Lecture 19

Linear programming I

Chinmay Nirkhe | CSE 421 Spring 2025

Optimization problems

- Optimization problems are the most of the problems we have seen
- An optimization problem is described by some function $\underbrace{f : \Sigma \to \mathbb{R}}_{\text{optimization function}}$ and a subset $\Gamma \subseteq \Sigma$.

feasible region

domain

  - Goal is to find $x \in \Gamma$ such that for all $y \in \Gamma, f(x) \geq f(y)$ — i.e. x is the argmax of f with respect to Γ.
  - Ex.: Knapsack. $\Sigma = \{S : S \subseteq [n]\}, \Gamma = \{S : \text{weight}(S) \leq W\}, f(S) = \text{value}(S)$
  - Ex. Shortest path $s \to t$. $\Sigma = \{\text{seq . of edges}\}, \Gamma = \{\text{paths}\}, f(p) = \sum_{e \in p} w(e)$
  - Ex. Greedy. $\Sigma = \{\text{job assignments}\}, \Gamma = \{\text{non} - \text{overlapping}\}, f(x) = \text{value}(x)$

Linear programming

- An optimization problem paradigm
- Both the optimization function f and feasible region Γ are linear.

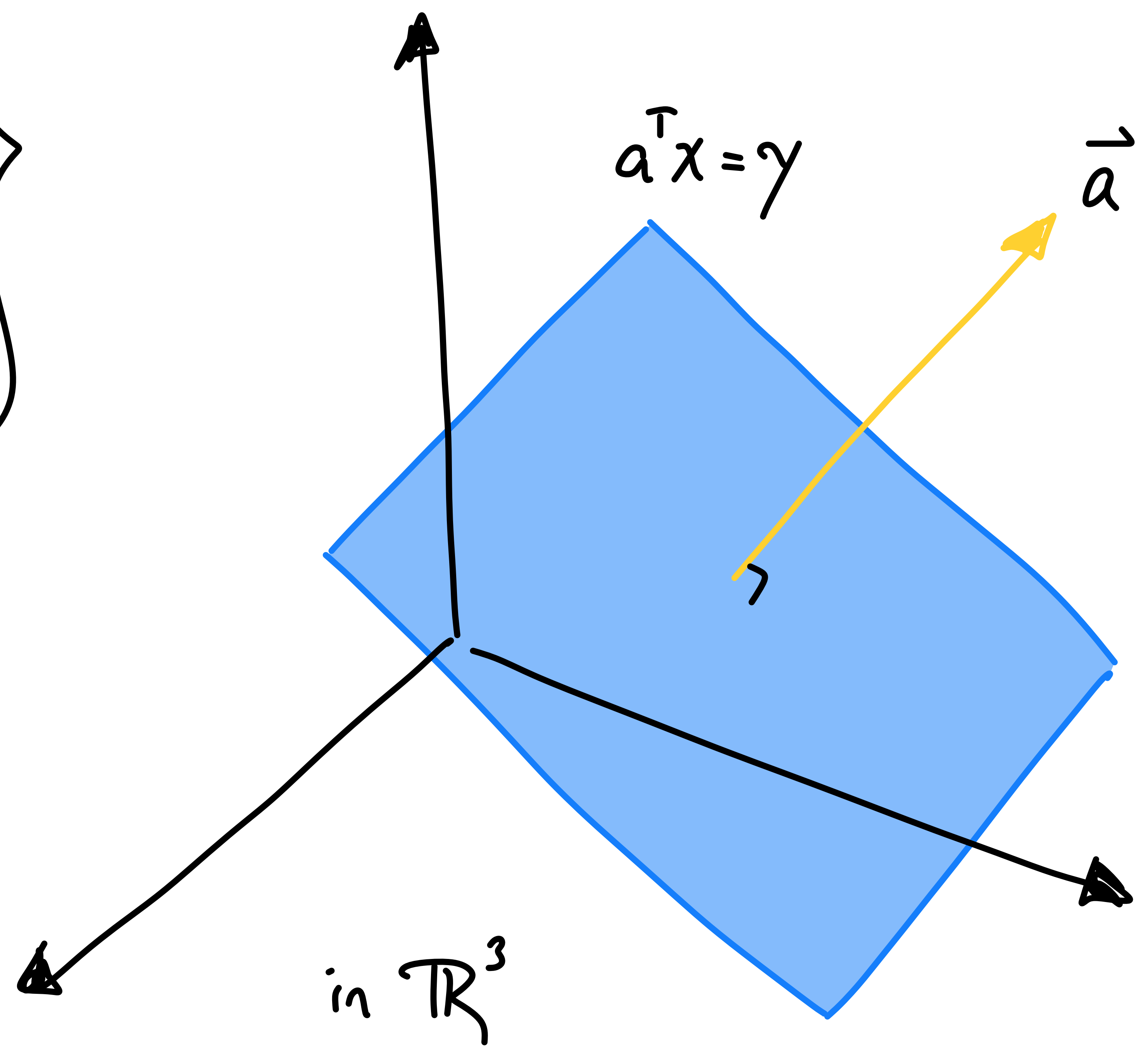
$$\max \underset{\mathbb{R}^n}{\begin{bmatrix} - & c & - \end{bmatrix}} \cdot \underset{\mathbb{R}^n}{\begin{bmatrix} | \\ x \\ | \end{bmatrix}} \quad \text{subject to} \quad \underset{\mathbb{R}^{m \times n}}{\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}} \cdot \begin{bmatrix} | \\ x \\ | \end{bmatrix} \leq \underset{\mathbb{R}^m}{\begin{bmatrix} | \\ b \\ | \end{bmatrix}} \quad \text{and} \quad \begin{bmatrix} | \\ x \\ | \end{bmatrix} \geq 0.$$

Global space $\Sigma = \mathbb{R}^n$, feasible region $\Gamma = \{x : Ax \leq b, x \geq 0\}$, opt. fn. $f(x) = c^T x$

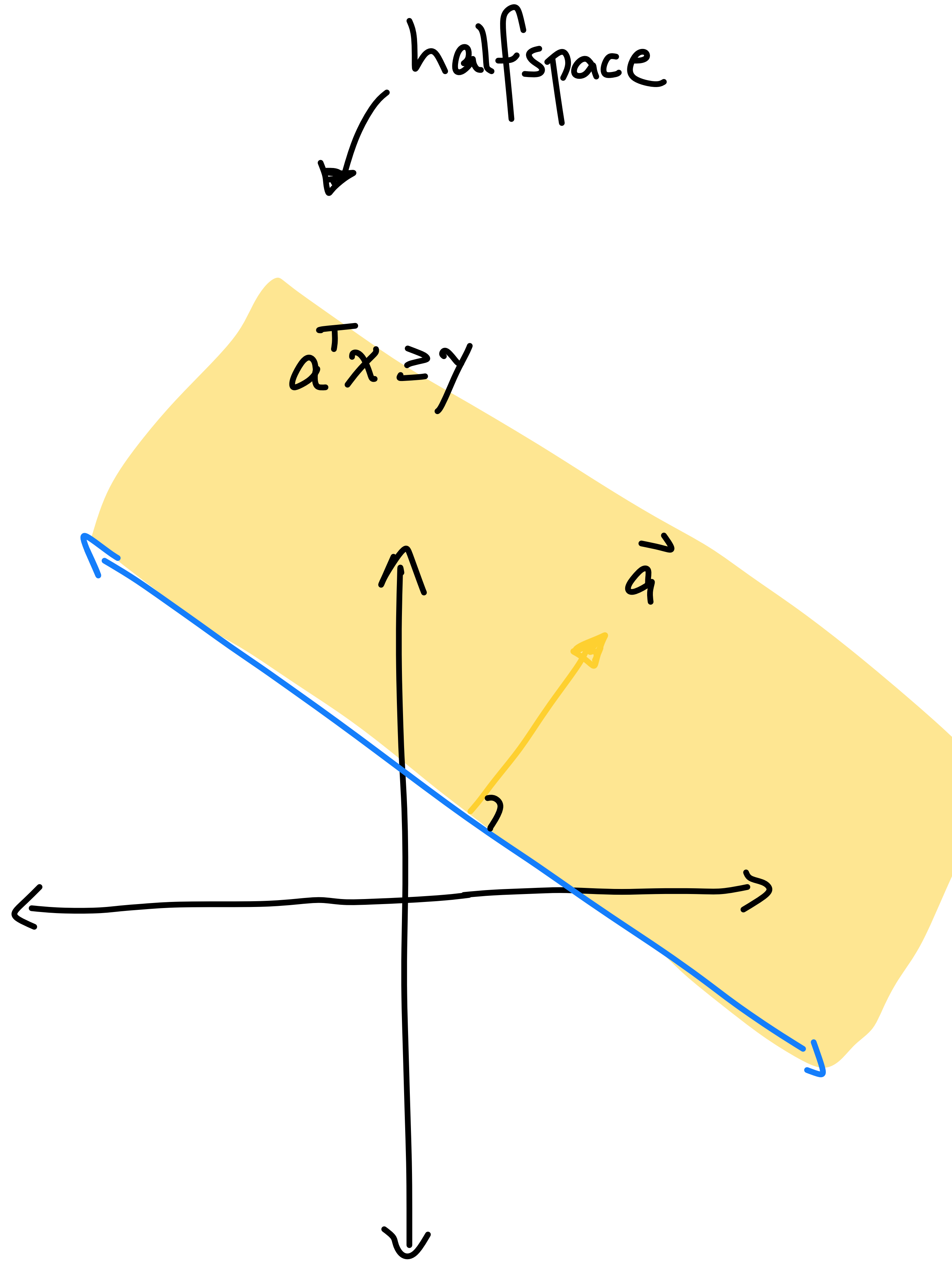
Linear algebra/geometry review

Within the space $\mathbb{R}^n$, an affine subspace of dim $n-1$ is defined by

$$\left\{ \begin{bmatrix} | \\ x \\ | \end{bmatrix} : \begin{bmatrix} - & a & - \end{bmatrix} \cdot \begin{bmatrix} | \\ x \\ | \end{bmatrix} = y \right\}$$



in $\mathbb{R}^3$

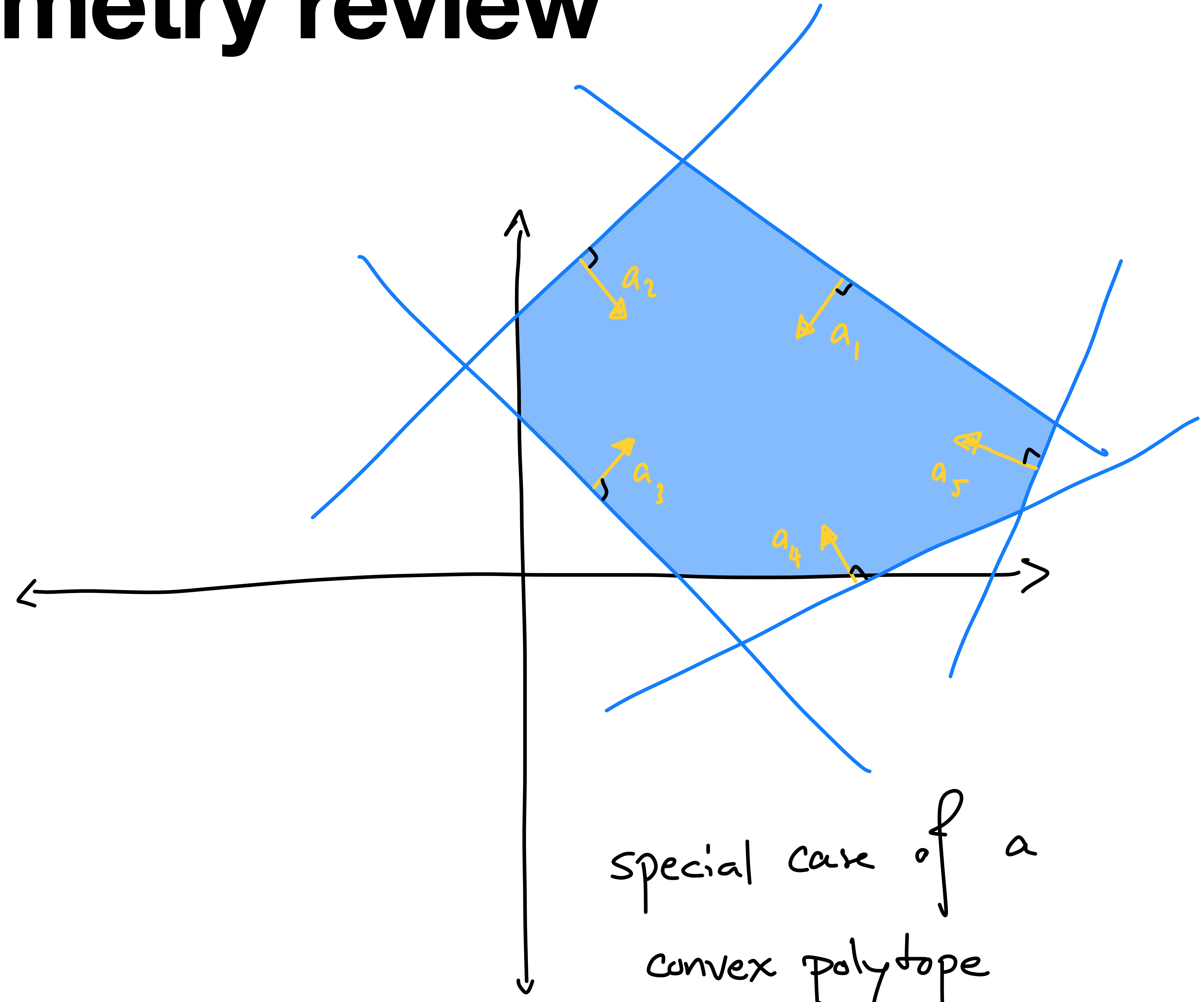


in $\mathbb{R}^2$

Linear algebra/geometry review

$$\begin{bmatrix} — \; a_1 \; — \\ — \; a_2 \; — \\ \vdots \\ — \; a_m \; — \end{bmatrix} \cdot \begin{bmatrix} | \\ x \\ | \end{bmatrix} \le \begin{bmatrix} | \\ b \\ | \end{bmatrix}$$

and $\begin{bmatrix} | \\ x \\ | \end{bmatrix} \ge 0.$



special case of a convex polytope

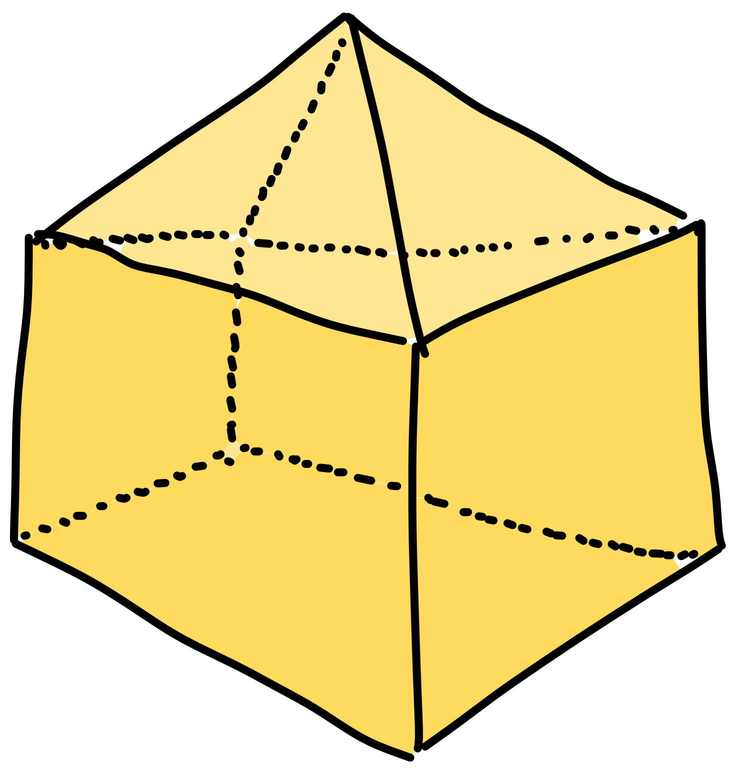
Convex polytope

- **Definition:** The following are equivalent.
  - For $a_1, \ldots, a_m \in \mathbb{R}^n$ and $b_1, \ldots, b_m \in \mathbb{R}^m$, the set of $x \in \mathbb{R}^n$ such that $a_i^\top x \leq b_i$ is a convex polytope.
  - Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, the set of $x \in \mathbb{R}^n$ such that $Ax \leq b$ is a convex polytope.
  - Given a set of points $y_1, \ldots, y_k \in \mathbb{R}^n$, the convex hull $\mathbf{conv}(y_1, \ldots, y_k)$ is a convex polytope. A convex hull $\mathbf{conv}(y_1, \ldots, y_k)$ is the intersection of all convex sets containing the points $y_1, \ldots, y_k$.

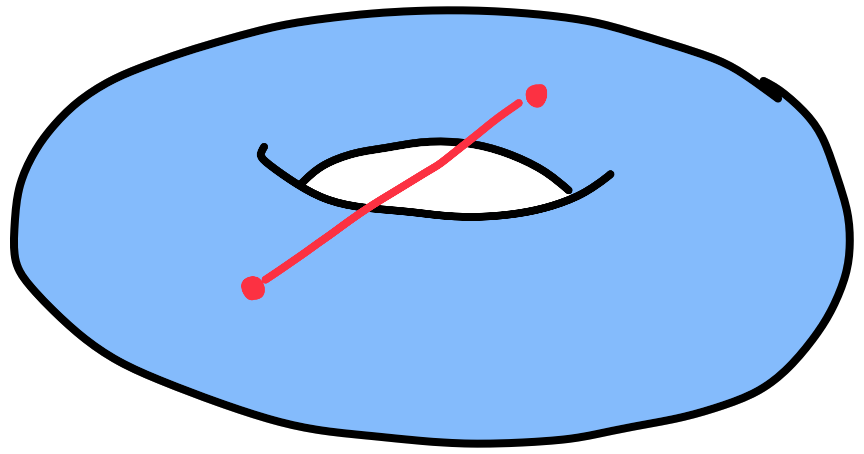
Meaning of convexity

- **Definition:** $F \subseteq \mathbb{R}^n$ is a **convex region** if for all $x, y \in F$, the line segment $\overline{xy}$ is contained in F – i.e. for $\lambda \in [0,1], \lambda x + (1-\lambda)y \in F$.

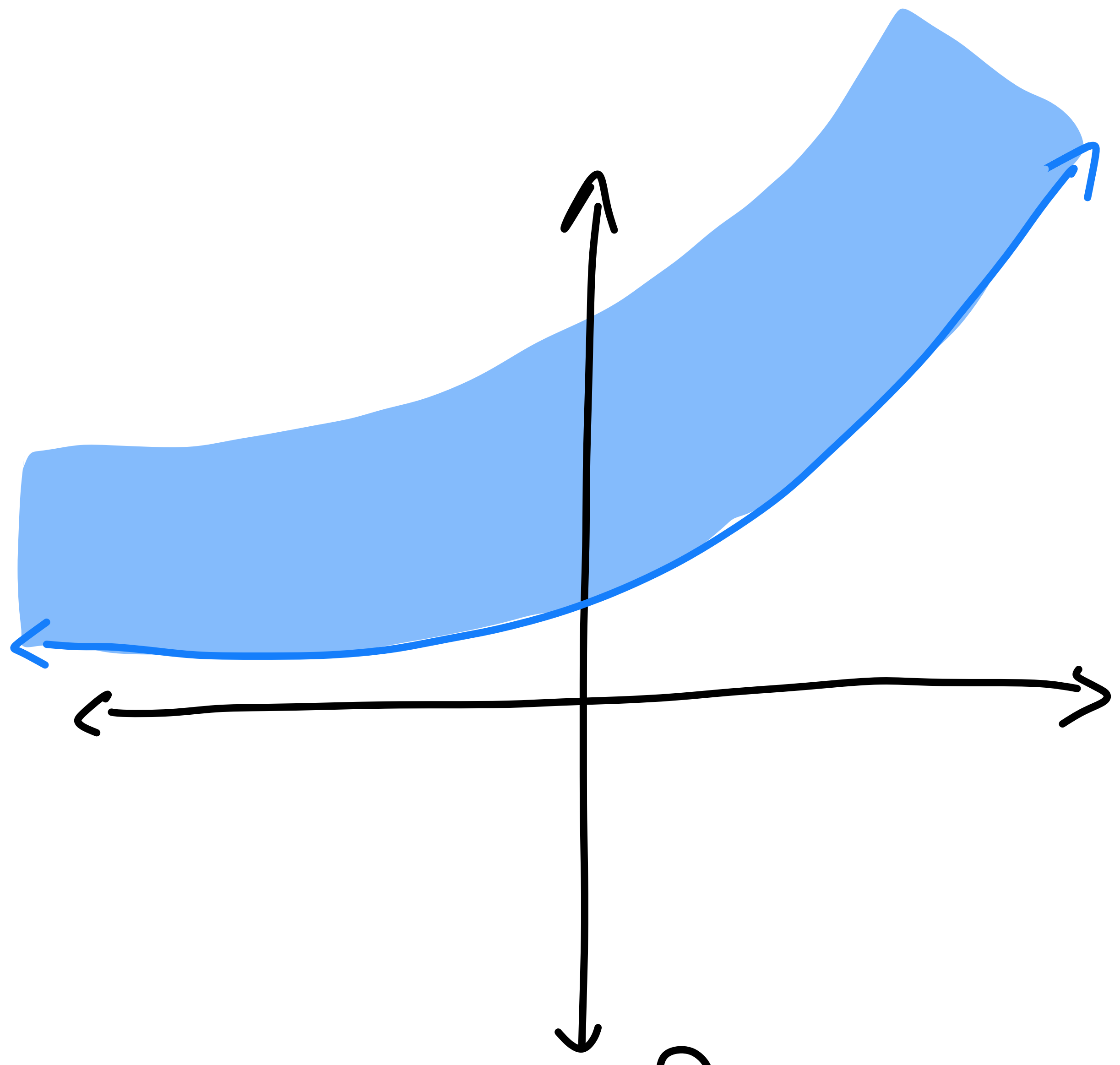
- **Definition:** A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if $\{(x,y) \in \mathbb{R}^{n+1} : y \geq f(x)\}$ is a convex region.



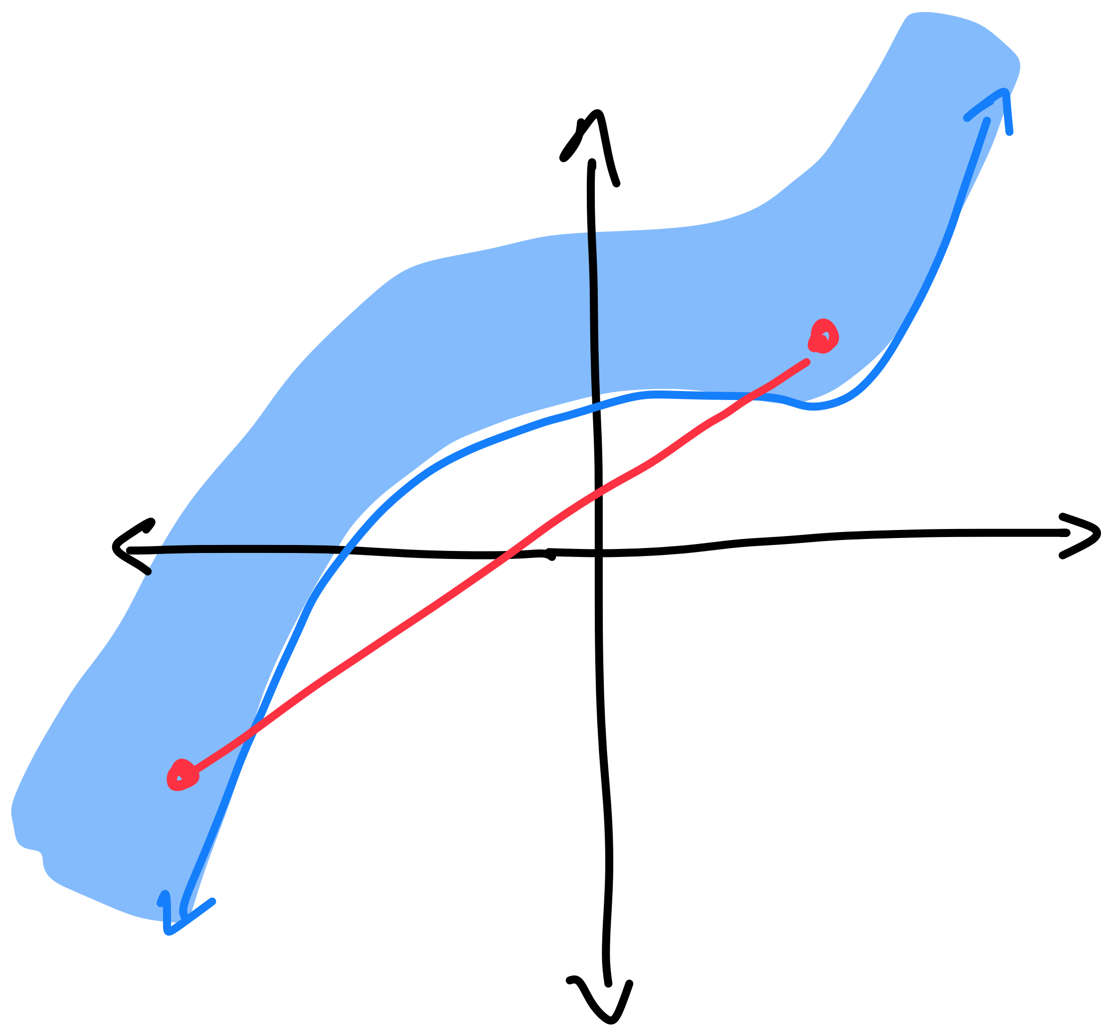
convex region



not convex



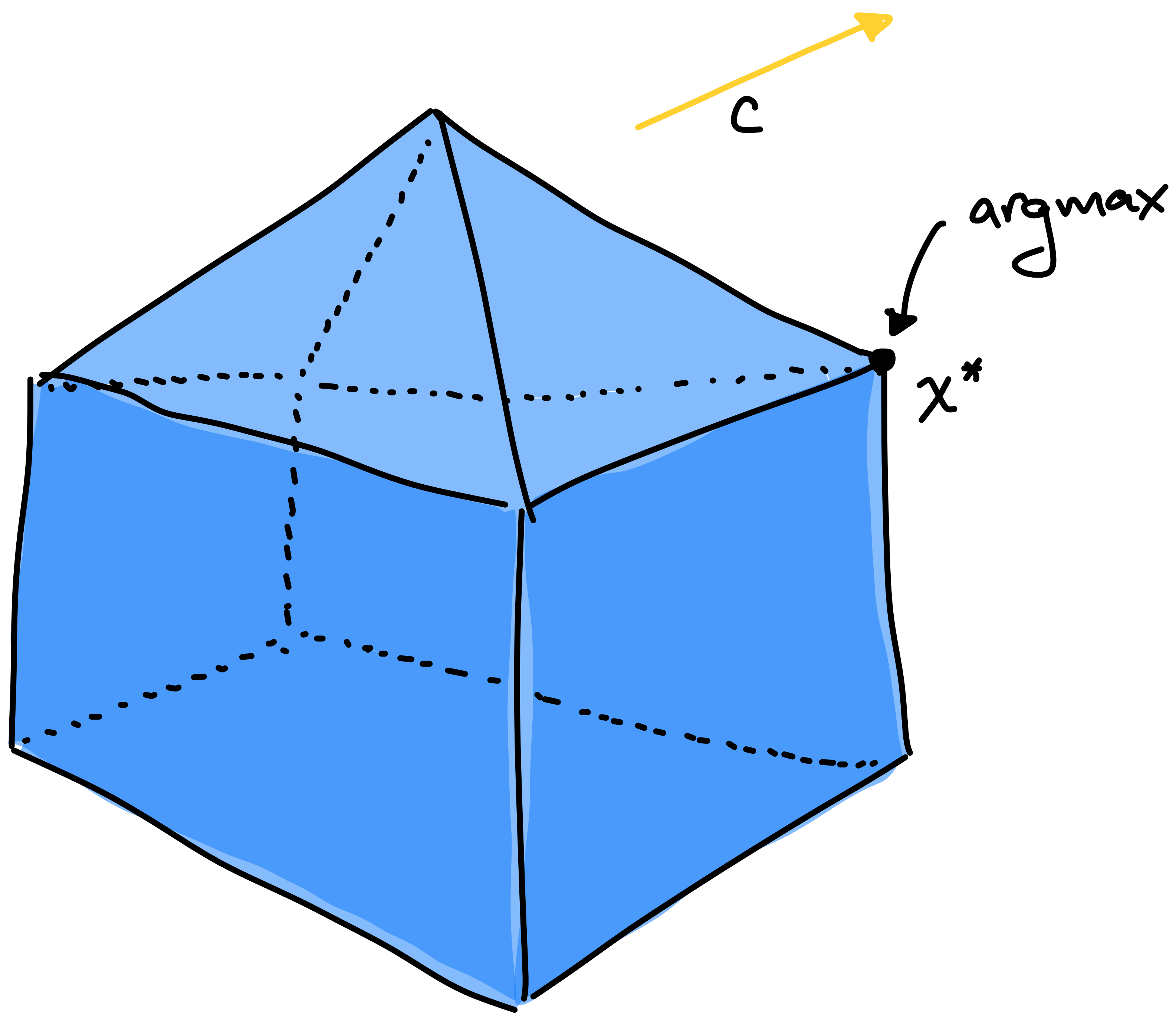
convex fn



non - convex fn

Optimizing a linear function

maximizing $c^T x$ subject to $x \in \Gamma$



Pictorally, the optimal pt will be the point $\in \Gamma$ most in the c direction.

Linear programming:

Optimizing a linear fn subject to a convex polytope.

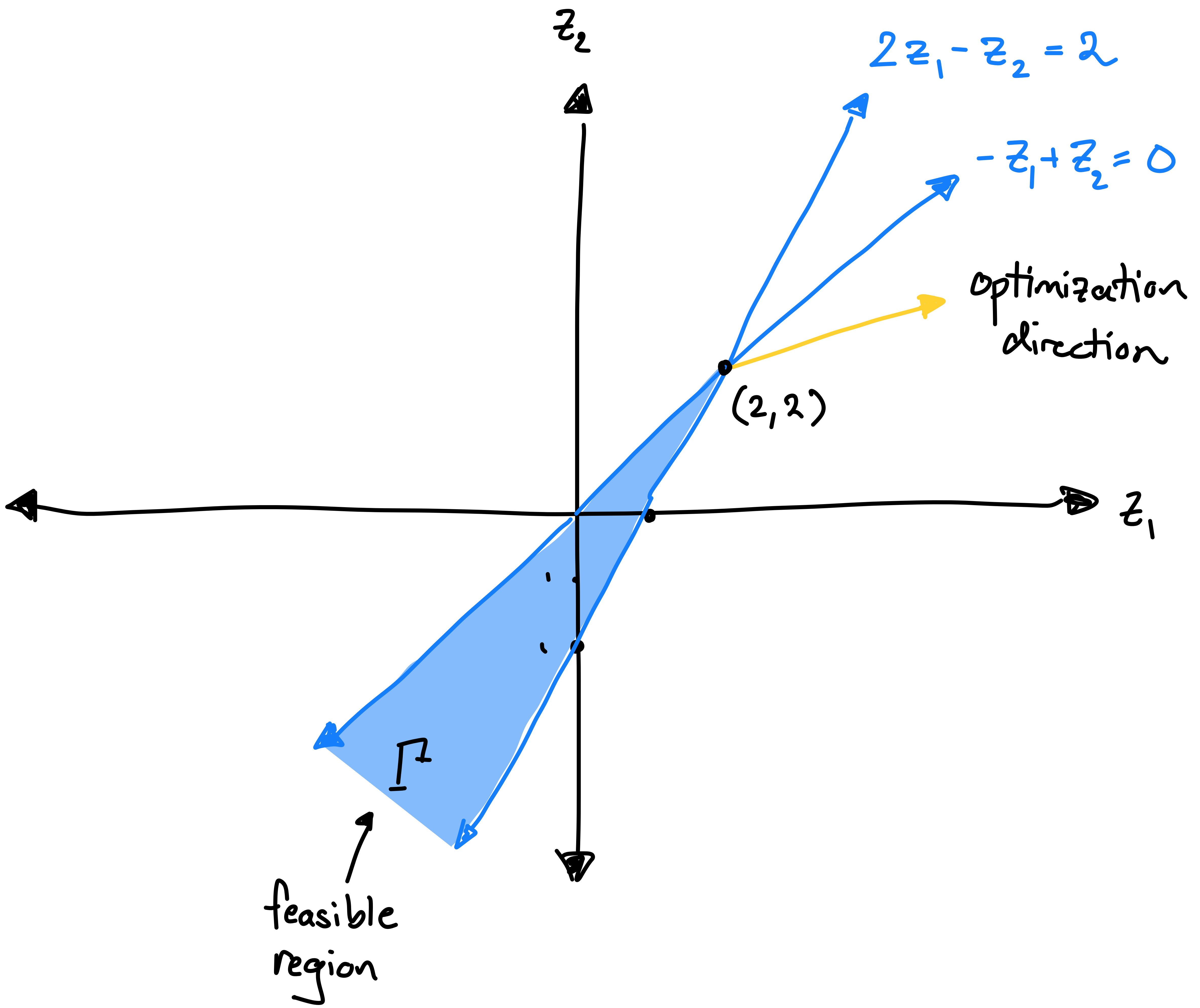
Linear programming example

General linear programming example:

$$\text{maximize} \quad 10 z_1 + z_2$$

$$\text{subject to} \quad 2z_1 - z_2 \le 2$$

$$-z_1 + z_2 \le 0$$



Linear programming standard form

Standard Form:

$$\max \quad c^T x$$

$$\text{s.t.} \quad \begin{cases} Ax \le b \\ x \ge 0 \end{cases}$$

Claim: Any optimization over convex polytope Γ is equiv. to an optimization over a LP of standard form

Proof: $\left\{ \begin{array}{ll} \max & c^T z \\ \text{s.t.} & Az \le b \end{array} \right\} = \left\{ \begin{array}{ll} \max & c_1 z_1 + c_2 z_2 + \dots + c z_n \\ \text{s.t.} & a_1^T z \le b_1, \dots, a_m^T z \le b_m \end{array} \right\}$

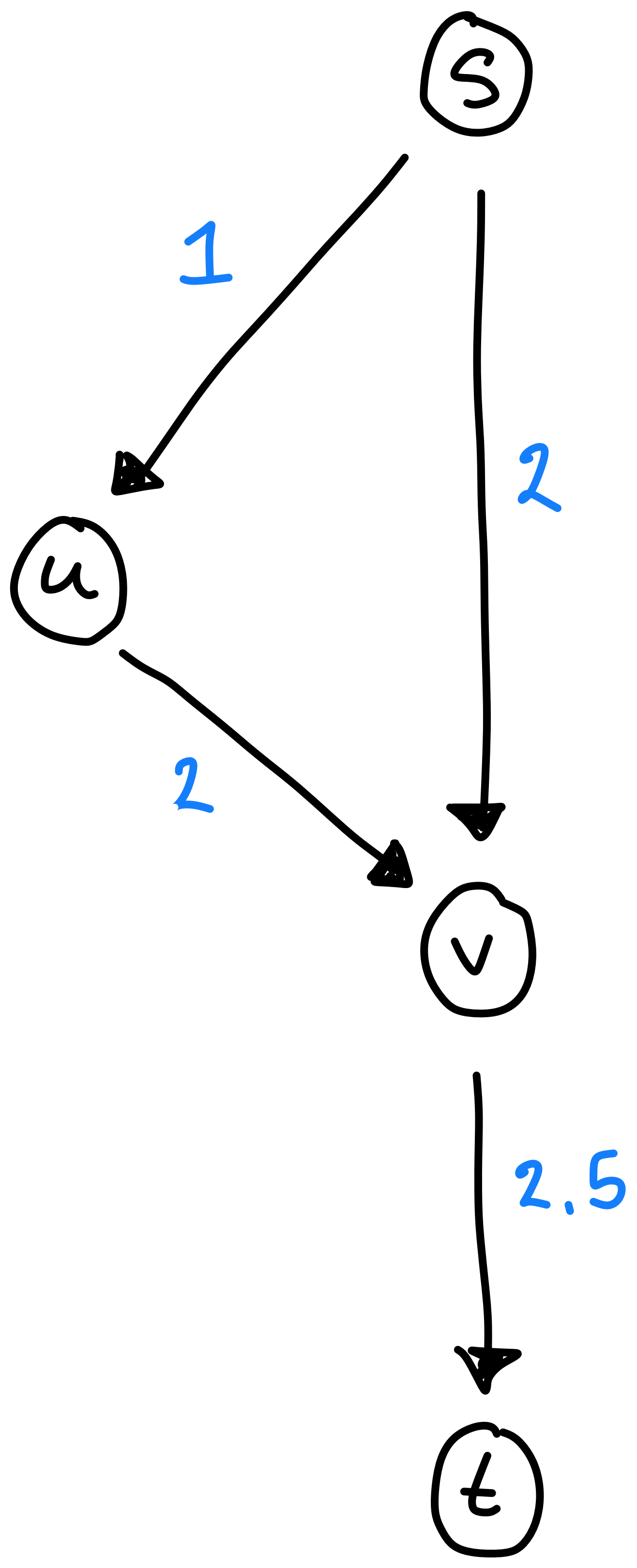
Replace z_i with $x_i^{(+)} - x_i^{(-)}$ with $x_i^{(+)}, x_i^{(-)} \ge 0$. $\underbrace{z = x^{(+)} - x^{(-)}}_{\text{vector eq.}}$

$$\left\{ \begin{array}{ll} \max & c_1(x_1^{(+)} - x_1^{(-)}) + \dots + c_n(x_n^{(+)} - x_n^{(-)}) \\ \text{s.t.} & a_1^T(x^{(+)} - x^{(-)}) \le b_1, \dots, a_m^T(x^{(+)} - x^{(-)}) \le b_m \\ & x^{(+)} \ge 0, \; x^{(-)} \ge 0 \end{array} \right\}$$

Linear programming examples

- Some we have seen
  - Max flow / min cut
  - Shortest paths
- Some we have not
  - Zero-sum games
  - Linear regression
  - Approximation algorithms for some NP-complete problems

Max flow as a linear program



max flow is equivalent to

$$\max \quad f_{su} + f_{sv}$$

s.t.

capacity constraints

$$\begin{aligned} 0 &\le f_{su} \le 1 \\ 0 &\le f_{sv} \le 2 \\ 0 &\le f_{uv} \le 2 \\ 0 &\le f_{vt} \le 2.5 \end{aligned}$$

conservation of flow

$$f_{su} - f_{uv} = 0$$

$$\begin{cases} f_{su} - f_{uv} \le 0 \\ -f_{su} + f_{uv} \le 0 \end{cases}$$

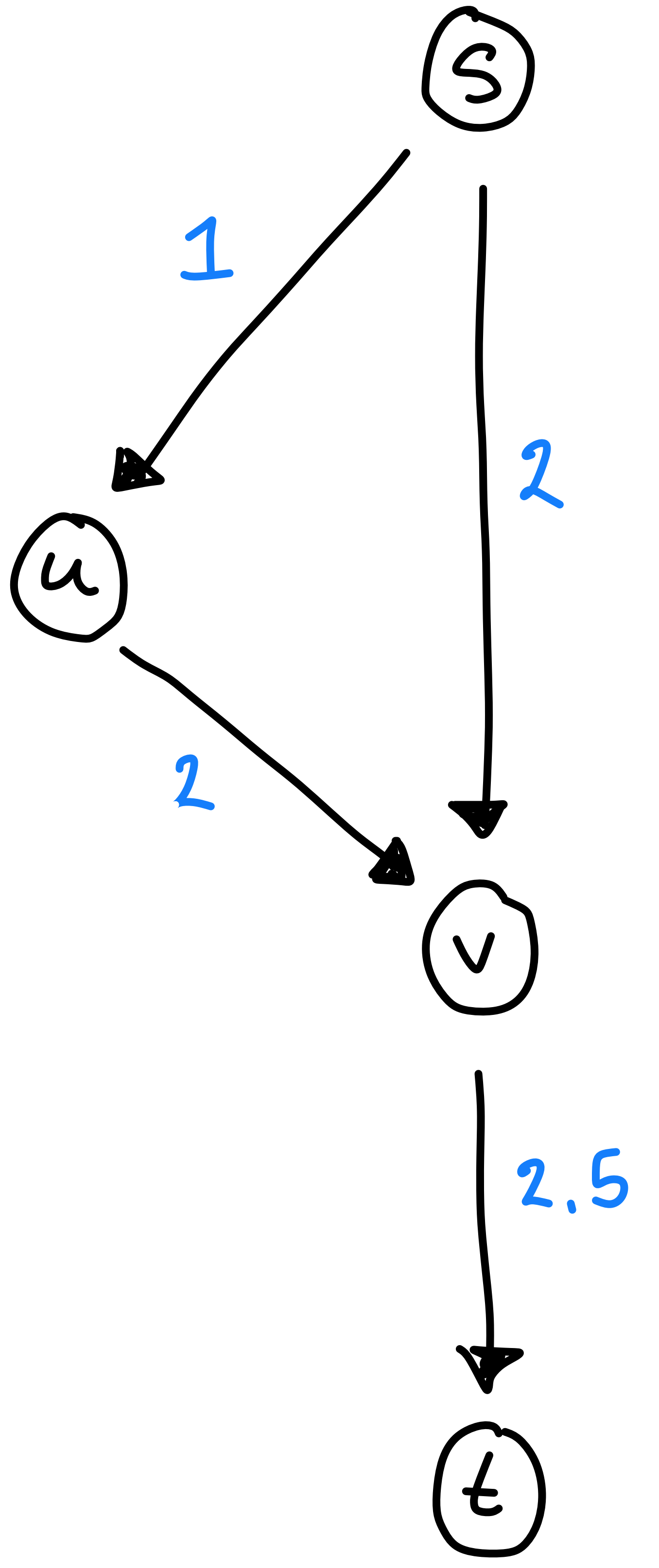
$$f_{sv} - f_{uv} - f_{vt} = 0$$

$$\begin{cases} f_{sv} - f_{uv} - f_{vt} \le 0, \\ -f_{sv} + f_{uv} + f_{vt} \le 0 \end{cases}$$

This is a linear program.

We can also express it in standard form (next slide).

Max flow as a linear program



Let $f = \begin{bmatrix} f_{su} \\ f_{sv} \\ f_{uv} \\ f_{vt} \end{bmatrix}$. Then, max flow is equivalent to:

$$\max \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \cdot f$$

$$\text{s.t.} \quad \left[\begin{array}{cccc} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ \hdashline 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 \end{array}\right] \cdot f \leq \left[\begin{array}{c} 1 \\ 2 \\ 2 \\ 2.5 \\ \hdashline 0 \\ 0 \\ 0 \\ 0 \end{array}\right],$$

$$f \geq 0.$$

Max flow as a linear program

- Let (G, c, s, t) be a flow network. Then the max flow $f \in \mathbb{R}^E$ is the vector optimizing the following LP:
  - Let $g = \mathbf{1}_{\{e \text{ out of } s\}}$
  - For each vertex $v \in V \backslash \{s, t\}$, let $h_v = +\mathbf{1}_{\{e \text{ out of } v\}} - \mathbf{1}_{\{e \text{ into } v\}}$.

max flow equals

$$\max \quad g^T f$$

$$\text{s.t.} \quad \begin{bmatrix} \mathbb{I}_E \\ \hdashline h_{v_1} \\ -h_{v_1} \\ \vdots \\ h_{v_n} \\ -h_{v_n} \end{bmatrix} \cdot f \le \begin{bmatrix} c \\ \hdashline 0 \end{bmatrix} \begin{array}{l} \} \text{ capacity constraints} \\ \} \text{ conservation of flow} \end{array},$$

$$f \ge 0.$$

Max flow as a linear program

- Max flow on a graph with $|V| = n, |E| = m$ is equivalent to a linear program over m variables and $m + 2(n-2) = O(m+n)$ constraints
- If we had a very fast algorithm for solving linear programs then it would imply a very fast algorithm for max flow.
- Second, since max flow is a special case of linear programs, the algorithms we discovered for max flow may inspire algorithms for LPs.
- We will see an algorithm for LPs in next lecture.

The value of expressing problems as LPs

- Due to the prevalence of LPs, many optimizations are known
- We know LPs can be solved in polynomial time
  - Makes writing down a problem as an LP a good first step
- Writing a problem as a linear program, can make a solution apparent
- Arguing correctness of an LP can be easier
- Applying duality (next!) can give a different perspective on the problem

Minimization linear programs

$$\left\{ \begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \le b \\ & x \ge 0 \end{array} \right\}$$

is equivalent to

$$\left\{ \begin{array}{ll} \max & (-c)^T x \\ \text{s.t.} & Ax \le b \\ & x \ge 0 \end{array} \right\}$$

Shortest paths as an LP

- **Input**: Directed graph $G = (V, E)$ and vertices s, t
- **Output**: (Length) of shortest path $s \rightsquigarrow t$
- **Claim**: The length of the shortest path is the solution to the following “flow-like” LP.
- **Proof (sketch)**:
- $(\Rightarrow)$:A path of length ℓ corresponds to a valid flow.
- $(\Leftarrow)$: A flow is the sum of $\leq m$ flows along paths. Since total flow is 1, the flow can be thought of as a probability distribution over paths. So, the LP’s feasible solution is an expectation over paths.

$$\text{minimize} \quad \mathbb{1}^T \cdot f$$

$$\text{s.t.} \quad \sum_{e \text{ out of } s} f(e) = 1,$$

$$\sum_{e \text{ into } t} f(e) = 1,$$

$$\forall v \in V \setminus \{s,t\}, \quad \sum_{e \text{ out of } v} f(e) = \sum_{e \text{ into } v} f(e),$$

$$f \geq 0$$

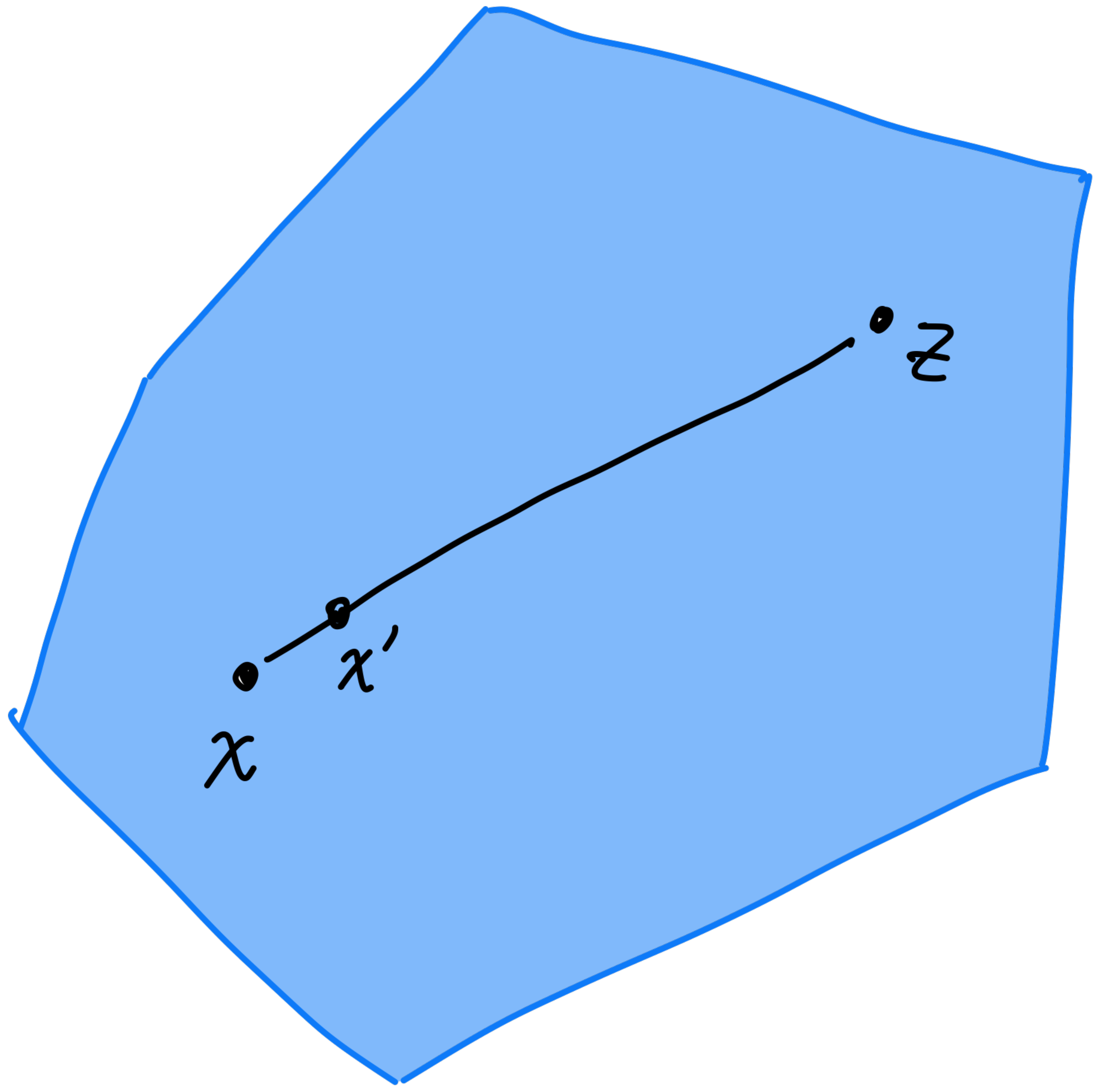
Linear programming feasibility

- Recall, the feasible region of a standard LP is $\Gamma = \{x : Ax \leq b, x \geq 0\}$.
- **Definition:** The LP is *infeasible* if $\Gamma = \varnothing$.
- **Definition:** The LP is *unbounded* if $c^\top x$ can be arbitrarily large for some $x \in \Gamma$.

- Even just deciding if a LP is feasible or not, seems like a challenging problem.

Where are the optimums of LPs

- **Theorem**: If an optimum exists for an LP, it is a global optimum.
- **Proof**: Recall we are maximizing $c^\top x$ subject to $x \in \Gamma$ and Γ is convex.
  - If $c^\top x < c^\top z$ for $x, z \in \Gamma$, then x is not a global optimum.
  - Consider the line $\overline{xz} \in \Gamma$. Then $x' := x + \epsilon(z - x) \in \Gamma$ for small $\epsilon > 0$ and
  - $c^\top x' = c^\top x + \epsilon c^\top (z - x) > c^\top x.$
  - So x is not a local optimum.
  - This proves the contrapositive.



Convex polytope

- **Definition:** A **vertex** z of a convex polytope Γ is any point such that z is not the midpoint of any line segment $\overline{xy} \in \Gamma$ for $x \neq y$.
- **Remark:** If $v_1, \ldots, v_k$ are **all** the vertices of a convex polytope Γ, then $\Gamma = \mathbf{conv}(v_1, \ldots, v_k)$, the convex hull of the vertices.
- **Theorem:** If the optimum of a standard linear program is finite, then the optimum must be achieved at some vertex.

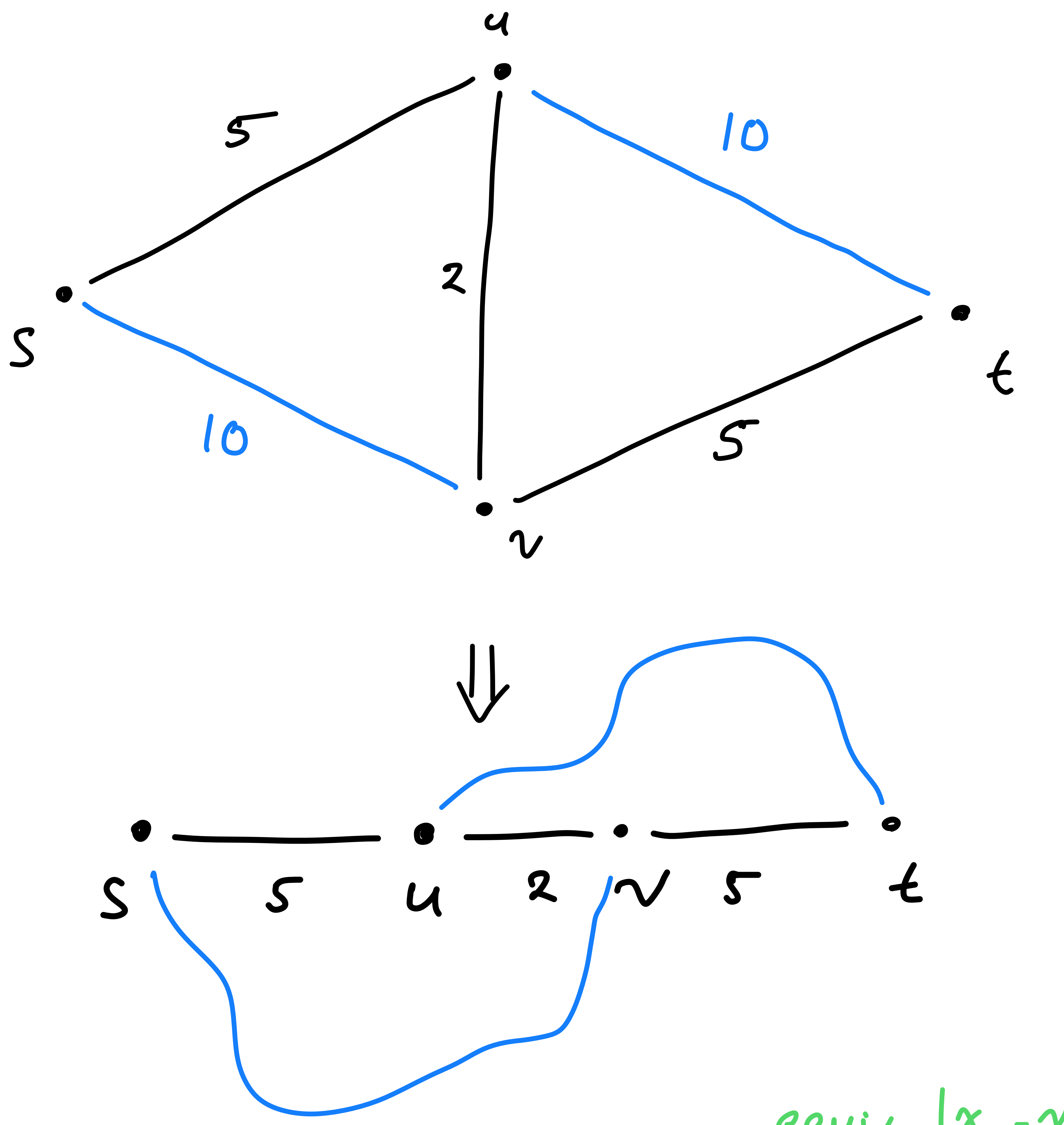
Convex polytope

- **Theorem:** If the optimum of a standard linear program is finite, then the optimum must be achieved at some vertex.
- **Proof:** Let $v_1, \ldots, v_k$ be the vertices of the feasible region Γ.
  - Then every point $x \in \Gamma$ equals $\sum_{i=1}^{k} \lambda_i v_i$ for $\lambda \geq 0$ and $\sum_{i=1}^{k} \lambda_i = 1$.
  - By linearity of objective function,
    - $c^\top x = \sum_{i=1}^{k} \lambda_i c^\top v_i \leq \max_{i=1}^{k} c^\top v_i$
    - So one of the vertices must do better than the vertex x.

The string example

Minimization as maximization

- Recall the shortest path problem from s to t
- It is easiest seen as a *minimization* problem
- Now, imagine each edge is a piece of yarn of length $w(e)$ with knots tied at the vertices
  - Pull the yarn apart at s and t till it is taut
  - The strings that are taut form the shortest path from s to t
  - And yet pulling the yarn sounds like a *maximization* problem

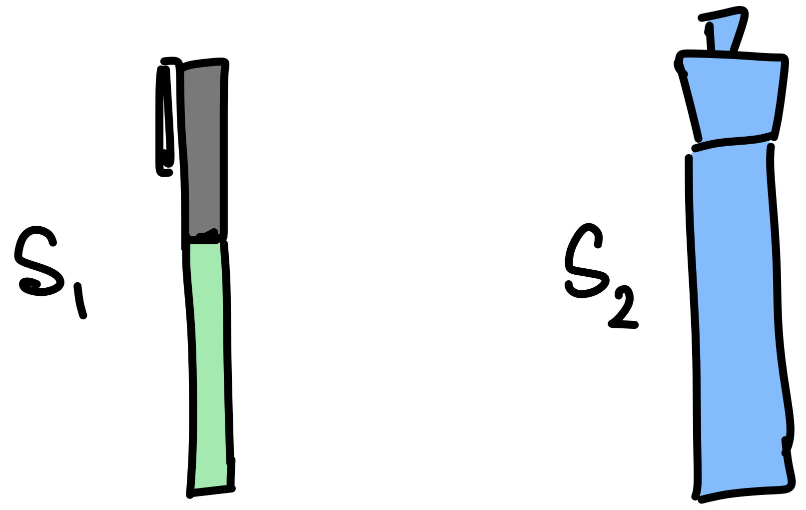


equiv. $|x_u - x_v| \le d(e)$

$$\begin{cases} \max & x_t - x_s \\ \text{s.t.} & \forall e=(u,v), \quad x_u - x_v \le d(e) \\ & \qquad\qquad\quad\;\; x_v - x_u \le d(e) \\ & x \ge 0. \end{cases}$$

Linear program duality

- Consider a salesman who sells either pens or markers.
- He sells pens for S_1 and markers for S_2.
- There are material restrictions due to labor, ink, and plastic.



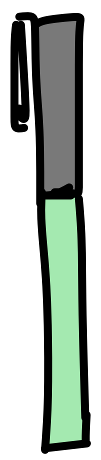
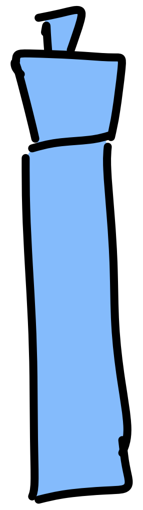
$$
\begin{aligned}
\max \quad & S_1 x_1 + S_2 x_2 \\
\text{s.t.} \quad & L_1 x_1 + L_2 x_2 \le L \\
& I_1 x_1 + I_2 x_2 \le I \\
& P_1 x_1 + P_2 x_2 \le P \\
& x_1, x_2 \ge 0.
\end{aligned}
$$

Linear programming duality

- Now let's imagine there are market prices for the 3 materials: y_L, y_I, y_P.
- It is only economical to sell a pen if $y_L L_1 + y_I I_1 + y_P I_P \geq S_1$
  - The left hand side is the cost to make a pen
  - And the right hand side is the profit
  - Similarly, sell markers only if $y_L L_2 + y_I I_2 + y_P P_2 \geq S_2$.
- Therefore, it is in the market's interest to **minimize** the total available materials while the salesman can still sell his goods. This is the **dual problem**.

Linear programming duality

$$
\begin{aligned}
\max \quad & S_1 x_1 + S_2 x_2 \\
\text{s.t.} \quad & L_1 x_1 + L_2 x_2 \le L \\
& I_1 x_1 + I_2 x_2 \le I \\
& P_1 x_1 + P_2 x_2 \le P \\
& x_1, x_2 \ge 0.
\end{aligned}
$$

S_1  S_2 

$$
\begin{aligned}
\min \quad & y_L L + y_I I + y_P P \\
\text{s.t.} \quad & y_L L_1 + y_I I_1 + y_P P_1 \ge S_1 \\
& y_L L_2 + y_I I_2 + y_P P_2 \ge S_2 \\
& y_L, y_I, y_P \ge 0.
\end{aligned}
$$

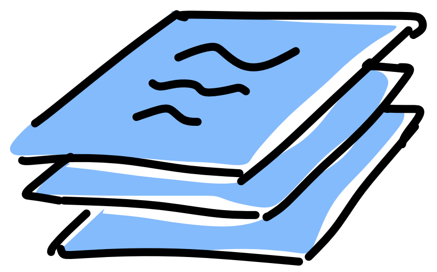
y_L

labor

y_I

ink

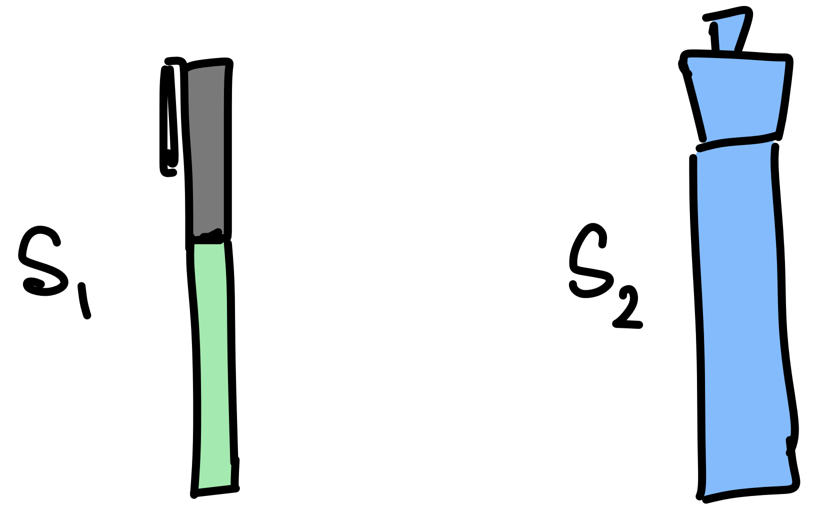
y_P



plastic

Linear programming duality

$$\begin{aligned} \max \quad & S_1 x_1 + S_2 x_2 \\ \text{s.t.} \quad & L_1 x_1 + L_2 x_2 \le L \\ & I_1 x_1 + I_2 x_2 \le I \\ & P_1 x_1 + P_2 x_2 \le P \\ & x_1, x_2 \ge 0. \end{aligned}$$



$$\begin{aligned} \min \quad & y_L L + y_I I + y_P P \\ \text{s.t.} \quad & y_L L_1 + y_I I_1 + y_P P_1 \ge S_1 \\ & y_L L_2 + y_I I_2 + y_P P_2 \ge S_2 \\ & y_L, y_I, y_P \ge 0. \end{aligned}$$

Linear programming duality

$$\max \quad S_1 x_1 + S_2 x_2$$

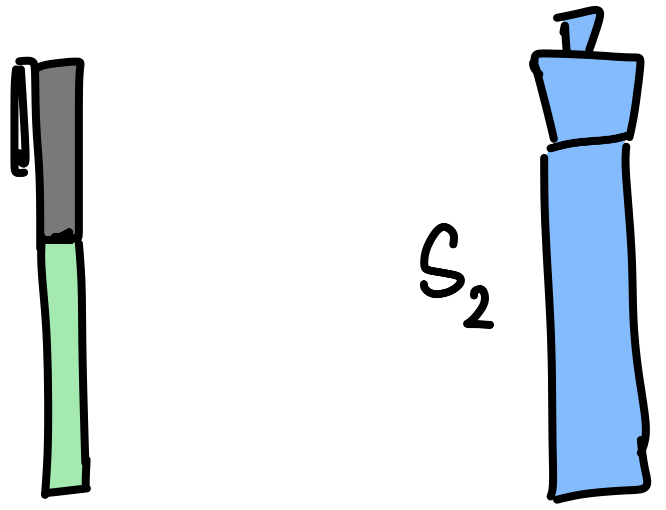
$$\text{s.t.} \quad L_1 x_1 + L_2 x_2 \le L$$

$$I_1 x_1 + I_2 x_2 \le I$$

$$P_1 x_1 + P_2 x_2 \le P$$

$$x_1, x_2 \ge 0.$$

S_1 S_2



$$\min \quad y_L L + y_I I + y_P P$$

$$\text{s.t.} \quad y_L L_1 + y_I I_1 + y_P P_1 \ge S_1$$

$$y_L L_2 + y_I I_2 + y_P P_2 \ge S_2$$

$$y_L, y_I, y_P \ge 0.$$

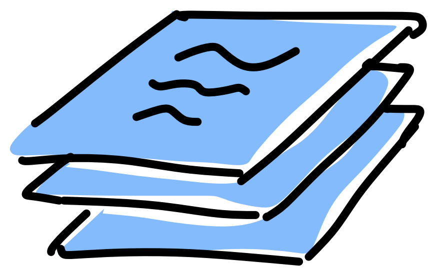
y_L

labor

y_I

ink

y_P

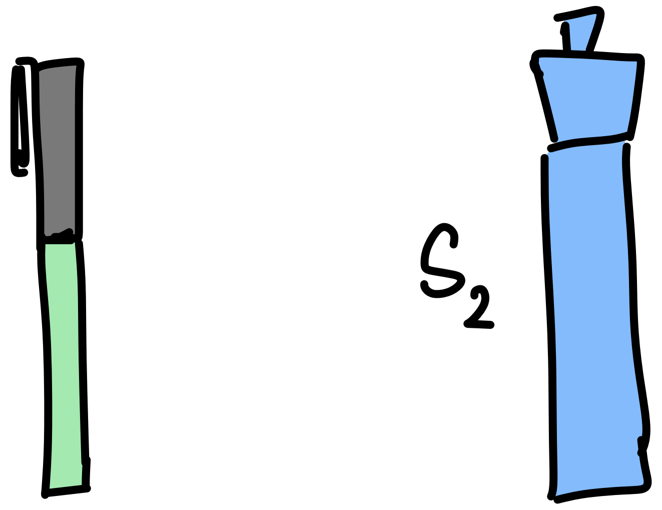


plastic

Linear programming duality

$$
\begin{aligned}
\max \quad & S_1 x_1 + S_2 x_2 \\
\text{s.t.} \quad & L_1 x_1 + L_2 x_2 \le L \\
& I_1 x_1 + I_2 x_2 \le I \\
& P_1 x_1 + P_2 x_2 \le P \\
& x_1, x_2 \ge 0.
\end{aligned}
$$

S_1 S_2



$$
\begin{aligned}
\min \quad & y_L L + y_I I + y_P P \\
\text{s.t.} \quad & y_L L_1 + y_I I_1 + y_P P_1 \ge S_1 \\
& y_L L_2 + y_I I_2 + y_P P_2 \ge S_2 \\
& y_L, y_I, y_P \ge 0.
\end{aligned}
$$

Linear programming duality

Primal linear program (P)

$$\begin{aligned} \max \quad & c^T x \quad \leftarrow \in \mathbb{R}^n \\ \text{s.t.} \quad & Ax \le b \\ & x \ge 0 \end{aligned}$$

Dual linear program (D)

$$\begin{aligned} \min \quad & b^T y \quad \leftarrow \in \mathbb{R}^m \\ \text{s.t.} \quad & A^T y \ge c \\ & y \ge 0 \end{aligned}$$

Linear programming duality

(Weak duality)

- **Theorem**:
  - If $x \in \mathbb{R}^n$ is feasible for $(\mathscr{P})$ and $y \in \mathbb{R}^m$ is feasible for $(\mathscr{D})$, then $c^\top x \leq y^\top Ax \leq b^\top y$.
  - If $(\mathscr{P})$ is unbounded, then $(\mathscr{D})$ is infeasible.
  - If $(\mathscr{D})$ is unbounded, then $(\mathscr{P})$ is infeasible.
  - If $c^\top x = b^\top y$ for $x \in \mathbb{R}^n$ is feasible for $(\mathscr{P})$ and $y \in \mathbb{R}^m$ is feasible for $(\mathscr{D})$, then x is an optimal solution for $(\mathscr{P})$ and y is an optimal solution for $(\mathscr{D})$.

Proving weak duality

- Let's prove when both LPs are feasible, that $c^\top x \le y^\top Ax \le b^\top y$.

Since x is feasible for (P),

$$Ax \le b, \quad x \ge 0. \quad (1)$$

Since y is feasible for (D),

$$A^T y \ge c, \quad y \ge 0. \quad (2)$$

Then, $y^T(Ax) \le y^T b$ by (1).

$$= b^T y$$

And, $c^T x \le (A^T y)^T x$

$$= (y^T A) x$$

$$= y^T A x.$$

Proving weak duality

- If $(\mathscr{P})$ is unbounded
    - Then for all $N \in \mathbb{N}$, there exists $x \in \Gamma$ such that $N < c^\top x$
- If $(\mathscr{D})$ is feasible,
    - then for any feasible y, $c^\top x \leq y^\top Ax \leq b^\top y$.
- Together, this proves that $b^\top y$ is not finite, a contradiction.
- Therefore, if $(\mathscr{P})$ is unbounded, then $(\mathscr{D})$ is infeasible.
- Similarly, if $(\mathscr{D})$ is unbounded, then $(\mathscr{P})$ is infeasible.

Proving weak duality

- Lastly, since $c^\top x = b^\top y$ for some feasible x and feasible y,
- Assume for contradiction, there exists x' s.t. $c^\top x' > c^\top x = b^\top y$.
  - Then, $c^\top x' \le y^\top A x' \le y^\top b$ by first argument in weak quality.
  - This is a contradiction, proving no x' exists. So x is optimal.
- Similar argument proves that y is also optimal.

Max flow/min cut is an example of duality

- We have actually seen this duality before!
- We saw that for any flow f and any s-t cut (S, T), that $v(f) \leq c(S, T)$.
- Max flow is an example of an LP.
    - And min cut is its dual LP.
    - We will formalize this on the next slide.
- Recall, our algorithm for min cut was to first solve max flow and then look at which edges are saturated with flow.

Max flow as a linear program

- Let (G, c, s, t) be a flow network. Then the max flow $f \in \mathbb{R}^E$ is the vector optimizing the following LP:
  - Let $g = \mathbf{1}_{\{e \text{ out of } s\}}$
  - For each vertex $v \in V \backslash \{s, t\}$, let $h_v = +\mathbf{1}_{\{e \text{ out of } v\}} - \mathbf{1}_{\{e \text{ into } v\}}$.

max flow equals

$$\max \quad g^T f$$

$$\text{s.t.} \quad \begin{bmatrix} \mathbb{I}_E \\ \hdashline h_{v_1} \\ -h_{v_1} \\ \vdots \\ h_{v_n} \\ -h_{v_n} \end{bmatrix} \cdot f \le \begin{bmatrix} c \\ \hdashline 0 \end{bmatrix},$$

capacity constraints

conservation of flow

$$f \ge 0.$$

An observation about duality

- If the primal $(\mathcal{P})$ is an optimization with n variables and m equations,
  - then the dual $(\mathcal{D})$ is an optimization with m variables and n equations
- **Lesson:** If we are interested in computing the dual of an LP, its often easier to first find an equivalent LP that has few equations (even at the cost of many variables)
- **Lesson:** The m equations of the primal $(\mathcal{P})$ correspond to the m equations of the dual $(\mathcal{D})$. We should see this resemblance.
-

Min cut LP

- The trouble is that our max flow LP has m variables and $m + 2n - 2$ equations
    - This will yield an “unnatural” LP for min cut with $m + 2n - 2$ equations
    - It will be hard to see that this LP is equivalent to the min cut problem

$$
(P) = \begin{cases} \max & g^T f \\ \text{s.t.} & \begin{bmatrix} \mathbb{I}_E \\ \hline h_{v_1} \\ -h_{v_1} \\ \vdots \\ h_{v_n} \\ -h_{v_n} \end{bmatrix} \cdot f \le \begin{bmatrix} | \\ c \\ | \\ \hline | \\ 0 \\ | \end{bmatrix}, \\ & f \ge 0. \end{cases}
\qquad
(D) = \begin{cases} \min & \begin{bmatrix} - c - & | & 0 \cdots 0 \end{bmatrix} \cdot y \\ \text{s.t.} & \left[\begin{array}{c:ccccc} \mathbb{I}_E & h_{v_1} & -h_{v_1} & \cdots & h_{v_n} & -h_{v_n} \end{array}\right] \cdot y \le \begin{bmatrix} | \\ g \\ | \end{bmatrix} \\ & y \ge 0. \end{cases}
$$

A different LP for max flow

- Let's come up with a different LP for max flow
- Let P be the set of paths $s \rightsquigarrow t$
    - $|P|$ could exponential in the number of vertices
- The new LP $(\mathscr{P}')$ will have $|P|$ variables and m equations
- Therefore, its dual $(\mathscr{D}')$ will have m variables and $|P|$ equations
- We will see that max flow $= (\mathscr{P}) = (\mathscr{P}') = (\mathscr{D}') =$ min cut

For each path $p: s \rightsquigarrow t$, let x_p be the variable representing how much flow is to be sent along p.

For any edge e, capacity constraints give

$$\sum_{p: e \in p} x_e \le c(e).$$

Since every path already respects conservation of flow, we don't need constraints corresponding to them.

$$\text{Total flow} = \sum_{p \in P} x_e.$$

A different LP for max flow

- Let's come up with a different LP for max flow
- Let P be the set of paths $s \rightsquigarrow t$
  - $|P|$ could exponential in the number of vertices
- The new LP $(\mathscr{P}')$ will have $|P|$ variables and m equations
- Therefore, its dual $(\mathscr{D}')$ will have m variables and $|P|$ equations
- We will see that max flow $= (\mathscr{P}) = (\mathscr{P}') = (\mathscr{D}') =$ min cut

$$(P') = \begin{cases} \max & \mathbb{1}^T \cdot x \\ \text{s.t.} & \sum_{p : e \in p} x_p \le c(e) \quad \forall e \in E, \\ & x \ge 0 \end{cases}$$

$$(D') = \begin{cases} \min & c^T y \\ \text{s.t.} & \sum_{e : e \in p} y_e \ge 1 \quad \forall p \in P, \\ & y \ge 0. \end{cases}$$

A different LP for max flow

- We need to show that min cut = $(\mathscr{D}')$.
- **(Proof Sketch):**
  - If we have an s-t cut (S, T), consider letting y be the indicator vector for the edges crossing the cut
  - Therefore, $(\mathscr{D}') \leq$ min cut
  - Conversely, a y minimizing $(\mathscr{D}')$, can be seen as an expectation over min cuts.
  - Therefore, $(\mathscr{D}') \geq$ min cut.

$$(P') = \begin{cases} \max & \mathbf{1}^T \cdot x \\ \text{s.t.} & \sum_{p : e \in p} x_p \leq c(e) \quad \forall e \in E, \\ & x \geq 0 \end{cases}$$

$$(D') = \begin{cases} \min & c^T y \\ \text{s.t.} & \sum_{e : e \in p} y_e \geq 1 \quad \forall p \in P, \\ & y \geq 0. \end{cases}$$

Lessons from duality

- We reproved the max flow/min cut duality from the flow unit of this course
- **Observation:** Min cut does not have an intuitive poly-sized LP
    - However, it does have a m variable and $|P|$ equations sized LP
    - Therefore, its has a dual (max flow) with $|P|$ variables and m equations
    - Max flow also has a simple poly-sized LP and an efficient algorithm
- Intuitively, this is why we solve min cut by solving max flow and looking at saturated edges. It's sometimes algorithmically easier to solve a problem over its dual.

Theorems worth knowing

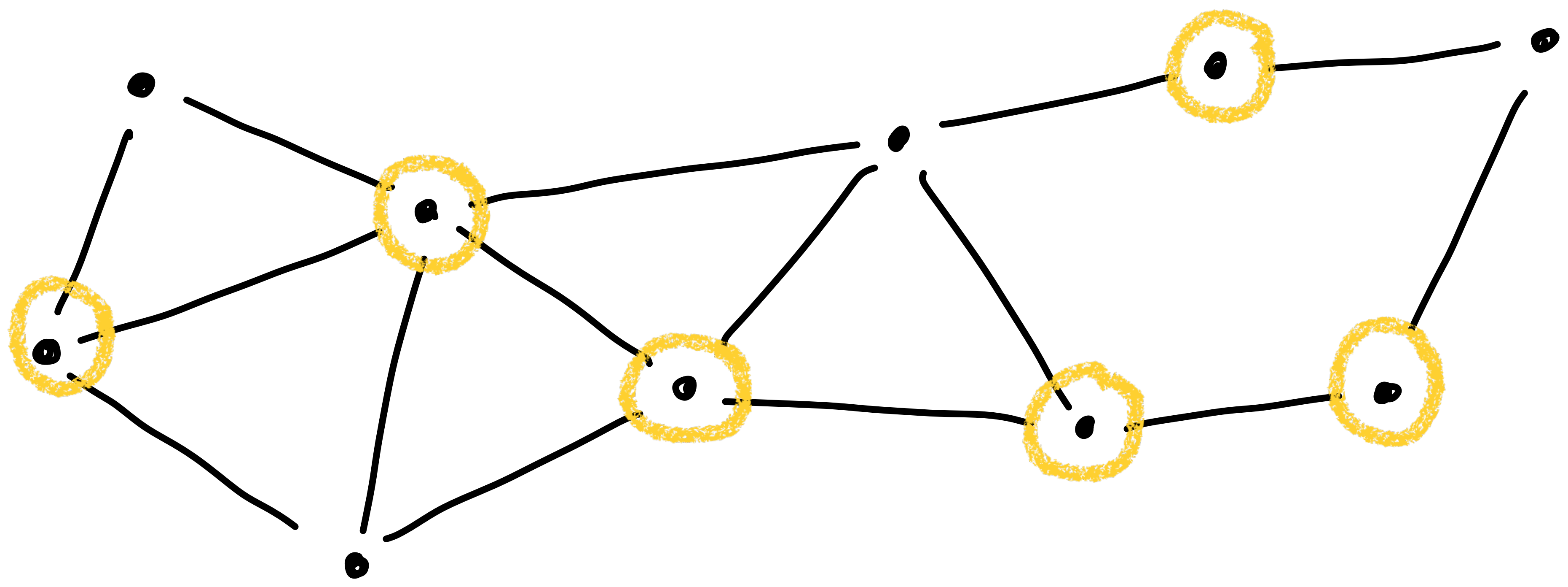
- **Weak duality theorem**
- **Theorem:** The dual of a dual is the original primal.
  - Proof is an exercise.
- **Theorem:** LPs of n variables and m equations can be solved in $\mathrm{poly}(n, m)$ time.
  - We will not prove this in this course. Algorithm is quite complex. We will, however, discuss algorithms for LPs.

What's a problem LPs can't solve?

Vertex cover

- **Input**: an undirected graph $G = (V, E)$
- **Output**: a *minimal* set $S \subseteq V$ such that every edge contains at least one endpoint from S.
- There seems to be a pretty obvious LP for this problem. What goes wrong?

There is nothing ensuring that the optimal solution x will be integer.



One variable x_v for every vertex v.

$$
\begin{cases}
\min \sum_{v \in V} x_v \\
\text{s.t.} \quad x_v \leq 1 \quad \forall v \in V \\
\quad x_u + x_v \geq 1 \quad \forall e = (u,v) \in E \\
\quad x \geq 0
\end{cases}
$$

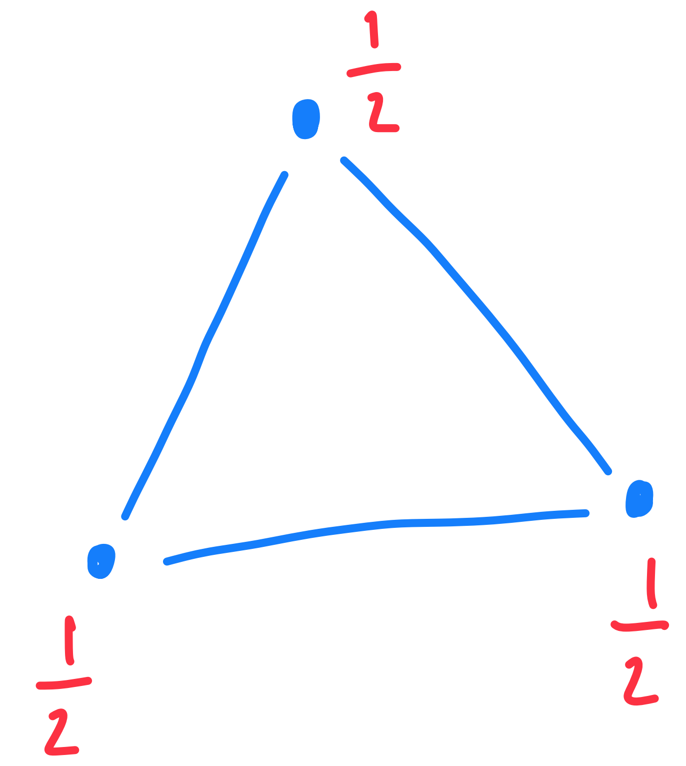
What's a problem LPs can't solve?

Vertex cover

- **Input**: an undirected graph $G = (V, E)$
- **Output**: a *minimal* set $S \subseteq V$ such that every edge contains at least one endpoint from S.
- There seems to be a pretty obvious LP for this problem. What goes wrong?

There is nothing ensuring that the optimal solution x will be integer.

Ex.



LP solution is $\frac{1}{2}$ on each vertex.

(a) LP min is $\frac{3}{2}$

(b) optimal sol has value 2.

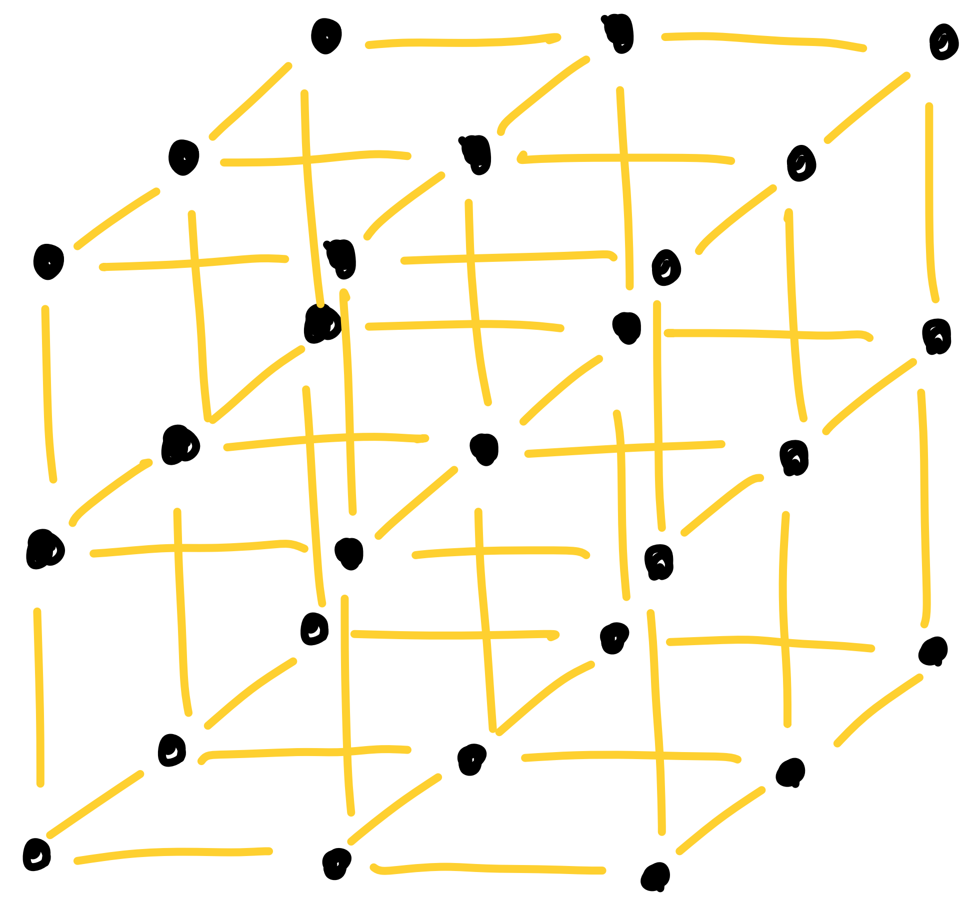
One variable x_v for every vertex v.

$$\begin{cases} \min \displaystyle\sum_{v \in V} x_v \\ \text{s.t.} \quad x_v \le 1 \quad \forall v \in V \\ \qquad x_u + x_v \ge 1 \quad \forall e = (u,v) \in E \\ \qquad x \ge 0 \end{cases}$$

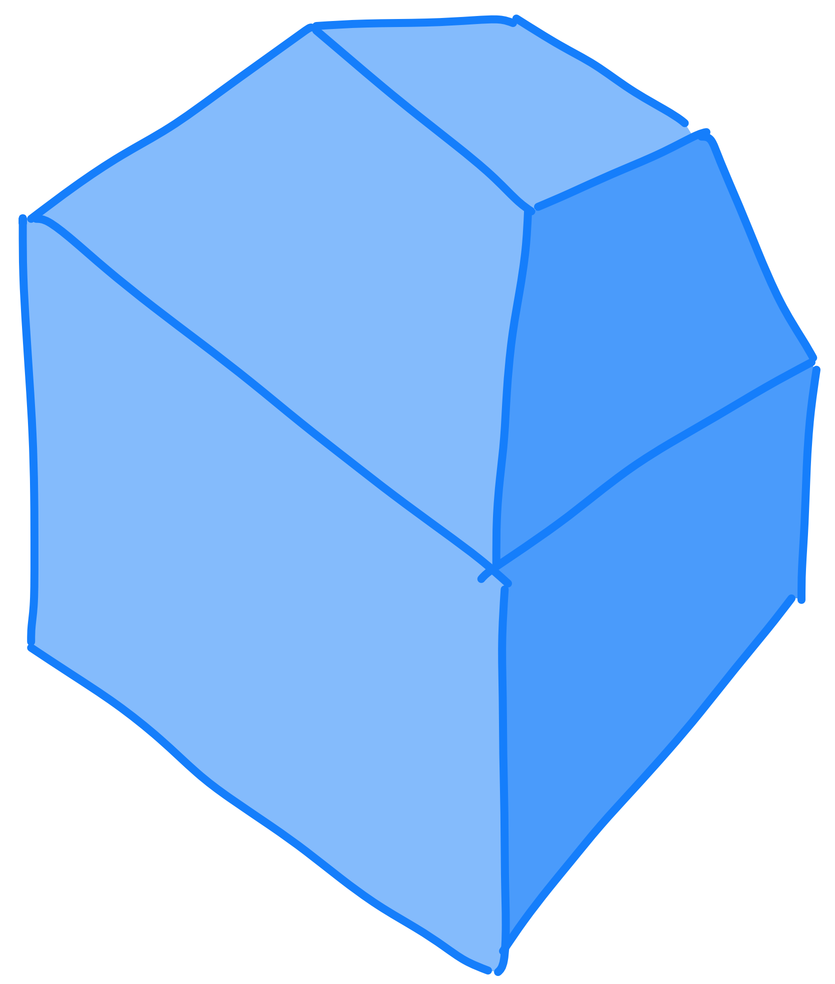
LP relaxation

Vertex cover

- The LP we tried to write for vertex cover yields a fractional solution
- It is a "relaxation" of the vertex cover problem from integer to fractional solutions
    - In the relaxation we increase the feasible space from integer coordinates to include all solutions
    - Can be used to generate randomized approximation algorithms for vertex cover.



integer coordinates



linear equations defining the vertex cover

Max flow versus vertex cover

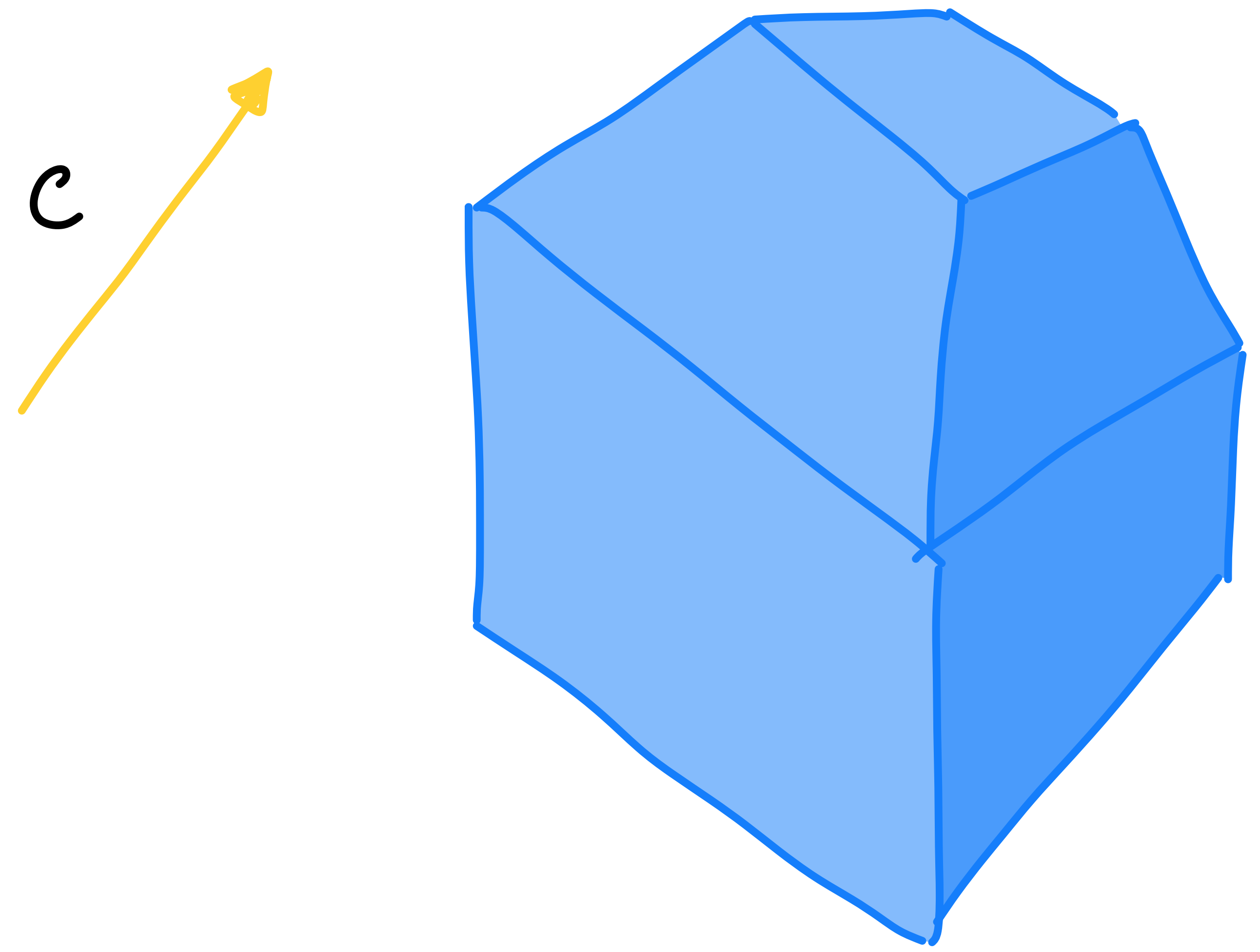
- Why can max flow natively guarantee integer solutions while vertex cover cannot?
- Recall, the optimum of an LP occurs at a vertex
  - The vertices of an LP correspond to points where linear equations are exactly satisfied
  - Turns out flow equations when exactly satisfied always have integer solutions
    - Quite a beautiful piece of mathematics
    - Too technical to warrant more time in this course

The simplex method

- Finally, we are going to cover an algorithm for solving LPs
- The algorithm is called **the simplex method** and it will be unique amongst the algorithms we study in this course
    - The simplex method runs incredibly fast in practice and is super useful
    - However, it can run in exponential time in the worst case even when there exist other polynomial time algorithms for the problem
- Later on, we will take a high-level glance at algorithms for solving LPs that are known to run in polynomial time

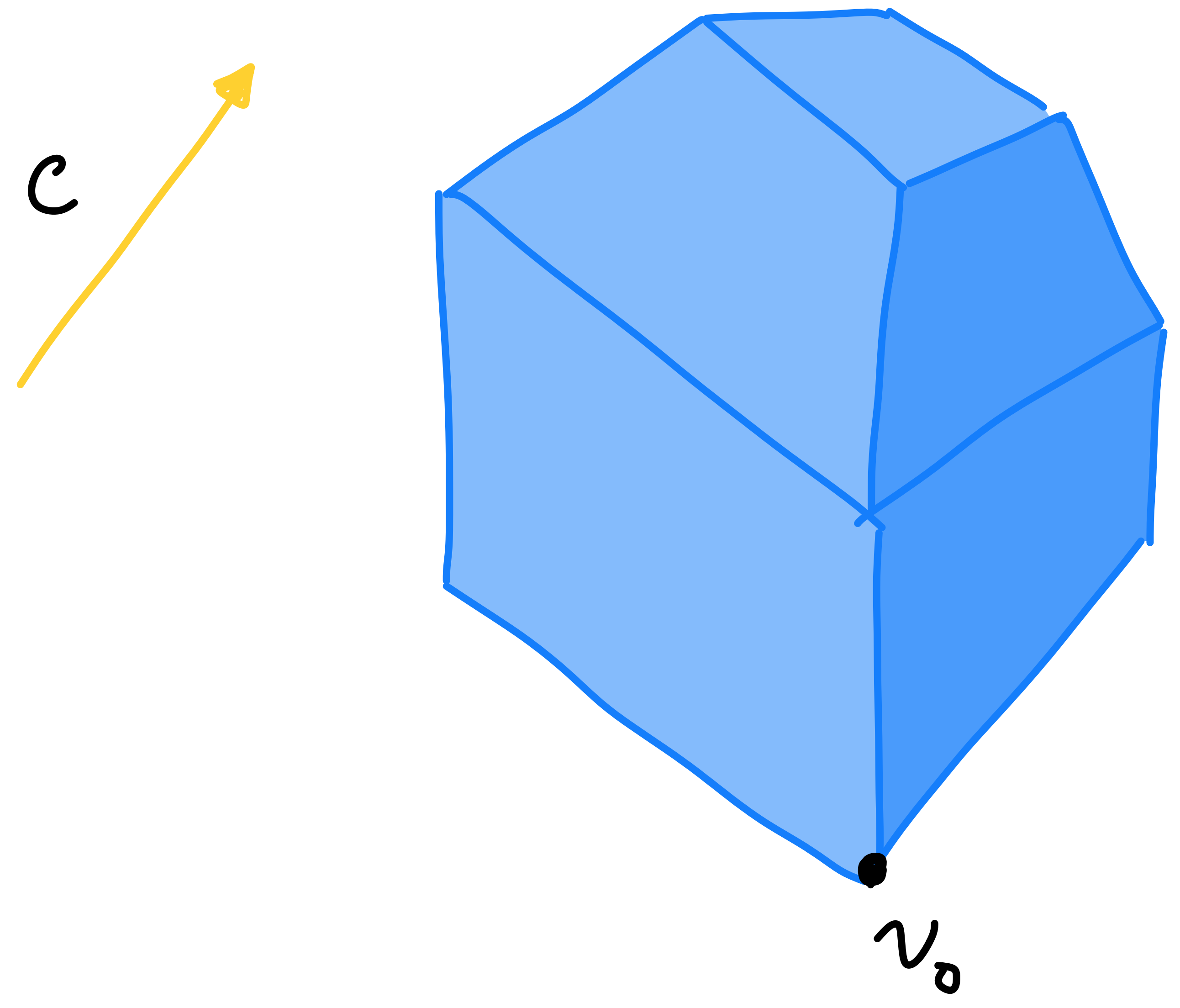
The simplex method

- Simplex is a greedy algorithm
- **High-level algorithm**:
  - Start from a vertex of the polytope
  - In each step, move to the neighboring vertex that optimizes $c^\top x$
  - Equivalently, move along the edge pointing the most in the c direction



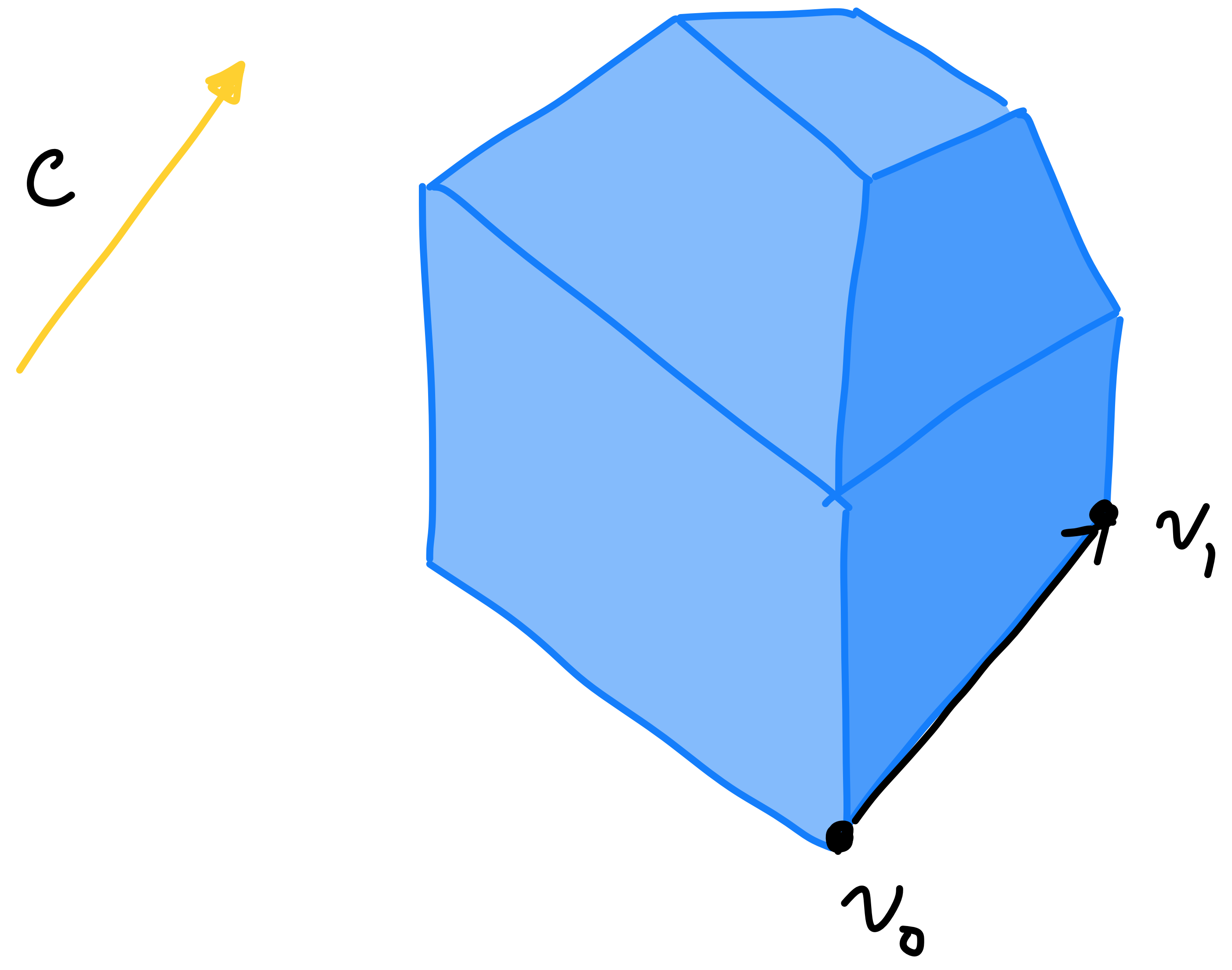
The simplex method

- Simplex is a greedy algorithm
- **High-level algorithm**:
    - Start from a vertex of the polytope
    - In each step, move to the neighboring vertex that optimizes $c^\top x$
    - Equivalently, move along the edge pointing the most in the c direction



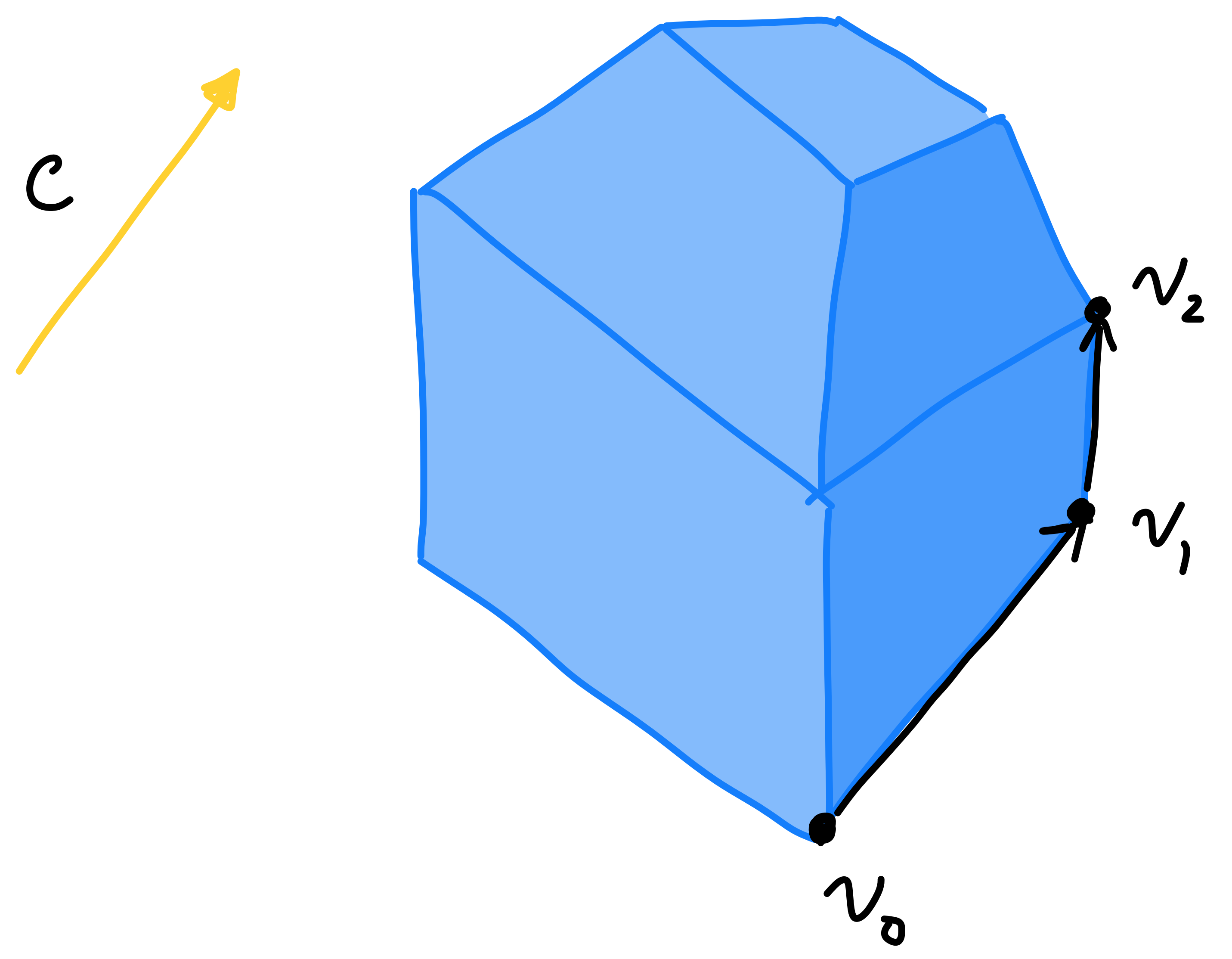
The simplex method

- Simplex is a greedy algorithm
- **High-level algorithm**:
  - Start from a vertex of the polytope
  - In each step, move to the neighboring vertex that optimizes $c^\top x$
  - Equivalently, move along the edge pointing the most in the c direction



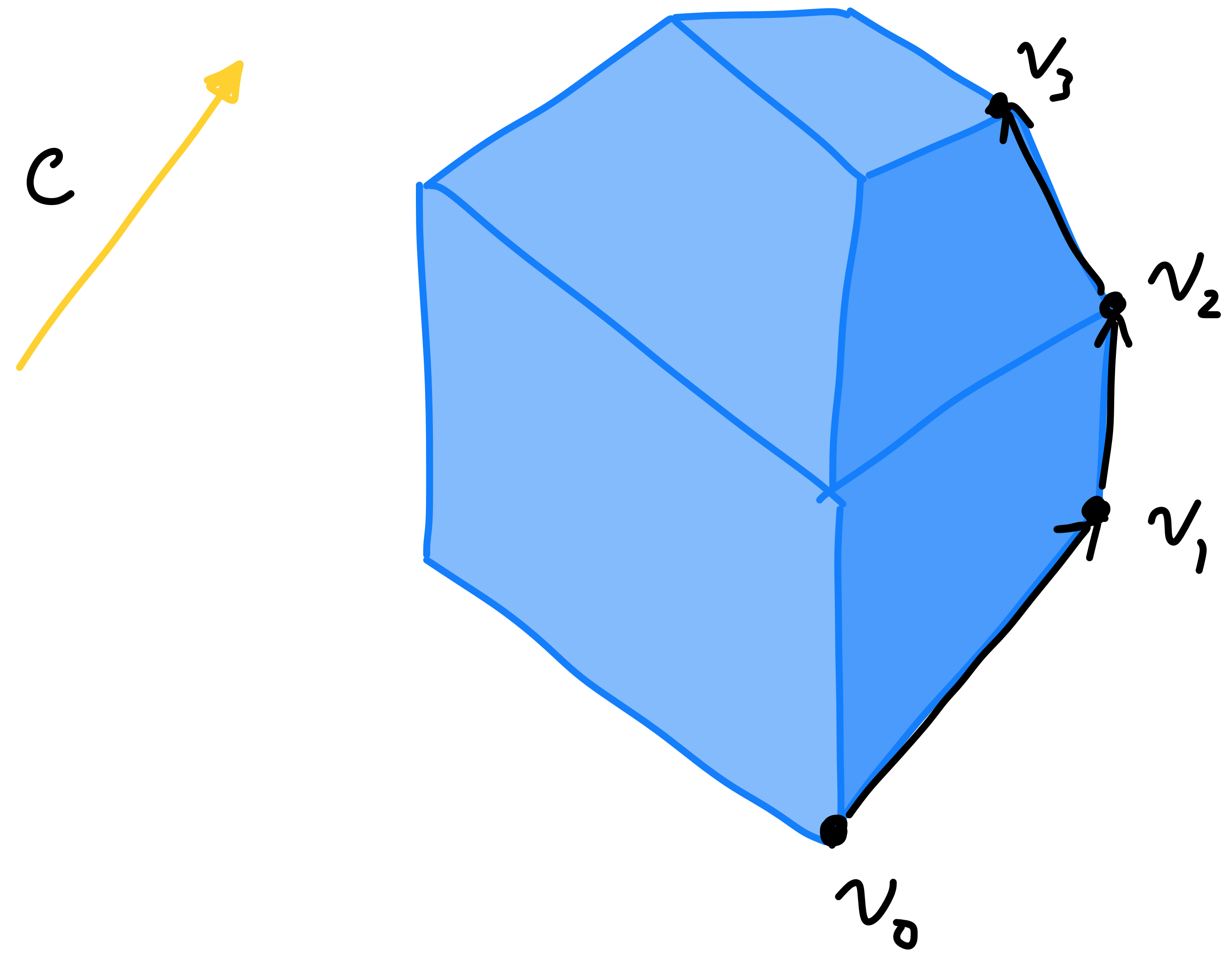
The simplex method

- Simplex is a greedy algorithm
- **High-level algorithm**:
  - Start from a vertex of the polytope
  - In each step, move to the neighboring vertex that optimizes $c^\top x$
  - Equivalently, move along the edge pointing the most in the c direction



The simplex method

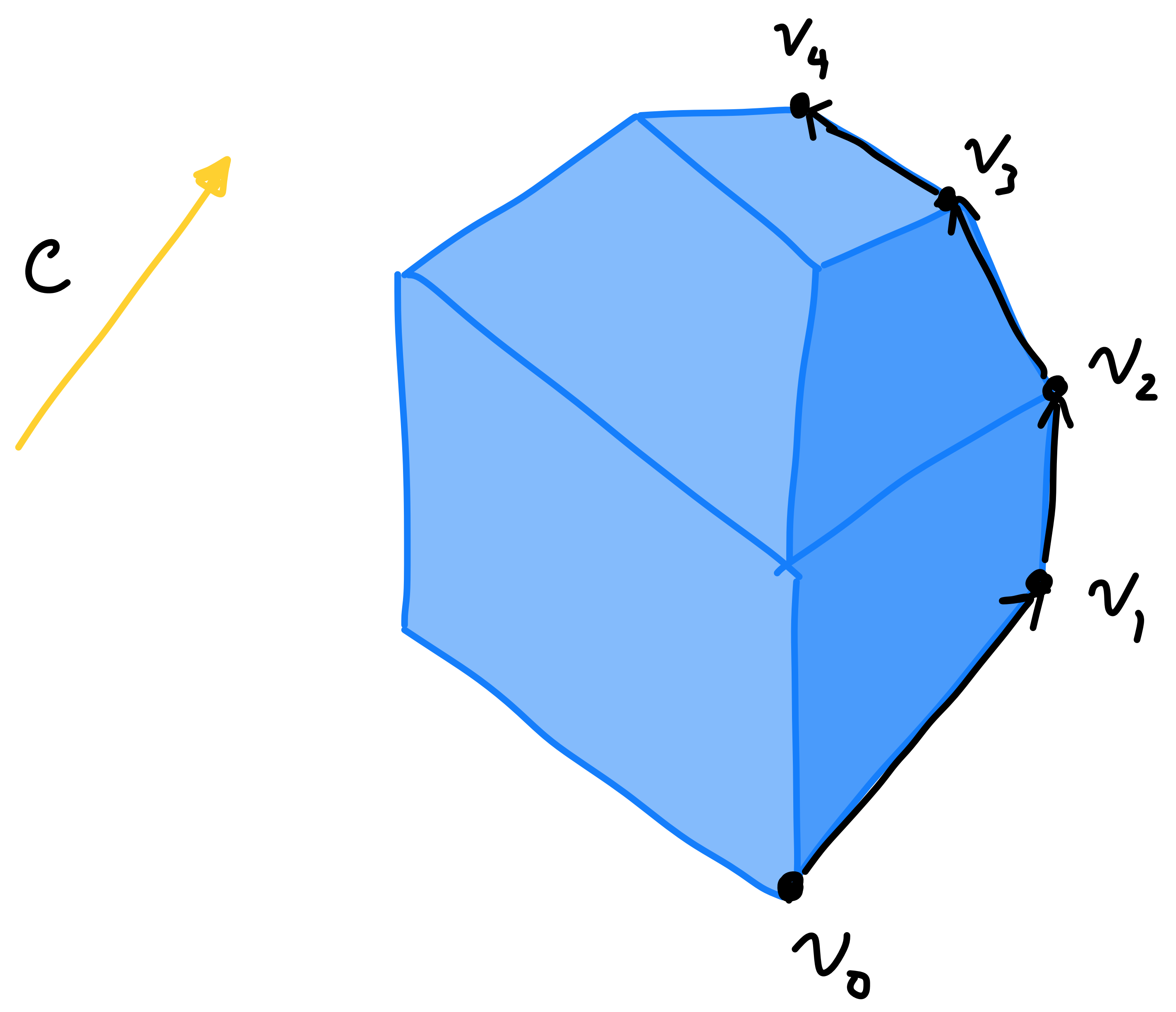
- Simplex is a greedy algorithm
- **High-level algorithm**:
  - Start from a vertex of the polytope
  - In each step, move to the neighboring vertex that optimizes $c^\top x$
  - Equivalently, move along the edge pointing the most in the c direction



The simplex method

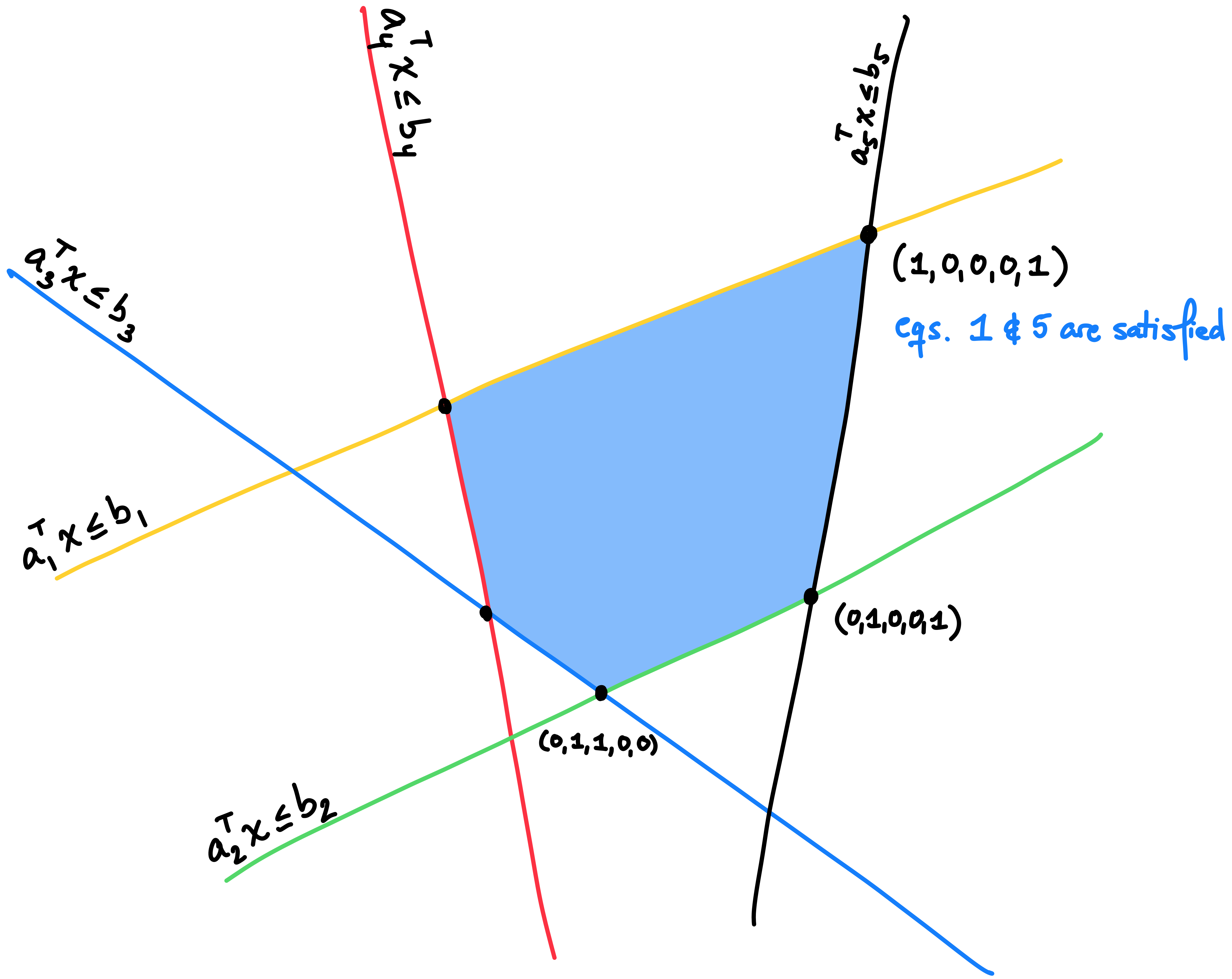
- Simplex is a greedy algorithm
- **High-level algorithm**:
  - Start from a vertex of the polytope
  - In each step, move to the neighboring vertex that optimizes $c^\top x$
  - Equivalently, move along the edge pointing the most in the c direction

all edges point away from c, so alg halts



The simplex method

- We are effectively consider a graph $G = (V, E)$ whose interior is the feasible region Γ.
- If we consider a feasible region defined by $\Gamma = \{Ax \leq b\}$ for $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$
    - Then, each vertex can be described by which n of the m equations are exactly satisfied
    - Describe vertices by points in $\{0,1\}^m$ of Hamming weight n
    - Two vertices are neighbors if they share all but 1 equation or equiv. the descriptions differ in two bits



The simplex method

Digging deeper into the algorithm

- Algorithm has two major steps:
    - Finding the first vertex (if one even exists as Γ could be infeasible)
    - Moving along an edge
- Moving along an edge:
    - Currently at a vertex described by n out of m equations
    - Can consider all possible vertices that share all but one equation
    - At most $n \cdot (m - n)$ neighbors
    - Gives a polynomial time algorithm for moving along an edge

The simplex method

Digging deeper into the algorithm

- **Finding the first vertex**

Input: $\max\ c^T x$

s.t. $Ax \leq b$

$x \geq 0$

The feasible region is

$\Gamma\ \{Ax \leq b,\ x \geq 0\}$

Goal: Output a vertex of Γ.

Consider a second LP

known as slack variables

$$\min\ z_1 + \dots + z_m$$

$$\text{s.t.}\quad b_i - a_i^T x \leq z_i \quad \forall\, i = 1, \dots, m$$

$$x \geq 0$$

$$z \geq 0.$$

Notice that $(x = 0,\ z = b^{(+)})$ is a vertex of 2^{nd} LP.

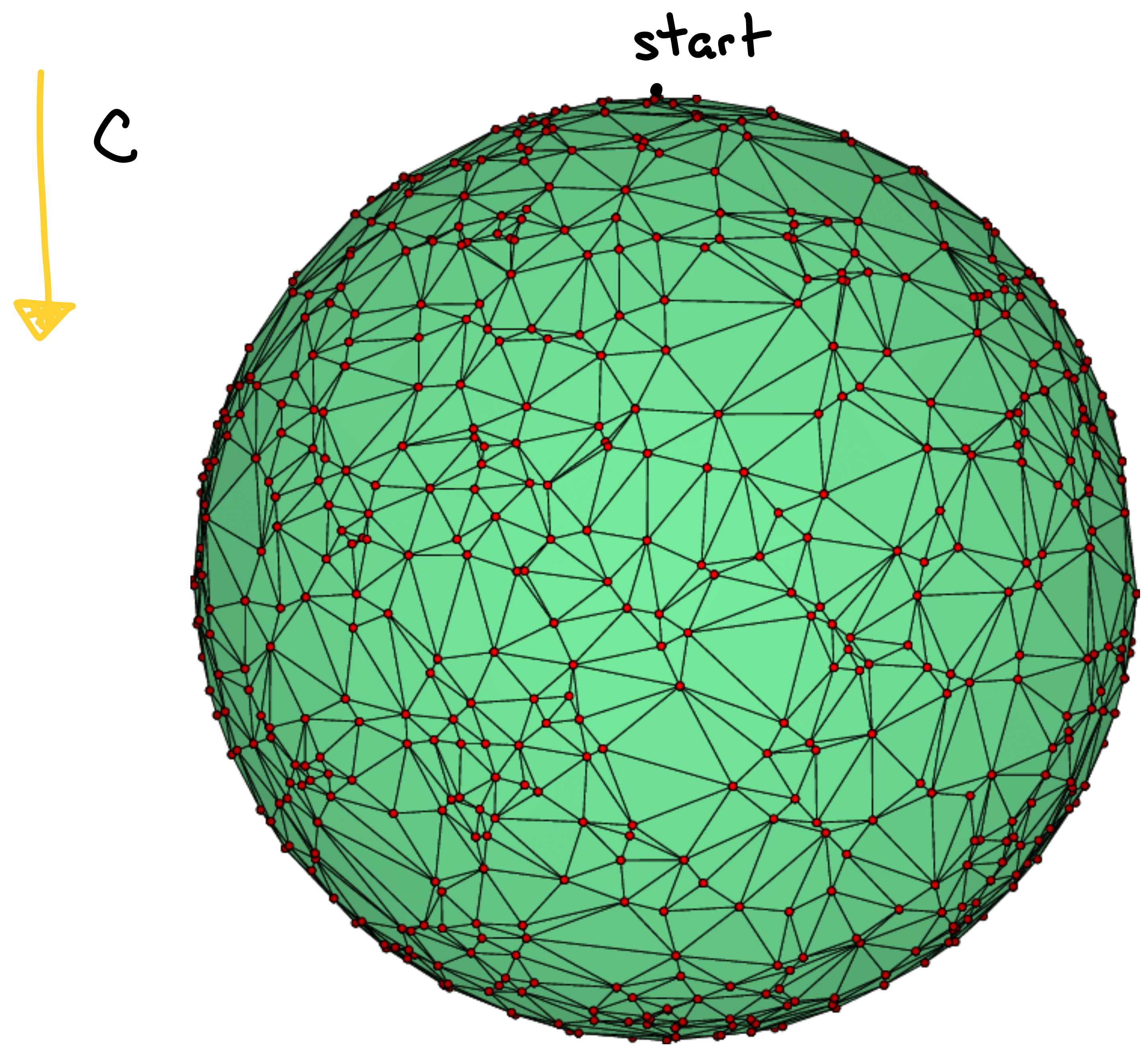
Since we know a vertex of 2^{nd} LP, we can find it's OPT with simplex.

Claim: If (x, z) is OPT of 2^{nd} LP, then x is a vertex of Γ. Proof: exercise.

We have found a vertex of original polytope.

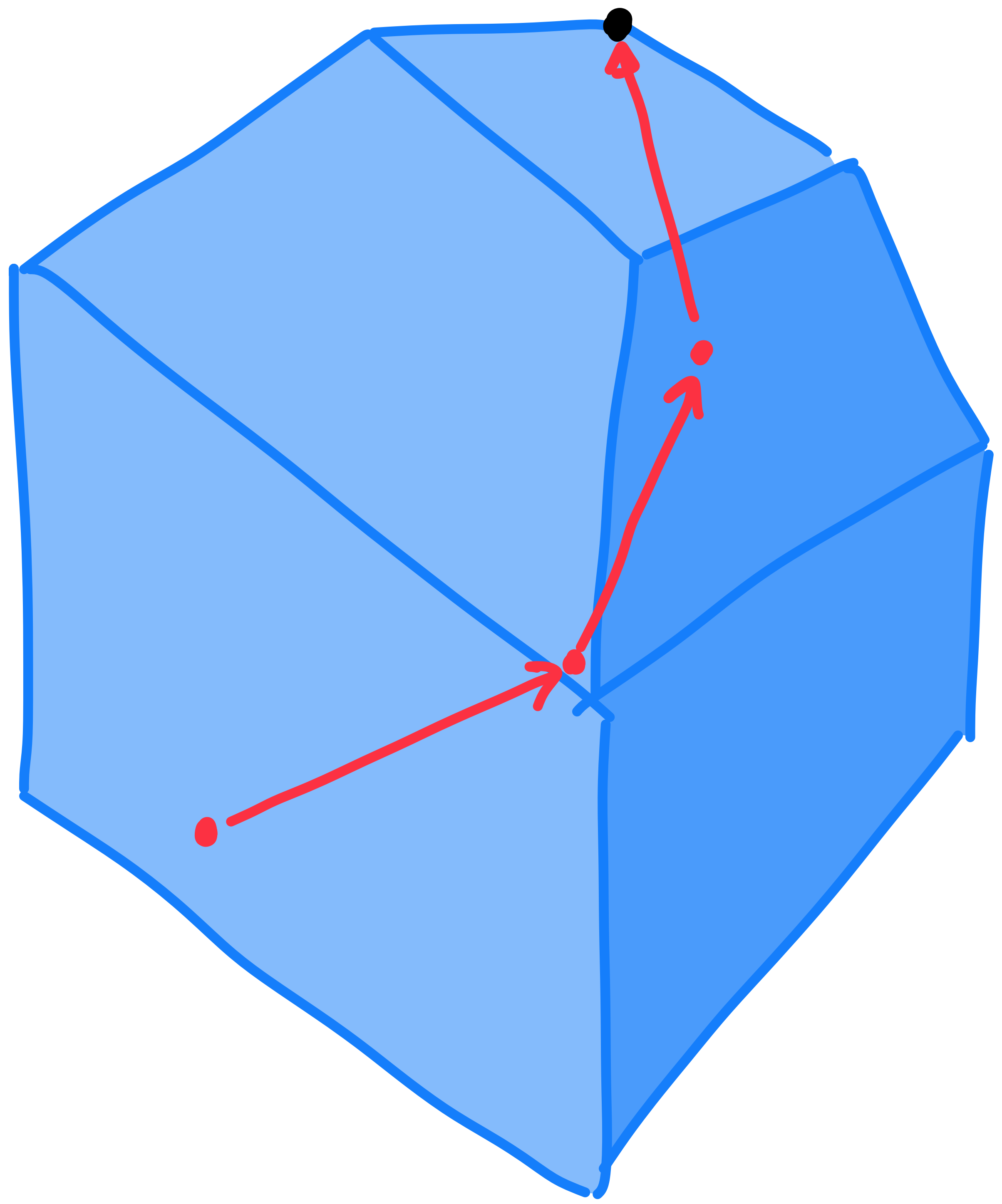
The simplex method

- We have not given runtimes for the simplex method on purpose
    - The runtime can be exponential because the algorithm goes on the *outside* of the polytope which could have lots of vertices, edges, and facets
    - However, simplex runs remarkably well in practice
    - Is there a reconciliation? An algorithm that may do okay in practice but has guaranteed worst case runtime that is polynomial?



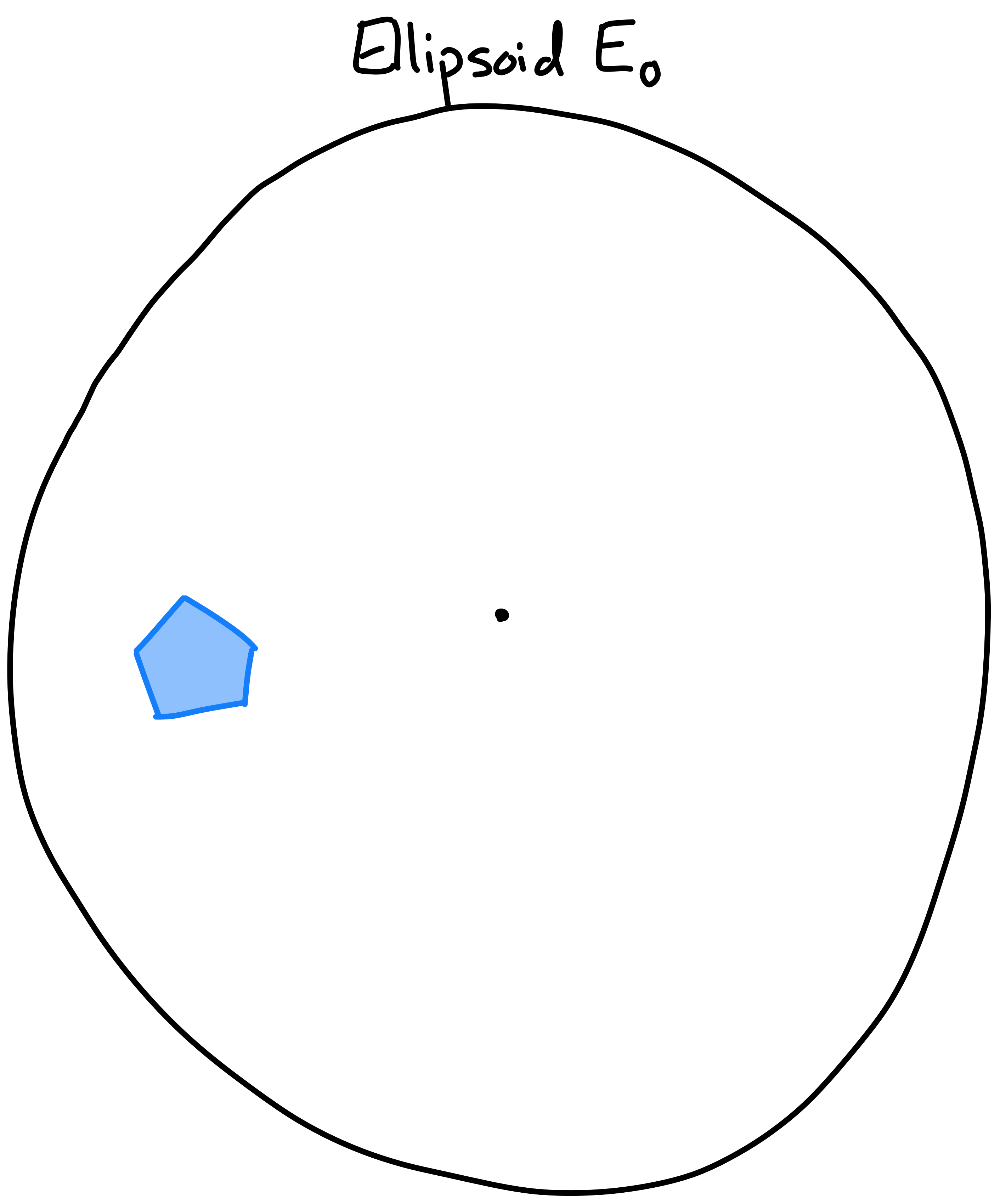
Interior point and ellipsoid methods

- **Interior point**:
  - Keep track of a point *inside* the polytope
  - Follow a trajectory through the interior to optimal solution
  - Solve a sequence of easier problems to approximate original LP, gradually becoming more accurate
  - Runs about as fast as simplex in practice and has guarantees on runtime
  - The "state-of-the-art" algorithm and a key step in optimal algorithms for problems like max flow



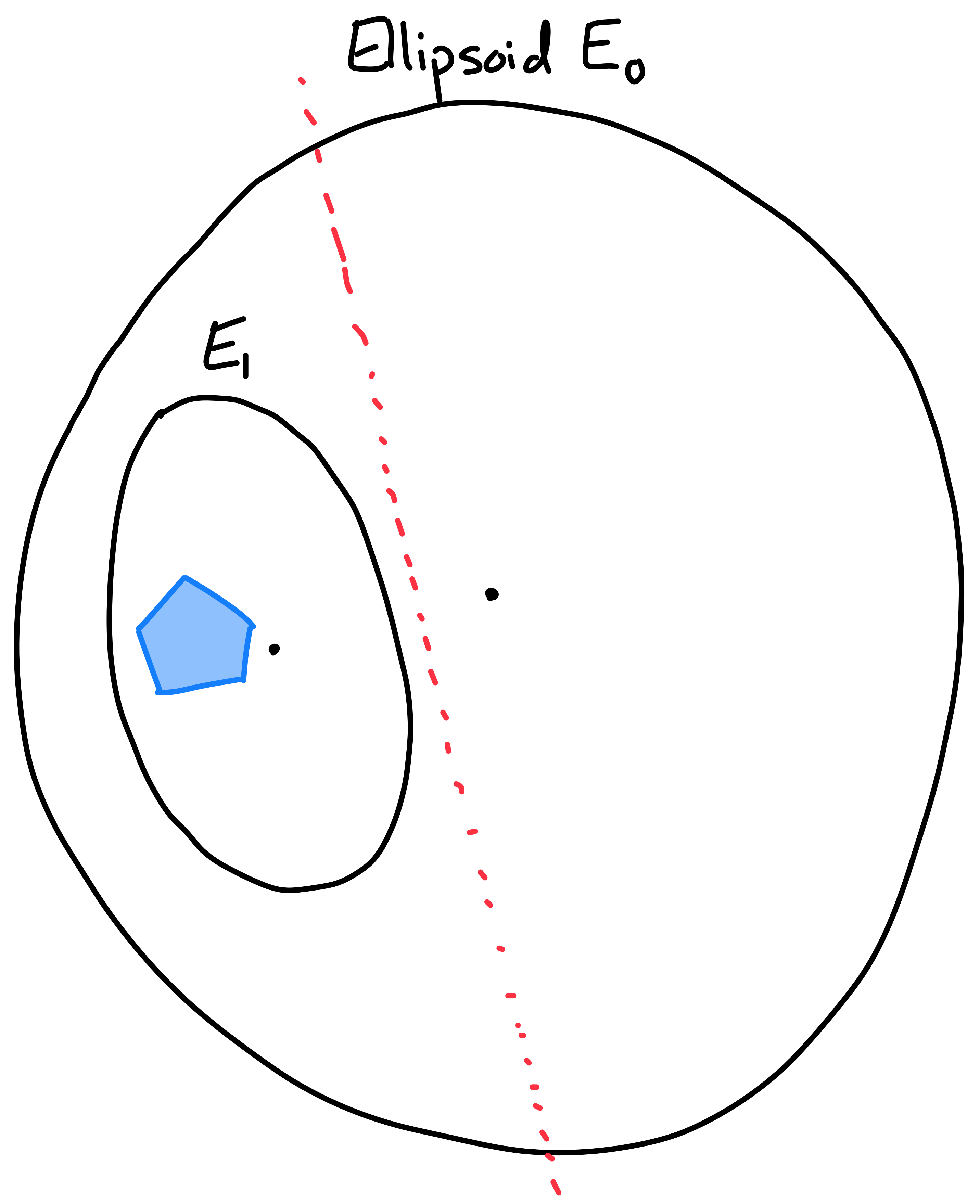
Interior point and ellipsoid methods

- **Ellipsoid method**:
    - Using LP duality, convert problem from optimizing a linear polytope to finding a feasible point in a different polytope Γ
    - Generate a sequence of ellipsoids that always contain Γ
    - Each time find a smaller ellipsoid (by guaranteed ratio) until the center of the ellipsoid must be in Γ
    - Very slow in practice but first guaranteed algorithm for solving LPs



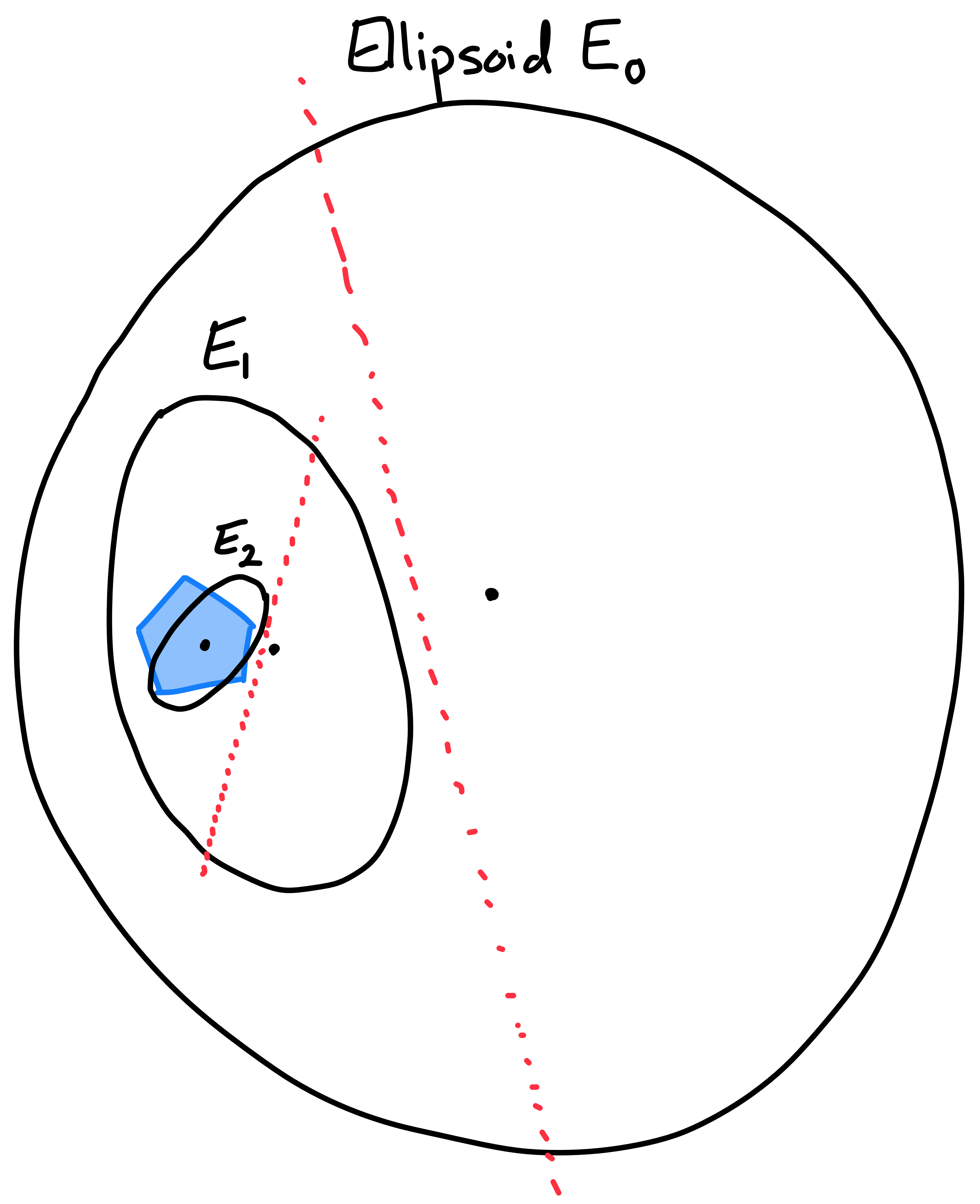
Interior point and ellipsoid methods

- **Ellipsoid method:**
  - Using LP duality, convert problem from optimizing a linear polytope to finding a feasible point in a different polytope Γ
  - Generate a sequence of ellipsoids that always contain Γ
  - Each time find a smaller ellipsoid (by guaranteed ratio) until the center of the ellipsoid must be in Γ
  - Very slow in practice but first guaranteed algorithm for solving LPs



Interior point and ellipsoid methods

- **Ellipsoid method:**
  - Using LP duality, convert problem from optimizing a linear polytope to finding a feasible point in a different polytope Γ
  - Generate a sequence of ellipsoids that always contain Γ
  - Each time find a smaller ellipsoid (by guaranteed ratio) until the center of the ellipsoid must be in Γ
  - Very slow in practice but first guaranteed algorithm for solving LPs



Zero-sum games