Lecture slides

Math 30210, Fall 2014

December 10, 2014

A diet problem [August 27]

I want to construct a nutritious diet using four basic foods: brownies, ice cream, Dr. Pepper, and cheesecake. I want to keep track of calories, chocolate, sugar and fat. The USDA tells me that

- one unit of brownies gives me 400 calories, 3g of chocolate, 2g of sugar and 2g of fat;
- one unit of ice cream gives me 200 calories, 2g of chocolate, 2g of sugar and 4g of fat;
- the numbers for Dr. Pepper are 150, 0, 4, and 1; and
- the numbers for cheesecake are 500, 0, 4, and 5.

My dietician tells me that a healthy daily diet requires at least 500 calories, at least 6 grams of chocolate, at least 10 grams of sugar, and at least 8 grams of fat.
A unit of each of the basic foods costs 50c, 20c, 30c, 80c, respectively.

What diet will satisfy all my dietician's requirements, while keeping the cost as low as possible?

A production problem (Ex 2.3.1 of text) [August 27]

A company produces small rowboats and canoes. Production consumes resources: aluminium, machine time, and labour time. The following table summarizes how each production item consumes resources, and how much profit the sale of each one makes:

	Aluminium	Machine time	Labour time	Profit
Rowboat	50 lb	6 min.	3 hr	$50
Canoe	30 lb	5 min.	5 hr	$60

In the near future the company has 1 ton of aluminium, 5 hours of machine time and 200 hours of labour time available. Assuming it can sell all that it produces, how many rowboats and canoes should it aim to produce to maximize profit?

A transportation problem [August 27]

The US airforce has bases in San Diego, Portland (ME), Tulsa and Kokomo (IN). It has supply warehouses in Seattle, Manhattan (KS) and San Antonio. Rations need to be shipped from the warehouses to the bases. The following table shows

- the available supply at each warehouse location;
- the demand at each base; and
- the cost of shipping a unit of rations from each warehouse to each base.

	San Diego	Portland	Tulsa	Kokomo	Supply
Seattle	3	8	5	4	800
Manhattan	5	4	3	3	600
San Antonio	4	5	3	5	900
Demand	400	500	300	1000	

What is a viable shipping scheme which satisfies each of the bases' demands, and has minimum cost?

A recreational problem [August 29]

What is the smallest number of queens that can be placed on a chessboard, so that every square is either occupied by a queen, or attacked by a queen?

What if no two of the queens are allowed to attack each other?

A production problem (Q23, p.33 of text) [August 29]

Weekly production schedule for chairs & sofas sought, to maximize profit

Data	Fabric	Wood	Labour	Profit
Chair	3	6	9	70
Sofa	8	5	4	60
Availability	96	90	120	

Mathematical problem: $x = \#(\text{chairs})$, $y = \#(\text{sofas})$. Maximize

$$70x + 60y$$

subject to constraints

$$\begin{aligned} 3x + 8y &\leq 96 \\ 6x + 5y &\leq 90 \\ 9x + 4y &\leq 120 \\ x, y &\geq 0 \text{ (and both integers)} \end{aligned}$$

Modified problem (Q23, p.33 of text) [September 1]

Still want weekly production schedule for chairs & sofas, to maximize profit

Data	Fabric	Wood	Labour	Profit
Chair	3	6	9	70
Sofa	8	5	4	60
Availability	96	90	120	

But now labour (and maybe materials) cost money!

- Case 1: labour cost \$5 per hour
- Case 2: labour cost \$5 per hour for the first \$80 hours, and \$8 per hour for the remaining \$40
- Case 3: labour is free, but now wood costs \$3 a unit for the first 60 units, and \$2 a unit for the next 30

Third modification of (Q23, p.33 of text) [September 3]

Want weekly production schedule for chairs & sofas, to maximize profit

Data	Fabric	Wood	Labour	Profit
Chair	3	6	9	70
Sofa	8	5	4	60
Availability	96	90	120	

Extra element: wood costs \$3 a unit for the first 60 units, and \$2 a unit for the next 30

Formulation of third modification [September 3]

Decision variables: $x = \#$(chairs produced), $y = \#$(sofas produced),
$w_e = \#$(units of expensive wood purchaed),
$w_c = \#$(units of cheap wood purchaed)

Auxiliary variable: b

Objective: Maximize profit $p = 70x + 60y - 3w_e - 2w_c$

subject to constraints

$$
\begin{array}{rrrrrcll}
3x & +8y & & & & \leq & 96 & \text{(fabric)} \\
6x & +5y & -w_e & -w_c & & \leq & 0 & \text{(wood)} \\
9x & +4y & & & & \leq & 120 & \text{(labour)} \\
 & & w_e & & & \leq & 60 & \text{(wood)} \\
 & & & w_c & & \leq & 30 & \text{(wood)} \\
 & & w_e & & -60b & \geq & 0 & \text{(auxiliary)} \\
 & & & w_c & -30b & \leq & 0 & \text{(auxiliary)} \\
 & & & & b & \leq & 1 & \text{(auxiliary)}
\end{array}
$$

as well as all variables non-negative, integral.

Terminology [September 3]

Linear programming (LP) problem: any problem that asks to maximize or minimize an objective function that is a *linear* combination of some variables, subject to some constraints that are *linear* equalities and inequalities involving the variables

Standard form of an LP:

- *minimization*
- all variables non-negative
- all constraints *equalities*

Example: Variables x_1, x_2, x_3, x_4, x_5

Minimize $z = 6x_1 + 3x_2 - 20x_5 - 22$

subject to constraints

$$\begin{array}{rrrrrcr} 3x_1 & +2x_2 & -7x_3 & & -.6x_5 & = & 11 \\ & -x_2 & -x_3 & & -4.7x_5 & = & 0 \\ -5x_1 & -2x_2 & +3x_3 & -4x_4 & +x_5 & = & -27 \end{array}$$

as well as $x_1, x_2, x_3, x_4, x_5 \geq 0$.

Standard form of third modification [September 3]

Variables: x, y, w_e, w_c, b (original variables), x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 (slack variables)

Objective: Minimize $z = -70x - 60y + 3w_e + 2w_c$

subject to constraints

$$
\begin{array}{ccccccccccccccc}
3x & +8y & & & & +x_1 & & & & & & & & = & 96 \\
6x & +5y & -w_e & -w_c & & & +x_2 & & & & & & & = & 0 \\
9x & +4y & & & & & & +x_3 & & & & & & = & 120 \\
 & & w_e & & & & & & +x_4 & & & & & = & 60 \\
 & & & w_c & & & & & & +x_5 & & & & = & 30 \\
 & & w_e & & -60b & & & & & & -x_6 & & & = & 0 \\
 & & & w_c & -30b & & & & & & & +x_7 & & = & 0 \\
 & & & & b & & & & & & & & +x_8 & = & 1
\end{array}
$$

as well as all variables non-negative, and all original variables integral.

Dealing with unrestricted variables [September 3]

A baker has 250g of yeast in stock, and 70k of flour. He can make baguettes, that use 3g of yeast, 1k of flour, and sell for \$2, and sourdough loafs, that use 2g of yeast, 1k of flour, and sell for \$1.50. If necessary he can buy in flour at \$1 per k, and at the end of the day he can sell any surplus flour he has at \$1 per k. What is is maximum possible profit?

Variables: b (baguettes), s (sourdough), f (flour in excess of 7k used: if $f > 0$, fk of extra flour is purchased; if $f < 0$, $-f$k is surplus available to sell)

Objective: Maximize $z = 2b + 1.5s - f$

subject to constraints

$$\begin{array}{rcrcl} 3b & +2s & \leq & 250 & \\ b & +s & = & 70 & +f \end{array}$$

with $b, s \geq 0$, and f unrestricted (and all variables integral)

Note 1: could have said $b + s \leq 70 + f$ (can't use more flour than have); but optimum doesn't change if we force equality, since no optimum solution will have $b + s < 70 + f$; this leaves unsold surplus flour

Note 2: could restrict f by $f \geq -70$, but no feasible solution will have $f < -70$, so no need

Standard form of baker's problem [September 3]

Variables: b, s, f', f'', x_1

Objective: Minimize $z = -2b - 1.5s + f' - f''$

subject to constraints

$$\begin{array}{rrrrrcl} 3b & +2s & & & +x_1 & = & 250 \\ b & +s & -f' & +f'' & & = & 70 \end{array}$$

as well as all variables non-negative, and all integral.

Terminology [September 5]

Canonical form for a system of m equations, n unknowns, $m \leq n$: any form in which a particular m of the variables (the **basic variables**) each appear in one one equation (a different one for each variable), and each with coefficient 1. The remaining $n - m$ variables are **non-basic**.

Basic solution corresponding to these basic variables: the (unique) solution to the system that is (easily) obtained by setting each of the non-basic variables to 0.

Basic feasible solution: a basic solution with all basic variables non-negative

Example:

$$\begin{array}{rrrrrcl} 3x_1 & +x_2 & & & -.6x_5 & = & 11 \\ x_1 & & x_3 & & -4.7x_5 & = & 0 \\ -5x_1 & & & +x_4 & & = & 3.3 \end{array}$$

is in canonical form with basic variables x_2, x_3, x_4; the associated basic solution is $(0, 11, 0, 3.3, 0)$; it's a basic feasible solution.

Thumbnail sketch of Simplex method [September 5]

The Simplex method tries to solve an LP by:

1. Putting the problem in standard form
2. Finding a basic feasible solution (*requires some work!*)
3. Checking if the basic feasible solution from step 2) is optimal (*will turn out to be easy*)
4. If not, moving to a different basic feasible solution that yields a smaller objective value (*will turn out to be easy, modulo some algebra*)
5. Checking if the basic feasible solution from step 4) is optimal (*easy*)
6. Repeating steps 4) and 5) until optimal solution is reached (*easy*)

Steps 2) and 6) raise three theoretical questions:

1. Is there always a basic feasible solution to start from (*requires a little work*)
2. Does the objective always reach its minimum at a basic feasible solution (*requires a little work*)
3. Are we sure to reach an optimal solution by moving around the basic feasible solutions (*a very delicate question!*)

Terminology [September 8]

Canonical form for an LP, with a particular named set of basic variables:

1. The problem is in standard form
2. The system of constraints is in canonical form, with the named set of variables as basic
3. The associated basic solution is feasible
4. The objective function is expressed solely in terms of non-basic variables

Example: Minimize $2x_1 - 2x_2 + 3x_3$ subject to $x_1, x_2, x_2 \geq 0$ and

$$\begin{array}{rrrcl} 2x_1 & -2x_2 & -2x_3 & = & 17 \\ x_1 & & -x_3 & = & 4 \end{array}$$

is not in canonical form; but the equivalent problem:

Minimize $5x_3 - 10$ subject to $x_1, x_2, x_3 \geq 0$ and

$$\begin{array}{rrrcl} x_1 & & -x_3 & = & 4 \\ & x_2 & & = & 9 \end{array}$$

is in canonical form with basic variables x_1, x_2.

Steps of the simplex method I [September 10]

Given a linear programming (LP) problem (maximize/minimize an objective function that is linear in some variables, subject to some linear constraints on the variables).

Step 1: Put the problem into *standard form* (minimization, all constraints equalities, all variables constrained to be non-negative)

Step 2: Find a collection of variables which can act as the basic variables in a basic feasible solution, and use pivoting operations to put the problem into canonical form (still standard, each basic variable appears once, each in a different constraint, each time with coefficient one, objective re-expressed in terms of non-basic variables only, all constants on right-hand sides of constraints non-negative [last not part of definition of canonical form; relates to associated basic solution being feasible])

Natural questions to ask:

1. How to find these basic variables? Trial-and-error (ugh)? And what if no such variables exist? For now, we'll use magic; but later we'll see a systematic way to find this initial basic feasible solution (if it exists); slightly circularly, this systematic way uses simplex method!
2. What if there are fewer variables than constraints? We'll deal with this later.
3. Why is what we are doing called "simplex" method? We'll (vaguely) answer this later.

Steps of the simplex method II [September 10]

A generic simplex problem in canonical form, with $m = 3$, $n = 6$:

Minimize z subject to constraints

$$\begin{array}{ccccccccccc} x_1 & & & + & a_{1,4}x_4 & + & a_{1,5}x_5 & + & a_{1,6}x_6 & = & b_1 \\ & x_2 & & + & a_{2,4}x_4 & + & a_{2,5}x_5 & + & a_{2,6}x_6 & = & b_2 \\ & & x_3 & + & a_{3,4}x_4 & + & a_{3,5}x_5 & + & a_{3,6}x_6 & = & b_3 \\ & & & & c_4x_4 & + & c_5x_5 & + & c_6x_6 & = & z_0 + z \end{array}$$

as well as $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$.

Here all the a's, b's, c's are constants, and because we're assuming that associated basic solution is feasible, also $b_1, b_2, b_3 \geq 0$

Step 3: Read off associated basic feasible solution, associated objective value. Here $(b_1, b_2, b_3, 0, 0, 0)$, $z = -z_0$.

Step 4: Check is this basic feasible solution optimal (and if it is, stop).

Theorem 3.4.1 (page 78, optimality criterion): If $c_4, c_5, c_6 \geq 0$, then the basic feasible solution $(b_1, b_2, b_3, 0, 0, 0)$ is optimal!

Step 5: If there's at least one c that is strictly negative; say $c_4 < 0$, then we can decrease z by increasing x_4 while keeping $x_5, x_6 = 0$. Decide just how much x_4 can be increased by, while keeping $x_5, x_6 = 0$, without moving out of feasibility.

Steps of the simplex method III [September 12]

Step 5, continued: With x_5, x_6 held at 0, constraint 1 says

$$x_1 + a_{1,4}x_4 = b_1.$$

If $a_{1,4}$ is 0, then we can increase x_4 all the way to $+\infty$ without violating this constraint (just by holding $x_1 = b_1$). If $a_{1,4}$ is negative, we can also increase x_4 all the way to $+\infty$ without violating this constraint (by increasing x_1 commensurately). Example: if constraint is $x_1 - 2x_4 = 5$, we can move x_4 from 0 to 100 by moving x_1 from 5 to 205.

The same consideration holds for the two other constraints, leading to

Theorem 3.4.2 (page 79, unboundedness criterion): If c_4 (or one of the other c's) is strictly < 0, and all of $a_{1,4}$, $a_{2,4}$, $a_{3,4}$ are ≤ 0, then the objective function of the problem can be made arbitrarily small (i.e., arbitrarily negative; think $-\infty$), and so the solution to the minimization problem is not bounded from below.

Steps of the simplex method IV [September 12]

Step 5, continued: Back to

$$x_1 + a_{1,4}x_4 = b_1.$$

If $a_{1,4}$ is positive, then this constraint allows us to increase x_4 until $a_{1,4}x_4$ reaches b_1, without causing x_1 to become negative. I.e., x_4 can be increased up to $b_1/a_{1,4}$. Example: if constraint is $x_1 + 2x_4 = 5$, we can move x_4 from 0 to 2.5 by moving x_1 5 down to 0, but we can't move x_4 up any higher without causing x_1 to become negative.

The same consideration for each of constraint 2, constraint 3, tells us exactly how far we are allowed to move x_4 up, without violating *any* constraint, while keeping $x_5, x_6 = 0$:

Theorem 3.4.3 (page 81, the departing variable): In the situation where $c_4 < 0$, and not all of $a_{1,4}, a_{2,4}, a_{3,4}$ are ≤ 0, check which is the minimum of

$$\frac{b_1}{a_{1,4}}, \frac{b_2}{a_{2,4}}, \frac{b_3}{a_{3,4}}$$

(actually, only consider those ratios where the a term is positive). Suppose it is $b_1/a_{1,4}$. Then x_4 can be increased to $b_1/a_{1,4}$, while keeping $x_5, x_6 = 0$, without dropping any of x_1, x_2, x_3 below zero, while maintaining feasibility. In other words, there is a basic feasible solution to the problem, with now x_2, x_3, x_4 as the basic variables, which has a better (lower) objective value than the basic feasible solution with x_1, x_2, x_3 as basic variables. Pivoting on $a_{1,4}x_4$ puts the problem into canonical form with basic variables x_2, x_3, x_4.

Steps of the simplex method V [September 12]

Step 6: pivot on $a_{1,4}x_4$ to put the problem into canonical form with basic variables x_2, x_3, x_4. Get:

$$\begin{array}{rcrcrcrcrcrcl}
\star x_1 & & & & & + & x_4 & + & \star x_5 & + & \star x_6 & = & b_1/a_{1,4} \\
\star x_1 & + & x_2 & & & + & & + & \star x_5 & + & \star x_6 & = & b_2 - a_{2,4}b_1/a_{1,4} \\
\star x_1 & & & + & x_3 & + & & + & \star x_5 & + & \star x_6 & = & b_3 - a_{3,4}b_1/a_{1,4} \\
\star x_1 & & & & & + & & + & \star x_5 & + & \star x_6 & = & z_0 + z - \frac{c_4 b_1}{a_{1,4}}
\end{array}$$

as well as $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$. Here $\star$'s represent coefficients whose values we don't care about right now. By or choice of which variable to kick out of the basic variables/which row to pivot on, we have made sure that the associated basic solution

$$(0, b_2 - a_{2,4}b_1/a_{1,4}, b_3 - a_{3,4}b_1/a_{1,4}, b_1/a_{1,4}, 0, 0)$$

is feasible, and by our choice of which variable to add to the basic variables/which column to pivot on, we have made sure that the associated objective value

$$z = -z_0 + \frac{c_4 b_1}{a_{1,4}}$$

is not any bigger than it had been previously, and *if* $b_1 > 0$ (*non-degeneracy*) then in fact the objective has become strictly smaller.

Steps of the simplex method VI [September 12]

Step 7: For the new basic feasible point, check

- is the solution optimal? (Thm 3.4.1) (if so, stop)
- is the objective unbounded from below? (Thm 3.4.2) (if so stop)

Otherwise, apply Thm 3.4.3 again.

Step 8: Repeat as necessary.

Natural questions:

1. What happens if we get to a degenerate basic feasible solution (some of the b's are zero, so some basic variables are 0)? Let's pretend (for the moment) that this never happens
2. Does the process terminate, eventually, at an optimal solution (or a solution of $-\infty$)? **YES!!!!** (assuming non-degeneracy)

A poor professor's diet problem [September 22]

I have three vitamin requirements: vitamins A, B and C. There are five breakfast cereals available to me: cereals 1, 2, 3, 4 and 5. I want to produce a blend of the cereals that satisfies my vitamin requirements at minimum cost. Here's a table of data showing the number of units of each vitamin in each cereal, my daily requirement of each vitamin, and the cost of each unit of cereal:

	Cer. 1	Cer. 2	Cer. 3	Cer. 4	Cer. 5	Vit. req.
Vit. A	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	req_1
Vit. B	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	req_2
Vit. C	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	req_3
Cost	cost_1	cost_2	cost_3	cost_4	cost_5	

Poor professor's problem (thanks to L. Trotter, Cornell): x_1, x_2, x_3, x_4, x_5 represent numbers of units of each cereal used.

Minimize $\text{cost}_1 x_1 + \text{cost}_2 x_2 + \text{cost}_3 x_3 + \text{cost}_4 x_4 + \text{cost}_5 x_5$ subject to

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & a_{14}x_4 & + & a_{15}x_5 & \geq & \text{req}_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & a_{24}x_4 & + & a_{25}x_5 & \geq & \text{req}_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & + & a_{34}x_4 & + & a_{35}x_5 & \geq & \text{req}_3 \end{array}$$

and $x_1, x_2, x_3, x_4, x_5 \geq 0$.

A rich corporation's sales problem [September 22]

General Nutrition wants me to take vitamin supplements instead of eating cereal, and wants to know what prices to set to encourage me to make the switch. If GN sets the price of Vitamin A at y_1 per unit, B at y_2 and C at y_3, then they want to make sure that the y's are set so that I can't extract the vitamins more cheaply from any of my favourite cereals. For example, I can already get a_{11} units of vitamin A, a_{21} units of vitamin B, and a_{31} units of vitamin C, at a cost of cost_1, by buying a unit of Cereal 1. The same cocktail of vitamins would cost me $a_{11}y_1 + a_{21}y_2 + a_{31}$ at GN. So GN better make sure

$$a_{11}y_1 + a_{21}y_2 + a_{31} \leq \text{cost}_1.$$

GN (a corporation) wants to maximize its revenue from me, and knows how much of each vitamin I need. They get the following LP problem:

Rich corporation's problem: y_1, y_2, y_3 represent costs of units of each vitamin.

Maximize $\text{req}_1 y_1 + \text{req}_2 y_2 + \text{req}_3 y_3$ subject to

$$\begin{array}{ccccccc}
a_{11}y_1 & + & a_{21}y_2 & + & a_{31}y_3 & \leq & \text{cost}_1 \\
a_{12}y_1 & + & a_{22}y_2 & + & a_{32}y_3 & \leq & \text{cost}_2 \\
a_{13}y_1 & + & a_{23}y_2 & + & a_{33}y_3 & \leq & \text{cost}_3 \\
a_{14}y_1 & + & a_{24}y_2 & + & a_{34}y_3 & \leq & \text{cost}_4 \\
a_{15}y_1 & + & a_{25}y_2 & + & a_{35}y_3 & \leq & \text{cost}_5
\end{array}$$

and $y_1, y_2, y_3 \geq 0$.

Terminology [September 22]

Max form: An LP is in *max form* if it is a maximization, all constraints are $\leq$, and all variables are non-negative.

Maximize $z = c_1x_1 + c_2x_2 + c_3x_3$ subject to

$$\begin{array}{rcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & \leq & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & \leq & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & \leq & b_4 \\ a_{51}x_1 & + & a_{52}x_2 & + & a_{53}x_3 & \leq & b_5 \end{array}$$

and $x_1, x_2, x_3 \geq 0$.

Example: The rich corporation's problem was in Max form

Theorem: Every LP can be expressed in Max form.

Proof: Start from standard form. Replace minimize with maximize by multiplying objective by -1. Replace each equality with two inequalities, e.g.,

$$3x + 5y - 6z = 3$$

becomes

$$\begin{array}{rcl} 3x + 5y - 6z & \leq & 3 \\ -3x - 5y + 6z & \leq & -3 \end{array}$$

Terminology [September 22]

Min form: An LP is in *min form* if it is a minimization, all constraints are $\geq$, and all variables are non-negative.

Minimize $z = c_1x_1 + c_2x_2 + c_3x_3$ subject to

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & \geq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & \geq & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & \geq & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & \geq & b_4 \\ a_{51}x_1 & + & a_{52}x_2 & + & a_{53}x_3 & \geq & b_5 \end{array}$$

and $x_1, x_2, x_3 \geq 0$.

Example: The poor professor's problem was in Min form

Theorem: Every LP can be expressed in Min form.

The Dual problem [September 22]

The Dual: Given a problem **P** *in max form*, the associated dual problem **D** has one variable for each constraint of **P**, and one constraint for each variable of **P**. It is a min form problem. If **P** reads off constraints from rows of a data array, **D** reads off constraints from the columns. If **P** uses the final row of the data array to construct the objective, **D** uses the final column.

P (*primal* problem): Maximize $z = c_1x_1 + c_2x_2 + c_3x_3$ subject to

$$\begin{array}{lcllcll} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & \leq & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & \leq & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & \leq & b_4 \\ a_{51}x_1 & + & a_{52}x_2 & + & a_{53}x_3 & \leq & b_5 \end{array}$$

and $x_1, x_2, x_3 \geq 0$.

D (*dual* problem): Minimize $v = b_1y_1 + b_2y_2 + b_3y_3 + b_4y_4 + b_5y_5$ subject to

$$\begin{array}{lcllcllcllcl} a_{11}y_1 & + & a_{21}y_2 & + & a_{31}y_3 & + & a_{41}y_4 & + & a_{51}y_5 & \geq & c_1 \\ a_{12}y_1 & + & a_{22}y_2 & + & a_{32}y_3 & + & a_{42}y_4 & + & a_{52}y_5 & \geq & c_2 \\ a_{13}y_1 & + & a_{23}y_2 & + & a_{33}y_3 & + & a_{43}y_4 & + & a_{53}y_5 & \geq & c_3 \end{array}$$

and $y_1, y_2, y_3, y_4, y_5 \geq 0$.

Using matrix notation [September 22]

Matrix notation is our friend! (See page 125 of the textbook) If

- A is the m-row, n-column matrix (m constraints, n variables), ij entry a_{ij}
- b is the column vector $(b_1, b_2, \ldots, b_m)^t$ (column of rhs's of constraints)
- c is the column vector $(c_1, c_2, \ldots, c_n)^t$ (column of objective coefficients)
- X is the column vector $(x_1, x_2, \ldots, x_n)^t$ of primal variables
- Y is the column vector $(y_1, y_2, \ldots, y_m)^t$ of dual variables

then:

$$\begin{array}{rlllll} \text{Primal } \mathbf{P}: & \text{Max } z = c \cdot X & \text{subject to} & AX \leq b & X \geq 0 \\ \text{Dual } \mathbf{D}: & \text{Min } v = b \cdot Y & \text{subject to} & A^t Y \geq c & Y \geq 0 \end{array}$$

The dual can be defined for *any* LP; just put it into Max form first! Using matrix notation, it's easy to see:

Theorem: The Dual of the Dual is the Primal. In other words:

- Start with any LP problem **P** in Max form
- Form its dual problem **D**
- Put **D** into Max form
- Now that **D** is in Max form, you can write down its dual $\mathbf{D}^2$
- You are back where you started! $\mathbf{D}^2$ is exactly the same as **P**

Artificial variables [September 24]

Artificial variables: An efficient way to find an initial basic feasible solution. If a problem in standard form doesn't have an obvious basic feasible solution:

- Add one new (artificial) non-negative variable to each constraint
- Use simplex to minimize the sum of the artificial variables, subject to the new constraints (this requires expressing the sum of the artificial variables in terms of the real variables; the artificial variables will be the basic variables when the simplex method starts to solve the artificial problem)
- If the optimum objective is 0, ignoring artificial variables (which are all 0 at optimum) gives a basic feasible solution to original problem. Use this to start simplex on the original problem (initially expressing the original objective in terms of non-basic variables, if necessary)
- If the optimum objective is > 0, the original problem had no feasible point (any such point would give feasible point for artificial problem, with objective value 0, just by setting all artificial variables to 0)
- One of the above two must happen: the artificial problem can't have negative objective value, so is not unbounded from below

Two notes on artificial variables [September 24]

- Any variable that appears in the original problem only once, with coefficient 1, can be used as a basic variable in the artificial problem; so it may not be necessary to add as many artificial variables as constraints. The artificial objective stays as the sum of the *artificial* variables in this case
- LP Assistant takes care of the initial correct expression of the artificial objective in terms of non-basic variables, and of keeping the original objective expressed in terms of non-basic variables

A possible problem: redundancy [September 24]

Problem: System of equations is *redundant* if some equations can be expressed as linear combinations of others. Redundant systems *can't* have basic feasible solutions! (In canonical form, there are clearly no dependencies among equations)

Solution: Add artificial variables. As before, artificial optimum > 0 reveals no feasible points. If artificial optimum is 0:

- if all artificial variables non-basic, everything is ok (have basic feasible solution for original problem)
- if some are basic, they are currently set to 0. For each one: scan its row. Pick an original, non-basic variable in row with non-zero coefficient, pivot on that coefficient to replace artificial basic variable (value 0) with original variable (also 0).
- if this deals with all basic variables, everything is ok (have basic feasible solution for original problem)
- any row this does not deal with, is a row that only mentions artificial variables, and for purposes of solving original problem, can be ignored! Again everything is ok: on deleting rows, get basic feasible solution for a scaled-down problem that has no redundancy. Solving the artificial problem roots out redundancy.

Simplex I: preparation [September 26]

Given any linear programming problem:

- Put it in standard form (all constraints equalities of the form $a_1x_1 + \ldots + a_nx_n = b$ with $b \geq 0$, all variables non-negative, objective is to be minimized)
- Are there at least as many variables as constraints?
  - YES: proceed
  - NO: add in a few new variables, all with zero coefficients everywhere
- Is there an obvious basic feasible solution (a collection of variables, one for each constraint, that only appears once in the constraints, with coefficient 1)?
  - YES: put the problem in canonical form (i.e., re-express the objective in terms of non-basic variables) and record the data of the problem in a tableau: first column records the names of the current basic variables, next columns are one for each variable, final column records right-hand side values of constraints, rows record constraints, final row records objective.
  - NO: add an artificial variable to each constraint that needs one; consider artificial problem of minimizing sum of artificial variables subject to new constraints. This has an obvious basic feasible solution; put artificial problem in canonical form and set up initial tableau.

Simplex II: tableau operation [September 26]

Given a tableau, repeat this sequence of steps until it stops:

- Optimality: Are all coefficients in the objective row at least 0? Optimality has been reached, STOP
- Unboundedness: Is there a negative coefficient in the objective row, with all coefficients above it 0 or negative? Objective is not bounded from below, STOP
- Otherwise:
  - Pick a negative coefficient in the objective row, with at least one positive coefficient above it (using an *entering rule* — e.g., first such coefficient one finds, going from left to right along objective row). Variable in this column will enter the set of basic variables
  - Find smallest ratio of right-hand side entry to *positive* constraint row entry in column on entering variable (breaking ties using a *departing rule* — e.g., choose topmost such ratio in case of tie). Basic variable in this row will leave set of basic variables.
  - Pivot on chosen entry to bring in the entering variable to, and remove the departing variable from, the basis

Simplex III: interpretation [September 26]

If you are working on the artificial problem:

- Termination with unboundedness will not happen
- If termination with optimality happens, and optimum is > 0, RECORD: original problem has no feasible point
- If termination with optimality happens, and optimum is 0, read off basic feasible solution to original problem, delete artificial variable columns and artificial objective row, original problem is now set up to perform tableau operations. **NOTE**: this process may require some pivoting, to remove zero-value artificial basic variables from the set of basic variables, and row-deletion, if redundancies are discovered

If you are working on the original problem:

- If termination with unboundedness happens, RECORD: original problem is not bounded from below
- If termination with optimality happens, RECORD: the current basic feasible solution, and the corresponding objective value (negative of the number in the bottom right corner of the tableau), gives the optimum objective. If necessary (e.g., if original original problem was a maximization), translate this back to the set-up of the problem before standardization

A fundamental theorem [September 29]

Theorem: Start with any linear programming problem. Assuming that at no point in either the artificial or original problem is there degeneracy (a basic feasible solution at which a basic variable has value 0) then the simplex algorithm, as described, produces, in a finite amount of time, either an optimal solution to the problem, or the information that the optimum is not bounded from below.

Tragic fact: If there is degeneracy, then with an unlucky choice of entering rule and departing rule, the simplex method may run forever without producing an optimum solution.

Happy fact:

- In practice: infinite loops in the simplex method are incredibly rare
- In theory: given any linear programming problem presented in canonical form, there is a finite sequence of pivoting steps that leads the simplex algorithm to terminate (either with a declaration of optimality or a declaration of unboundedness).

Bland's rule to avoid cycling [September 29]

Bland's rule for pivoting: If there are multiple negative entries in the objective row, choose the first one, reading left to right. If there are multiple smallest ratios corresponding to this choice, choose the one that throws out from the basis the current basic variable with the smallest index.

Theorem (proved in an elementary but quite convoluted way by Bland in 1977): Start with any linear programming problem, and run the simplex algorithm using Bland's rule for pivoting. Whether degeneracy is encountered or not, the algorithm will terminate in a finite amount of time, either with an optimal solution to the problem, or the information that the optimum is not bounded from below.

Drawbacks to Bland's rule:

- Often takes longer than other pivoting rules (such as "choose the *most negative* entry in the objective row") would.
- Might force you to divide by very small numbers, leading to rounding errors.

How many iterations does the simplex method need? [September 29]

Typically: If a problem has n variables and n constraints, then almost always simplex will solve it after no more than $\approx 3n$ pivots

Worst case: For almost every particular pivot rule that has been proposed, there are examples of problems with n variables and n constraints that it takes simplex about 2^n pivots to solve

Open question: Is there a pivot rule that allows simplex to solve *every* problem with n variables and n constraints using, say, no more than n^{10} pivots?

Fact: There are methods, completely different from simplex, that solve *every* Linear Programming problem quickly

Recalling Max form and Min form [September 29]

Max form: An LP is in *max form* if it is a maximization, all constraints are $\leq$, and all variables are non-negative.

Maximize $z = c_1x_1 + c_2x_2 + c_3x_3$ subject to

$$\begin{array}{rcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & \leq & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & \leq & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & \leq & b_4 \\ a_{51}x_1 & + & a_{52}x_2 & + & a_{53}x_3 & \leq & b_5 \end{array}$$

and $x_1, x_2, x_3 \geq 0$.

Min form: An LP is in *min form* if it is a minimization, all constraints are $\geq$, and all variables are non-negative.

Minimize $z = c_1x_1 + c_2x_2 + c_3x_3$ subject to

$$\begin{array}{rcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & \geq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & \geq & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & \geq & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & \geq & b_4 \\ a_{51}x_1 & + & a_{52}x_2 & + & a_{53}x_3 & \geq & b_5 \end{array}$$

and $x_1, x_2, x_3 \geq 0$.

Theorem: Every LP can be expressed in both Max form and in Min form.

Recalling the Dual problem [September 29]

The Dual: Given a problem **P** *in max form*, the associated dual problem **D** has one variable for each constraint of **P**, and one constraint for each variable of **P**. It is a min form problem. If **P** reads off constraints from rows of a data array, **D** reads off constraints from the columns. If **P** uses the final row of the data array to construct the objective, **D** uses the final column.

P (*primal* problem): Maximize $z = c_1x_1 + c_2x_2 + c_3x_3$ subject to

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & \leq & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & \leq & b_3 \\ a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & \leq & b_4 \\ a_{51}x_1 & + & a_{52}x_2 & + & a_{53}x_3 & \leq & b_5 \end{array}$$

and $x_1, x_2, x_3 \geq 0$.

D (*dual* problem): Minimize $v = b_1y_1 + b_2y_2 + b_3y_3 + b_4y_4 + b_5y_5$ subject to

$$\begin{array}{ccccccccccc} a_{11}y_1 & + & a_{21}y_2 & + & a_{31}y_3 & + & a_{41}y_4 & + & a_{51}y_5 & \geq & c_1 \\ a_{12}y_1 & + & a_{22}y_2 & + & a_{32}y_3 & + & a_{42}y_4 & + & a_{52}y_5 & \geq & c_2 \\ a_{13}y_1 & + & a_{23}y_2 & + & a_{33}y_3 & + & a_{43}y_4 & + & a_{53}y_5 & \geq & c_3 \end{array}$$

and $y_1, y_2, y_3, y_4, y_5 \geq 0$.

Using matrix notation [September 29]

Matrix notation is our friend! (See page 125 of the textbook) If

- A is the m-row, n-column matrix (m constraints, n variables), ij entry a_{ij}
- b is the column vector $(b_1, b_2, \ldots, b_m)^t$ (column of rhs's of constraints)
- c is the column vector $(c_1, c_2, \ldots, c_n)^t$ (column of objective coefficients)
- X is the column vector $(x_1, x_2, \ldots, x_n)^t$ of primal variables
- Y is the column vector $(y_1, y_2, \ldots, y_m)^t$ of dual variables

then:

$$\begin{array}{rlllll} \text{Primal } \mathbf{P}: & \text{Max } z = c \cdot X & \text{subject to} & AX \leq b & X \geq 0 \\ \text{Dual } \mathbf{D}: & \text{Min } v = b \cdot Y & \text{subject to} & A^t Y \geq c & Y \geq 0 \end{array}$$

The dual can be defined for *any* LP; just put it into Max form first! Using matrix notation, it's easy to see:

Theorem: The Dual of the Dual is the Primal. In other words:

- Start with any LP problem **P** in Max form
- Form its dual problem **D**
- Put **D** into Max form
- Now that **D** is in Max form, you can write down its dual $\mathbf{D}^2$
- You are back where you started! $\mathbf{D}^2$ is exactly the same as **P**

Shortcuts to duality [October 1]

Given primal problem **P**:

- if it is a maximization, change all $\geq$ constraints to $\leq$ by multiplying by -1
- if it is a minimization, change all $\leq$ constraints to $\geq$ by multiplying by -1
- Don't bother changing $=$ constraints to pair of $\leq$ or $\geq$
- Don't bother replacing unrestricted variables with pair of restricted variables

Construct dual by these operations:

Maximaization problem	goes to	minimization problem
$\leq$ constraint $\#(j)$	goes to	non-negative variable $\#(j)$
$=$ constraint $\#(j)$	goes to	unrestricted variable $\#(j)$
non-negative variable $\#(i)$	goes to	$\geq$ constraint $\#(i)$
unrestricted variable $\#(i)$	goes to	$=$ constraint $\#(i)$
constraint $\#(j)$ coefficients	go to	variable $\#(j)$ coefficients
variable $\#(i)$ coefficients	go to	constraint $\#(i)$ coefficients
objective coefficients	go to	constraint rhs's
constraint rhs's	go to	objective coefficients

Table can be read in both directions

Interpreting the diet problem dual [October 1]

The Primal problem (Poor Professor)

- Objective: minimize cost
- Variables: one for each resource, amount of each resource used in blend
- Constraints: one for each nutrition requirement; blend must have at least a certain amount
- Objective coefficients: cost per unit of each resource
- Right-hand side of constraints: amount of each nutrition required

The Dual problem (Rich Corporation)

- Objective: maximize revenue
- Variables: one for each nutrient, cost per unit of each nutrient
- Constraints: one for each resources; cost of reproducing nutritional content of each resource should be no more than a certain amount
- Objective coefficients: amount of each nutrition required
- Right-hand side of constraints: cost per unit of each resource

Dice problem [October 6]

Ordinary die: comes up 1, 2, 3, 4, 5, 6, each with probability $1/6$. Average value:

$$1(1/6) + 2(1/6) + 3(1/6) + 4(1/6) + 5(1/6) + 6(1/6) = 3\ 1/2$$

Weighted/loaded die: comes up 1, 2, 3, 4, 5, 6, with probabilities $p_1, p_2, p_3, p_4, p_5, p_6$ respectively, each $p_i \geq 0$, p_i's add to 1. Average value:

$$1p_1 + 2p_2 + 3p_3 + 4p_4 + 5p_5 + 6p_6$$

Question: Which loaded die with the same average value as an ordinary die has greatest bias towards 6?

Maximize p_6 subject to

$$\begin{aligned} p_1 + p_2 + p_3 + p + 4 + p_5 + p_6 &= 1 \\ p_1 + 2p_2 + 3p_3 + 4p + 4 + 5p_5 + 6p_6 &= 3\ 1/2 \end{aligned}$$

and all $p_i \geq 0$

Example for duality theorem proof [October 8]

Primal: Maximize $30x_1 + 6x_2 - 5x_3 + 18x_4$ subject to

$$\begin{aligned} 1x_1 + 0x_2 + 2x_3 + 1x_4 &\leq 20 \\ -2x_1 + 1x_2 + 0x_3 - 1x_4 &\leq 15 \\ 6x_1 + 2x_2 - 3x_3 + 0x_4 &\leq 54 \end{aligned}$$

$x_1, x_2, x_3, x_4 \geq 0$.

Dual: Minimize $20y_1 + 15y_2 + 54y_3$ subject to

$$\begin{aligned} 1y_1 - 2y_2 + 6y_3 &\geq 30 \\ 0y_1 + 1y_2 + 2y_3 &\geq 6 \\ 2y_1 + 0y_2 - 3y_3 &\geq -5 \\ 1y_1 - 1y_2 + 0y_3 &\geq 18 \end{aligned}$$

$y_1, y_2, y_3 \geq 0$.

Observation: At optimum simplex tableau for primal, objective row coefficients of slack variables gives optimum solution to dual problem!

A more precise duality theorem obsv. [October 10]

Given a maximization problem, in Max form, with finite optimum:

- Multiply all constraints with negative right-hand sides by -1
- Add slack variables to put into standard form
- Run simplex (possibly with artificial variables initially)
- When simplex has reached optimum tableau, **the vector of coefficients of slack variables in the objective row of the optimum tableau gives an optimum feasible point for the dual problem** (and the optimum value for the dual is the same as that for the primal)

Given a minimization problem, in Min form, with finite optimum:

- **Exactly the same thing happens**

Dealing with equality constraints [October 10]

Given an LP problem with finite optimum:

- Organize constraints so that right-hand sides are all positive
- Add slack variables to $\leq$ and $\geq$ constraints, if necessary multiply objective by -1 to put into standard form
- Run simplex (possibly with artificial variables initially, for $\geq$ constraints and for $=$ constraints)
- When simplex has reached optimum tableau, you can read off the solution to the dual problem:
  - For each $\leq$ or $\geq$ primal constraint, the coefficient of the associated slack variable in the final (real) objective row is the value of the associated dual variable at optimality
  - For each $=$ primal constraint, the value of the associated dual variable at optimality comes from the coefficient of the associated *artificial* variable in the final (real) objective row: if the primal was originally a maximization problem, take the coefficient value, and if the primal was originally a minimization problem, take the negative of the coefficient value.
- The objective value of the dual at optimum is the same as that of the primal

Integer programming examples I [October 13]

Discrete allocation I: I have \$83,500 to invest, and I have the following opportunities:

- A city bond for \$20,000, with 6% return after one year
- A city bond for \$30,000, with 5.5% return after one year
- Treasury bills in units of \$5,000, with 3% return after one year
- Up to 50 shares at \$625 per share, with 5.2% return after one year

I am not allowed to purchase both city bonds. How do I invest to maximize return after 1 year?

Discrete allocation II: I have 4 items, numbered 1 through 4. Item i has value $\$v_i$ and weighs w_i lbs. I can carry at most W lbs. Which items do I choose to carry, to maximize the total value of the items I am carrying?

Integer programming examples II [October 17]

Fixed charges: A factory can produce each of items A and B. A unit of each item has an associated profit in dollars, and uses a number of labor hours and a number of pounds of raw material. Additionally, for each item there is a fixed setup charge in dollars if the decision is made to produce even one of that item. The data for the problem is summarized in the table below:

Data	A	B	Availability
Labor	3	4	100
Materials	2	1	80
Fixed charge	800	600	
Profit	20	30	

How many of each item should be produced to maximize profit?

Sliding charges: Consider two modifications to the above problem:

1. Suppose 10 hours extra labour is available at \$10/hour, and 20 hours at \$12/hour.
2. Suppose that union rules mean that all 20 hours of the more expensive labor must be used before any of the cheaper hours can be used.

Sudoku through integer programming [October 17]

Variables: x_{ijk}, $1 \leq i \leq 9$, $1 \leq j \leq 9$, $1 \leq k \leq 9$ (729 in all)

Interpretation: $x_{ijk} = 1$ if entry in cell (i, j) is k, is 0 otherwise

First family of constraints: For each i, j, k

$$x_{ijk} \leq 1, \text{ all integers}$$

729 constraints

Second family of constraints: For each i, j

$$x_{ij1} + x_{ij2} + \ldots x_{ij9} = 1$$

(one number per cell) — 81 constraints

Third family of constraints: For each i, k,

$$x_{i1k} + x_{i2k} + \ldots x_{i9k} = 1$$

(number k appears exactly once in row i) — 81 constraints

Fourth and fifth family of constraints: encoding that number k appears exactly once in each column, each 3-by-3 box — 162 constraints

Sixth family of constraints: For each cell (i, j) with given number k, $x_{ijk} = 1$

Integer programming problem: Minimize 0 subject to above constraints, all variables non-negative

Parameters: 729 variables, (972+number of givens) constraints

Integer programming examples III [October 27]

Sliding charges again: Material is to be purchased, at a cost of

- \$10 per unit for first 100 units,
- \$8 per unit for next 200 units,
- \$6 per unit thereafter.

Production constraints force that at most 1000 units will ever be needed. Encode the cost of materials purchase using linear equations.

Rental charges in transportation problems: Three warehouse, supplies s_1, s_2, s_3. Three outlets, demands d_1, d_2, d_3; available supply significantly exceed total demand. Shipping cost per unit from warehouse i to outlet j is c_{ij}. Rental cost for warehouse i is r_i, payable (for future use) only if warehouse i still has some units left after shipping. Find minimum cost shipping scheme.

Either/or constraints: Encode using linear constraints that

- a variable x is either 0 or at least 50
- at least one of the constraints $x + y \leq 10$, $x + 2y \leq 16$, $2x + y \leq 16$ holds
- at least two of $x + y \leq 10$, $x + 2y \leq 16$, $2x + y \leq 16$ hold

Gomory's cutting plane algorithm [October 29]

Given: A pure integer programming problem, that is, a linear programming problem where all variables are constrained to be integers.

Gomory's algorithm (1958):

1. Solve the problem using simplex, ignoring the integrality constraints.
2. If the solution is fully integral, problem solved.
3. If not, add one new constraint to the problem that
   1. cuts out the current optimal solution from the feasible set, but
   2. doesn't cut out any feasible points that are all integral.
4. Return to step one.

Load balancing problem [October 29]

Given: Weights $w_1, w_2, \ldots, w_n$, total weight W.

Problem: Split them into two groups of as near equal weight as possible

Equivalent problem: Pick out a subset of the weights whose total weight is as close to $W/2$ as possible, without exceeding it

Equivalent problem, when all the weights are integers: Pick out a subset of the weights whose total weight is as close to $[W/2]$ as possible, without exceeding it, where $[W/2]$ is the biggest integer not exceeding $W/2$

Example: I have a penny, a nickel, a dime and a quarter, and I want to split them into two groups of as near equal value as possible.

LP formulation: x_1 is 0 if I don't choose the penny, 1 if I do, similarly for x_2, x_3, x_4; so the x_i's are integral (actually, binary)

Maximize $x_1 + 5x_2 + 10x_3 + 25x_4$ subject to

$$\begin{aligned} x_1 + 5x_2 + 10x_3 + 25x_4 + x_5 &= 20 \\ x_1 + x_6 &= 1 \\ x_2 + x_7 &= 1 \\ x_3 + x_8 &= 1 \\ x_4 + x_9 &= 1 \end{aligned}$$

all $x_i \geq 0$, all x_i integers

Gomory's cutting plane algorithm II [October 31]

Given: A pure integer programming problem

1. Solve problem using simplex, ignoring integrality constraints
2. If solution is fully integral, problem solved
3. If not
   1. pick out row $\sum_i a_i x_i = b$ in final tableau with b not integral
   2. rewrite as $-d + \sum_i d_i x_i = e - \sum_i e_i x_i$ with
      - $0 \leq d_i < 1$ for each i, and $0 < d < 1$
      - $e, e_1, e_2, \ldots, e_n$ all integers
   3. add variable $x_{n+1} \geq 0$, integral, constraint $-d + \sum_i d_i x_i = x_{n+1}$
4. key observations:
   1. if $(x_1, \ldots, x_n)$ feasible integral point for original problem, there's x_{n+1} with $(x_1, \ldots, x_n, x_{n+1})$ feasible integral point for original problem
   2. if $(x_1^\star, \ldots, x_n^\star)$ the optimum picked out by simplex for original problem with integrality ignored, there is no $x_{n+1}^\star$ with $(x_1^\star, \ldots, x_n^\star, x_{n+1}^\star)$ feasible for new problem with integrality ignored
5. Return to step one
6. Rules for choosing row exist, that ensure termination of algorithm with optimum integral point; in practice, just choose non-integral b with largest fractional part

Branch-and-bound algorithm [November 5]

Given: An LP problem P with some integer constraints

1. Form P' by ignoring integrality constraints, solve using simplex
2. If solution is integral where it should be (*good* solution), STOP
3. If not
   1. pick variable that should be integer but isn't, say $x_i = b_i$
   2. Form subproblem P_1: old problem (P) together with extra constraint $x_i \leq [b_i]$, where $[b_i]$ is largest integer not exceeding b_i, and x_i still integral
   3. Form subproblem P_2: old problem (P) together with extra constraint $x_i \geq [b_i] + 1$, and x_i still integral
4. Optimum for P must be feasible for one of P_1, P_2, so return to step 1 with both subproblems (this is *branching*)
5. STOP when all branches have terminated; branch with best good solution has the optimum for P

Reasons for a branch to terminate [November 5]

1. subproblem has good solution
2. subproblem has no feasible solution or is unbounded
3. subproblem has an optimum, that is not as advantageous as optimum on some other branch that has terminated with a good solution (**NB**: for maximization problems, objectives get smaller along a branch; for minimization problems, objectives get bigger along a branch) (this is *bounding*)
4. subproblem has an optimum, that is more advantageous than the optimum on some other branch that has terminated with a good solution, but more advantageous only by a fraction less than 1 (this only if we know that the optimum objective value will be an integer)

Transportation algorithm I [November 10]

1. **Step 1**: find a basic feasible solution using Northwest rule:
   - fill in cell $(1, 1)$ as much as possible
   - if warehouse 1 now empty, move down one cell; if outlet 1 now satisfied, move across 1 cell
   - repeat
2. **Step 2**: check for optimality:
   - assign variables $u_1, \ldots, u_m$ to the warehouses, $v_1, \ldots, v_n$ to the outlets
   - set $u_1 = 0$, and fix remaining so that for each basic cell (i, j), $u_i + v_j = c_{ij}$
   - calculate $u_i + v_j$ for all remaining cells, and compare with $c_i j$; if $u_i + v_j$ always at most c_{ij}, current solution is optimal, if not, it is not
3. **Step 3**: if not optimal, decide on a new (entering) basic variable:
   - typically one picks that cell (i, j) for which $u_i + v_j - c_{ij}$ is biggest

Transportation algorithm II [November 10]

1. **Step 4**: enter this new basic variable:
   - find a loop: a chain of cells

     $$\text{cell } 1 \to \text{cell } 2 \to \text{cell } 3 \ldots \to \text{cell } \ell \to \text{cell } 1$$

     with these properties:
     - ⋆ cell 1 is the entering variable, all other cells are basic variables
     - ⋆ consecutive cells are either in same row or same column
     - ⋆ chain takes a 90 degree turn at each cell
   - enter value x in the entering variable cell, and adjust the values in all other cells by adding/subtracting x so that supply and demand constraints are still met
   - choose x as large as possible so that all entries are still non-negative
   - pick one of the cells that ends up with a zero value to be the departing variable, and erase this value
2. **Step 5**: return to Step 2 and repeat

Transportation algorithm III [November 14]

Some modifications to setup

1. What if supply exceeds demand? Set up a dummy outlet, whose demand is the surplus, and with all shipping costs 0
2. What if demand exceeds supply? Problem is unfeasible. But, one can still ask "what is the best shipping scheme that ships out all available units, and doesn't exceed any single outlet demand?". Set up a dummy warehouse, whose supply is the deficit, and with all shipping costs 0
3. What if some shipping cost are *negative* (e.g., due to subsidies received)? Proceed as before; correctness of algorithm does not rely on positivity of costs
4. What if there is some warehouse - outlet pair along which nothing can be shipped? Invent a large dummy cost for this route, so large that the cost of sending even one unit exceeds the total cost of any feasible scheme (if one exists). For example, set the dummy cost to be one greater than the total supply times the largest real cost. If the optimum solution uses any of the forbidden routes, that says that there was no feasible solution to the original problem.

Minimum spanning tree [November 14]

n cities, $C_1, C_2, \ldots, C_n$, want to connect themselves up with a road network. It's required that in the end, it should be possible to travel between any two cities, perhaps via some other cities. For each pair of cities (i, j) the cost c_{ij} of connected them up with a road is known. What configuration of roads achieves this at minimum cost?

$n(n-1)/2$ potential roads, so $2^{n(n-1)/2}$ possible configurations (not all feasible)

Optimal configuration will have no *cycles*: cities $C_{i_1}, C_{i_2}, \ldots, C_{i_k}$ with C_{i_1} joined directly to C_{i_2}, C_{i_2} joined directly to C_{i_3}, etc, and C_{i_k} joined directly to C_{i_1}. A connected configuration with no cycles is called a *tree*. But, there are a lot of trees on n cities: n^{n-2} of them. **FACT**: all trees have $n-1$ connections.

Minimum spanning tree problem can be encoded as an integer (binary) programming problem, with one variable for each potential connection, one constraint for each variable (to say that it is binary), one constraint for each potential cycle (to say that not all connections are present), and one constraint to say that there are $n-1$ connections.

- $n(n-1)/2$ variables, $n(n-1)/2 + \left(\sum_{k=3}^{n} \frac{n(n-1)\ldots(n-(k-1))}{2k}\right) + 1$ constraints
- $n = 4$: 6 variables, 14 constraints
- $n = 40$: 780 variables, 2.8×10^{46} constraints

Simplex plus Gomory or Branch & bound probably won't help!

IP formulation of 4 city MST problem [November 17]

Variables: X_{AB}, X_{AC}, X_{AD}, X_{BC}, X_{BD}, X_{CD},

$$X_{AB} = \begin{cases} 1 & \text{if connection from } A \text{ to } B \text{ used} \\ 0 & \text{if not, etc.} \end{cases}$$

Objective: Minimize $5X_{AB} + 6X_{AC} + 5X_{AD} + 3X_{BC} + 4X_{BD} + 3X_{CD}$

Constraints: all X's non-negative and integral, as well as

$$\begin{aligned} X_{AB} + X_{AC} + X_{BC} &\leq 2 \\ X_{AB} + X_{AD} + X_{BD} &\leq 2 \\ X_{AC} + X_{AD} + X_{CD} &\leq 2 \\ X_{BC} + X_{BD} + X_{CD} &\leq 2 \\ X_{AB} + X_{AD} + X_{BC} + X_{CD} &\leq 3 \\ X_{AB} + X_{AC} + X_{BD} + X_{CD} &\leq 3 \\ X_{AC} + X_{AD} + X_{BC} + X_{BD} &\leq 3 \\ X_{AB} + X_{AC} + X_{AD} + X_{BC} + X_{BD} + X_{CD} &= 4 \end{aligned}$$

and

$$X_{AB},\ X_{AC},\ X_{AD},\ X_{BC},\ X_{BD},\ X_{CD} \text{ each } \leq 1.$$

Kruskal's MST algorithm [November 17]

Joseph Kruskal discovered the following in 1956:

1. Start with no connections between the n cities
2. Select the cheapest available edge (break ties arbitrarily)
3. Delete from future consideration all connections whose addition would create a cycle
4. Go back to step 2
5. When no connections are left available, algorithm stops; selected connections form a minimum weight spanning tree

This is an example of a *greedy* algorithm: it selects the optimum continuation at each stage, without considering the possibility that a slightly less advantageous choice now might lead to significant gains later.

Sometimes greed is good (as in Kruskal's algorithm); sometimes not.

Proof of Kruskal I — preliminaries [November 17]

1. No cycles ever created, by design
2. As long as network not yet fully connected, there are available connections that don't create cycle, and algorithm continues; so when it stops, network connected
3. Final network is optimal — takes some work

Lemma: F, F' two cycle-free networks on n cities, with F having fewer connections than F'. There's connection in F' that can be added to F, that joins two clusters of F (so keeps it cycle-free)

Proof:

- If F has k clusters, then since each cluster has one fewer connections than cities, F has $n-k$ connections; if F' has k' clusters, then it has $n-k'$ connections
- Since $n-k' > n-k$, have $k' < k$
- Assume for contradiction: every connection in F' links two cities that are in same cluster in F
  - Then each cluster in F' sits inside a cluster in F
  - So $k' \geq k$
- This contradiction gives lemma

Proof of Kruskal II — minimality [November 19]

Proof that final network is optimal, using lemma:

- Let $T = \{t_1, \ldots, t_{n-1}\}$ be final network produced by Kruskal, with connections t_i ordered, as created, from cheapest to most expensive ($c(t_1) \leq c(t_2) \leq \ldots c(t_{n-1})$, where $c(t_i)$ is cost of connection t_i)
- Assume for contradiction that $H = \{h_1, \ldots, h_{n-1}\}$ (with $c(h_1) \leq c(h_2) \leq \ldots c(h_{n-1})$) is a cheaper network
- Let k be *first* step at which H beats T ($\sum_{i=1}^{k} c(h_i) < \sum_{i=1}^{k} c(t_i)$); because Kruskal starts with globally cheapest connection, $k \neq 1$
- Since $\sum_{i=1}^{k-1} c(h_i) \geq \sum_{i=1}^{k-1} c(t_i)$, must have $c(h_k) < c(t_k)$
- Look at networks $T_{k-1} = \{t_1, \ldots, t_{k-1}\}$ and $H_k = \{h_1, \ldots, h_k\}$ (both without cycles)
- By Lemma, there's connection, say h_ℓ, in H_k that joins two clusters in T_{k-1}
- h_ℓ was available at step k in running of Kruskal's algorithm, and $c(h_\ell) \leq c(h_k) < c(t_k)$, so Kruskal should *not* have chosen connection t_k at this point.
- This contradiction proves that H couldn't have existed, T is minimum cost connected network

Three other MST algorithms [November 19]

1. **Greedy subtraction**: Essentially the same as Kruskal
   - Start with all possible connections
   - One-by-one delete most expensive connection whose deletion would not disconnect the network
   - Stop when no connection can be deleted without disconnecting the network
2. **Prim's algorithm**: Independently discovered by Jarnik (19030) Prim (1957) and Dijkstra (1959)
   - Pick an arbitrary city to start from
   - Build the cheapest connection out of that city (may not be globally cheapest city), to form cluster of size 2
   - Build the cheapest connection out of the cluster of size 2, to form cluster of size 3
   - Repeat, until cluster has grown to size n
3. **Greedy and Selfish algorithm**: (Only works if all connection costs are different)
   - Each city, independently of all the others, begins building the cheapest connection out of the city, stopping either when connections reaches target city, or meets target city's connection half-way
   - There are now a number of clusters of cities. Each cluster, independently of all the others, begins building the cheapest connection out of the cluster
   - Repeat until there's one cluster

Games [November 21]

Here are some of the characteristics of the games that we will study:

1. Two players [US politics, not European politics]
2. Zero-sum: whatever Player 1 wins in the end, Player 2 loses, and vice-versa [MLB, not most economic trade]
3. Perfect information: both players have full access to the rules, full access to the current state of play, and full knowledge of the moves that are available to both players [Chess, not Battleship]
4. Deterministic: both players decisions on how to play fully determines the outcome [Tic-Tac-Toe, not poker]

Some simple examples:

1. **Rock-Paper-Scissors**: Two players each throw a hand, either clenched (rock), open (paper), or with middle and index finger split (scissors). Rock crushes (beats) scissors cuts (beats) paper covers (beats) rock; players showing the same item draw. The loser pays the winner $1.
2. **Two-finger Morra**: Two players each throw out a hand, showing some number of fingers (between 1 and 2), while at the same time shouting out a guess (some number between 2 and 4) as to the total number of fingers that will be shown. A player who guesses correctly get points from the opponent equal to the number of fingers showing. If no-one guesses correctly the game is a tie.

Chess as a two-person zero-sum game [November 21]

Game tree:

- White has 20 possible opening moves
- Black has 20 possible responses
- From here on, number of possible moves depends on previous moves
- Get big "Game tree": $\approx 10^{47}$ nodes (finite by "draw rule")

Strategy for White:

- Specification of opening move M_1
- For each possible response i of Black, specification of second move M_{2i}
- For each i, and response j of Black to M_{2i}, specification of third move M_{3ij}
- ...

Strategy for Black:

- For each possible opening move i of Black, specification of opening move N_{1i}
- ...

Payoffs: if White uses strategy A, Black uses strategy B

- $+1$ if unique game determined by A and B ends with White winning
- -1 if it ends with Black winning
- 0 if it ends with draw

Notation for games [November 21]

Matrix games:

- P_1 (Player 1) has available strategy set $\{s_1, \ldots, s_m\}$
- P_2 (Player 2) has available strategy set $\{t_1, \ldots, t_n\}$
- If P_1 plays strategy s_i, and P_2 plays strategy t_j, then there is a payoff a_{ij} (from P_2 to P_1 if $a_{ij} \geq 0$, from P_1 to P_2 if $a_{ij} < 0$)
- *Payoff matrix*: $A = (a_{ij})_{1 \leq i \leq m,\ 1 \leq j \leq n}$
- Both players know all rules, and payoff matrix

Some meta comments:

- We'll always assume a_{ij} are known real numbers; *utility theory* deals with setting these values
- Conditions under which decision might be made:
  - *with certainty*: as in all of our LP examples
  - *with risk*: in the presence of randomness, with known probabilities
  - *with uncertainty*: in the presence of randomness, with *un*known probabilities — our games roughly fall into this category

Maximum security strategies [November 24]

First principle of games (maximization of *security level* (worst-case gain)):

A rational player will play in such a way as to maximize his gain under the assumption that the opponent will respond in the way that is most damaging for the player (as opposed to most advantageous)

Security level for $P1$ **for strategy** s_i:

$$\min_j a_{ij} \text{ (minimum entry in row } i \text{ of payoff matrix)}$$

Security level for $P2$ **for strategy** t_j:

$$\max_i a_{ij} \text{ (maximum entry in column } j \text{ of payoff matrix)}$$

Maximum security level for $P1$:

$$u_1 = \max_i \min_j a_{ij} \text{ (convention: occurs in row } h)$$

Maximum security level for $P2$:

$$u_2 = \min_j \max_i a_{ij} \text{ (convention: occurs in column } k)$$

Observations about maximum security strategies [November 24]

The following are equivalent:

1. $u_1 = u_2$
2. (s_h, t_k) is a stable pair of strategies
3. A has a *saddle point*: an entry that is simultaneously a row minimum and a column maximum; in particular, entry (h, k) is a saddle point

When any one of these things happens, say that (s_h, t_k) is a *solution* to the game, and the *value* of the game is the common value of u_1, u_2

The following are equivalent:

1. $u_1 < u_2$
2. (s_h, t_k) is a not a stable pair of strategies
3. A does not have a saddle point

Second principle of games (desire for equilibrium):

Rational players tend to play in a way that is stable

Mixed strategies [December 1]

Mixed strategy for P_1: $(x_1, \ldots, x_m)$, each $x_i \geq 0$, $\sum_i x_i = 1$ (interpretation: P_1 plays strategy s_i with probability x_i). Denote by S the set of all possible strategies for P_1 (S includes the *pure* strategies, "always play s_i").

For P_2: $(y_1, \ldots, y_n)$, each $x_i \geq 0$, $\sum_i x_i = 1$; T is set of mixed strategies for P_2.

Expected payoff when P_1 uses strategy $X = (x_1, \ldots, x_m)$ and P_2 uses strategy $Y = (y_1, \ldots, y_n)$:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_i y_j = XAT^t.$$

Security level for P_1 playing strategy X_1: worst-case expected payoff over all of Y's strategies, or

$$\min_{Y \in T} X_1 A Y^t.$$

For P_2 playing Y_2: $\max_{X \in S} XAY_1^t$.

Theorem 9.4.1: The worst-case expected payoff for P_1 playing X_1 is always achieved at one of Y' pure strategies:

$$\min_{Y \in T} X_1 A Y^t = \min_{1 \leq j \leq n} X_1 A e_j^t$$

where e_j is the jth standard basis vector in $\mathbb{R}^n$; analogous statement for P_2.

Optimal security levels [December 3]

Optimal security level for P_1: Maximum, over all mixed strategies X, of the security level of X (ie, the maximum over all X of the worst-case expected payoff to P_1 when he plays X)

$$\begin{aligned} v_1 &= \max_{X \in S} \min_{Y \in T} XAY^t \\ &= \max_{X \in S} \{\text{security level of } X\} \\ &= \max_{X \in S} \min_{j=1,\ldots,n} XAe_j^t \end{aligned}$$

For P_2:

$$\begin{aligned} v_2 &= \min_{Y \in T} \max_{X \in S} XAY^t \\ &= \min_{Y \in T} \{\text{security level of } Y\} \\ &= \min_{Y \in T} \max_{i=1,\ldots,m} e_i AY^t \end{aligned}$$

The fundamental theorem [December 3]

Theorem (proved by Jon von Neumann in 1928): for all matrix games,

$$v_1 = v_2$$

In other words, there is some number v, the *value* of the game, such that P_1 can obtain a security level of v by playing a certain strategy X_0, but can't obtain a higher security level; and P_2 can obtain a security level of v by playing a certain strategy Y_0, but can't obtain a lower security level. The pair of strategies (X_0, Y_0) constitutes a *solution* to the game. It satisfies the principle of maximizing security for each player, and it satisfies the principle of stability:

Stability (equilibrium): In response to Y_0, no strategy has a greater expected payoff than X_0 ($XAY_0^t \leq X_0AY_0^t$ for all $X \in S$; so P_1 has no incentive to change strategy from X_0, knowing that P_2 will be playing Y_0), AND, in response to X_0, no strategy has a lower expected payoff than Y_0 ($X_0AY_0^t \leq X_0AY^t$ for all $Y \in T$; so P_2 has no incentive to change strategy from Y_0, knowing that P_1 will be playing X_0)

Proof idea: P_1 sets up the problem of calculating v_1 as a linear programming problem, and P_2 does the same for v_2. the two linear programming problems turn out to be dual to one another, so the theorem is a corollary of the duality theorem.

Fairness: If $v = v_1 = v_2 = 0$, game is *fair*; if $v > 0$, game favours P_1, if $v < 0$, game favours P_2.

Game theory LP summary [December 5]

Game matrix: $A = (a_{ij})_{i=1,\ldots,m,\ j=1,\ldots,n}$ with all $a_{ij} \geq 0$ (if not, add universal constant to all entries, and subtract it from final value)

P_1's linear program to find v_1: Minimize $1/w = x_1' + x_2' + \ldots + x_m'$ subject to

$$\begin{aligned} a_{11}x_1' + a_{21}x_2' + \ldots + a_{m1}x_m' &\geq 1 \\ a_{12}x_1' + a_{22}x_2' + \ldots + a_{m2}x_m' &\geq 1 \\ &\ldots \\ a_{1n}x_1' + a_{2n}x_2' + \ldots + a_{mn}x_m' &\geq 1 \end{aligned}$$

and all $x_i' \geq 0$. Minimization problem, constraints read off *columns* of A.
$v_1 = w'$ at min; optimum strategy $X_0 = (x_1, \ldots, x_m)$ found by $x_i = v_1 x_i'$

P_2's linear program to find v_2: Maximize $1/z = y_1' + y_2' + \ldots + y_m'$ subject to

$$\begin{aligned} a_{11}y_1' + a_{12}y_2' + \ldots + a_{1n}y_n' &\leq 1 \\ a_{21}y_1' + a_{22}y_2' + \ldots + a_{2n}y_n' &\leq 1 \\ &\ldots \\ a_{m1}y_1' + a_{m2}y_2' + \ldots + a_{mn}y_n' &\leq 1 \end{aligned}$$

and all $y_i' \geq 0$. Maximization problem, constraints read off *rows* of A.
$v_2 = z'$ at max; optimum strategy $Y_0 = (y_1, \ldots, y_n)$ found by $y_i = v_2 y_i'$

Duality: $v_1 = v_2$ since P_1 and P_2's problems are dual to one another

Gale-Shapley algorithm [December 10]

Input: n men, n women, their full preference lists.

Algorithm: As long as there are some unengaged men and women:

- Choose an unengaged women, W say, arbitrarily.
- W proposes to the highest ranked man on her list to whom she has not yet proposed, m say.
- m responds by
  - accepting W's proposal (and becoming engaged to W) if either
    - ⋆ — m is currently unengaged, or
    - ⋆ — m prefers W to his current partner (in which case his current partner becomes unengaged)
  - rejecting W's proposal if — m prefers his current partner to W.

When there are no unengaged men and women, all engaged couples marry, and algorithm stops.

Gale-Shapley theorem (1962): For *all* $n!^{n^2}$ possible input preference lists, output is a stable marriage.

Proof of Gale-Shapley I [December 10]

Algorithm terminates with matching:

- Algorithm doesn't allow for polygamy in engagements, so while there are unengaged people there are as many unengaged women as men.
- A man will always accept first proposal received.
- So, while there are unengaged people, there is an unengaged woman W, and there is at least one m who has not yet received a proposal, and will accept a proposal from W.
- This shows that algorithm will terminate in no more than n^2 steps (time taken for each woman to go completely through her preference list).

Proof of Gale-Shapley II [December 10]

Output matching is stable:

- Pick pair (W, m) with Wm not in the matching.
- Let n be W's partner, and V be m's partner, in the final matching.
- Suppose that $m >_W n$ (W prefers m to her current partner). Is it possible that $W >_m V$ (m prefers W to his current partner)?
- Well: since $m >_W n$, we know that at some point W proposed to m (W works *down* her preference list to her final partner), and since W ended up matched to n in final matching, we know that at some subsequent point (or maybe at the moment of proposal) m rejected W.
- Why would m reject W? Because he was either engaged to at the time of the proposal, or received a proposal later from, some women Z that he preferred to W. Maybe $Z = V$, or maybe V came later still, but either way, since m works *up* his preference list as the algorithm goes on, we conclude that m prefers V to W.
- Hence, (W, m) is *not* an instability, and the final matching is stable.

Whom does the Gale-Shapley algorithm favour? [December 10]

Women propose: The resulting stable matching is independent of the order in which proposals happen. It is best-case for the *women*: each women ends up matched with the highest ranked man on her list that she could ever be matched with in any stable matching. It is worst-case for the men: each man ends up matched with the lowest ranked woman on his list that he could ever be matched with in any stable matching.

Men propose: The resulting stable matching is independent of the order in which proposals happen. It is best-case for the *men*, worst-case for the women.

Median matching: Each woman looks at all possible stable matchings. She writes down a list of all the men she could possibly be matched with in a stable matching, with repetitions if there are multiple stable matchings in which she is matched to the same man. She orders that multi-list in order of her preference, and finds the man whose name appears in the middle of the list. This man is her *median partner*. **Fact**: if m is W's median partner, then W is m's median partner. The matching in which each women is matched with her median partner (equivalently, each man is matched with his median partner) is stable. **Another fact**: know efficient algorithm is known to find this "median" stable matching.

Variants of the problem [December 10]

- **Hospitals-residents**: partial preferences allowed, "polygamy" allowed (hospitals take in multiple residents) (Google: "National Resident Matching Program").
- **Maximum weight assignment**: each possible partnership gets a score; aim is to pair all of the men and women into n partnerships in a way that maximizes the sum of the scores
- **Stable roommates**: $2n$ people, each has preference ranking of other $2n - 1$, what's required is n pairs without an instability (gender-blind).

2012 Nobel prize for Economics was awarded to Alvin Roth and Lloyd Shapely, for their work on these problems.