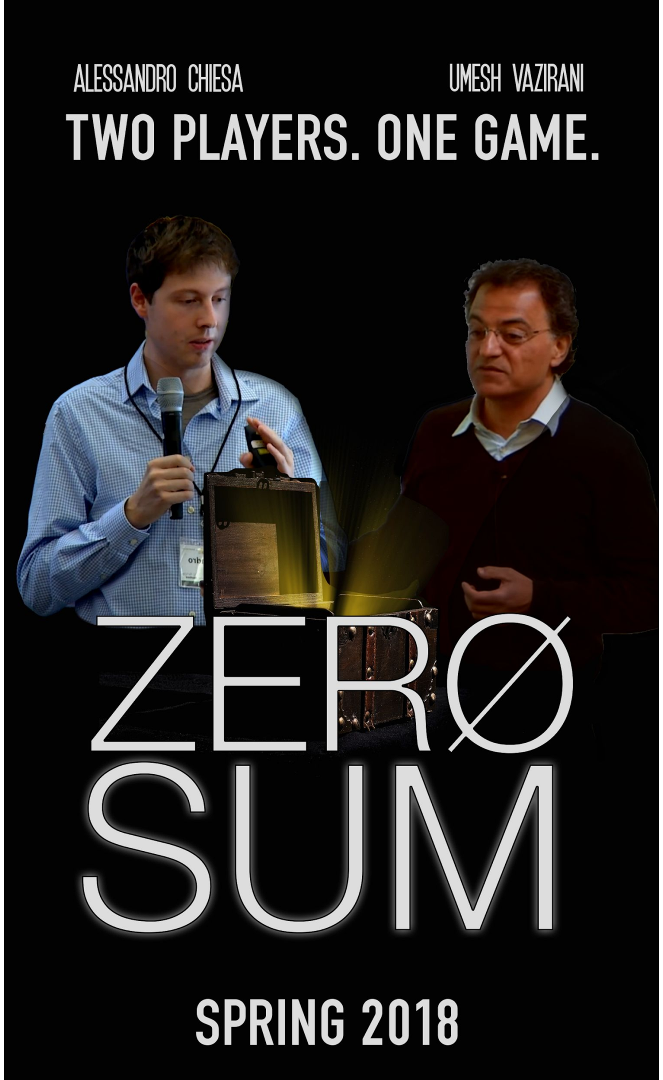
CS 170 Section 9
Zero-Sum Games, Reductions

Owen Jow | owenjow@berkeley.edu

Zero-Sum Games

The professors have been busy lately…



Reductions

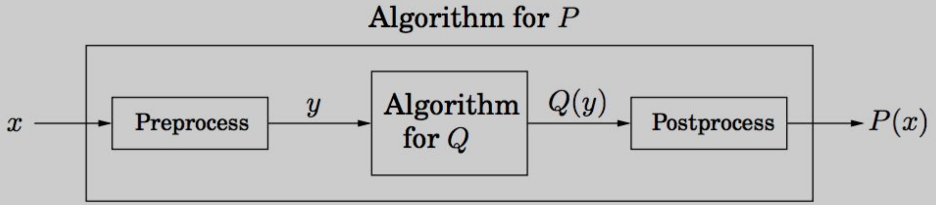
- Transform problem P into problem Q
- Solve problem Q
- Transform solution for Q into solution for P

If P reduces to Q, then Q is at least as difficult as P.
(It wouldn't make sense the other way: "to solve a hard problem, we can just solve this easy problem.")

If an algorithm that solves Q can be used to solve P, then P must reduce to Q.

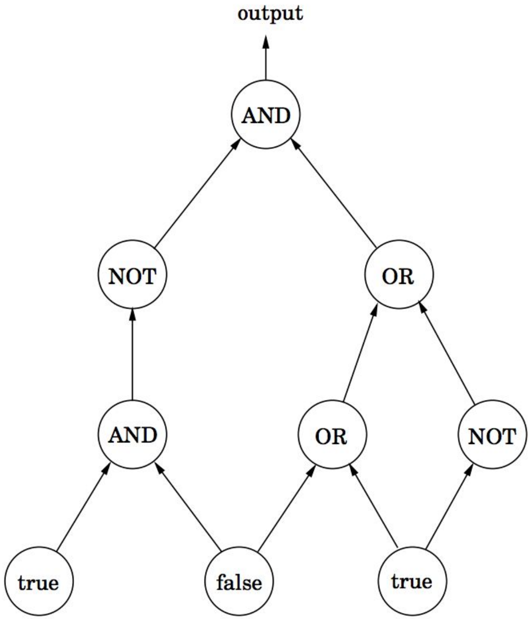
Examples:

- bipartite matching → max flow
- anything in **P** → linear programming



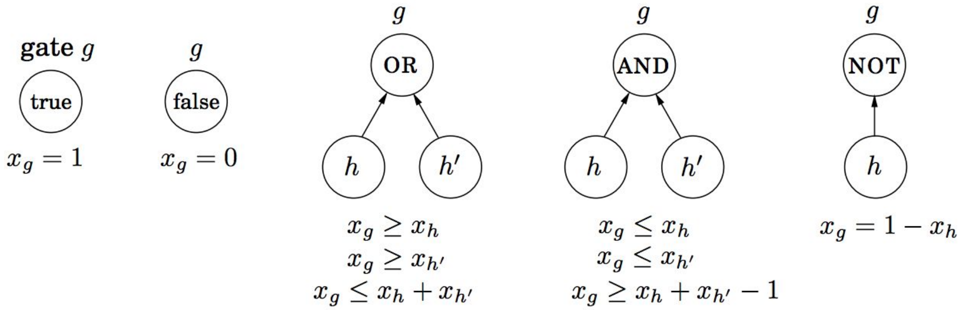
Linear Programming

- All problems solvable in polynomial time reduce to **Boolean circuit evaluation**



Linear Programming

- Boolean circuit evaluation can be reduced to linear programming



- Therefore, all problems solvable in polynomial time reduce to linear programming

Linear programming is one of the two most general algorithmic techniques! (The other is dynamic programming, to which circuit evaluation also reduces.)

Maximal Matching

Have:

- an undirected graph G = (V, E)

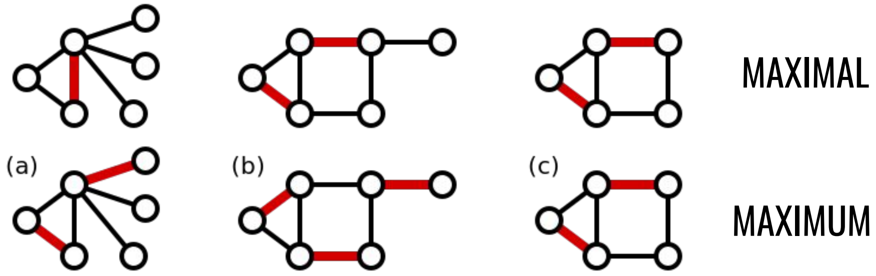
Want to show:

- the size of every maximal matching M is at least half the size of a maxi**mum** matching M*

Matching: a set of edges with no common vertices

Maximal matching: a matching for which no edge can be added without introducing a common vertex

Maximum matching: a matching with the largest possible number of edges



Maximal Matching Solution

Have:

- an undirected graph G = (V, E)

Want to show:

- the size of every maximal matching M is at least half the size of a maxi**mum** matching M*

Matching: a set of edges with no common vertices

Maximal matching: a matching for which no edge can be added without introducing a common vertex

Maximum matching: a matching with the largest possible number of edges

At least one endpoint of every edge in M* must be involved in M. (Otherwise M would not be a maximal matching, because we could add the entirely uninvolved edge to it.)

Therefore the number of **vertices** covered by any M must be at least |M*|. Then, since each edge covers two vertices, the number of **edges** (i.e. the size) of M must be at least |M*| / 2.

Reducing Vertex Cover to Set Cover

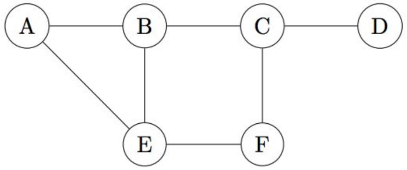
Recall the **minimum vertex cover** problem.

Have:

- an undirected graph G = (V, E)

Want to find:

- the smallest set of vertices U ⊆ V that covers the set of edges E



e.g. {A, E, C, D} is a vertex cover, *but not a minimum vertex cover.* {B, E, C} **is** a minimum vertex cover.

Recall the **minimum set cover** problem.

Have:

- an set U of elements
- a collection S1, …, Sm of subsets of U

Want to find:

- the smallest collection of subsets whose union equals U

e.g. if U is {a, b, c, d}, S1 is {a, b, c}, S2 is {b, c}, and S3 is {c, d}, then {S1, S3} is a minimum set cover.

Reduce the minimum vertex cover problem to the minimum set cover problem.

Reducing Vertex Cover to Set Cover Solution

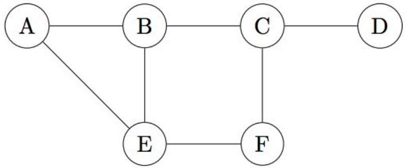
Goal: solve vertex cover, even though all we have is an algorithm to solve set cover.

We need to turn the minimum vertex cover problem into a minimum set cover problem.

Let U = the set of edges E.
For each v ∈ V, the set Sv should contain the set of edges which are adjacent to v.

Let $\{Sv^{(1)}, \ldots, Sv^{(k)}\}$ be a set cover. Then the corresponding vertex cover is $\{v^{(1)}, \ldots, v^{(k)}\}$.

Furthermore, if $\{Sv^{(1)}, \ldots, Sv^{(k)}\}$ is the minimum set cover then $\{v^{(1)}, \ldots, v^{(k)}\}$ is the minimum vertex cover!



e.g. U = {(a b), (a e), (b c), (b e), (c d), (c f), (e f)}
Sa = {(a b), (a e)}, Sb = {(a b), (b c), (b e)}, Sc = {(b c), (c d), (c f)}, Sd = {(c d)},
Se = {(a e), (b e), (e f)}, Sf = {(e f), (c f)}

then (one) minimum set cover is {Sb, Se, Sc}.

Midterm 2 Debrief

– Questions

- Length?
- Killer problems?
- Difficulty?
- Poor wording?
- Best time of your life?