Computation Intractability II

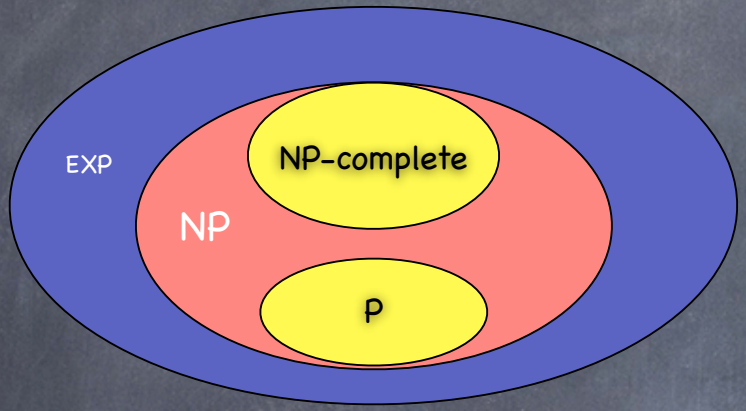
Review: Mystery Problems

Huge class of natural and interesting problems for which

- we don't know any polynomial time algorithm
- we can't prove that none exists

Examples: Steiner Tree, Knapsack, Traveling Salesman, Vertex Cover, Independent Set, Facility Design, etc.

NP-completeness



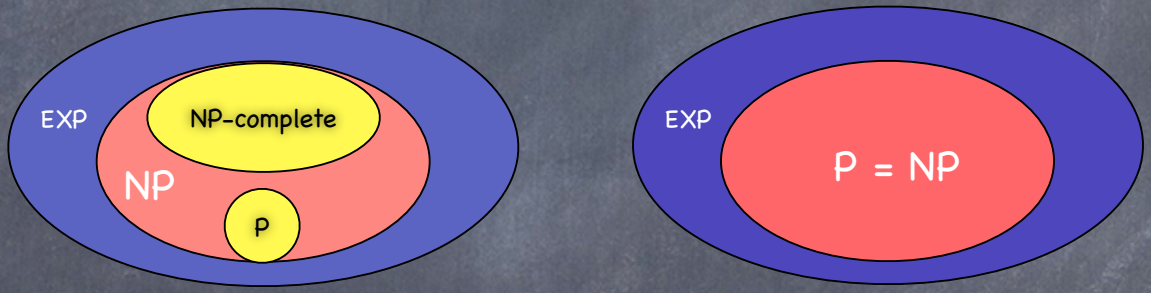
NP-complete: class of problems that are "as hard" as every other problem in NP

A polynomial-time algorithm for any NP-complete problem implies one for every problem in NP

Today: explore NP-completeness and its implications

P != NP?

Two possibilities (we don't know which is true, but we think P != NP)



$1M prize if you can figure out the answer
(one of Clay institute's seven Millennium Problems)

Plan

- Polynomial Time Reductions
- Define NP
- Explore NP-completeness

Polynomial-Time Reduction

- $Y \leq_p X \rightarrow$ can solve Y in polynomial-time using polynomial-time algorithm for X as a black box (if one exists!)
- Statement about relative hardness:
  - If $Y \leq_p X$ and $X \in P$, then $Y \in P$
  - If $Y \leq_p X$ and $Y \notin P$, then $X \notin P$

Basic Reduction Strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover

SET COVER: Given a set U of elements, a collection S_1, S_2, . . . , S_m of subsets of U, and an integer k, does there exist a collection of $\leq$ k of these sets whose union is equal to U?

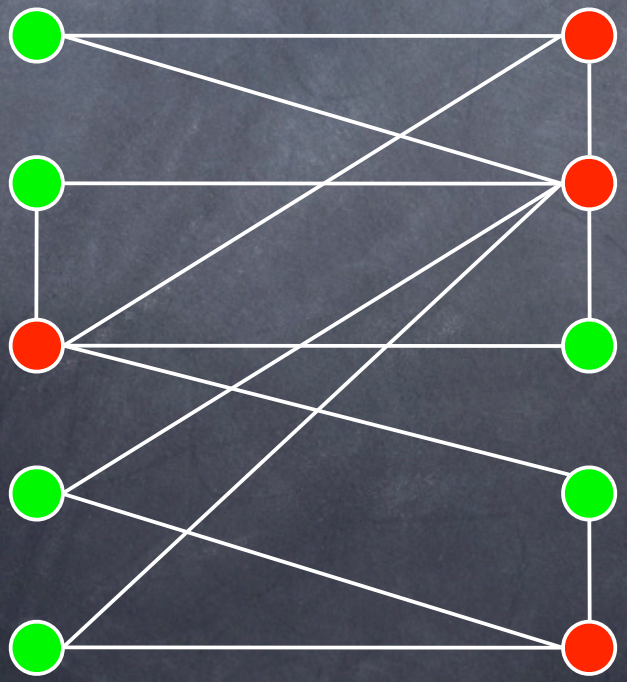
Example. U = {A, G, H, P, SH}

S1 = {A, H}
S2 = {G, SH}
S3 = {A, H, SH}
S4 = {P}
S5 = {G, H, SH}

k = 3 → yes
k = 2 → no

Vertex Cover

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?



Basic Reduction Strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Satisfiability

Important!

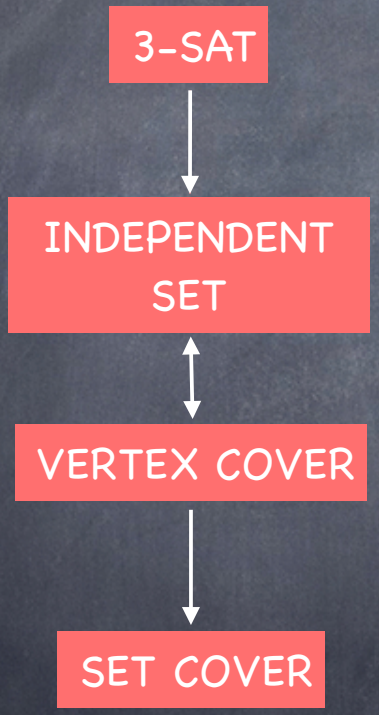
- Term: A Boolean variable or its negation.
  x_i OR $\overline{x_i}$
- Clause: A disjunction ("or") of terms.
  $C_j = x_1 \vee x_2 \vee x_3$
- Formula Φ: A conjunction ("and") of clauses
  $C_1 \wedge C_2 \wedge C_3 \wedge C_4$
- SAT: Given a formula, is there a truth assignment that satisfies all clauses? (i.e. all clauses evaluate to "true").
  Example on board.
- 3-SAT: special case where each clause has exactly 3 terms

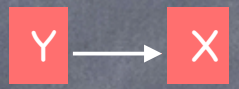
3-SAT is Reducible to Independent Set

- Claim. 3-SAT $\leq_p$ INDEPENDENT-SET.
- Proof. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

Do reduction on board

Reductions, Pictorially





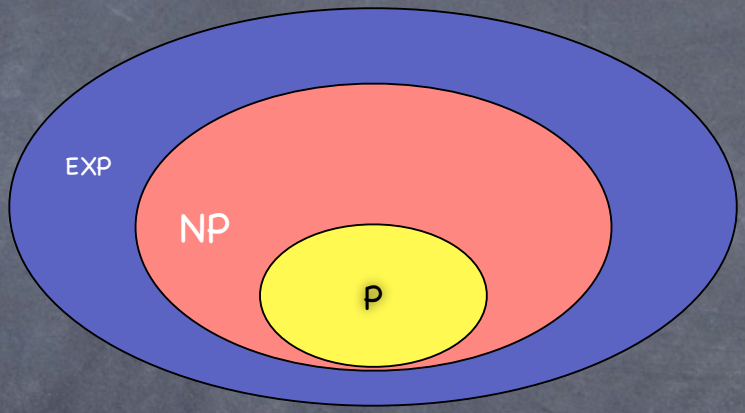
means $Y \leq_P X$

Partial map of problems we can use to solve other ones in polynomial time by transitivity of reductions:

If $Y \leq_P X$ and $X \leq_P W$, then $Y \leq_P W$.

Proof idea: Compose the two algorithms.

Toward an (Informal) Definition of NP



What is special about the mystery problems?

First: Let's Just Focus on Decision Problems, OK?

- Decision problem. Does there exist a vertex cover of size $\leq$ k?
- Search problem. Find vertex cover of minimum size.
- Self-reducibility. Search problem $\leq_P$ decision version.
  - Applies to all (NP-complete) problems in this chapter.
  - Justifies our focus on decision problems.

P and NP

- P. Decision problems for which there is a polynomial time algorithm
- NP. Decision problems for which there is a polynomial time certifier
- Let's see some examples...

Solving vs. Certifying

- E.g., Independent Set
  - Solve => Find an independent set of size k for a graph G

    S = FindIndeptSet (G, k)
  - Certify => Check whether S is an independent set of size k

    isIndept = CertifyIndeptSet (G, k, S)

—No known polynomial time solution to solve independent set

✓Easy to certify a solution in polynomial time

Example: Non-Prime

- COMPOSITES. Given an integer s, is s non-prime?
- Solution or "certificate". A nontrivial factor t of s. Note that such a certificate exists iff s is non-prime.

- Certifier(s, t):
  Return "true" if $1 \le t \le s$, and s/t is an integer

- Instance:
  s = 437,669.
- Certificate:
  t = 541 or 809.

Example: 3-SAT

- 3-SAT. Given formula Φ, is there a satisfying assignment?

  $(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$

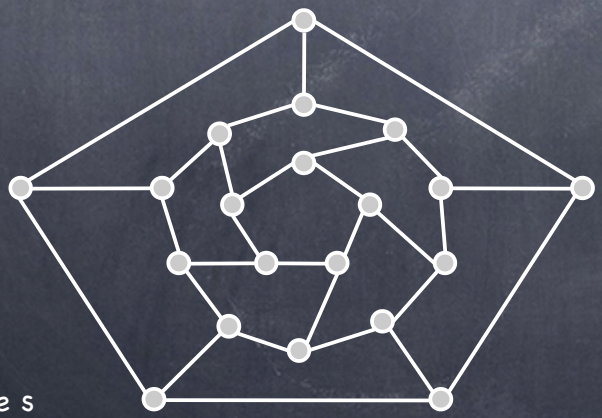
- Certificate. An assignment of truth values to the n boolean variables.

  x_1 = true, x_2 = true, x_3 = false, x_4 = true

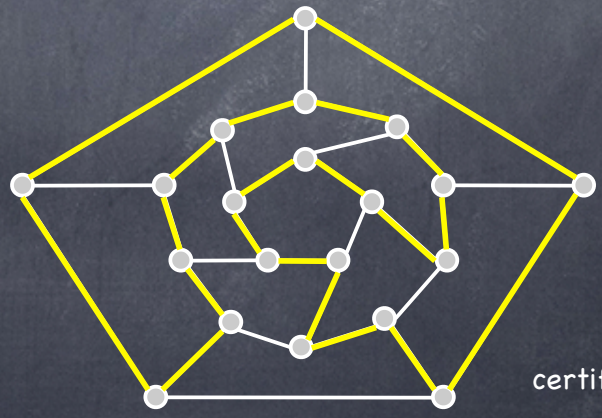
- Certifier. check that each clause in Φ has at least one true literal.

Example: Hamiltonian Cycle

- HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?
- Certificate. A permutation of the n nodes.

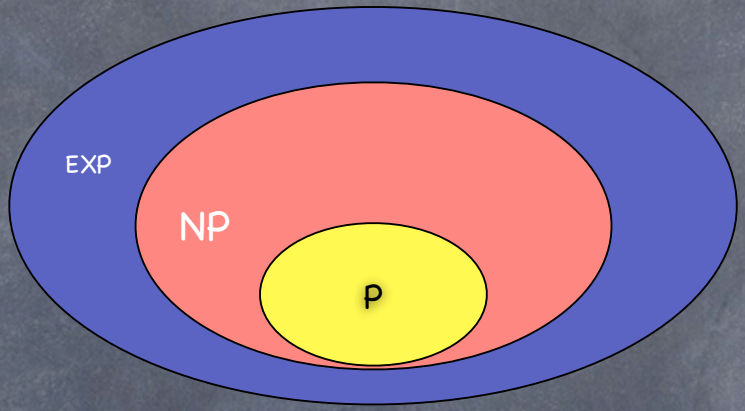


instance s



certificate t

Takeaway

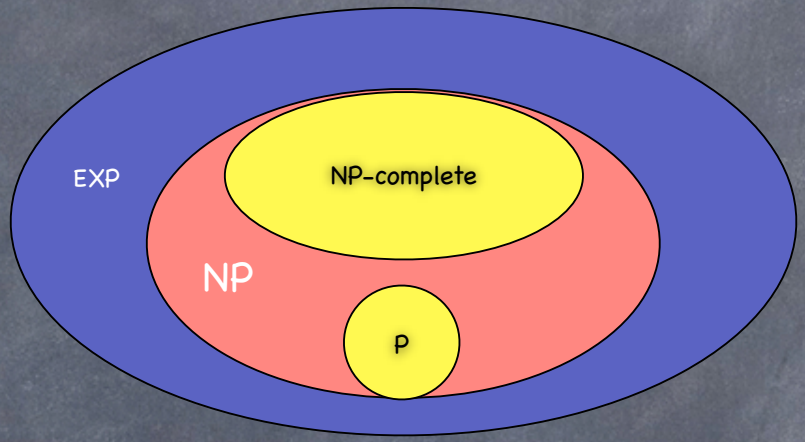


- INDEPENDENT-SET, NON-PRIME, 3-SAT, and HAMILTONIAN-CYCLE are in NP
- So are many more problems…

P, NP

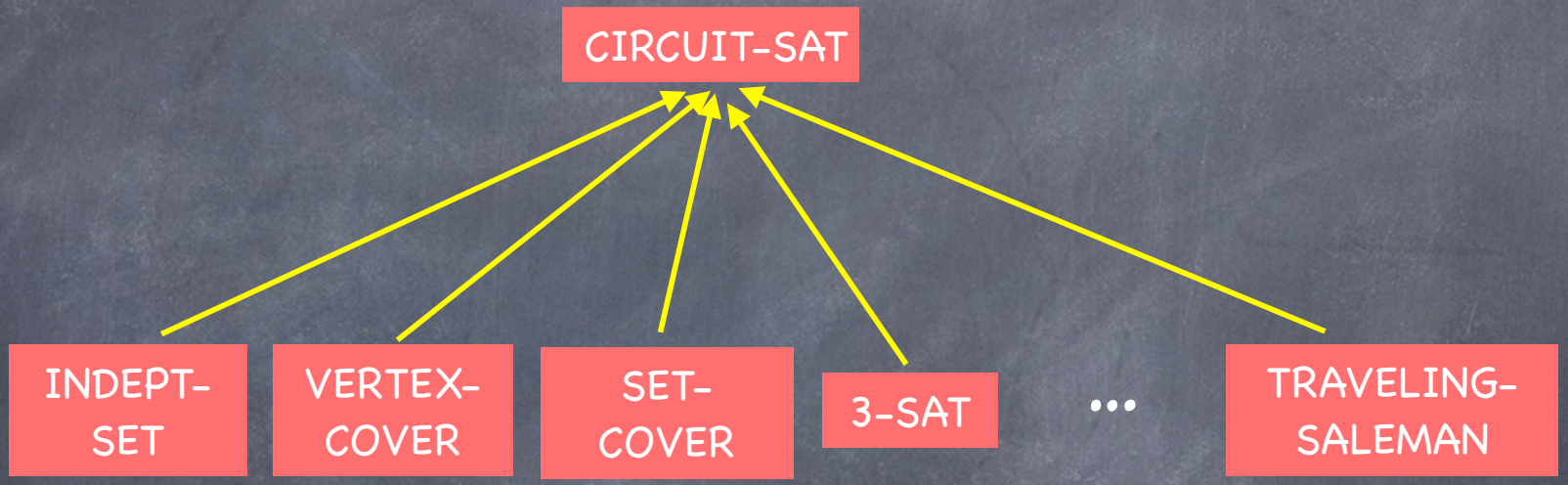
- Claim. $P \subseteq NP$.
- If we can solve the problem in polynomial time, we can always certify a solution in polynomial time. (Easy to prove formally given a slightly more formal definition of NP)

NP-Complete



- NP-complete. A problem Y in NP with the property that for every problem X in NP, $X \leq_P Y$.
- WHAT?!

NP-Completeness



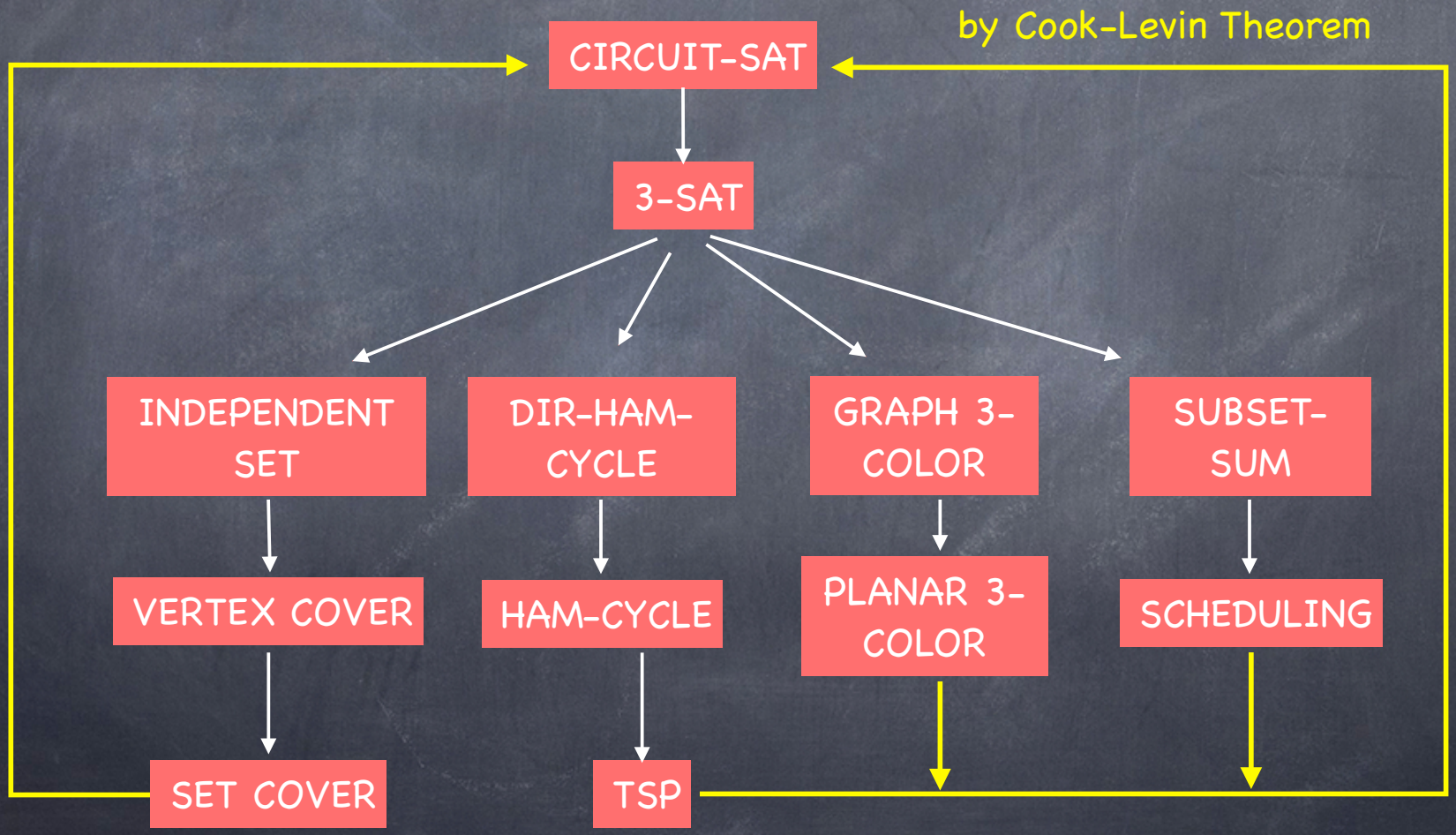
All of NP (note: this includes P too!)

Cook-Levin Theorem. In 1971, Cook and Levin independently showed that particular problems were NP-Complete.

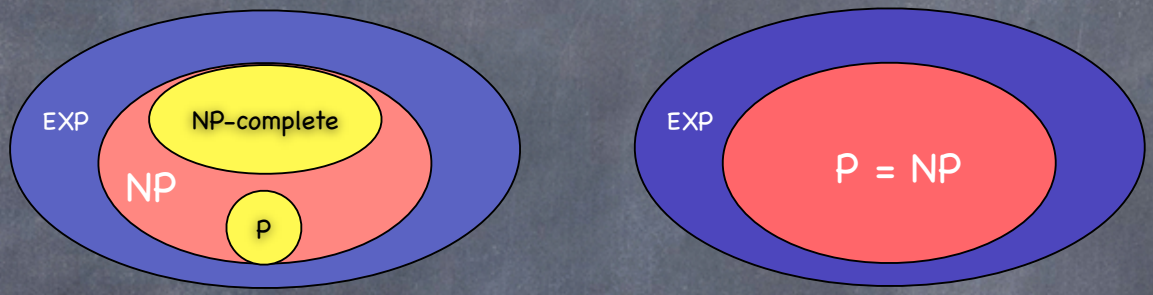
We'll look at CIRCUIT-SAT as canonical NP-Complete problem

NP-Completeness

Given a canonical NP-complete problem, others follow by polynomial-time reductions.



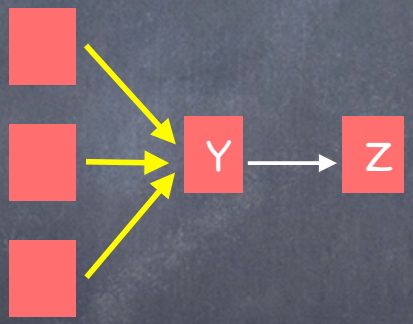
NP-Complete = Intractable (Probably)



- If there is a polynomial-time algorithm for any NP-complete problem, then P = NP. (We don't think this is true)

What To Know

How to prove problem Z is NP-complete:
(1) Show that Z is in NP
(2) Find an NP-complete problem Y and show that $Y \leq_p Z$

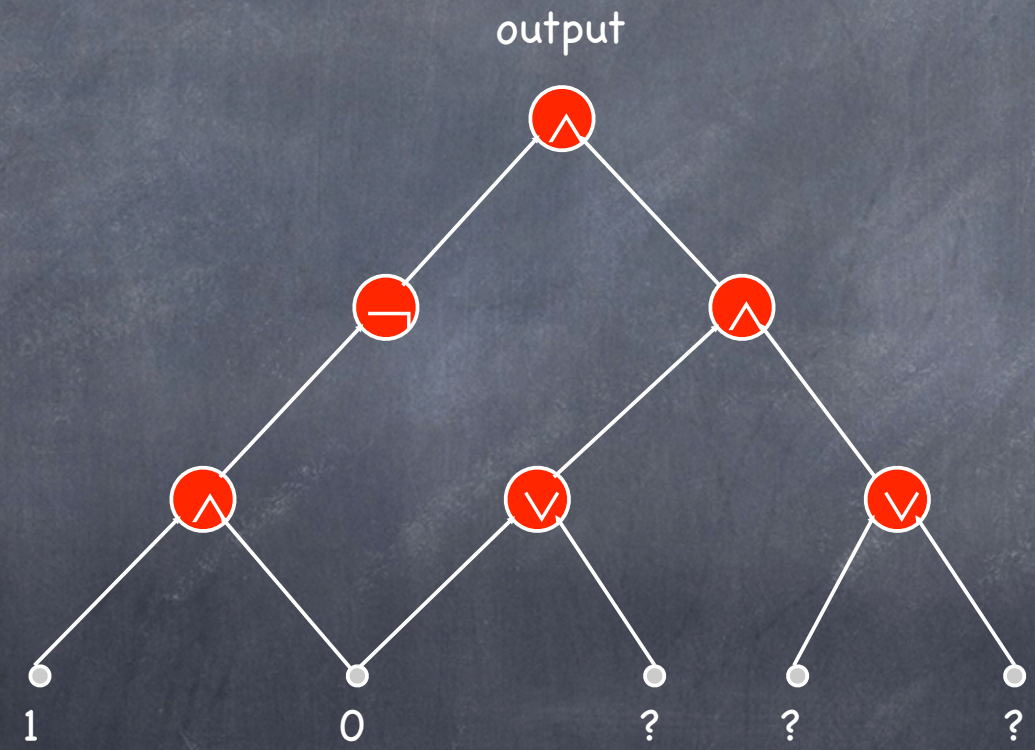


If P != NP, there is no polynomial-time algorithm for Z

Overview of Cook-Levin Theorem

Circuit Satisfiability

- CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?



CIRCUIT-SAT is NP-Complete

- Theorem. CIRCUIT-SAT is NP-complete.

- Proof idea: encode certifier algorithm as circuit

  For any problem in NP, the certifier C(s, t) is a polynomial-time algorithm that outputs yes/no.

  Encode C(s, t) as a circuit of polynomial size that hard-codes the problem s, and is satisfiable iff there is a certificate t causes C(s, t) to output "yes".

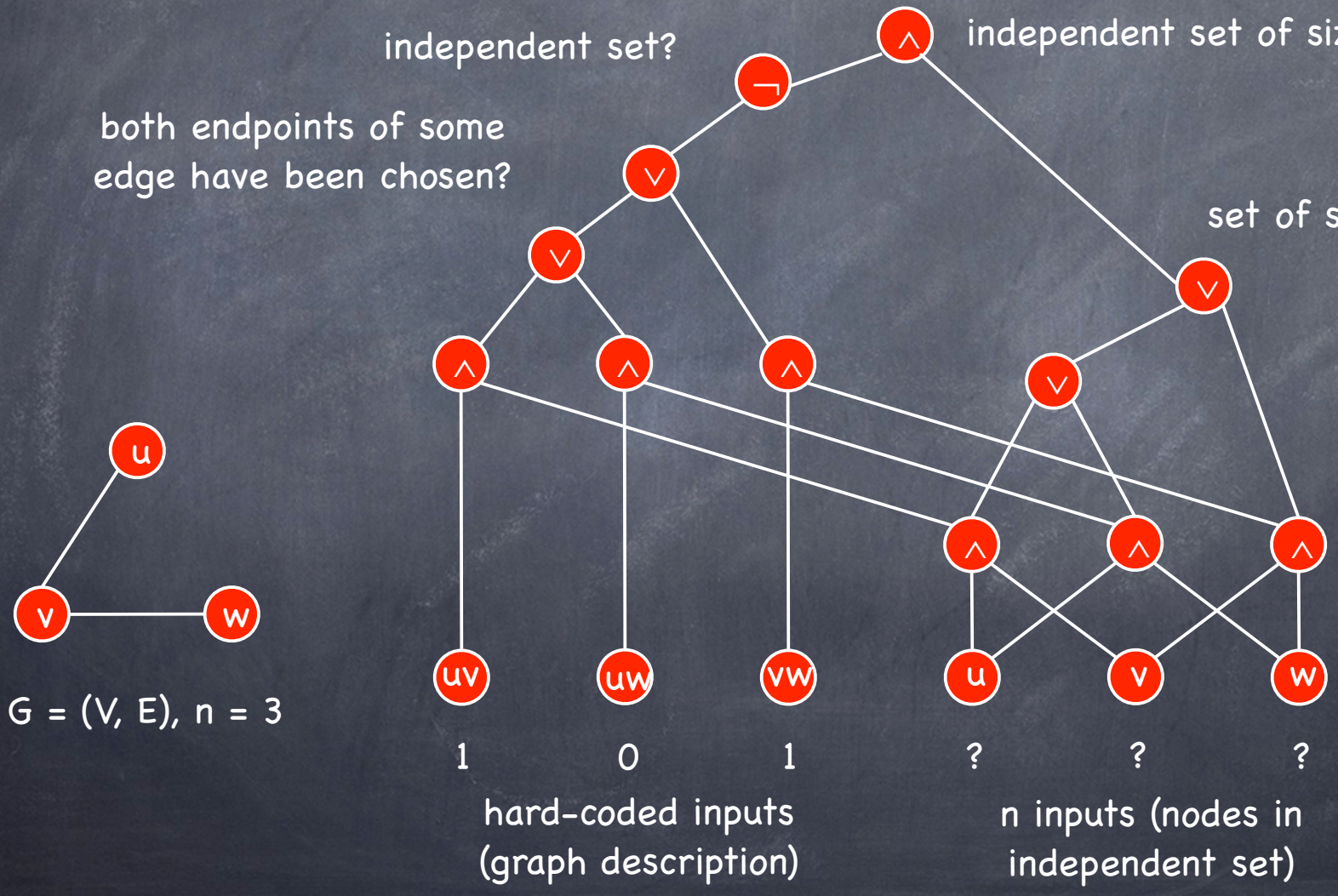
  The circuit can "guess" the certificate.

Example: Independent Set

- Independent Set: Input s = (G, k). Is there an independent set S of size at least k?
- Certificate: $t \subseteq V$ (the candidate independent set)
- Certifier: C(s, t): check if t is an independent set of size at least k

Example

Circuit that outputs true iff graph G has an independent set of size 2.

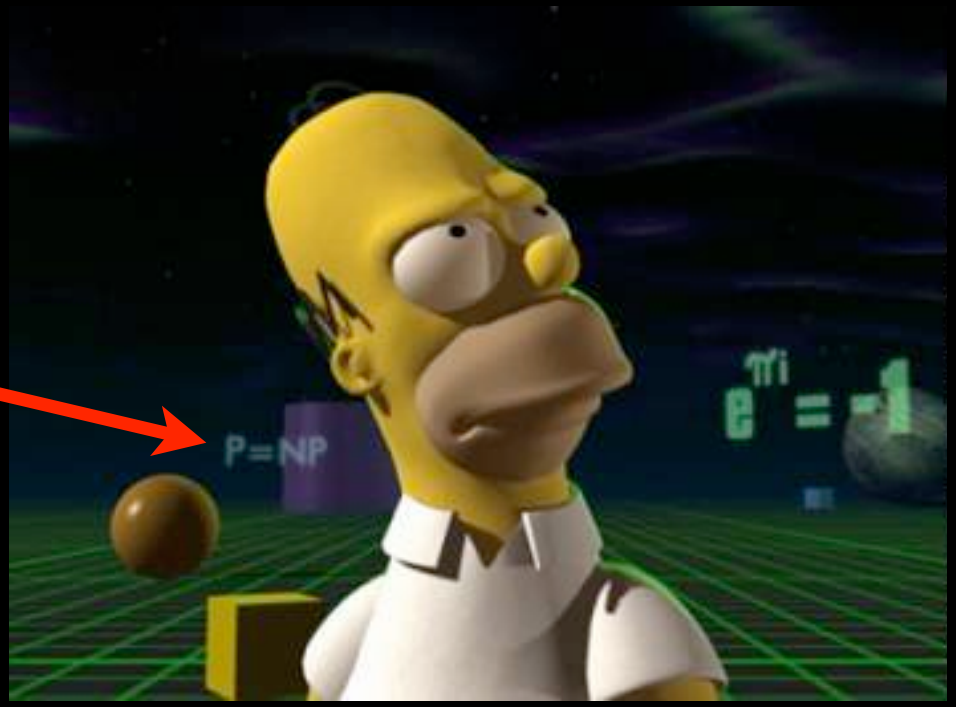


Some NP-Complete Problems

- Thousands of known NP-Complete problems. Genres:
  - Packing: SET-PACKING, INDEPENDENT SET.
  - Covering: SET-COVER, VERTEX-COVER.
  - Constraint satisfaction: SAT, 3-SAT.
  - Sequencing: HAMILTONIAN-CYCLE, TSP.
  - Partitioning: 3D-MATCHING 3-COLOR.
  - Numerical: SUBSET-SUM, KNAPSACK.
- In practice. Most NP problems are either known to be in P or NP-complete.
- Notable exceptions. Factoring, graph isomorphism

Impact of NP-Completeness

- Prime intellectual export of CS to other disciplines
- You should know about this if you are a chemist, physicist, mathematician….



Copyright © 1990, Matt Groening

My Hobby:

Embedding NP-Complete Problems in Restaurant Orders