This file contains the exercises, hints, and solutions for Chapter 10 of the book "Introduction to the Design and Analysis of Algorithms," 3rd edition, by A. Levitin. The problems that might be challenging for at least some students are marked by ▷; those that might be difficult for a majority of students are marked by ►.

Exercises 10.1

1. Consider the following version of the post office location problem: (Problem 3 in Exercises 3.3): Given n integers x_1, x_2, ..., x_n representing coordinates of n villages located along a straight road, find a location for a post office that minimizes the average distance between the villages. The post office may be, but is not required to be, located at one of the villages. Devise an interative-improvement algorithm for this problem. Is this an efficient way to solve this problem?

2. Solve the following linear programming problems geometrically.

   (a)
   $$\begin{aligned} \text{maximize} \quad & 3x + y \\ \text{subject to} \quad & -x + y \le 1 \\ & 2x + y \le 4 \\ & x \ge 0, \;\; y \ge 0 \end{aligned}$$

   (b)
   $$\begin{aligned} \text{maximize} \quad & x + 2y \\ \text{subject to} \quad & 4x \ge y \\ & y \le 3 + x \\ & x \ge 0, \;\; y \ge 0 \end{aligned}$$

3. Consider the linear programming problem
   $$\begin{aligned} \text{minimize} \quad & c_1 x + c_2 y \\ \text{subject to} \quad & x + y \ge 4 \\ & x + 3y \ge 6 \\ & x \ge 0, \;\; y \ge 0 \end{aligned}$$
   where c_1 and c_2 are some real numbers not both equal to zero.

   (a) Give an example of the coefficient values c_1 and c_2 for which the problem has a unique optimal solution.

   (b) Give an example of the coefficient values c_1 and c_2 for which the problem has infinitely many optimal solutions.

   (c) Give an example of the coefficient values c_1 and c_2 for which the problem does not have an optimal solution.

4. Would the solution to problem (10.2) be different if its inequality constraints were strict, i.e., $x + y < 4$ and $x + 3y < 6$, respectively?

5. Trace the simplex method on

   (a) the problem of Exercise 2a.

   (b) the problem of Exercise 2b.

6. Trace the simplex method on the problem of Example 1 in Section 6.6

   (a) ▷ by hand.

   (b) by using one of the implementations available on the Internet.

7. Determine how many iterations the simplex method needs to solve the problem

$$\begin{array}{rl} \text{maximize} & \sum_{j=1}^{n} x_j \\ \text{subject to} & 0 \le x_j \le b_j, \ \text{ where } b_j > 0 \text{ for } j = 1, 2, ..., n. \end{array}$$

8. Can we apply the simplex method to solve the knapsack problem (see Example 2 in Section 6.6)? If you answer yes, indicate whether it is a good algorithm for the problem in question; if you answer no, explain why not.

9. ▷ Prove that no linear programming problem can have exactly $k \ge 1$ optimal solutions unless $k = 1$.

10. If a linear programming problem

$$\begin{array}{rl} \text{maximize} & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \le b_i \ \text{ for } i = 1, 2, ..., m \\ & x_1, x_2, ..., x_n \ge 0 \end{array}$$

    is considered as ***primal***, then its ***dual*** is defined as the linear programming problem

$$\begin{array}{rl} \text{minimize} & \sum_{i=1}^{m} b_i y_i \\ \text{subject to} & \sum_{i=1}^{m} a_{ij} y_i \ge c_j \ \text{ for } j = 1, 2, ..., n \\ & y_1, y_2, ..., y_m \ge 0. \end{array}$$

    (a) Express the primal and dual problems in matrix notations.

    (b) Find the dual of the linear programming problem

$$\begin{array}{rl} \text{maximize} & x_1 + 4x_2 - x_3 \\ \text{subject to} & x_1 + x_2 + x_3 \le 6 \\ & x_1 - x_2 - 2x_3 \le 2 \\ & x_1, x_2, x_3 \ge 0. \end{array}$$

    (c) Solve the primal and dual problems and compare the optimal values of their objective functions.

Hints to Exercises 10.1

1. Start at an arbitrary integer point x and investigate whether a neighboring point is a better location for the post office than x is.

2. Sketch the feasible region of the problem in question. Follow this up by either applying the Extreme-Point Theorem or by inspecting level lines, whichever is more appropriate. Both methods were illustrated in the text.

3. Sketch the feasible region of the problem. Then choose values of the parameters c_1 and c_2 to obtain a desired behavior of the objective function's level lines.

4. What is the principal difference between maximizing a linear function, say, $f(x) = 2x$, on a closed vs. semi-open interval, e.g., $0 \le x \le 1$ vs. $0 \le x < 1$?

5. Trace the simplex method on the instances given, as was done for an example in the text.

6. When solving the problem by hand, you might want to start by getting rid of fractional coefficients in the problem's statement. Also, note that the problem's specifics make it possible to replace its equality constraint by one inequality constraint. You were asked to solve this problem directly in Exercises 6.6 (see Problem 8).

7. The specifics of the problem make it possible to see the optimal solution at once. Sketching its feasible region for $n = 2$ or $n = 3$, though not necessary, may help to see both this solution and the number of iterations needed by the simplex method to solve it.

8. Consider separately two versions of the problem: continuous and 0-1 (see Example 2 in Section 6.6).

9. If $x' = (x'_1, x'_2, ..., x'_n)$ and $x'' = (x''_1, x''_2, ..., x''_n)$ are two distinct optimal solutions to the same linear programming problem, what can we say about any point of the line segment with the endpoints at x' and x''? Note that any such point x can be expressed as $x = tx' + (1-t)x'' = (tx'_1 + (1-t)x''_1, tx'_2 + (1-t)x''_2, ..., tx'_n + (1-t)x''_n)$ where $0 \le t \le 1$.

10. a. You will need to use the notion of a matrix transpose, defined as the matrix whose rows are the columns of the given matrix.
    b. Apply the general definition to the specific problem given. Note the change from maximization to minimization, the change of the roles played by objective function's coefficients and constraints' right-hand sides, the transposition of the constraints and the reversal of their signs.
    c. You may use either the simplex method or the geometric approach.

Solutions to Exercises 10.1

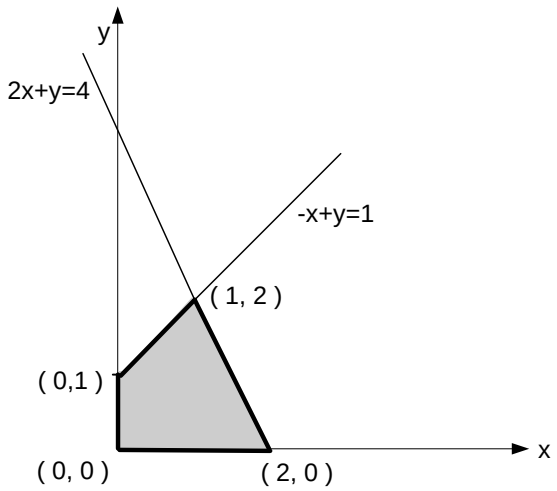
1. The problem is to find a value of x that minimizes $f(x) = \frac{1}{n}\sum_{i=1}^{n} |x - x_i|$, which can be simplified to minimizing $g(x) = \sum_{i=1}^{n} |x - x_i|$ since $\frac{1}{n}$ is a constant. Here is an iterative improvement algorithm that does this. Select an arbitrary integer location x and compute $g(x)$, then compute $g(x-1)$ and $g(x+1)$. If $g(x-1) < g(x)$ or $g(x+1) < g(x)$, replace x by $x-1$or $x+1$, respectively, and repeat this step with the new value of x; otherwise, return the current value of x as the post office location. Note that since the functions $f(x)$ and $g(x)$ are piecewise linear and convex, the algorithm will stop after a finite number of iterations for any initial integer value of x.

   The solution to the problem is the median of x_1, x_2, ..., x_n, which can be found either by sorting the points given or by a partition-based algorithm (see Section 4.5). Although the iterative improvement algorithm can find the solution faster if the initial value of x happens to be close to the median, it can hardly be considered competitive with either the presorting-based algorithm or quickselect. Also note that the latter work for noninteger input, whereas the iterative improvement algorithm does not.

2. a. The feasible region of the linear programming problem

$$\begin{array}{rl} \text{maximize} & 3x + y \\ \text{subject to} & -x + y \le 1 \\ & 2x + y \le 4 \\ & x \ge 0, \;\; y \ge 0 \end{array}$$

   is given in the following figure:



   Since the feasible region of the problem is nonempty and bounded, an

optimal solution exists and can be found at one of the extreme points of the feasible region. The extreme points and the corresponding values of the objective function are given in the following table:

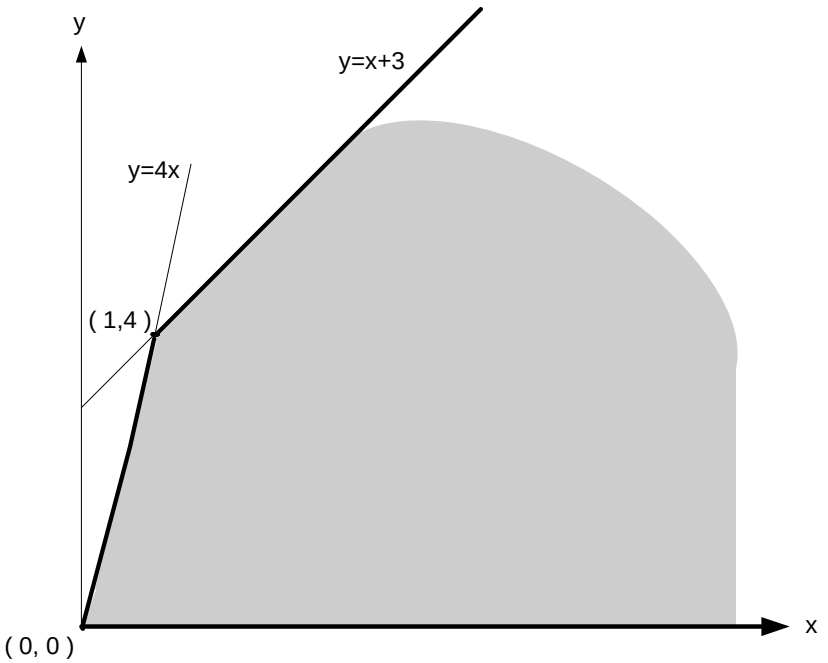
Extreme point	Value of $3x + y$
$(0, 0)$	0
$(2, 0)$	6
$(1, 2)$	5
$(0, 1)$	1

Hence, the optimal solution is $x = 2$, $y = 0$, with the corresponding value of the objective function equal to 6.

b. The feasible region of the linear programming problem

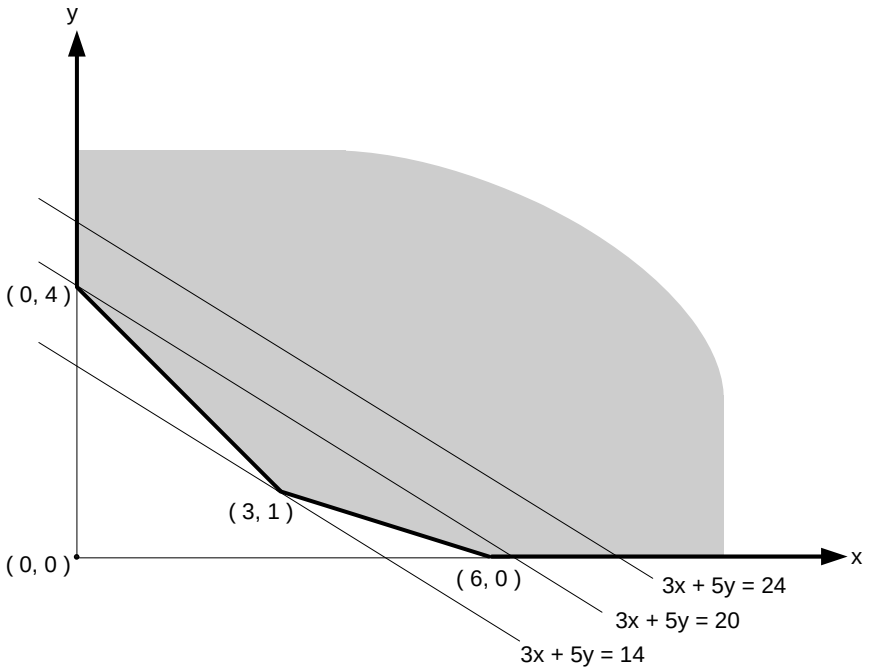
$$\begin{array}{rl} \text{maximize} & x + 2y \\ \text{subject to} & 4x \geq y \\ & x + 3 \geq y \\ & x \geq 0, \;\; y \geq 0 \end{array}$$

is given in the following figure:



The feasible region is unbounded. On considering a few level lines $x+2y = z$, it is easy to see that level lines can be moved in the north-east direction, which corresponds to increasing values of z, as far as we wish without losing common points with the feasible region. (Alternatively, we can consider a simple sequence of points, such as $(n, 0)$, which are feasible for any nonnegative integer value of n and make the objective function $z = x + 2y$ as large as we wish as n goes to infinity.) Hence, the problem is unbounded and therefore does not have a finite optimal solution.

3. The feasible region of the problem with the constraints $x + y \geq 4$, $x + 3y \geq 6$, $x \geq 0$, $y \geq 0$ is given in Figure 10.3 of Section 10.1:



a. Minimization of $z = c_1x + c_2y$ on this feasible region, with $c_1, c_2 > 0$, pushes level lines $c_1x + c_2y = z$ in the south-west direction. Any family of such lines with a slope strictly between the slopes of the boundary lines $x + y = 4$ and $x + 3y = 6$ will hit the extreme point $(3, 1)$ as the only minimum solution to the problem. For example, as mentioned in Section 10.1, we can minimize $3x + 5y$ among infinitely many other possible answers.

b. For a linear programming problem to have infinitely many solutions, the optimal level line of its objective function must contain a line segment that is a part of the feasible region's boundary. Hence, there are four qualitatively distinct answers:

i. minimize $1 \cdot x + 0 \cdot y$ (or, more generally, any objective function of the form $c \cdot x + 0 \cdot y$, where $c > 0$);

ii. minimize $x + y$ (or, more generally, any objective function of the form $cx + cy$, where $c > 0$);

iii. minimize $x + 3y$ (or, more generally, any objective function of the form $cx + 3cy$, where $c > 0$);

iv. minimize $0 \cdot x + 1 \cdot y$ (or, more generally, any objective function of the form $0 \cdot x + c \cdot y$, where $c > 0$).

c. Among infinitely many examples, there are the following: $-1 \cdot x + -1 \cdot y$, $-1 \cdot x + 0 \cdot y$, $0 \cdot x - 1 \cdot y$.

4. The problem with the strict inequalities does not have an optimal solution: The objective function values of any sequence of feasible points approaching $x_1 = 3$, $x_2 = 1$ (the optimal solution to Problem (10.2) in the text) will approach $z = 14$ as its limit, but this value will not be attained at any feasible point of the problem with the strict inequalities.

5. a. The standard form of the problem given is

$$\begin{array}{ll} \text{maximize} & 3x + y \\ \text{subject to} & -x + y + u \qquad = 1 \\ & 2x + y + \quad +v = 4 \\ & x, y, u, v \geq 0. \end{array}$$

Here are the tableaux generated by the simplex method in solving this problem:

	x	y	u	v	
u	−1	1	1	0	1
← v	2	1	0	1	4
	−3	−1	0	0	0

$\theta_v = \frac{4}{2}$

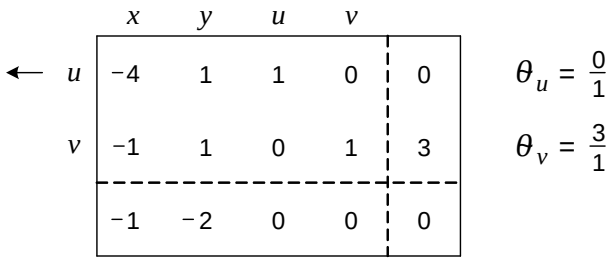
	x	y	u	v	
u	0	$\frac{3}{2}$	1	$\frac{1}{2}$	3
x	1	$\frac{1}{2}$	0	$\frac{1}{2}$	2
	0	$\frac{1}{2}$	0	$\frac{3}{2}$	6

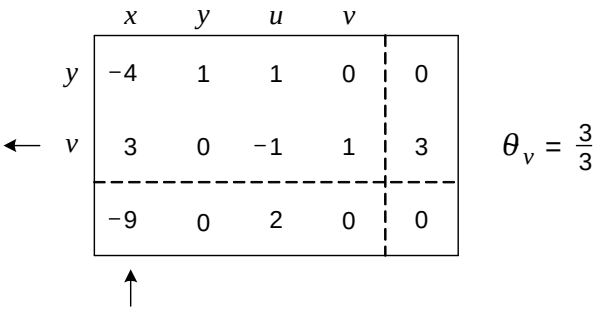
The optimal solution found is $x = 2$, $y = 0$, with the maximal value of the objective function equal to 6.

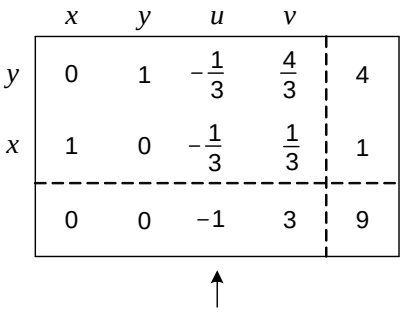
b. The standard form of the problem given is

$$\begin{array}{ll} \text{maximize} & x + 2y \\ \text{subject to} & -4x + y + u \qquad = 0 \\ & -x + y + \quad +v = 3 \\ & x, y, u, v \geq 0. \end{array}$$

Here are the tableaux generated by the simplex method in solving this problem:







Since there are no positive elements in the pivot column of the last tableau, the problem is unbounded.

6. a. To simplify the task of getting an initial basic feasible solution here, we can replace the equality constraint $x + y + z = 100$ by the inequality $x + y + z \le 100$, because an optimal solution (x^*, y^*, z^*) to the problem with the latter constraint must satisfy the former one.. (Otherwise, we would've been able to increase the maximal value of the objective function by increasing the value of z^*.) then, after an optional replacement of the objective function and the constraints by the equivalent ones without fractional coefficients, the problem can be presented in the standard form as follows

$$\begin{aligned} \text{maximize} \quad & 10x + 7y + 3z \\ \text{subject to} \quad & x + y + z + u = 100 \\ & 3x - y + \quad + v \quad = 0 \\ & x + y - 4z + \quad + w = 0 \\ & x, y, z, u, v, w \ge 0. \end{aligned}$$

Here are the tableaux generated by the simplex method in solving this problem:

		x	y	z	u	v	w		
	u	1	1	1	1	0	0	100	$\theta_u = \frac{100}{1}$
←	v	3	−1	0	0	1	0	0	$\theta_v = \frac{0}{3}$
	w	1	1	−4	0	0	1	0	$\theta_w = \frac{0}{1}$
		−10	−7	−3	0	0	0	0	

↑

		x	y	z	u	v	w		
	u	0	$\frac{4}{3}$	1	1	$-\frac{1}{3}$	0	100	$\theta_u = \frac{100}{4/3}$
	x	1	$-\frac{1}{3}$	0	0	$\frac{1}{3}$	0	0	
←	w	0	$\frac{4}{3}$	−4	0	$-\frac{1}{3}$	1	0	$\theta_w = \frac{0}{4/3}$
		0	$-\frac{31}{3}$	−3	0	$\frac{10}{3}$	0	0	

↑

		x	y	z	u	v	w		
←	u	0	0	5	1	0	−1	100	$\theta_u = \frac{100}{5}$
	x	1	0	−1	0	$\frac{1}{4}$	$\frac{1}{4}$	0	
	y	0	1	−3	0	$-\frac{1}{4}$	$\frac{3}{4}$	0	
		0	0	−34	0	$\frac{3}{4}$	$\frac{31}{4}$	0	

↑

	x	y	z	u	v	w	
z	0	0	1	$\frac{1}{5}$	0	$-\frac{1}{5}$	20
x	1	0	0	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{20}$	20
y	0	1	0	$\frac{3}{5}$	$-\frac{1}{4}$	$\frac{3}{20}$	60
	0	0	0	$\frac{34}{5}$	$\frac{3}{4}$	$\frac{19}{20}$	680

The optimal solution found is $x = 20$, $y = 60$, $z = 20$, with the maximal value of the objective function equal to 680.

7. The optimal solution to the problem is $x_1 = b_1, ..., x_n = b_n$. After introducing a slack variable s_i in the ith inequality $x_i = b_i$ to get to the standard form and starting with $x_1 = 0, ..., x_n = 0$, $s_1 = b_1, ..., s_n = b_n$, the simplex method will need n iterations to get to an optimal solution $x_1 = b_1, ..., x_n = b_n, s_1 = 0, ..., s_n = 0$. It follows from the fact that on each iteration the simplex method replaces only one basic variable. Here, on each of its n iterations, it will replace some slack variable s_i by the corresponding x_i.

8. The continuous version of the knapsack problem can be solved by the simplex method, because it is a special case of the general linear programming problem (see Example 2 in Section 6.6). However, it is hardly a good method for solving this problem because it can be solved more efficiently by a much simpler algorithm based on the greedy approach. You may want to design such an algorithm by yourself before looking it up in the book.
   The 0-1 version of the knapsack problem cannot be solved by the simplex method because of the integrality (0-1) constraints imposed on the problem's variables.

9. The assertion follows immediately from the fact that if $x' = (x'_1, ..., x'_n)$ and $x'' = (x''_1, ..., x''_n)$ are two distinct optimal solutions to the same linear programming problem, then any of the infinite number of points of the line segment with endpoints at x' and x'' will be an optimal solution to this problem as well. Indeed, let x^t be such a point:

$$x^t = tx' + (1-t)x'' = (tx'_1 + (1-t)x''_1, ..., tx'_n + (1-t)x''_n), \text{ where } 0 \le t \le 1.$$

First, x^t will satisfy all the constraints of the problem, whether they are linear inequalities or linear equations, because both x' and x'' do. Indeed, let the ith constraint be inequality $\sum_{j=1}^{n} a_{ij}x_j \le b_i$. Then $\sum_{j=1}^{n} a_{ij}x'_j \le b_i$ and $\sum_{j=1}^{n} a_{ij}x''_j \le b_i$. Multiplying these inequalities by t and $1-t$, respectively, and adding the results, we obtain

$$t\sum_{j=1}^{n} a_{ij}x'_j + (1-t)\sum_{j=1}^{n} a_{ij}x''_j \le tb_i + (1-t)b_i$$

or

$$\sum_{j=1}^{n}(ta_{ij}x'_j + (1-t)a_{ij}x''_j) = \sum_{j=1}^{n} a_{ij}(tx'_j + (1-t)x''_j) = \sum_{j=1}^{n} a_{ij}x^t_j \le b_i,$$

i.e., x satisfies the inequality. The same argument holds for inequalities $\sum_{j=1}^{n} a_{ij}x_j \ge b_i$ and equations $\sum_{j=1}^{n} a_{ij}x_j = b_i$.
Second, $x^t = tx' + (1-t)x''$ will maximize the value of the objective function. Indeed, if the maximal value of the objective function is z^*, then

$$\sum_{j=1}^{n} c_j x'_j = z^* \quad \text{and} \quad \sum_{j=1}^{n} c_j x''_j = z^*.$$

Multiplying these equalities by t and $1-t$, respectively, and adding the results, we will obtain

$$t\sum_{j=1}^{n} c_j x'_j + (1-t)\sum_{j=1}^{n} c_j x''_j = tz^* + (1-t)z^*$$

or

$$\sum_{j=1}^{n}(tc_j x'_j + (1-t)c_j x''_j) = \sum_{j=1}^{n} c_j(tx'_j + (1-t)x''_j) = \sum_{j=1}^{n} c_j x^t_j = z^*.$$

I.e., we proved that x^t does maximize the objective function and hence will be an optimal solution to the problem in question for any $0 \le t \le 1$.

Note: What we actually proved is the fact that the set of optimal solutions to a linear programming problem is convex. And any nonempty convex set can contain either a single point or infinitely many points.

10. a. A linear programming problem

$$\begin{array}{rl} \text{maximize} & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{for } i = 1, 2, ..., m \\ & x_1, x_2, ..., x_n \ge 0 \end{array}$$

can be compactly written using the matrix notations as follows:

$$\begin{array}{rl} \text{maximize} & cx \\ \text{subject to} & Ax \le b \\ & x \ge 0, \end{array}$$

where

$$c = \begin{bmatrix} c_1 & ... & c_n \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad A = \begin{bmatrix} a_{11} & ... & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & ... & a_{mn} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix},$$

$Ax \le b$ holds if and only if each coordinate of the product Ax is less than or equal to the corresponding coordinate of vector b, and $x \ge 0$ is shorthand for the nonnegativity requirement for all the variables. The ***dual*** can be written as follows:

$$\begin{array}{rl} \text{minimize} & b^T y \\ \text{subject to} & A^T y \ge c^T \\ & y \ge 0, \end{array}$$

where b^T is the transpose of b (i.e., $b^T = [b_1, ..., b_m]$), c^T is the transpose of c (i.e., c^T is the column-vector made up of the coordinates of the row-vector c), A^T is the transpose of A (i.e., the $n \times m$ matrix whose jth row is the jth column of matrix A, $j = 1, 2, ..., n$), and y is the vector-column of m new unknowns $y_1, ..., y_m$.

b. The dual of the linear programming problem

$$\begin{array}{rl} \text{maximize} & x_1 + 4x_2 - x_3 \\ \text{subject to} & x_1 + x_2 + x_3 \le 6 \\ & x_1 - x_2 - 2x_3 \le 2 \\ & x_1, x_2, x_3 \ge 0 \end{array}$$

is

$$\begin{array}{rl} \text{minimize} & 6y_1 + 2y_2 \\ \text{subject to} & y_1 + y_2 \ge 1 \\ & y_1 - y_2 \ge 4 \\ & y_1 - 2y_2 \ge -1 \\ & y_1, y_2 \ge 0. \end{array}$$

c. The standard form of the primal problem is

$$\begin{array}{ll} \text{maximize} & x_1 + 4x_2 - x_3 \\ \text{subject to} & x_1 + x_2 + x_3 + x_4 \qquad = 6 \\ & x_1 - x_2 - 2x_3 + \qquad x_5 = 2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0. \end{array}$$

The simplex method yields the following tableaux:

	x_1	x_2	x_3	x_4	x_5	
← x_4	1	1	1	1	0	6
x_5	1	−1	−2	0	1	2
	−1	−4	1	0	0	0

↑

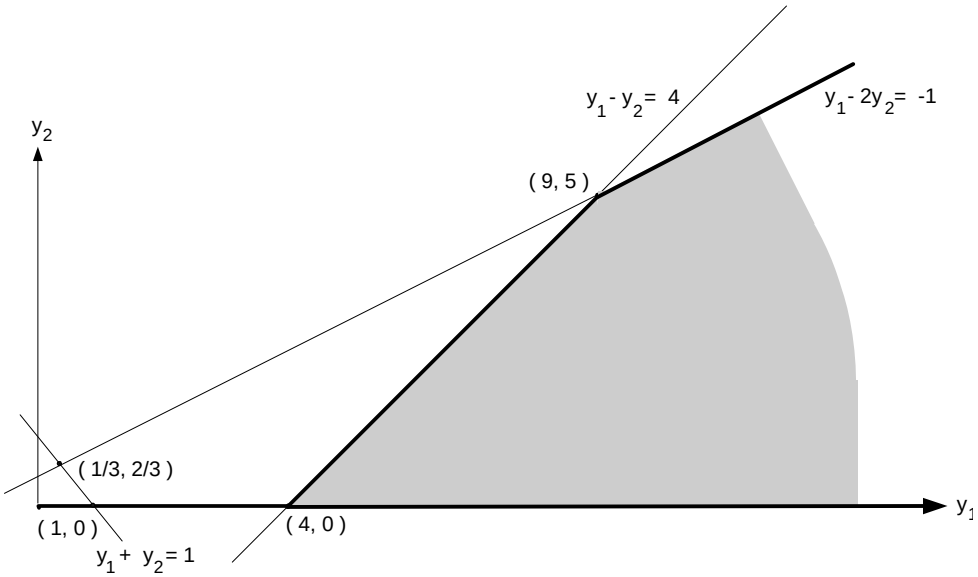
$\theta_{x_4} = \frac{6}{1}$

	x_1	x_2	x_3	x_4	x_5	
x_2	1	1	1	1	0	6
x_5	2	0	−1	1	1	8
	3	0	5	4	0	24

The found optimal solution is $x_1 = 0$, $x_2 = 6$, $x_3 = 0$.

Since the dual problem has just two variables, it is easier to solve it

geometrically. Its feasible region is presented in the following figure:



Although it is unbounded, the minimization problem in question does have a finite optimal solution $y_1 = 4$, $y_2 = 0$. Note that the optimal values of the objective functions in the primal and dual problems are equal to each other:

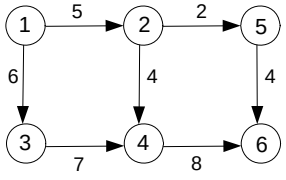
$$0 + 4 \cdot 6 - 0 = 6 \cdot 4 + 2 \cdot 0.$$

This is the principal assertion of the Duality Theorem, one of the most important facts in linear programming theory.

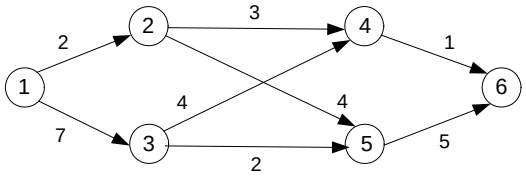
Exercises 10.2

1. Since maximum-flow algorithms require processing edges in both directions, it is convenient to modify the adjacency matrix representation of a network as follows: If there is a directed edge from vertex i to vertex j of capacity u_{ij}, then the element in the ith row and the jth column is set to u_{ij}, and the element in the jth row and the ith column is set to $-u_{ij}$; if there is no edge between vertices i and j, both these elements are set to zero. Outline a simple algorithm for identifying a source and a sink in a network presented by such a matrix and indicate its time efficiency.

2. Apply the shortest-augmenting path algorithm to find a maximum flow and a minimum cut in the following networks:
   a.



   b.



3. a. Does the maximum-flow problem always have a unique solution? Would your answer be different for networks with different capacities on all their edges?

   b. Answer the same questions for the minimum-cut problem of finding a cut of the smallest capacity in a given network.

4. a. Explain how the maximum-flow problem for a network with several sources and sinks can be transformed to the same problem for a network with a single source and a single sink.

   b. Some networks have capacity constraints on flow amounts that can flow through their intermediate vertices. Explain how the maximum-flow problem for such a network can be transformed to the maximum-flow problem for a network with edge capacity constraints only.

5. ▷ Consider a network that is a rooted tree, with the root as its source, the leaves as its sinks, and all the edges directed along the paths from the

root to the leaves. Design an efficient algorithm for finding a maximum flow in such a network What is the time efficiency of your algorithm?

6. ▷ a. Prove equality (10.9).

   ▷ b. Prove that for any flow in a network and any cut in it, the value of the flow is equal to the flow across the cut (see equality (10.12)). Explain the relationship between this property and equality (10.9).

7. a. Express the maximum-flow problem for the network of Figure 10.4 as a linear programming problem.

   b. Solve this linear programming problem by the simplex method.

8. As an alternative to the shortest-augmenting-path algorithm, Edmonds and Karp [Edm72] suggested the maximal-augmenting-path algorithm in which a flow is augmented along the path that increases the flow by the largest amount. Implement both these algorithms in the language of your choice and perform an empirical investigation of their relative efficiency.

9. Write a report on a more advanced maximum-flow algorithm such as (i) Dinitz's algorithm, (ii) Karzanov's algorithm, (iii) Malhotra-Kamar-Maheshwari algorithm, or (iv) Goldberg-Tarjan algorithm.

10. *Dining problem* Several families go out to dinner together. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Show how to find a seating arrangement that meets this objective (or prove that no such arrangement exists) by using a maximum flow problem. Assume that the dinner contingent has p families and that the ith family has a_i members. Also assume that q tables are available and the jth table has a seating capacity of b_j [Ahu93].

Hints to Exercises 10.2

1. What properties of the adjacency matrix elements stem from the source and sink definitions, respectively?

2. See the algorithm and an example illustrating it in the text.

3. Of course, the value (capacity) of an optimal flow (cut) is the same for any optimal solution. The question is whether distinct flows (cuts) can yield the same optimal value.

4. a. Add extra vertices and edges to the network given.

   b. If an intermediate vertex has a constraint on a flow amount that can flow through it, split the vertex into two.

5. Take advantage of the recursive structure of a rooted tree.

6. a. Sum the equations expressing the flow-conservation requirement.

   b. Sum the equations defining the flow value and flow-conservation requirements for the vertices in set X inducing the cut.

7. a. Use template (10.11) given in the text.

   b. Use either an add-on tool of your spreadsheet or some software available on the Internet.

8. n/a

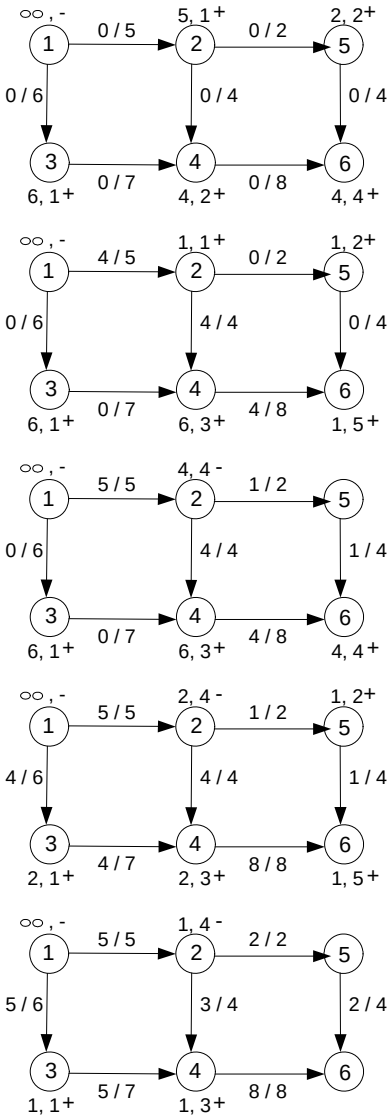
9. n/a

10. Use edge capacities to impose the problem's constraints. Also, take advantage of the solution to Problem 4a.

Solutions to Exercises 10.2

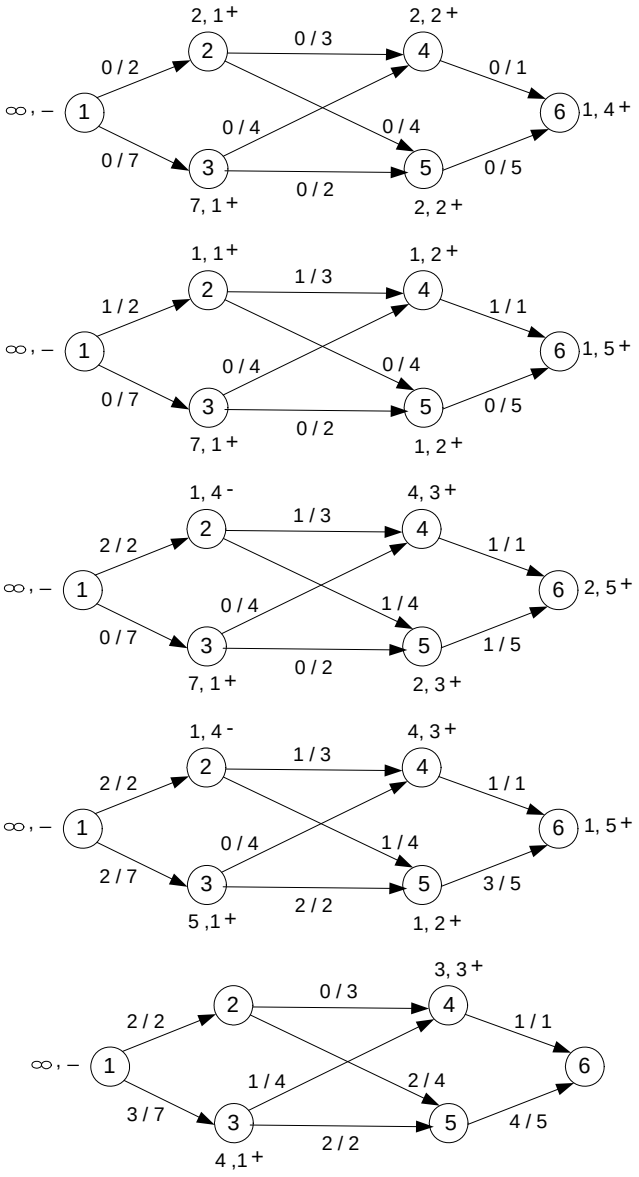
1. The definition of a source implies that a vertex is a source if and only if there are no negative elements in its row in the modified adjacency matrix. Similarly, the definition of a sink implies that a vertex is a sink if and only if there are no positive elements in its row of the modified adjacency matrix. Thus a simple scan of the adjacency matrix rows solves the problem in $O(n^2)$ time, where n is the number of vertices in the network.

2. a. Here is an application of the shortest-augmenting path algorithm to the network of Problem 2a:



The maximum flow is shown on the last diagram above. The minimum cut found is $\{(2,5\},(4,6)\}$.

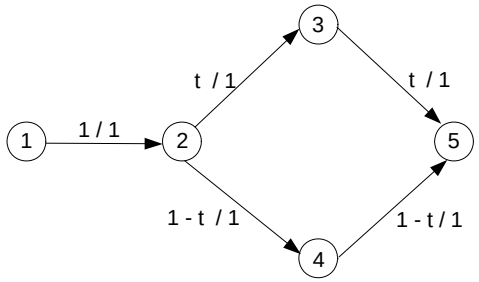
b. Here is an application of the shortest-augmenting path algorithm to the network of Problem 10.2b:



The maximum flow of value 5 is shown on the last diagram above. The

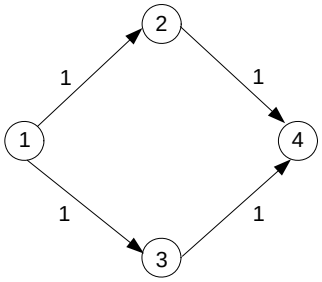
minimum cut found is $\{(1,2),(3,5),(4,6)\}$.

3. a. The maximum-flow problem may have more than one optimal solution. In fact, there may be infinitely many of them if we allow (as the definition does) non-integer edge flows. For example, for any $0 \le t \le 1$, the flow depicted in the diagram below is a maximum flow of value 1. Exactly two of them—for $t = 0$ and $t = 1$—are integer flows.



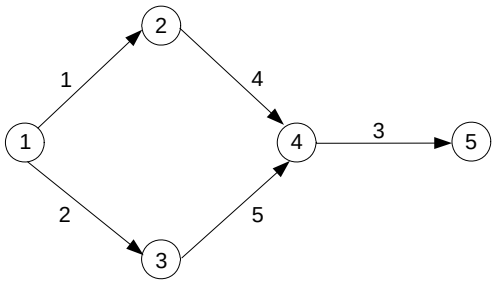
The answer does not change for networks with distinct capacities: e.g., consider the previous example with the capacities of edges $(2,3)$, $(3,5)$, $(2,4)$, and $(4,5)$ changed to, say, $2, 3, 4$, and 5, respectively.

b. The answer for the number of distinct minimum cuts is analogous to that for maximum flows: there can be more than one of them in the same network (though, of course, their number must always be finite because the number of all edge subsets is finite to begin with). For example, the network



has four minimum cuts: $\{(1,2),(1,3)\}$, $\{(1,2),(3,4)\}$, $\{(1,3),(2,4)\}$, and $\{(2,4),(3,4)\}$. The answer does not change for networks with distinct edge

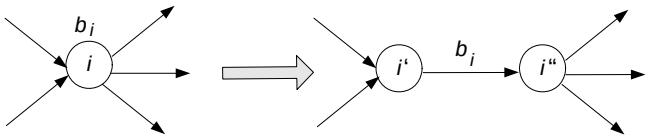
capacities. For example, the network



has two minimum cuts: $\{(1,2),(1,3)\}$ and $\{(4,5)\}$.

4. a. Add two vertices to the network given to serve as the source and sink of the new network, respectively. Connect the new source to each of the original sources and each of the original sinks to the new sink with edges of some large capacity M. (It suffices to take M greater than or equal to the sum of the capacities of the edges leaving each source of the original network.)

   b. Replace each intermediate vertex i with an upper bound b_i on a flow amount that can flow through it with two vertices i' and i'' connected by an edge of capacity b_i as shown below:



   Note that all the edges entering and leaving i in the original network should enter i' and leave i'', respectively, in the new one.

5. The problem can be solved by calling *TreeFlow*$(T(r), \infty)$, where
   Algorithm *TreeFlow*$(T(r), v)$
   //Finds a maximum flow for tree $T(r)$ rooted at r,
   //whose value doesn't exceed v (available at the root),
   //and returns its value
   if r is a leaf *maxflowval* $\leftarrow v$
   else
   maxflowval $\leftarrow 0$
   for every child c of r **do**
   $x_{rc} \leftarrow$ *TreeFlow*$(T(c), \min\{u_{rc}, v\})$
   $v \leftarrow v - x_{rc}$
   maxflowval $\leftarrow$ *maxflowval* $+ x_{rc}$
   return *maxflowval*

The efficiency of the algorithm is clearly linear because a $\Theta(1)$ call is made for each of the nodes of the tree.

6. a. Adding the $n-2$ equalities expressing the flow conservation requirements yields

$$\sum_{i=2}^{n-1} \sum_{j} x_{ji} = \sum_{i=2}^{n-1} \sum_{j} x_{ij}.$$

For any edge from an intermediate vertex p to an intermediate vertex q $(2 \le p, q \le n-1)$, the edge flow x_{pq} occurs once in both the left- and right-hand sides of the last equality and hence will cancel out. The remaining terms yield:

$$\sum_{i=2}^{n-1} x_{1i} = \sum_{i=2}^{n-1} x_{in}.$$

Adding x_{1n} to both sides of the last equation, if there is an edge from source 1 to sink n, results in the desired equality:

$$\sum_{i} x_{1i} = \sum_{i} x_{in}.$$

b. Summing up the flow-value definition $v = \sum_{j} x_{1j}$ and the flow-conservation requirement $\sum_{j} x_{ji} = \sum_{j} x_{ij}$ for every $i \in X$ $(i > 1)$, we obtain

$$v + \sum_{i:\ i \in X,\ i>1} \sum_{j} x_{ji} = \sum_{j} x_{1j} + \sum_{i:\ i \in X,\ i>1} \sum_{j} x_{ij},$$

or, since there are no edges entering the source,

$$v + \sum_{i \in X} \sum_{j} x_{ji} = \sum_{i \in X} \sum_{j} x_{ij}.$$

Moving the summation from the left-hand side to the right-hand side and splitting the sum into the sum over the vertices in X and the sum over the vertices in $\bar{X}$, we obtain:

$$\begin{aligned} v &= \sum_{i \in X} \sum_{j} x_{ij} - \sum_{i \in X} \sum_{j} x_{ji} \\ &= \sum_{i \in X} \sum_{j \in X} x_{ij} + \sum_{i \in X} \sum_{j \in \bar{X}} x_{ij} - \sum_{i \in X} \sum_{j \in X} x_{ji} - \sum_{i \in X} \sum_{j \in \bar{X}} x_{ji} \\ &= \sum_{i \in X} \sum_{j \in \bar{X}} x_{ij} - \sum_{i \in X} \sum_{j \in \bar{X}} x_{ji}, \qquad \text{Q.E.D.} \end{aligned}$$

Note that equation (10.9) expresses this general property for two special cuts: $C_1(X_1, \bar{X}_1)$ induced by $X_1 = \{1\}$ and $C_2(X_2, \bar{X}_2)$ induced by $X_2 = V - \{n\}$.

7. a.

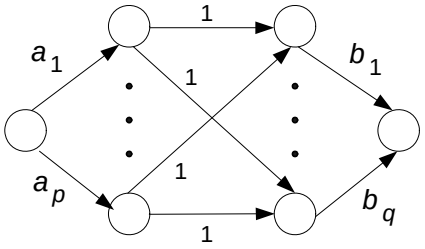
$$\begin{aligned} \text{maximize} \quad & v = x_{12} + x_{14} \\ \\ \text{subject to} \quad & x_{12} - x_{23} - x_{25} = 0 \\ & x_{23} + x_{43} - x_{36} = 0 \\ & x_{14} - x_{43} = 0 \\ & x_{25} - x_{56} = 0 \\ & 0 \le x_{12} \le 2 \\ & 0 \le x_{14} \le 3 \\ & 0 \le x_{23} \le 5 \\ & 0 \le x_{25} \le 3 \\ & 0 \le x_{36} \le 2 \\ & 0 \le x_{43} \le 1 \\ & 0 \le x_{56} \le 4. \end{aligned}$$

b. The optimal solution is $x_{12} = 2$, $x_{14} = 1$, $x_{23} = 1$, $x_{25} = 1$, $x_{36} = 2$, $x_{43} = 1$, $x_{56} = 1$.

8. n/a

9. n/a

10. Solve the maximum flow problem for the following network:

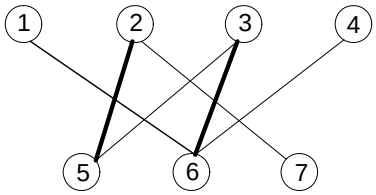


If the maximum flow value is equal to $\sum_{i=1}^{p} a_i$, then the problem has a solution indicated by the full edges of capacity 1 in the maximum flow; otherwise, the problem does not have a solution.

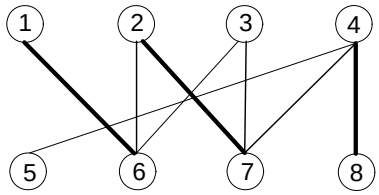
Exercises 10.3

1. For each matching shown below in bold, find an augmentation or explain why no augmentation exists.

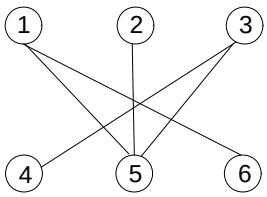
a.



b.



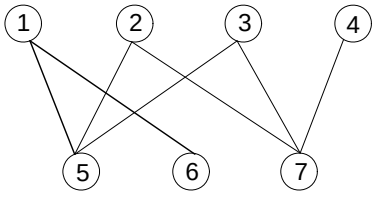
2. Apply the maximum-matching algorithm to the following bipartite graph:



3. a. What is the largest and what is the smallest possible cardinality of a matching in a bipartite graph $G = \langle V, U, E \rangle$ with n vertices in each vertex set V and U and at least n edges?

   b. What is the largest and what is the smallest number of distinct solutions the maximum-cardinality matching problem can have for a bipartite graph $G = \langle V, U, E \rangle$ with n vertices in each vertex set V and U and at least n edges?

4. a. ***Hall's Marriage Theorem*** asserts that a bipartite graph $G = \langle V, U, E \rangle$ has a matching that matches all vertices of the set V if and only if for each subset $S \subseteq V$, $|R(S)| \geq |S|$ where $R(S)$ is the set of all vertices adjacent to a vertex in S. Check this property for the following graph with (i) $V = \{1, 2, 3, 4\}$ and (ii) $V = \{5, 6, 7\}$.



b. You have to devise an algorithm that returns "yes" if there is a matching in a bipartite graph $G = \langle V, U, E \rangle$ that matches all vertices in V and returns "no" otherwise. Would you base your algorithm on checking the condition of Hall's Marriage Theorem?

5. Suppose there are five committees A, B, C, D, and E composed of six persons a, b, c, d, e, and f as follows: committee A's members are b and e; committee B's members are b, d, and e; committee C's members are a, c, d, e, and f; committee D's members are b, d, and e; committee E's members are b and e. Is there a ***system of distinct representatives***, i.e., is it possible to select a representative from each committee so that all the selected persons are distinct?

6. Show how the maximum-cardinality-matching problem for a bipartite graph can be reduced to the maximum-flow problem discussed in Section 10.2.

7. Consider the following greedy algorithm for finding a maximum matching in a bipartite graph $G = \langle V, U, E \rangle$: Sort all the vertices in nondecreasing order of their degrees. Scan this sorted list to add to the current matching (initially empty) the edge from the list's free vertex to an adjacent free vertex of the lowest degree. If the list's vertex is matched or if there are no adjacent free vertices for it, the vertex is simply skipped. Does this algorithm always produce a maximum matching in a bipartite graph?

8. Design a linear-time algorithm for finding a maximum matching in a tree.

9. Implement the maximum-matching algorithm of this section in the language of your choice. Experiment with its performance on bipartite graphs with n vertices in each of the vertex sets and randomly generated edges (in both dense and sparse modes) to compare the observed running time with the algorithm's theoretical efficiency.

10. *Domino puzzle* A domino is a 2×1 tile that can be oriented either horizontally or vertically. A tiling of a given board composed of 1×1 squares is covering it with dominoes exactly and without overlap. Is it possible to tile with dominoes an 8×8 board without two unit squares at its diagonally opposite corners?

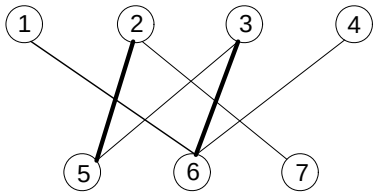
Hints to Exercises 10.3

1. You may but do not have to use the algorithm described in the section.
2. See an application of this algorithm to another bipartite graph in the section.
3. The definition of a matching and its cardinality should lead you to the answers to these questions with no difficulty.
4. a. You do not have to check the inequality for each subset S of V if you can point out a subset for which the inequality does not hold. Otherwise, fill in a table for all the subsets S of the indicated set V with columns for S, $R(S)$, and $|R(S)| \geq |S|$.

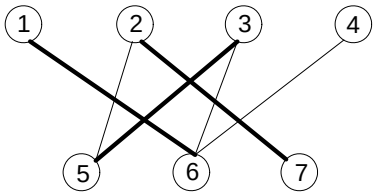
   b. Think time efficiency.
5. Reduce the problem to finding a maximum matching in a bipartite graph.
6. Transform a given bipartite graph into a network by making vertices of the former be intermediate vertices of the latter.
7. Since this greedy algorithm is arguably simpler than the augmenting path algorithm given in the section, should we expect a positive or negative answer? Of course, this point cannot be substituted for a more specific argument or a counterexample.
8. Start by presenting a tree given as a BFS tree.
9. For pointers regarding an efficient implementation of the algorithm, see [Pap82, Section 10.2].
10. Although not necessary, thinking about the problem as one dealing with matching squares of a chessboard might lead you to a short and elegant proof that this well-known puzzle has no solution.

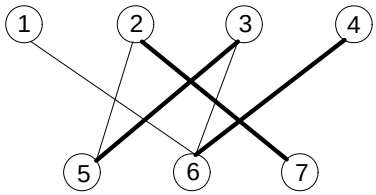
Solutions to Exercises 10.3

1. a. The matching given in the exercise is reproduced below:

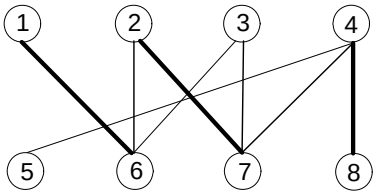


Its possible augmentations are:



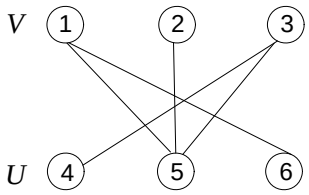


b. No augmentation of the matching given in part b (reproduced below) is possible.

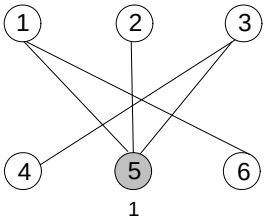


This conclusion can be arrived at either by applying the maximum-matching algorithm or by simply noting that only one of vertices 5 and 8 can be matched (and, hence, no more than three of vertices 5, 6, 7, and 8 can be matched in any matching).

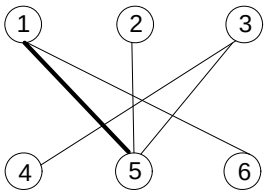
2. Here is a trace of the maximum-matching algorithm applied to the bipartite graph in question:



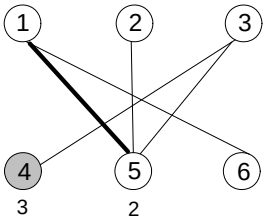
Queue: 1 2 3



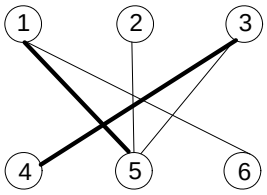
Queue: 1 2 3
↑
Augment from 5



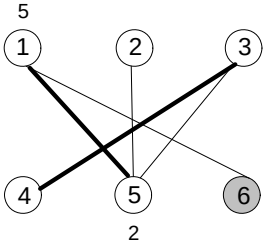
Queue: 2 3



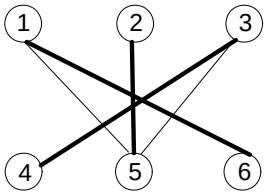
Queue: 2 3 5
↑ ↑
Augment from 4



Queue: 2



Queue: 2 5 1
↑ ↑ ↑
Augment from 6



No free vertices ⇒ maximum (here perfect) matching

3. a. The largest cardinalilty of a matching is n when all the vertices of a graph are matched (perfect matching). For example, if $V = \{v_1, v_2, ..., v_n\}$,

$U = \{u_1, u_2, ..., u_n\}$ and $E = \{(v_1, u_1), (v_2, u_2), ..., (v_n, u_n)\}$, then $M = E$ is a perfect matching of size n.
The smallest cardinality of a matching is 1. (It cannot be zero because the number of edges is assumed to be at least $n \geq 1$.) For example, in the bipartite graph with $V = \{v_1, v_2, ..., v_n\}$, $U = \{u_1, u_2, ..., u_n\}$, and $E = \{(v_1, u_1), (v_1, u_2), ..., (v_1, u_n)\}$, the size of any matching is 1.

b. Consider $K_{n,n}$, the bipartite graph in which each of the n vertices in V is connected to each of the n vertices in U. To obtain a perfect matching, there are n possible mates for vertex v_1, $n-1$ possible remaining mates for v_2, and so on until there is just one possible remaining mate for v_n. Therefore the total number of distinct perfect matchings for $K_{n,n}$ is $n(n-1)...1 = n!$.

The smallest number of distinct maximum matchings is 1. For example, if $V = \{v_1, v_2, ..., v_n\}$, $U = \{u_1, u_2, ..., u_n\}$, and $E = \{(v_1, u_1), (v_2, u_2), ..., (v_n, u_n)\}$, $M = E$ is the only perfect (and hence maximum) matching.

4. a. (i) For $V = \{1, 2, 3, 4\}$, the inequality obviously fails for $S = V$ since $|R(S)| = |U| = 3$ while $|S| = 4$.
Hence, according to Hall's Marriage Theorem, there is no matching that matches all the vertices of the set $\{1, 2, 3, 4\}$.

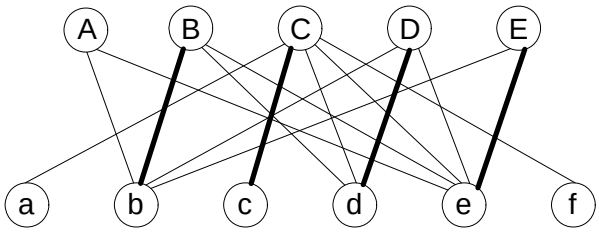
(ii) For subsets S of $V = \{5, 6, 7\}$, we have the following table:

S	$R(S)$	$\lvert R(S)\rvert \geq \lvert S\rvert$
{5}	{1,2,3}	$3 \geq 1$
{6}	{1}	$1 \geq 1$
{7}	{2,3,4}	$3 \geq 1$
{5,6}	{1,2,3}	$3 \geq 2$
{5,7}	{1,2,3,4}	$4 \geq 2$
{6,7}	{1,2,3,4}	$4 \geq 2$
{5,6,7}	{1,2,3,4}	$4 \geq 3$

Hence, according to Hall's Marriage Theorem, there is a matching that matches all the vertices of the set $\{5, 6, 7\}$. (Obviously, such a matching must be maximal.)

b. Since Hall's theorem requires checking an inequality for each subset $S \subseteq V$, the worst-case time efficiency of an algorithm based on it would be in $\Omega(2^{|V|})$. A much better solution is to find a maximum matching M^* (e.g., by the section's algorithm) and return "yes" if $|M^*| = |V|$ and "no" otherwise.

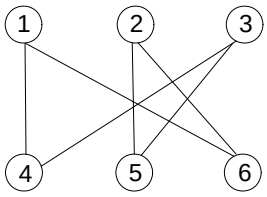
5. It is convenient to model the situation by a bipartite graph $G = \langle V, U, E \rangle$ where V represents the committees, U represents the committee members, and $(v, u) \in E$ if and only if u belongs to committee v:



There exists no matching that would match all the vertices of the set V. One way to prove it is based on Hall's Marriage Theorem (see Problem 4) whose necessary condition is violated for set $S = \{A, B, D, E\}$ with $R(S) = \{b, d, e\}$. Another way is to select a matching such as $M = \{(B, b), (C, c), (D, d), (E, e)\}$ (shown in bold) and check that the maximum-matching algorithm fails to find an augmenting path for it.

6. Add one source vertex s and connect it to each of the vertices in the set V by directed edges leaving s. Add one sink vertex t and connect each of the vertices in the set U to t by a directed edge entering t. Direct all the edges of the original graph to point from V to U. Assign 1 as the capacity of every edge in the network. A solution to the maximum-flow problem for the network yields a maximum matching for the original bipartite graph: it consists of the full edges (edges with the unit flow on them) between vertices of the original graph.

7. The greedy algorithm does not always find a maximum matching. As a counterexample, consider the bipartite graph shown below:



Since all its vertices have the same degree of 2, we can order them in numerical order of their labels: 1, 2, 3, 4, 5, 6. Using the same rule to break ties for selecting mates for vertices on the list, the greedy algorithm yields the matching $M = \{(1, 4), (2, 5)\}$, which is smaller than, say, $M^* = \{(1, 4), (2, 6), (3, 5)\}$.

8. A crucial observation is that for any edge between a leaf and its parent there is a maximum matching containing this edge. Indeed, consider a leaf v and its parent p. Let M be a maximum matching in the tree. If (v, p) is in M, we have a maximum matching desired. If M does not include (v, p), it must contain an edge (u, p) from some vertex $u \neq v$ to p because otherwise we could have added (v, p) to M to get a larger matching. But then simply replacing (u, p) by (v, p) in the matching M yields a maximum marching containing (v, p). This operation is to be repeated recursively for the smaller forest obtained.

   Based on this observation, Manber ([Man89], p. 431) suggested the following recursive algorithm. Take an arbitrary leaf v and match it to its parent.p. Remove from the tree both v and p along with all the edges incident to p. Also remove all the former sibling leaves of v, if any. This operation is to be repeated recursively for the smaller forest obtained.

   Thieling Chen [Thieling Chen, "Maximum Matching and Minimum Vertex Covers of Trees," *The Western Journal of Graduate Research*, vol. 10, no. 1, 2001, pp. 10-14] suggested to implement the same idea by processing vertices of the tree in the reverse BFS order (i.e., bottom up and right to left across each level) as in the following pseudocode:

   Algorithm *Tree-Max-Matching*
   //Constructs a maximum matching in a free tree
   //Input: A tree T
   //Output: A maximum cardinality matching M in T
   initialize matching M to the empty set
   starting at an arbitrary vertex, traverse T by BFS, numbering the visited vertices sequentially from 0 to $n-1$ and saving pointers to a visited vertex and its parent
   for $i \leftarrow n-1$ **downto** 0 **do**
       if vertex numbered i and its parent are both not marked
           add the edge between them to M
           mark the parent
   return M

   The time efficiency of this algorithm is obviously in $\Theta(n)$, where n is thenumber of vertices in an input tree.

10. No domino tiling of such a board is possible. Think of the board as being an 8×8 chessboard (with two missing squares at the diagonally opposite corners) whose squares of the board colored alternatingly black and white. Since a single domino would cover (match) exactly one black and one white square, no tiling of the board is possible because it has 32 squares of one color and 30 squares of the other color.

Exercises 10.4

1. Consider an instance of the stable marriage problem given by the following ranking matrix:

	A	B	C
α	$1, 3$	$2, 2$	$3, 1$
β	$3, 1$	$1, 3$	$2, 2$
γ	$2, 2$	$3, 1$	$1, 3$

   For each of its marriage matchings, indicate whether it is stable or not. For the unstable matchings, specify a blocking pair. For the stable matchings, indicate whether they are man-optimal, woman-optimal, or neither. (Assume that the Greek and Roman letters denote the men and women, respectively.)

2. Design a simple algorithm for checking whether a given marriage matching is stable and determine its time efficiency class.

3. Find a stable-marriage matching for the instance given in Problem 1 by applying the stable-marriage algorithm

   (a) in its men-proposing version.

   (b) in its women-proposing version.

4. Find a stable-marriage matching for the instance defined by the following ranking matrix:

	A	B	C	D
α	$1, 3$	$2, 3$	$3, 2$	$4, 3$
β	$1, 4$	$4, 1$	$3, 4$	$2, 2$
γ	$2, 2$	$1, 4$	$3, 3$	$4, 1$
δ	$4, 1$	$2, 2$	$3, 1$	$1, 4$

5. Determine the time-efficiency class of the stable-marriage algorithm

   (a) in the worst case.

   (b) in the best case.

6. Prove that a man-optimal stable marriage set is always unique. Is it also true for a woman-optimal stable marriage matching?

7. Prove that in the man-optimal stable matching, each woman has the worst partner that she can have in any stable marriage matching.

8. Implement the stable-marriage algorithm given in Section 10.4 so that its running time is in $O(n^2)$. Run an experiment to ascertain its average-case efficiency.

9. Write a report on the ***college admission problem*** (residents-hospitals assignment) that generalizes the stable marriage problem in that a college can accept "proposals" from more than one applicant.

10. ▷ Consider the ***problem of the roommates***, which is related to but more difficult than the stable marriage problem: "An even number of boys wish to divide up into pairs of roommates. A set of pairings is called stable if under it there are no two boys who are not roommates and who prefer each other to their actual roommates." [Gal62] Give an instance of this problem that does *not* have a stable pairing.

Hints to Exercises 10.4

1. A marriage matching is obtained by selecting three matrix cells, one cell from each row and column. To determine the stability of a given marriage matching, check each of the remaining matrix cells for containing a blocking pair.

2. It suffices to consider each member of one sex (say, the men) as a potential member of a blocking pair.

3. An application of the men-proposing version to another instance is given in the section. For the women-proposing version, reverse the roles of the sexes.

4. You may use either the men-proposing or women-proposing version of the algorithm.

5. The time efficiency is clearly defined by the number of proposals made. You may but are not required to provide the exact number of proposals in the worst and best cases, respectively; an appropriate Θ class will suffice.

6. Prove it by contradiction.

7. Prove it by contradiction.

8. Choose data structures so that the innermost loop of the algorithm can run in constant time.

9. The principal references are [Gal62] and [Gus89].

10. Consider four boys, three of whom rate the fourth boy as the least desired roommate. Complete these rankings to obtain an instance with no stable pairing.

Solutions to Exercises 10.4

1. There are the total of $3! = 6$ one-one matchings of two disjoint 3-element sets:

	A	B	C
α	$\boxed{1,3}$	$2,2$	$3,1$
β	$3,1$	$\boxed{1,3}$	$2,2$
γ	$2,2$	$3,1$	$\boxed{1,3}$

$\{(\alpha, A), (\beta, B), (\gamma, C)\}$ is stable: no other cell can be blocking since each man has his best choice. This is obviously the man-optimal matching.

	A	B	C
α	$\boxed{1,3}$	$2,2$	$3,1$
β	$3,1$	$1,3$	$\boxed{2,2}$
γ	$2,2$	$\boxed{3,1}$	$1,3$

$\{(\alpha, A), (\beta, C), (\gamma, B)\}$ is unstable: (γ, A) is a blocking pair.

	A	B	C
α	$1,3$	$\boxed{2,2}$	$3,1$
β	$\boxed{3,1}$	$1,3$	$2,2$
γ	$2,2$	$3,1$	$\boxed{1,3}$

$\{(\alpha, B), (\beta, A), (\gamma, C)\}$ is unstable: (β, C) is a blocking pair.

	A	B	C
α	$1,3$	$\boxed{2,2}$	$3,1$
β	$3,1$	$1,3$	$\boxed{2,2}$
γ	$\boxed{2,2}$	$3,1$	$1,3$

$\{(\alpha, B), (\beta, C), (\gamma, A)\}$ is stable: all the other cells contain a 3 (the lowest rank) and hence cannot be a blocking pair. This is neither a man-optimal nor a woman-optimal matching since it's inferior to $\{(\alpha, A), (\beta, B), (\gamma, C)\}$ for the men and inferior to $\{(\alpha, C), (\beta, A), (\gamma, B)\}$ for the women.

	A	B	C
α	$1,3$	$2,2$	$\boxed{3,1}$
β	$\boxed{3,1}$	$1,3$	$2,2$
γ	$2,2$	$\boxed{3,1}$	$1,3$

$\{(\alpha, C), (\beta, A), (\gamma, B)\}$ is stable: no other cell can be blocking since each woman has her best choice. This is obviously the woman-optimal matching.

	A	B	C
α	$1, 3$	$2, 2$	$\boxed{3,1}$
β	$3, 1$	$\boxed{1,3}$	$2, 2$
γ	$\boxed{2,2}$	$3, 1$	$1, 3$

$\{(\alpha, C), (\beta, B), (\gamma, A)\}$ is unstable: (α, B) is a blocking pair.

2. **Stability-checking algorithm**
Input: A marriage matching M of n (m, w) pairs along with rankings of the
women by each man and rankings of the men by each woman
Output: "yes" if the input is stable and a blocking pair otherwise
for $m \leftarrow 1$ **to** n **do**
for each w such that m prefers w to his mate in M **do**
if w prefers m to her mate in M
return (m, w)
return "yes"

With appropriate data structures, it is not difficult to implement this algorithm to run in $O(n^2)$ time. For example, the mates of the men and the mates of the women in a current matching can be stored in two arrays of size n and all the preferences can be stored in the $n \times n$ ranking matrix containing two rankings in each cell.

3. a.

		A	B	C	
Free men:	α	1,3	$2,2$	$3,1$	α proposed to A
α, β, γ	β	$3,1$	$1,3$	$2,2$	A accepted
	γ	$2,2$	$3,1$	$1,3$	

		A	B	C	
Free men:	α	1,3	$2,2$	$3,1$	β proposed to B
β, γ	β	$3,1$	1,3	$2,2$	B accepted
	γ	$2,2$	$3,1$	$1,3$	

		A	B	C	
Free men:	α	1,3	$2,2$	$3,1$	γ proposed to C
γ	β	$3,1$	1,3	$2,2$	C accepted
	γ	$2,2$	$3,1$	1,3	

The (man-optimal) stable marriage matching is $M = \{(\alpha, A), (\beta, B), (\gamma, C)\}$.

b.

		A	B	C	
Free women:	α	$1,3$	$2,2$	$3,1$	A proposed to β
A, B, C	β	3,1	$1,3$	$2,2$	β accepted
	γ	$2,2$	$3,1$	$1,3$	

		A	B	C	
Free women:	α	$1,3$	$2,2$	$3,1$	B proposed to γ
B, C	β	3,1	$1,3$	$2,2$	γ accepted
	γ	$2,2$	3,1	$1,3$	

		A	B	C	
Free women:	α	$1,3$	$2,2$	3,1	C proposed to α
C	β	3,1	$1,3$	$2,2$	α accepted
	γ	$2,2$	3,1	$1,3$	

The (woman-optimal) stable marriage matching is $M = \{(\beta, A), (\gamma, B), (\alpha, C)\}$.

4.

iteration 1

Free men: $\alpha, \beta, \gamma, \delta$

	A	B	C	D
α	1,3	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	$2,2$
γ	$2,2$	$1,4$	$3,3$	$4,1$
δ	$4,1$	$2,2$	$3,1$	$1,4$

α proposed to A; A accepted

iteration 2

Free men: β, γ, δ

	A	B	C	D
α	1,3	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	$2,2$
γ	$2,2$	$1,4$	$3,3$	$4,1$
δ	$4,1$	$2,2$	$3,1$	$1,4$

β proposed to A; A rejected

iteration 3

Free men: β, γ, δ

	A	B	C	D
α	1,3	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	$2,2$	$1,4$	$3,3$	$4,1$
δ	$4,1$	$2,2$	$3,1$	$1,4$

β proposed to D; D accepted

iteration 4

Free men: γ, δ

	A	B	C	D
α	1,3	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	$2,2$	1,4	$3,3$	$4,1$
δ	$4,1$	$2,2$	$3,1$	$1,4$

γ proposed to B; B accepted

iteration 5

Free men: δ

	A	B	C	D
α	1,3	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	$2,2$	1,4	$3,3$	$4,1$
δ	$4,1$	$2,2$	$3,1$	$\underline{1,4}$

δ proposed to D; D rejected

iteration 6

Free men: δ

	A	B	C	D
α	1,3	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	$2,2$	$1,4$	$3,3$	$4,1$
δ	$4,1$	2,2	$3,1$	$1,4$

δ proposed to B; B replaced γ with δ

iteration 7

Free men: γ

	A	B	C	D
α	$1,3$	$2,3$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	2,2	$1,4$	$3,3$	$4,1$
δ	$4,1$	2,2	$3,1$	$1,4$

γ proposed to A; A replaced α with γ

iteration 8

Free men: α

	A	B	C	D
α	$1,3$	$\underline{2,3}$	$3,2$	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	2,2	$1,4$	$3,3$	$4,1$
δ	$4,1$	2,2	$3,1$	$1,4$

α proposed to B; B rejected

iteration 9

Free men: α

	A	B	C	D
α	$1,3$	$2,3$	3,2	$4,3$
β	$1,4$	$4,1$	$3,4$	2,2
γ	2,2	$1,4$	$3,3$	$4,1$
δ	$4,1$	2,2	$3,1$	$1,4$

α proposed to C; C accepted

Free men: none

$M = \{(\alpha, C), (\beta, D), (\gamma, A), (\delta, B)\}$

5. a. The worst-case time efficiency of the algorithm is $\Theta(n^2)$. On the one hand, the total number of the proposals, $P(n)$, cannot exceed n^2, the total number of possible partners for n men, because a man does not propose to the same woman more than once. On the other hand, for the instance of size n where all the men and women have the identical preference list 1, 2, ..., n, $P(n) = \sum_{i=1}^{n} i = n(n+1)/2$. Thus, if $P_w(n)$ is the number of proposals made by the algorithm in the worst case,

$$n(n+1)/2 \le P_w(n) \le n^2,$$

i.e., $P_w(n) \in \Theta(n^2)$.

b. The best-case time efficiency of the algorithm is $\Theta(n)$: the algorithm makes the minimum of n proposals, one by each man, on the input that ranks first a different woman for each of the n men.

6. Assume that there are two distinct man-optimal solutions to an instance of the stable marriage problem. Then there must exist at least one man m matched to two different women w_1 and w_2 in these solutions. Since no ties are allowed in the rankings, m must prefer one of these two women to the other, say, w_1 to w_2. But then the marriage matching in which m is matched with w_2 is not man-optimal in contradiction to the assumption.

Of course, a woman-optimal solution is unique too due to the complete symmetry of these notions.

7. Assume that on the contrary there exists a man-optimal stable matching M in which some woman w doesn't have the worst possible partner in a stable matching, i.e., w prefers her partner m in M to her partner $\bar{m}$ in another stable matching $\bar{M}$. Since $(m, w) \notin \bar{M}$, m must prefer his partner $\bar{w}$ in $\bar{M}$ to w because otherwise (m, w) would be a blocking pair for $\bar{M}$. But this contradicts the assumption that M is a man-optimal stable matching in which every man, including m, has the best possible partner in a stable matching.

8. n/a

9. n/a

10. Consider an instance of the problem of the roommates with four boys α, β, γ, and δ and the following preference lists ($*$ stands for any legitimate

rating):

boy	rank 1	rank 2	rank 3
α	β	γ	δ
β	γ	α	δ
γ	α	β	δ
δ	$*$	$*$	$*$

Any pairing would have to pair one of the boys α, β, γ with δ. But any such pairing would be unstable since whoever is paired with δ will want to move out and one of the other two boys, having him rated first, will prefer him to his current roommate. For example, if α is paired with δ then β and γ are paired too while γ prefers α to β (in addition to α preferring γ to δ).

Note: This example is from the seminal paper by D. Gale and L. S. Shapley "College Admissions and the Stability of Marriage", *American Mathematical Monthly*, vol. 69 (Jan. 1962), 9-15. For an in-depth discussion of the problem of the roommates see the monograph by D. Gusfield and R.W. Irwing *The Stable Marriage Problem*: *Structure and Algorithms,*. MIT Press, 1989.