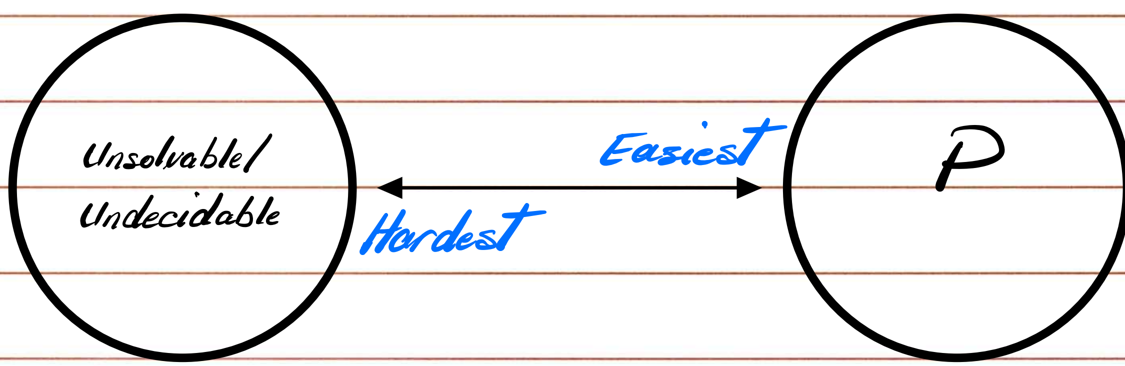
Tractability & Computational Complexity Classes

Computational Complexity Spectrum



ex.: Halting Problem

Plan: Explore the space of computationally hard problems to arrive at a mathematical characterization of a large class of them.

Technique: Compare relative difficulty of different problems.

Loose Definition:

If a problem X is at least as hard as problem Y, it means that if we could solve X, we could also solve Y.

Formal Definition:

$Y \leq_p X$ (Y is polynomial time reducible to X)

if Y can be solved using a polynomial number of standard computational steps plus a polynomial number of calls to a blackbox that solves X.

Suppose $Y \leq_p X$, if X can be solved in polynomial time, then Y can be solved in polynomial time.

Suppose $Y \leq_p X$, if Y cannot be solved in polynomial time, then X cannot be solved in polynomial time.

Independent Set

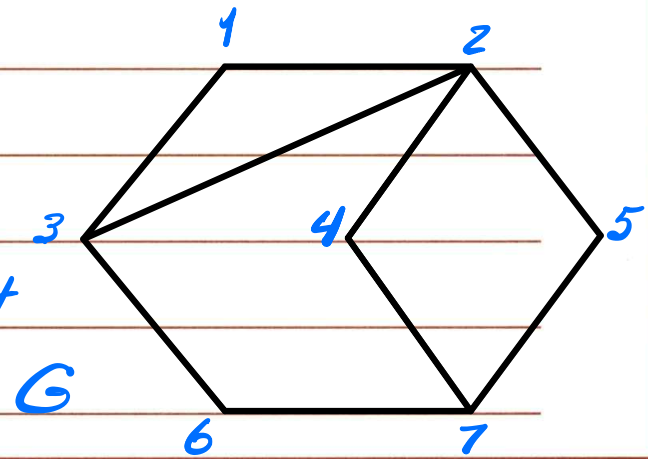
Def.: In a graph $G=(V,E)$, we say that a set of nodes $S \subseteq V$ is "independent" if no two nodes in S are joined by an edge.

Some independent sets in G:

$\{2\}$

$\{1,7\}$

$\{1,4,5,6\}$ lagest indep. set



Independent Set Problem

- Find the largest independent set in graph G. (optimization version)

- Given a graph G and a number k, does G contain an independent set of size at least k? (decision version)

Vertex Cover

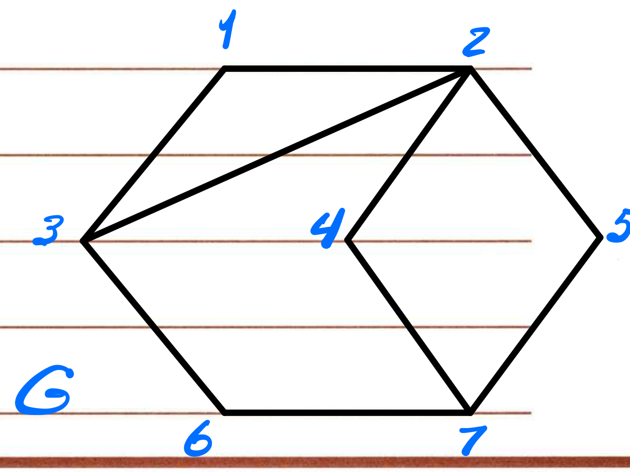
Def.: In a graph $G=(V,E)$, we say that a set of nodes $S \subseteq V$ is a vertex cover if every edge in E has at least one end in S.

Some vertex cover sets in G:

$\{1,2,3,4,5,6,7\}$

$\{2,3,4,5,7\}$

$\{2,3,7\}$ smallest vertex cover set



Vertex Cover Problem

- Find the smallest vertex cover set in G.
  (optimization version)

- Given a graph G and a number k, does G contain a vertex cover set of size at most k?
  (decision version)

FACT: Let $G=(V,E)$ be a graph, then S is an independent set if and only if its complement $V-S$ is a vertex cover set.

Proof: A) First suppose that S is an independent set.

Consider edge uv

There are 3 possibilities:

1_ u is in S and v is not
$\Rightarrow V-S$ will have v and not u

2_ v is in S and u is not
$\Rightarrow V-S$ will have u and not v

3_ Neither v nor u is in S
$\Rightarrow V-S$ will have both u and v

B) Suppose that $V-S$ is a vertex cover ...

Claim: Independent Set $\leq_p$ Vertex Cover

Proof: Having proven that the complement of a vertex cover set is an independent set, given a black box that solves the vertex cover problem, we can decide if G has an independent set of size at least k, by asking the black box if G has a vertex cover of size at most n-k.

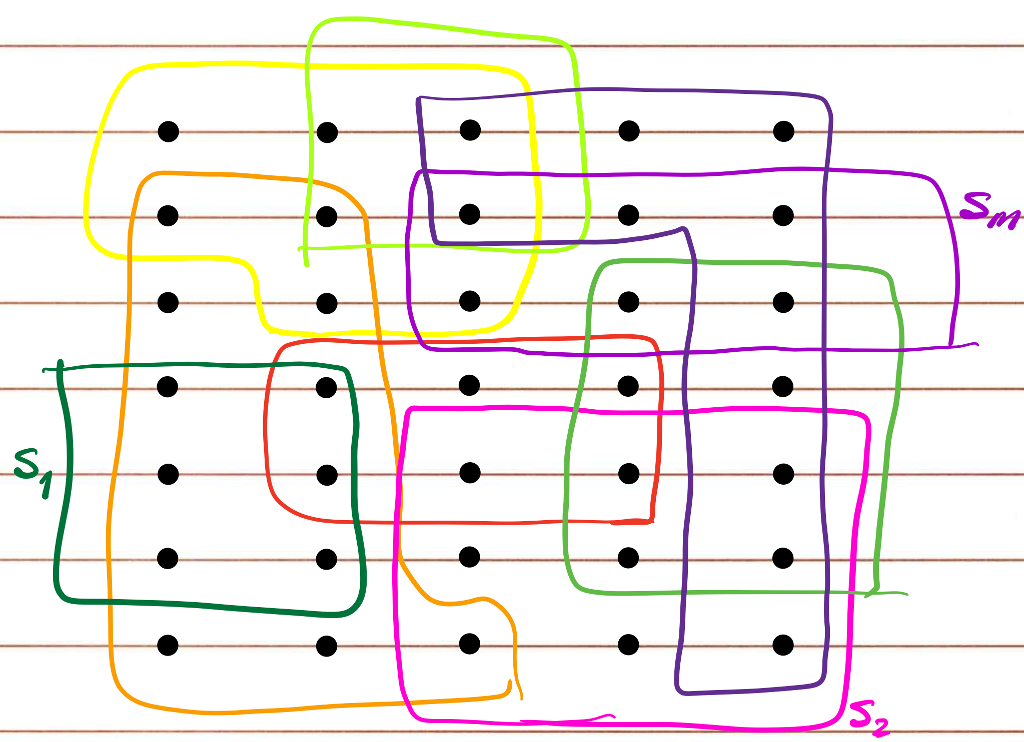
Claim: Vertex Cover $\leq_p$ Independent Set

Proof: Having proven that the complement of an independent set is a vertex cover set, given a black box that solves the indep. set problem, we can decide if G has a vertex cover set of size at most k, by asking the black box if G has an independent set of size at least n-k.

Set Cover Problem

Given a set U of n elements, a collection $S_1, S_2, \dots, S_m$ of subsets of U, and a number k, does there exist a collection of at most k of these sets whose union is equal to all of U?

Each • represents an element in the set U



Claim: Vertex Cover $\leq_p$ Set Cover

Plan: Given an instance of the vertex cover problem (G, k), we will construct a set of sets such that there is a vertex cover of size $\underline{k}$ is G iff there are $\underline{k}$ sets whose union contains all elements of all sets.

$S_1 = \{(1,2),(1,3)\}$

$S_2 = \{(1,2),(2,3),$
$(2,4),(2,5)\}$

$S_3 =$

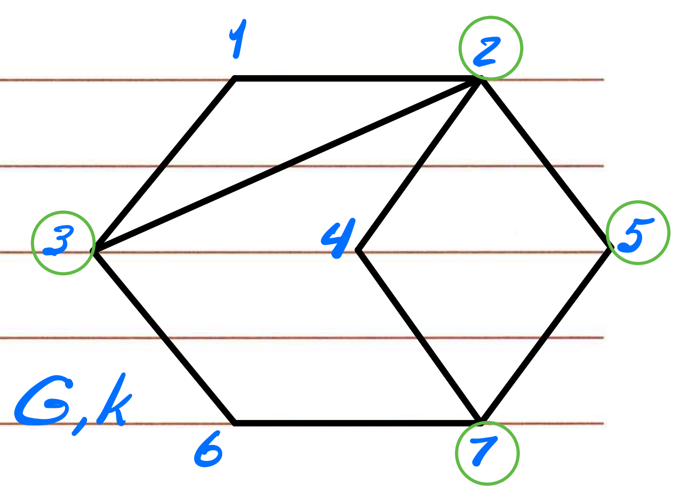
$S_4 = \quad \vdots$

$S_5 =$

$S_6 =$

$S_7 = \{(4,7),(5,7),(6,7)\}$

k



Proof:

We need to show that G has a vertex cover of size k, iff the corresponding set cover instance has k sets whose union contains all edges in G.

A) If we have a vertex cover set of size k in G, we can find a collection of k sets whose union contains all edges in G.

Explain how ...

B) If we have k sets whose union contains all edges in G, we can find a vertex cover set of size k in G.

Explain how ...

Reduction Using Gadgets

Some Definitions:

- Given n Boolean variables $x_1 \dots x_n$, a clause is a disjunction of terms $t_1 \vee t_2 \vee \dots \vee t_\ell$ where $t_i \in \{x_1 \dots x_n, \bar{x}_1 \dots \bar{x}_n\}$

- A truth assignment for X is an assignment of values 0 or 1 to each x_i.

- An assignment satisfies a clause C if it cause C to evaluate to 1.

e.g. for claus $(x_1 \vee \bar{x}_2)$.

$x_1=0$, $x_2=0$ is a satisfying truth assignment

$x_1=1$, $x_2=0$ is a satisfying truth assignment

$x_1=0$, $x_2=1$ is not...

- An assignment satisfies a collection of clauses if $C_1 \wedge C_2 \dots C_k$ evaluates to 1.

e.g. $(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$

$x_1=1$, $x_2=1$, $x_3=1$ Not a satisfying truth assignment

$x_1=0$, $x_2=0$, $x_3=0$ Satisfying truth assignment

$x_1=1$, $x_2=0$, $x_3=0$ Satisfying truth assignment

Problem Statement

Given a set of clauses $C_1 \ldots C_k$ over a set of variables $X = \{x_1 \ldots x_n\}$, does there exist a satisfying truth assignment?

This is the general form of the satisfiability problem (SAT)

Problem Statement

Given a set of clauses $C_1 \ldots C_k$ each of length $\underline{3}$ over a set of variables $X = \{x_1 \ldots x_n\}$, does there exist a satisfying truth assignment?

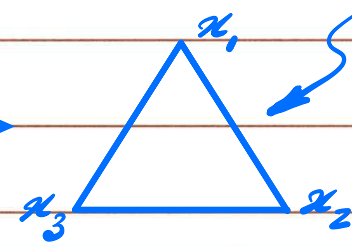
(3-SAT)

Claim: 3-SAT $\leq_p$ Independent Set

Plan: Given an instance of 3-SAT with $\underline{k}$ clauses, we will construct a graph G that has an indep. set of size $\underline{k}$ iff the 3-SAT instance is satisfiable.

A triangle-shaped gadget representing one clause

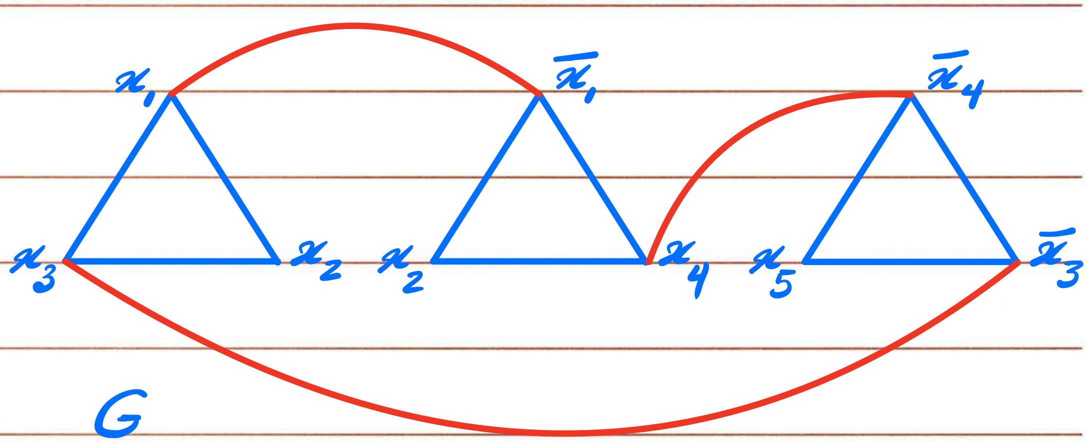
$(x_1 \vee x_2 \vee x_3) \longrightarrow$



ex. $C_1 = (x_1 \vee x_2 \vee x_3)$

$C_2 = (\bar{x}_1 \vee x_2 \vee x_4)$

$C_3 = (\bar{x}_4 \vee x_5 \vee \bar{x}_3)$

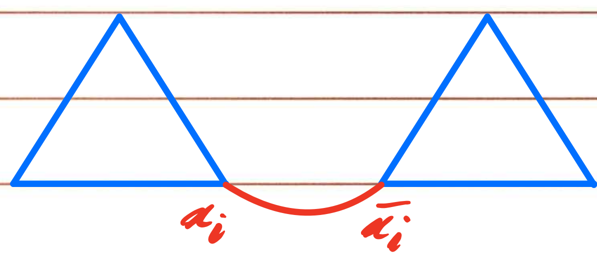


G

Claim: The 3_SAT instance is satisfiable iff the graph G has an independent set of size k.

Proof: A) If the 3_SAT instance is satisfiable, then there is at least one node label per triangle that evaluates to 1.

Let S be a set containing one such node from each triangle



Since both x_i and $\bar{x}_i$ cannot evaluate to 1, then nodes in the set S (one per triangle) cannot have an edge between them. I.e., they form an indep. set.

B) Suppose G has an independent set S of size at least $\underline{k}$.

If x_i appears as a label in S, then set x_i to $\underline{1}$

If $\overline{x}_i$ appears as a label in S, then set x_i to $\underline{0}$

If neither x_i nor $\overline{x}_i$ appear as a label in S, then set x_i to either $\underline{0}$ or $\underline{1}$

Efficient Certification

A problem has efficient certification if given a solution, the correctness of the solution can be confirmed in polynomial time.

To show efficient certification, we need

1- Polynomial length certificate

2- Polynomial time certifier

Showing Efficient Certification

3-SAT:

Certificate $\underline{t}$ is an assignment of truth values to variables (x_i)

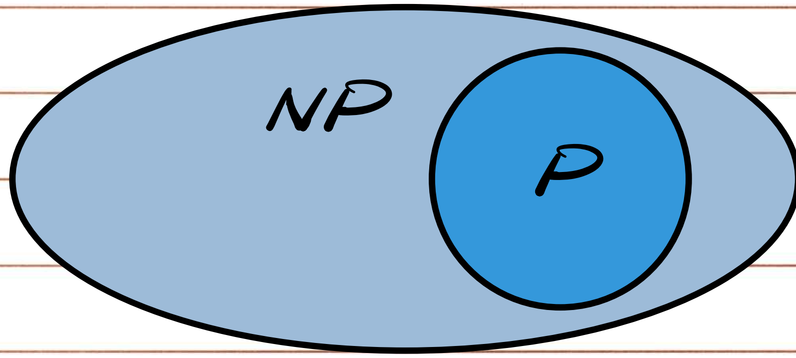
Certifier: Evaluate the clauses. If all of them evaluate to 1, then it answers "yes".

Independent Set:

Certificate $\underline{t}$ is a set of nodes of size at least $\underline{k}$ in G.

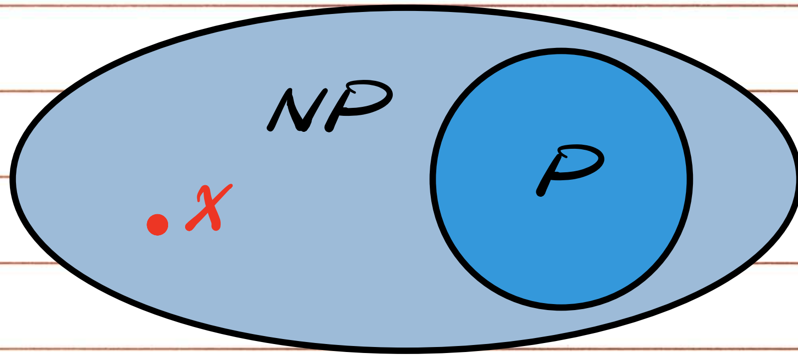
Certifier: Check each edge to make sure no edges have both ends in the set.
Check size of the set to be $\geq k$
Check that there no repeating nodes in the set.

Def.: Class NP is the set of all decision problems for which there exists an efficient certifier.



All problems in P are in NP

Are all problems in NP in P?
i.e. is P = NP ?   We don't know!
This is an open problem.



If $X \in NP$ and for all $Y \in NP$ $Y \leq_p X$, Then X is the hardest problem in NP.

3_SAT has been proven to be the hardest problem in NP.

Such a problem (hardest problem in NP) is called NP_Complete.

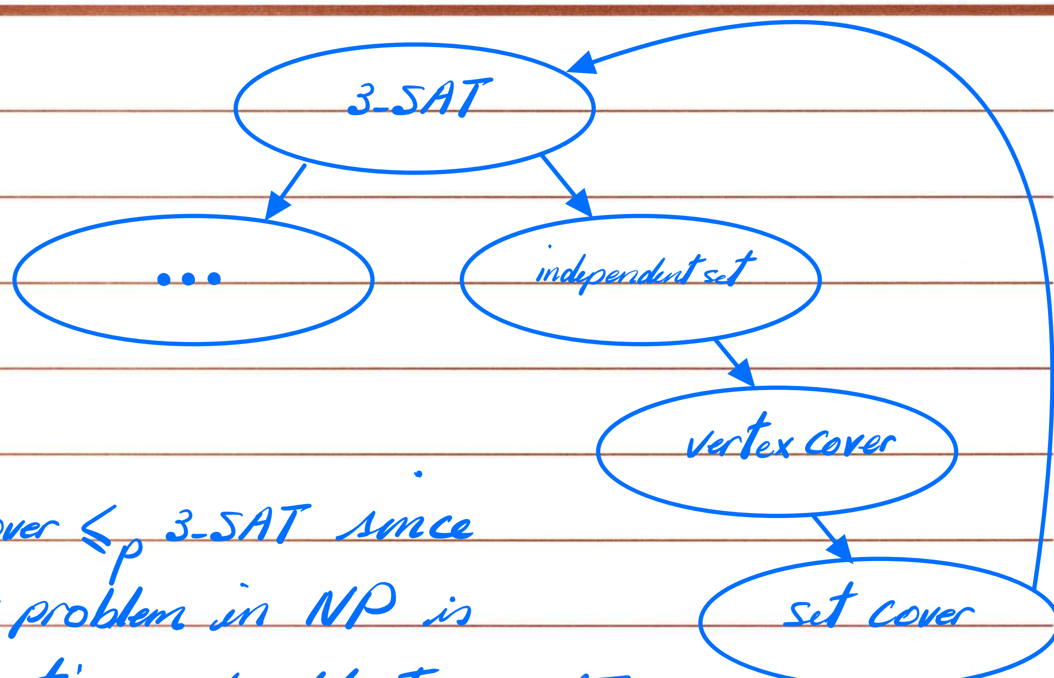
Transitivity in polyn. time reduction

if $Z \leq_p Y$ and $Y \leq_p X$, then $Z \leq_p X$

3-SAT $\leq_p$ independent set

independent set $\leq_p$ vertex cover

vertex cover $\leq_p$ set cover



set cover $\leq_p$ 3-SAT since every problem in NP is polyn. time reducible to 3-SAT.

So, by the transitivity property, set cover $\leq_p$ vertex cover

Basic strategy to prove a problem X is NP-Complete:

1. Prove $X \in NP$

2. Choose a problem Y that is known to be NP-Complete.

3. Prove $Y \leq_p X$

Def.: NP-hard is the class of problems that are at least as hard as NP-Complete problems.