CSE 421 Winter 2025
Lecture 20: Linear Programming

Glenn Sun (substitute)

http://www.cs.uw.edu/421

Today's lecture will involve pollev.com (not graded).

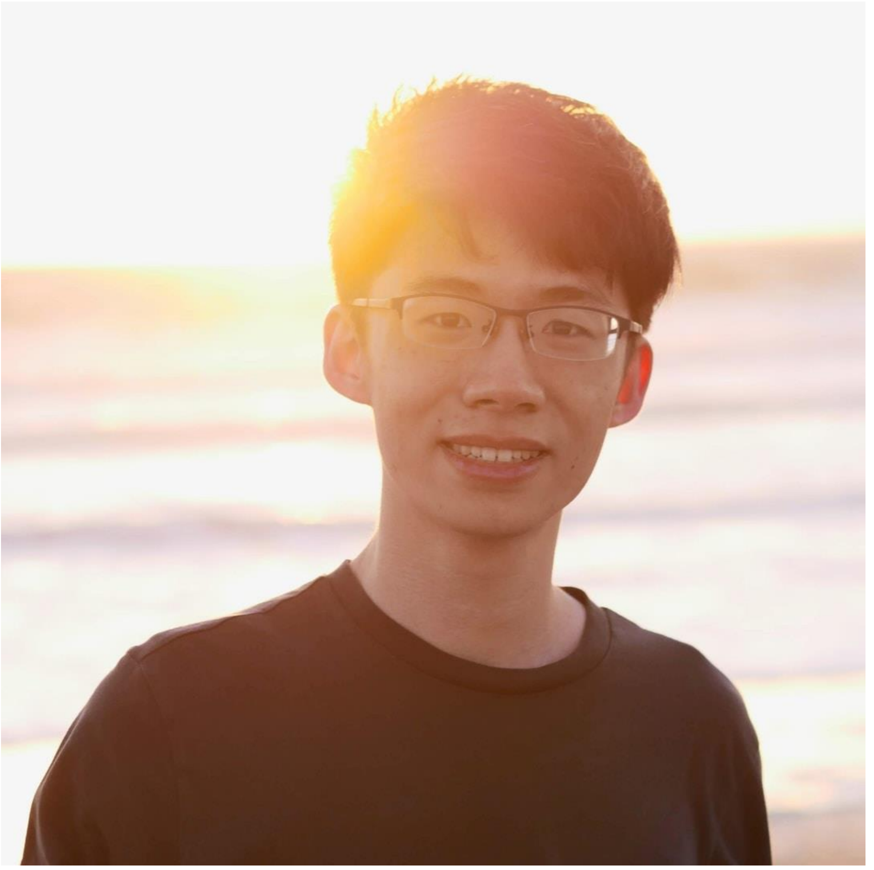
pollev.com/glennsun

Use **UW login**, not guest

Hi!

I’m Glenn, one of your TAs.

2nd year PhD student in Theory,
under Paul Beame



Extra OH:

In Nathan’s office after class today,
2:30pm–3:30pm

What is linear programming?

- Optimize **real-valued, linear** functions with constraints
- Widely used in business and operations modeling/research
- Many packages for Python, Excel, etc.
- We cover basics — take MATH 407 to learn more!

A boba shop has **1000 oz of boba** and **3000 oz of tea** and can make drinks of any size.

A **standard** drink is **10% boba, 90% tea** and sells for **20 cents/oz**.

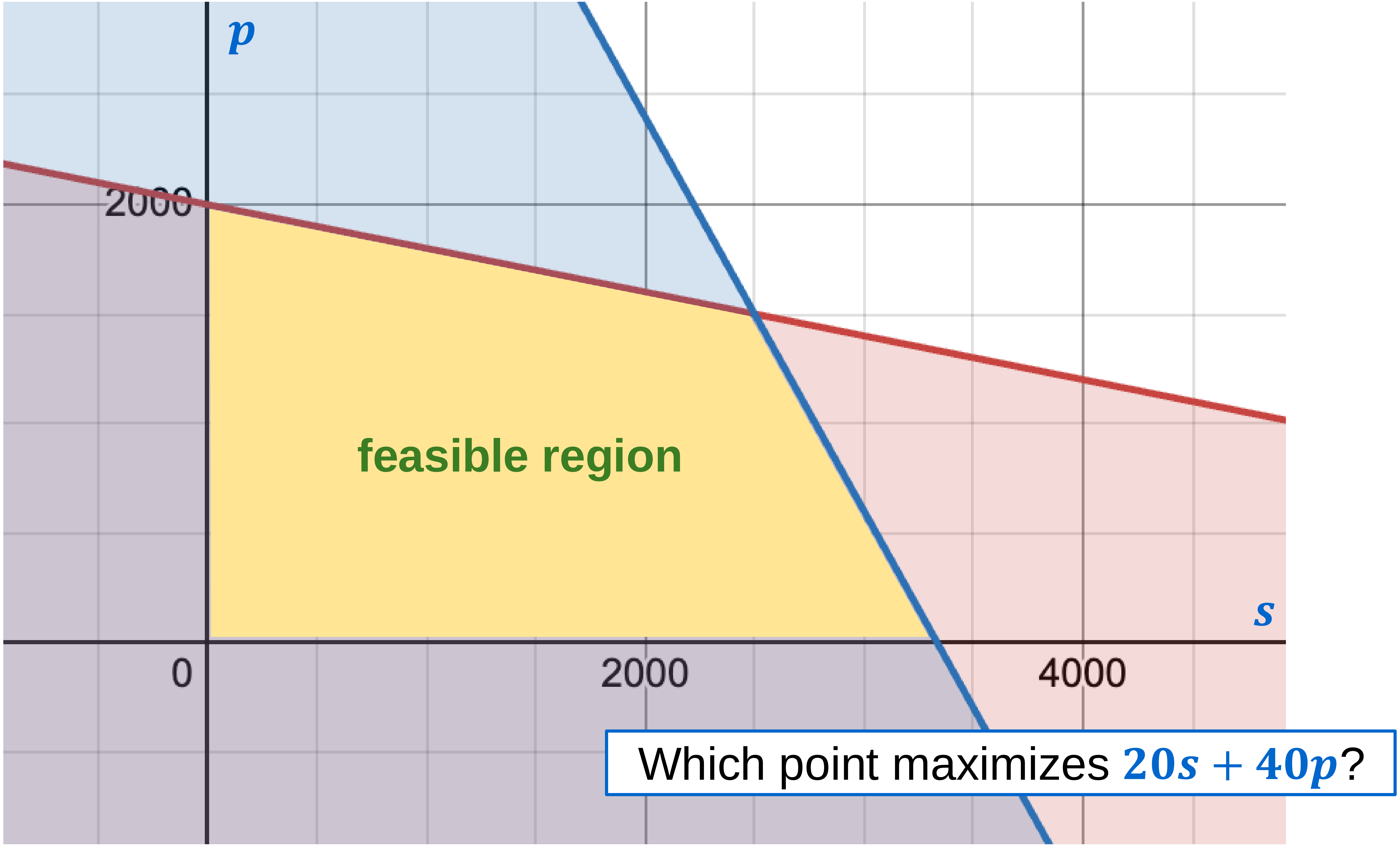
A **premium** drink is **50% boba, 50% tea** and sells for **40 cents/oz**.

What is the **maximum revenue** that the boba shop can make?

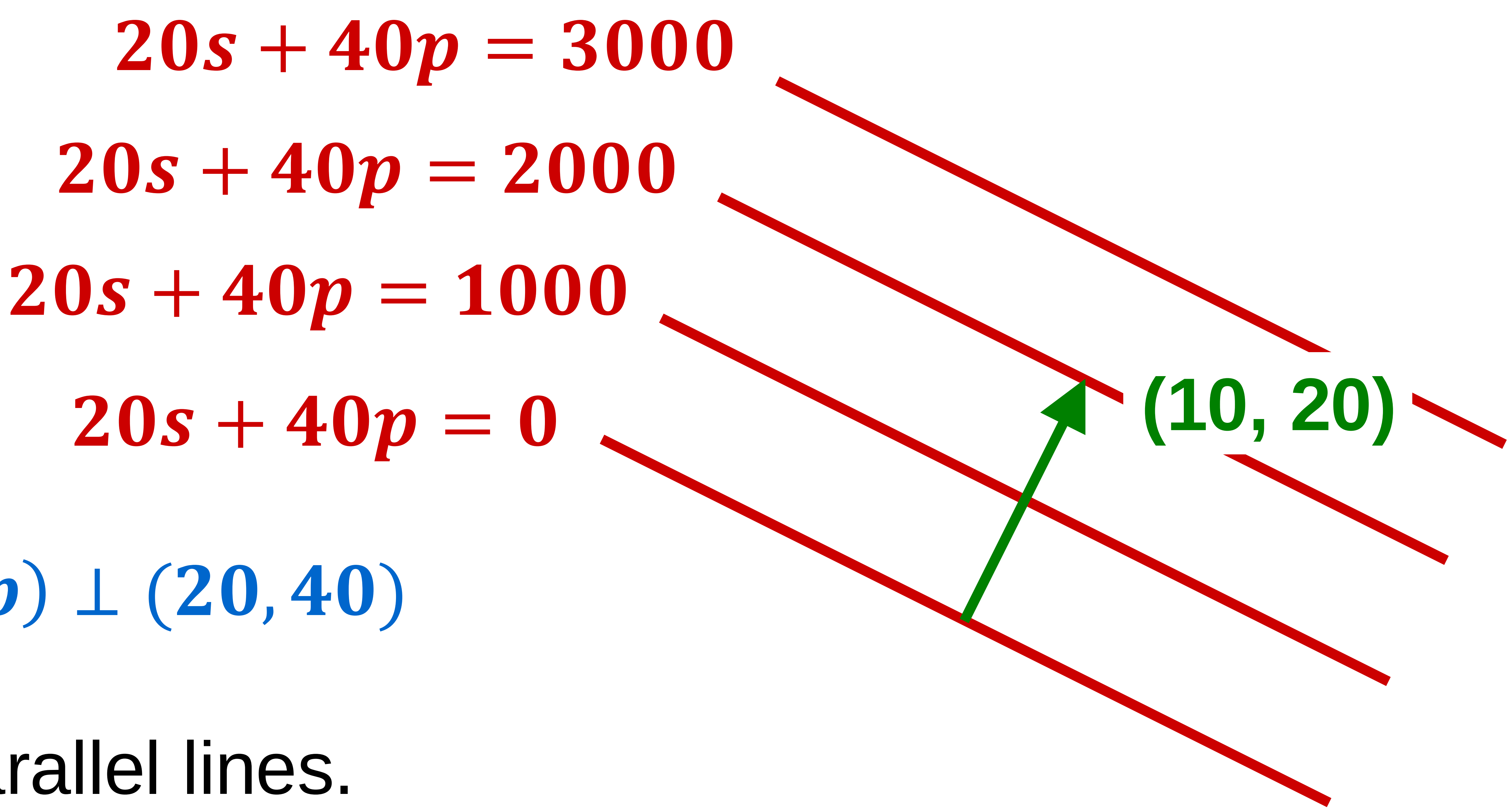
- s = amount of **standard** drink produced in oz
- p = amount of **premium** drink produced in oz

$$\begin{aligned} \textbf{maximize} \quad & 20s + 40p \\ \textbf{subject to} \quad & 0.1s + 0.5p \leq 1000 \\ & 0.9s + 0.5p \leq 3000 \\ & s, p \geq 0 \end{aligned}$$

https://www.desmos.com/calculator/4zlr9g7tnn


p
2000
feasible region
s
0
2000
4000
Which point maximizes $20s + 40p$?

Intuition: To maximize $20s + 40p$, we should **go in the (20, 40) direction** as far as possible.

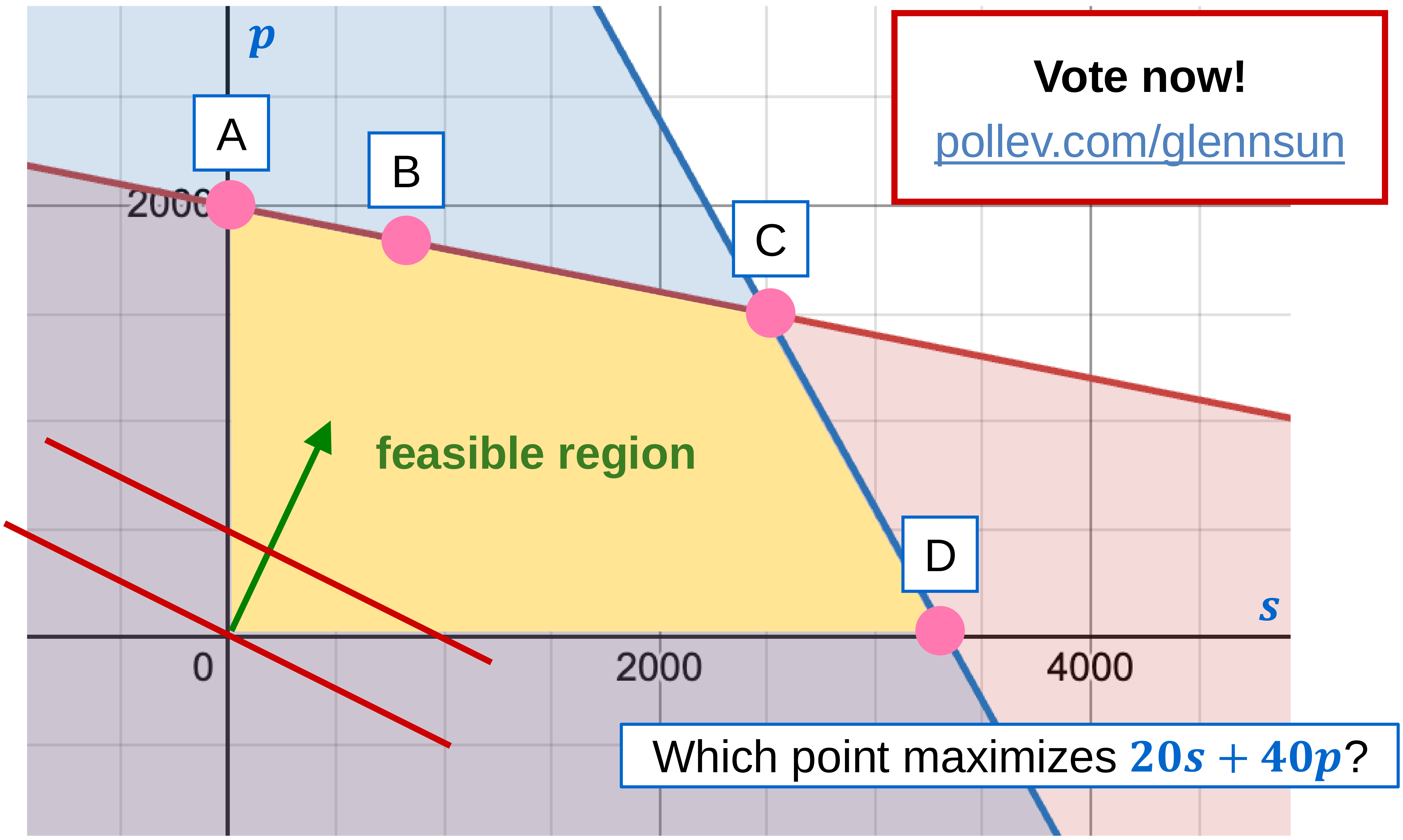


What I mean by this:

$20s + 40p = 0$ when $(s, p) \perp (20, 40)$

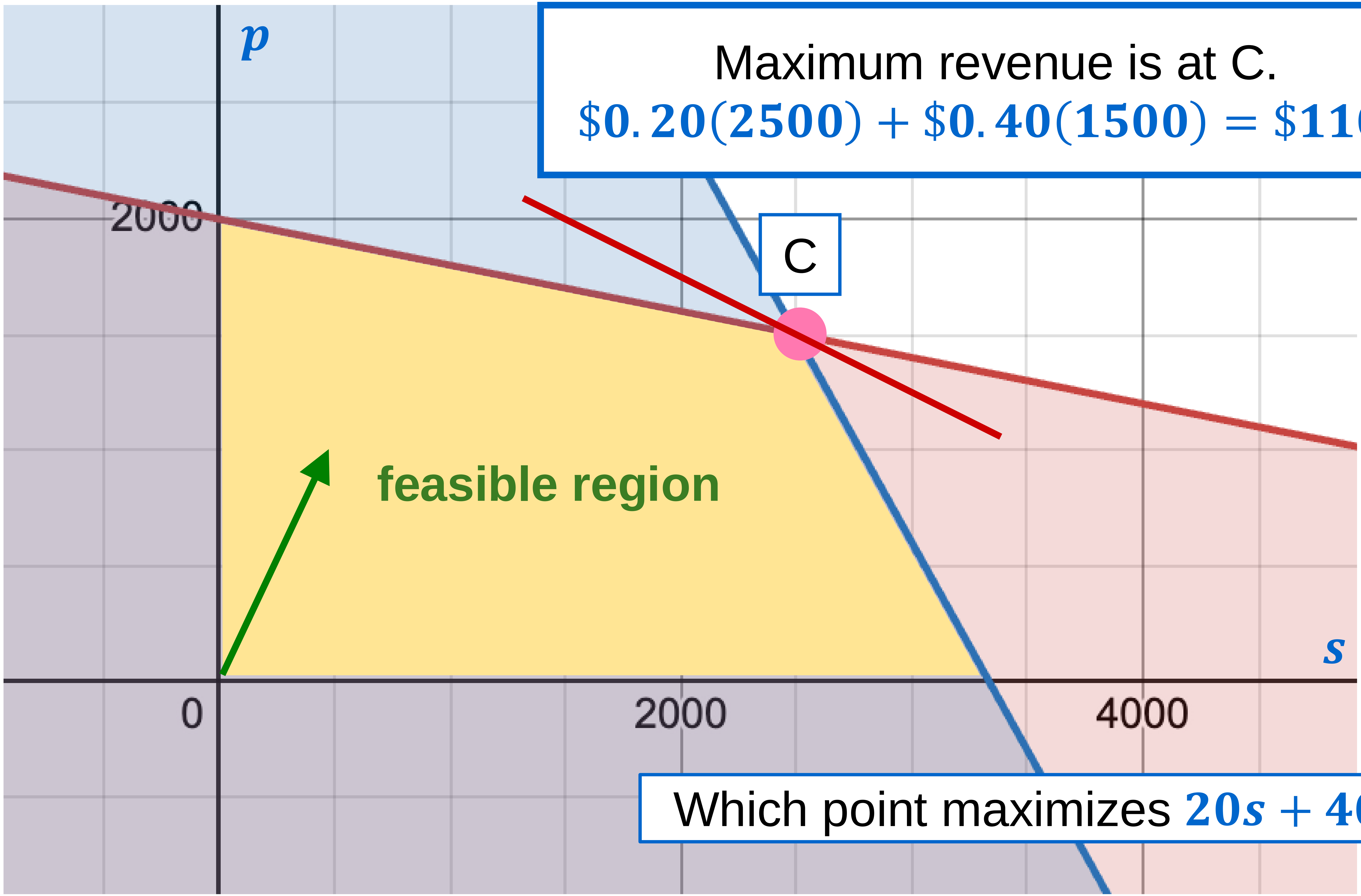
So $20s + 40p = k$ are parallel lines.

To maximize, find the **farthest line** in the (20, 40) direction that still **touches the feasible region.**


p
A
B
C
D
2000
feasible region
s
0
2000
4000
Vote now!
pollev.com/glennsun
Which point maximizes $20s + 40p$?

Maximum revenue is at C.

$\$0.20(2500) + \$0.40(1500) = \$1100$



Which point maximizes $20s + 40p$?

Theorem. (Fundamental Theorem of Linear Programming)
If the feasible region is bounded and nonempty, then **some vertex is an optimal solution**.

$\Rightarrow$ Brute force algorithm: Compute the objective at every vertex

But that takes exponential time.

- An $\boldsymbol{n}$-dimensional cube is formed by $\mathbf{2}\boldsymbol{n}$ constraints/faces and has $\mathbf{2}^{\boldsymbol{n}}$ vertices.

There ***are*** fast algorithms for linear programming. (Lecture 25)

Today: How to **set up** various problems as linear programs.

Linear programming solves problems of the form:

$$\begin{aligned} \textbf{maximize} \quad & c_1x_1 + \cdots + c_nx_n \\ \textbf{subject to} \quad & a_{11}x_1 + \cdots + a_{1n}x_n \le b_1 \\ & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \le b_m \\ & x_1, \ldots, x_n \ge 0 \end{aligned}$$

(equivalently)

$$\begin{aligned} \textbf{maximize} \quad & c^\top x \\ \textbf{subject to} \quad & Ax \le b \\ & x \ge 0 \end{aligned}$$

This is **standard form:**

- maximization
- $\le$ inequalities with constant RHS
- nonnegative x's

Vote now!

pollev.com/glennsun

Which of these situations can be converted into standard form?

- a constraint $\boldsymbol{x \geq y}$
- a constraint $\boldsymbol{x + y = 3}$
- a constraint $\boldsymbol{xy \geq 5}$
- a constraint $\boldsymbol{x \leq \min(y, z)}$
- an objective to minimize $\boldsymbol{x + 2y}$
- an objective to maximize $\boldsymbol{x^2 + y^2}$
- a variable $\boldsymbol{x}$ that may be negative

$$
\begin{array}{rr}
\textbf{maximize} & c_1x_1 + \cdots + c_nx_n \\
\textbf{subject to} & a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\
& \vdots \\
& a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \\
& x_1, \ldots, x_n \geq 0
\end{array}
$$

(standard form reminder)

Which of these situations can be converted into standard form?

- a constraint $\boldsymbol{x \geq y}$

  Yes! $\boldsymbol{-x \leq -y}$, then get $\boldsymbol{-x + y \leq 0}$

- a constraint $\boldsymbol{x + y = 3}$

  Yes! $\boldsymbol{x + y \leq 3}$ and $\boldsymbol{x + y \geq 3}$ (i.e. $\boldsymbol{-x - y \leq -3}$)

- a constraint $\boldsymbol{xy \geq 5}$

  Not possible.

- a constraint $\boldsymbol{x \leq \min(y, z)}$

  Yes! $\boldsymbol{x \leq y}$ and $\boldsymbol{x \leq z}$ (i.e. $\boldsymbol{x - y \leq 0}$ and $\boldsymbol{x - z \leq 0}$)

Which of these situations can be converted into standard form?

- an objective to minimize $x + 2y$

  Yes! maximize $-x - 2y$

- an objective to maximize $x^2 + y^2$

  Not possible.

Which of these situations can be converted into standard form?

- a variable $\boldsymbol{x}$ that may be negative

  Yes! Make two new variables $\boldsymbol{x'}$ and $\boldsymbol{x''}$, then replace every occurrence of $\boldsymbol{x}$ with $\boldsymbol{x' - x''}$. We can now have $\boldsymbol{x', x'' \geq 0}$.

$$\begin{array}{rl} \textbf{maximize} & x + 3y \\ \textbf{subject to} & -x + y \leq 10 \\ & 2x + y \leq 4 \\ & y \geq 0 \end{array} \longrightarrow \begin{array}{rl} \textbf{maximize} & x' - x'' + 3y \\ \textbf{subject to} & -x' + x'' + y \leq 10 \\ & 2x' - 2x'' + y \leq 4 \\ & x', x'', y \geq 0 \end{array}$$

Max Flow

Input: A flow network $\boldsymbol{G = (V, E)}$, source $\boldsymbol{s}$, sink $\boldsymbol{t}$, and $\boldsymbol{c : E \to \mathbb{R}^{\geq 0}}$

Goal:

maximize flow out of $\boldsymbol{s}$

subject to respecting capacities and flow conservation

We want $\boldsymbol{x_e}$ = flow on edge $\boldsymbol{e \in E}$.

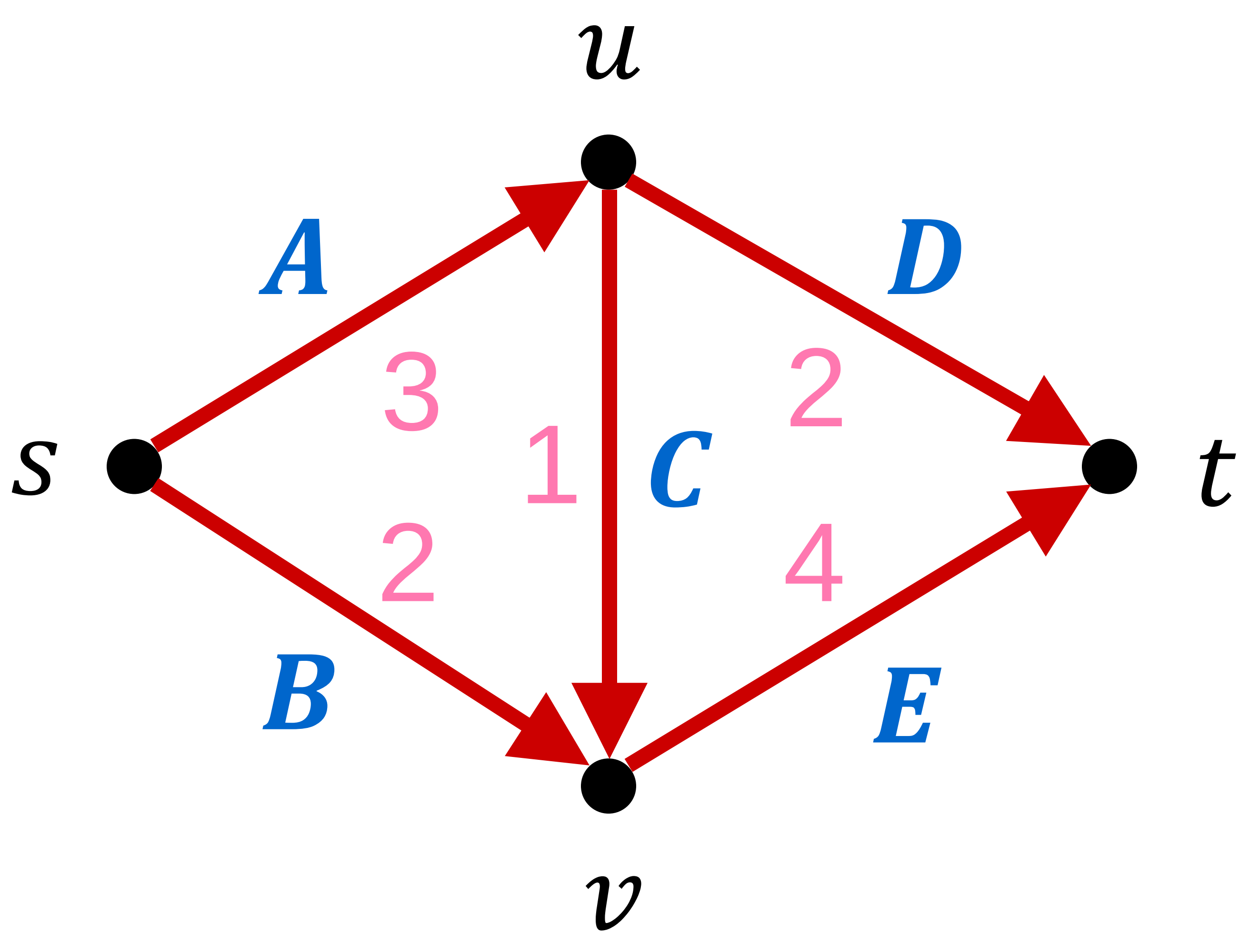
$$\begin{array}{rl} \textbf{maximize} & \sum_{e \text{ out of } s} (x_e) \\ \textbf{subject to} & 0 \leq x_e \leq c(e) \text{ for all } e \in E \\ & \sum_{e \text{ out of } v} (x_e) = \sum_{e \text{ into } v} (x_e) \text{ for all } v \in V \setminus \{s, t\} \end{array}$$

Max Flow

maximize $\sum_{e \text{ out of } s}(x_e)$

subject to $0 \le x_e \le c(e)$ for all $e \in E$

$\sum_{e \text{ out of } v}(x_e) = \sum_{e \text{ into } v}(x_e)$ for all $v \in V \setminus \{s, t\}$



maximize $x_A + x_B$

subject to $0 \le x_A \le 3$

$0 \le x_B \le 2$

$0 \le x_C \le 1$

$0 \le x_D \le 2$

$0 \le x_E \le 4$

$x_C + x_D = x_A$

$x_E = x_B + x_C$

Max Flow

maximize $\sum_{e \text{ out of } s}(x_e)$

subject to $0 \leq x_e \leq c(e)$ for all $e \in E$

$\sum_{e \text{ out of } v}(x_e) = \sum_{e \text{ into } v}(x_e)$ for all $v \in V \setminus \{s,t\}$

In standard form:

maximize $\sum_{e \text{ out of } s}(x_e)$

subject to $x_e \leq c(e)$ for all $e \in E$

$\sum_{e \text{ out of } v}(x_e) - \sum_{e \text{ into } v}(x_e) \leq 0$ for all $v \in V \setminus \{s,t\}$

$\sum_{e \text{ into } v}(x_e) - \sum_{e \text{ out of } v}(x_e) \leq 0$ for all $v \in V \setminus \{s,t\}$

$x_e \geq 0$ for all $e \in E$

Shortest Path

Input: A directed graph $\boldsymbol{G = (V, E)}$ with vertices $\boldsymbol{s}$, $\boldsymbol{t}$, and (possibly negative) weights $\boldsymbol{w : E \to \mathbb{R}}$

Goal: compute length of shortest path from $\boldsymbol{s}$ to $\boldsymbol{t}$

We want $\boldsymbol{x_v =}$ length of shortest path from $\boldsymbol{s}$ to $\boldsymbol{v}$.

maximize	$\boldsymbol{x_t}$
subject to	$\boldsymbol{x_v \le x_u + w(e)}$ for all edges $e = (u, v) \in E$
	$\boldsymbol{x_s = 0}$

Shortest Path

$$\begin{aligned} \textbf{maximize} \quad & x_t \\ \textbf{subject to} \quad & x_v \leq x_u + w(e) \text{ for all edges } e = (u, v) \in E \\ & x_s = 0 \end{aligned}$$

If x_v = the length of the shortest path from s to v,

then it is true that $x_v \leq x_u + w(e)$ and $x_s = 0$.

That's why we could safely include this as a constraint.

To prove "LP computes shortest path", we need the converse!

Shortest Path

Claim. The LP calculates the shortest path from s to t.

Proof. We will show that the length of the shortest path from s to t is the maximum x_t satisfying the constraints.

In general, "maximum" means: (1) possible, and (2) upper bound.

Here, need to show:

1. "There is a feasible solution to the LP in which x_t is the length of the shortest path from s to t."
2. "For all feasible solutions to the LP, $x_t \leq$ the length of the shortest path from s to t."

Shortest Path

1. "There is a feasible solution to the LP in which x_t is the length of the shortest path from s to t."

Setting $x_v =$ length of shortest path from s for all $v \in V$ is feasible.

2. "For all feasible solutions to the LP, $x_t \leq$ the length of the shortest path from s to t."

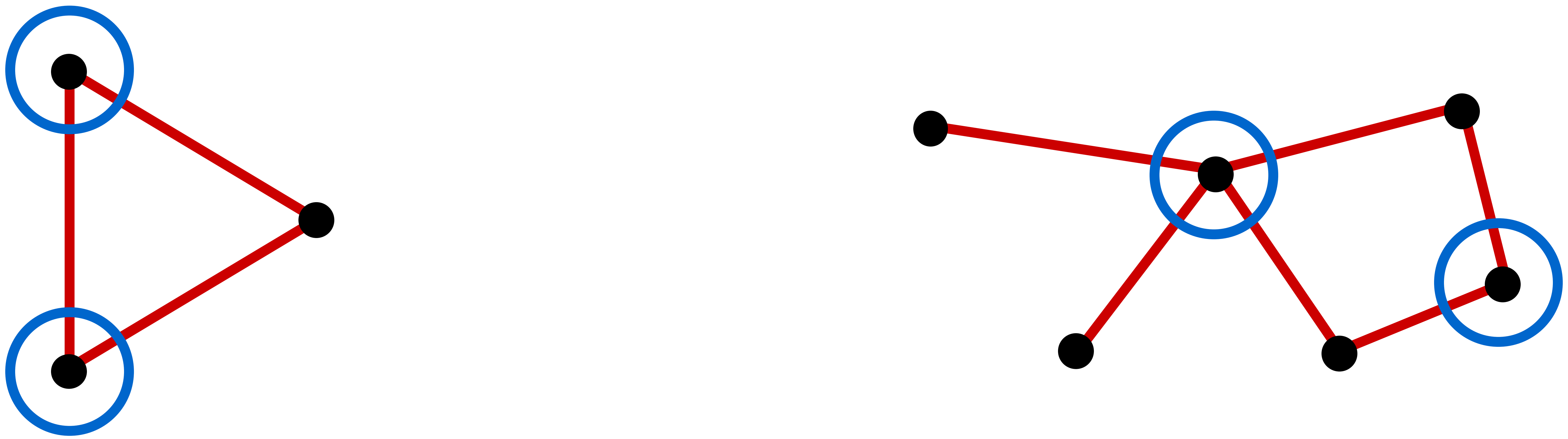
Let x_v be a feasible solution and $(s, v_1, \dots, v_k, t)$ be a shortest path.

$$
\begin{aligned}
x_t &\leq x_{v_k} + w(v_k, t) \\
&\leq x_{v_{k-1}} + w(v_{k-1}, v_k) + w(v_k, t) \\
&\ \vdots \\
&\leq 0 + w(s, v_1) + \cdots + w(v_k, t) \\
&= \text{length of shortest path from } s \text{ to } t
\end{aligned}
$$

Sneak peek: Vertex Cover?

Input: An undirected graph $G = (V, E)$

Goal: smallest subset of vertices touching all edges of G



What variables to pick?

No good choices — want to make a binary decision for vertices (in the vertex cover or not), but LPs work with real-valued variables.

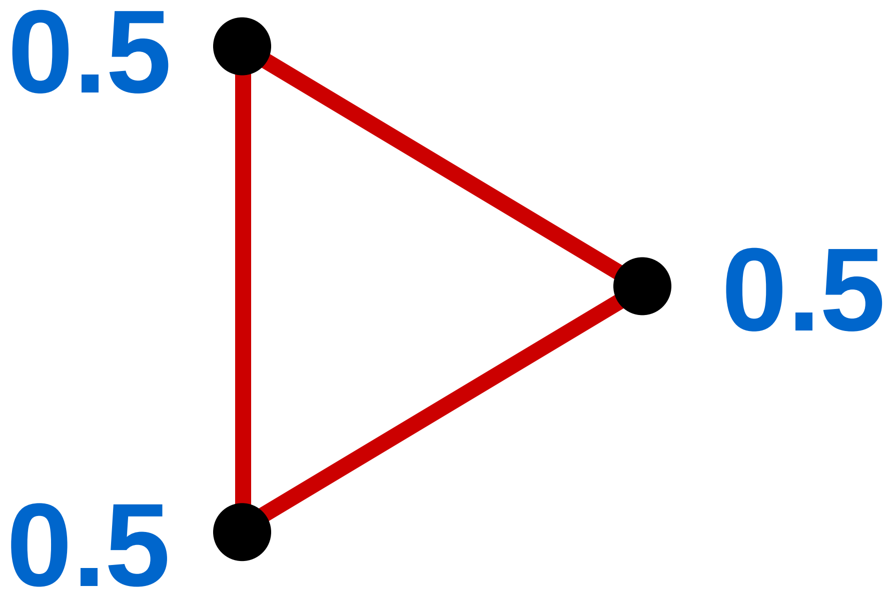
Sneak peek: Vertex Cover?

LP Relaxation: Instead of $x_v = 0$ or 1 (out/in), have $\mathbf{0} \leq x_v \leq \mathbf{1}$

$$\begin{array}{rl} \textbf{minimize} & \sum_{v \in V}(x_v) \\ \textbf{subject to} & x_u + x_v \geq \mathbf{1} \text{ for all edges } (u, v) \in E \\ & \mathbf{0} \leq x_v \leq \mathbf{1} \text{ for all vertices } v \in V \end{array}$$

Might give **"fractional solutions"**:

LP optimum = 1.5, true optimum = 2



Still useful for *approximation algorithms*, wait for Lecture 24!

Coming up on Friday…

Substitute instructor: **Owen**

Duality in Linear Programming

$$\begin{aligned} \textbf{minimize} \quad & b^\top y \\ \textbf{subject to} \quad & A^\top y \geq c \\ & y \geq 0 \end{aligned}$$