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... A

"Separation oracle for an LP": a subroutine that on input $x = (x_1, ..., x_n)$ 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 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! II

E.g. task: Min-cost directed $\text{spanning tree}$ (aka "arborescence")

Given directed graph $G = (V, E)$, "root" vertex $s \in V$.

Goal: output a "set of edges" of minimal total cost so that $\exists$ path from $i$ to any other vertex $v$.

*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 $\sum_{e \in E} C_e x_e$  \hspace{1cm} \text{objective} \hspace{1cm} x_e \geq 0 \forall e \in E$
\begin{align*}
\text{s.t.} \quad & x_e \quad \text{with } s \in R, \\
& \forall e \in V, \\
& \forall R \subseteq V
\end{align*}
\hspace{1cm} \text{constraints}, \quad \sum_{e \in (uv)} x_e \geq 1.

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

# of constraints is exponential! $m + 2^{n-1} - 1$.

So don't just generate them all to hand to an LP solver — that takes exp. time! 

Lucky fact: specific to this LP:

Every vertex (corner) $\hat{x}$ of this feasible region is "integral": $\hat{x}_e \in \{0,1\} \forall e$.

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 $e \in V \setminus R$, $R$ is an "st-cut".
Constraint says "capacity$(R) \geq 1$." (kind of like saying a feasible solution $x$ should support a unit of flow across $R$)

Separation oracle for this LP:
- On input $x$:
  - Check if $x \geq 0$ hold. O($m$) time
    - If not, return violated constr.
  - Else, now we know $x \geq 0$ hold.
    - For each $e \in V \setminus R$,
      - Solve Min-st-Cut problem. O($\text{poly}(m)$) time using flow alg.
      - Say $R^*$ is Min-st-Cut.
      - If $\text{cap}(R^*) < 1$, return "capacity$(R^*) \geq 1" as violated constraint!

// if here, we know capacity $\geq 1$ for any cut between $R$ & rest of vertices, so...
return "$x$ is feasible".

\textbf{Key: } We can give a poly-time separation oracle for this LP (using our knowledge of min-cut alg's) despite many constrs.

$\Rightarrow$ Can solve LP in poly time using Ellipsoid Alg!
OK, but why study this LP? What does it have to do with Min-cost dir. spanning tree problem? II

**Fact 1** Say $T \subseteq E$ is a directed forest. 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. It satisfies all constraints.

**Proof:** $x_e \geq 0 \forall e \in V$.

And fix any $R \subseteq V$, $R \neq \emptyset$. Need to show $\sum_{u \in R, v \notin R} x_{uv} \geq 1$.

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

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

**Fact 2** Say $x$ is feasible.

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

Then, letting $T = \{ e : x_e = 1 \}$, $T$ is a directed forest.

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

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

$= \sum_{u \in E} x_{uv} \geq 1$ $\Rightarrow$ $T$ has at least one edge out of $R$, say to $V_1$.

Now consider $R = \{ v_1, v_3 \}$.

Constr. $\Rightarrow$ $T$ has $\geq 1$ edge out of $\{ v_1, v_3 \}$, say to $v_2$.

Constr. on $R = \{ v_1, v_2, v_3 \}$ $\Rightarrow$ $T$ 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 $\square$ in notes.