Linear Programming:

The profit maximization problem in DPV textbook section 7.1.1 together with Figure 7.1
Solving linear programs:

- *Feasible region* is the set of values of variables that satisfy all constraints. Feasible region can be:
  - Empty, e.g., constraints $x_1 \geqslant 2$ and $x_1 \leqslant 1$; no solution to linear program in this case.
  - Unbounded, e.g., constraints $x_1 \geqslant 0, x_2 \geqslant 0, x_1 + x_2 \geqslant 10$; no solution to linear program in this case (value of objective function can be arbitrarily large so there is no maximum) – if objective function is minimization, then treat this as bounded.
  - Bounded, e.g., constraints $x_1 \geqslant 0, x_2 \geqslant 0, x_1 + x_2 \leqslant 10$; either one or infinitely many solutions to linear program in this case, depending on objective function, e.g., with same constraints, unique solution to maximize objective function $2x_1 - x_2$ (solution: $x_1 = 10, x_2 = 0$) but infinitely many solutions to maximize objective function $x_1 + x_2$ (any two nonnegative values that add up to 10).
- Simplex method solves linear programs by, intuitively, moving from vertex to vertex along the boundary of the feasible region, using algebraic manipulations similar to Gaussian elimination. It runs in worst-case exponential time but in practice, this behaviour is very rarely encountered (most of the time it is quite efficient).
- *Interior point* methods solve linear programs in worst-case polynomial time but are just recently starting to be competitive with Simplex in practice.

Examples:
Political Advertising problem (from "Introduction to Algorithms, 2nd ed." by Cormen et al., pp. 770-772): A political party can advertise on four different platforms: building roads, gun control, farm subsidies, and gasoline tax. Voters come from three types of ridings: urban, suburban, and rural. Assume that advertising can only be done globally: all advertising is seen in all three types of ridings.

Market research has provided the party with the following information: for each advertising platform and riding type, this table summarizes number of voters gained or lost in ridings of that type, for each dollar of advertising spent on that platform.

	urban	suburb	rural
roads	-2	5	3
guns	8	2	-5
farm	0	0	10
gas	10	0	-2

Leaders of the party have figured out that the party needs to gain at least 50,000 urban voters, 100,000 suburban voters, and 25,000 rural voters. Your task is to figure out how much to spend on advertising for each platform in order to gain the required number of votes in each type of riding, while spending as little as possible overall.

Problem representation: Here is one way to represent the problem.

What are we looking for exactly? Amount to advertise on each platform. So introduce variables:

- x_1 = advertising budget on building roads
- x_2 = advertising budget on gun controls
- x_3 = advertising budget on farm subsidies
- x_4 = advertising budget on gasoline tax

What's our goal? Spend as little as possible, in other words, minimize

$$x_1 + x_2 + x_3 + x_4$$

What are the constraints? The need to gain some number of voters in each riding type, which can be expressed by linear inequalities:

- $-2x_1 + 8x_2 + 0x_3 + 10x_4 \geqslant 50,000$ (for urban ridings)
- $5x_1 + 2x_2 + 0x_3 + 0x_4 \geqslant 100,000$ (for suburban ridings)
- $3x_1 - 5x_2 + 10x_3 - 2x_4 \geqslant 25,000$ (for rural ridings)

Anything else? Numerically, we cannot spend negative amounts, so

$$x_1, x_2, x_3, x_4 \geqslant 0$$

This is known as a *linear program.*

LPs in standard form:

In general we can represent a linear program as an optimization problem consisting some variables: $x_1, x_2, ..., x_n$ (real numbers) with an objective function: $c_1x_1 + c_2x_2 + ... + c_nx_n$ where c_i are real number constants. Moreover, there are some linear constraints that should be satisfied: $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leqslant / = / \geqslant b_i$ for $i = 1, 2, ..., m$. Often written $Ax \leqslant / = / \geqslant b$ for matrix A, vector of variables x and vector of constants b.
Example for political advertising (including non-negativity):

$$\begin{pmatrix} -2 & 8 & 0 & 10 \\ 5 & 2 & 0 & 0 \\ 3 & -5 & 10 & -2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \geqslant \begin{pmatrix} 50 \\ 100 \\ 25 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Goal: Identify the real values of x_i's that maximize/minimize objective function and satisfy all constraints.

Network Flow: Given network $N = (V, E)$ with capacities $c(e)$ for $e \in E$, construct a linear program with variables f_e (one for each $e \in E$):

maximize: $\sum_{(s,u)inE} f_(s, u)$
subject to:

- $f_e \geqslant 0$ for all $e \in E$
- $f_e \leqslant c(e)$ for all $e \in E$
- $\sum_{(u,v)inE} f_(u, v) - \sum_{(v,u)inE} f_(v, u) = 0$ for all $v \in V$

This is a direct re-statement of network flow problem, i.e., any valid flow in N yield values for f_e that satisfy all constraints (*feasible* values), and any feasible values for f_e is a valid flow in N. So finding max flow in N equivalent to maximizing objective function.

Might this allow us to solve max flow problem more easily or faster?
Unfortunately not: solving LP no more efficient than solving max flow (not surprisingly since problem is more

general).

One more catch: LP solution cannot guarantee integer flow on all edges (even when all edge capacities are integer), in contrast with Ford-Fulkerson algorithm that guarantees integer flows in that case.
Shortest *s*-*t* path: Given graph $G = (V, E)$ with weights $w(e)$ for all $e \in E$, construct linear program with variables d_v for each $v \in V$:
maximize: d_t
subject to:

- $d_v \leqslant d_u + w(u, v)$ for each $(u, v) \in E$
- $d_s = 0$
- $d_v \geqslant 0$ for each $v \in V$

Minimizing d_t doesn't work because it allows settings of d_v smaller than true distances (e.g., $d_v = 0$ for all $v \in V$). Maximizing works because constraints force d_v to be no more than shortest distance and maximization forces d_v to be at least shortest distance, for all v.

Example: Graph with vertices $V = \{s, a, b, t\}$ and edges (s, a) with weight 1, (s, t) with weight 6, (a, b) with weight 2, (a, t) with weight 4, (b, t) with weight 1.
Linear program (each edge yields two constraints):
maximize: d_t
subject to:

- $d_s = 0$
- $0 \leqslant d_a \leqslant d_s + 1$ and $0 \leqslant d_s \leqslant d_a + 1$
- $0 \leqslant d_b \leqslant d_a + 2$ and $0 \leqslant d_a \leqslant d_b + 2$
- $0 \leqslant d_t \leqslant d_s + 6$ and $0 \leqslant d_s \leqslant d_t + 6$
- $0 \leqslant d_t \leqslant d_a + 4$ and $0 \leqslant d_a \leqslant d_t + 4$
- $0 \leqslant d_t \leqslant d_b + 1$ and $0 \leqslant d_b \leqslant d_t + 1$

Integer programming: more restricted version where all constants and variables are integers. NP-complete (no efficient algorithm).
Example: Minimum Vertex Cover: Given an undirected graph $G = (V, E)$, Identify a subset of vertices C that *covers* every edge (i.e., each edge has at least one endpoint in C), with minimum size.

We represent this problem as an integer program: use variable x_i for each vertex $v_i \in V$
minimize: $x_1 + x_2 + ... + x_n$
subject to:

- $x_i + x_j \geqslant 1$ for all $(v_i, v_j) \in E$
- $x_i \in \{0, 1\}$ for all $v_i \in V$

This 0-1 integer program is completely equivalent to original problem, through correspondence: v_i in cover iff $x_i = 1$. In more detail:

- Any vertex cover C yields feasible solution $x_i = 1$ if $v_i \in C$, 0 if $v_i \notin C$ because each constraint $x_i + x_j \geqslant 1$ satisfied (C must include one endpoint of each edge).

- Any feasible solution to LP yields vertex cover $C = \{v_i \in V : x_i = 1\}$ because for each edge (v_i, v_j), constraint $x_i + x_j \geqslant 1$ ensures C contains at least one of v_i, v_j.

Unfortunately, Integer Programming (IP) is NP-hard, so the problem cannot be solved in polytime this way. In the next lecture we will propose an algorithm to approximate the solution.