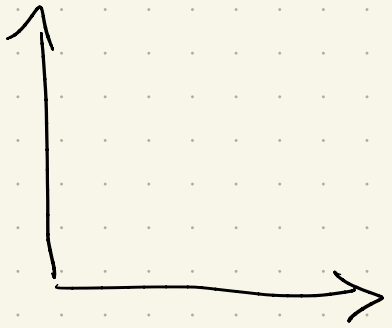
Complexity & Algorithms, Spring 2026

LP: Simplex,
& approximation

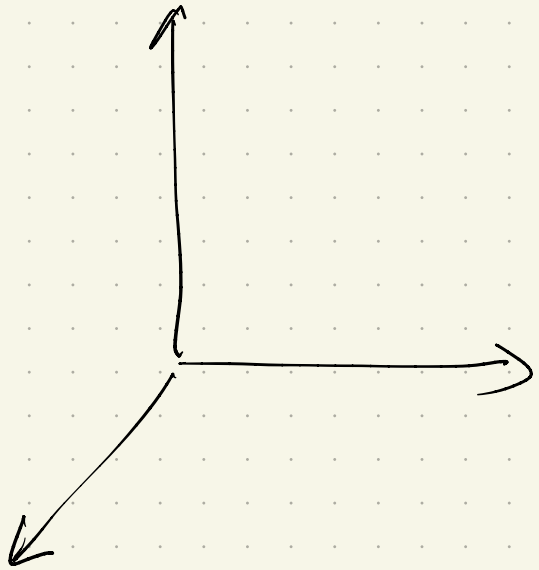
Simplex Setup

Each LP equality or inequality describes a hyperplane in $\mathbb{R}^d$.

2d: $ax+by \leq c$



3d: $ax+by+cz \leq d$



$\mathbb{R}^d$: $a_1x_1+a_2x_2+\cdots+a_dx_d \leq c$

What is our input?

Recall:

$$\text{maximize} \sum_{j=1}^{d} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{d} a_{ij} x_j \leq b_i \quad \text{for each } i = 1..n$$

$$x_j \geq 0 \quad \text{for each } j = 1..d$$

So $d =$

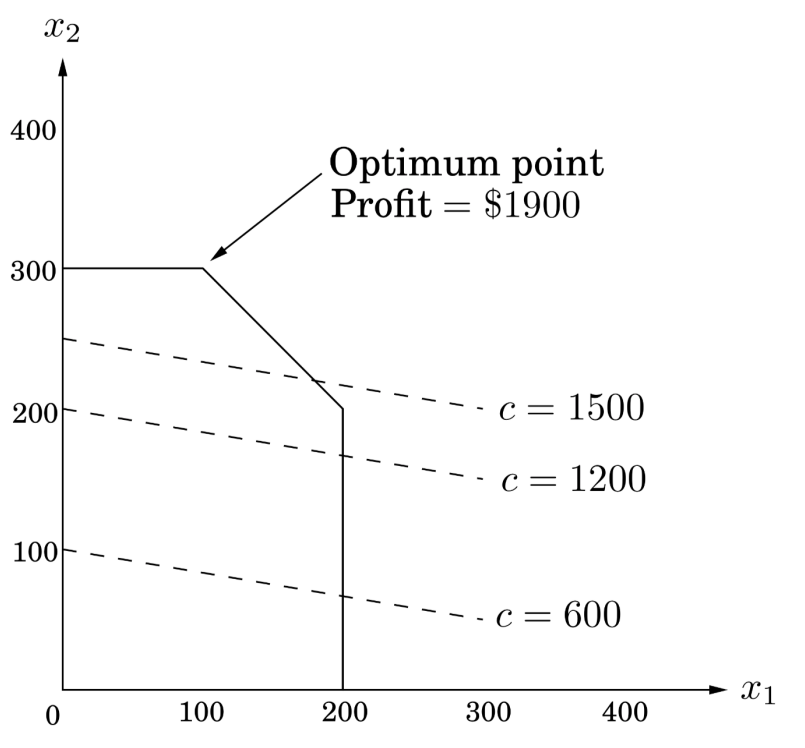
$n =$

In our groceries/nutriton example:

Vertices:

These happen when $> d$ hyperplanes meet in $\mathbb{R}^d$.

In $\mathbb{R}^2$ a

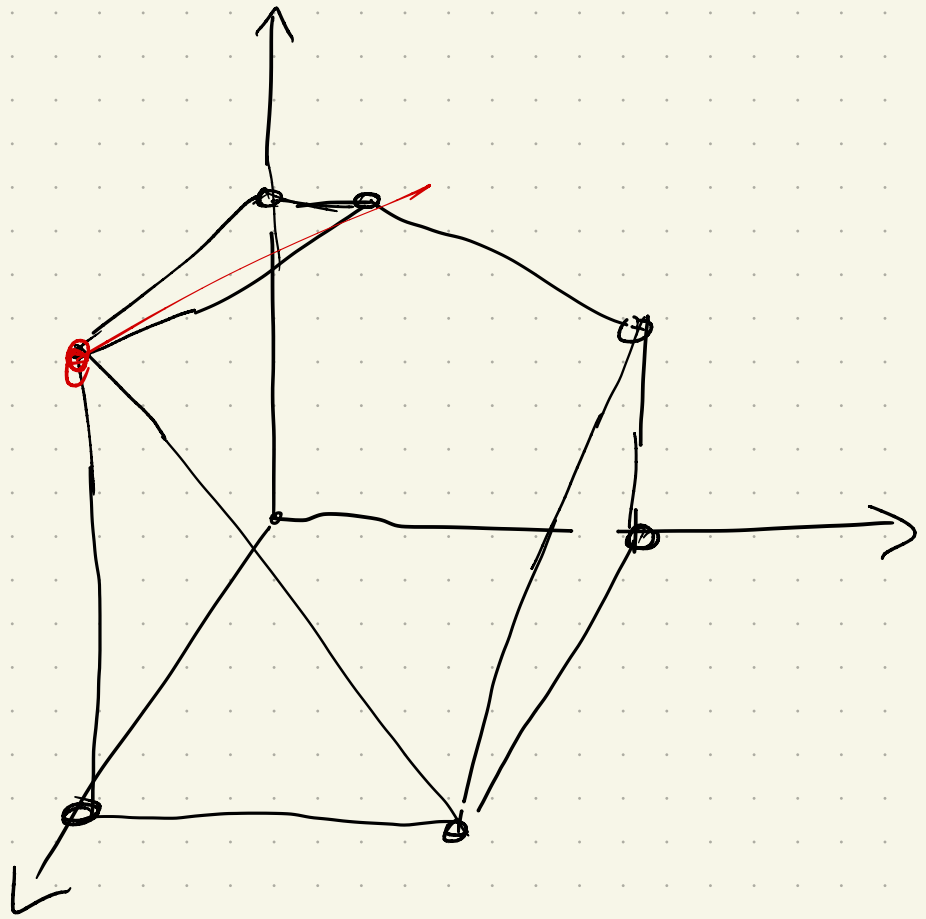


Note:
Not every pair will meet!

In $\mathbb{R}^3$

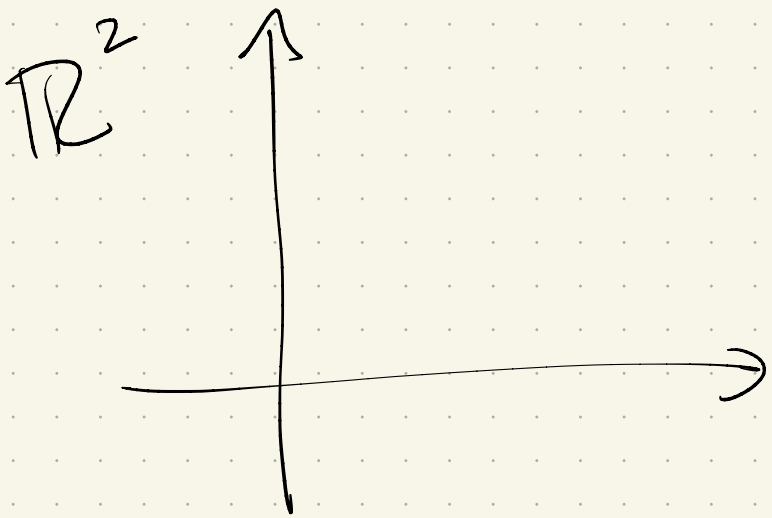
Any 2 intersect in a line.

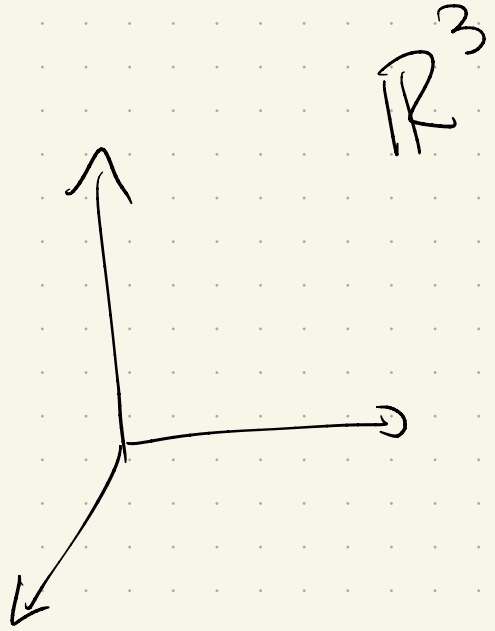
Any 3 intersect in a vertex (assuming not parallel)



In $\mathbb{R}^d$: d equations, d variables $\Rightarrow$ one point

Aside: When don't they intersect, or when do more than d intersect?



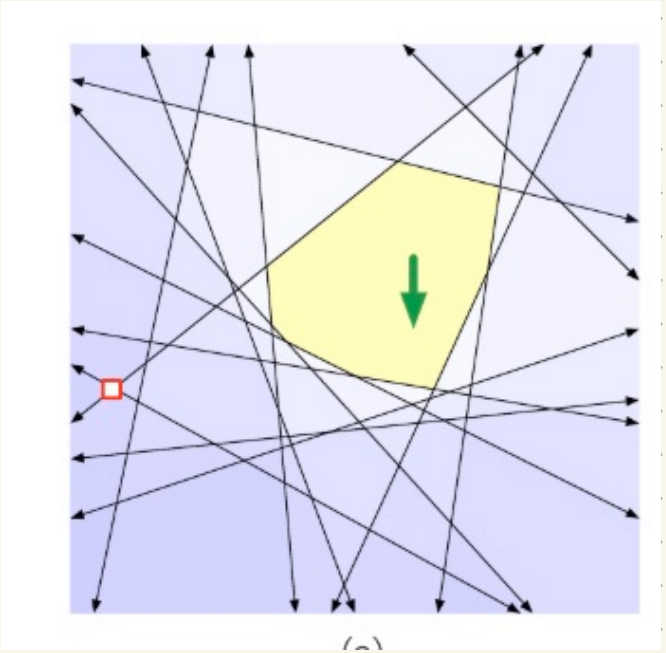


Dfn: Pick a subset of inequalities

If there is a unique point that satisfies all with equality, & it is feasible

↳ this is a vertex of the solution

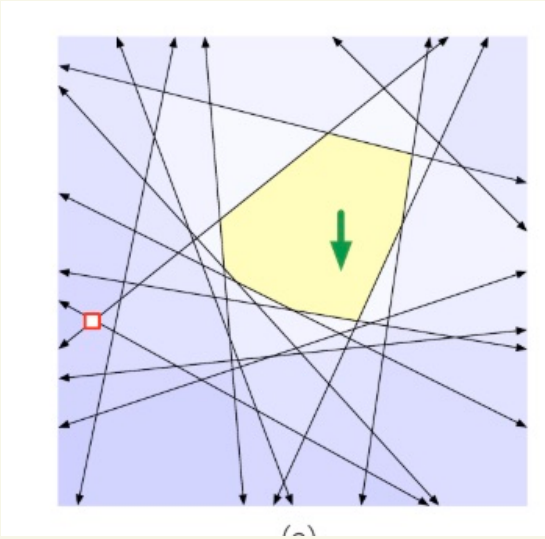
When do we not hit a vertex?



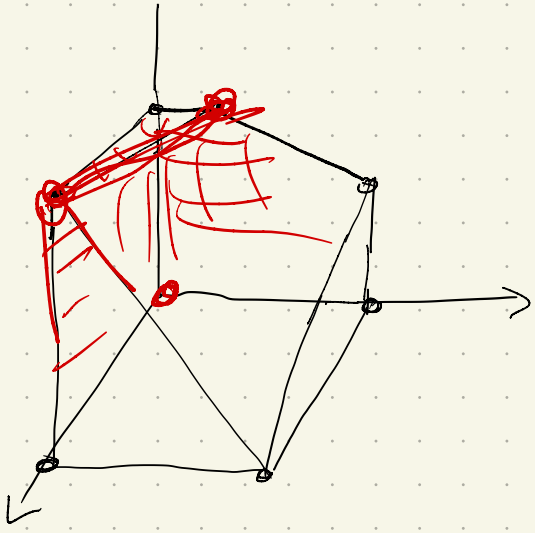
Neighbors:

Since each vertex is specified by d equations, we call any two that share $d-1$ of them neighbors:

2d:



3d:



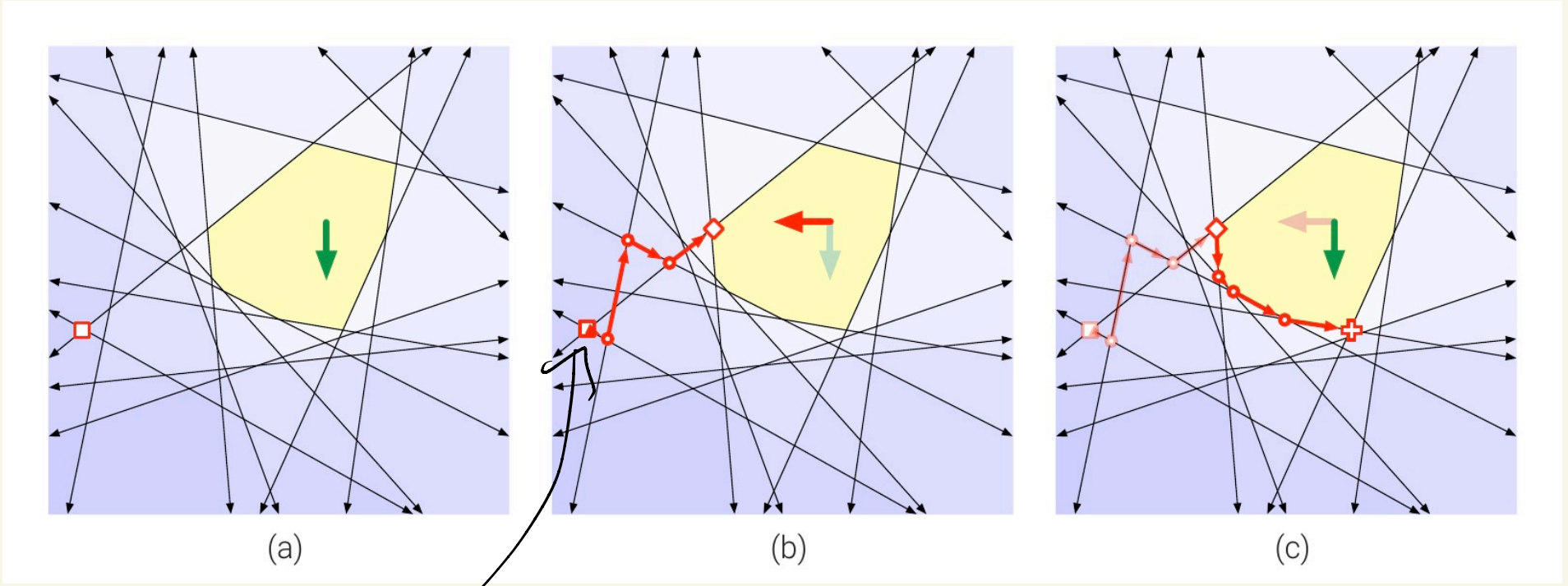
Simplex algorithm:

Find some vertex in the feasible region.

1. Check if current vertex is optimal
2. If not choose a neighbor that gives a better score under the objective function, & return to 1.

Finding a feasible vertex:

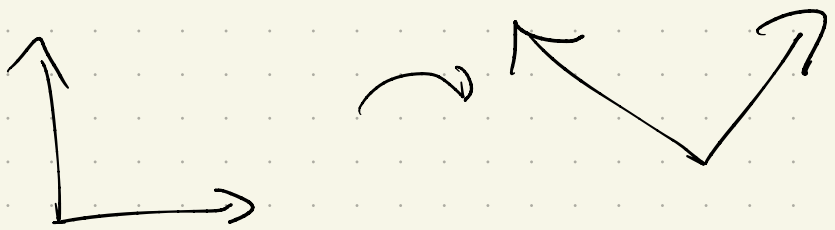
Use another LP! (a dual one)



Note: in duals we're already feasible.

How do we translate / rotate?

Rotation: a linear change of basis



Pick Q sending 1 basis to the other.

Q is an invertible matrix: $QQ^T = I$

Then $x = Qy \Rightarrow$

$$\max c^T x \iff$$
$$\text{s.t. } Ax \leq b$$

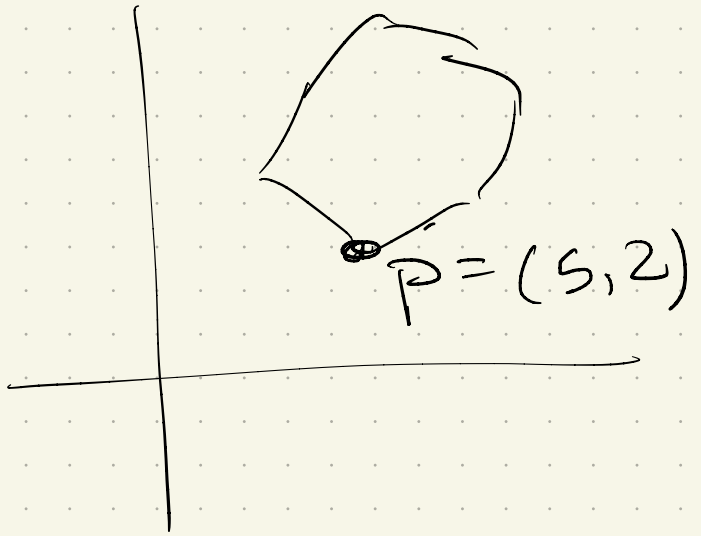
New LP: $\max (Q^T c)^T y$

$$\text{s.t. } (AQ)y \leq b$$

Translate: move (a,b) over in $\mathbb{R}^2$

For each x_i, let

$$y_i = x_i - p$$



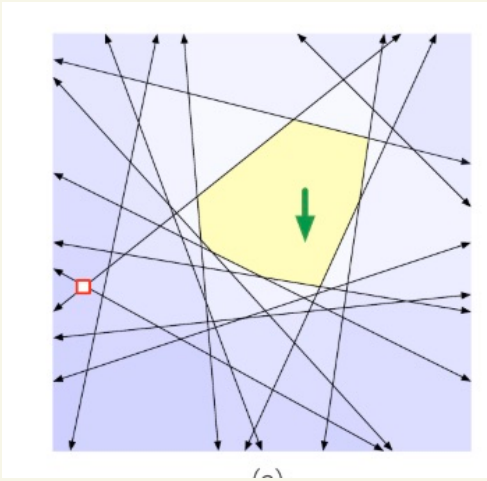
$$Ax \leq b \implies A(y+p) \leq b$$

$$\Rightarrow$$

Back to primal:

Now we have a feasible vertex.

How to iteratively make it better?



Runtime:

Consider a vertex $\vec{v} \in \mathbb{R}^d$ with m inequalities & d variables

How many possible neighbors?

each removes 1 eqn & replaces with another

$\Rightarrow \leq d \cdot (m-d)$

Checking if it is a neighbor & is feasible: basically dot product & Guassian elimination.

So: each iteration is

$$\leq d \cdot (m-d) \cdot [\text{time for G.E}]$$

$$[\text{time for G.E}] = n^{\omega}$$

Can improve slightly:

- just need one $c_i > 0$ + rescaling to $\vec{0}$ is easy.

So can get to $O(m \cdot n)$

How many iterations?

{m+d inequalities
any d give a vertex} $\Longrightarrow$ $\binom{m+d}{d}$

$\leq (m+d)^d$

So exponential in worst case.

(Remember, for a while people thought this might be NP-Hard)

Klee & Minty gave examples in the 70's that actually take exponential time.

Can we avoid this by choosing the "best" neighbor in our update?

No ideal way!

Many proposed, but for almost every one, there is some input polyhedron that needs an exponential number of pivots.

Reading #2: Approximation

First: Integer linear programming

Maximize $\vec{c}^T \vec{x}$

s.t. $A\vec{x} \leq b$

$\vec{x} \geq 0$

and $x_i \in \mathbb{Z}$ (not $\mathbb{R}$)

Variants:
- Mixed
- Binary

This is NP-Hard

Reduce From 3SAT:

Using this for approximation:

- Build an ILP for a problem.
- "Relax" into an LP
  - ↳ fractional answers
- Use these fractions somehow.

Vertex Cover:

- Each $v \in V$ has a weight c_v
- Goal: "Cover" edges using minimum weight possible.

$$\begin{array}{lll} \min & \sum_{v \in V} c_v x_v & \\ \text{such that} & x_v \in \{0,1\} & \forall v \in V \\ & x_v + x_u \geq 1 & \forall vu \in E. \end{array}$$

Relax:

Now what?

Each v has a value between 0 and 1.

For each edge uv, $x_u + x_v \geq 1$.

Could flip coins

↳ but would we guarantee a cover?

Instead:

Note: For each edge, some endpoint is $\geq \frac{1}{2}$.

So → choose all v with $x_i \geq \frac{1}{2}$.

Cost? We know our LP cost $\alpha = \sum_v c_v x_v$ is $\leq$ opt.

Why?

Now, for each vertex chosen,

$$x_v \geq \frac{1}{2}, \quad \text{so} \quad 1 \leq 2x_v.$$

Then multiply by v's weight:

$$c_v \leq 2c_v x_v$$

Sum over all v's we chose:

$$\sum_{v \in S} c_v \leq \sum_{v \in S} 2c_v x_v$$

$$\leq 2 \cdot \sum_{\text{all } v} c_v x_v$$

Set Cover:

Set Cover

Instance: $(S, \mathcal{F})$
S - a set of n elements
$\mathcal{F}$ - a family of subsets of S, s.t. $\bigcup_{X \in \mathcal{F}} X = S$.
Question: The set $\mathcal{X} \subseteq F$ such that $\mathcal{X}$ contains as few sets as possible, and $\mathcal{X}$ covers S.

ILP:

$$\begin{aligned} \min \quad & \alpha = \sum_{U \in \mathcal{F}} x_U, & \\ \text{s.t.} \quad & x_U \in \{0,1\} & \forall U \in \mathcal{F}, \\ & \sum_{U \in \mathcal{F}, s \in U} x_U \geq 1 & \forall s \in S. \end{aligned}$$

relax!

Then: Include set U with probability $X_U \rightarrow$ flip a biased coin!

Expected cost:

Problem: If we include with some probability, might not cover everything.

So: repeat!

Each time, expected cost is $\leq OPT$.

Repeat $\log n$ times:

$$\Pr[s \text{ not covered in a round}]$$
$$= P[\text{no } U \text{ containing } s \text{ was picked}]$$
$$= \prod_{U \text{ containing } s} (1 - x_U)$$
$$\leq \prod e^{-x_U} = e^{-(\sum x_U)} \leq e^{-1}$$

Result:

With high probability, after $\approx \log n$ rounds, we cover everyone.

Cost: $\leq$ (# rounds) · (cost per round)