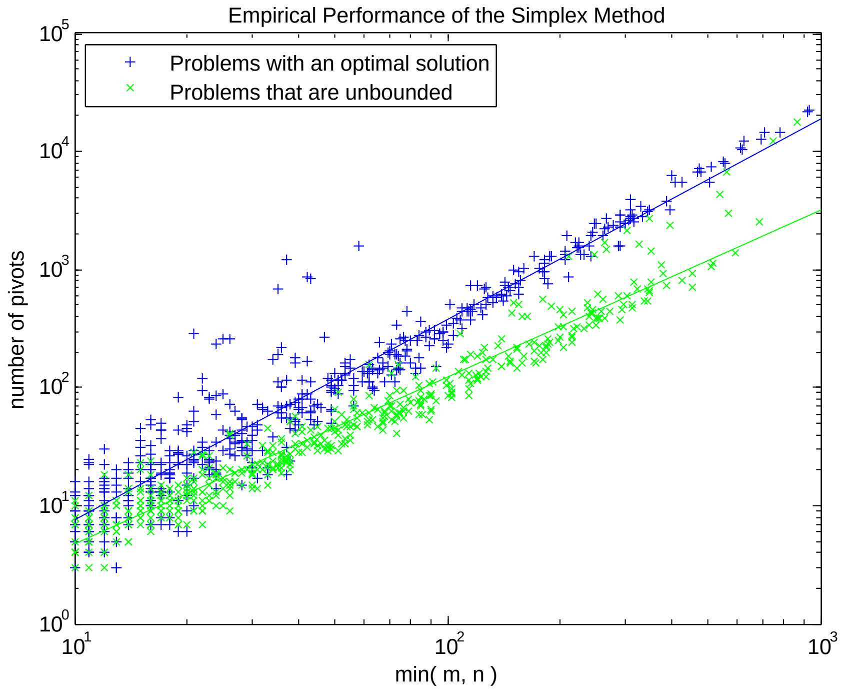

FIGURE 4.3. The same data as before but plotted against the minimum of m and n.

Exercises

In solving the following problems, the simple pivot tool can be used to check your arithmetic:

www.princeton.edu/∼rvdb/JAVA/pivot/simple.html

4.1 Compare the performance of the largest-coefficient and the smallest-index pivoting rules on the following linear program:

$$
\begin{array}{lrcl}
\text{maximize} & 4x_1 + 5x_2 & & \\
\text{subject to} & 2x_1 + 2x_2 & \leq & 9 \\
& x_1 \qquad & \leq & 4 \\
& x_2 & \leq & 3 \\
& x_1,\ x_2 & \geq & 0.
\end{array}
$$

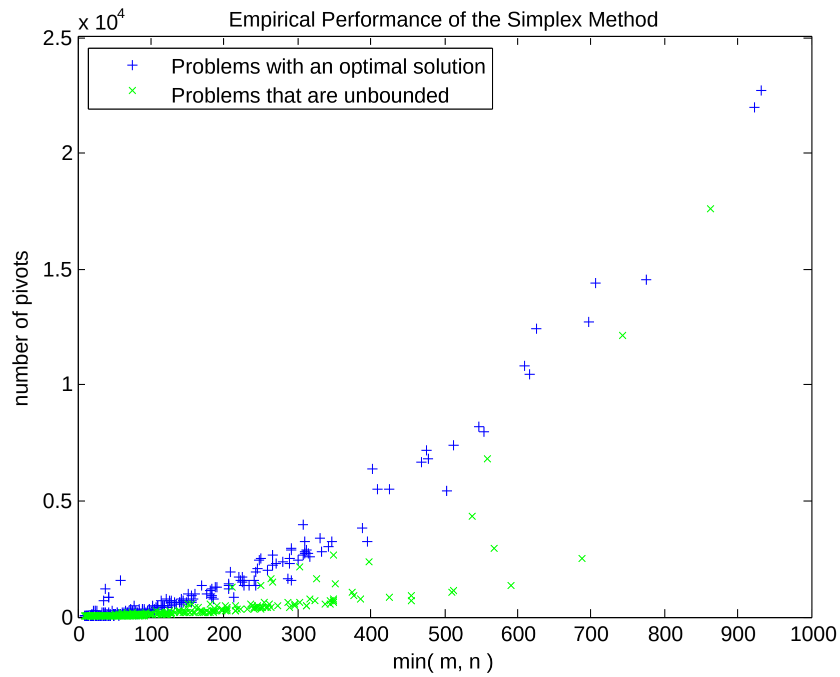

FIGURE 4.4. The same comparison as in Figure 4.3 but plot linearly rather than log-log. This version makes clear that the number of pivots grows faster than linearly.

4.2 Compare the performance of the largest-coefficient and the smallest-index pivoting rules on the following linear program:

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

4.3 Compare the performance of the largest-coefficient and the smallest-index pivoting rules on the following linear program:

$$\begin{array}{ll} \text{maximize} & 3x_1 + 5x_2 \\ \text{subject to} & x_1 + 2x_2 \leq 5 \\ & x_1 \qquad\; \leq 3 \\ & \qquad x_2 \leq 2 \\ & x_1,\ x_2 \geq 0. \end{array}$$

4.4 Solve the Klee–Minty problem (4.1) for $n = 3$.

4.5 Solve the 4 variable Klee-Minty problem using the online pivot tool:

www.princeton.edu/∼rvdb/JAVA/pivot/kleeminty.html

4.6 Consider the dictionary

$$\begin{aligned} \zeta &= -\sum_{j=1}^{n} \epsilon_j 10^{n-j} \left(\frac{1}{2} b_j - x_j \right) \\ w_i &= \sum_{j=1}^{i-1} \epsilon_i \epsilon_j 10^{i-j} (b_j - 2x_j) + (b_i - x_i) \qquad i = 1, 2, \ldots, n, \end{aligned}$$

where the b_i's are as in the Klee–Minty problem (4.2) and where each ϵ_i is ± 1. Fix k and consider the pivot in which x_k enters the basis and w_k leaves the basis. Show that the resulting dictionary is of the same form as before. How are the new ϵ_i's related to the old ϵ_i's?

4.7 Use the result of the previous problem to show that the Klee–Minty problem (4.2) requires $2^n - 1$ iterations.

4.8 Consider the Klee–Minty problem (4.2). Suppose that $b_i = \beta^{i-1}$ for some $\beta > 1$. Find the greatest lower bound on the set of β's for which the this problem requires $2^n - 1$ iterations.

4.9 Show that, for any integer n,

$$\frac{1}{2n} 2^{2n} \leq \binom{2n}{n} \leq 2^{2n}.$$

4.10 Consider a linear programming problem that has an optimal dictionary in which exactly k of the original slack variables are nonbasic. Show that by ignoring feasibility preservation of intermediate dictionaries this dictionary can be arrived at in exactly k pivots. Don't forget to allow for the fact that some pivot elements might be zero. *Hint: see Exercise 2.15.*

4.11 (MATLAB required.) Modify the MATLAB code posted at

www.princeton.edu/∼rvdb/LPbook/complexity/primalsimplex.m

so that data elements in A, b, and c are not rounded off to integers. Run the code and compare the results to those shown in Figure 4.3.

4.12 (MATLAB required.) Modify the MATLAB code posted at

www.princeton.edu/∼rvdb/LPbook/complexity/primalsimplex.m

so that the output is a log-log plot of the number of pivots versus the product m times n. Run the code and compare the results to those shown in Figure 4.3.

Notes

The first example of a linear programming problem in n variables and n constraints taking $2^n - 1$ iterations to solve was published by Klee & Minty (1972). Several researchers, including Smale (1983), Borgwardt (1982), Borgwardt (1987*a*), Adler & Megiddo (1985), and Todd (1986), have studied the average number of iterations. For a survey of probabilistic methods, the reader should consult Borgwardt (1987*b*).

Roughly speaking, a class of problems is said to have *polynomial complexity* if there is a polynomial p for which every problem of "size" n in the class can be solved by some algorithm in at most $p(n)$ operations. For many years it was unknown whether linear programming had polynomial complexity. The Klee–Minty examples show that, if linear programming is polynomial, then the simplex method is not the algorithm that gives the polynomial bound, since 2^n is not dominated by any polynomial. In 1979, Khachian (1979) gave a new algorithm for linear programming, called the *ellipsoid method*, which *is* polynomial and therefore established once and for all that linear programming has polynomial complexity. The collection of all problem classes having polynomial complexity is usually denoted by $\mathcal{P}$. A class of problems is said to belong to the class $\mathcal{NP}$ if, given a (proposed) solution, one can verify its optimality in a number of operations that is bounded by some polynomial in the "size" of the problem. Clearly, $\mathcal{P} \subset \mathcal{NP}$ (since, if we can solve from scratch in a polynomial amount of time, surely we can verify optimality at least that fast). An important problem in theoretical computer science is to determine whether or not $\mathcal{P}$ is a strict subset of $\mathcal{NP}$.

The study of how difficult it is to solve a class of problems is called *complexity theory*. Readers interested in pursuing this subject further should consult Garey & Johnson (1977).