The greatest flood has the soonest ebb;
the sorest tempest the most sudden calm;
the hottest love the coldest end; and
from the deepest desire oftentimes ensues the deadliest hate.

— Socrates

Th' extremes of glory and of shame,
Like east and west, become the same.

— Samuel Butler, *Hudibras* Part II, Canto I (c. 1670)

Extremes meet, and there is no better example
than the haughtiness of humility.

— Ralph Waldo Emerson, "Greatness",
in *Letters and Social Aims* (1876)

⋆I Linear Programming

The maximum flow/minimum cut problem is a special case of a very general class of problems called *linear programming*. Many other optimization problems fall into this class, including minimum spanning trees and shortest paths, as well as several common problems in scheduling, logistics, and economics. Linear programming was used implicitly by Fourier in the early 1800s, but it was first formalized and applied to problems in economics in the 1930s by Leonid Kantorovich. Kantorivich's work was hidden behind the Iron Curtain (where it was largely ignored) and therefore unknown in the West. Linear programming was rediscovered and applied to shipping problems in the early 1940s by Tjalling Koopmans. The first complete algorithm to solve linear programming problems, called the *simplex method*, was published by George Dantzig in 1947. Koopmans first proposed the name "linear programming" in a discussion with Dantzig in 1948. Kantorovich and Koopmans shared the 1975 Nobel Prize in Economics "for their contributions to the theory of optimum allocation of resources". Dantzig did not; his work was apparently too pure. Koopmans wrote to Kantorovich suggesting that they refuse the prize in protest of Dantzig's exclusion, but Kantorovich saw the prize as a vindication of his use of mathematics in economics, which his Soviet colleagues had written off as "a means for apologists of capitalism".

A linear programming problem asks for a vector $x \in \mathbb{R}^d$ that maximizes (or equivalently, minimizes) a given linear function, among all vectors x that satisfy a given set of linear inequalities. The general form of a linear programming problem is the following:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^{d} c_j x_j \\
\text{subject to} \quad & \sum_{j=1}^{d} a_{ij} x_j \le b_i \quad \text{for each } i = 1 \,..\, p \\
& \sum_{j=1}^{d} a_{ij} x_j = b_i \quad \text{for each } i = p+1 \,..\, p+q \\
& \sum_{j=1}^{d} a_{ij} x_j \ge b_i \quad \text{for each } i = p+q+1 \,..\, n
\end{aligned}$$

Here, the input consists of a matrix $A = (a_{ij}) \in \mathbb{R}^{n \times d}$, a column vector $b \in \mathbb{R}^n$, and a row vector $c \in \mathbb{R}^d$. Each coordinate of the vector x is called a *variable*. Each of the linear inequalities is called a *constraint*. The function $x \mapsto c \cdot x$ is called the *objective function*. I will always use d to denote the number of variables, also known as the *dimension* of the problem. The number of constraints is usually denoted n.

A linear programming problem is said to be in *canonical form*[1] if it has the following structure:

$$\begin{aligned} \text{maximize } & \sum_{j=1}^{d} c_j x_j \\ \text{subject to } & \sum_{j=1}^{d} a_{ij} x_j \le b_i \quad \text{for each } i = 1 \mathinner{..} n \\ & x_j \ge 0 \quad \text{for each } j = 1 \mathinner{..} d \end{aligned}$$

We can express this canonical form more compactly as follows. For two vectors $x = (x_1, x_2, \ldots, x_d)$ and $y = (y_1, y_2, \ldots, y_d)$, the expression $x \ge y$ means that $x_i \ge y_i$ for every index i.

$$\boxed{\begin{aligned} \max\quad & c \cdot x \\ \text{s.t. } & Ax \le b \\ & x \ge 0 \end{aligned}}$$

Any linear programming problem can be converted into canonical form as follows:

- For each variable x_j, add the equality constraint $x_j = x_j^+ - x_j^-$ and the inequalities $x_j^+ \ge 0$ and $x_j^- \ge 0$.
- Replace any equality constraint $\sum_j a_{ij} x_j = b_i$ with two inequality constraints $\sum_j a_{ij} x_j \ge b_i$ and $\sum_j a_{ij} x_j \le b_i$.
- Replace any upper bound $\sum_j a_{ij} x_j \ge b_i$ with the equivalent lower bound $\sum_j -a_{ij} x_j \le -b_i$.

This conversion potentially double the number of variables and the number of constraints; fortunately, it is rarely necessary in practice.

Another useful format for linear programming problems is *slack form*[2], in which every inequality is of the form $x_j \ge 0$:

$$\boxed{\begin{aligned} \max\quad & c \cdot x \\ \text{s.t. } & Ax = b \\ & x \ge 0 \end{aligned}}$$

It's fairly easy to convert any linear programming problem into slack form. Slack form is especially useful in executing the simplex algorithm (which we'll see in the next lecture).

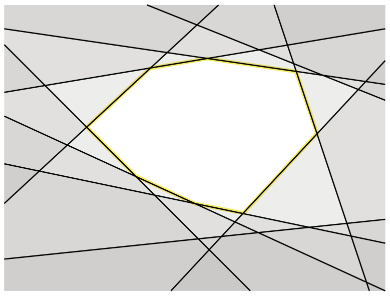
I.1 The Geometry of Linear Programming

A point $x \in \mathbb{R}^d$ is *feasible* with respect to some linear programming problem if it satisfies all the linear constraints. The set of all feasible points is called the *feasible region* for that linear program. The feasible region has a particularly nice geometric structure that lends some useful intuition to the linear programming algorithms we'll see later.

Any linear equation in d variables defines a *hyperplane* in $\mathbb{R}^d$; think of a line when $d = 2$, or a plane when $d = 3$. This hyperplane divides $\mathbb{R}^d$ into two *halfspaces*; each halfspace is the set of points that satisfy some linear inequality. Thus, the set of feasible points is the intersection of several hyperplanes

[1]Confusingly, some authors call this *standard form*.
[2]Confusingly, some authors call this *standard form*.

(one for each equality constraint) and halfspaces (one for each inequality constraint). The intersection of a finite number of hyperplanes and halfspaces is called a *polyhedron*. It's not hard to verify that any halfspace, and therefore any polyhedron, is *convex*—if a polyhedron contains two points x and y, then it contains the entire line segment $\overline{xy}$.

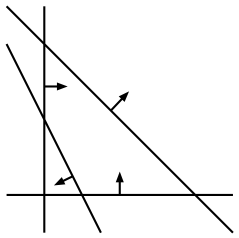

A two-dimensional polyhedron (white) defined by 10 linear inequalities.

By rotating $\mathbb{R}^d$ (or choosing a coordinate frame) so that the objective function points downward, we can express *any* linear programming problem in the following geometric form:

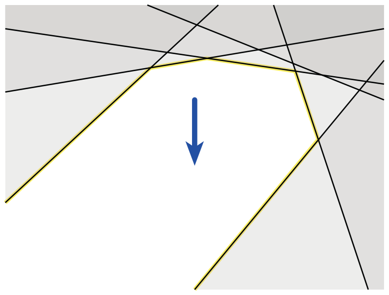
$$\boxed{\text{Find the lowest point in a given polyhedron.}}$$

With this geometry in hand, we can easily picture two pathological cases where a given linear programming problem has no solution. The first possibility is that there are no feasible points; in this case the problem is called *infeasible*. For example, the following LP problem is infeasible:

$$\begin{aligned} \text{maximize } & x - y \\ \text{subject to } & 2x + y \le 1 \\ & x + y \ge 2 \\ & x, y \ge 0 \end{aligned}$$


An infeasible linear programming problem; arrows indicate the constraints.

The second possibility is that there are feasible points at which the objective function is arbitrarily large; in this case, we call the problem *unbounded*. The same polyhedron could be unbounded for some objective functions but not others, or it could be unbounded for every objective function.


A two-dimensional polyhedron (white) that is unbounded downward but bounded upward.

I.2 Example 1: Shortest Paths

We can compute the length of the shortest path from s to t in a weighted directed graph by solving the following very simple linear programming problem.

$$\begin{array}{ll} \text{maximize} & d_t \\ \text{subject to} & d_s = 0 \\ & d_v - d_u \le \ell_{u\to v} \quad \text{for every edge } u\to v \end{array}$$

Here, $\ell_{u\to v}$ is the length of the edge $u\to v$. Each variable d_v represents a tentative shortest-path distance from s to v. The constraints mirror the requirement that every edge in the graph must be relaxed. These relaxation constraints imply that in any feasible solution, d_v is *at most* the shortest path distance from s to v. Thus, somewhat counterintuitively, we are correctly *maximizing* the objective function to compute the *shortest* path! In the optimal solution, the objective function d_t is the actual shortest-path distance from s to t, but for any vertex v that is not on the shortest path from s to t, d_v may be an underestimate of the true distance from s to v. However, we can obtain the true distances from s to every other vertex by modifying the objective function:

$$\begin{array}{ll} \text{maximize} & \sum_v d_v \\ \text{subject to} & d_s = 0 \\ & d_v - d_u \le \ell_{u\to v} \quad \text{for every edge } u\to v \end{array}$$

There is another formulation of shortest paths as an LP minimization problem using an indicator variable $x_{u\to v}$ for each edge $u\to v$.

$$\begin{array}{ll} \text{minimize} & \sum_{u\to v} \ell_{u\to v} \cdot x_{u\to v} \\ \text{subject to} & \sum_u x_{u\to s} - \sum_w x_{s\to w} = 1 \\ & \sum_u x_{u\to t} - \sum_w x_{t\to w} = -1 \\ & \sum_u x_{u\to v} - \sum_w x_{v\to w} = 0 \quad \text{for every vertex } v \ne s, t \\ & x_{u\to v} \ge 0 \quad \text{for every edge } u\to v \end{array}$$

Intuitively, $x_{u\to v} = 1$ means $u\to v$ lies on the shortest path from s to t, and $x_{u\to v} = 0$ means $u\to v$ does not lie on this shortest path. The constraints merely state that the path should start at s, end at t, and either pass through or avoid every other vertex v. Any path from s to t—in particular, the shortest path—clearly implies a feasible point for this linear program.

However, there are other feasible solutions, possibly even *optimal* solutions, with non-integral values that do not represent paths. Nevertheless, there is always an optimal solution in which every x_e is either 0 or 1 and the edges e with $x_e = 1$ comprise the shortest path. (This fact is by no means obvious, but a proof is beyond the scope of these notes.) Moreover, in any optimal solution, even if not every x_e is an integer, the objective function gives the shortest path distance!

I.3 Example 2: Maximum Flows and Minimum Cuts

Recall that the input to the maximum (s,t)-flow problem consists of a weighted directed graph $G = (V, E)$, two special vertices s and t, and a function assigning a non-negative *capacity* c_e to each edge e. Our task is to choose the flow f_e across each edge e, as follows:

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_w f_{s\to w} - \sum_u f_{u\to s} & \\ \text{subject to} & \displaystyle\sum_w f_{v\to w} - \sum_u f_{u\to v} = 0 & \text{for every vertex } v \neq s,t \\ & f_{u\to v} \leq c_{u\to v} & \text{for every edge } u\to v \\ & f_{u\to v} \geq 0 & \text{for every edge } u\to v \end{array}$$

Similarly, the minimum cut problem can be formulated using 'indicator' variables similarly to the shortest path problem. We have a variable S_v for each vertex v, indicating whether $v \in S$ or $v \in T$, and a variable $X_{u\to v}$ for each edge $u\to v$, indicating whether $u \in S$ and $v \in T$, where (S,T) is some (s,t)-cut.[3]

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{u\to v} c_{u\to v} \cdot X_{u\to v} & \\ \text{subject to} & X_{u\to v} + S_v - S_u \geq 0 & \text{for every edge } u\to v \\ & X_{u\to v} \geq 0 & \text{for every edge } u\to v \\ & S_s = 1 & \\ & S_t = 0 & \end{array}$$

Like the minimization LP for shortest paths, there can be optimal solutions that assign fractional values to the variables. Nevertheless, the minimum value for the objective function is the cost of the minimum cut, and there is an optimal solution for which every variable is either 0 or 1, representing an actual minimum cut. No, this is not obvious; in particular, my claim is not a proof!

I.4 Linear Programming Duality

Each of these pairs of linear programming problems is related by a transformation called *duality*. For any linear programming problem, there is a corresponding dual linear program that can be obtained by a mechanical translation, essentially by swapping the constraints and the variables. The translation is simplest when the LP is in canonical form:

$$\underset{\textbf{Primal }(\Pi)}{\boxed{\begin{array}{rl} \max & c\cdot x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}}} \iff \underset{\textbf{Dual }(\amalg)}{\boxed{\begin{array}{rl} \min & y\cdot b \\ \text{s.t.} & yA \geq c \\ & y \geq 0 \end{array}}}$$

We can also write the dual linear program in exactly the same canonical form as the primal, by swapping the coefficient vector c and the objective vector b, negating both vectors, and replacing the constraint matrix A with its negative transpose.[4]

$$\underset{\textbf{Primal }(\Pi)}{\boxed{\begin{array}{rl} \max & c\cdot x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}}} \iff \underset{\textbf{Dual }(\amalg)}{\boxed{\begin{array}{rl} \max & -b^\top \cdot y^\top \\ \text{s.t.} & -A^\top y^\top \leq -c \\ & y^\top \geq 0 \end{array}}}$$

[3]These two linear programs are not quite *syntactic* duals; I've added two redundant variables S_s and S_t to the min-cut program to increase readability.

[4]For the notational purists: In these formulations, x and b are column vectors, and y and c are row vectors. This is a somewhat nonstandard choice. Yes, that means the dot in $c\cdot x$ is redundant. Sue me.

Written in this form, it should be immediately clear that duality is an *involution*: The dual of the dual linear program $\amalg$ is identical to the primal linear program Π. The choice of which LP to call the 'primal' and which to call the 'dual' is totally arbitrary.[5]

The Fundamental Theorem of Linear Programming. *A linear program Π has an optimal solution x^* if and only if the dual linear program $\amalg$ has an optimal solution y^* such that $c \cdot x^* = y^* A x^* = y^* \cdot b$.*

The weak form of this theorem is trivial to prove.

Weak Duality Theorem. *If x is a feasible solution for a canonical linear program Π and y is a feasible solution for its dual $\amalg$, then $c \cdot x \le yAx \le y \cdot b$.*

Proof: Because x is feasible for Π, we have $Ax \le b$. Since y is positive, we can multiply both sides of the inequality to obtain $yAx \le y \cdot b$. Conversely, y is feasible for $\amalg$ and x is positive, so $yAx \ge c \cdot x$. $\square$

It immediately follows that if $c \cdot x = y \cdot b$, then x and y are optimal solutions to their respective linear programs. This is in fact a fairly common way to prove that we have the optimal value for a linear program.

I.5 Duality Example

Before I prove the stronger duality theorem, let me first provide some intuition about where this duality thing comes from in the first place.[6] Consider the following linear programming problem:

$$\begin{array}{rl}
\text{maximize} & 4x_1 + x_2 + 3x_3 \\
\text{subject to} & x_1 + 4x_2 \le 2 \\
& 3x_1 - x_2 + x_3 \le 4 \\
& x_1, x_2, x_3 \ge 0
\end{array}$$

Let σ^* denote the optimum objective value for this LP. The feasible solution $x = (1, 0, 0)$ gives us a lower bound $\sigma^* \ge 4$. A different feasible solution $x = (0, 0, 3)$ gives us a better lower bound $\sigma^* \ge 9$. We could play this game all day, finding different feasible solutions and getting ever larger lower bounds. How do we know when we're done? Is there a way to prove an *upper* bound on σ^*?

In fact, there is. Let's multiply each of the constraints in our LP by a new non-negative scalar value y_i:

$$\begin{array}{rl}
\text{maximize} & 4x_1 + x_2 + 3x_3 \\
\text{subject to} & y_1(x_1 + 4x_2) \le 2y_1 \\
& y_2(3x_1 - x_2 + x_3) \le 4y_2 \\
& x_1, x_2, x_3 \ge 0
\end{array}$$

Because each y_i is non-negative, we do not reverse any of the inequalities. Any feasible solution (x_1, x_2, x_3) must satisfy both of these inequalities, so it must also satisfy their sum:

$$(y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2 x_3 \le 2y_1 + 4y_2.$$

[5]For historical reasons, maximization LPs tend to be called 'primal' and minimization LPs tend to be called 'dual'. This is a pointless religious tradition, nothing more. Duality is a relationship between LP problems, not a type of LP problem.

[6]This example is taken from Robert Vanderbei's excellent textbook *Linear Programming: Foundations and Extensions* [Springer, 2001], but the idea appears earlier in Jens Clausen's 1997 paper 'Teaching Duality in Linear Programming: The Multiplier Approach'.

Now suppose that each y_i is larger than the ith coefficient of the objective function:

$$y_1 + 3y_2 \geq 4, \qquad 4y_1 - y_2 \geq 1, \qquad y_2 \geq 3.$$

This assumption lets us derive an upper bound on the objective value of *any* feasible solution:

$$4x_1 + x_2 + 3x_3 \;\leq\; (y_1 + 3y_2)x_1 + (4y_1 - y_2)x_2 + y_2 x_3 \;\leq\; 2y_1 + 4y_2. \tag{$*$}$$

In particular, by plugging in the optimal solution (x_1^*, x_2^*, x_3^*) for the original LP, we obtain the following upper bound on σ^*:

$$\sigma^* \;=\; 4x_1^* + x_2^* + 3x_3^* \;\leq\; 2y_1 + 4y_2.$$

Now it's natural to ask how tight we can make this upper bound. How small can we make the expression $2y_1 + 4y_2$ without violating any of the inequalities we used to prove the upper bound? This is just another linear programming problem.

$$\begin{array}{rr} \text{minimize} & 2y_1 + 4y_2 \\ \text{subject to} & y_1 + 3y_2 \geq 4 \\ & 4y_1 - \;\; y_2 \geq 1 \\ & y_2 \geq 3 \\ & y_1, y_2 \geq 0 \end{array}$$

In fact, this is precisely the dual of our original linear program! Moreover, inequality $(*)$ is just an instantiation of the Weak Duality Theorem.

I.6 Strong Duality

The Fundamental Theorem can be rephrased in the following form:

Strong Duality Theorem. *If x^* is an optimal solution for a canonical linear program Π, then there is an optimal solution y^* for its dual $\amalg$, such that $c \cdot x^* = y^* A x^* = y^* \cdot b$.*

Proof (Sketch): I'll prove the theorem only for ***non-degenerate*** linear programs, in which (a) the optimal solution (if one exists) is a unique vertex of the feasible region, and (b) at most d constraint planes pass through any point. These non-degeneracy assumptions are relatively easy to enforce in practice and can be removed from the proof at the expense of some technical detail. I will also prove the theorem only for the case $n \geq d$; the argument for under-constrained LPs is similar (if not simpler).

Let x^* be the optimal solution for the linear program Π; non-degeneracy implies that this solution is unique, and that exactly d of the n linear constraints are satisfied with equality. Without loss of generality (by permuting the rows of A), we can assume that these are the first d constraints.

So let $A_\bullet$ be the $d \times d$ matrix containing the first d rows of A, and let $A_\circ$ denote the other $n - d$ rows. Similarly, partition b into its first d coordinates $b_\bullet$ and everything else $b_\circ$. Thus, we have partitioned the inequality $Ax^* \leq b$ into a system of equations $A_\bullet x^* = b_\bullet$ and a system of strict inequalities $A_\circ x^* < b_\circ$.

Now let $y^* = (y_\bullet^*, y_\circ^*)$ where $y_\bullet^* = cA_\bullet^{-1}$ and $y_\circ^* = 0$. We easily verify that $y^* \cdot b = c \cdot x^*$:

$$y^* \cdot b \;=\; y_\bullet^* \cdot b_\bullet + y_\circ^* \cdot b_\circ \;=\; y_\bullet^* \cdot b_\bullet \;=\; cA_\bullet^{-1} b_\bullet \;=\; c \cdot x^*.$$

(The existence of the inverse matrix $A_\bullet^{-1}$ follows from our non-degeneracy assumption.) Similarly, it's easy to verify that $y^* A \geq c$:

$$y^* A \;=\; y_\bullet^* A_\bullet^* + y_\circ^* A_\circ^* \;=\; y_\bullet^* A_\bullet^* \;=\; c.$$

Once we prove that y^* is non-negative, and therefore feasible, the Weak Duality Theorem implies the result. We chose $y^*_\circ = 0$. As we will see below, the optimality of x^* implies the strict inequality $y^*_\bullet > 0$—we had to use optimality somewhere! This is the hardest part of the proof.

The key insight is to give a geometric interpretation to the vector $y^*_\bullet = cA_\bullet^{-1}$. Each row of the linear system $A_\bullet x^* = b_\bullet$ describes a hyperplane $a_i \cdot x^* = b_i$ in $\mathbb{R}^d$. The vector a_i is normal to this hyperplane and points *out* of the feasible region. The vectors $a_1, \ldots, a_d$ are linearly independent (by non-degeneracy) and thus describe a coordinate frame for the vector space $\mathbb{R}^d$. The definition of $y^*_\bullet$ can be rewritten as follows:

$$c = y^*_\bullet A_\bullet = \sum_{i=1}^{d} y^*_i a_i.$$

In other words, $y^*_\bullet$ lists the coefficients of the objective vector c in the coordinate frame $a_1, \ldots, a_d$.

The point x^* lies on exactly d constraint hyperplanes; any $d-1$ of these hyperplanes determine a line through x^*. For each $1 \le i \le d$, let ℓ_i denote the line that lies on all but the ith constraint plane, and let v_i denote a vector based at x^* that points into the halfspace $a_i x \le b_i$ along the line ℓ_i. This vector lies along an edge of the feasible polytope. For all $j \ne i$, we have $a_j \cdot v_i = 0$. Thus, we can write

$$A_\bullet v_i = (0, \ldots, 0,\ a_i \cdot v_i,\ 0, \ldots, 0)^\top$$

where the scalar $a_i \cdot v_i$ appears in the ith coordinate. It follows that

$$c \cdot v_i = y^*_\bullet A_\bullet v_i = y^*_i (a_i \cdot v_i).$$

The optimality of x^* implies that $c \cdot v_i < 0$, and because v_i points into the feasible region while a_i points out, we have $a_i \cdot v_i < 0$. We conclude that $y^*_i > 0$. We're done! □

Exercises

1. (a) Describe precisely how to dualize a linear program written in slack form:

$$\boxed{\begin{array}{rl} \max & c \cdot x \\ \text{s.t.} & Ax = b \\ & x \ge 0 \end{array}}$$

   (b) Describe precisely how to dualize a linear program written in general form:

$$\begin{array}{rll} \text{maximize} & \displaystyle\sum_{j=1}^{d} c_j x_j & \\ \text{subject to} & \displaystyle\sum_{j=1}^{d} a_{ij} x_j \le b_i & \text{for each } i = 1 \,..\, p \\ & \displaystyle\sum_{j=1}^{d} a_{ij} x_j = b_i & \text{for each } i = p+1 \,..\, p+q \\ & \displaystyle\sum_{j=1}^{d} a_{ij} x_j \ge b_i & \text{for each } i = p+q+1 \,..\, n \end{array}$$

   In both cases, keep the number of dual variables as small as possible.

2. A matrix $A = (a_{ij})$ is *skew-symmetric* if and only if $a_{ji} = -a_{ij}$ for all indices $i \neq j$; in particular, every skew-symmetric matrix is square. A canonical linear program $\max\{c \cdot x \mid Ax \leq b; x \geq 0\}$ is *self-dual* if the matrix A is skew-symmetric and the objective vector c is *equal* to the constraint vector b.

   (a) Prove any self-dual linear program Π is equivalent to its dual program $\amalg$.

   (b) Show that any linear program Π with d variables and n constraints can be transformed into a self-dual linear program with $n + d$ variables and $n + d$ constraints. The optimal solution to the self-dual program should include both the optimal solution for Π (in d of the variables) and the optimal solution for the dual program $\amalg$ (in the other n variables).

3. (a) Model the maximum-cardinality bipartite matching problem as a linear programming problem. The input is a bipartite graph $G = (U \cup V; E)$, where $E \subseteq U \times V$; the output is the largest matching in G. Your linear program should have one variable for each edge.

   (b) Now dualize the linear program from part (a). What do the dual variables represent? What does the objective function represent? What problem is this!?

4. An *integer program* is a linear program with the additional constraint that the variables must take only integer values.

   (a) Prove that deciding whether an integer program has a feasible solution is NP-complete.

   (b) Prove that finding the optimal feasible solution to an integer program is NP-hard.

   [Hint: Almost any NP-hard decision problem can be formulated as an integer program. Pick your favorite.]

*5. *Helly's theorem* states that for any collection of convex bodies in $\mathbb{R}^d$, if every $d + 1$ of them intersect, then there is a point lying in the intersection of all of them. Prove Helly's theorem for the special case where the convex bodies are halfspaces. Equivalently, show that if a system of linear inequalities $Ax \leq b$ does not have a solution, then we can select $d + 1$ of the inequalities such that the resulting subsystem also does not have a solution. *[Hint: Construct a dual LP from the system by choosing a 0 cost vector.]*