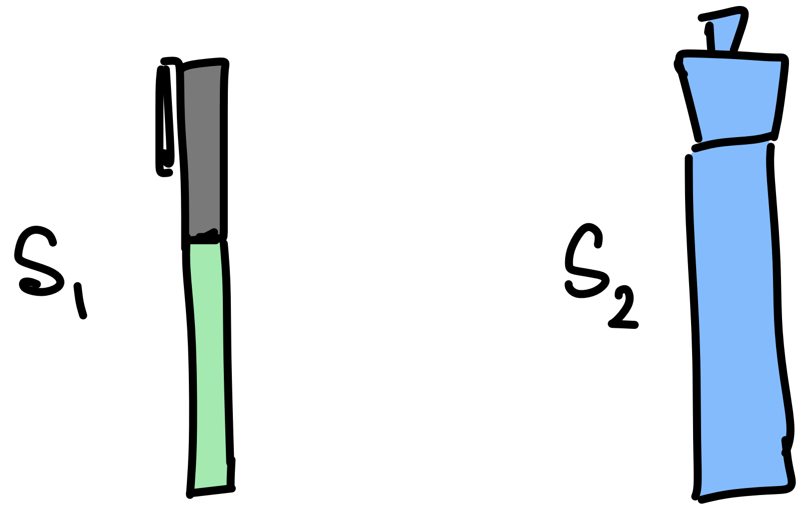
Lecture 21

Linear programming III

Chinmay Nirkhe | CSE 421 Spring 2025

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.
- Recall, L_1 is the amount of labor required for a pen, I_1 is the amount of ink required for a pen, etc.
- It is only economical to **buy** a pen if $y_L L_1 + y_I I_1 + y_P P_1 \geq S_1$
    - The left hand side is the cost to make a pen **at market price**
    - And the right hand side is the cost to **buy a pen**
    - Similarly, buy markers only if $y_L L_2 + y_I I_2 + y_P P_2 \geq S_2$.
- The dual perspective is calculating the **minimal** total materials price ($y_L L + y_I I + y_P P$) while its still able to sell pens and markers. This is the **dual problem**.
- The primal perspective is calculating the **maximal** total profit subject to the material restrictions.

Linear programming duality

- Now let's imagine there are market prices for the 3 materials: y_L, y_I, y_P.
- Recall, L_1 is the amount of labor required for a pen, I_1 is the amount of ink required for a pen, etc.
- If $y_L L_1 + y_I I_1 + y_P P_1 < S_1$, then it would not be economical to **buy** a pen
  - The left hand side is the cost to make a pen **at market price**
  - And the right hand side is the cost to **buy a pen**
  - Buy pens only if $y_L L_1 + y_I I_1 + y_P P_1 \geq S_1$ and buy markers only if $y_L L_2 + y_I I_2 + y_P P_2 \geq S_2$.
- The dual perspective is calculating the **minimal** total materials price ($y_L L + y_I I + y_P P$) while its still able to sell pens and markers. This is the **dual problem**.
- The primal perspective is calculating the **maximal** total profit subject to the material restrictions.

Linear programming duality

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

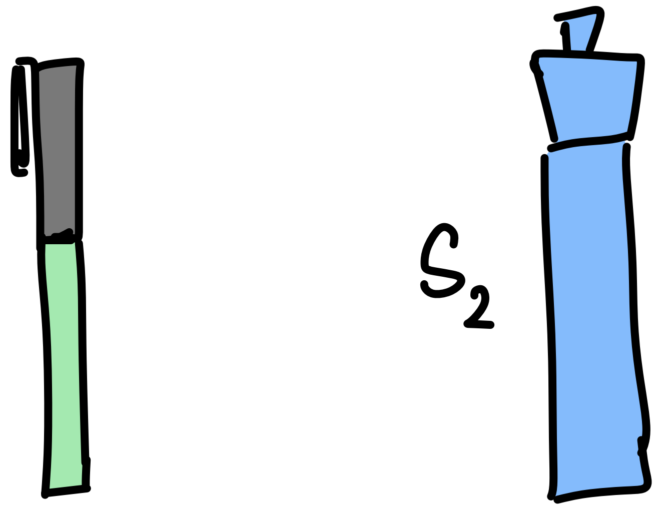
$$\text{s.t.} \quad L_1 x_1 + L_2 x_2 \le L$$

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

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

$$x_1, x_2 \ge 0.$$

S_1 S_2



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

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

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

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

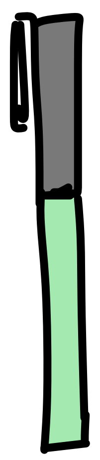
y_L y_I y_P

labor ink plastic

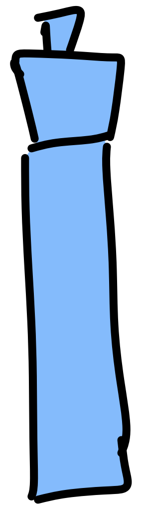
Linear programming duality

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

S_1



S_2



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

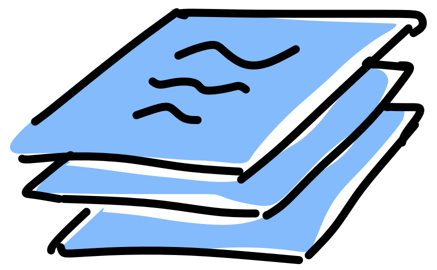
y_L

labor

y_I

ink

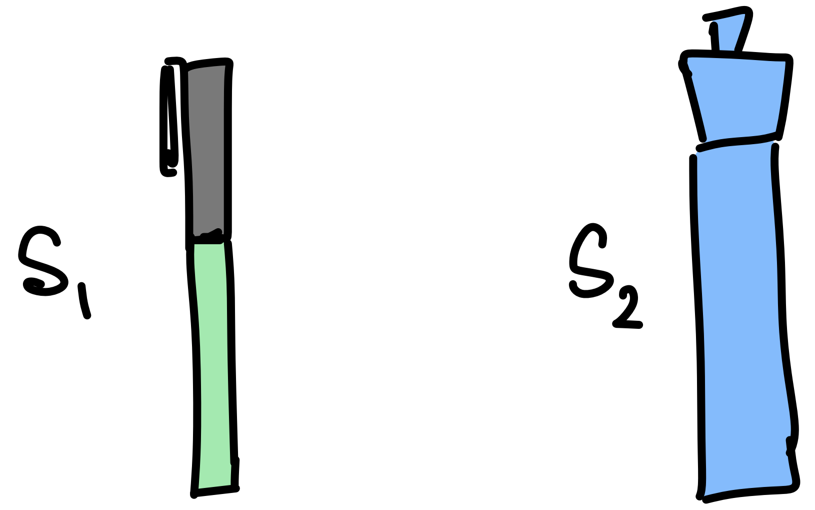
y_P



plastic

Linear programming duality

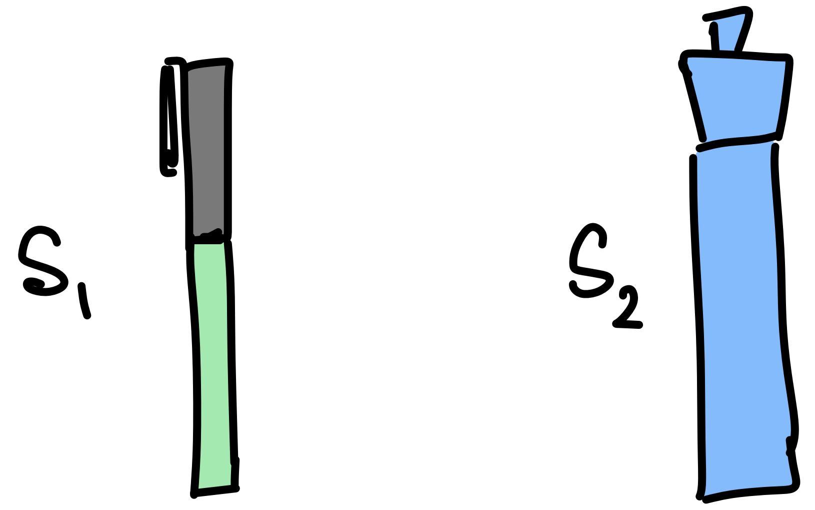
$$\begin{aligned}
\max \quad & s_1 x_1 + s_2 x_2 \\
\text{s.t.} \quad & L_1 x_1 + L_2 x_2 \le L \\
& I_1 x_1 + I_2 x_2 \le I \\
& P_1 x_1 + P_2 x_2 \le P \\
& x_1, x_2 \ge 0.
\end{aligned}$$



$$\begin{aligned}
\min \quad & y_L L + y_I I + y_P P \\
\text{s.t.} \quad & y_L L_1 + y_I I_1 + y_P P_1 \ge s_1 \\
& y_L L_2 + y_I I_2 + y_P P_2 \ge s_2 \\
& y_L, y_I, y_P \ge 0.
\end{aligned}$$

Linear programming duality

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



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

Linear programming duality

Primal linear program (P)

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

Dual linear program (D)

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

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 $c^\top x \leq y^\top A x \leq b^\top y$.
  - 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 $c^\top x \leq y^\top A x \leq b^\top y$.

Since x is feasible for (P),

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

Since y is feasible for (D),

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

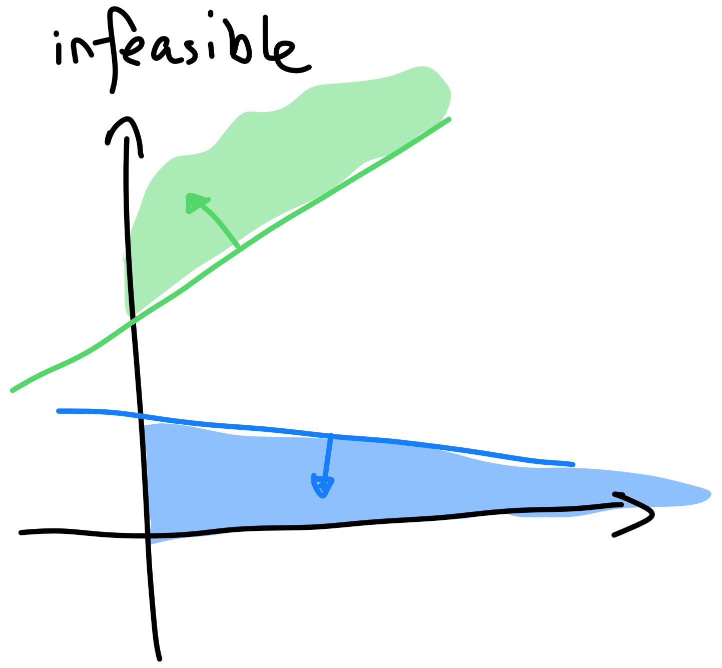
Then,

$$y^T(Ax) \le y^T b \quad \text{by (1).}$$
$$= b^T y$$

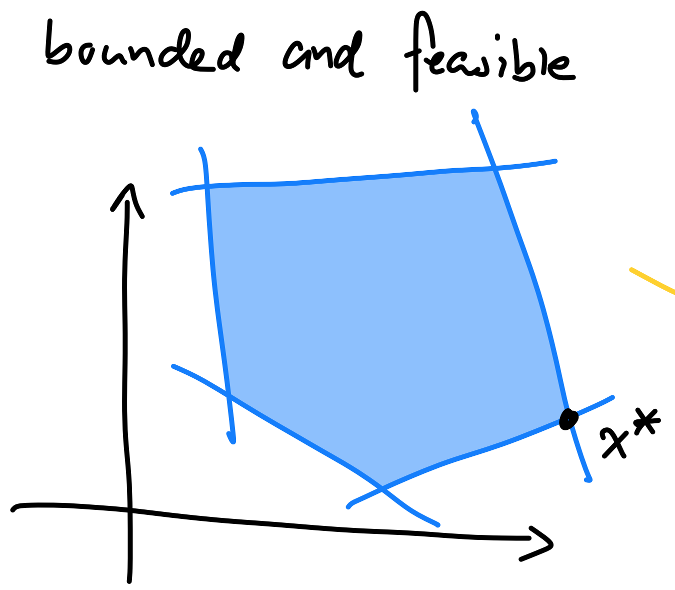
And,

$$c^T x \le (A^T y)^T x$$
$$= (y^T A) x$$
$$= y^T A x.$$

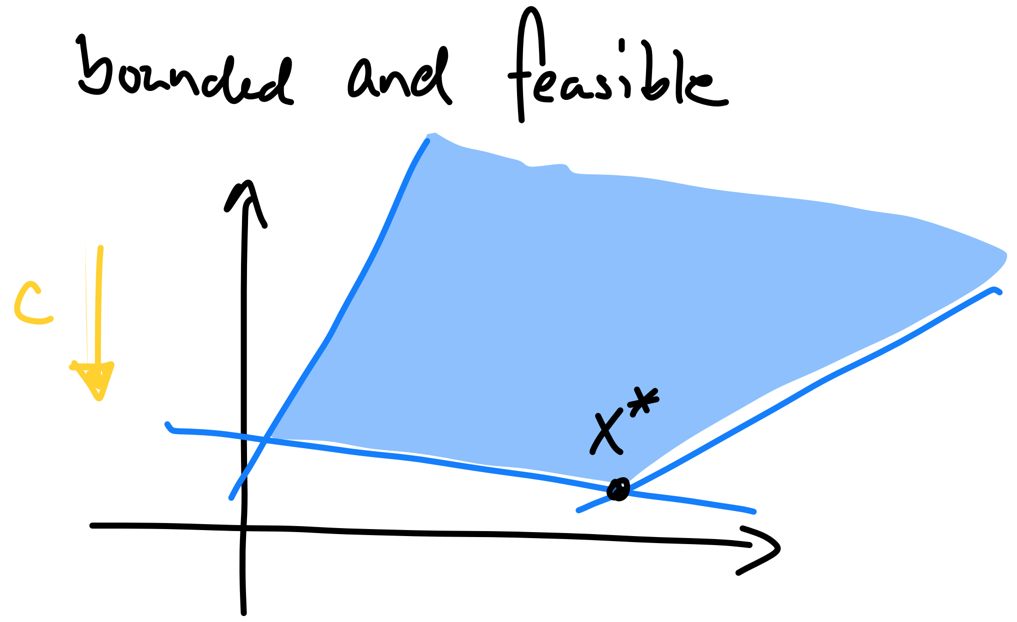
Proving weak duality



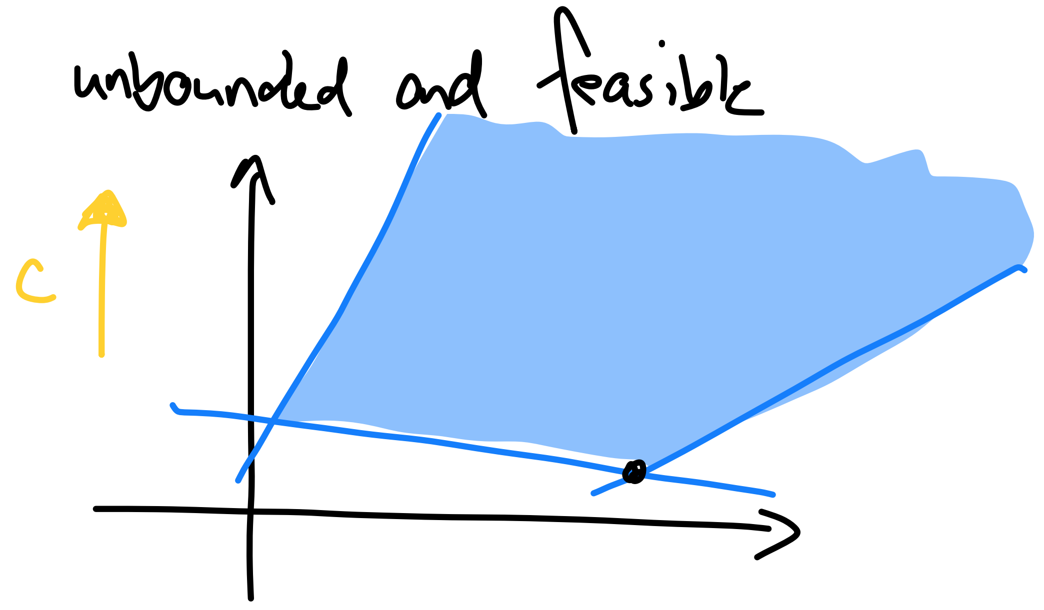
- If $(\mathscr{P})$ is unbounded
  - Then for all $N \in \mathbb{N}$, there exists $x \in \Gamma$ such that $N < c^\top x$.



- If $(\mathscr{D})$ is feasible,
  - then for any feasible y, $c^\top x \le y^\top A x \le b^\top y$.



- 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, $c^\top x' \leq y^\top A x' \leq y^\top b$ 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

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

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

$$f \ge 0.$$

capacity constraints

conservation of flow

An observation about duality

- If the primal $(\mathscr{P})$ is an optimization with n variables and m equations,
  - then the dual $(\mathscr{D})$ 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 $(\mathscr{P})$ correspond to the m variables of the dual $(\mathscr{D})$. 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 $m + 2n - 2$ variables
  - It will be hard to see that this LP is equivalent to the min cut problem

$$(P) = \begin{cases} \max & g^T f \\ \text{s.t.} & \begin{bmatrix} \mathbb{I}_E \\ h_{v_1} \\ -h_{v_1} \\ \vdots \\ h_{v_n} \\ -h_{v_n} \end{bmatrix} \cdot f \le \begin{bmatrix} c \\ 0 \end{bmatrix}, \\ & f \ge 0. \end{cases}$$

$$(D) = \begin{cases} \min & [\,c \mid 0 \cdots 0\,] \cdot y \\ \text{s.t.} & \begin{bmatrix} \mathbb{I}_E & h_{v_1} & -h_{v_1} & \cdots & h_{v_n} & -h_{v_n} \end{bmatrix} \cdot y \ge \begin{bmatrix} g \end{bmatrix} \\ & y \ge 0. \end{cases}$$

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 be 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

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

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

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

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 be 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

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

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

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.

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

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

Lessons from duality

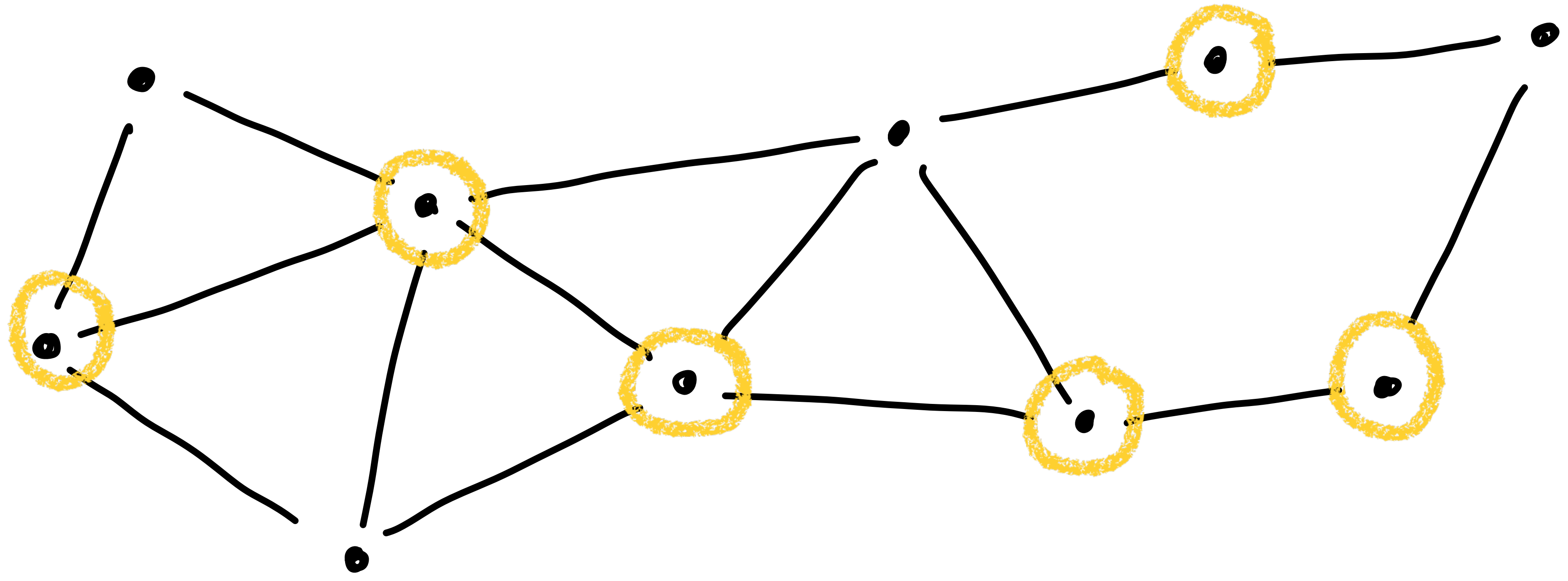
- We reproved the max flow/min cut duality from the flow unit of this course
- **Observation:** Min cut does not have an intuitive poly-sized LP
  - However, it does have a m variable and $|P|$ equations sized LP
  - Therefore, its has a dual (max flow) with $|P|$ variables and m equations
  - Max flow 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?

One variable x_v for every vertex v.

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

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

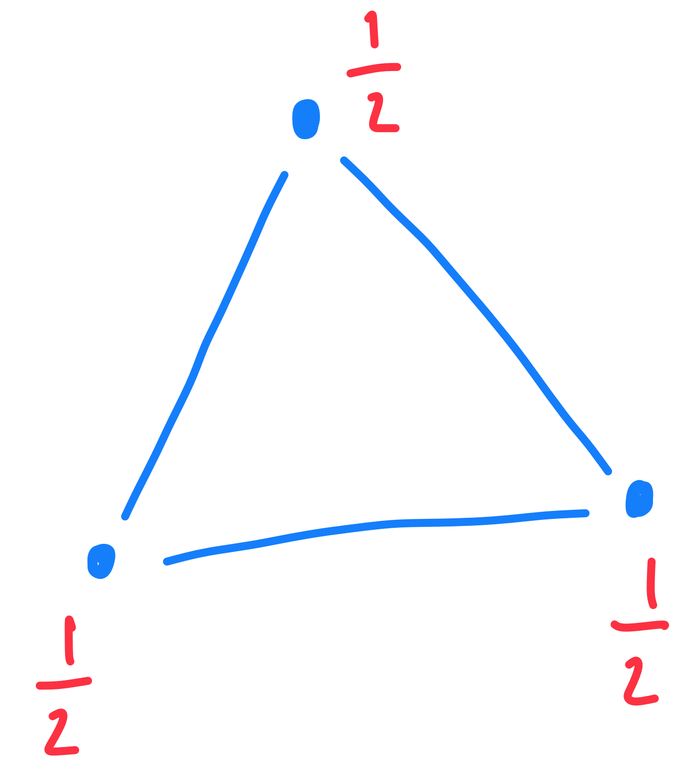
What's a problem LPs can't solve?

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.

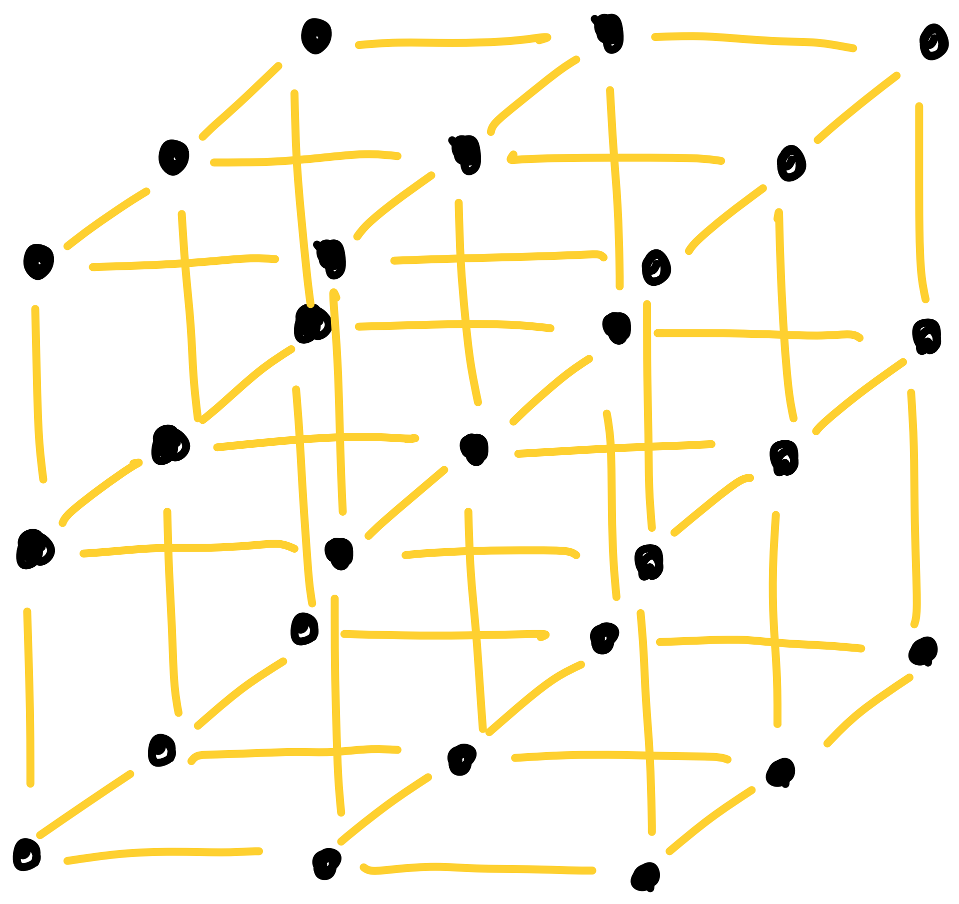
One variable x_v for every vertex v.

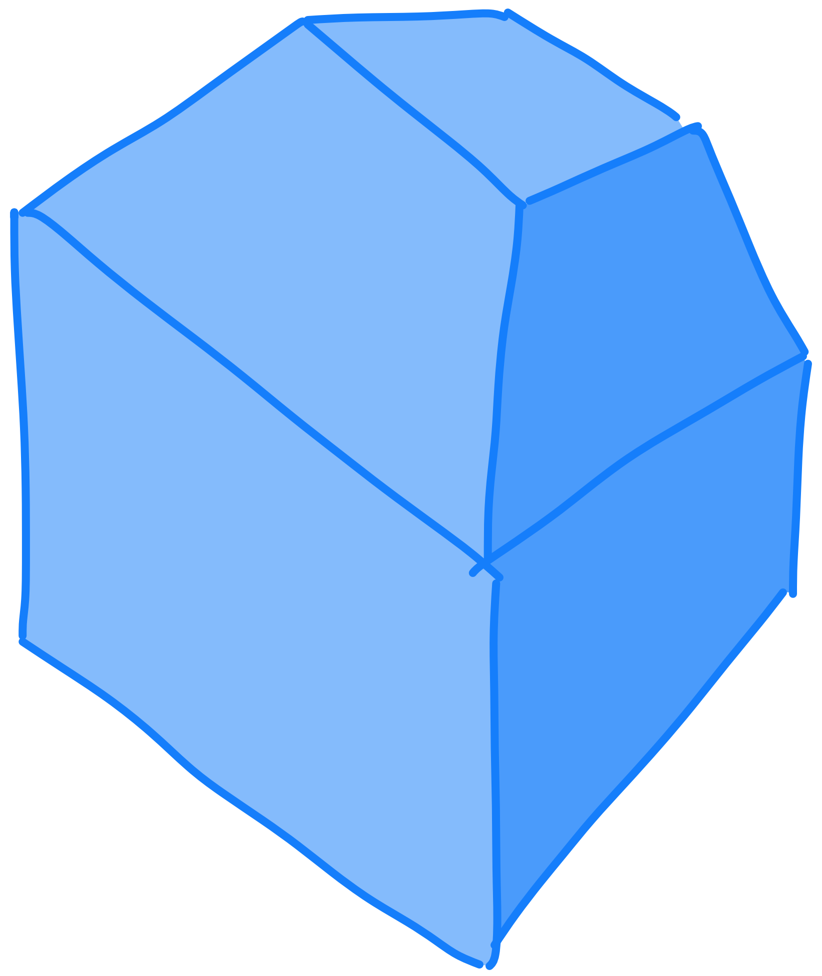
$$\begin{cases} \min \sum_{v \in V} x_v \\ \text{s.t.} \quad x_v \leq 1 \quad \forall v \in V \\ x_u + x_v \geq 1 \quad \forall e = (u,v) \in E \\ x \geq 0 \end{cases}$$

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.





Max flow versus vertex cover

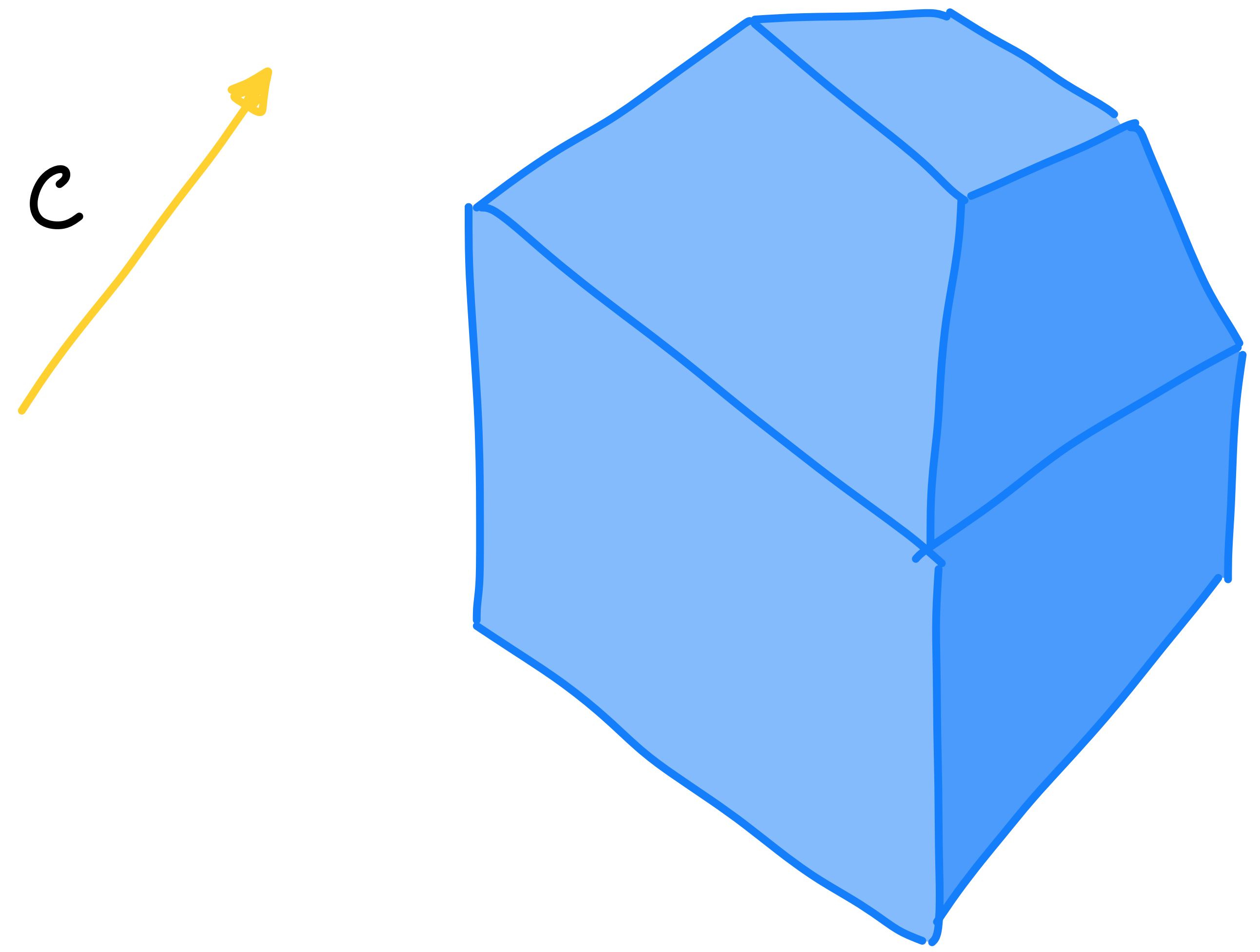
- Why can max flow natively guarantee integer solutions while vertex cover cannot?
- Recall, the optimum of an LP occurs at a vertex
  - The vertices of an LP correspond to points where linear equations are exactly satisfied
  - Turns out flow equations when exactly satisfied always have integer solutions
    - 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

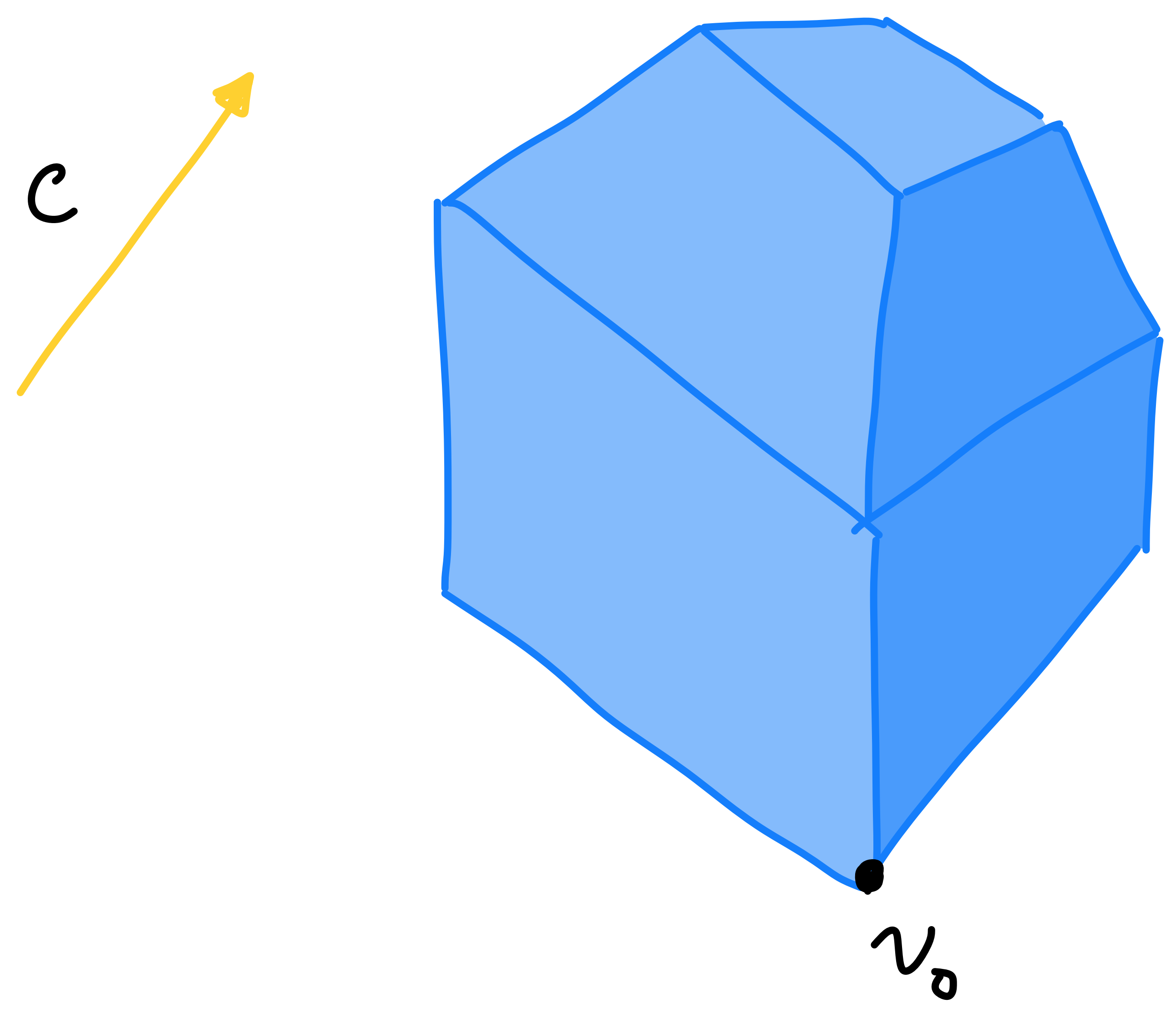
The simplex method

- 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



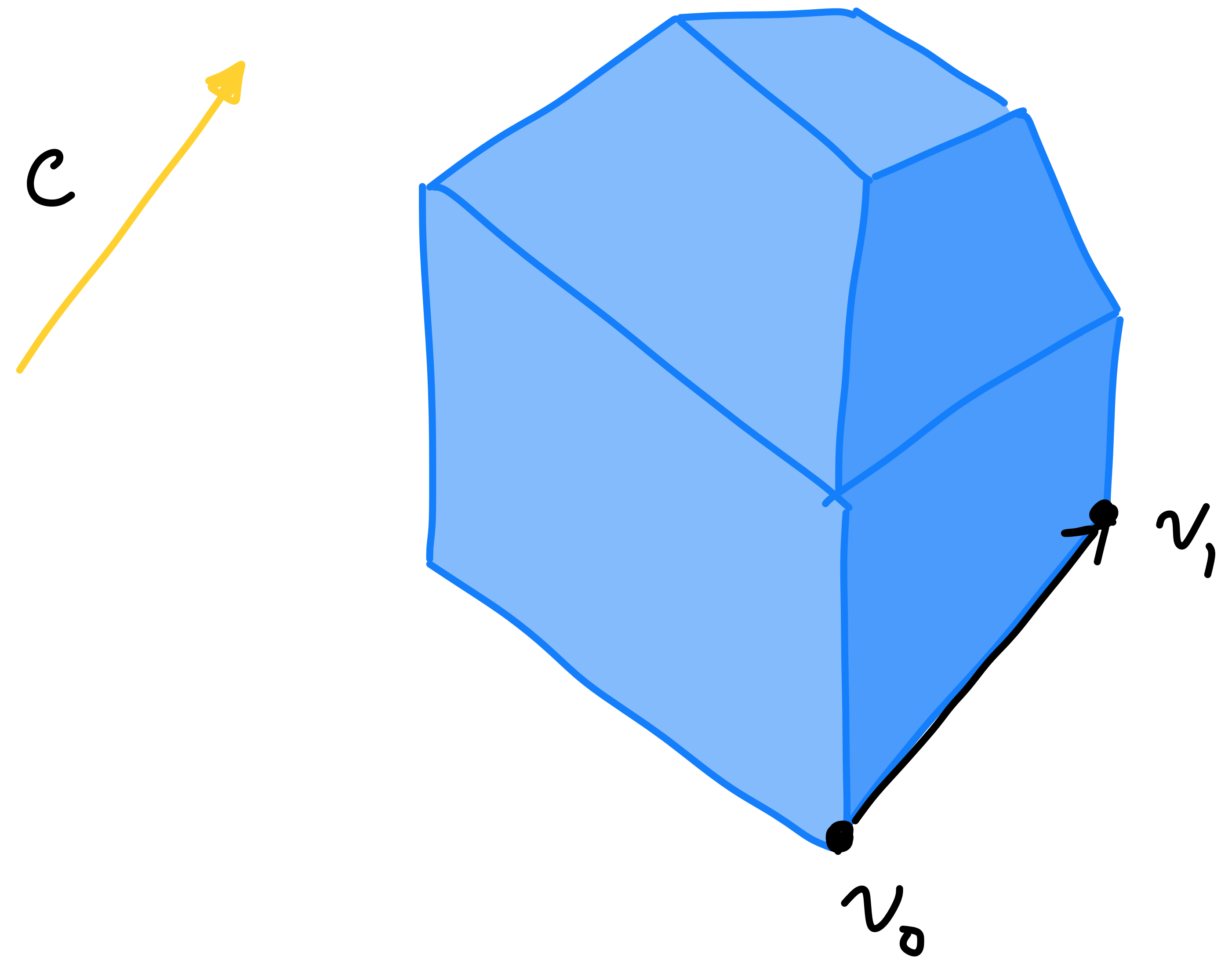
The simplex method

- 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



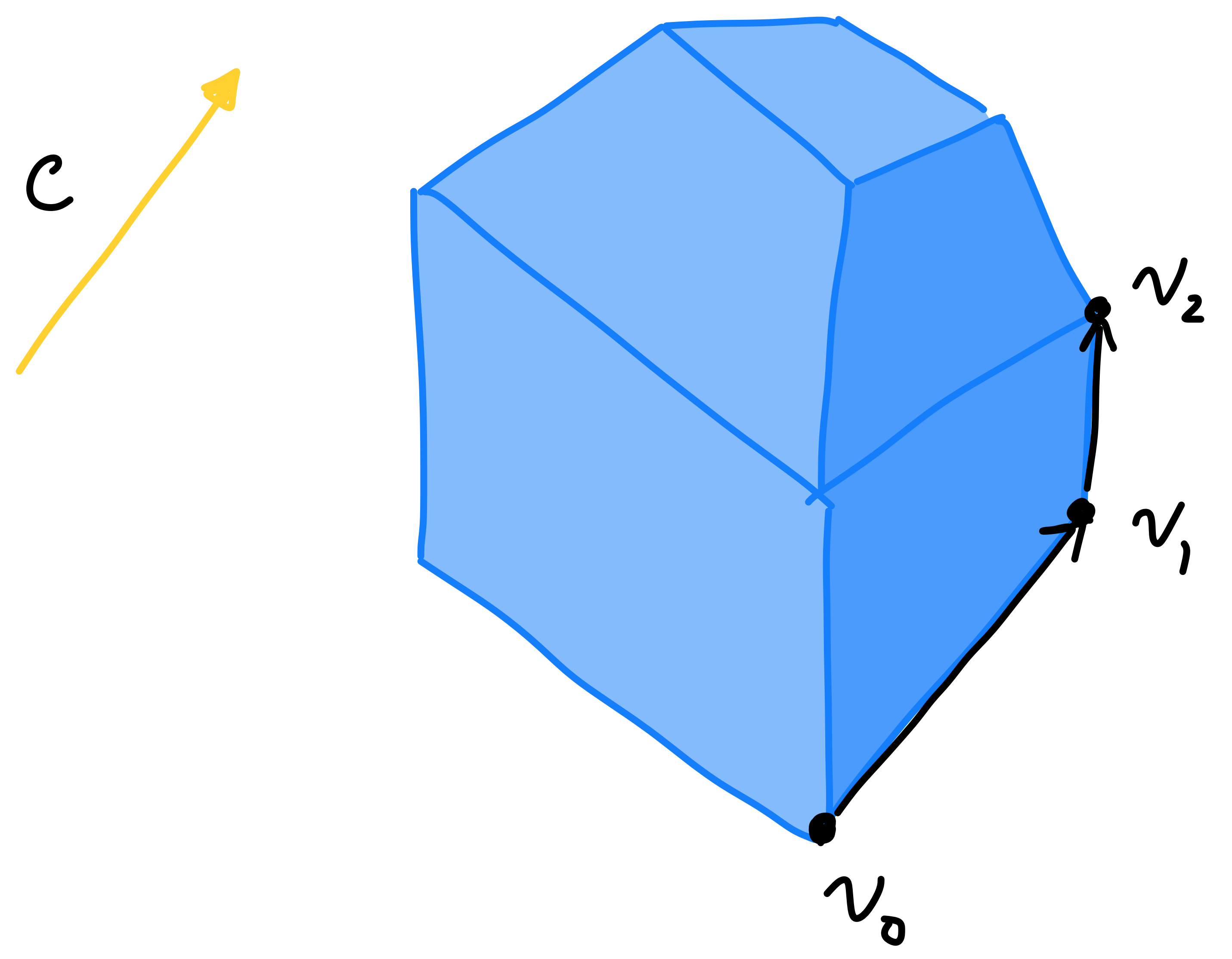
The simplex method

- 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



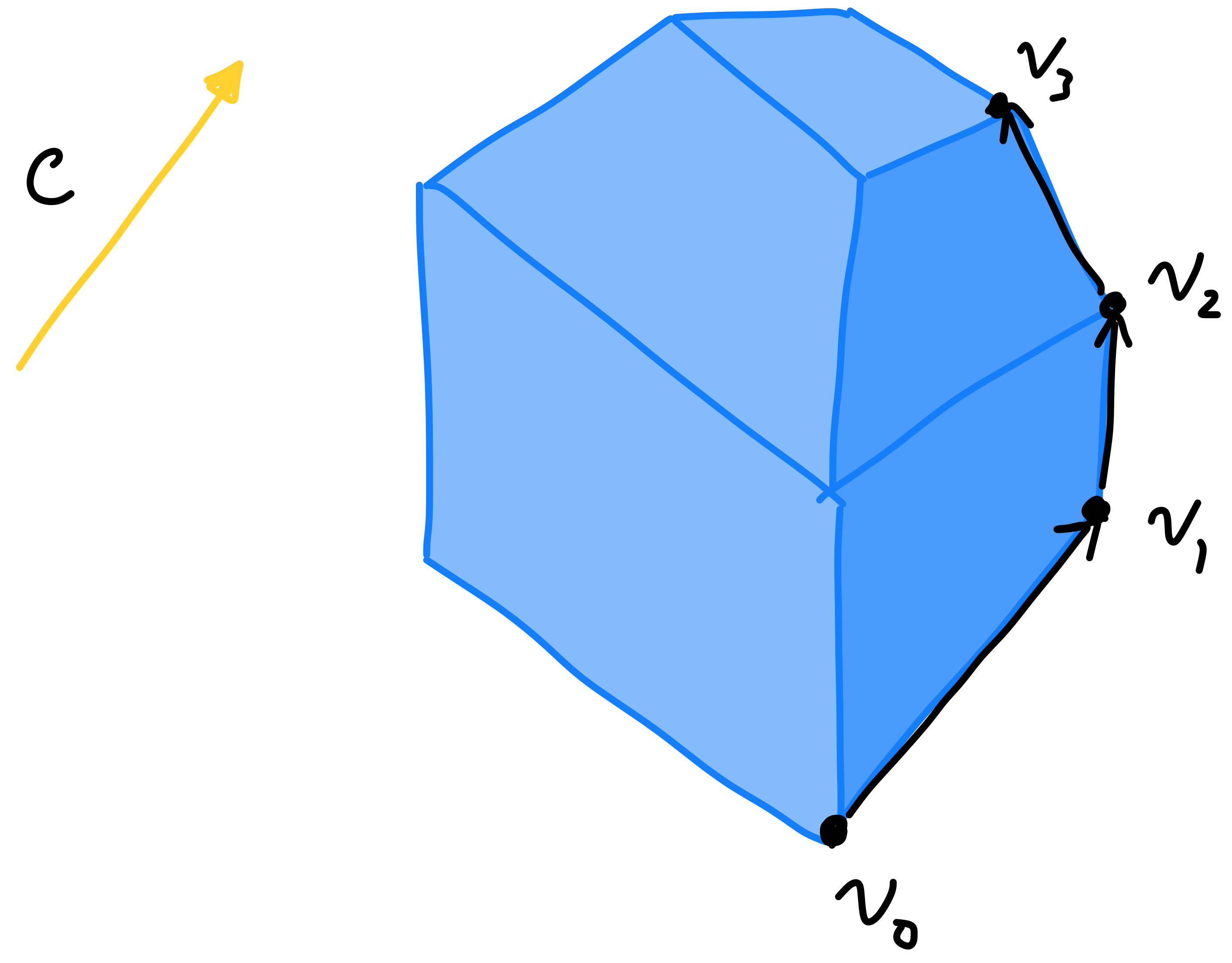
The simplex method

- 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



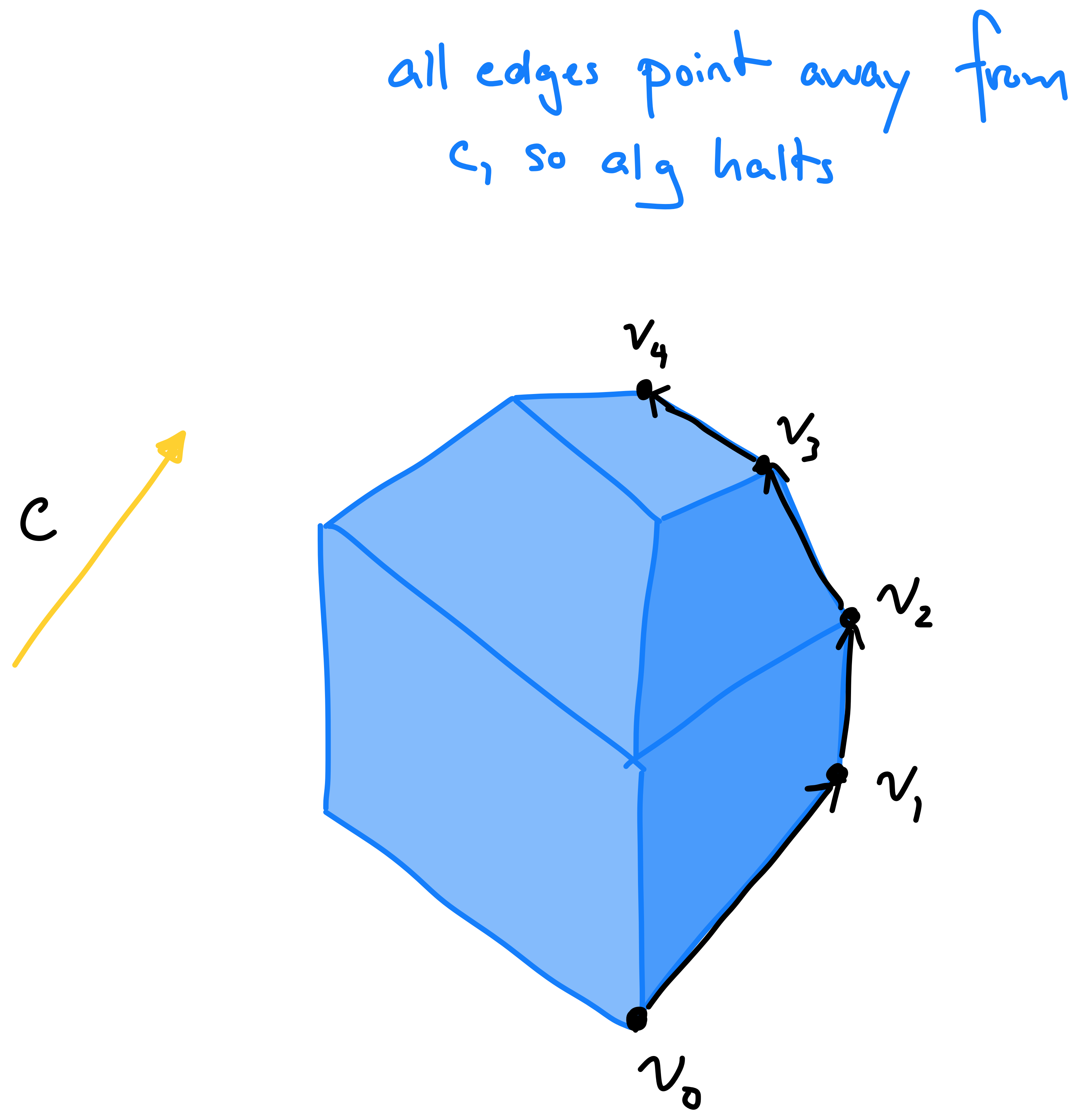
The simplex method

- 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



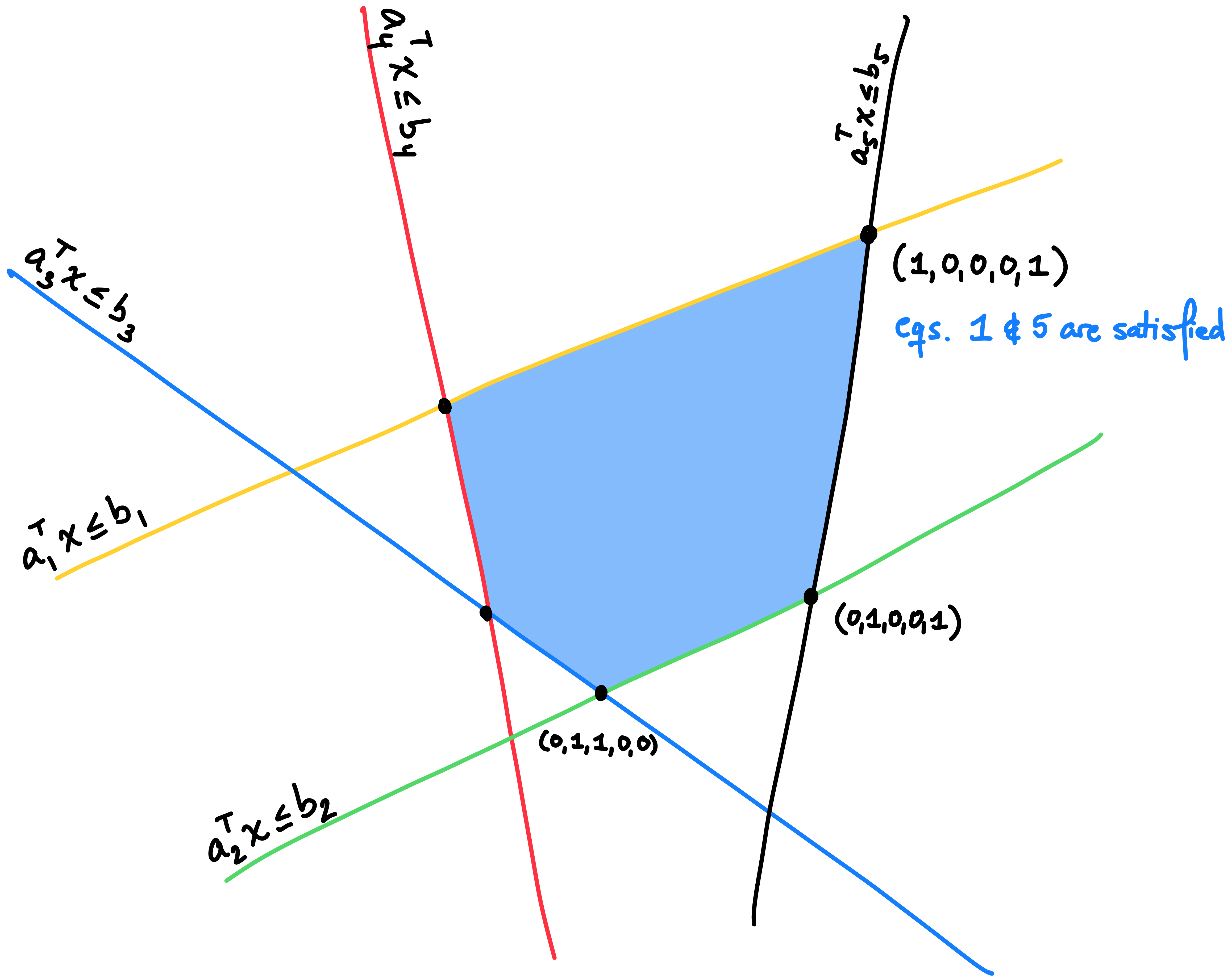
The simplex method

- 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



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. $Ax \le b$, $x \ge 0$

The feasible region is $\Gamma\ \{Ax \le b,\ x \ge 0\}$

Goal: Output a vertex of Γ.

Consider a second LP

$\min\ z_1 + \dots + z_m$

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

$x \ge 0$

$z \ge 0$.

known as slack variables

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

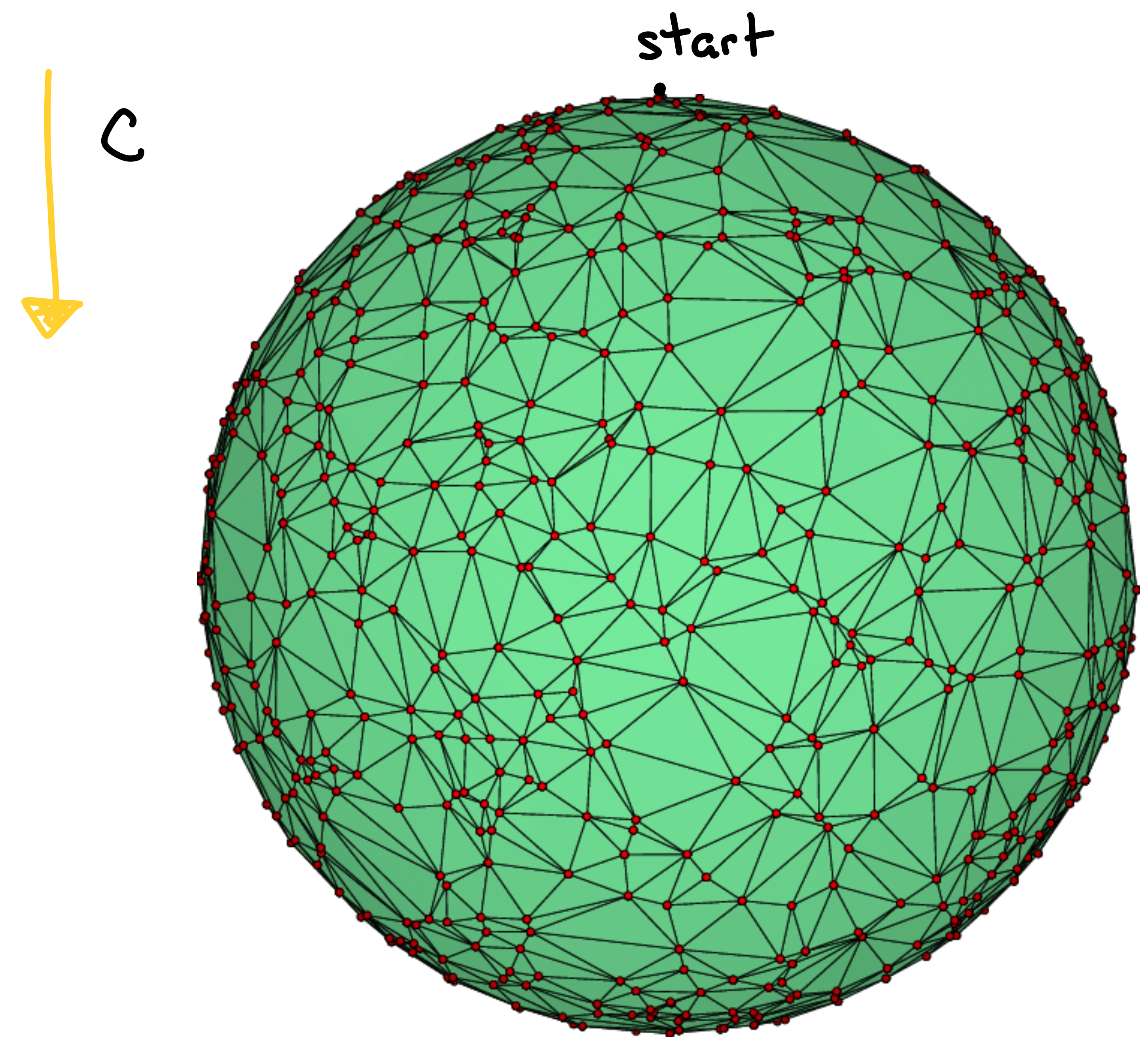
Since we know a vertex of 2^{nd} LP, we can find it's OPT with simplex.

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

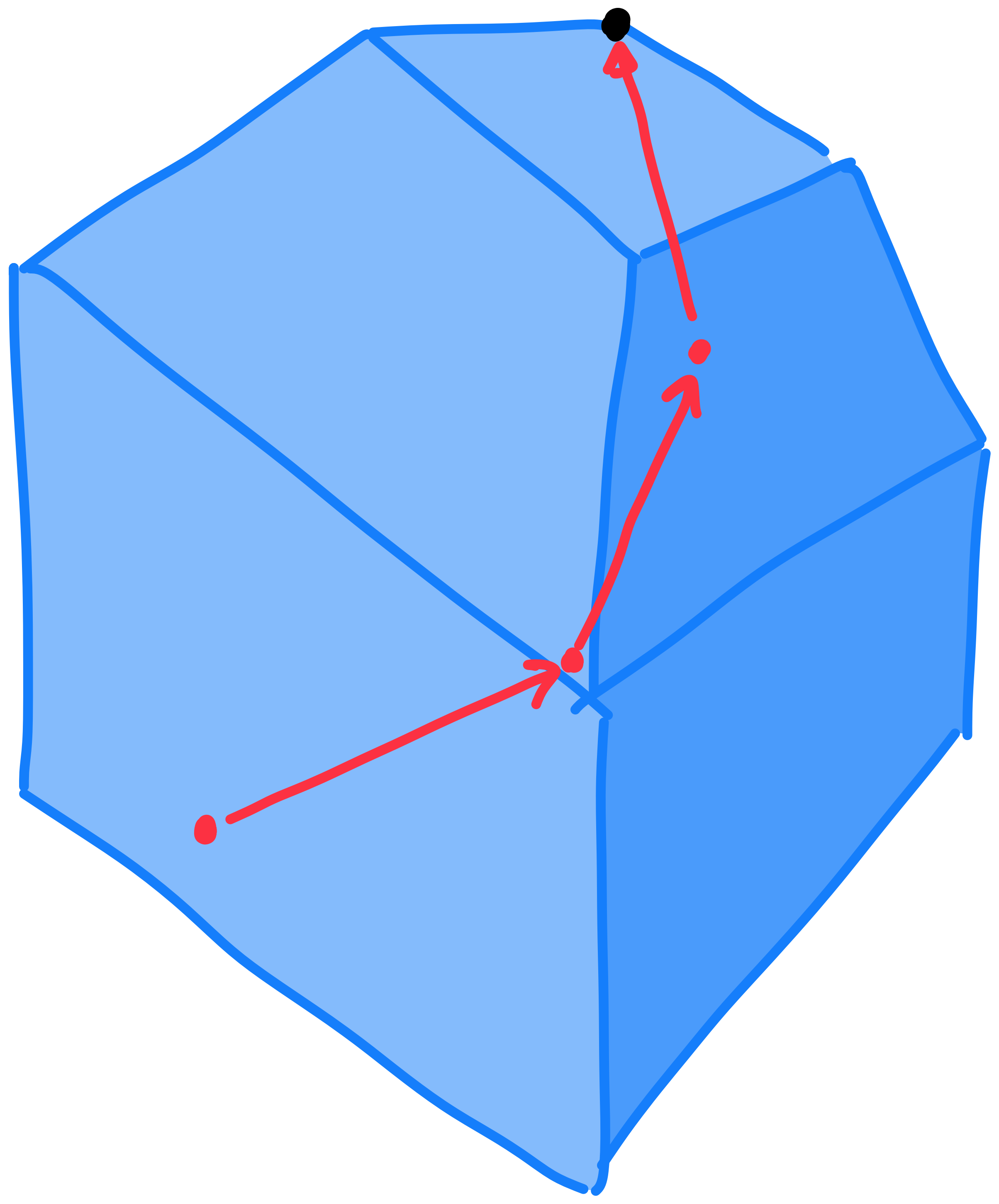
- We have not given runtimes for the simplex method on purpose
  - The runtime can be exponential because the algorithm goes on the *outside* of the polytope which could have lots of vertices, edges, and facets
  - However, simplex runs remarkably well in practice
  - Is there a reconciliation? An algorithm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



Interior point and ellipsoid methods

Interior point

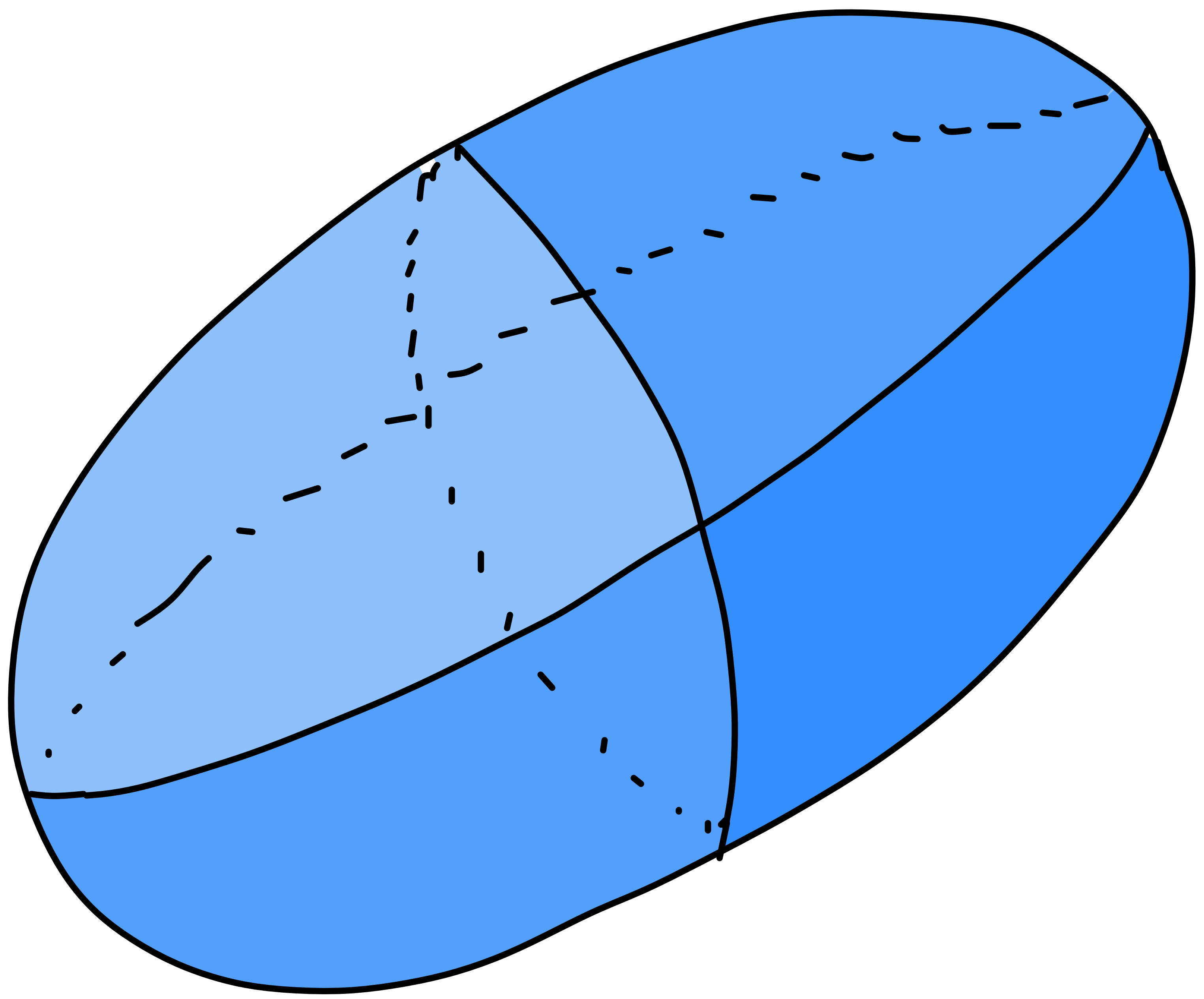
- Keep track of a point *inside* the polytope
- Follow a trajectory through the interior to optimal solution
- Solve a sequence of easier problems to approximate original LP, gradually becoming more accurate
- Runs about as fast as simplex in practice and has guarantees on runtime
- The "state-of-the-art"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



Interior point and ellipsoid methods

Ellipsoid method

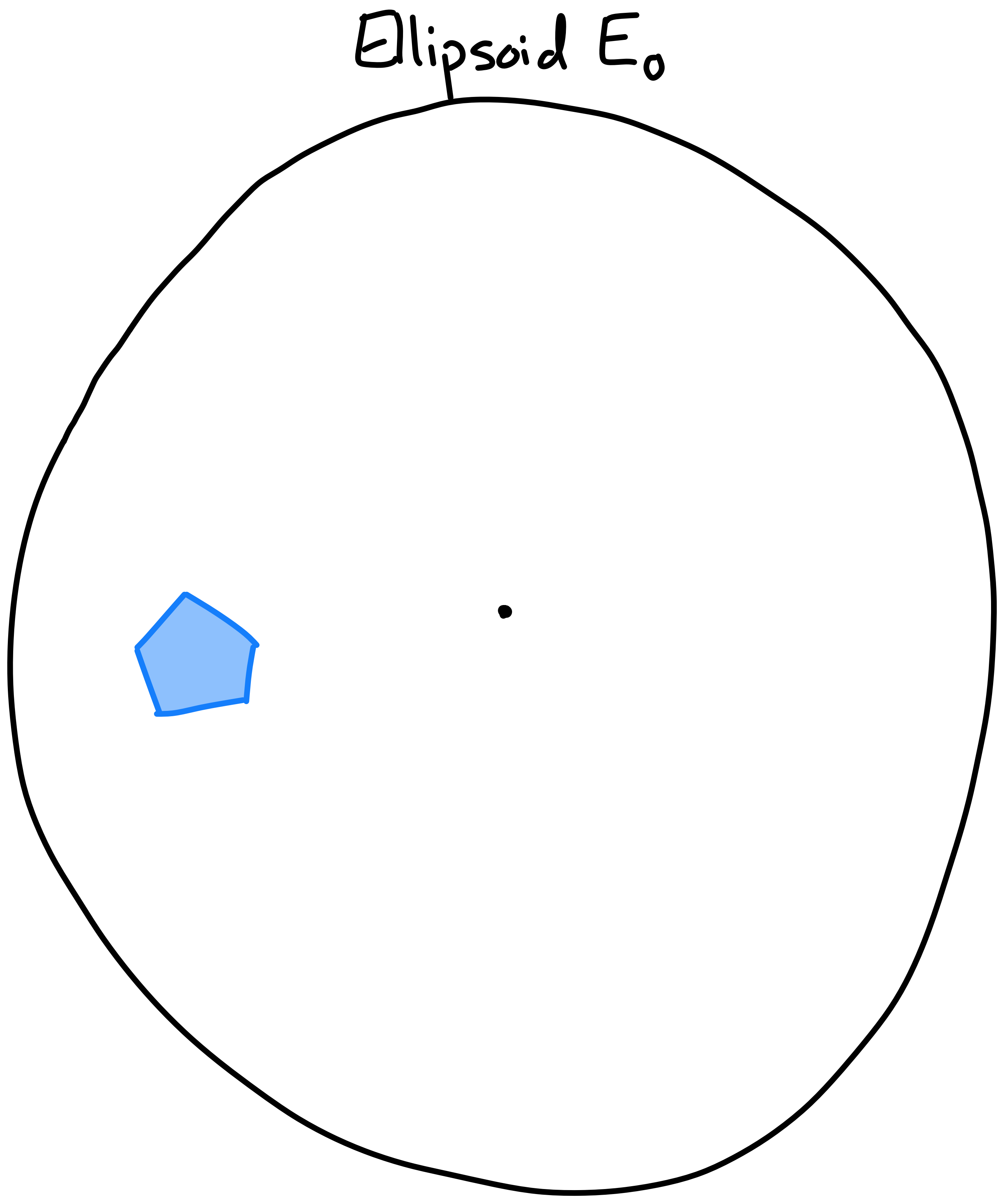
- What is an ellipsoid?
- An ellipsoid is a stretched sphere (in any direction)
- Can be defined by a quadratic equation



Interior point and ellipsoid methods

Ellipsoid method

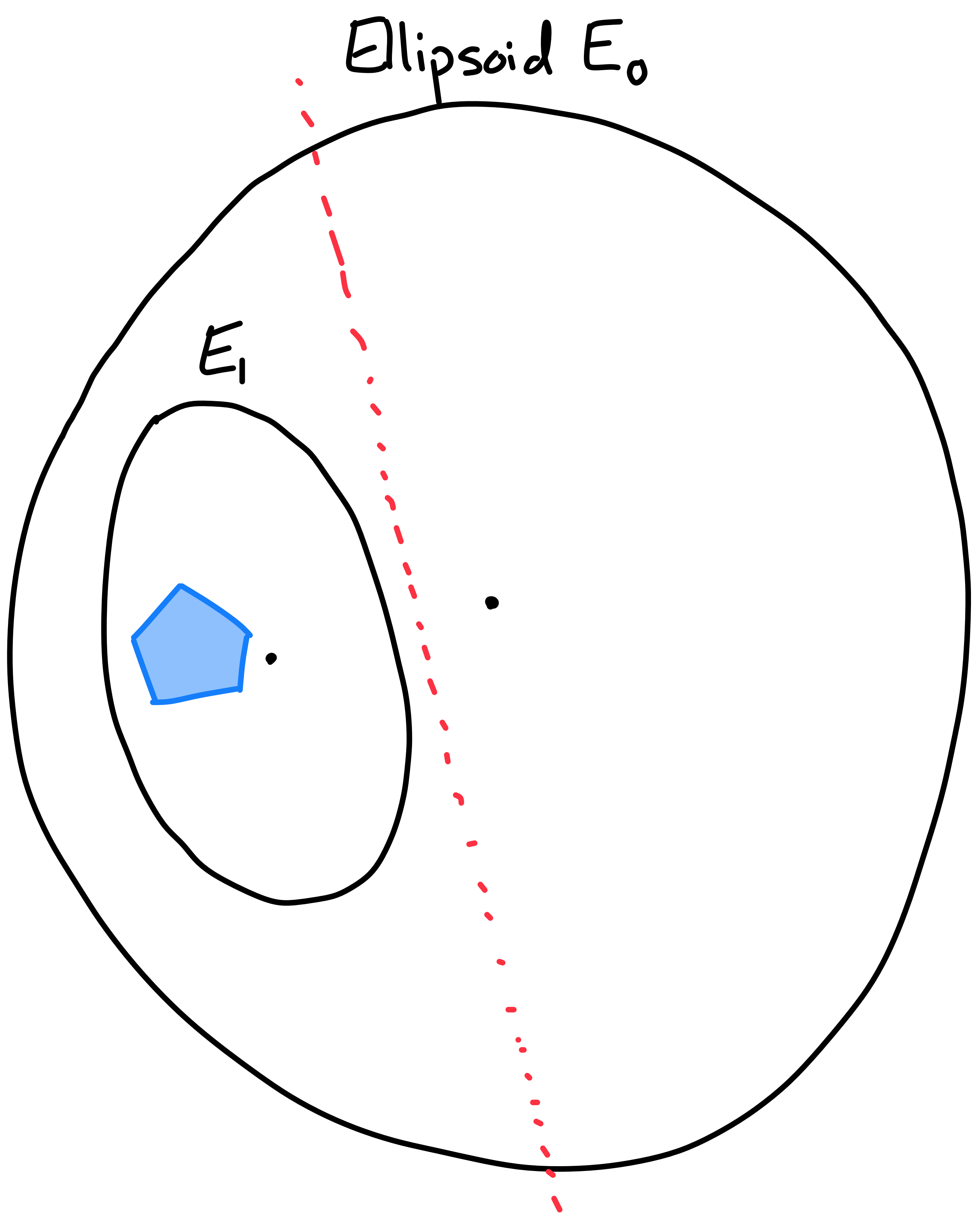
- Using LP duality, convert problem from optimizing a linear polytope to finding a feasible point in a different polytope Γ
- Generate a sequence of ellipsoids that always contain Γ
- Each time find a smaller ellipsoid (by guaranteed ratio) until the center of the ellipsoid must be in Γ
- Very slow in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



Interior point and ellipsoid methods

Ellipsoid method

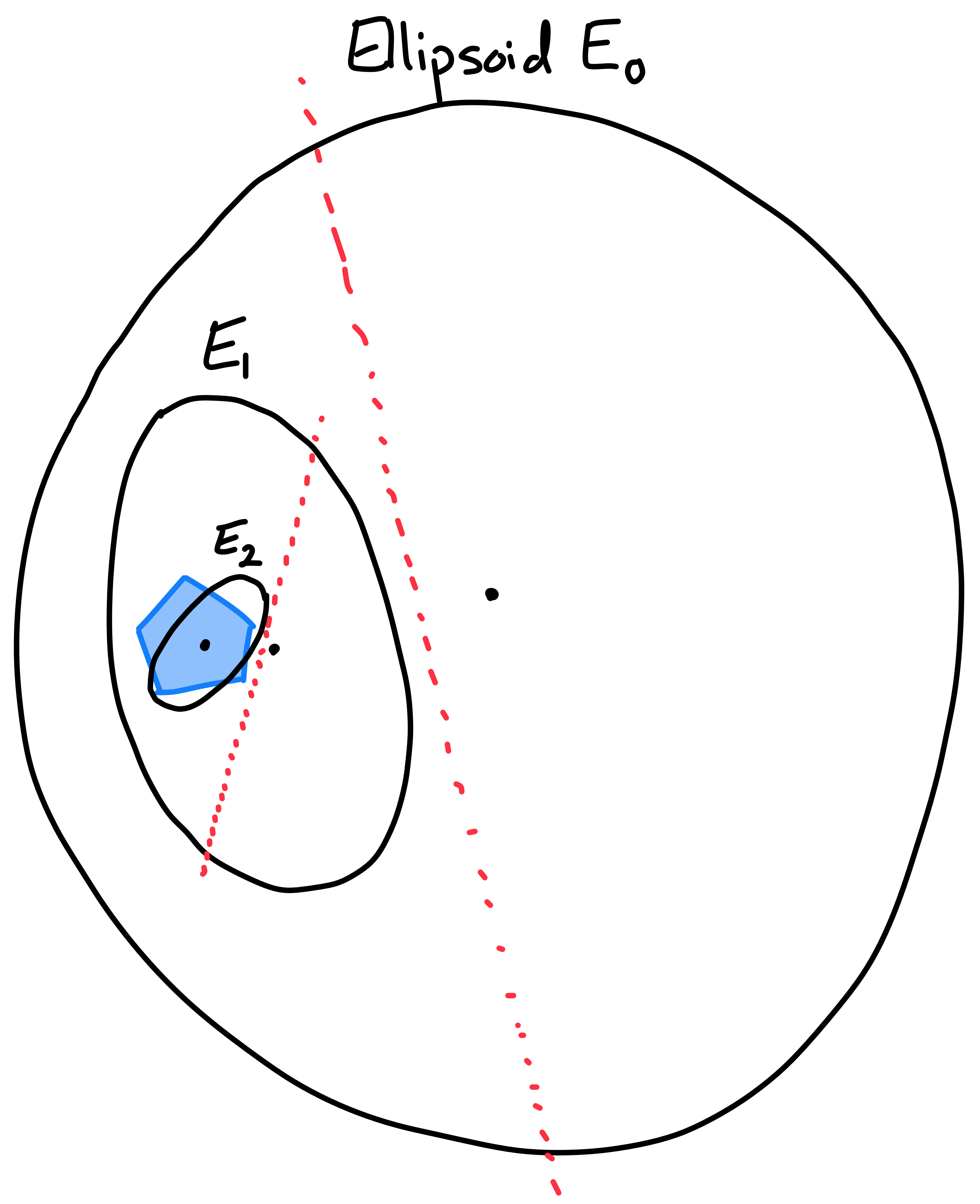
- Using LP duality, convert problem from optimizing a linear polytope to finding a feasible point in a different polytope Γ
- Generate a sequence of ellipsoids that always contain Γ
- Each time find a smaller ellipsoid (by guaranteed ratio) until the center of the ellipsoid must be in Γ
- Very slow in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



Interior point and ellipsoid methods

Ellipsoid method

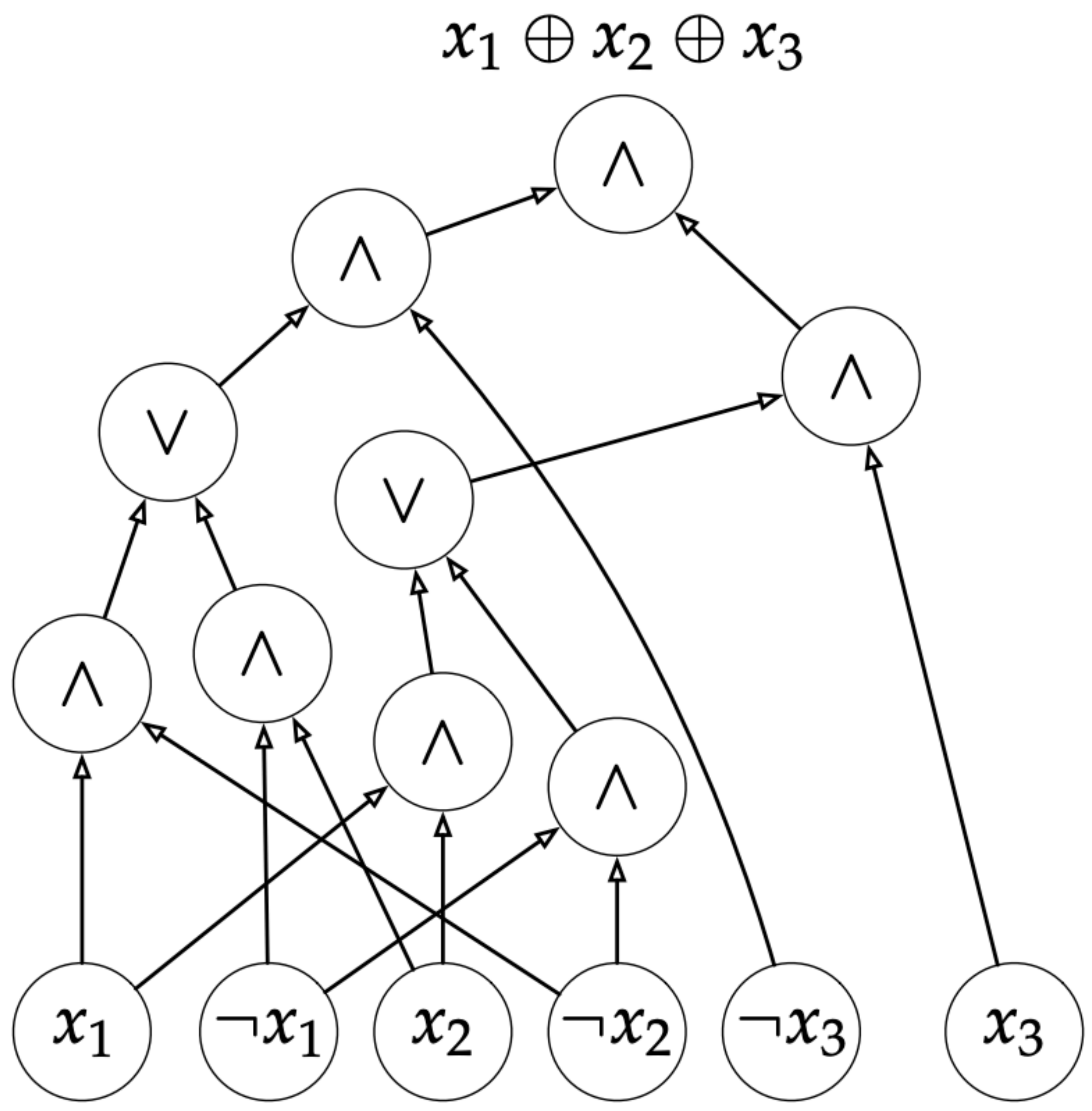
- Using LP duality, convert problem from optimizing a linear polytope to finding a feasible point in a different polytope Γ
- Generate a sequence of ellipsoids that always contain Γ
- Each time find a smaller ellipsoid (by guaranteed ratio) until the center of the ellipsoid must be in Γ
- Very slow in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



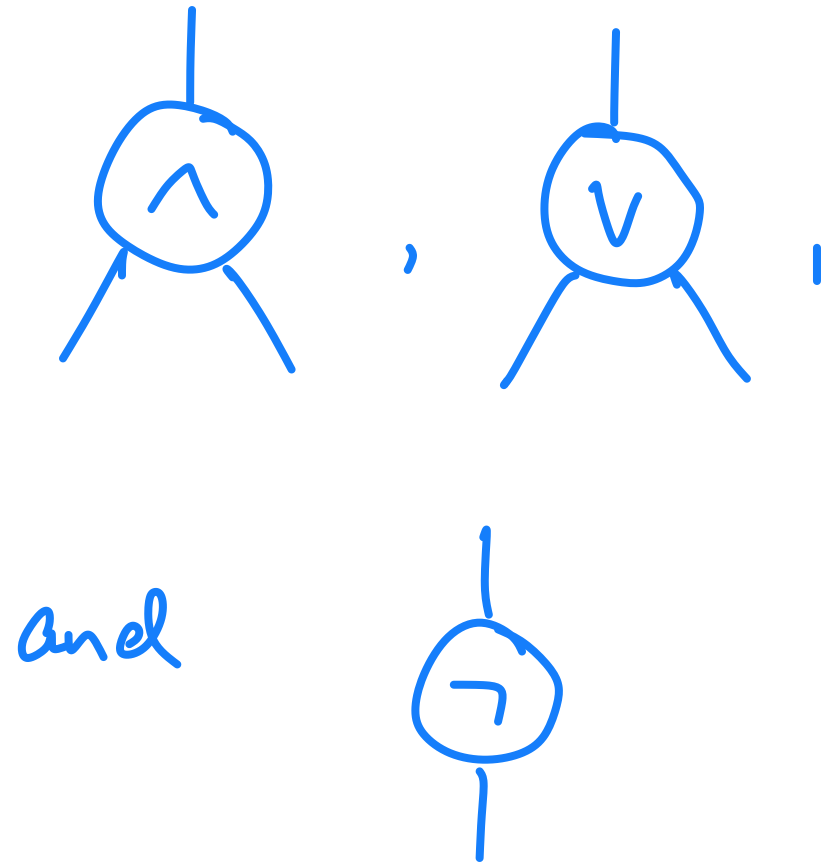
Why is linear programming so important?

- **Fact:** Every boolean function $f: \{0,1\}^n \rightarrow \{0,1\}^n$ that can be computed in time T can be computed by a boolean circuit with $O(T \log T)$ gates.

- **Theorem:** Every boolean function can be expressible as a linear program with $O(T \log T)$ variables and constraints.



Boolean circuits are built from elementary gates:



Why is linear programming so important?

- **Fact:** Every boolean function $f : \{0,1\}^n \to \{0,1\}^n$ that can be computed in time T can be computed by a boolean circuit with $O(T \log T)$ gates.

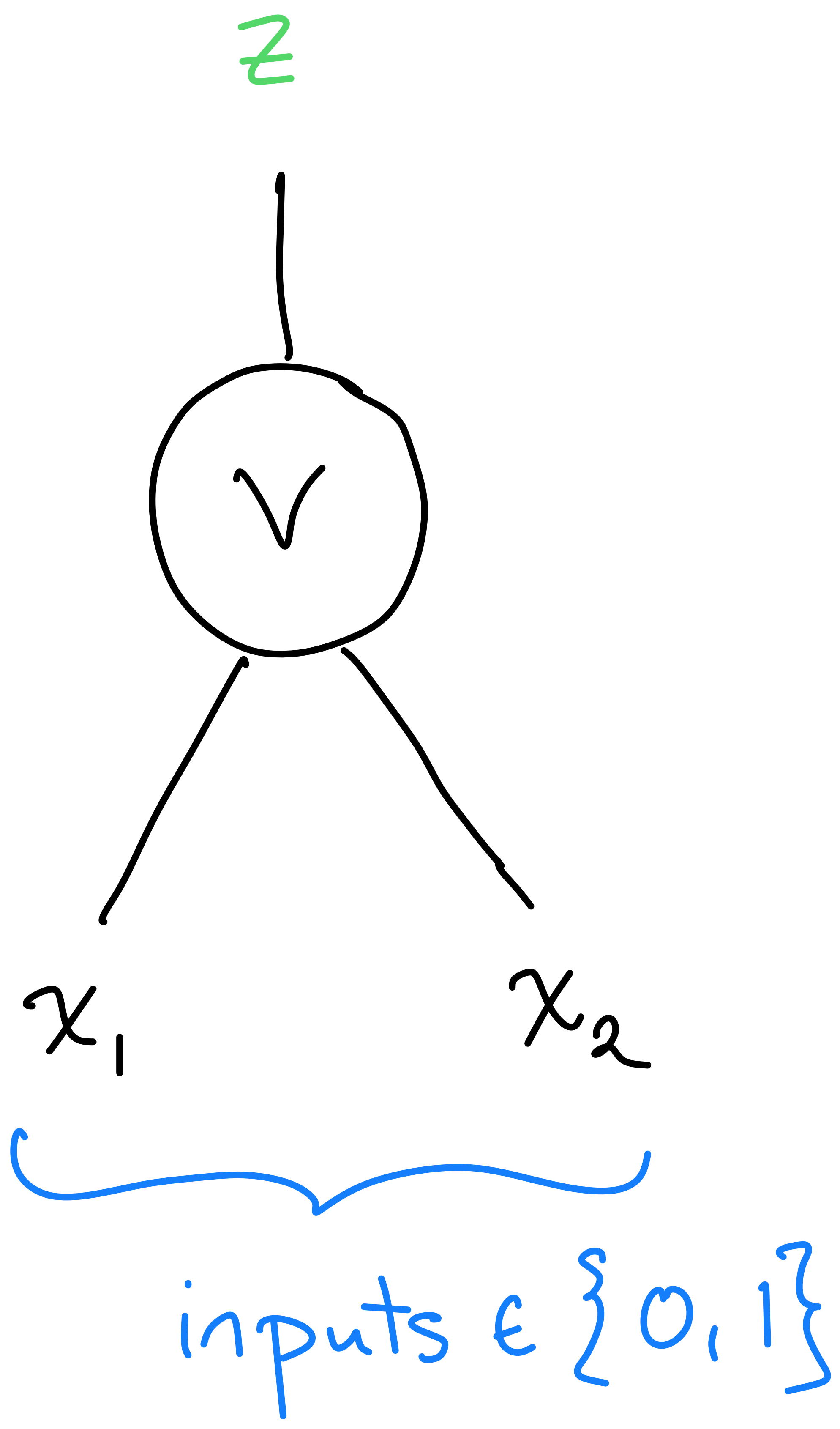
- **Theorem:** Every boolean function can be expressible as a linear program with $O(T \log T)$ variables and constraints.

Boolean circuits are built from elementary gates:

We create linear programming "gadgets" to handle each of the possible gates.

Converting Boolean circuits to LPs

OR gate



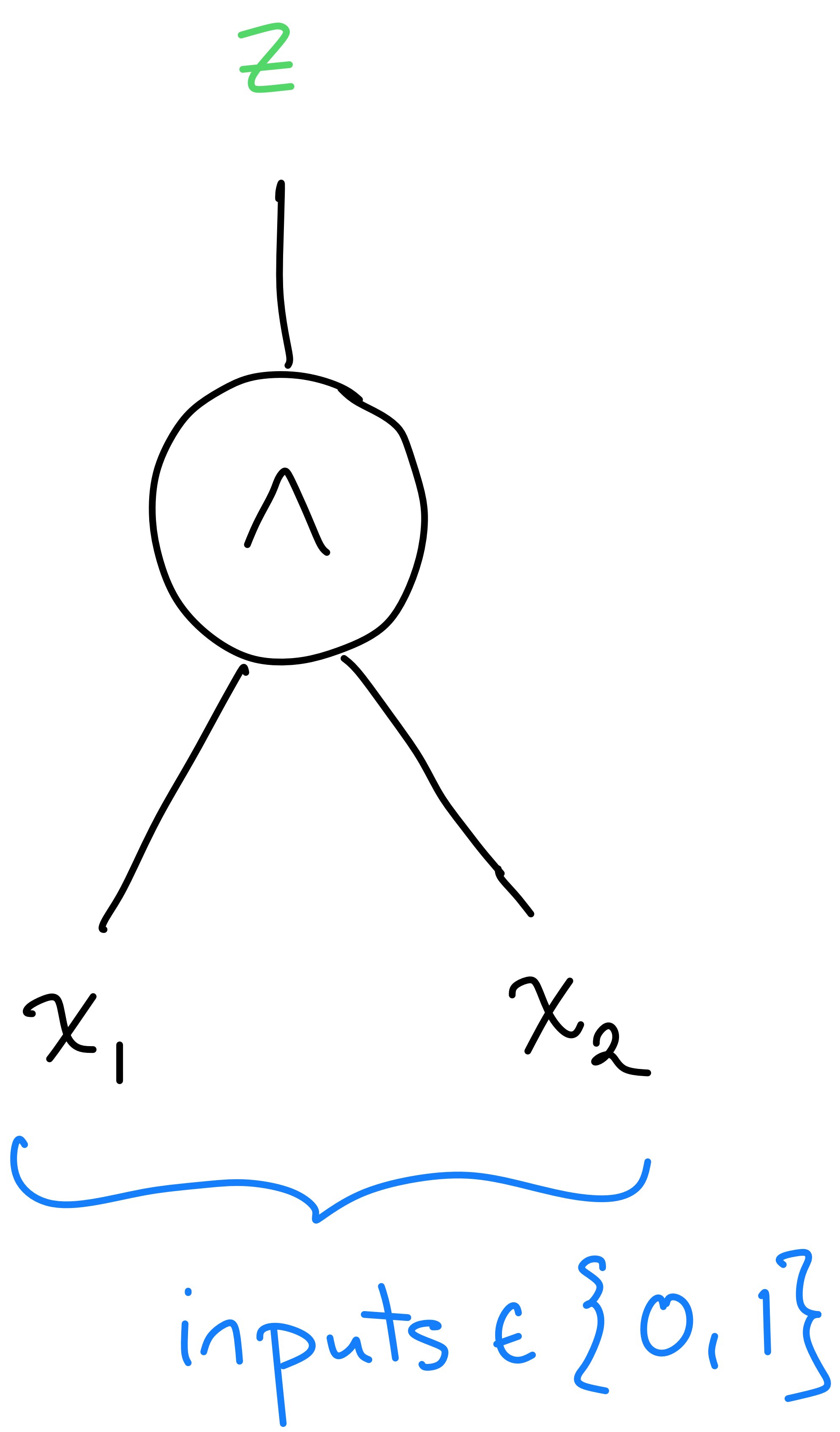
x_1	x_2	z
0	0	0
0	1	1
1	0	1
1	1	1

Observe: $z = \max(x_1, x_2)$

$$= \begin{cases} z \geq x_1 \\ z \geq x_2 \\ z \geq x_1 + x_2 \\ 0 \leq z \leq 1 \end{cases}$$

Converting Boolean circuits to LPs

AND gate



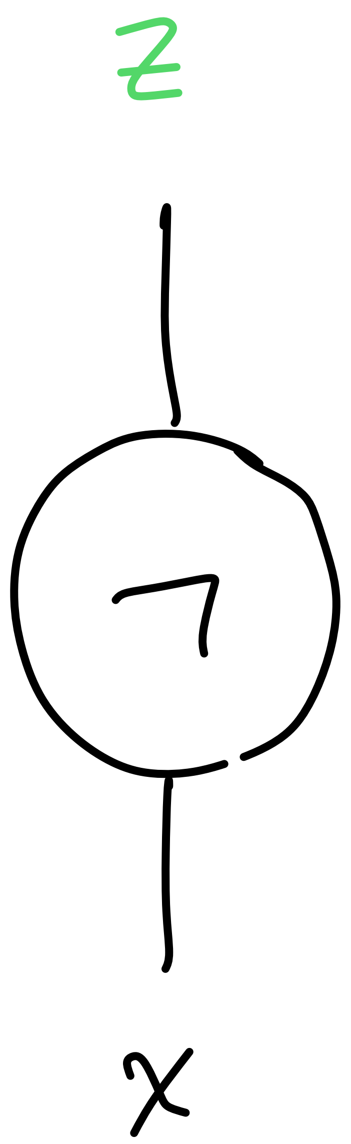
x_1	x_2	z
0	0	0
0	1	0
1	0	0
1	1	1

Observe: $z = \min(x_1, x_2)$

$$= \begin{cases} z \leq x_1 \\ z \leq x_2 \\ z \geq x_1 + x_2 - 1 \\ 0 \leq z \leq 1 \end{cases}$$

Converting Boolean circuits to LPs

NOT gate



input $\in \{0, 1\}$

x	z
0	1
1	0

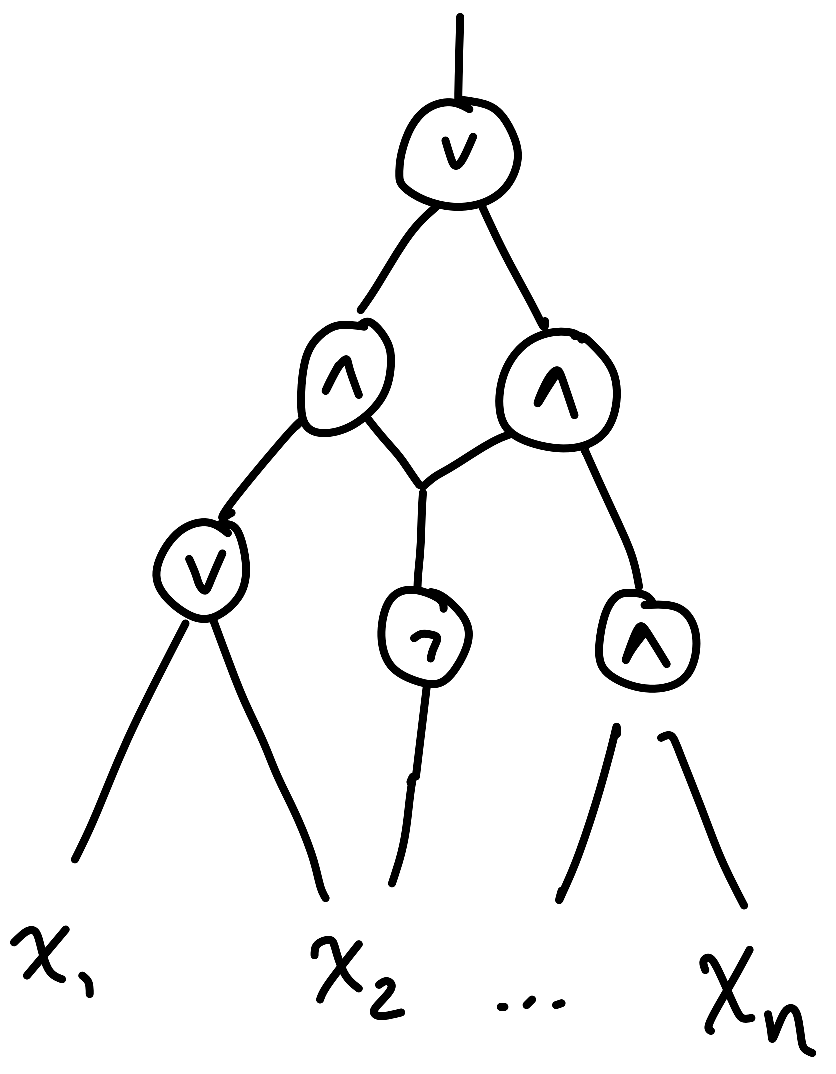
Observe: $z = 1 - x$

$$= \begin{cases} z \geq 1 - x \\ z \geq x - 1 \end{cases}$$

Converting Boolean circuits to LPs

Given the ability to convert an elementary gate to a system of lin. eqs.,

we take the full circuit and create a system of lin. eqs.

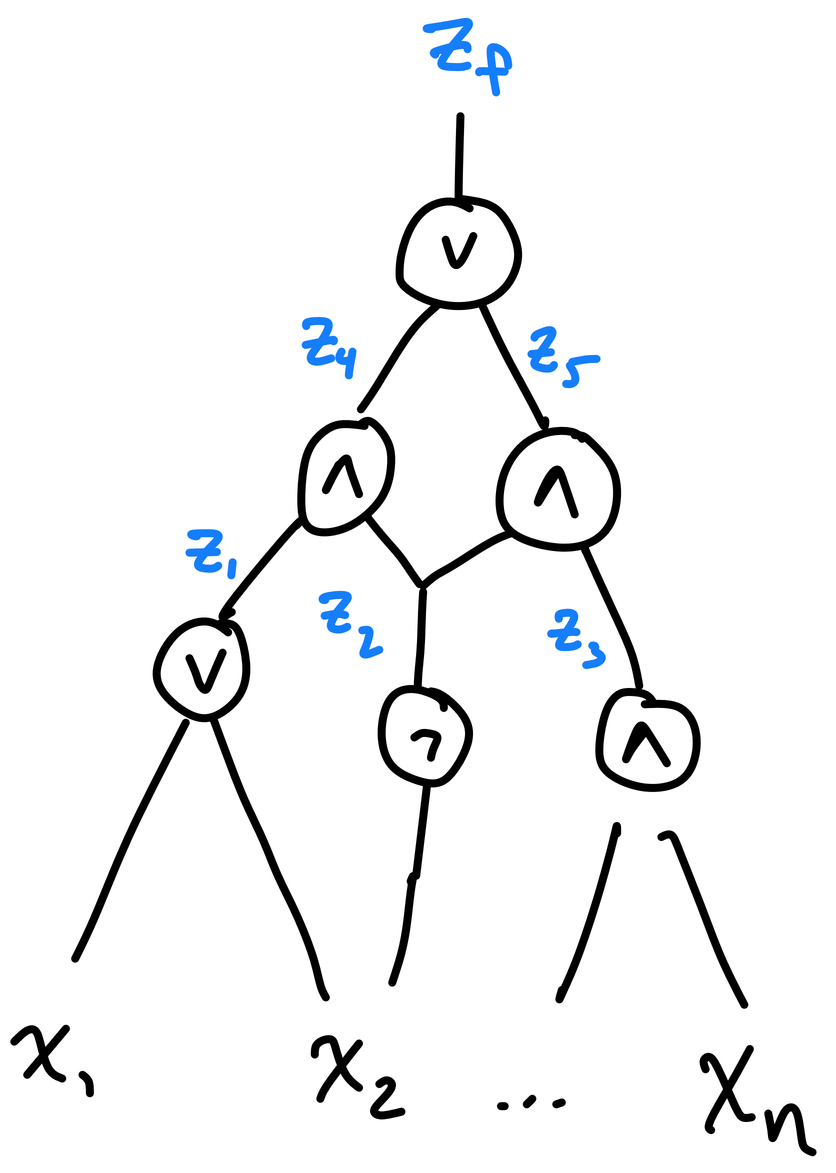


Converting Boolean circuits to LPs

Given the ability to convert an elementary gate to a system of lin. eqs.,

we take the full circuit and create a system of lin. eqs.

Assign a variable for the intermediate "wires".



$$\begin{cases} \max & z_f \\ \text{s.t.} & \forall \text{ gates } g. \ \boxed{\text{eqs about } g} \\ & z \geq 0 \end{cases}$$

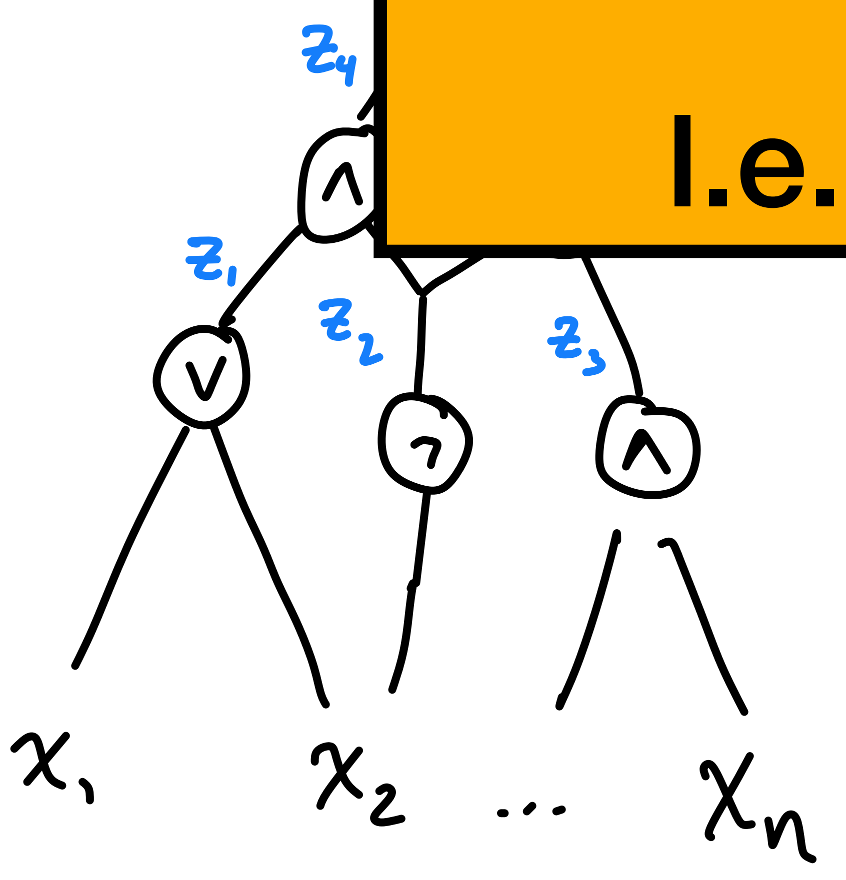
Converting Boolean circuits to LPs

Given the ability to convert an elementary gate to a system of lin. eqs.,

we to

Therefore, every computational problem computable by a boolean circuit of size T can be expressed as a linear program of size $O(T \log T)$.

I.e. linear programming is *universal* for computation



s.t. ∀ gates g. eqs about g

$z \geq 0$