linear program for which the solution has the property that d_v is the shortest-path weight from s to v for each vertex $v \in V$.

29.2-4
Write out explicitly the linear program corresponding to finding the maximum flow in Figure 26.1(a).

29.2-5
Rewrite the linear program for maximum flow (29.47)–(29.50) so that it uses only $O(V + E)$ constraints.

29.2-6
Write a linear program that, given a bipartite graph $G = (V, E)$, solves the maximum-bipartite-matching problem.

29.2-7
In the ***minimum-cost multicommodity-flow problem***, we are given directed graph $G = (V, E)$ in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$ and a cost $a(u, v)$. As in the multicommodity-flow problem, we are given k different commodities, $K_1, K_2, \ldots, K_k$, where we specify commodity i by the triple $K_i = (s_i, t_i, d_i)$. We define the flow f_i for commodity i and the aggregate flow f_{uv} on edge (u, v) as in the multicommodity-flow problem. A feasible flow is one in which the aggregate flow on each edge (u, v) is no more than the capacity of edge (u, v). The cost of a flow is $\sum_{u,v \in V} a(u, v) f_{uv}$, and the goal is to find the feasible flow of minimum cost. Express this problem as a linear program.

29.3 The simplex algorithm

The simplex algorithm is the classical method for solving linear programs. In contrast to most of the other algorithms in this book, its running time is not polynomial in the worst case. It does yield insight into linear programs, however, and is often remarkably fast in practice.

In addition to having a geometric interpretation, described earlier in this chapter, the simplex algorithm bears some similarity to Gaussian elimination, discussed in Section 28.1. Gaussian elimination begins with a system of linear equalities whose solution is unknown. In each iteration, we rewrite this system in an equivalent form that has some additional structure. After some number of iterations, we have rewritten the system so that the solution is simple to obtain. The simplex algorithm proceeds in a similar manner, and we can view it as Gaussian elimination for inequalities.

We now describe the main idea behind an iteration of the simplex algorithm. Associated with each iteration will be a "basic solution" that we can easily obtain from the slack form of the linear program: set each nonbasic variable to 0 and compute the values of the basic variables from the equality constraints. An iteration converts one slack form into an equivalent slack form. The objective value of the associated basic feasible solution will be no less than that at the previous iteration, and usually greater. To achieve this increase in the objective value, we choose a nonbasic variable such that if we were to increase that variable's value from 0, then the objective value would increase, too. The amount by which we can increase the variable is limited by the other constraints. In particular, we raise it until some basic variable becomes 0. We then rewrite the slack form, exchanging the roles of that basic variable and the chosen nonbasic variable. Although we have used a particular setting of the variables to guide the algorithm, and we shall use it in our proofs, the algorithm does not explicitly maintain this solution. It simply rewrites the linear program until an optimal solution becomes "obvious."

An example of the simplex algorithm

We begin with an extended example. Consider the following linear program in standard form:

$$\begin{array}{lrcrcrcrl}
\text{maximize} & 3x_1 & + & x_2 & + & 2x_3 & & & \quad (29.53)\\
\text{subject to} & & & & & & & & \\
 & x_1 & + & x_2 & + & 3x_3 & \le & 30 & \quad (29.54)\\
 & 2x_1 & + & 2x_2 & + & 5x_3 & \le & 24 & \quad (29.55)\\
 & 4x_1 & + & x_2 & + & 2x_3 & \le & 36 & \quad (29.56)\\
 & & & x_1, x_2, x_3 & & & \ge & 0\ . & \quad (29.57)
\end{array}$$

In order to use the simplex algorithm, we must convert the linear program into slack form; we saw how to do so in Section 29.1. In addition to being an algebraic manipulation, slack is a useful algorithmic concept. Recalling from Section 29.1 that each variable has a corresponding nonnegativity constraint, we say that an equality constraint is ***tight*** for a particular setting of its nonbasic variables if they cause the constraint's basic variable to become 0. Similarly, a setting of the non-basic variables that would make a basic variable become negative ***violates*** that constraint. Thus, the slack variables explicitly maintain how far each constraint is from being tight, and so they help to determine how much we can increase values of nonbasic variables without violating any constraints.

Associating the slack variables x_4, x_5, and x_6 with inequalities (29.54)–(29.56), respectively, and putting the linear program into slack form, we obtain

$$
\begin{aligned}
z &= \qquad\; 3x_1 + x_2 + 2x_3 && (29.58)\\
x_4 &= 30 - x_1 - x_2 - 3x_3 && (29.59)\\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 && (29.60)\\
x_6 &= 36 - 4x_1 - x_2 - 2x_3\ . && (29.61)
\end{aligned}
$$

The system of constraints (29.59)–(29.61) has 3 equations and 6 variables. Any setting of the variables x_1, x_2, and x_3 defines values for x_4, x_5, and x_6; therefore, we have an infinite number of solutions to this system of equations. A solution is feasible if all of $x_1, x_2, \ldots, x_6$ are nonnegative, and there can be an infinite number of feasible solutions as well. The infinite number of possible solutions to a system such as this one will be useful in later proofs. We focus on the ***basic solution***: set all the (nonbasic) variables on the right-hand side to 0 and then compute the values of the (basic) variables on the left-hand side. In this example, the basic solution is $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_6) = (0, 0, 0, 30, 24, 36)$ and it has objective value $z = (3 \cdot 0) + (1 \cdot 0) + (2 \cdot 0) = 0$. Observe that this basic solution sets $\bar{x}_i = b_i$ for each $i \in B$. An iteration of the simplex algorithm rewrites the set of equations and the objective function so as to put a different set of variables on the right-hand side. Thus, a different basic solution is associated with the rewritten problem. We emphasize that the rewrite does not in any way change the underlying linear-programming problem; the problem at one iteration has the identical set of feasible solutions as the problem at the previous iteration. The problem does, however, have a different basic solution than that of the previous iteration.

If a basic solution is also feasible, we call it a ***basic feasible solution***. As we run the simplex algorithm, the basic solution is almost always a basic feasible solution. We shall see in Section 29.5, however, that for the first few iterations of the simplex algorithm, the basic solution might not be feasible.

Our goal, in each iteration, is to reformulate the linear program so that the basic solution has a greater objective value. We select a nonbasic variable x_e whose coefficient in the objective function is positive, and we increase the value of x_e as much as possible without violating any of the constraints. The variable x_e becomes basic, and some other variable x_l becomes nonbasic. The values of other basic variables and of the objective function may also change.

To continue the example, let's think about increasing the value of x_1. As we increase x_1, the values of x_4, x_5, and x_6 all decrease. Because we have a nonnegativity constraint for each variable, we cannot allow any of them to become negative. If x_1 increases above 30, then x_4 becomes negative, and x_5 and x_6 become negative when x_1 increases above 12 and 9, respectively. The third constraint (29.61) is the tightest constraint, and it limits how much we can increase x_1. Therefore, we switch the roles of x_1 and x_6. We solve equation (29.61) for x_1 and obtain

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}\ . \qquad (29.62)$$

To rewrite the other equations with x_6 on the right-hand side, we substitute for x_1 using equation (29.62). Doing so for equation (29.59), we obtain

$$\begin{aligned} x_4 &= 30 - x_1 - x_2 - 3x_3 \\ &= 30 - \left(9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}\right) - x_2 - 3x_3 \\ &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} . \end{aligned} \tag{29.63}$$

Similarly, we combine equation (29.62) with constraint (29.60) and with objective function (29.58) to rewrite our linear program in the following form:

$$z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \tag{29.64}$$
$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \tag{29.65}$$
$$x_4 = 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \tag{29.66}$$
$$x_5 = 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2} . \tag{29.67}$$

We call this operation a ***pivot***. As demonstrated above, a pivot chooses a nonbasic variable x_e, called the ***entering variable***, and a basic variable x_l, called the ***leaving variable***, and exchanges their roles.

The linear program described in equations (29.64)–(29.67) is equivalent to the linear program described in equations (29.58)–(29.61). We perform two operations in the simplex algorithm: rewrite equations so that variables move between the left-hand side and the right-hand side, and substitute one equation into another. The first operation trivially creates an equivalent problem, and the second, by elementary linear algebra, also creates an equivalent problem. (See Exercise 29.3-3.)

To demonstrate this equivalence, observe that our original basic solution $(0, 0, 0, 30, 24, 36)$ satisfies the new equations (29.65)–(29.67) and has objective value $27 + (1/4) \cdot 0 + (1/2) \cdot 0 - (3/4) \cdot 36 = 0$. The basic solution associated with the new linear program sets the nonbasic values to 0 and is $(9, 0, 0, 21, 6, 0)$, with objective value $z = 27$. Simple arithmetic verifies that this solution also satisfies equations (29.59)–(29.61) and, when plugged into objective function (29.58), has objective value $(3 \cdot 9) + (1 \cdot 0) + (2 \cdot 0) = 27$.

Continuing the example, we wish to find a new variable whose value we wish to increase. We do not want to increase x_6, since as its value increases, the objective value decreases. We can attempt to increase either x_2 or x_3; let us choose x_3. How far can we increase x_3 without violating any of the constraints? Constraint (29.65) limits it to 18, constraint (29.66) limits it to $42/5$, and constraint (29.67) limits it to $3/2$. The third constraint is again the tightest one, and therefore we rewrite the third constraint so that x_3 is on the left-hand side and x_5 is on the right-hand side. We then substitute this new equation, $x_3 = 3/2 - 3x_2/8 - x_5/4 + x_6/8$, into equations (29.64)–(29.66) and obtain the new, but equivalent, system

$$z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \tag{29.68}$$

$$x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \tag{29.69}$$

$$x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \tag{29.70}$$

$$x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} . \tag{29.71}$$

This system has the associated basic solution $(33/4, 0, 3/2, 69/4, 0, 0)$, with objective value $111/4$. Now the only way to increase the objective value is to increase x_2. The three constraints give upper bounds of 132, 4, and ∞, respectively. (We get an upper bound of ∞ from constraint (29.71) because, as we increase x_2, the value of the basic variable x_4 increases also. This constraint, therefore, places no restriction on how much we can increase x_2.) We increase x_2 to 4, and it becomes nonbasic. Then we solve equation (29.70) for x_2 and substitute in the other equations to obtain

$$z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \tag{29.72}$$

$$x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \tag{29.73}$$

$$x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \tag{29.74}$$

$$x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} . \tag{29.75}$$

At this point, all coefficients in the objective function are negative. As we shall see later in this chapter, this situation occurs only when we have rewritten the linear program so that the basic solution is an optimal solution. Thus, for this problem, the solution $(8, 4, 0, 18, 0, 0)$, with objective value 28, is optimal. We can now return to our original linear program given in (29.53)–(29.57). The only variables in the original linear program are x_1, x_2, and x_3, and so our solution is $x_1 = 8$, $x_2 = 4$, and $x_3 = 0$, with objective value $(3 \cdot 8) + (1 \cdot 4) + (2 \cdot 0) = 28$. Note that the values of the slack variables in the final solution measure how much slack remains in each inequality. Slack variable x_4 is 18, and in inequality (29.54), the left-hand side, with value $8 + 4 + 0 = 12$, is 18 less than the right-hand side of 30. Slack variables x_5 and x_6 are 0 and indeed, in inequalities (29.55) and (29.56), the left-hand and right-hand sides are equal. Observe also that even though the coefficients in the original slack form are integral, the coefficients in the other linear programs are not necessarily integral, and the intermediate solutions are not necessarily integral. Furthermore, the final solution to a linear program need not be integral; it is purely coincidental that this example has an integral solution.

Pivoting

We now formalize the procedure for pivoting. The procedure PIVOT takes as input a slack form, given by the tuple (N, B, A, b, c, ν), the index l of the leaving variable x_l, and the index e of the entering variable x_e. It returns the tuple $(\widehat{N}, \widehat{B}, \widehat{A}, \widehat{b}, \widehat{c}, \widehat{\nu})$ describing the new slack form. (Recall again that the entries of the $m \times n$ matrices A and $\widehat{A}$ are actually the negatives of the coefficients that appear in the slack form.)

```
PIVOT$(N, B, A, b, c, \nu, l, e)$
 1  // Compute the coefficients of the equation for new basic variable $x_e$.
 2  let $\widehat{A}$ be a new $m \times n$ matrix
 3  $\widehat{b}_e = b_l / a_{le}$
 4  for each $j \in N - \{e\}$
 5      $\widehat{a}_{ej} = a_{lj} / a_{le}$
 6  $\widehat{a}_{el} = 1 / a_{le}$
 7  // Compute the coefficients of the remaining constraints.
 8  for each $i \in B - \{l\}$
 9      $\widehat{b}_i = b_i - a_{ie}\widehat{b}_e$
10      for each $j \in N - \{e\}$
11          $\widehat{a}_{ij} = a_{ij} - a_{ie}\widehat{a}_{ej}$
12      $\widehat{a}_{il} = -a_{ie}\widehat{a}_{el}$
13  // Compute the objective function.
14  $\widehat{\nu} = \nu + c_e\widehat{b}_e$
15  for each $j \in N - \{e\}$
16      $\widehat{c}_j = c_j - c_e\widehat{a}_{ej}$
17  $\widehat{c}_l = -c_e\widehat{a}_{el}$
18  // Compute new sets of basic and nonbasic variables.
19  $\widehat{N} = N - \{e\} \cup \{l\}$
20  $\widehat{B} = B - \{l\} \cup \{e\}$
21  return $(\widehat{N}, \widehat{B}, \widehat{A}, \widehat{b}, \widehat{c}, \widehat{\nu})$
```

PIVOT works as follows. Lines 3–6 compute the coefficients in the new equation for x_e by rewriting the equation that has x_l on the left-hand side to instead have x_e on the left-hand side. Lines 8–12 update the remaining equations by substituting the right-hand side of this new equation for each occurrence of x_e. Lines 14–17 do the same substitution for the objective function, and lines 19 and 20 update the sets of nonbasic and basic variables. Line 21 returns the new slack form. As given, if $a_{le} = 0$, PIVOT would cause an error by dividing by 0, but as we shall see in the proofs of Lemmas 29.2 and 29.12, we call PIVOT only when $a_{le} \neq 0$.

We now summarize the effect that PIVOT has on the values of the variables in the basic solution.

Lemma 29.1
Consider a call to PIVOT(N, B, A, b, c, v, l, e) in which $a_{le} \neq 0$. Let the values returned from the call be $(\widehat{N}, \widehat{B}, \widehat{A}, \widehat{b}, \widehat{c}, \widehat{v})$, and let $\bar{x}$ denote the basic solution after the call. Then

1. $\bar{x}_j = 0$ for each $j \in \widehat{N}$.
2. $\bar{x}_e = b_l / a_{le}$.
3. $\bar{x}_i = b_i - a_{ie}\widehat{b}_e$ for each $i \in \widehat{B} - \{e\}$.

Proof The first statement is true because the basic solution always sets all nonbasic variables to 0. When we set each nonbasic variable to 0 in a constraint

$$x_i = \widehat{b}_i - \sum_{j \in \widehat{N}} \widehat{a}_{ij} x_j ,$$

we have that $\bar{x}_i = \widehat{b}_i$ for each $i \in \widehat{B}$. Since $e \in \widehat{B}$, line 3 of PIVOT gives

$$\bar{x}_e = \widehat{b}_e = b_l / a_{le} ,$$

which proves the second statement. Similarly, using line 9 for each $i \in \widehat{B} - \{e\}$, we have

$$\bar{x}_i = \widehat{b}_i = b_i - a_{ie}\widehat{b}_e ,$$

which proves the third statement. ■

The formal simplex algorithm

We are now ready to formalize the simplex algorithm, which we demonstrated by example. That example was a particularly nice one, and we could have had several other issues to address:

- How do we determine whether a linear program is feasible?
- What do we do if the linear program is feasible, but the initial basic solution is not feasible?
- How do we determine whether a linear program is unbounded?
- How do we choose the entering and leaving variables?

In Section 29.5, we shall show how to determine whether a problem is feasible, and if so, how to find a slack form in which the initial basic solution is feasible. Therefore, let us assume that we have a procedure INITIALIZE-SIMPLEX(A, b, c) that takes as input a linear program in standard form, that is, an $m \times n$ matrix $A = (a_{ij})$, an m-vector $b = (b_i)$, and an n-vector $c = (c_j)$. If the problem is infeasible, the procedure returns a message that the program is infeasible and then terminates. Otherwise, the procedure returns a slack form for which the initial basic solution is feasible.

The procedure SIMPLEX takes as input a linear program in standard form, as just described. It returns an n-vector $\bar{x} = (\bar{x}_j)$ that is an optimal solution to the linear program described in (29.19)–(29.21).

```
SIMPLEX(A, b, c)
 1  (N, B, A, b, c, v) = INITIALIZE-SIMPLEX(A, b, c)
 2  let Δ be a new vector of length m
 3  while some index j ∈ N has c_j > 0
 4      choose an index e ∈ N for which c_e > 0
 5      for each index i ∈ B
 6          if a_ie > 0
 7              Δ_i = b_i / a_ie
 8          else Δ_i = ∞
 9      choose an index l ∈ B that minimizes Δ_l
10      if Δ_l == ∞
11          return "unbounded"
12      else (N, B, A, b, c, v) = PIVOT(N, B, A, b, c, v, l, e)
13  for i = 1 to n
14      if i ∈ B
15          x̄_i = b_i
16      else x̄_i = 0
17  return (x̄_1, x̄_2, ..., x̄_n)
```

The SIMPLEX procedure works as follows. In line 1, it calls the procedure INITIALIZE-SIMPLEX(A, b, c), described above, which either determines that the linear program is infeasible or returns a slack form for which the basic solution is feasible. The **while** loop of lines 3–12 forms the main part of the algorithm. If all coefficients in the objective function are negative, then the **while** loop terminates. Otherwise, line 4 selects a variable x_e, whose coefficient in the objective function is positive, as the entering variable. Although we may choose any such variable as the entering variable, we assume that we use some prespecified deterministic rule. Next, lines 5–9 check each constraint and pick the one that most severely limits the amount by which we can increase x_e without violating any of the nonnegativity constraints; the basic variable associated with this constraint is x_l. Again, we are free to choose one of several variables as the leaving variable, but we assume that we use some prespecified deterministic rule. If none of the constraints limits the amount by which the entering variable can increase, the algorithm returns "unbounded" in line 11. Otherwise, line 12 exchanges the roles of the entering and leaving variables by calling PIVOT$(N, B, A, b, c, \nu, l, e)$, as described above. Lines 13–16 compute a solution $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ for the original linear-programming variables by setting all the nonbasic variables to 0 and each basic variable $\bar{x}_i$ to b_i, and line 17 returns these values.

To show that SIMPLEX is correct, we first show that if SIMPLEX has an initial feasible solution and eventually terminates, then it either returns a feasible solution or determines that the linear program is unbounded. Then, we show that SIMPLEX terminates. Finally, in Section 29.4 (Theorem 29.10) we show that the solution returned is optimal.

Lemma 29.2
Given a linear program (A, b, c), suppose that the call to INITIALIZE-SIMPLEX in line 1 of SIMPLEX returns a slack form for which the basic solution is feasible. Then if SIMPLEX returns a solution in line 17, that solution is a feasible solution to the linear program. If SIMPLEX returns "unbounded" in line 11, the linear program is unbounded.

Proof We use the following three-part loop invariant:

At the start of each iteration of the **while** loop of lines 3–12,

1. the slack form is equivalent to the slack form returned by the call of INITIALIZE-SIMPLEX,
2. for each $i \in B$, we have $b_i \geq 0$, and
3. the basic solution associated with the slack form is feasible.

Initialization: The equivalence of the slack forms is trivial for the first iteration. We assume, in the statement of the lemma, that the call to INITIALIZE-SIMPLEX in line 1 of SIMPLEX returns a slack form for which the basic solution is feasible. Thus, the third part of the invariant is true. Because the basic solution is feasible, each basic variable x_i is nonnegative. Furthermore, since the basic solution sets each basic variable x_i to b_i, we have that $b_i \geq 0$ for all $i \in B$. Thus, the second part of the invariant holds.

Maintenance: We shall show that each iteration of the **while** loop maintains the loop invariant, assuming that the **return** statement in line 11 does not execute. We shall handle the case in which line 11 executes when we discuss termination.

An iteration of the **while** loop exchanges the role of a basic and a nonbasic variable by calling the PIVOT procedure. By Exercise 29.3-3, the slack form is equivalent to the one from the previous iteration which, by the loop invariant, is equivalent to the initial slack form.

We now demonstrate the second part of the loop invariant. We assume that at the start of each iteration of the **while** loop, $b_i \geq 0$ for each $i \in B$, and we shall show that these inequalities remain true after the call to PIVOT in line 12. Since the only changes to the variables b_i and the set B of basic variables occur in this assignment, it suffices to show that line 12 maintains this part of the invariant. We let b_i, a_{ij}, and B refer to values before the call of PIVOT, and $\widehat{b}_i$ refer to values returned from PIVOT.

First, we observe that $\widehat{b}_e \geq 0$ because $b_l \geq 0$ by the loop invariant, $a_{le} > 0$ by lines 6 and 9 of SIMPLEX, and $\widehat{b}_e = b_l/a_{le}$ by line 3 of PIVOT.

For the remaining indices $i \in B - \{l\}$, we have that

$$\begin{aligned} \widehat{b}_i &= b_i - a_{ie}\widehat{b}_e && \text{(by line 9 of PIVOT)} \\ &= b_i - a_{ie}(b_l/a_{le}) && \text{(by line 3 of PIVOT)} . \end{aligned} \tag{29.76}$$

We have two cases to consider, depending on whether $a_{ie} > 0$ or $a_{ie} \leq 0$. If $a_{ie} > 0$, then since we chose l such that

$$b_l/a_{le} \leq b_i/a_{ie} \quad \text{for all } i \in B , \tag{29.77}$$

we have

$$\begin{aligned} \widehat{b}_i &= b_i - a_{ie}(b_l/a_{le}) && \text{(by equation (29.76))} \\ &\geq b_i - a_{ie}(b_i/a_{ie}) && \text{(by inequality (29.77))} \\ &= b_i - b_i \\ &= 0 , \end{aligned}$$

and thus $\widehat{b}_i \geq 0$. If $a_{ie} \leq 0$, then because a_{le}, b_i, and b_l are all nonnegative, equation (29.76) implies that $\widehat{b}_i$ must be nonnegative, too.

We now argue that the basic solution is feasible, i.e., that all variables have nonnegative values. The nonbasic variables are set to 0 and thus are nonnegative. Each basic variable x_i is defined by the equation

$$x_i = b_i - \sum_{j \in N} a_{ij}x_j .$$

The basic solution sets $\bar{x}_i = b_i$. Using the second part of the loop invariant, we conclude that each basic variable $\bar{x}_i$ is nonnegative.

Termination: The **while** loop can terminate in one of two ways. If it terminates because of the condition in line 3, then the current basic solution is feasible and line 17 returns this solution. The other way it terminates is by returning "unbounded" in line 11. In this case, for each iteration of the **for** loop in lines 5–8, when line 6 is executed, we find that $a_{ie} \le 0$. Consider the solution $\bar{x}$ defined as

$$\bar{x}_i = \begin{cases} \infty & \text{if } i = e \ , \\ 0 & \text{if } i \in N - \{e\} \ , \\ b_i - \sum_{j \in N} a_{ij}\bar{x}_j & \text{if } i \in B \ . \end{cases}$$

We now show that this solution is feasible, i.e., that all variables are nonnegative. The nonbasic variables other than $\bar{x}_e$ are 0, and $\bar{x}_e = \infty > 0$; thus all nonbasic variables are nonnegative. For each basic variable $\bar{x}_i$, we have

$$\begin{aligned} \bar{x}_i &= b_i - \sum_{j \in N} a_{ij}\bar{x}_j \\ &= b_i - a_{ie}\bar{x}_e \ . \end{aligned}$$

The loop invariant implies that $b_i \ge 0$, and we have $a_{ie} \le 0$ and $\bar{x}_e = \infty > 0$. Thus, $\bar{x}_i \ge 0$.

Now we show that the objective value for the solution $\bar{x}$ is unbounded. From equation (29.42), the objective value is

$$\begin{aligned} z &= v + \sum_{j \in N} c_j\bar{x}_j \\ &= v + c_e\bar{x}_e \ . \end{aligned}$$

Since $c_e > 0$ (by line 4 of SIMPLEX) and $\bar{x}_e = \infty$, the objective value is ∞, and thus the linear program is unbounded. ■

It remains to show that SIMPLEX terminates, and when it does terminate, the solution it returns is optimal. Section 29.4 will address optimality. We now discuss termination.

Termination

In the example given in the beginning of this section, each iteration of the simplex algorithm increased the objective value associated with the basic solution. As Exercise 29.3-2 asks you to show, no iteration of SIMPLEX can decrease the objective value associated with the basic solution. Unfortunately, it is possible that an iteration leaves the objective value unchanged. This phenomenon is called ***degeneracy***, and we shall now study it in greater detail.

The assignment in line 14 of PIVOT, $\widehat{v} = v + c_e \widehat{b}_e$, changes the objective value. Since SIMPLEX calls PIVOT only when $c_e > 0$, the only way for the objective value to remain unchanged (i.e., $\widehat{v} = v$) is for $\widehat{b}_e$ to be 0. This value is assigned as $\widehat{b}_e = b_l / a_{le}$ in line 3 of PIVOT. Since we always call PIVOT with $a_{le} \neq 0$, we see that for $\widehat{b}_e$ to equal 0, and hence the objective value to be unchanged, we must have $b_l = 0$.

Indeed, this situation can occur. Consider the linear program

$$\begin{array}{rcrcrcrcr} z & = & & & x_1 & + & x_2 & + & x_3 \\ x_4 & = & 8 & - & x_1 & - & x_2 & & \\ x_5 & = & & & & & x_2 & - & x_3 \ . \end{array}$$

Suppose that we choose x_1 as the entering variable and x_4 as the leaving variable. After pivoting, we obtain

$$\begin{array}{rcrcrcrcr} z & = & 8 & & & + & x_3 & - & x_4 \\ x_1 & = & 8 & - & x_2 & & & - & x_4 \\ x_5 & = & & & x_2 & - & x_3 \ . & & \end{array}$$

At this point, our only choice is to pivot with x_3 entering and x_5 leaving. Since $b_5 = 0$, the objective value of 8 remains unchanged after pivoting:

$$\begin{array}{rcrcrcrcr} z & = & 8 & + & x_2 & - & x_4 & - & x_5 \\ x_1 & = & 8 & - & x_2 & - & x_4 & & \\ x_3 & = & & & x_2 & & & - & x_5 \ . \end{array}$$

The objective value has not changed, but our slack form has. Fortunately, if we pivot again, with x_2 entering and x_1 leaving, the objective value increases (to 16), and the simplex algorithm can continue.

Degeneracy can prevent the simplex algorithm from terminating, because it can lead to a phenomenon known as ***cycling***: the slack forms at two different iterations of SIMPLEX are identical. Because of degeneracy, SIMPLEX could choose a sequence of pivot operations that leave the objective value unchanged but repeat a slack form within the sequence. Since SIMPLEX is a deterministic algorithm, if it cycles, then it will cycle through the same series of slack forms forever, never terminating.

Cycling is the only reason that SIMPLEX might not terminate. To show this fact, we must first develop some additional machinery.

At each iteration, SIMPLEX maintains A, b, c, and v in addition to the sets N and B. Although we need to explicitly maintain A, b, c, and v in order to implement the simplex algorithm efficiently, we can get by without maintaining them. In other words, the sets of basic and nonbasic variables suffice to uniquely determine the slack form. Before proving this fact, we prove a useful algebraic lemma.

Lemma 29.3

Let I be a set of indices. For each $j \in I$, let α_j and β_j be real numbers, and let x_j be a real-valued variable. Let γ be any real number. Suppose that for any settings of the x_j, we have

$$\sum_{j \in I} \alpha_j x_j = \gamma + \sum_{j \in I} \beta_j x_j \ . \tag{29.78}$$

Then $\alpha_j = \beta_j$ for each $j \in I$, and $\gamma = 0$.

Proof Since equation (29.78) holds for any values of the x_j, we can use particular values to draw conclusions about α, β, and γ. If we let $x_j = 0$ for each $j \in I$, we conclude that $\gamma = 0$. Now pick an arbitrary index $j \in I$, and set $x_j = 1$ and $x_k = 0$ for all $k \neq j$. Then we must have $\alpha_j = \beta_j$. Since we picked j as any index in I, we conclude that $\alpha_j = \beta_j$ for each $j \in I$. ■

A particular linear program has many different slack forms; recall that each slack form has the same set of feasible and optimal solutions as the original linear program. We now show that the slack form of a linear program is uniquely determined by the set of basic variables. That is, given the set of basic variables, a unique slack form (unique set of coefficients and right-hand sides) is associated with those basic variables.

Lemma 29.4

Let (A, b, c) be a linear program in standard form. Given a set B of basic variables, the associated slack form is uniquely determined.

Proof Assume for the purpose of contradiction that there are two different slack forms with the same set B of basic variables. The slack forms must also have identical sets $N = \{1, 2, \ldots, n + m\} - B$ of nonbasic variables. We write the first slack form as

$$z = \nu + \sum_{j \in N} c_j x_j \tag{29.79}$$

$$x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B \ , \tag{29.80}$$

and the second as

$$z = \nu' + \sum_{j \in N} c'_j x_j \tag{29.81}$$

$$x_i = b'_i - \sum_{j \in N} a'_{ij} x_j \quad \text{for } i \in B \ . \tag{29.82}$$

Consider the system of equations formed by subtracting each equation in line (29.82) from the corresponding equation in line (29.80). The resulting system is

$$0 = (b_i - b'_i) - \sum_{j \in N} (a_{ij} - a'_{ij}) x_j \quad \text{for } i \in B$$

or, equivalently,

$$\sum_{j \in N} a_{ij} x_j = (b_i - b'_i) + \sum_{j \in N} a'_{ij} x_j \quad \text{for } i \in B \ .$$

Now, for each $i \in B$, apply Lemma 29.3 with $\alpha_j = a_{ij}$, $\beta_j = a'_{ij}$, $\gamma = b_i - b'_i$, and $I = N$. Since $\alpha_j = \beta_j$, we have that $a_{ij} = a'_{ij}$ for each $j \in N$, and since $\gamma = 0$, we have that $b_i = b'_i$. Thus, for the two slack forms, A and b are identical to A' and b'. Using a similar argument, Exercise 29.3-1 shows that it must also be the case that $c = c'$ and $\nu = \nu'$, and hence that the slack forms must be identical. ■

We can now show that cycling is the only possible reason that SIMPLEX might not terminate.

Lemma 29.5
If SIMPLEX fails to terminate in at most $\binom{n+m}{m}$ iterations, then it cycles.

Proof By Lemma 29.4, the set B of basic variables uniquely determines a slack form. There are $n + m$ variables and $|B| = m$, and therefore, there are at most $\binom{n+m}{m}$ ways to choose B. Thus, there are only at most $\binom{n+m}{m}$ unique slack forms. Therefore, if SIMPLEX runs for more than $\binom{n+m}{m}$ iterations, it must cycle. ■

Cycling is theoretically possible, but extremely rare. We can prevent it by choosing the entering and leaving variables somewhat more carefully. One option is to perturb the input slightly so that it is impossible to have two solutions with the same objective value. Another option is to break ties by always choosing the variable with the smallest index, a strategy known as ***Bland's rule***. We omit the proof that these strategies avoid cycling.

Lemma 29.6
If lines 4 and 9 of SIMPLEX always break ties by choosing the variable with the smallest index, then SIMPLEX must terminate. ■

We conclude this section with the following lemma.

Lemma 29.7

Assuming that INITIALIZE-SIMPLEX returns a slack form for which the basic solution is feasible, SIMPLEX either reports that a linear program is unbounded, or it terminates with a feasible solution in at most $\binom{n+m}{m}$ iterations.

Proof Lemmas 29.2 and 29.6 show that if INITIALIZE-SIMPLEX returns a slack form for which the basic solution is feasible, SIMPLEX either reports that a linear program is unbounded, or it terminates with a feasible solution. By the contrapositive of Lemma 29.5, if SIMPLEX terminates with a feasible solution, then it terminates in at most $\binom{n+m}{m}$ iterations. ∎

Exercises

29.3-1

Complete the proof of Lemma 29.4 by showing that it must be the case that $c = c'$ and $\nu = \nu'$.

29.3-2

Show that the call to PIVOT in line 12 of SIMPLEX never decreases the value of ν.

29.3-3

Prove that the slack form given to the PIVOT procedure and the slack form that the procedure returns are equivalent.

29.3-4

Suppose we convert a linear program (A, b, c) in standard form to slack form. Show that the basic solution is feasible if and only if $b_i \geq 0$ for $i = 1, 2, \ldots, m$.

29.3-5

Solve the following linear program using SIMPLEX:

$$\begin{array}{llcrcl}
\text{maximize} & 18x_1 & + & 12.5x_2 & & \\
\text{subject to} & & & & & \\
& x_1 & + & x_2 & \leq & 20 \\
& x_1 & & & \leq & 12 \\
& & & x_2 & \leq & 16 \\
& x_1, x_2 & & & \geq & 0 \; .
\end{array}$$

29.3-6

Solve the following linear program using SIMPLEX:

$$\begin{array}{lrcrcl} \text{maximize} & 5x_1 & - & 3x_2 & & \\ \text{subject to} & & & & & \\ & x_1 & - & x_2 & \leq & 1 \\ & 2x_1 & + & x_2 & \leq & 2 \\ & x_1, x_2 & & & \geq & 0 \ . \end{array}$$

29.3-7

Solve the following linear program using SIMPLEX:

$$\begin{array}{lrcrcrcl} \text{minimize} & x_1 & + & x_2 & + & x_3 & & \\ \text{subject to} & & & & & & & \\ & 2x_1 & + & 7.5x_2 & + & 3x_3 & \geq & 10000 \\ & 20x_1 & + & 5x_2 & + & 10x_3 & \geq & 30000 \\ & x_1, x_2, x_3 & & & & & \geq & 0 \ . \end{array}$$

29.3-8

In the proof of Lemma 29.5, we argued that there are at most $\binom{m+n}{n}$ ways to choose a set B of basic variables. Give an example of a linear program in which there are strictly fewer than $\binom{m+n}{n}$ ways to choose the set B.

29.4 Duality

We have proven that, under certain assumptions, SIMPLEX terminates. We have not yet shown that it actually finds an optimal solution to a linear program, however. In order to do so, we introduce a powerful concept called ***linear-programming duality***.

Duality enables us to prove that a solution is indeed optimal. We saw an example of duality in Chapter 26 with Theorem 26.6, the max-flow min-cut theorem. Suppose that, given an instance of a maximum-flow problem, we find a flow f with value $|f|$. How do we know whether f is a maximum flow? By the max-flow min-cut theorem, if we can find a cut whose value is also $|f|$, then we have verified that f is indeed a maximum flow. This relationship provides an example of duality: given a maximization problem, we define a related minimization problem such that the two problems have the same optimal objective values.

Given a linear program in which the objective is to maximize, we shall describe how to formulate a ***dual*** linear program in which the objective is to minimize and