Thursday, Sept 1 Happy First Day!

1. Introductions
   - ↳ who's me?
   - ↳ graphs
   - ↳ who's you?
2. Logistics
   - ↳ syllabus
   - ↳ calendar
   - ↳ moodle
   - ↳ drive

Pause: build a sentence as a table that describes how you want the classroom to feel.

3. What is Discrete Op?
   - ↳ motivating ex
4. Outro
   - ↳ small work: Intro Survey

Math 494: Discrete Optimization

Example: Example: One of the the things we're gonna work on as we go in this class is to trust our mathematical instincts. Being as specific as you can be, what does your table think discrete optimization entails?

max or min val. of something, best/worst case scenario, perfect matching, optimizing, scheduling

Shortest/longest path

In general, optimization problems of any sort have two distinct parts:

- objective: what are you trying to optimize?
- Constraints: whats holding you back?

In particular, discrete optimization looks at finding some sort of desirable substructure in a discrete object, often a *graph*. Let's go through some examples.

Example: Suppose I have a very fussy nephew who will eat three types of foods: apples, bananas, and cucumbers. As a growing boy, he also needs to meet his daily percentages of three vitamins: A, C, and K. As a parent of young kids, my sister wants to do this as cheaply as possible. Below is a table of useful information.

	A	C	K	\$
Apples	60	26	6	1.53
Bananas	3	33	1	0.37
Cucumbers	2	7	12	0.18

a.) What are we trying to optimize? What conditions do we need to meet?

$\hookrightarrow$ minimize total cost $\hookrightarrow$ vitamin intake ≥ 100

b.) What's *a* solution to this problem? Remember, a solution doesn't have to be an optimal solution! What do you think this means about number of solutions we need to check?

2A, 2B, 8C, 50 C, 100 of all $\hookrightarrow$ technically ∞

c.) Before we can solve an optimization problem, we need to mathematically formulate it. What sorts of variables/equations/functions do you need to fully encapsulate this problem? Try to build these equations, defining any variables you need.

A = # apples
B = # bananas
C = # cucumbers

$$\min \ 1.53A + 0.37B + 0.18C$$

$$60A + 3B + 2C \geq 100$$

$$26A + 33B + 7C \geq 100$$

$$2A + 7B + 12C \geq 100$$

$$\begin{bmatrix} 60 & 3 & 2 \\ 26 & 33 & 7 \\ 2 & 7 & 12 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \geq \begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix}$$

d.) This question is intentionally a bit vague: what sort of *geometry* are we looking at here?

Planes in $\mathbb{R}^3$

Example: A politician running for office would like to visit all 7 major cities in her district. She would like to be as efficient as possible.

a.) What are we trying to optimize? What conditions do we need to meet?

min distance traveled need to get to all and back to start.

b.) What sort of information do you need to begin to formulate this problem?

distances between cities

c.) This question in particular is is a *discrete* optimization problem. Why? What sort of structure are we looking at?

looking for a hamiltonian cycle in a graph!

d.) What sorts of challenges come from formulating this symbolically?

$$\min_{c \text{ cycle...}} C(c)$$

*e.) Suppose we wanted to solve this problem by hand. How many possible solutions are we going to need to check?

$\frac{7!}{2}$ $\qquad$ $\frac{(n-1)!}{2}$

f.) What might be a reasonable way to come up with a "good" solution?

Closest one! not a bad idea, but not guaranteed to give best

This problem is an *instance* of a well known archetype: euclidean traveling salesman

These examples are helping us get at the two steps of an optimization problem:

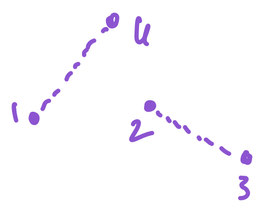
- formulation: assigning var., making equations, identifying archetypes
- solution: solver (SAGE, APMonitor, simplex), algorithm.

While most of the mathematical theory will fall into the second step, do not discount the practice needed for the first! A lot of our time will be spent trying to recognize a problem as a particular type and giving it the mathematical structure to pass to a solver.

As for that second step, its feasibility is highly dependent on the type of problem we're looking at. Let's compare our Euclidean TSP with another sort of problem, the Euclidean matching problem.

Example: Suppose we have a set of $2n$ points in the plane, and we want to draw n lines between them such that the total distance of the lines is minimal.

a.) Draw six arbitrary points, then try to come up with an optimal solution.



(1,6)
(2,3)
(4,5)

b.) What's our objective? What conditions need to be met? What might a solution look like?

minimize distances

every point gets exactly 1 line

lines pairs.

c.) If we wanted to check every possible solution by hand, how many are we looking at?

$$\frac{(2n)!}{n!\,2^n}$$

( )( )( )( ) .. ( )

d.) What might be a way to come up with a "good solution"?

pick the closest

Hopefully we can see some similarities in our Euclidean op problems:

points in plane, really big total soln space,
greedy isn't great.

However, only matching → Blossom Alg is *efficiently* solvable! What does that mean?

there a set of instructions that are executable
to find the solution quickly

Our first foothold for Discrete Op is going to be the *linear program*. An LP is a general framework for describing an optimization problem, and it (and it's cousin the *integer program*) will follow us all semester.

Example: A farmer has 12 acres of land to plant either soybeans or corn. At least 7 acres have to be planted. Planting one acre of soybeans costs \$200 and one acre of corn costs \$100. Budget for planting is \$1500. The sale from one acre of soybeans is \$500 and from corn is \$300. How many acres of what should be planted to maximize profit?

We're going to have two variables s and c.

a.) What's our objective? Write a function for it.

b.) What are our constraints? Write functions for them as well.

c.) Build a linear program.

It cannot be overstressed how *useful* a linear program is. The development of LPs in the late 1940s by George Dantzig is credited as one of the most profitable developments in mathematics in the 20th century.

Next time: the general definition of an LP, more formulation,