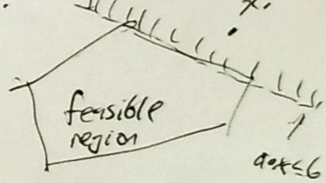
Directed Minimum Spanning Tree.

[We ended last time on the Ellipsoid Algorithm for Linear Programming, which was first used to formally show LP solvable in poly time. Not great in practice, but great for showing — in theory — problems can be solved in poly time (at which point you can have the confidence to work harder improving the polynomial). One feature of Ellipsoid Alg. is that it can be run only using...]

"Separation oracle for an LP": a subroutine that on input $x = (x_1, \ldots, x_n)$ either affirms x satisfies all constraints, or else returns a violated constraint "$a \cdot x \leq b$".
[aka "separating hyperplane"]

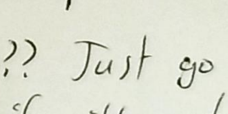


Ellipsoid Alg ⇒
Can solve an n-variable LP in poly(n) time if you can ~~solve~~ make a poly(n)-time separation oracle.

[But what's hard about making a sep. oracle?? Just go thru all constraints one by one, plug in x, see if it's satisfied...? Well, what's cool is you can sometimes do it even if there are exponentially many constraints!]

E.g. task: Min-cost directed ~~MST~~ spanning tree (aka "arborescence")
Given directed graph $G = (V, E)$, "root" vertex $s \in V$, cost c_e for each $e \in E$.

Goal: output a "~~[illegible]~~" of minimal total cost
"directed arborescence" — set of edges so that ∃ path from s to any other vertex



* Assume G strongly connected, so ∃ paths from s to every other vtx.

[I don't insist it makes a "tree", but best solution will...]

[[Edmonds gave a "combinatorial" poly-time alg. for this... but it's a little complicated. Let's see another reason to "know" it's solvable in polynomial time.]]

Idea: Make an LP with one var. x_e per $e \in E$.
"Supposed to be" 1 if e is chosen for the dir MST, 0 else.

[[Note: you cannot add a constraint to LPs saying "this variable should be 0 or 1". Just roll with it for a bit though. We will discuss "integer linear programming" much more in this lecture]]

LP: minimize $\sum_{e \in E} c_e x_e$ ← objective

$x_e \geq 0 \quad \forall e \in E$

s.t. $\forall R \subset V$ with $s \in R$, $\sum_{\substack{e = (u,v) \\ \text{s.t. } u \in R \\ v \notin R}} x_e \geq 1$. ← constraints

[[We'll talk about meaning of constraints soon, but first note...]]

\# of constraints is exponential! $\underbrace{m + 2^{n-1} - 1}$ [[one for every subset of $V \setminus \{r\}$ except the whole set]]

[[So don't just generate them all to hand to an LP solver — that takes expon. time!]] (*) → [[jump ahead in notes]]

Lucky fact, specific to this LP: ☺
Every vertex (corner) $\hat{x}$ of this feasible region is "integral": $\hat{x}_e \in \{0,1\} \quad \forall e$!

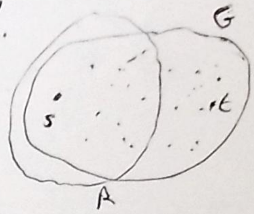
[[A mild miracle which doesn't always happen!]]
Won't prove this, but will prove it for another, similar LP where the same miracle happens, later.
[[So just take my word for it now.]]

Great! So LP solver can use separation oracle to find min-cost directed spanning tree in poly time!

* Constraint with R looks familiar:

Think of $x \geq 0$ as defining capacities in a flow network.

For any $t \in V \setminus R$, R is an "st-cut".



Constraint says "capacity$(R) \geq 1$."

[[kind of like saying a feasible solution x should be able to support a unit of flow across R]]

Separation oracle for this LP:

- On input x:
  - check if $x_e \geq 0$ $\forall e$ // $O(m)$ time
    If not, return violated constr.
  - Else // now we know $x_e \geq 0$ $\forall e$
    - for each $t \in V \setminus \{s\}$
      - Solve Min-st-Cut problem, // poly(m) time using flow algs; even nearly-linear time using fanciest known one.
      - Say R^* is Min-st-Cut.
        If $\text{cap}(R^*) < 1$, return "$\text{cap}(R^*) \geq 1$" as violated constraint!
    - // if here, we know capacity ≥ 1 for any cut between r & rest of vertices, so...
    return "x is feasible"

$\boxed{\text{Key}}$: We can give a poly-time separation oracle for this LP [[using our knowledge of min-cut algs]] despite expon many constrs.

$\Rightarrow$ Can solve LP in poly time using Ellipsoid Alg!

[[Now back to ☺ in notes]]

[[OK, but why study this LP? What does it have to do with Min-cost dir. spanning tree problem?]]

Fact 1: Say $T \subseteq E$ is a directed arborescence from s. Define x by

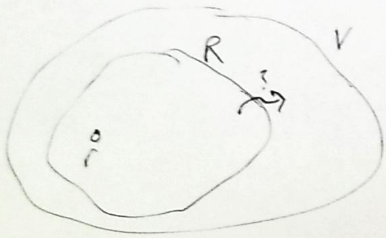
$$x_e = \begin{cases} 1 & \text{if } e \in T \\ 0 & \text{if } e \notin T. \end{cases}$$

Then x is feasible. [[satisfies all constraints]]

Proof: $x_e \geq 0 \ \forall e$ ✓

And fix any $R \subset V$, $R \ni r$. Need to show $\sum_{\substack{uv \in E \\ u \in R, v \notin R}} x_{uv} \geq 1$.

strict



equals # edges of T going from in R to not in R

Better be ≥ 1, else T doesn't connect r to the vertices of $V \setminus R$. □

Fact 2: Say $\hat{x}$ is feasible.

Say by some miracle we also have $\hat{x}_e \in \{0,1\} \ \forall e$.

Then, letting $T = \{e : \hat{x}_e = 1\}$, T is a directed arborescence from s.

Proof: Need to show T has an s-t path $\forall t \in V$. Fix any t.

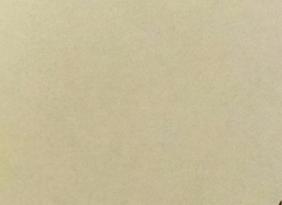
Consider constr. with $R = \{r\}$. $\sum_{\substack{uv \in E \\ u \in R, v \notin R}} x_{uv} \geq 1$

$= \sum_{rv \in E} x_{rv} \geq 1 \Rightarrow$ T has at least one edge out of r, say to v_1.

Now consider $R = \{r, v_1\}$.

Constr. $\Rightarrow$ T has ≥ 1 edge out of $\{r, v_1\}$, say to v_2

Constr. on $R = \{r, v_1, v_2\} \Rightarrow$ T has at least one edge out to some v_3.



Keep going.... All of $v_1, v_2, v_3, \ldots$ reachable from r in T.

Eventually will hit t!

[[Now back to ☺ □ in notes]]