up to $2K$, is there a subset that adds up to exactly K? This is known as the PARTITION problem, not to be confused with the k-PARTITION problem above. (Start from KNAPSACK, and add appropriate new items.)

(b) INTEGER KNAPSACK is this problem: We are given n integers and a goal K. We are asked whether we can choose *zero, one,* or *more* copies of each number so that the resulting multiset of integers adds up to K. Show that this problem is **NP**-complete. (Modify an instance of the ordinary KNAPSACK problem so that each item can be used at most once.)

9.5.34 Linear and integer programming. INTEGER PROGRAMMING is the problem of deciding whether a given system of linear equations has a nonnegative integer solution. It is of course **NP**-complete, as just about *all* **NP**-complete problems easily reduce to it... Actually, the difficult part here is showing that it is in **NP**; but it can be done, see

- C. H. Papadimitriou "On the complexity of integer programming", *J.ACM, 28,* 2, pp. 765-769, 1981.

In fact, in the paper above it is also shown that there is a pseudopolynomial algorithm for INTEGER PROGRAMMING when the number of equations is bounded by a constant, thus generalizing Proposition 9.4. (Naturally, the general INTEGER PROGRAMMING problem is strongly **NP**-complete.)

A different but equivalent formulation of INTEGER PROGRAMMING is in terms of a system of *inequalities* instead of equalities, and variables unrestricted in sign. For this form we have a more dramatic result: When the number of variables is bounded by a constant, there is a *polynomial-time* algorithm for the problem, based on the important *basis reduction technique*; see

- A. K. Lenstra, H. W. Lenstra, and L. Lovász "Factoring polynomials with rational coefficients," *Math. Ann, 261,* pp. 515–534, 1982, and
- M. Grötschel, L. Lovász, and A. Schrijver *Geometric Algorithms and Combinatorial Optimization,* Springer, Berlin, 1988.

In contrast, linear programming (the version in which we are allowed to have fractional solutions), is much easier: Despite the fact that the classical, empirically successful, and influential *simplex method,* see

- G. B. Dantzig *Linear Programming and Extensions,* Princeton Univ. Press, Princeton, N.J., 1963,

is exponential in the worst-case, polynomial-time algorithms have been discovered. The first polynomial algorithm for linear programming was the *ellipsoid method*

- L. G. Khachiyan "A polynomial algorithm for linear programming," *Dokl. Akad. Nauk SSSR, 244,* pp. 1093–1096, 1979. English Translation *Soviet Math. Doklad 20,* pp. 191–194, 1979;

while a more recent algorithm seems to be much more promising in practice:

- N. Karmarkar "A new polynomial-time algorithm for linear programming," *Combinatorica, 4,* pp. 373–395, 1984.

See also the books

- A. Schrijver *Theory of Linear and Integer Programming,* Wiley, New York, 1986, and
- C. H. Papadimitriou and K. Steiglitz *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

Problem: (a) Show that any instance of SAT can be expressed very easily as an instance of INTEGER PROGRAMMING with inequalities. Conclude that INTEGER PROGRAMMING is **NP**-complete even if the inequalities are known to have a fractional solution. (Start with an instance of SAT with at least two distinct literals per clause.)

(b) Express the existence of an integer flow of value K in a network with integer capacities as a set of linear inequalities.

(c) Is the MAX FLOW problem a special case of linear, or of integer programming? (On the surface it appears to be a special case of integer programming, since integer flows are required; but a little thought shows that the optimum solution will always be integer anyway—assuming all capacities are. So, the integrality constraint is superfluous.)