3. In each weighted graph of Figure 12.6, find a maximum-weight matching.

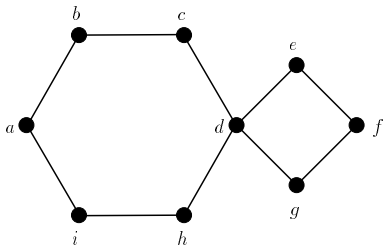

Figure 12.5: Graph for Exercise 2, Section 12.1.

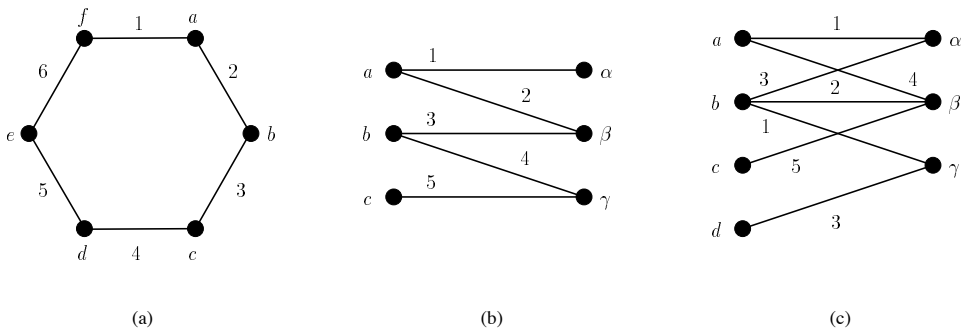

Figure 12.6: Weighted graphs for Exercise 3, Section 12.1.

4. A company has five positions to fill: computer technician (c), receptionist (r), accountant (a), word processor operator (w), and data analyst (d). There are seven applicants for jobs. The first applicant is qualified for the positions c, r, a; the second for the positions w, d; the third for the positions r, w, d; the fourth for the positions c, a; the fifth for the positions c, r, a, w; the sixth for the position c only; and the seventh for all the positions. Can all the positions be filled? Formulate this problem in the language of this section. What are we looking for? You are not asked to solve the problem at this point.

5. A company assigns its workers to temporary jobs. The i, j entry of a matrix gives the distance from worker i's home to the jth job. This matrix is given information. If every worker is suited for every job, find an assignment of workers to jobs that minimizes the sum of the distances the workers have to travel. Formulate this problem in the language of this section. Do not solve. What are we looking for?

6. There are six students looking for rooms in a dormitory: Ingram, Knight, Marks, Odell, Quincy, and Spencer. Ingram likes Knight, Marks, Odell, and Spencer; Knight likes Ingram, Odell, Quincy, and Spencer; Marks likes Ingram, Quincy, and Spencer; Odell likes Ingram, Knight, and Quincy; Quincy likes Knight, Marks, and Odell; and Spencer likes Ingram, Knight, and Marks. Note that a likes b iff b likes a. We wish to assign roommates, two to a room, such that each person only gets a roommate whom he likes. Can we assign each person to a room? If not, what is the largest number of people we can assign to rooms? Formulate this problem in the language of this section. Do not solve. What are we looking for?

7. (Ahuja, Magnanti, and Orlin [1993], Brogan [1989]) Two infrared sensors are used to identify objects in space. Each is used to provide a line of sight to the object, and

the two lines help us determine the location of the object. Suppose that we are given p lines from the first sensor and p lines from the second, but don't know which line corresponds to which object. A line from one sensor might intersect more than one line from the second sensor. Also, lines from the two sensors corresponding to the same object might not intersect due to measurement error. We wish to match up lines from the two sensors. Formulate this as a matching problem: Define the graph, define any appropriate measure r_{xy}, and comment on the type of matching problem involved: M-saturating, maximum-cardinality matching on a bipartite graph, and so on.

8. (Ahuja, Magnanti, and Orlin [1993], Brogan [1989], Kolitz [1991]) In missile defense and other applications, we wish to estimate speed and direction of objects in space. We can take "snapshots" of a variety of objects at two different times. If we can match up the first and second snapshots corresponding to an object, we can estimate its speed (from the time between snapshots) and direction of movement. Let r_{xy} denote the distance between snapshots. If we assume that snapshots are taken very quickly, we can try to match up objects with small r_{xy}. Formulate this problem as a matching problem, as in Exercise 7.

9. (Ahuja, Magnanti, and Orlin [1993], Derman and Klein [1959]) Items in our inventory (books, chemicals, etc.) may either gain or lose value over time. Suppose that at the beginning, we know the age of every item in our inventory. Suppose that we have a set of times when we must remove an item from inventory to use in some process, such as a manufacturing process. When should we remove a given item? Suppose that we can measure the value of an item of age a using a utility function $u(a)$. Then we can measure the utility u_{ij} of removing at time t_j an item of age a_i at the beginning of the process. Formulate the problem of finding a plan to remove items from inventory as a matching problem, as in Exercise 7.

10. Show that to describe a general solution to the maximum-weight matching problem for a weighted graph G, we may assume that:

    (a) G has an even number of vertices. (b) G is a complete graph.

11. Show that to describe a general solution to the maximum-weight matching problem for a bipartite graph G, we may assume that the graph is complete bipartite and both classes have the same number of vertices.

12. Show that the maximum-weight matching for a weighted bipartite graph does not necessarily saturate the first class of vertices, even if that class has no more than half the vertices.

13. Draw a bipartite graph $G = (X, Y, E)$ with $|X| = |Y| = 3$ such that G has a unique X-saturating matching and $|E|$ is as large as possible. Explain why a larger G does not exist.

14. (Ahuja, Magnanti, and Orlin [1993]) A ski rental shop wishes to assign n pairs of skis to n skiers. The skis come in lengths $l_1 \leq l_2 \leq \cdots \leq l_n$ and the skiers have heights $h_1 \leq h_2 \leq \cdots \leq h_n$. Ideally, a skier of height h will receive a ski of length αh, where α is a fixed constant. However, given the limited supply of skis, the shop can only try to minimize the sum of the (absolute) differences between αh_i and the length of the ski assigned to skier i. Show that if ski pairs and skiers are labeled as above, the assignment of the ith pair of skis to the ith skier is optimal.

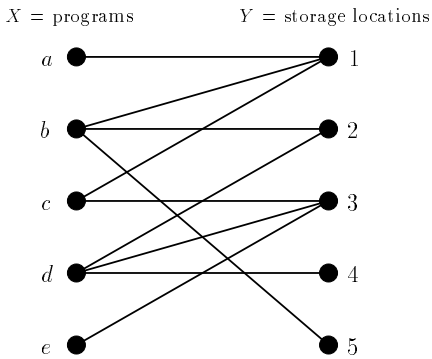

Figure 12.7: Bipartite graph for computer program storage. The vertices in X are the programs, the vertices in Y are the storage locations, and there is an edge between x in X and y in Y if and only if y has sufficient capacity to store x.

12.2 SOME EXISTENCE RESULTS: BIPARTITE MATCHING AND SYSTEMS OF DISTINCT REPRESENTATIVES

12.2.1 Bipartite Matching

In many of the examples of the preceding section, the graph constructed was a bipartite graph $G = (X, Y, E)$. Here we shall ask the question: Given a bipartite graph $G = (X, Y, E)$, under what conditions does there exist a matching that saturates X?

Consider the computer program storage assignment graph of Figure 12.7. Note that the three programs a, c, e have together only two possible locations they could be stored in, locations 1 and 3. Thus, there is no storage assignment that assigns each program to a location of sufficient storage capacity. There is no X-saturating matching. To generalize this example, it is clear that for there to exist an X-saturating matching, if S is any set of vertices in X and $N(S)$ is the *open neighborhood* of S, the set of all vertices y joined to some x in S by an edge, then $N(S)$ must have at least as many elements as S. In our example, S was $\{a, c, e\}$ and $N(S)$ was $\{1, 3\}$. What is startling is that this obvious necessary condition is sufficient as well.

Theorem 12.1 (Philip Hall's Theorem) (Hall [1935]) Let $G = (X, Y, E)$ be a bipartite graph. Then there exists an X-saturating matching if and only if for all subsets S of X, $|N(S)| \geq |S|$.

We prove Theorem 12.1 in Section 12.4.

To see whether the bipartite graph of Figure 12.1 has an X-saturating matching, where $X = \{\text{Smith, Jones, Brown, Black, White}\}$, we have to compute $N(S)$ for all subsets S of X. There are $2^{|S|} = 2^5 = 32$ such subsets. To make the computation, we note for instance that

$$N(\{\text{Smith, Jones, Black}\}) = \{\text{Cutting, Shaping, Gluing, Packaging}\},$$

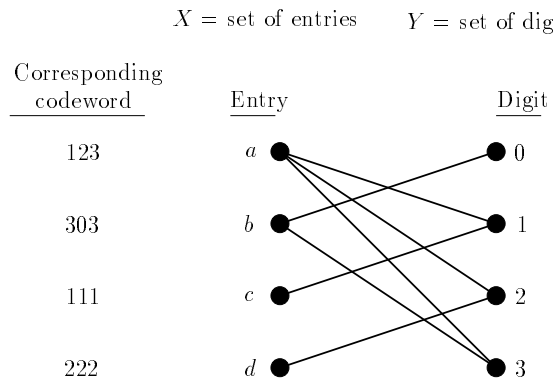

Figure 12.8: Bipartite graph for the coding problem with $n = 3$.

Table 12.1: Subsets S of X and Corresponding Neighborhoods $N(S)$, for Graph of Figure 12.8

S	$\emptyset$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{a,b\}$
$N(S)$	$\emptyset$	$\{1,2,3\}$	$\{0,3\}$	$\{1\}$	$\{2\}$	$\{0,1,2,3\}$
S	$\{a,c\}$	$\{a,d\}$	$\{b,c\}$	$\{b,d\}$	$\{c,d\}$	$\{a,b,c\}$
$N(S)$	$\{1,2,3\}$	$\{1,2,3\}$	$\{0,1,3\}$	$\{0,2,3\}$	$\{1,2\}$	$\{0,1,2,3\}$
S	$\{a,b,d\}$	$\{a,c,d\}$	$\{b,c,d\}$	$\{a,b,c,d\}$		
$N(S)$	$\{0,1,2,3\}$	$\{1,2,3\}$	$\{0,1,2,3\}$	$\{0,1,2,3\}$		

so $N(S)$ has four elements, whereas S has three. A similar computation for all 32 cases shows that there is an X-saturating matching. (In this case, finding one directly is faster.)

Example 12.10 A Coding Problem Each entry in a small file has a three-digit codeword associated with it, using the digits $0, 1, 2, \ldots, n$. Can we associate with each entry in the file just one of the digits of its codeword so that the entry can be recovered uniquely from this single digit? We can formulate this problem graph-theoretically as follows. The vertices of a bipartite graph $G = (X, Y, E)$ consist of X, the set of entries, and Y, the set of digits $0, 1, 2, \ldots, n$. There is an edge from entry x to digit y if y is used in the codeword for x. Figure 12.8 gives an example. We seek an X-saturating matching in the bipartite graph G. In our example, Table 12.1 lists all subsets S of X and the corresponding neighborhood $N(S)$. Note that in each case, $|N(S)| \geq |S|$. Thus, an X-saturating matching exists. This does not help us to find one. However, it is easy to see that the edges $\{a,3\}, \{b,0\}, \{c,1\}, \{d,2\}$ form such a matching. ■

A graph G is called *regular* if every vertex has the same degree, that is, the same number of neighbors.

Corollary 12.1.1 Suppose that $G = (X, Y, E)$ is a regular bipartite graph with at least one edge. Then G has an X-saturating matching.

Proof. Let S be a subset of X. Let E_1 be the collection of edges leading from vertices of S, and E_2 be the collection of edges leading from vertices of $N(S)$. Since every edge of E_1 leads from a vertex of S to a vertex of $N(S)$, we have $E_1 \subseteq E_2$. Thus, $|E_1| \leq |E_2|$. Moreover, since every vertex has the same number of neighbors, say k, and since $S \subseteq X$, it follows that $|E_1| = k|S|$ and $|E_2| = k|N(S)|$. Thus, $k|S| \leq k|N(S)|$. Since G has an edge, k is positive, so we conclude that $|S| \leq |N(S)|$. Q.E.D.

As an application of Corollary 12.1.1, suppose in Example 12.10 that every codeword uses exactly k of the digits $0, 1, 2, \ldots, n$ and every digit among $0, 1, 2, \ldots, n$ appears in exactly k codewords. Then there is an association of one digit with each file entry that allows a unique decoding.

Note that the X-saturating matching of a regular bipartite graph $G = (X, Y, E)$ is also Y-saturating, for every edge of G touches both a vertex of X and a vertex of Y. Thus, if k is the number of neighbors of each vertex, $k|X| = |E|$ and $k|Y| = |E|$. Thus, $k|X| = k|Y|$ and $|X| = |Y|$. Thus, every X-saturating matching must also be Y-saturating. A matching that saturates every vertex of a graph is called *perfect*. Perfect matchings in job assignments mean that every worker gets a job and every job gets a worker. The question of when in general there exists a perfect matching is discussed further in Section 12.3 (see also Exercises 18, 21, 24, and 26).

12.2.2 Systems of Distinct Representatives

Suppose that $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ is a family of sets, not necessarily distinct. Let $T = (a_1, a_2, \ldots, a_p)$ be a p-tuple with $a_1 \in S_1, a_2 \in S_2, \ldots, a_p \in S_p$. Then T is called a *system of representatives* for $\mathcal{F}$. If, in addition, all the a_i are distinct, T is called a *system of distinct representatives* (SDR) for $\mathcal{F}$. For instance, suppose that

$$\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5\},$$

where

$$S_1 = \{a, b, c\}, S_2 = \{b, c, d\}, S_3 = \{c, d, e\}$$
$$S_4 = \{d, e\}, S_5 = \{e, a, b\}. \tag{12.1}$$

Then $T = (a, b, c, d, e)$ is an SDR for $\mathcal{F}$. Next, suppose that $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$, where

$$S_1 = \{a, b\}, S_2 = \{b, c\}, S_3 = \{a, b, c\},$$
$$S_4 = \{b, c, d\}, S_5 = \{a, c\}, S_6 = \{c, d\}. \tag{12.2}$$

Then $T = (a, b, c, d, a, d)$ is a system of representatives for $\mathcal{F}$. However, there is no SDR, as we show below.

Example 12.11 List Colorings (Example 3.22 Revisited) In Example 3.22 we introduced the notion of list coloring a graph G that has a list assignment with a list of acceptable colors, $L(x)$, for each vertex x. A list coloring is an ordinary graph coloring with the color assigned to vertex x chosen from $L(x)$. If G is a complete graph, a list coloring corresponds to an SDR for the lists $L(x)$. ■

Example 12.12 Hospital Internships In a given year, suppose that p medical school graduates have applied for internships in hospitals. For the ith medical school graduate, let S_i be the set of all hospitals that find i acceptable. Then a system of representatives for the family $\mathcal{F} = \{S_i : i = 1, 2, \ldots, p\}$ would assign each potential intern to a hospital that is willing to take him or her. An SDR would make sure that, in addition, no hospital gets more than one intern. In practice, SDRs could be used in the following way. Modify S_i to include i. Find an SDR. This assigns each i to a hospital or to himself or herself; the latter is interpreted to mean that on the first round, i is not assigned to a hospital. (Exercise 7 asks the reader to show that an SDR exists.) After the initial assignment based on an SDR, the hospitals would be asked to modify their list of acceptable applicants among those not yet assigned. (Hopefully, at least some are assigned in the first round.) Then a new SDR would be found; and so on. A similar procedure could be used to place applicants for admission to graduate school. More complicated procedures actually in use in the National Resident Matching Program since 1952 make use of hospitals' and also applicants' ratings or rankings of the alternatives. We return to this idea in Section 12.7. ■

The problem of finding an SDR can be formulated graph-theoretically as follows. Suppose that X is the collection of sets S_i in $\mathcal{F}$ and Y is the collection of points in $\cup S_i$. Let $G = (X, Y, E)$, be a bipartite graph, with an edge from x in X to y in Y iff y is in x. Then an SDR is simply an X-saturating matching in G.

For instance, consider the family of sets in (12.2). The corresponding bipartite graph is shown in Figure 12.9. Note that if $S = \{S_1, S_2, S_3, S_5\}$, then $N(S) = \{a, b, c\}$. Thus, $|S| > |N(S)|$, so by Philip Hall's Theorem, there is no X-saturating matching and hence no SDR. Using the language of SDRs, we may now reformulate Philip Hall's Theorem as follows.

Corollary 12.1.2 The family $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ possesses an SDR iff for all $k = 1, 2, \ldots, n$, any k S_i's together contain at least k elements of $\cup S_i$.

Example 12.13 Extending Latin Rectangles A *Latin rectangle* is an $r \times s$ array using the elements $1, 2, \ldots, n$, such that each row and column uses each element from $1, 2, \ldots, n$ at most once. A Latin rectangle is called *complete* if $n = s$. One way to build up an $n \times n$ Latin square is to try building it up one row at a time, that is, by creating a complete $r \times n$ Latin rectangle and then extending it. For instance, consider the 2×6 Latin rectangle of Figure 12.10. Can we extend this to a 6×6 Latin square? More particularly, can we add a third row to obtain a 3×6 Latin rectangle? One approach to this question is to use rook polynomials (see Exercise 15, Section 5.1). Another approach is to ask what numbers could be

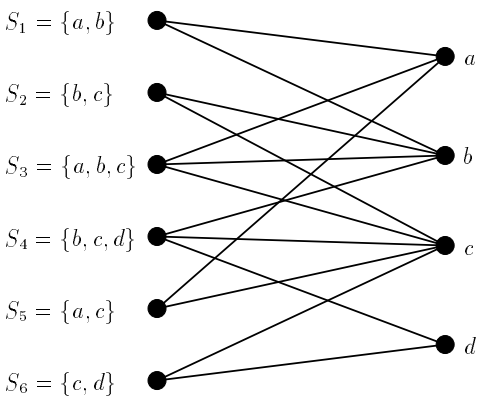

Figure 12.9: Bipartite graph representing the family of sets of (12.2).

1	2	3	4	5	6
4	3	6	5	1	2

Figure 12.10: A 2×6 Latin rectangle.

1	2	3	4	5	6
4	3	6	5	1	2
2	1	4	3	6	5

Figure 12.11: A 3×6 Latin rectangle obtained by extending the Latin rectangle of Figure 12.10.

placed in the new row in the ith column. Let S_i be the set of numbers not yet occurring in the ith column. In our example,

$$S_1 = \{2, 3, 5, 6\}, \quad S_2 = \{1, 4, 5, 6\}, \quad S_3 = \{1, 2, 4, 5\},$$
$$S_4 = \{1, 2, 3, 6\}, \quad S_5 = \{2, 3, 4, 6\}, \quad S_6 = \{1, 3, 4, 5\}.$$

We want to pick one element from each S_i and we want these elements to be distinct. Thus, we want an SDR. One SDR is (2, 1, 4, 3, 6, 5). Thus, we can use this as a new third row, obtaining the Latin rectangle of Figure 12.11. This idea generalizes. In general, given an $r \times n$ complete Latin rectangle, let S_i be the set of numbers not yet occurring in the ith column. To add an $(r+1)$st row, we need an SDR for the family of S_i. We shall show that we can always find one. ■

Theorem 12.2 If $r < n$, then every $r \times n$ complete Latin rectangle L can be extended to an $(r+1) \times n$ complete Latin rectangle.

Proof.[1] If S_i is defined as in Example 12.13, we shall show that the family of S_i possesses an SDR. Pick k of the sets in this family, $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$. By Corollary 12.1.2, it suffices to show that $A = S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k}$ has at least k elements. Each S_{i_j} has $n - r$ elements. Thus, A has $k(n-r)$ elements, including repetitions. Since the $r \times n$ Latin rectangle L is complete, each number in $1, 2, \ldots, n$ appears

[1] The proof may be omitted.

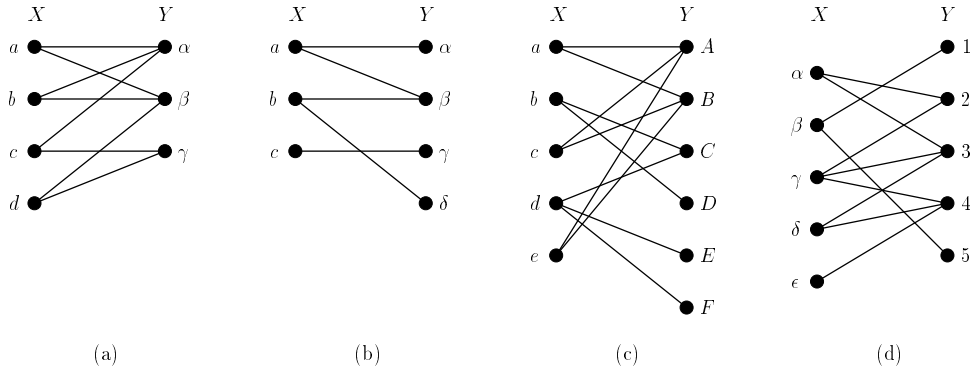

Figure 12.12: Bipartite graphs for exercises of Section 12.2.

exactly once in each row, so each number in $1, 2, \ldots, n$ appears exactly r times in L. Thus, each number in $1, 2, \ldots, n$ is in exactly $n - r$ of the sets $S_1, S_2, \ldots, S_n$. Thus, each number in A appears in at most $n - r$ of the sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$. Now if we make a list of elements of A, including repetitions, we have $k(n - r)$ elements. Each number in A appears in this list at most $n - r$ times. By the pigeonhole principle (Theorem 2.15), there must be at least k distinct numbers in A. Q.E.D.

EXERCISES FOR SECTION 12.2

1. Find $N(S)$ if G and S are as follows:
   (a) G = graph of Figure 12.1 and $S = \{$Smith, White$\}$
   (b) G = graph of Figure 12.8 and $S = \{a, c, d\}$
   (c) G = graph of Figure 12.9 and $S = \{S_1, S_3, S_4\}$
2. Find $N(S)$ if G and S are as follows:
   (a) G = graph of Figure 12.1 and $S = \{$Cutting, Shaping$\}$
   (b) G = graph of Figure 12.1 and $S = \{$Smith, Black, Polishing$\}$
   (c) G = graph of Figure 12.5 and $S = \{a, b, e\}$
3. For each bipartite graph $G = (X, Y, E)$ of Figure 12.12, determine if G has an X-saturating matching.
4. (a) Find a bipartite graph corresponding to the family of sets of (12.1).
   (b) Use Philip Hall's Theorem to show that there is an X-saturating matching in this graph.
5. For each of the following families of sets, find a system of representatives.
   (a) $S_1 = \{a, b, f\}$, $S_2 = \{a\}$, $S_3 = \{a, b, d, f\}$, $S_4 = \{a, b\}$, $S_5 = \{b, f\}$, $S_6 = \{d, e, f\}$, $S_7 = \{a, f\}$

(b) $S_1 = \{a, c\},\ S_2 = \{a, c\},\ S_3 = \{a, c\},\ S_4 = \{a, b, c, d, e\}$

(c) $S_1 = \{a_1, a_2, a_3\},\ S_2 = \{a_3, a_4\},\ S_3 = \{a_1, a_2, a_3\},\ S_4 = \{a_2, a_4\}$

(d) $S_1 = \{b_3, b_5\},\ S_2 = \{b_1, b_3, b_4\},\ S_3 = \{b_3, b_4, b_5\},\ S_4 = \{b_3, b_4, b_5\}$

(e) $S_1 = \{x, y\},\ S_2 = \{x\},\ S_3 = \{u, v, w\},\ S_4 = \{x, y, z\},\ S_5 = \{y, z\}$

(f) $S_1 = \{a, b, c\},\ S_2 = \{b, c\},\ S_3 = \{c, e, f\},\ S_4 = \{a, b\},\ S_5 = \{a, c\},\ S_6 = \{d, e, f\}$

6. For each of the families of sets in Exercise 5, determine if there is a system of distinct representatives and if so, find one.

7. In Example 12.12, show that if i is in S_i, an SDR exists.

8. Find the number of SDRs of each of the following families of sets.

   (a) $S_1 = \{1, 2\}, S_2 = \{2, 3\}, S_3 = \{3, 4\}, S_4 = \{4, 5\}, S_5 = \{5, 1\}, S_6 = \{6, 1\}$

   (b) $S_1 = \{1, 2\}, S_2 = \{2, 3\}, S_3 = \{3, 4\}, \ldots, S_n = \{n, 1\}$

   (c) $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}, S_4 = \{7, 8\}, S_5 = \{9, 10\}$

   (d) $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}, \ldots, S_n = \{2n - 1, 2n\}$

9. There are six committees of a state legislature: Finance, Environment, Health, Transportation, Education, and Housing. In Chapter 1 we discussed a meeting assignment problem for these committees. Here, we ask a different question. Suppose that there are 12 legislators who need to be assigned to committees, each to one committee. The following matrix has its i, j entry equal to 1 iff the ith legislator would like to serve on the jth committee.

	Finance	Environment	Health	Transportation	Education	Housing
Allen	1	1	1	0	0	0
Barnes	1	1	0	1	1	0
Cash	1	1	1	0	0	0
Dunn	1	0	0	1	1	1
Ecker	0	1	1	0	0	0
Frank	1	1	0	0	0	0
Graham	1	1	1	0	0	0
Hall	1	0	0	0	0	0
Inman	1	1	1	0	0	0
Johnson	1	1	0	0	0	0

   Suppose that we want to choose exactly one new member for each committee, choosing only a legislator who would like to serve. Can we do so? (Not every legislator needs to be assigned to a committee, and no legislator can be assigned to more than one committee.)

10. Consider the complete graph on the vertices $\{a, b, c, d, e, f\}$ with lists $L(a) = \{2, 3, 4\}$, $L(b) = \{3, 4\}$, $L(c) = \{1, 4, 6\}$, $L(d) = \{2, 3\}$, $L(e) = \{2, 4\}$, $L(f) = \{1, 5, 6\}$. Is there a list coloring?

11. Suppose that $G = (X, Y, E)$ is a complete bipartite graph with a list $L(a)$ assigned to each vertex a in $X \cup Y$. Give proof or counterexample: G has an L-list coloring iff the set of lists $L(a)$ has an SDR.

12. An *edge coloring* of graph G is an assignment of a color to each edge of G so that if two edges have a vertex in common, they get different colors.

(a) Show that an edge coloring with k colors decomposes a graph into k edge-disjoint matchings.

(b) Show that if a bipartite graph G is regular with each vertex of degree k, there is an edge coloring with k colors.

(c) Show that if G is a bipartite graph and $\Delta(G)$ is the maximum degree of a vertex in G, then G has an edge coloring using $\Delta(G)$ colors. (*Hint:* Make G regular by adding vertices and edges.)

13. If q is a positive integer and $S_1, S_2, \ldots, S_p$ are sets of integers, we say that an SDR $(a_1, a_2, \ldots, a_p)$ is a system of *q-distant representatives* if $|a_i - a_j| \geq q$ for all $i \neq j$. This concept, that arises in channel assignment, is due to Fiala, Kratochvíl, and Proskurowski [2001]. Suppose that G is a complete graph and that a list $L(a)$ of integers is assigned to each vertex a of G. Show that G has a T-coloring (Example 3.20) for a certain set T iff the lists $L(a)$ have a system of q-distant representatives.

14. Suppose that we are given a collection of codewords and we would like to store the collection on a computer as succinctly as possible, namely by picking one letter from each word to store.

(a) Can we do this for the following set of codewords in such a way that the codewords can be uniquely decoded? Words: $abcd, cde, a, b, ce$.

(b) If so, in how many ways can it be done?

15. Solve Exercise 4, Section 12.1.

16. Consider the following arrays. In each case, can we add one more column to the array, using the numbers $1, 2, 3, 4, 5, 6, 7$, so that the entries in this new column are all different and so that the entries in each row of the new array are all different?

(a)
123
265
371
436
542

(b)
67345
12734
73456
34567

17. Give proof or counterexample: Every regular bipartite graph with at least one edge has a perfect matching.

18. (a) What circuits Z_n have perfect matchings?

(b) What chains L_n of n vertices have perfect matchings?

(c) A wheel graph W_n is obtained from Z_n by adding a vertex adjacent to all vertices of Z_n. What wheel graphs W_n have perfect matchings?

(d) Which trees have perfect matchings?

19. There are n men and n women, and each man likes exactly p women and each woman likes exactly p men. Assuming that a likes b iff b likes a, show that the $2n$ people can be paired off into couples so that each man is paired with a woman he likes and who likes him.

20. Is the result of Exercise 19 still true if each man likes at least p women and each woman likes at most p men? Why?

21. (Tucker [1980]) There are n men and n women enrolled in a computer dating service. Suppose that the service has made nm pairings, so that each man dates m different women and each woman dates m different men, and suppose that $m < n$.

    (a) Suppose that we want to schedule all the dates over a period of m nights. Show that it is possible to do so. That is, show that the pairings can be divided into m perfect matchings.

    (b) Show that we can make the division in part (a) one step at a time; that is, show that no matter how the first k perfect matchings are chosen, we can always find a $(k+1)$st.

22. Let $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ be a family of sets that has an SDR.

    (a) If x is in at least one of the S_i, show that there is an SDR that chooses x as the representative of at least one of the S_i.

    (b) For each S_i that contains x, is there necessarily an SDR in which x is picked as the representative of S_i? Why?

23. Let $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$, where $S_i = \{1, 2, \ldots, n\} - \{i\}$.

    (a) Show that $\mathcal{F}$ has an SDR.

    (b) Show that the number of different SDRs is the number of derangements of n elements, D_n.

24. Is it possible for a tree to have more than one perfect matching? Why?

25. Consider a bipartite graph $G = (X, Y, E)$. Suppose that $m \geq 1$ is the maximum degree of a vertex of G and X_1 is the subset of X consisting of all vertices of degree m. Assume that X_1 is nonempty. Give a proof or a counterexample of the following assertion: There is a matching of G in which all vertices of X_1 are saturated.

26. (a) If H is a graph, let $o(H)$ count the number of components of H with an odd number of vertices. If S is a set of vertices in graph G, $G - S$ is the subgraph of G generated by vertices not in S. Show that if G has a perfect matching, $o(G - S) \leq |S|$ for all $S \subseteq V(G)$. (Tutte [1947] proves this result and also its converse.)

    (b) (Peterson [1891]) Generalizing the ideas of Section 11.2, let an edge $\{a, b\}$ in a graph G be called a *bridge* if removal of the edge, but not its end vertices a and b, increases the number of components. Suppose that G is a graph in which every vertex has degree 3 and suppose that G does not have any bridges. Assuming the converse in part (a), show that G has a perfect matching.

27. Suppose that $A_1, A_2, \ldots, A_m$ and $B_1, B_2, \ldots, B_m$ are two partitions of the same set S with the same number of A_i and B_j and with all $A_i \neq \emptyset$ and all $B_j \neq \emptyset$. (A *partition* of S is a division of all elements of S into disjoint subsets.) Let E be an m-element subset of S such that $A_i \cap E \neq \emptyset$ for all i and $B_j \cap E \neq \emptyset$ for all j. Then it is clear that $|A_i \cap E| = |B_j \cap E| = 1$ for all i, j. The set E is called a *system of common representatives* (SCR) for the partitions A_i and B_j.

    (a) Find an SCR for the following partitions of $S = \{1, 2, 3, 4, 5, 6\}$: $A_1 = \{1, 2, 3\}, A_2 = \{4\}, A_3 = \{5, 6\}$; $B_1 = \{1, 3\}, B_2 = \{2, 6\}, B_3 = \{4, 5\}$.

(b) Show that an SCR exists for the partitions A_i and B_j if and only if there is a suitable renumbering of the components of the partitions A_i and B_j such that for all i, $A_i \cap B_i \neq \emptyset$.

(c) Show that the partitions A_i and B_j have an SCR if and only if for all $k = 1, 2, \ldots, m$ and all $i_1, i_2, \ldots, i_k$ and $j_1, j_2, \ldots, j_{k-1}$ from $1, 2, \ldots, m$, the set $A_{i_1} \cup A_{i_2} \cup \cdots \cup A_{i_k}$ is not a subset of the set $B_{j_1} \cup B_{j_2} \cup \cdots \cup B_{j_{k-1}}$. (*Hint:* Form a bipartite graph and construct a matching.)

12.3 THE EXISTENCE OF PERFECT MATCHINGS FOR ARBITRARY GRAPHS[2]

Example 12.14 Pilots for RAF Planes (Example 12.4 Revisited) Let us consider whether or not, given a group of pilots, it is possible to assign two compatible pilots to each airplane, in such a way that every pilot is assigned to a plane. This is the case if and only if the graph constructed in Example 12.4 has a perfect matching, a matching in which every vertex is saturated. In this section we investigate one condition that guarantees the existence of a perfect matching. ■

Theorem 12.3 If a graph G has $2n$ vertices and each vertex has degree $\geq n$, the graph has a perfect matching.

We show why this theorem is true by trying to build up a matching M one step at a time. Here is an algorithm for doing this.

Algorithm 12.1: Finding a Perfect Matching

Input: A graph G with $2n$ vertices and each vertex having degree $\geq n$.
Output: A perfect matching of G.

Step 1. Set $M = \emptyset$.

Step 2. Find any pair of unsaturated vertices a and b that are joined by an edge of G, and place edge $\{a, b\}$ in M.

Step 3. If there is a pair of unsaturated vertices of G joined by an edge of G, return to Step 2. Otherwise, go to Step 4.

Step 4. If M has n edges, stop and output M. If M does not have n edges, go to Step 5.

Step 5. Find a pair of unsaturated vertices a and b in G. These will not be joined by an edge of G. (Why?) Find an edge $\{u, v\}$ in M such that $\{a, u\}$ and $\{b, v\}$ are edges of G. Remove edge $\{u, v\}$ from M and place edges $\{a, u\}$ and $\{b, v\}$ in M. Return to Step 4.

[2]This section may be omitted without loss of continuity.

Table 12.2: The Edges in M at Each Stage of Algorithm 12.1 Applied to the Graph of Figure 12.13

Stage	1	2	3	4
Edges in M	None	$\{1,6\}$	$\{1,6\},\{3,5\}$	$\{1,6\},\{2,3\},\{4,5\}$

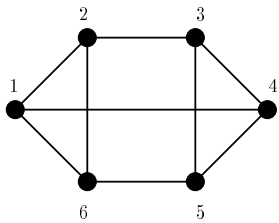

Figure 12.13: Graph used to illustrate Algorithm 12.1.

We illustrate this procedure with the graph of Figure 12.13. We show below why the procedure works. Note that in this graph, there are $2n = 2(3) = 6$ vertices, and each has degree $\geq n = 3$. Start in Step 2 by placing edge $\{1,6\}$ in M. Table 12.2 shows the successive edges of M. Go to Step 3. Next, since there is a pair of unsaturated vertices joined by an edge of G, go to Step 2. Find such a pair, say $3,5$, and place edge $\{3,5\}$ in M. Now there is no pair of unsaturated vertices joined by an edge of G, and since M has fewer than n edges, we proceed from Step 3 to Step 4 to Step 5. Pick vertices 2 and 4, that are unsaturated in G and not joined by an edge of G. Note that 3 and 5 are matched in M, and there are edges $\{2,3\}$ and $\{4,5\}$ in G. Thus, remove $\{3,5\}$ from M, add $\{2,3\}$ and $\{4,5\}$, and return to Step 4. Since we have $n = 3$ elements in M, we stop, having obtained a perfect matching.

To see why this procedure works, note that at each stage we have a matching. Moreover, if the algorithm stops, we always end up with n edges in M, and since there are $2n$ vertices in G, we must have a perfect matching. The crucial question is this: How do we know that Step 5 works? In particular, how do we know that there always exist u and v as called for in Step 5? To see why,[3] suppose that a and b are unsaturated in M and not joined by an edge of G. There are at present $r < n$ edges in M. If Step 5 could not be carried out, then for every edge $\{u,v\}$ in M, at most two of the edges $\{a,u\}$, $\{a,v\}$, $\{b,u\}$, $\{b,v\}$ are in G. The number of edges from a or b to matched edges $\{u,v\}$ in G is thus at most $2r$. Every edge from a or from b goes to some edge in M, for otherwise a or b is joined by an edge of G to an unsaturated vertex of G, and we would not have gotten to Step 5 in the algorithm. Thus, degree of a + degree of $b \leq 2r$. Since $r < n$, degree of a + degree of $b < 2n$. But by hypothesis, degree of a + degree of $b \geq n + n = 2n$, which is a contradiction. We conclude that Step 5 can always be carried out.

Let us examine the computational complexity of Algorithm 12.1. To get a crude upper bound, suppose that the ith iteration of the algorithm is the procedure

[3] This argument may be omitted.

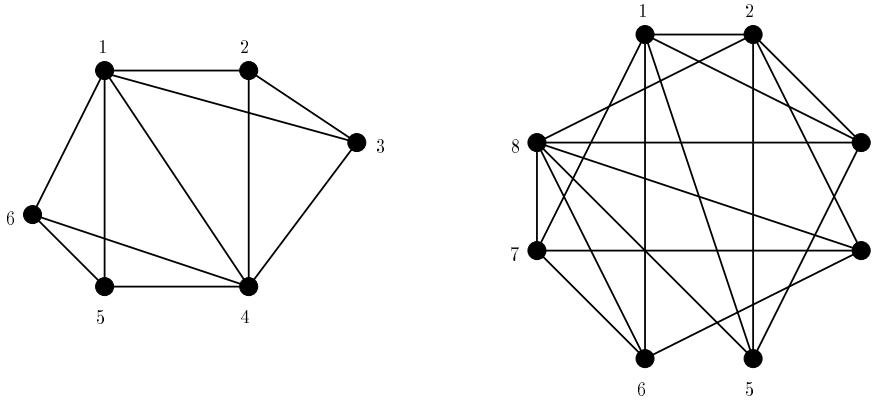

Figure 12.14: Graphs for exercises of Sections 12.3 and 12.4.

between the time the matching M has $i-1$ edges and the time it has i edges. Thus, there are n iterations in all. We can keep a record of the unsaturated vertices. Thus, it takes at most e steps to find an edge joining a pair of unsaturated vertices, where $e = |E(G)|$. If there is no such edge, we pick an arbitrary pair of unsaturated vertices a and b. For each edge $\{u, v\}$ in M, we ask if $\{a, u\}$ and $\{b, v\}$ are edges of G or if $\{a, v\}$ and $\{b, u\}$ are edges of G. This requires four questions for each of at most e edges, so at most $4e$ steps. Thus, we have at most $e + 4e = 5e$ steps in each iteration, and at most $n(5e)$ steps in all. Since $e \leq \binom{n}{2} \leq n^2$, we have at most $5n^3$ steps, a polynomial bound. In the terminology of Section 2.18, we have an $O(n^3)$ algorithm.

EXERCISES FOR SECTION 12.3

1. Suppose that K_{2p} is the complete graph of $2p$ vertices, $p \geq 2$, and G is obtained from K_{2p} by removing an edge. Does G have a perfect matching?

2. The *complete p-partite graph* $K(n_1, n_2, \ldots, n_p)$ has p classes of vertices, n_i vertices in the ith class, and all vertices in class i joined to all vertices in class j, for $i \neq j$, with no other vertices joined. Which of the following graphs have perfect matchings?

   (a) $K(2,2)$ (b) $K(3,3)$ (c) $K(1,5)$ (d) $K(2,3)$
   (e) $K(2,2,2)$ (f) $K(3,3,3)$ (g) $K(4,4,4)$ (h) $K(2,2,2,2)$

3. In Exercise 6, Section 12.1, can all students be assigned roommates?

4. For each graph of Figure 12.14, use Algorithm 12.1 to find a perfect matching.

5. Suppose that G has $2n$ vertices and for all $a \neq b$ such that $\{a, b\}$ is not an edge of G, degree of a + degree of $b \geq 2n$. Show that G has a perfect matching.

12.4 MAXIMUM MATCHINGS AND MINIMUM COVERINGS

12.4.1 Vertex Coverings

A *vertex covering* in a graph, or a *covering* (or *cover*) for short, is any set of vertices such that each edge of the graph has at least one of its end vertices in the set. For example, in the graph of Figure 12.13, the set of vertices $\{1,2,3,4,5\}$ forms a covering. So does the set of vertices $\{1,3,5,6\}$. In this section we study *minimum-cardinality coverings* (*minimum coverings* for short). These are coverings that use as few vertices as possible. We shall relate minimum coverings to *maximum-cardinality matchings* (*maximum matchings* for short). Such matchings arose in Example 12.4.

Example 12.15 Police Surveillance A police officer standing at a street intersection in a city can watch one block in any direction. If we attempt to locate a criminal, we want to be sure that all the blocks in a neighborhood are being watched simultaneously. Assuming that police officers stand only at street intersections, what is the smallest number of police officers we need, and where do we put them? The answer is obtained by letting the street intersections in the neighborhood be vertices of a graph G and joining two vertices by an edge if and only if they are joined by a one-block street. Then we seek a minimum covering of the graph G. ■

Example 12.16 Smallpox Vaccinations (Example 12.3 Revisited) In Example 12.3 we investigated assignments of people to vaccination facilities not more than 10 miles from their home. There, we assumed that vaccination facilities were already established. But what if we want to locate as small a number of facilities as we can so that every person is within 10 miles of one of the facilities. We could let each square block of the city be a vertex and join two vertices with an edge if they are within 10 miles of each other. Assuming that each square block is inhabited, we seek a minimum-size set of vertices (square blocks) so that every edge has one of its end vertices in that set. Such a set is a minimum covering of the graph. ■

Next we investigate the relation between matchings and coverings. Let G be a graph, let M be a matching of G, and let K be a covering. Then $|M| \leq |K|$, for given any edge $\alpha = \{x, y\}$ of M, either x or y is in K. Let $f(\alpha)$ be whichever of x and y is in K, picking either if both are in K. Note that since M is a matching, if α and β are two edges of $M, f(\alpha)$ must be different from $f(\beta)$. Why? Thus, we assign each edge of M to a different element of K. Hence, M must have no more elements than K.

It follows that if M^* is a maximum matching and K^* is a minimum covering, $|M^*| \leq |K^*|$. Note that $|M^*|$ can be strictly less than $|K^*|$. In the graph of Figure 12.13, a minimum covering K^* consists of vertices $1,3,5,6$, so $|K^*| = 4$. However, a maximum matching consists of three edges. For instance, $M^* = \{\{1,2\},\{3,4\},\{5,6\}\}$ is a maximum matching here. Hence, $|M^*| = 3 < |K^*|$. A crucial result is the following.

Theorem 12.4 If M is a matching and K is a covering and $|M| = |K|$, K is a minimum covering and M is a maximum matching.

Proof. If M^* is any maximum matching and K^* is any minimum covering, we have

$$|M| \leq |M^*| \leq |K^*| \leq |K|. \tag{12.3}$$

Since $|M| = |K|$, all terms in (12.3) must be equal, and in particular $|M| = |M^*|$ and $|K| = |K^*|$ hold. Q.E.D.

For example, consider the graph of Figure 12.9. Here there is a matching M consisting of the four edges $\{S_1, a\}, \{S_3, b\}, \{S_4, c\}$, and $\{S_6, d\}$, and a covering K consisting of the four vertices a, b, c, d. Thus, by Theorem 12.4, M is a maximum matching and K is a minimum covering. This example illustrates the following result.

Theorem 12.5 (König [1931]) Suppose that $G = (X, Y, E)$ is a bipartite graph. Then the number of edges in a maximum matching equals the number of vertices in a minimum covering.

This theorem is proved in Section 13.3.8.

As a corollary of this theorem, we can now derive Philip Hall's Theorem, Theorem 12.1.

Proof of Philip Hall's Theorem (Theorem 12.1).[4] It remains to show that if $|N(S)| \geq |S|$ for all $S \subseteq X$, there is an X-saturating matching. By Theorem 12.5, since G is bipartite, an X-saturating matching exists if and only if each cover K has $|K| \geq |X|$. Let K be a cover and let $X - S$ be the set of X-vertices in K. Then vertices in S are not in K, so all vertices of $N(S)$ must be in K. It follows that

$$|K| \geq |X - S| + |N(S)| = |X| - |S| + |N(S)| \geq |X|.$$

Hence, $|K| \geq |X|$ for all K, so an X-saturating matching exists. Q.E.D.

Another corollary of Theorem 12.5 will be important in Section 12.7 in an algorithm for the optimal assignment problem (Example 12.6). To state this corollary, let $\mathbf{A}$ be a matrix of 0's and 1's. A *line* of this matrix is either a row or a column. A set of 0's in this matrix is called *independent* if no two 0's lie in the same line. For instance, suppose that

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}. \tag{12.4}$$

Then the $(2,1)$ and $(1,3)$ entries form an independent set of 0's of $\mathbf{A}$, as do the $(2,1), (4,2)$, and $(3,3)$ entries.

[4] This proof may be omitted.

Corollary 12.5.1 (König-Egerváry Theorem[5]) If **A** is a matrix of 0's and 1's, a maximum independent set of 0's has the same number of elements as a minimum set of lines covering all the 0's of **A**.

Proof. Build a bipartite graph $G = (X, Y, E)$ by letting X be the rows of **A**, Y the columns of **A**, and joining rows i and j by an edge if and only if the i, j entry of **A** is 0. Then a maximum independent set of 0's of **A** corresponds to a maximum matching of G, and a minimum set of lines covering all the 0's of **A** corresponds to a minimum covering of G. Q.E.D.

To illustrate this result, for **A** of (12.4), a maximum independent set of 0's is the set of three 0's at the $(2,1), (4,2)$, and $(3,3)$ entries. A minimum set of lines covering all the 0's of **A** consists of the second row and the second and third columns.

12.4.2 Edge Coverings

A collection F of edges in a graph is called an *edge covering* if every vertex of the graph is on one of the edges in F. An edge covering is called *minimum* if no other edge covering has fewer edges. Minimum edge coverings have applications to switching functions in computer engineering, to crystal physics, and to other fields (see Deo [1974, pp. 184ff.]). We investigate edge coverings and their relations to matchings and vertex coverings in the exercises.

EXERCISES FOR SECTION 12.4

1. In graph (a) of Figure 12.14, find:
   (a) A covering of five vertices (b) A minimum covering
2. In the graph of Figure 12.1, find:
   (a) A covering of six vertices (b) A minimum covering
3. In an attempt to improve the security of its computer operations, a company has put in 20 special passwords. Each password is known by exactly two people in the company. Find the smallest set of people who together know all the passwords. Formulate this problem using the terminology of this section.
4. Suppose that **A** is a matrix of 0's and 1's. Suppose that we find k lines covering all 0's of **A** and we find k independent 0's. What conclusion can we draw?
5. Is the König-Egerváry Theorem still true if the matrix is allowed to have any integers as entries? Why?
6. In each of the following graphs, find an edge covering that is not minimum and an edge covering that is minimum.
   (a) The graph of Figure 12.13 (b) Graph (a) of Figure 12.14
   (c) Graph (b) of Figure 12.14 (d) Graph (b) of Figure 12.12

[5] This theorem is based on work of König [1931] and Egerváry [1931].

7. (a) Can a graph with isolated vertices have an edge covering?
   (b) What is the smallest conceivable number of edges in an edge covering of a graph of n vertices?
8. A patrol car is parked in the middle of a block and can observe the two street intersections at the end of the block. How would you go about finding the smallest number of patrol cars required to keep all street corners in a neighborhood under surveillance? (See Gondran and Minoux [1984].)
9. (Minieka [1978]) A committee is to be chosen with at least one member from each of the 50 states and at least one member from each of the 65 major ethnic groups in the United States. What is the smallest committee that can be constructed from a group of volunteers if the requirements are to be met? To answer this question, let the vertices of a graph be the 50 states and the 65 ethnic groups, and let a volunteer correspond to an edge joining his or her state and ethnic group. What are we looking for to answer the question?
10. Illustrate Theorem 12.5 on the grid graphs G_{n_1,n_2} defined in Section 11.2.4 by calculating maximum matchings and minimum coverings in:
    (a) $G_{1,2}$ (b) $G_{1,m}$ (c) $G_{2,2}$ (d) $G_{n,n}$
11. (a) Show that Theorem 12.5 fails for at least some annular grids $AN(c,s)$ defined in Section 11.2.5.
    (b) Does it hold for any annular grids?
12. (a) Can a minimum edge covering of a graph contain a circuit? Why?
    (b) Show that every minimum edge covering of a graph of n vertices has at most $n-1$ edges.
13. (Gallai [1959]) Suppose that G is a graph without isolated vertices, K^* is a minimum (vertex) covering of G, and I^* is a maximum independent set of vertices. Show that $|K^*| + |I^*| = |V|$.
14. (Norman and Rabin [1959], Gallai [1959]) Suppose that G is a graph without isolated vertices, M^* is a maximum matching, and F^* is a minimum edge covering. Show that $|M^*| + |F^*| = |V|$.
15. Suppose that I is an independent set of vertices in a graph G without isolated vertices and F is an edge covering of G. Show that $|I| \leq |F|$.
16. Suppose that we find an independent set I of vertices in a graph G without isolated vertices and an edge covering F in G such that $|I| = |F|$. What conclusion can we draw? Why?
17. Suppose that $G = (X, Y, E)$ is a bipartite graph without isolated vertices. Let I^* be an independent set with a maximum number of vertices (a *maximum independent set*) and F^* be a minimum edge covering of G. Show from Exercises 13 and 14 that $|I^*| = |F^*|$.
18. (Minieka [1978]) Consider the following algorithms on a graph with no isolated vertices. *Algorithm 1* starts with a matching M and selects any unsaturated vertex x. It adds to M any edge incident to x and repeats the procedure until every vertex is saturated. The resulting set of edges is called C'. *Algorithm 2* starts with an edge covering C and selects any vertex x covered by more than one edge in C. It removes from C an edge that covers x and repeats the procedure until no vertex is covered by more than one edge. The resulting set of edges is called M'.

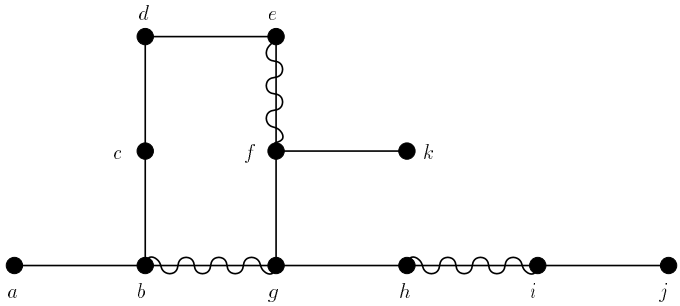

Figure 12.15: A matching M is shown by the wiggly edges.

(a) Show that C' as produced by Algorithm 1 is an edge covering.

(b) Show that M' as produced by Algorithm 2 is a matching.

(c) Show that if M is a maximum matching, then C' is a minimum edge covering.

(d) Show that if C is a minimum edge covering, then M' is a maximum matching.

19. Derive Philip Hall's Theorem (Theorem 12.1) from Theorem 12.4.

12.5 FINDING A MAXIMUM MATCHING

12.5.1 M-Augmenting Chains

In this section we present a procedure for finding a maximum matching in a graph G. Suppose that M is some matching of G. An *M-alternating chain* in G is a simple chain

$$u_1, e_1, u_2, e_2, \ldots, u_t, e_t, u_{t+1} \tag{12.5}$$

such that e_1 is not in M, e_2 is in M, e_3 is not in M, e_4 is in M, and so on. For instance, consider the matching shown in Figure 12.15 by wiggly edges. Then the chain $a, \{a,b\}, b, \{b,g\}, g, \{g,f\}, f, \{f,e\}, e$ is M-alternating. If an M-alternating chain (12.5) joins two vertices u_1 and u_{t+1} that are not M-saturated (i.e., not on any edge of M), we call the chain an *M-augmenting chain*. Our example is not M-augmenting, since e is M-saturated. However, the chain

$$a, \{a,b\}, b, \{b,g\}, g, \{g,f\}, f, \{f,e\}, e, \{e,d\}, d \tag{12.6}$$

is M-augmenting. Let us now find a new matching M' by deleting from M all edges of M used in the M-augmenting chain (12.6) and adding all edges of (12.6) not in M. Then M' is shown in Figure 12.16. It is indeed a matching. Moreover, M' has one more edge than M. This procedure always gives us a larger matching.

Theorem 12.6 Suppose that M is a matching and C is an M-augmenting chain of M. Let M' be M less edges of C in M plus edges of C not in M. Then M' is a matching and $|M'| > |M|$.

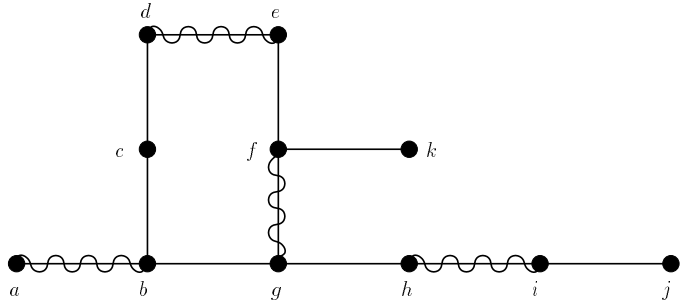

Figure 12.16: The matching M' obtained from the matching M of Figure 12.15 by using the M-augmenting chain (12.6).

Proof. To see that $|M'| > |M|$, note that in an M-augmenting chain (12.5), e_1 and e_t cannot be in M, so t is odd. Hence, we add the edges $e_1, e_3, \ldots, e_t$ and delete the edges $e_2, e_4, \ldots, e_{t-1}$, and we add one more edge than we take away. To see that M' is a matching, note that if j is odd, edge e_j is added and edge e_{j+1} is subtracted. Thus, if j is odd and $j \neq 1, t+1$, u_j was previously only on e_{j-1} and is now only on e_j. If j is even, u_j was previously only on e_j and is now only on e_{j-1}. Also, u_1 was previously unmatched and is now only on e_1, and u_{t+1} was previously unmatched and is now only on e_t. Q.E.D.

As a corollary of Theorem 12.6, we observe that if M has an M-augmenting chain, M could not be a maximum matching. In fact, the converse is also true.

Theorem 12.7 (Berge [1957], Norman and Rabin [1959]) A matching M of G is maximum if and only if G contains no M-augmenting chain.

To apply this theorem, note that the matching M' in Figure 12.16 is not maximum, since there is an M'-augmenting chain

$$j, \{j,i\}, i, \{i,h\}, h, \{h,g\}, g, \{g,f\}, f, \{f,k\}, k. \tag{12.7}$$

If we modify M' by deleting edges $\{i,h\}$ and $\{g,f\}$, the M' edges of the chain (12.7), and adding edges $\{j,i\}, \{h,g\}$, and $\{f,k\}$, the non-M' edges of (12.7), we obtain the matching M'' shown in Figure 12.17. There is no M''-augmenting chain, since there is only one unsaturated vertex. Thus, M'' is maximum.

12.5.2 Proof of Theorem 12.7[6]

To prove Theorem 12.7, it remains to show that if there is a matching M' such that $|M'| > |M|$, there is an M-augmenting chain. Let H be the subgraph of G consisting of all vertices of G and all edges of G that are in M or in M' but not in both M and M'. Note that in H, there are more M' edges than M edges. Moreover, each

[6] This subsection may be omitted.

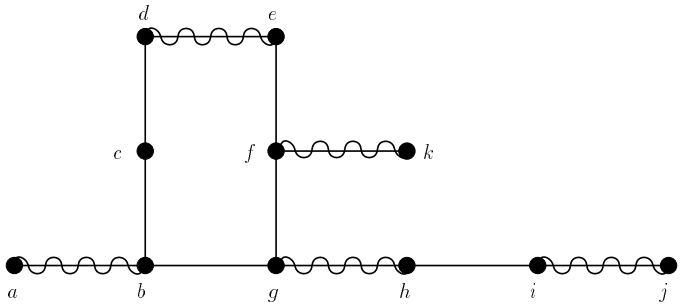

Figure 12.17: The matching M'' obtained from the matching M' of Figure 12.16 by using the M'-augmenting chain (12.7).

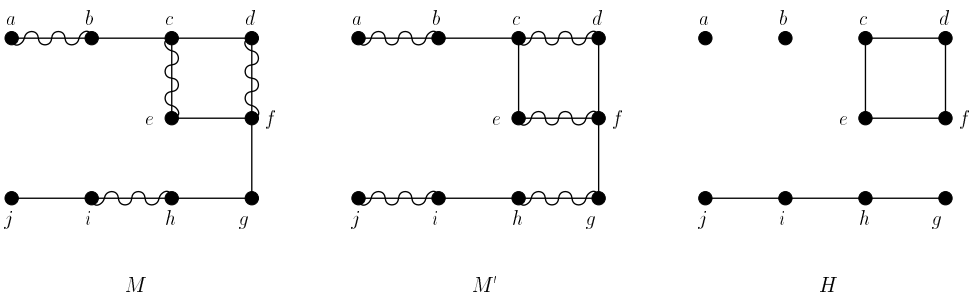

Figure 12.18: The graph H is obtained from the matchings M and M'.

vertex of H has at most two neighbors in H, for it can have at most one neighbor in M and at most one neighbor in M'. Using the latter observation, one can show (see Exercise 10) that each connected component of H is either a circuit Z_n or a simple chain L_n of n vertices. Moreover, the edges in Z_n or L_n must alternate between M and M' because M and M' are matchings. It follows that each Z_n has an equal number of edges from M and from M'. Thus, since H has more edges of M' than of M, some component of the form L_n has this property. This L_n must be a chain of the form (12.5) with $e_1, e_3, \ldots, e_t$ in M' and not in M, and $e_2, e_4, \ldots, e_{t-1}$ in M and not in M'. Moreover, u_1 and u_{t+1} cannot be M-saturated, for otherwise L_n would not be a component of H. Hence, the chain L_n is an M-augmenting chain. This completes the proof.

To illustrate this proof, consider the matchings M and M' of Figure 12.18. Then the graph H is also shown in that figure. The chain

$$g, \{g,h\}, h, \{h,i\}, i, \{i,j\}, j$$

is a simple chain of two M' edges and one M edge.

12.5.3 An Algorithm for Finding a Maximum Matching

We next describe an algorithm for finding a maximum matching. This algorithm originates in the work of L. R. Ford and D. R. Fulkerson (see Ford and Fulkerson [1962]). The algorithm has two basic subroutines. Subroutine 1 searches for an M-augmenting chain starting with an unsaturated vertex x, and subroutine 2 builds a larger matching M' if subroutine 1 finds an M-augmenting chain. The algorithm chooses an unsaturated vertex x and applies subroutine 1. If an M-augmenting chain starting with x is found, subroutine 2 is called. If no such chain is found, another unsaturated vertex y is used in subroutine 1. The procedure is repeated until either an M-augmenting chain is found or no unsaturated vertices remain. In the latter case, we conclude that there is no M-augmenting chain, so we have a maximum matching.

Subroutine 2 works exactly as described in Theorem 12.6. We now present subroutine 1. It is easiest to describe this subroutine for the case of a bipartite graph G. If G is not bipartite, the procedure is more complicated. It was Edmonds [1965a] who first observed that something much more subtle was needed here (see Exercise 16). For more details on bipartite and nonbipartite matching, see, for example, Ahuja, Magnanti, and Orlin [1993], Cook, *et al.* [1998], Lawler [1976], Lovász and Plummer [1986], Minieka [1978], or Papadimitriou and Steiglitz [1982].

Subroutine 1 begins with a matching M and a vertex x unsaturated in M and builds in stages a tree T called an *alternating tree*. The idea is that x is in T and all simple chains in T beginning at x are M-alternating chains. For instance, in Figure 12.15, if $x = a$, one such alternating tree consists of the edges $\{a, b\}, \{b, g\}, \{g, h\}$, and $\{g, f\}$. The vertices in the alternating tree T are called outer or inner. Vertex y is called *outer* if the unique simple chain[7] between x and y ends in an edge of M, and *inner* otherwise. The vertex x is called *outer*. If an alternating tree T with an unsaturated vertex $y \neq x$ is found, the unique simple chain between x and y is an M-augmenting chain. Vertices and edges are added to T until either such a chain is found or no more vertices can be added to T. In the latter case, there is no M-augmenting chain. We now present the subroutine in detail.

Algorithm 12.2: Subroutine 1: Searching for an M-Augmenting Chain Beginning with Vertex x

Input: A bipartite graph G, a matching M of G, and a vertex x unsaturated in M.

Output: An M-augmenting chain beginning at x or the message that no such chain exists.

Step 1. Set $T = \emptyset$ and $T' = \emptyset$. (T is the set of edges in the tree and T' the set of edges definitely not in the tree.) Set $O = \{x\}$ and $I = \emptyset$. (O is the set of outer vertices and I is the set of inner vertices.)

[7] We are using the result of Theorem 3.18—that in a tree, there is a unique simple chain between any pair of vertices.

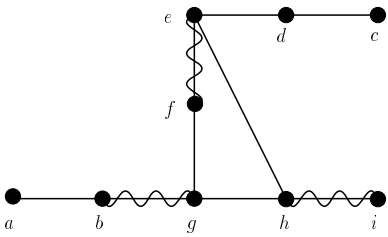

Figure 12.19: A matching M is shown by wiggly edges.

Step 2.

> ***Step 2.1.*** Among the edges not in T or T', if there is no edge between an outer vertex (a vertex of O) and any other vertex, go to Step 3. Otherwise, let $\{u, v\}$ be such an edge with $u \in O$.
>
> ***Step 2.2.*** If vertex v is an inner vertex (is in I), place edge $\{u, v\}$ in T' and repeat Step 2.1. If vertex v is neither inner nor outer, place edge $\{u, v\}$ in T, and go to Step 2.3. (Since G is bipartite, v cannot be an outer vertex, for otherwise one can show that G has an odd circuit, which is impossible for bipartite graphs by Theorem 3.4; see Exercise 11.)
>
> ***Step 2.3.*** If v is unsaturated, stop. The unique chain in T from x to v forms an M-augmenting chain from x to v. If v is saturated, there is a unique edge $\{v, w\}$ in M. Then place $\{v, w\}$ in T, v in I, and w in O. Return to Step 2.1.

Step 3. We get to this step only when no further assignment of edges to T or T' is possible. Stop and give the message that there is no M-augmenting chain beginning with x.

Note that the algorithm stops in two ways, having found an M-augmenting chain or having found an alternating tree T where it is impossible to add edges to either T or T'. In the former case, the procedure goes to subroutine 2. In the latter case, one can show that there is no M-augmenting chain starting from the vertex x. We repeat this subroutine for another unsaturated vertex x. If in repeated applications of subroutine 1, we fail to find an M-augmenting chain beginning from an unsaturated vertex x, we conclude that the matching is maximum.

We illustrate the algorithm on the matching M of Figure 12.19. Pick unsaturated vertex x to be a, and call a an outer vertex. This is Step 1. Go to Step 2.1 and select the edge $\{a, b\}$ that is neither in T nor in T' and joins an outer vertex. Since b is not inner and not outer, in Step 2.2 we place this edge in T and go to Step 2.3. Since b is saturated, we consider the unique edge $\{b, g\}$ of M. We place this edge in T, call b inner and g outer, and return to Step 2.1. (See Table 12.3 for a summary. For the purposes of this summary, an iteration is considered a return to Step 2.1.)

In Step 2.1 we consider edges not in T and not in T' and joining an outer vertex. There are two such edges, $\{g, h\}$ and $\{g, f\}$. Let us suppose that we choose edge $\{g, f\}$. Since f is neither inner nor outer, in Step 2.2 we place $\{g, f\}$ in T. Now f

Table 12.3: Steps in Algorithm 12.2 Applied to M of Figure 12.19 and Starting with Vertex $x = a$

Iteration	T	T'	I (inner vertices)	O (outer vertices)
1	$\emptyset$	$\emptyset$	$\emptyset$	a
2	$\{a,b\}$	$\emptyset$	$\emptyset$	a
	$\{a,b\},\{b,g\}$	$\emptyset$	b	a,g
3	$\{a,b\},\{b,g\},\{g,f\}$	$\emptyset$	b	a,g
	$\{a,b\},\{b,g\},\{g,f\},\{f,e\}$	$\emptyset$	b,f	a,g,e
4	$\{a,b\},\{b,g\},\{g,f\},\{f,e\},\{g,h\}$	$\emptyset$	b,f	a,g,e
	$\{a,b\},\{b,g\},\{g,f\},\{f,e\},\{g,h\},\{h,i\}$	$\emptyset$	b,f,h	a,g,e,i
5	$\{a,b\},\{b,g\},\{g,f\},\{f,e\},\{g,h\},\{h,i\}$	$\{e.h\}$	b,f,h	a,g,e,i
6	$\{a,b\},\{b,g\},\{g,f\},\{f,e\},\{g,h\},\{h,i\},\{e,d\}$	$\{e,h\}$	b,f,h	a,g,e,i

is saturated, so we consider the unique edge $\{f,e\}$ of M. We place this edge in T, call f inner and e outer, and return to Step 2.1.

Next, we again consider edges not in T and not in T' and joining an outer vertex. The possible edges are $\{g,h\}, \{e,h\}$, and $\{e,d\}$. Suppose that we choose $\{g,h\}$. Then in Step 2.2, since h is neither inner nor outer, we place edge $\{g,h\}$ in T. Then in Step 2.3, since h is saturated, we place edge $\{h,i\}$ in T and call h inner and i outer.

We again consider edges not in T and not in T' and joining an outer vertex. The possible edges are $\{e,h\}$ and $\{e,d\}$. Suppose that we pick the former. Then vertex h is inner, so in Step 2.2 we place edge $\{e,h\}$ in T', and repeat Step 2.1.

In Step 2.1 we now choose edge $\{e,d\}$ and in Step 2.2 we add edge $\{e,d\}$ to T. Then in Step 2.3, since d is unsaturated, we stop and find the unique chain in T from x to v, that is, from a to d. This is the chain a, $\{a,b\}$, b, $\{b,g\}$, g, $\{g,f\}$, f, $\{f,e\}$, e, $\{e,d\}$, d. It is an M-augmenting chain.

In closing, we note that it can be shown that the algorithm described can be modified to take on the order of $[\min\{|X|,|Y|\}] \cdot |E|$ steps, given a bipartite graph $G = (X, Y, E)$. Thus, it is a polynomial algorithm in the number of vertices $n = |V|$. For $|E| \leq \binom{n}{2} \leq n^2$ and $\min\{|X|,|Y|\} \leq n$. Thus, the algorithm concludes in a number of steps that is on the order of n^3. In the notation of Section 2.18, we say it is an $O(n^3)$ algorithm. A related algorithm for arbitrary G also takes on the order of n^3 steps. As of this writing, the fastest known algorithms for finding a maximum matching take on the order of $|E||V|^{1/2}$ steps. In terms of n, these algorithms take on the order of $n^{5/2}$ steps. They are due to Hopcroft and Karp [1973] for the bipartite case and to Micali and Vazirani [1980] for the general case. These algorithms relate matching to network flows. (We discuss this relation in Section 13.3.8.) For a more detailed discussion of the complexity for matching algorithms, see Ahuja, Magnanti, and Orlin [1993], Cook, *et al.* [1998], Lawler

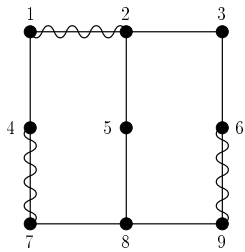

Figure 12.20: Matching for exercises of Section 12.5.

[1976], or Papadimitriou and Steiglitz [1982].

EXERCISES FOR SECTION 12.5

1. In the matching of Figure 12.20:
   (a) Find an M-alternating chain that is not M-augmenting.
   (b) Find an M-augmenting chain if one exists.
   (c) Use the chain in part (b), if it exists, to find a larger matching.
2. Repeat Exercise 1 for the matching of Figure 12.21.
3. Repeat Exercise 1 for the matching of Figure 12.22.
4. Repeat Exercise 1 for the matching of Figure 12.23.
5. Repeat Exercise 1 for the matching of Figure 12.24.
6. Apply subroutine 1 to:
   (a) The matching M of Figure 12.20 and the vertex 5
   (b) The matching M of Figure 12.20 and the vertex 3
   (c) The matching M of Figure 12.21 and the vertex 12
7. Apply subroutine 1 to the matching M of Figure 12.24 by starting with vertex 1 and, in successive iterations, choosing $v = 2, v = 4, v = 6, v = 4, v = 8$.
8. Show that if subroutine 1 is applied to the matching of Figure 12.23, starting with the vertex 3, it is possible for the edge $\{u, v\}$ chosen at Step 2.1 to join two outer vertices. Why can this happen?
9. Find an alternating tree starting at vertex x in the following situations.
   (a) $x = 6$ and the matching of Figure 12.21
   (b) $x = 8$ and the matching of Figure 12.20
10. Show that in the proof of Theorem 12.7, each component of H is either Z_n or L_n.
11. Prove that in Step 2.2 of Algorithm 12.2, v cannot be an outer vertex.
12. When applying subroutine 2, show that once a vertex is saturated, it stays saturated.
13. Prove that Algorithm 12.2 works.

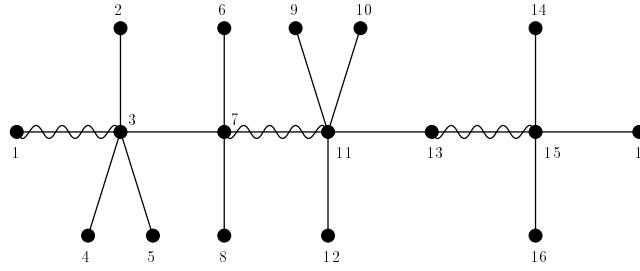

Figure 12.21: Matching for exercises of Section 12.5.

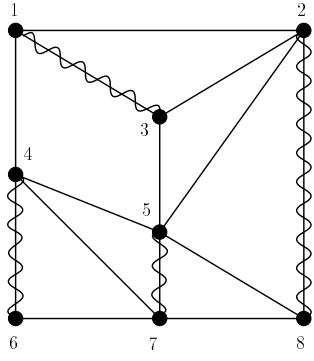

Figure 12.22: Matching for exercises of Section 12.5.

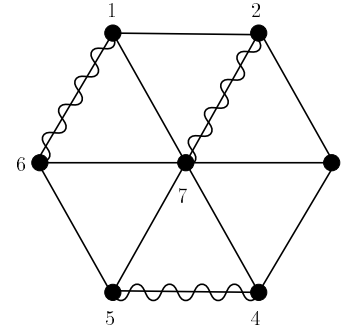

Figure 12.23: Matching for exercises of Section 12.5.

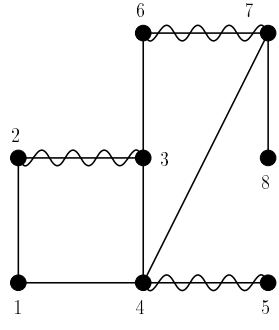

Figure 12.24: Matching for exercises of Section 12.5.

14. If a vertex x is unsaturated in a matching M and there is no M-augmenting chain starting at x, show that there is a maximum-cardinality matching in which x is unsaturated.

15. A subset S of vertices is called *matchable* if every vertex in S is matched in some matching. Show that if S is matchable, every vertex in S is matched in some maximum-cardinality matching.

16. Suppose that M is a matching in a graph G. A *blossom* relative to M is an odd-length circuit B of $2k+1$ vertices having k edges in M. Show that if there are no blossoms relative to M, subroutine 1 finds an M-augmenting chain beginning at x if there is one. (Thus, the algorithm we have described for finding a maximum matching must be modified only if blossoms are found. The modification due to Edmonds [1965a] works by searching for blossoms, shrinking them to single vertices, and searching for M-augmenting chains in the resulting graph.)

12.6 MATCHING AS MANY ELEMENTS OF X AS POSSIBLE

Suppose that $G = (X, Y, E)$ is a bipartite graph. If there is no X-saturating matching in G, we may at least wish to find a matching that matches as large a number of elements of X as possible. Let $m(G)$ be the largest number of elements of X that can be matched in a matching of G. We show how to compute $m(G)$. First, we mention a couple of applications of this idea.

Example 12.17 Telephone Switching Networks At a telephone switching station, phone calls are routed through incoming lines to outgoing trunk lines. A *switching network* connects each incoming line to some of the outgoing lines. When a call comes in on an incoming line, it is routed through the switching network to an outgoing line. The switching network can be represented by a bipartite graph $G = (X, Y, E)$. The vertices of X are the incoming lines, the vertices of Y are the outgoing lines, and there is an edge between an incoming line and an outgoing line if and only if the former is connected to the latter. Suppose that a number of incoming calls come in at once. We want to be able to send all of them out at the same time if possible. If S is the set of incoming lines on which there are calls, we want an assignment of each line of S to an edge of G leading to a different outgoing trunk line. That is, we want an S-saturating matching of G. If this is impossible to find, we want to match a large number of lines in S. In general, we want to design our switching network so that if calls come in on all possible incoming lines, we can match a large number of them with outgoing lines. That is, we want to design a switching network so that it has a maximum matching that has a large number of edges. Put another way, we want to find a bipartite graph $G = (X, Y, E)$ with given X and Y so that $m(G)$ is large enough to be reasonable. ■

Example 12.18 Smallpox Vaccinations (Example 12.3 Revisited) In planning ahead for a public health emergency such as a smallpox outbreak, we want to

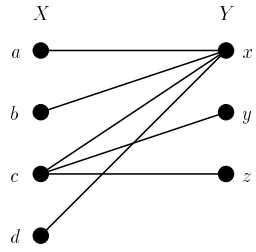

Figure 12.25: A graph G with $\delta(G) = 2$.

be sure that we have enough facilities for vaccinating people. However, the vaccination facilities might not have enough capacity to vaccinate everyone, at least not in a short time. Suppose that each facility can handle at most a certain number of people a day. We want to arrange assignments of people to facilities so that we can vaccinate as many people as possible in the first day. In other words, we want to make the assignment of people to facilities, defining a graph G, so that $m(G)$ is as large as possible. ■

If $S \subseteq X$, let the *deficiency* of S, $\delta(S)$, be defined as $|S| - |N(S)|$, and let the *deficiency* of G, $\delta(G)$, be $\max_{S \subseteq X} \delta(S)$. Note that $\delta(\emptyset) = 0$, so by Philip Hall's Theorem, G has an X-saturating matching if and only if $\delta(G) = 0$.

Theorem 12.8 (König [1931]) If $G = (X, Y, E)$ is a bipartite graph, $m(G) = |X| - \delta(G)$.

To illustrate this theorem, consider the graph of Figure 12.25. Note that $\delta(\{a, b, d\}) = 2$ and this is maximum, so $\delta(G) = 2$. Thus, $m(G) = |X| - \delta(G) = 4 - 2 = 2$. The largest subset of X that can be matched has two elements. An example of such a set is $\{a, c\}$.

A proof of Theorem 12.8 is sketched in Exercises 6 and 7.

EXERCISES FOR SECTION 12.6

1. Compute $\delta(G)$ and $m(G)$ for each graph of Figure 12.12.
2. Consider a switching station with 9 incoming lines and 6 outgoing trunk lines. Suppose that by engineering considerations, each incoming line is to be connected in the switching network to exactly two outgoing lines, and each outgoing line can be connected to at most 3 incoming lines.
   (a) Consider any set S of p incoming lines. Show that in the switching network, the sum of the degrees of the vertices in S is at most the sum of the degrees of the vertices in $N(S)$.
   (b) Using the result in part (a), show that for any set S of p incoming lines, $N(S)$ has at least $\frac{2}{3}p$ elements.
   (c) Conclude that for any set S of p incoming lines, $\delta(S) \leq \frac{1}{3}p$.
   (d) Conclude that no matter how the switching network is designed, there is always a matching that matches at least 6 incoming calls.

3. Suppose that the situation of Exercise 2 is modified so that there are 12 incoming lines and 10 outgoing lines. If each incoming line is to be connected to exactly 3 outgoing lines and each outgoing line to at most 4 incoming lines, show that no matter how the switching network is designed, there is always a matching that matches at least 9 incoming calls.
4. Suppose that the situation of Exercise 2 is modified so that there are two kinds of incoming lines, 4 of type I and 4 of type II, and there are 8 outgoing lines, all of the same type. Assume that each type I incoming line is connected to exactly 3 outgoing trunk lines and each type II incoming line to exactly 6 outgoing trunk lines, and assume that each outgoing line is connected to at most 4 incoming lines. Show that no matter how the switching network is designed, there is always a matching that matches at least 7 incoming calls. Do so by considering a set S of p incoming lines of type I and q incoming lines of type II. Show that
$$\delta(S) \leq (p+q) - \frac{3p+6q}{4}.$$
Conclude that since $p \leq 4$ and $q \geq 0$, $\delta(G) \leq 1$.
5. Suppose that in a small town there are four smallpox vaccination facilities and 56 people. Additionally, each person will be vaccinated at one of two facilities of their choosing, but each facility has a daily capacity of 14 people. Define a bipartite graph G so that $m(G)$ corresponds to the maximum number of people that can be vaccinated in the town in a single day.
6. As a preliminary to the proof of Theorem 12.8, prove the following: A bipartite graph $G = (X, Y, E)$ has a matching of k elements if and only if for all subsets $S \subseteq X$, $|N(S)| \geq |S| + k - |X| = k - |X - S|$. (*Hint:* Modify G by adding $|X| - k$ vertices to Y and joining each new vertex of Y to each vertex of X.)
7. Use the result of Exercise 6 to prove Theorem 12.8.
8. This exercise presents an alternative proof of the König-Egerváry Theorem (Corollary 12.5.1). Let $\mathbf{A}$ be a matrix of 0's and 1's and build a bipartite graph $G = (X, Y, E)$ from $\mathbf{A}$ as in the proof of the König-Egerváry Theorem. Let S be a subset of X, the set of rows, such that $\delta(S) = \delta(G)$.
   (a) Show that the rows corresponding to vertices of $X - S$ together with the columns corresponding to vertices of $N(S)$ contain all the 0's in $\mathbf{A}$.
   (b) Show that the total number of rows and columns in part (a) is $|X| - \delta(G)$.
   (c) Conclude that the minimum number of lines of $\mathbf{A}$ covering all 0's is at most $|X| - \delta(G)$.
   (d) Show that the maximum number of independent 0's in G is $|X| - \delta(G)$.
   (e) Prove the König-Egerváry Theorem.

12.7 MAXIMUM-WEIGHT MATCHING

In Examples 12.5–12.9 of Section 12.1, we introduced the problem of finding a matching of maximum (or minimum) weight in a graph where we have weights on its edges. Here we discuss the problem in more detail, including an algorithm for solving it presented in the language of one of these problems.

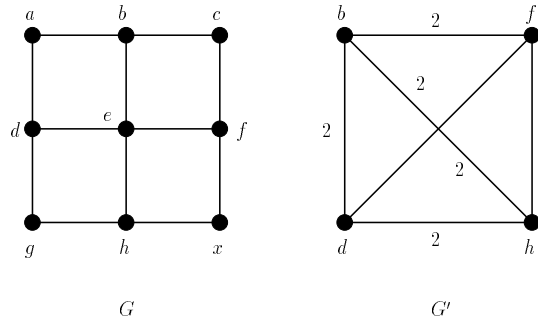

Figure 12.26: The graph G of Figure 11.28 and the corresponding graph G'.

12.7.1 The "Chinese Postman" Problem Revisited

In Section 11.4.1 we discussed the problem of a mail carrier who wishes to find the smallest number of blocks to walk, yet cover all the blocks on an assigned route. We formulated the problem as follows. Given a graph G, we study *feasible multigraphs* H, multigraphs that have an eulerian closed chain and are obtained from G by adding copies of edges of G. We seek an *optimal multigraph*, a feasible multigraph with a minimum number of edges. In Section 11.4.2 we observed that the same problem arises in connection with computer graph plotting. In Exercise 11, Section 11.4, the reader was asked to observe that in any optimal multigraph H, the newly added edges could be divided up into chains joining vertices of odd degree in G, with any such vertex an end vertex of exactly one such chain. Now suppose that for every pair of vertices u and v of odd degree in G, we find between them a shortest chain in G. [We can use an algorithm like Dijkstra's Algorithm (Section 13.2.2) to do this.] Let us build a graph G' by taking as vertices all odd-degree vertices in G, joining each pair of vertices by an edge, and putting on this edge a weight equal to the length of the shortest chain between u and v. To illustrate, suppose that G is the graph of Figure 11.28, that is repeated in Figure 12.26. Then there are four vertices of odd degree in G: b, f, h, and d. Between each pair of these, a shortest chain clearly has length 2. Hence, the graph G' is as shown in the second part of Figure 12.26. Now any optimal multigraph H obtained from G corresponds to a collection of chains joining pairs of odd-degree vertices in G, with each such vertex an end vertex of exactly one such chain. Such a collection of chains defines a perfect matching in the graph G'. Moreover, an optimum H corresponds to a collection of such chains, the sum of whose lengths is as small as possible. This collection in turn defines a perfect matching in G' of minimum weight. If G' is changed by taking the negative of each weight, we seek a maximum-weight matching in G'. (Such a matching will be perfect. Why?) This is a problem that we discussed briefly in Examples 12.5–12.9. In our example, a minimum-weight perfect matching in G' consists of any two edges that do not touch, such as, $\{b, d\}$ and $\{f, h\}$. Then the shortest chains corresponding to the chosen edges will define the optimal multigraph H. Here a shortest chain between b and d is b, e, d, and one between f and h is

f, x, h. If we add to G the edges on these chains, we get the first optimal multigraph H of Figure 11.28. This is now known to be the smallest number of edges that could be added to give an H that has an eulerian closed chain. The eulerian closed chain in H will give us an optimal mail carrier's route in G.

The approach we have described is due to Edmonds (see Edmonds [1965b] and Edmonds and Johnson [1973]). It should be pointed out that the procedure described is an efficient one. For an efficient shortest path algorithm (Dijkstra's Algorithm of Section 13.2.2) can be completed in a number of steps on the order of n^2 and the minimum- (maximum-) weight matching can be found in a number of steps on the order of n^3 (Lawler [1976]).

12.7.2 An Algorithm for the Optimal Assignment Problem (Maximum-Weight Matching)[8]

In Example 12.6 we discussed the following job assignment problem. There are n workers and m jobs, every worker is suited for every job, and worker i's potential performance on job j is given a rating r_{ij}. We wish to assign workers to jobs so as to maximize the sum of the performance ratings. This is an example of a maximum-weight matching problem. Let us assume that $m = n$ and let x_{ij} be a variable that is 1 if worker i is assigned to job j and 0 otherwise. Then, since $m = n$, we want to maximize

$$\sum_{\substack{i=1 \\ j=1}}^{\substack{m \\ n}} r_{ij} x_{ij}.$$

We require that no two workers get the same job, that is, that

$$\sum_{i=1}^{n} x_{ij} \leq 1. \tag{12.8}$$

We also require that every worker get a job, that is, that

$$\sum_{j=1}^{n} x_{ij} \geq 1. \tag{12.9}$$

Then since the number of workers equals the number of jobs, (12.8) and (12.9) give us the constraints

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{and} \quad \sum_{j=1}^{n} x_{ij} = 1. \tag{12.10}$$

We now present an algorithm for solving the optimal assignment problem. Suppose that we are given an $n \times n$ matrix (c_{ij}) and we wish to find numbers $x_{ij} = 0$ or 1 such that we minimize $\sum_{i,j} c_{ij} x_{ij}$ and such that (12.10) holds. Note that if a

[8] This subsection may be deferred until Section 13.4.

constant p_k is subtracted from all entries in the kth row of (c_{ij}), giving rise to a matrix (c'_{ij}), then

$$\sum_{i,j} c'_{ij}x_{ij} = \sum_{i,j} c_{ij}x_{ij} - p_k \sum_j x_{kj} = \sum_{i,j} c_{ij}x_{ij} - p_k,$$

by (12.10). Thus, the assignment x_{ij} that minimizes $\sum_{i,j} c_{ij}x_{ij}$ will also minimize $\sum_{i,j} c'_{ij}x_{ij}$. The same is true if a constant q_l is subtracted from each entry in the lth column of (c_{ij}). Indeed, the same is true if we subtract constants p_i from the ith row of (c_{ij}) for all i and q_j from the jth column of (c_{ij}) for all j.

Using this observation, we let p_i be the minimum element in the ith row of (c_{ij}). For each i, we subtract p_i from each element in the ith row of (c_{ij}), obtaining a matrix (c'_{ij}). We now let q_j be the minimum element in the jth column of (c'_{ij}), and for each j, subtract q_j from each element in the jth column of (c'_{ij}), obtaining a matrix $(\bar{c}_{ij})$. This is called the *reduced matrix*. By what we have observed, we may as well solve the optimal assignment problem using the reduced matrix.

To illustrate this procedure, suppose that $n = 4$ and

$$(c_{ij}) = \begin{bmatrix} 12 & 14 & 15 & 14 \\ 9 & 6 & 11 & 8 \\ 10 & 9 & 16 & 14 \\ 12 & 13 & 13 & 10 \end{bmatrix}. \tag{12.11}$$

Then $p_1 = 12, p_2 = 6, p_3 = 9, p_4 = 10$, and

$$(c'_{ij}) = \begin{bmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{bmatrix}.$$

Now $q_1 = 0, q_2 = 0, q_3 = 3$, and $q_4 = 0$, so

$$(\bar{c}_{ij}) = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & 0 & 4 & 5 \\ 2 & 3 & 0 & 0 \end{bmatrix}. \tag{12.12}$$

Note that the reduced matrix $(\bar{c}_{ij})$ always has nonnegative entries. (Why?) A job assignment x_{ij} giving each worker one job and each job one worker corresponds to a choice of n entries of this matrix, one in each row and one in each column. We have $x_{ij} = 1$ if and only if the i, j entry is picked. Now suppose that we can find a choice of n entries, one in each row and one in each column, so that all the entries are 0. Let us take $x_{ij} = 1$ for precisely these n i, j entries and $x_{ij} = 0$ otherwise. Then the corresponding $\sum \bar{c}_{ij}x_{ij}$ will be 0, because when x_{ij} is 1, $\bar{c}_{ij} = 0$. For instance, if

$$(\bar{c}_{ij}) = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 5 & 4 \\ 0 & 3 & 0 \end{bmatrix},$$

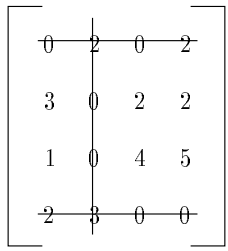

Figure 12.27: A minimum covering of 0's by lines.

we can pick the $1,2$, the $2,1$, and the $3,3$ entries. If we take $x_{12} = 1, x_{21} = 1, x_{33} = 1$, and $x_{ij} = 0$ otherwise, then $\sum \bar{c}_{ij}x_{ij} = 0$. This is clearly a minimum, since $\sum \bar{c}_{ij}x_{ij} \geq 0$ because $\bar{c}_{ij} \geq 0$.

Recall from Section 12.4.1 that an *independent set of* 0's in a matrix is a collection of 0's no two of which are in the some row and no two of which are in the same column. What we have just observed is that if we can find an independent set of n 0's in $(\bar{c}_{ij})$, we can find an optimal job assignment by taking the corresponding x_{ij} to be 1. Since there can be no independent set of more than n 0's in $(\bar{c}_{ij})$, we look for a maximum independent set of 0's. How do we find such a set? Let us change all positive entries in $(\bar{c}_{ij})$ to 1. Then since $\bar{c}_{ij}$ is nonnegative, the resulting matrix is a matrix of 0's and 1's. Recall from Section 12.4.1 that a *line* of a matrix is either a row or a column. Then by the König-Egerváry Theorem (Corollary 12.5.1), the maximum number of independent 0's in $(\bar{c}_{ij})$, equivalently in its modified matrix of 0's and 1's, is equal to the minimum number of lines that cover all the 0's. Thus, we can check to see if there is a set of n independent 0's by checking to see if there is a set of n lines that cover all the 0's. Alternatively, from the proof of the König-Egerváry Theorem, we recall that a maximum independent set of 0's in the matrix $(\bar{c}_{ij})$, equivalently in the modified matrix of 0's and 1's, corresponds to a maximum matching in the bipartite graph $G = (X, Y, E)$, where $X = Y = \{1, 2, \ldots, n\}$ and where there is an edge between i in X and j in Y iff $\bar{c}_{ij} = 0$. Thus, we can apply the maximum matching algorithm of Section 12.5.3 to see if there is a set of n independent 0's.

In our example of Equation (12.11), we have derived $(\bar{c}_{ij})$ in Equation (12.12). Note that the minimum number of lines covering all the 0's in $(\bar{c}_{ij})$ is 3: Use the first and fourth rows and the second column (see Figure 12.27). Thus, the maximum independent set of 0's has three elements, not enough to give us a job assignment, as we need four independent 0's.

The algorithm for solving the optimal assignment problem proceeds by successively modifying the reduced matrix $(\bar{c}_{ij})$ so that eventually we obtain one where we can find n independent 0's. The modification step is to use the minimum covering of 0's by lines, such as shown in Figure 12.27, and find the smallest uncovered element. Then subtract this from each uncovered element and add it to each twice-covered element and find a new matrix in which to search for independent 0's. In our example, the smallest uncovered element is 1, the $3,1$ element. We subtract 1

from the $2,1$, the $2,3$, the $2,4$, the $3,1$, the $3,3$, and the $3,4$ entries of $(\bar{c}_{ij})$, and add it to the $1,2$ and the $4,2$ entries, obtaining the new reduced matrix

$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 2 & 0 & 1 & 1 \\ 0 & 0 & 3 & 4 \\ 2 & 4 & 0 & 0 \end{bmatrix}. \tag{12.13}$$

It is not hard to show that the new reduced matrix has been obtained from the preceding one by adding or subtracting a constant from different rows or columns (Exercise 8). Thus, solving the optimal assignment problem with this matrix is the same as solving it with the previous reduced matrix. Also, the new reduced matrix has all entries nonnegative. Thus, once again with this new matrix, we can seek an independent set of n 0's. We seek this by finding a minimum covering by lines. In (12.13), a minimum covering uses four lines, and hence there must be an independent set of four 0's. One such set consists of the $1,3$ entry, the $2,2$ entry, the $3,1$ entry, and the $4,4$ entry. Hence, we can find an optimal assignment by letting $x_{13} = 1, x_{22} = 1, x_{31} = 1, x_{44} = 1$, and all other $x_{ij} = 0$. If we had not found a set of n independent 0's, we would have repeated the modification of the reduced matrix.

The algorithm we have described, called the *Hungarian Algorithm*, is due to Kuhn [1955]. We summarize it as follows.

Algorithm 12.3: The Hungarian Algorithm for the Optimal Assignment Problem

Input: An $n \times n$ matrix (c_{ij}).
Output: An optimal assignment x_{ij}.

Step 1. (*Initialization*)

> ***Step 1.1.*** For each i, let p_i be the minimum element in the ith row of (c_{ij}).
>
> ***Step 1.2.*** Let (c'_{ij}) be computed from (c_{ij}) by subtracting p_i from each element of the ith row, for all i.
>
> ***Step 1.3.*** For each j, let q_j be the minimum element in the jth column of (c'_{ij}).
>
> ***Step 1.4.*** Let $(\bar{c}_{ij})$ be computed from (c'_{ij}) by subtracting q_j from each element of the jth column, for all j.

Step 2.

> ***Step 2.1.*** Find a minimum collection of lines covering the 0's of $(\bar{c}_{ij})$.
>
> ***Step 2.2.*** If this collection of lines has fewer than n elements, go to Step 3. Otherwise, go to Step 4.

Step 3. (*Modification of Reduced Matrix*)

> ***Step 3.1.*** Using the covering obtained in Step 2.1, let p be the smallest uncovered element in $(\bar{c}_{ij})$.

Step 3.2. Change the reduced matrix $(\bar{c}_{ij})$ by subtracting p from each uncovered element and adding p to each twice-covered element. Return to Step 2.

Step 4.

Step 4.1. Find a set of n independent 0's in $(\bar{c}_{ij})$.

Step 4.2. Let x_{ij} be 1 if the i,j entry of $(\bar{c}_{ij})$ is one of the independent 0's, and let x_{ij} be 0 otherwise. Output this solution x_{ij} and stop.

Theorem 12.9 If all of the c_{ij} are integers, the Hungarian Algorithm gives an optimal assignment.

Proof.[9] We have already observed that if the algorithm gives an assignment x_{ij}, this must be optimal. But how do we know that the algorithm will ever give an assignment? The reason it does is because the reduced matrix always has nonnegative integer entries and because at each modification, the sum of the entries in this matrix decreases by an integer at least 1, as we show below. Thus, in a finite number of steps, if we have not yet reached an optimal solution, all the entries of the reduced matrix will be 0. In this case, there is, of course, a collection of n independent 0's, so we find an optimal solution.

We now show that if $(\bar{d}_{ij})$ is the modified reduced matrix obtained in Step 3 from the reduced matrix $(\bar{c}_{ij})$, then

$$\sum_{i,j}(\bar{c}_{ij}) - \sum_{i,j}(\bar{d}_{ij}) = \text{an integer} \geq 1.$$

Recall that we have a covering of $(\bar{c}_{ij})$ by $k < n$ lines. Let S_r be the set of uncovered rows of $(\bar{c}_{ij})$, S_c be the set of uncovered columns, $\bar{S}_r$ be the set of covered rows, and $\bar{S}_c$ the set of covered columns. Let $\alpha = |S_r|$ and $\beta = |S_c|$. Then $k = (n-\alpha)+(n-\beta)$. Also, recall that p is the smallest uncovered entry of $(\bar{c}_{ij})$. Then, remembering how we get $(\bar{d}_{ij})$ from $(\bar{c}_{ij})$, we have

$$\begin{aligned}
\textstyle\sum_{i,j}\bar{c}_{ij} - \sum_{i,j}\bar{d}_{ij} &= \textstyle\sum_{\substack{i\in S_r\\ j\in S_c}}[\bar{c}_{ij}-\bar{d}_{ij}] + \sum_{\substack{i\in S_r\\ j\in \bar{S}_c}}[\bar{c}_{ij}-\bar{d}_{ij}] \\
&\quad + \textstyle\sum_{\substack{i\in \bar{S}_r\\ j\in S_c}}[\bar{c}_{ij}-\bar{d}_{ij}] + \sum_{\substack{i\in \bar{S}_r\\ j\in \bar{S}_c}}[\bar{c}_{ij}-\bar{d}_{ij}] \\
&= \textstyle\sum_{\substack{i\in S_r\\ j\in S_c}} p + \sum_{\substack{i\in S_r\\ j\in \bar{S}_c}} 0 + \sum_{\substack{i\in \bar{S}_r\\ j\in S_c}} 0 + \sum_{\substack{i\in \bar{S}_r\\ j\in \bar{S}_c}} (-p) \\
&= \alpha\beta p - (n-\alpha)(n-\beta)p \\
&= n(\alpha+\beta-n)p.
\end{aligned}$$

But $\alpha+\beta$ is the number of uncovered rows and columns, so

$$\alpha+\beta-n = (2n-k)-n = n-k > 0,$$

since $k = (n-\alpha)+(n-\beta)$ and $k < n$. Thus,

$$\sum_{i,j}\bar{c}_{ij} - \sum_{i,j}\bar{d}_{ij} = n(\alpha+\beta-n)p = \text{an integer} \geq 1,$$

[9] The proof may be omitted.

since $n, \alpha + \beta - n$, and p are all positive integers. Q.E.D.

We close by observing that if there are n workers and n jobs, the Hungarian Algorithm can be implemented in $O(n^3)$ time, to use the terminology of Section 2.18. For a discussion of this point, see Papadimitriou and Steiglitz [1982].

EXERCISES FOR SECTION 12.7

1. For each graph G of Figure 11.39, find the corresponding graph G' as described in the solution to the "Chinese Postman" Problem in Section 12.7.1. Find a minimum-weight perfect matching in G' and translate this into a solution to the "Chinese Postman" Problem.

2. Each of the following matrices give the cost c_{ij} of using worker i on job j. Use the Hungarian Algorithm to find a minimum-cost job assignment.

(a)
	1	2	3	4
1	8	3	2	4
2	10	9	3	6
3	2	1	1	5
4	3	8	2	1

(b)
	1	2	3	4
1	17	5	8	11
2	3	9	2	10
3	4	2	8	6
4	7	6	4	5

(c)
	1	2	3	4	5
1	8	7	5	11	4
2	9	7	6	11	3
3	12	9	4	8	2
4	1	2	3	5	6
5	11	4	2	8	2

3. The following matrix gives the rating r_{ij} of worker i on job j. Use the Hungarian Algorithm to find an optimum (i.e., highest rated) job assignment.

	1	2	3	4	5
1	8	7	5	9	6
2	11	9	7	4	8
3	12	6	7	5	10
4	9	7	6	9	6
5	3	9	8	9	8

4. A company has purchased five new machines of different types. In its factory, there are five locations where the machines can be located. For a given machine, its location in a particular spot would have an effect on the ease of handling materials. For instance, if the machine is near the work center that produces materials for it, this would be efficient. For machine i at location j, the hourly cost of handling materials to be brought to the machine can be estimated. The following matrix gives this information.

		Location				
		1	2	3	4	5
Machine	1	4	6	8	5	7
	2	2	4	6	9	5
	3	1	7	6	8	3
	4	4	6	7	5	7
	5	10	4	5	4	5

How would we assign machines to locations so as to minimize the resulting materials handling costs?

5. The following matrix gives the frequency response f_{ij} when speakers i and j are paired (see Example 12.8).

   (a) Use the Hungarian Algorithm to find a minimum-frequency response pairing.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 4 & 8 & 8 & 7 & 8 \\ 4 & 0 & 6 & 8 & 1 & 6 \\ 8 & 6 & 0 & 5 & 7 & 4 \\ 8 & 8 & 5 & 0 & 4 & 2 \\ 7 & 1 & 7 & 4 & 0 & 7 \\ 8 & 6 & 4 & 2 & 7 & 0 \end{pmatrix} \end{array}$$

   (b) If all entries in a single row were the same (except for the diagonal entry, which must be 0), how would this affect your use of the Hungarian Algorithm?

6. Recall the real estate transactions problem of Example 12.5 (Section 12.1). Suppose that an agent has three listed houses, H_1, H_2, and H_3, and four buyers, B_1, B_2, B_3, and B_4. She stands to receive 20, 18, and 8 thousand dollars if B_1 buys H_1, H_2, or H_3, respectively. She also stands to receive 20, 12, and 16 if B_2 buys a house; 18, 20, and 20 if B_3 buys a house; and 16, 18, and 16 if B_4 buys a house. Is there a unique solution to her problem of maximizing profits? (*Hint:* Introduce a "dummy" fourth house.)

7. (a) Suppose that certain people have to be assigned to certain facilities in the smallpox vaccination problem of Example 12.7 (Section 12.1). Explain how you would accommodate this requirement. In particular, what changes would you make to the matrix input for the Hungarian Algorithm?

   (b) Now suppose that certain people cannot be assigned to certain facilities in the smallpox vaccination problem of Example 12.7 (Section 12.1). Explain how you would accommodate this requirement. In particular, what changes would you make to the matrix input for the Hungarian Algorithm?

8. Show that in the Hungarian Algorithm, the new reduced matrix is obtained from the old by adding or subtracting a constant from different rows or columns.

12.8 STABLE MATCHINGS

In Example 12.12 we introduced the problem of how to assign medical school graduates to internships in hospitals. In 1952, an algorithm called the National Intern Matching Program was developed to make such an assignment. The idea was that interns would submit a ranking of the hospitals to which they had applied and the hospitals would similarly rank their applicants. Essentially unchanged, the algorithm now called the National Resident Matching Program is still in use today. It is aimed at producing a "stable" matching.

The idea of combining preferences with matchings to form a stable matching was formally introduced by Gale and Shapley [1962]. We discussed the idea briefly in Section 4.2.5. This work has led to numerous theoretical extensions as well as further practical applications. See Gusfield and Irving [1989] and Roth and Sotomayor

Table 12.4: Preference Orderings for a Size 4 Stable Marriage Problem

Men's Preferences				Women's Preferences			
m_1	m_2	m_3	m_4	w_1	w_2	w_3	w_4
w_1	w_2	w_3	w_4	m_4	m_3	m_2	m_1
w_2	w_1	w_4	w_3	m_3	m_4	m_1	m_2
w_3	w_4	w_1	w_2	m_2	m_1	m_4	m_3
w_4	w_3	w_2	w_1	m_1	m_2	m_3	m_4

[1990] for extended discussions. (For another approach to stability, Balinski and Ratier [1997] redeveloped the idea of stable matchings in the language of directed graphs.)

One particular type of stable matching introduced by Gale and Shapley [1962] solves a stable marriage problem. This *stable marriage problem of size* n involves a set of n men and a set of n women. Each person supplies a (strict) preference ordering of all members of the opposite sex. If monogamous marriages are set up for each person, this set of marriages will be called a matching, i.e., a one-to-one correspondence between the men and women. A matching is called *unstable* if some man and woman are each willing, based on their preference ordering, to leave their respective partners for each other. Such a man−woman pair will be called a *blocking pair*. If no blocking pair exists, the matching is called *stable*.

In this section we consider the existence problem, the counting problem, and the optimization problem as they relate to the stable marriage problem. The following example highlights these points.

Example 12.19 Stable Marriages[10] Suppose that four men and four women are to be married to each other. Their preference orderings are given in Table 12.4. We denote the marriage between m_i and w_j by $m_i - w_j$. Note that $M = \{m_1 - w_4, m_2 - w_3, m_3 - w_1, m_4 - w_2\}$ is a stable set of marriages. To see why, note that w_4 and w_3 married their first choice; so neither would be willing to leave their partner, m_1 and m_2, respectively. Also, m_3 and m_4 are doing their best with the remaining women, that is, w_1 and w_2, respectively.

While M is stable, there are actually nine other stable matchings associated with these people and their preferences. This is out of a total of $4! = 24$ matchings. (Why?) $M_u = \{m_1 - w_1, m_2 - w_3, m_3 - w_4, m_4 - w_2\}$ is one example of an unstable matching. To see why, note that m_2, although married to w_3, actually prefers w_4 (as well as w_1 and w_2). And w_4, married to m_3, actually prefers m_2 (and m_1). Since m_2 and w_4 prefer each other to their respective partners, they form a blocking pair and hence M_u is unstable.

Two other obvious stable matchings are $M_m = \{m_1 - w_1, m_2 - w_2, m_3 - w_3, m_4 - w_4\}$ and $M_w = \{m_1 - w_4, m_2 - w_3, m_3 - w_2, m_4 - w_1\}$. These are both stable since each man in M_m and each woman in M_w got their first choice. Also, because of

[10] This example is from Gusfield and Irving [1989].

this fact, note that any man would prefer M_m and any woman would prefer M_w to any of the other stable matchings. ■

12.8.1 Gale-Shapley Algorithm

Does a stable set of marriages exist for every stable marriage problem? This existence question was settled positively by Gale and Shapley [1962]. We describe the Gale-Shapley (GS) Algorithm for finding a stable matching. The GS Algorithm allows for interim "engagements" until its final stage, when all engagements are finalized in marriage. A stage is a two-step process where all free (unengaged) men propose and then all women with proposals become engaged.

The GS Algorithm starts with each man proposing to the number 1 woman on his preference list. Each woman then becomes engaged to the proposer who is highest on her preference list. Stage 2 has all remaining free men proposing to their number 2 woman and then each woman becoming engaged to the best of either her new proposal(s) or her fiancé, if one exists. (Note that at any stage, some women may not be making any decision, as they may not have been proposed to in that stage.)

Each subsequent stage proceeds in the same manner. All free men propose to the highest woman on their preference list to whom they haven't yet proposed, and each woman compares her new proposal(s) and fiancé for the best choice and enters into her new engagement. The GS Algorithm terminates when all women (men) are engaged.

Is the matching produced by the GS Algorithm stable? Suppose that m_i and w_j are not married but m_i prefers w_j to his wife w_k. It must be the case that m_i proposed to w_j before w_k. However, when w_j received m_i's proposal, either she rejected it in favor of someone whom she preferred or became engaged to m_i and later rejected him in favor of a better proposal. In either case, w_j prefers her husband to m_i and the matching produced by the GS Algorithm must be stable.

Note that the GS Algorithm could also be carried out with the women doing the proposing, also producing a stable matching. This stable matching could be different from that obtained using the "man proposing" version.

To compute the complexity of the GS Algorithm, we note that at each step, a woman who receives a proposal either (i) receives a proposal for the first time or (ii) rejects a new proposal or breaks an engagement. She does (i) exactly once and (ii) at most $n-1$ times. Thus, the algorithm has $O(n)$ steps for each woman and therefore $O(n^2)$ steps in all.

One final note about the GS Algorithm. We described a stage by saying that all free men propose to the next-highest woman on their preference list; however, no order in which the men should do so is given. Gusfield and Irving [1989] have shown that a particular ordering is inconsequential to the outcome of the GS Algorithm.

Theorem 12.10 (Gusfield and Irving [1989]) All possible executions of the GS Algorithm yield the same stable matching.

To see the GS Algorithm in action, consider the preference lists for the four

Table 12.5: Preference Orderings for a Size 4 Stable Marriage Problem

Men's Preferences				Women's Preferences			
$\underline{m_1}$	$\underline{m_2}$	$\underline{m_3}$	$\underline{m_4}$	$\underline{w_1}$	$\underline{w_2}$	$\underline{w_3}$	$\underline{w_4}$
w_1	w_2	w_3	w_2	m_4	m_3	m_2	m_1
w_4	w_1	w_4	w_3	m_3	m_4	m_1	m_2
w_3	w_4	w_1	w_4	m_2	m_1	m_4	m_3
w_2	w_3	w_2	w_1	m_1	m_2	m_3	m_4

Table 12.6: Preference Orderings for a Size 2 Stable Marriage Problem

Men's Preferences		Women's Preferences	
$\underline{m_1}$	$\underline{m_2}$	$\underline{w_1}$	$\underline{w_2}$
w_1	w_2	m_2	m_1
w_2	w_1	m_1	m_2

men and four women in Table 12.5. The GS Algorithm starts with m_1, m_2, m_3, m_4 proposing to w_1, w_2, w_3, w_2, respectively. The engagements resulting from these proposals are $m_1 - w_1, m_3 - w_3$, and $m_4 - w_2$. (Note that w_2 chose m_4 since he was higher on her list than m_2.) The second stage has m_2 proposing to his second choice, w_1. She accepts his proposal, switching engagements from m_1, since m_2 is higher on her list than m_1. The third stage has m_1 proposing to his second choice, w_4. She accepts his proposal and they become engaged. At this point, all women (and men) are engaged, so the GS Algorithm stops with all engagements becoming (stable) marriages.

Although the GS Algorithm assures the existence of at least one stable matching, there are instances of stable marriage problems with exactly one stable matching. That is, the GS Algorithm produces the only stable matching (see Exercise 6). In the next section we consider the maximum number of stable matchings that are possible over all stable marriage problems of size n.

12.8.2 Numbers of Stable Matchings

In any stable marriage problem of size 2, there are a total of $2! = 2$ matchings. By the GS Algorithm, there is always at least one stable matching. The next example shows that all (both) of the matchings could be stable. Consider the stable marriage problem of size 2 with the preference lists of Table 12.6. It is easy to check that $M_m = \{m_1 - w_1, m_2 - w_2\}$ and $M_w = \{m_1 - w_2, m_2 - w_1\}$ are both stable. Note that M_m is the output of the GS Algorithm if the men do the proposing, while M_w is the output if the women do the proposing.

In Example 12.19 we said that there were 10 stable matchings for this problem out of a total of 24 matchings. For small examples it is not too difficult to enumerate

all the matchings and to check individually if each one is stable. (Exercise 7 asks the reader to devise an algorithm to test a matching for stability.) However, for larger problems, enumeration is not possible since stable marriage problems of size n have $n!$ matchings. The next theorem gives a lower bound for the number of sets of stable marriages for problems of size n. (Finding the largest possible number of sets of stable marriages for problems of size n is an open problem; see Knuth [1997].)

Theorem 12.11 (Irving and Leather [1986]) For each $n \geq 1$, n a power of 2, there is a stable marriage problem of size n with at least 2^{n-1} stable matchings.

Proof. We will prove this by induction on n. For $n = 1 = 2^0$, $2^{n-1} = 2^0 = 1$, and by the GS Algorithm there must be (at least) one stable matching in any stable marriage problem, including size 1. For $n = 2 = 2^1$, the problem described in Table 12.6 gives a stable marriage problem of size 2 with $2 = 2^{2-1}$ stable matchings. This instance of a stable marriage problem will be important for this proof. Assume that the theorem is true for $n = 2^k$. That is, assume that there is a stable marriage problem of size $n = 2^k$ with at least $2^{n-1} = 2^{2^k-1}$ stable matchings. Suppose that the men and women in this case are labeled $b_1, b_2, \ldots, b_{2^k}$ and $g_1, g_2, \ldots, g_{2^k}$, respectively.

Recall that m_1, m_2 and w_1, w_2 are the men and women, respectively, from the stable marriage problem of size 2 in Table 12.6. To finish this proof, we need to create a new stable marriage problem that will have size 2^{k+1} and at least $2^{2^{k+1}-1}$ stable matchings.

Consider the stable matching problem of size 2^{k+1} with

$$\text{men} : (b_1, m_1), \ldots, (b_{2^k}, m_1), (b_1, m_2), \ldots, (b_{2^k}, m_2)$$

and

$$\text{women} : (g_1, w_1), \ldots, (g_{2^k}, w_1), (g_1, w_2), \ldots, (g_{2^k}, w_2).$$

The preference listings for each man will be as follows:

(b_i, m_j) prefers (g_k, w_l) to $(g_{k'}, w_{l'})$ if m_j prefers w_l to $w_{l'}$, or if $l = l'$ and b_i prefers g_k to $g_{k'}$.

The preference listings for the women are defined similarly. It now suffices to show that there are at least $2^{2^{k+1}-1}$ stable matchings for this problem.

Let $M_1, M_2, \ldots, M_n$ be any sequence of (not necessarily distinct) stable matchings from the stable marriage problem of Table 12.6; note that there are two choices for each M_i. Let M be any one of the (at least) 2^{2^k-1} stable matchings from the size $n = 2^k$ problem from above. Thus, the total number of possible choices available for these $n+1$ stable matchings, $M_1, M_2, \ldots, M_n, M$, is at least $2 \cdot 2 \cdot 2 \cdot \cdots \cdot 2 \cdot 2^{2^k-1}$, which equals $2^n \cdot 2^{2^k-1} = 2^{2^k} \cdot 2^{2^k-1} = 2^{2^{k+1}-1}$. We next show that a distinct stable matching in our new stable marriage problem of size 2^{k+1} can be defined from any one of these combinations. This would mean that we have found at least $2^{2^{k+1}-1}$

stable matchings for a stable marriage problem of size 2^{k+1} and the proof would be complete.

From the sequence $M_1, M_2, \ldots, M_n, M$ of stable matchings, we define a marriage as follows:

$$(b_i, m_j) \text{ marries } (b_i\text{'s partner in } M, m_j\text{'s partner in } M_i). \tag{12.14}$$

The proof is finished by showing that these marriages form a matching, each one is different, and that the matching is stable. These three facts are left to the exercises. Q.E.D.

Among other things, Theorem 12.11 shows that the number of stable matchings for stable marriage problems grows exponentially in the size of the problem.[11] Although finding all of the stable matchings for a given stable marriage problem is difficult, both algorithmically and numerically, we can still comment on the structure of the stable matchings. We do this next.

12.8.3 Structure of Stable Matchings

As in Section 4.2.5, we can define a *dominance* relation for stable matchings as follows: A person x prefers one stable matching, M_i, over another, M_j, if x prefers his/her partner in M_i to his/her partner in M_j. A *man-oriented dominance relation* says that M_i dominates M_j if every man either prefers M_i to M_j or is indifferent between them. (Recall that you are indifferent between alternatives A and B if you neither prefer A to B nor B to A.) A *woman-oriented dominance relation* is defined similarly.

As we stated in Section 4.2.5, stable matchings form a partial order (actually, they form a lattice) under either of the man-oriented or woman-oriented dominance relations. (The proof of this fact is part of Exercise 9.) Consider all of the stable matchings associated with the stable marriage problem of Table 12.4. There are 10 of them:

$$\begin{aligned}
M_0 &= \{m_1 - w_4, m_2 - w_3, m_3 - w_2, m_4 - w_1\} \\
M_1 &= \{m_1 - w_3, m_2 - w_4, m_3 - w_2, m_4 - w_1\} \\
M_2 &= \{m_1 - w_4, m_2 - w_3, m_3 - w_1, m_4 - w_2\} \\
M_3 &= \{m_1 - w_3, m_2 - w_4, m_3 - w_1, m_4 - w_2\} \\
M_4 &= \{m_1 - w_3, m_2 - w_1, m_3 - w_4, m_4 - w_2\} \\
M_5 &= \{m_1 - w_2, m_2 - w_4, m_3 - w_1, m_4 - w_3\} \\
M_6 &= \{m_1 - w_2, m_2 - w_1, m_3 - w_4, m_4 - w_3\} \\
M_7 &= \{m_1 - w_1, m_2 - w_2, m_3 - w_4, m_4 - w_3\} \\
M_8 &= \{m_1 - w_2, m_2 - w_1, m_3 - w_3, m_4 - w_4\} \\
M_9 &= \{m_1 - w_1, m_2 - w_2, m_3 - w_3, m_4 - w_4\}.
\end{aligned} \tag{12.15}$$

[11] Irving and Leather [1986] found a recurrence relation that improves greatly on Theorem 12.11.

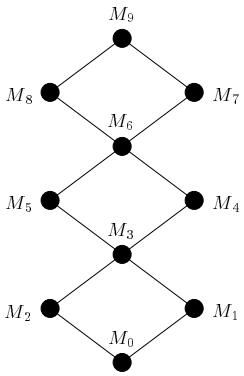

Figure 12.28: A lattice of all stable matchings (12.15) for the preferences in Table 12.4 under the man-oriented dominance relation.

The lattice diagram for this example, under the man-oriented dominance relation, is shown in Figure 12.28. One property of lattices is that they have a maximum and a minimum element. The maximum stable matching and minimum stable matching in the man-oriented lattice are called *man-optimal* and *man-pessimal*, respectively. Because of how man-oriented dominance is defined, of all the women to whom each man is married in any of the stable matchings, he ranks highest the one that he is married to in the maximum element of the man-oriented lattice. Notice, however, that w_1 is married to either m_1, m_2, m_3, or m_4 when considering all of the stable matchings in the lattice. Of the four men, she least prefers m_1 and he is the one she is married to in the maximum element of the man-oriented lattice. This is the case for all of the women; of all the men each woman is married to in any of the stable matchings, she least prefers the one that she is married to in the maximum element of the man-oriented lattice. This is not by coincidence.

Theorem 12.12 (McVitie and Wilson [1971]) In the man-optimal stable matching, each woman has her least preferred partner of all her stable marriage partners.

Proof. The proof will be by contradiction. Let M_m be the man-optimal stable matching. Suppose that woman w prefers m_1 to m_2 but she is married to m_1 in M_m and m_2 in another stable matching M. Since the marriage $w - m_1$ occurs in M_m, it must be the case that m_1 prefers w to any other woman in any of the other stable matchings, including M. A contradiction is reached by noting that m_1 and w must be a blocking pair in M. Q.E.D.

Corollary 12.12.1 The maximum element in the man-oriented lattice is the minimum element in the woman-oriented lattice, i.e., the man-optimal stable matching is the woman-pessimal stable matching.

Similar results hold for the woman-oriented dominance relation and lattice.

Finally, it is interesting to see which element in the man-oriented lattice is the stable matching that results from the GS Algorithm.

Table 12.7: Preference Lists for a 5-Resident, 2-Hospital Stable Matching Problem

Residents' Preferences					Hospitals' Preferences	
r_1	r_2	r_3	r_4	r_5	H_a	H_b
H_a	H_b	H_a	H_a	H_a	r_2	r_1
H_b	H_a	H_b	H_b	H_b	r_3	r_3
					r_4	r_5
					r_5	r_2
					r_1	r_4

Theorem 12.13 In the GS Algorithm's stable matching, each man has the best partner that he can have in any stable matching.

It should not be surprising that the GS Algorithm produces the maximum (man-optimal) element in the man-oriented lattice. For it is the men proposing and doing so from their top choice on down. Theorem 12.13 is proved in Exercise 10.

12.8.4 Stable Marriage Extensions

The stable marriage problem that was introduced in this section is fairly restrictive. Namely, the sets of men and women must be of equal size; each person's preference list contains every member of the other gender; there are no ties; marriages must be monogamous and heterosexual; and the list goes on. We discuss a number of variants of the stable marriage problem in this section as well as in the exercises.

Example 12.20 National Resident Matching Program When students graduate from medical school, they apply to hospitals to do their internships. Each student (resident) ranks the list of hospitals and the hospitals rank the list of residents. The National Resident Matching Program (NRMP) was devised to find a stable matching of the hospitals and residents. (See Roth [1982] for an extensive treatment of the NRMP.)

The NRMP is different from the stable marriage problem in a number of respects. First, the number of residents and hospitals are not necessarily the same. Also, each hospital typically needs to be matched with more than one resident. Note that these two conditions are not unrelated.

The NRMP Algorithm, that is a slight modification of the GS Algorithm, will be used to find a stable matching in this situation. Suppose that hospital H_i has a quota of q_i residents. As opposed to engagements in the GS Algorithm, the NRMP Algorithm uses waiting lists to commit temporarily to a resident. When the algorithm terminates, the waiting lists become internships.

As with the GS Algorithm, each stage of the NRMP Algorithm will be a two-step process where all free (not on waiting list) residents apply to the highest hospital

Table 12.8: The NRMP Algorithm as Applied to the Preference Lists of Table 12.7

Stage 1		Stage 2		Stage 3		Stage 4	
Apps.	Wait Lists: H_a H_b	Apps.	Wait Lists: H_a H_b	Apps.	Wait Lists: H_a H_b	Apps.	Wait Lists: H_a H_b
$r_1 - H_a$	r_3 r_2	$r_1 - H_b$	r_3 r_1	$r_1 -$	r_2 r_1	$r_1 -$	r_2 r_1
$r_2 - H_b$	r_4	$r_2 -$	r_4 r_5	$r_2 - H_a$	r_3 r_5	$r_2 -$	r_3 r_5
$r_3 - H_a$		$r_3 -$		$r_3 -$		$r_3 -$	
$r_4 - H_a$		$r_4 -$		$r_4 -$		$r_4 - H_b$	
$r_5 - H_a$		$r_5 - H_b$		$r_5 -$		$r_5 -$	

on their preference list to which they haven't yet applied. Then each hospital compares its newest applicant(s) to those already on its waiting list and chooses, up to its quota and based on its preference list, a revised waiting list. The algorithm terminates when each applicant is either on a waiting list or has been rejected by every hospital on its preference list. (A proof of stability, similar to the one showing that the GS Algorithm produces a stable matching, is left to the exercises.)

For example, suppose that two hospitals, H_a and H_b, each has a quota of two residents to accept into their internship programs and there are five residents, $r_1, r_2, \ldots, r_5$. The residents and hospitals have the preference lists shown in Table 12.7. (Note that one resident will not be matched, which was not possible in the stable marriage problem with equal-sized lists of men and women.) The stages of the NRMP Algorithm are shown in Table 12.8. Since hospital H_a was matched with its two highest preferred residents, it could not be part of a blocking pair. Therefore, in this case, the results of the NRMP Algorithm produced a stable matching.

Besides the differences mentioned above, the NRMP Algorithm also includes the possibility that not every hospital is on everyone's preference list. With a slight modification, it is also possible to allow each hospital to supply a preference list of only its applicants. ■

Both the stable marriage problem and internship matching have in common the fact that the (stable) matching is taking place between two distinct groups of objects: men and women, residents and hospitals. When colleges, for example, are assigning roommates, they are working with a single group of objects (men or women). The question of finding stable matchings for roommates is due to Gale and Shapley [1962].

Example 12.21 Stable Roommates Consider the case that four people need to share two rooms—two to a room. Each person has a strict preference ranking of the other three people. Room assignments must be made so that there is stability in the assignment. That is, based on their preference lists, two people would not do better by leaving their respective roommates to room together. Suppose that their

preference lists are as follows:

$\underline{p_1}$	$\underline{p_2}$	$\underline{p_3}$	$\underline{p_4}$
p_4	p_1	p_2	p_2
p_2	p_3	p_4	p_1
p_3	p_4	p_1	p_3

If the pairings were $R = \{p_1 - p_2, p_3 - p_4\}$, then p_1 and p_4 would each be willing to leave their respective roommates to room with each other; p_1 is higher on p_4's list than p_3 and p_4 is higher on p_1's list than p_2. Thus, R is not a stable matching. However, $S = \{p_1 - p_4, p_2 - p_3\}$ is easily seen to be stable. ■

The stable roommates problem was one of 12 open problems stated by Donald Knuth in 1976; see Knuth [1997]. He asked: Each person from a set of $2n$ persons rates the other $2n - 1$ people according to a certain order of preference. Find an efficient algorithm permitting us to obtain, if possible, a stable set of n couples. An answer to Knuth's question was given by Irving [1985], who found a polynomial-time algorithm that produced a stable matching, if one existed. Unlike the stable marriage problem, stability is not always possible in the stable roommates problem. Exercise 11 presents such an example. Other references to the stable roommates problem include Subramanian [1994] and Feder [1992].

Stability has found its way into many diverse areas. Besides internships and roommates, some connections exist between stable matchings and auctions, computer science (e.g., hashing and data structures), mathematics (e.g., coloring and shortest path problems), and coupon collecting, among others. (See Galvin [1995], Knuth [1974], Knuth [1997], Roth and Sotomayor [1990], Spira [1973], and Ullman [1972].)

EXERCISES FOR SECTION 12.8

1. How many different stable marriage problems of size n are possible? Explain.
2. In a stable marriage problem of size n, how many possible matchings exist? Explain.
3. Find all of the stable matchings for the stable marriage problem of size 4 given in Table 12.5.
4. Show that $M = \{m_1 - w_3, m_2 - w_2, m_3 - w_1, m_4 - w_4\}$ is not a stable matching for the stable marriage problem of size 4 given in Table 12.4.
5. (a) Apply the GS Algorithm to the stable marriage problem of Table 12.9.
   (b) Apply a "woman proposing" GS Algorithm to the stable marriage problem of Table 12.9.
   (c) Explain the significance of the results of parts (a) and (b).
6. Produce a stable marriage problem of size 3 that has exactly one stable matching.
7. Write an algorithm for testing stability of a matching for a stable marriage problem of size n.

Table 12.9: A Stable Marriage Problem of Size 4

Men's Preferences				Women's Preferences			
$\underline{m_1}$	$\underline{m_2}$	$\underline{m_3}$	$\underline{m_4}$	$\underline{w_1}$	$\underline{w_2}$	$\underline{w_3}$	$\underline{w_4}$
w_1	w_2	w_3	w_4	m_4	m_3	m_2	m_1
w_2	w_1	w_4	w_3	m_3	m_4	m_1	m_2
w_3	w_4	w_1	w_2	m_2	m_1	m_4	m_3
w_4	w_3	w_2	w_1	m_1	m_2	m_3	m_4

Table 12.10: Roommate Preference Lists.

$\underline{p_1}$	$\underline{p_2}$	$\underline{p_3}$	$\underline{p_4}$
p_2	p_3	p_1	p_1
p_3	p_1	p_2	p_2
p_4	p_4	p_4	p_3

8. This exercise finishes the proof of Theorem 12.11.
   (a) Show that the marriages defined in (12.14) form a matching.
   (b) Show that the matchings are all different.
   (c) Show that the matching resulting from the marriages defined in (12.14) is stable.
9. For any given stable marriage problem, prove that the stable matchings form a lattice under either the man-oriented or woman-oriented dominance relation.
10. This exercise produces a proof of Theorem 12.13 by contradiction. Let M_{GS} and M be two stable matchings in the man-oriented lattice of a stable marriage problem of size n. Suppose that man m is married to w_i and w_j in M_{GS} and M, respectively, and that m prefers w_j to w_i.
    (a) Explain why w_j must have broken her engagement to m, in favor of, say, m', in the GS Algorithm.
    (b) Suppose, without loss of generality, that w_j's choice of m' was the first such broken engagement to a stable partner in some stable matching during the GS Algorithm. Of all of m''s partners in all the stable matchings, show that m' prefers w_j.
    (c) Explain why m' and w_j are a blocking pair in the stable matching M. (This would then be the contradiction.)
11. Consider the roommates problem of Table 12.10.
    (a) Show that there does not exist a stable matching.
    (b) If person p_4 changed his or her preference list, could a stable matching be found?

REFERENCES FOR CHAPTER 12

AHUJA, R. K., MAGNANTI, T. L., and ORLIN, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

BALINSKI, M., and RATIER, G., "Of Stable Marriages and Graphs, and Strategy and Polytopes," *SIAM Rev., 39* (1997), 575–604.

BERGE, C., "Two Theorems in Graph Theory," *Proc. Natl. Acad. Sci. USA, 43* (1957), 842–844.

BERGE, C., *Graphs and Hypergraphs*, American Elsevier, New York, 1973.

BROGAN, W. L., "Algorithm for Ranked Assignments with Application to Multiobject Tracking," *J. Guidance,* (1989), 357–364.

CHRISTOFIDES, N., *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975.

COOK, W. J., CUNNINGHAM, W. H., PULLEYBLANK, W. R., and SCHRIJVER, A., *Combinatorial Optimization*, Wiley, New York, 1998.

DEO, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1974.

DERMAN, C., and KLEIN, M., "A Note on the Optimal Depletion of Inventory," *Management Sci., 5* (1959), 210–214.

DEVINE, M. V., "A Model for Minimizing the Cost of Drilling Dual Completion Oil Wells," *Management Sci., 20* (1973), 532–535.

EDMONDS, J., "Paths, Trees and Flowers," *Canad. J. Math., 17* (1965), 449–467. (a)

EDMONDS, J., "The Chinese Postman Problem," *Oper. Res., 13, S1* (1965), 373. (b)

EDMONDS, J., and JOHNSON, E. L., "Matching, Euler Tours and the Chinese Postman," *Math. Program., 5* (1973), 88–124.

EGERVÁRY, E., "Matrixok Kombinatóricus Tuladjdonságairól," *Mat. Fiz. Lapok, 38* (1931), 16–28. ("On Combinatorial Properties of Matrices," translated by H. W. Kuhn, Office of Naval Research Logistics Project Report, Department of Mathematics, Princeton University, Princeton, NJ, 1953.)

FEDER, T., "A New Fixed Point Approach for Stable Networks and Stable Marriages," *J. Comput. System Sci., 45* (1992), 233–284.

FIALA, J., KRATOCHVÍL, J., and PROSKUROWSKI, A., "Distance Constrained Labeling of Precolored Trees," in A. Restivo, S. Ronchi Della Rocca, and L. Roversi (eds.), *Theoretical Computer Science: Proceedings of the 7th Italian Conference, ICTCS 2001, Torino, Italy, October 2001*, Lecture Notes in Computer Science, 2202, Springer-Verlag, Berlin, 2001, 285–292.

FORD, L. R., and FULKERSON, D. R., *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

GALE, D., and SHAPLEY, L. S., "College Admissions and the Stability of Marriage," *Amer. Math. Monthly, 69* (1962), 9–15.

GALLAI, T., "Über Extreme Punkt- und Kantenmengen," *Ann. Univ. Sci. Budap., Eötvös Sect. Math., 2* (1959), 133–138.

GALVIN, F., "The List Chromatic Index of a Bipartite Multigraph," *J. Combin. Theory, Ser. B, 63* (1995), 153–158.

GONDRAN, M., and MINOUX, M., *Graphs and Algorithms*, Wiley, New York, 1984.

GRÖTSCHEL, M., LOVÁSZ, L., and SCHRIJVER, A., *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., Springer-Verlag, Berlin, 1993.

GUSFIELD, D., and IRVING, R. W., *The Stable Marriage Problem: Structure and Algorithms*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1989.

HALL, P., "On Representatives of Subsets," *J. Lond. Math. Soc., 10* (1935), 26–30.

HOPCROFT, J. E., and KARP, R. M., "A $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," *SIAM J. Comput., 2* (1973), 225–231.

IRVING, R. W., "An Efficient Algorithm for the Stable Roommates Problem," *J. Alg., 6* (1985), 577–595.

IRVING, R. W., and LEATHER, P., "The Complexity of Counting Stable Marriages," *SIAM J. Comput., 15* (1986), 655–667.

KNUTH, D. E., "Structured Programming with **go to** Statements," *Comput. Surveys, 6* (1974), 261–301.

KNUTH, D. E., *Stable Marriage and Its Relation to Other Combinatorial Problems*, CRM Proceedings & Lecture Notes, 10, American Mathematical Society, Providence, RI, 1997. (Translated by M. Goldstein with revisions by D. Knuth from *Mariages Stables et Leurs Relations Avec d'Autres Problèmes Combinatoires*, Collection de la Chaire Aisenstadt, Les Presses de l'Université de Montréal, Montreal, Canada, 1976.)

KOLITZ, S., personal communication to R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, 1991.

KÖNIG, D., "Graphen und Matrizen," *Mat. Fiz. Lapok, 38* (1931), 116–119.

KUHN, H. W., "The Hungarian Method for the Assignment Problem," *Naval Res. Logist. Quart., 2* (1955), 83–97.

LAWLER, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

LOVÁSZ, L., and PLUMMER, M. D., *Matching Theory*, North-Holland, Amsterdam, 1986.

MASON, A. J., and PHILPOTT, A. B., "Pairing Stereo Speakers Using Matching Algorithms," *Asia-Pacific J. Oper. Res., 5* (1988), 101–116.

MCVITIE, D. G., and WILSON, L. B., "The Stable Marriage Problem," *Comm. ACM, 14* (1971), 486–490.

MICALI, S., and VAZIRANI, V. V., "An $O(\sqrt{|V|} \cdot |E|)$ Algorithm for Finding Maximum Matching in General Graphs," *Proceedings of the Twenty-First Annual Symposium on the Foundations of Computer Science*, IEEE, Long Beach, CA, 1980, 17–27.

MINIEKA, E., *Optimization Algorithms for Networks and Graphs*, Dekker, New York, 1978.

NORMAN, R. Z., and RABIN, M. O., "An Algorithm for a Minimum Cover of a Graph," *Proc. Amer. Math. Soc., 10* (1959), 315–319.

PAPADIMITRIOU, C. H., and STEIGLITZ, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

PETERSON, J., "Die Theorie der Regulären Graphen," *Acta Math., 15* (1891), 193–220.

ROTH, A. E., "The Economics of Matching: Stability and Incentives," *Math. Oper. Res., 7* (1982), 617–628.

ROTH, A. E., and SOTOMAYOR, M. A. O., *Two-Sided Matching*, Cambridge University Press, Cambridge, MA, 1990.

SPIRA, P. M., "A New Algorithm for Finding All Shortest Paths in a Graph of Positive Arcs in Average Time $O(n^2 \log^2 n)$," *SIAM J. Comput., 2* (1973), 28–32.

SUBRAMANIAN, A., "A New Approach to Stable Matching Problems," *SIAM J. Comput., 23* (1994), 671–700.

TUCKER, A. C., *Applied Combinatorics*, Wiley, New York, 1980.

TUTTE, W. T., "The Factorization of Linear Graphs," *J. Lond. Math. Soc., 22* (1947), 107–111.

ULLMAN, J. D., "A Note on the Efficiency of Hashing Functions," *J. Assoc. Comput. Mach., 19* (1972), 569–575.