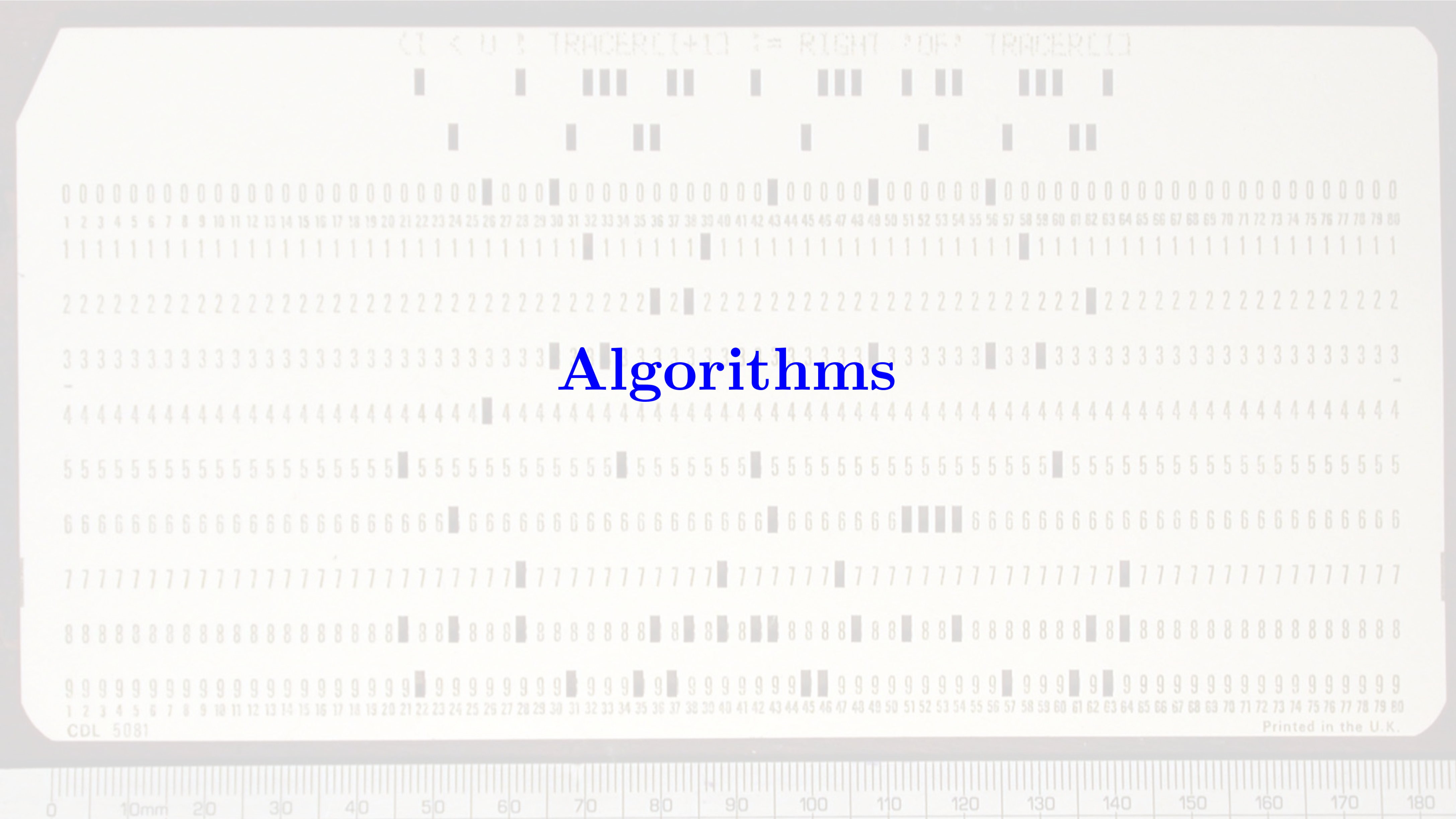
Algorithms



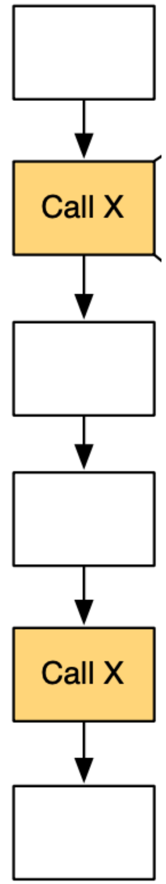
Reductions as tool for hardness

We want prove some problems are computationally difficult.

As a first step, we settle for relative judgements:

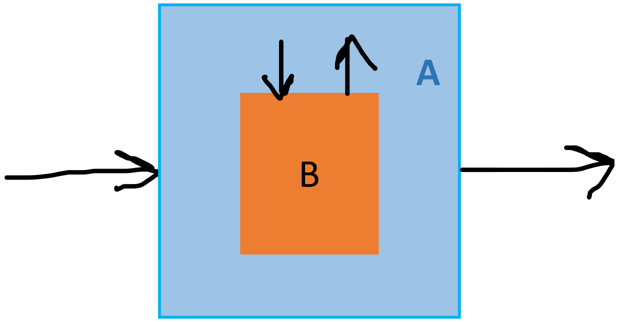
Problem X is at least as hard as problem Y

To prove such a statement, we reduce problem Y to problem X



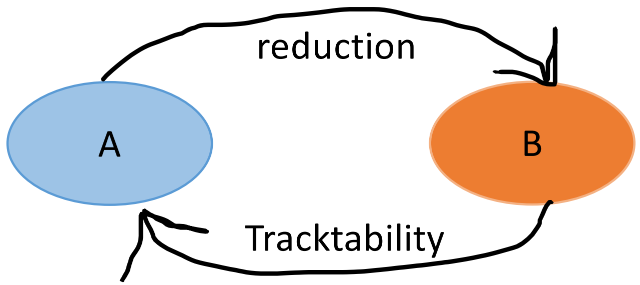
COMPUTATIONAL COMPLEXITY

Definition: A problem A is polynomial-time reducible to a problem B, if an algorithm that solves B can be *easily translated* to solve problem A.

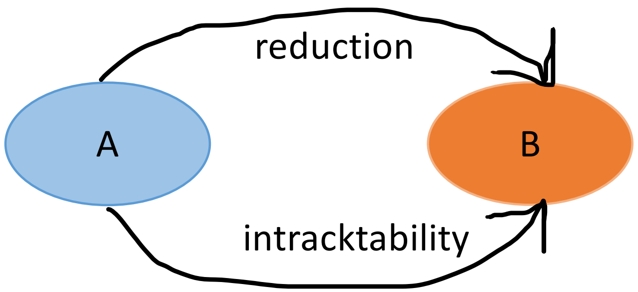


Notation. $A \leq_P B$: Problem A is polynomially reducible to problem B
It means that B is a least as hard as A

$\boldsymbol{A \leq_P B}$ **implies that:** If B has a polynomial time solution, than so does A



If $\boldsymbol{A \leq_P B}$, and A is NP-hard, then B is NP-hard



How to prove that an algorithm B is NP-hard?
Choose a NP-hard problem A
Prove that A can be reduced to B
It helps to know some of these!

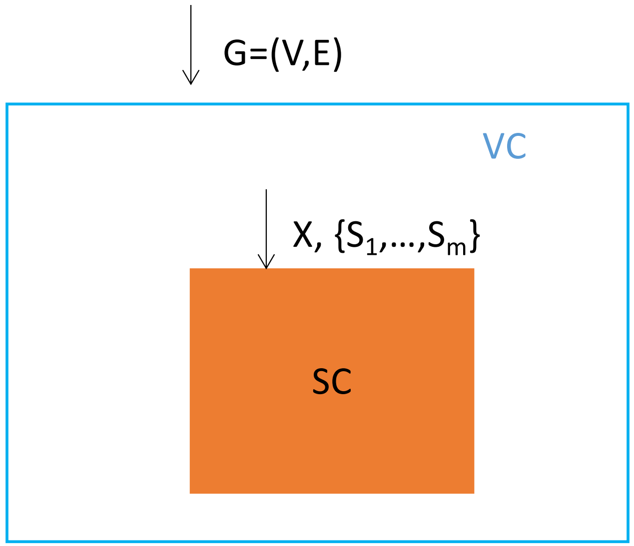
Polytime reductions

Is Vertex Cover $\leq_P$ Independent Set? *Yes*

Is Independent Set $\leq_P$ Vertex Cover? *Yes*

The complement of a minimum vertex cover is a maximum independent set and vice versa.

Exercise Is Vertex Cover $\leq_P$ Set Cover? *Yes*



For the input graph G to the vertex cover problem, we construct an equivalent set cover problem *in polynomial time* as follows.

The ground set X is the set of edges in G.

Corresponding to each vertex v in G, we construct a set $S_v \subseteq X$ consisting of the edges in G that are incident to v.

The set cover problem is a generalization of the vertex cover problem.

Satisfiability

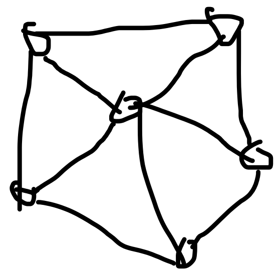
- Many of the problems we have seen so far are optimization problems: Max flow, minimum path, min cover
- We will now look at different class of problems, feasibility problems.
- Given a set of constraints, is there a feasible solution within these constraints?
- These problems can be encoded as satisfiability problems

Graph coloring

Given an undirected graph G=(V,E), and a positive integer k, we want to color the vertices so no edge is monochromatic.

.

Is it 4 colorable?
3?



Theorem: every planar graph is 4 colorable (can be drawn on a paper without its edge crossing)

Satisfiability Logic

$x_1, x_2, \cdots, x_n$: n boolean variables

Each variable takes a value of either `False` or `True` . *`False` and `True` are often represented by 0 and 1 respectively*

A *literal* is either a variable (e.g. x_i) or its negation (e.g. $\bar{x}_i$).

positive literal *negative literal*

A *clause* is a disjunction of literals: $l_1 \vee l_2 \vee \cdots \vee l_k$.

A *formula* is a conjunction of clauses: $c_1 \wedge c_2 \wedge \cdots \wedge c_r$.

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

Given a formula, a satisfying assignment is an assignment of `False` or `True` values to the variables so that the formula is true.

Satisfiability

Super important problem in theoretical computer science

SAT problem. Given a formula, decide if it is satisfiable i.e., there is a satisfying assignment.

k-**SAT** problem. A special case of the SAT problem where every clause in the input formula has $\leq k$ literals.

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

Is this a 2-SAT formula? Yes.

Is $x_1 = x_2 = x_3 =$ `True` *a satisfying assignment? No.*

Is there any satisfying assignment? Yes. $x_1 = x_2 = x_3 =$ `False`.

Is the satisfying unique? No.

Satisfiability

Example. $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4)$

Is this a 2-SAT formula? No. *Is this a 3-SAT formula? Yes.*

Is it satisfiable? Yes.

Give an example of a 2-SAT formula that is **not** satisfiable.

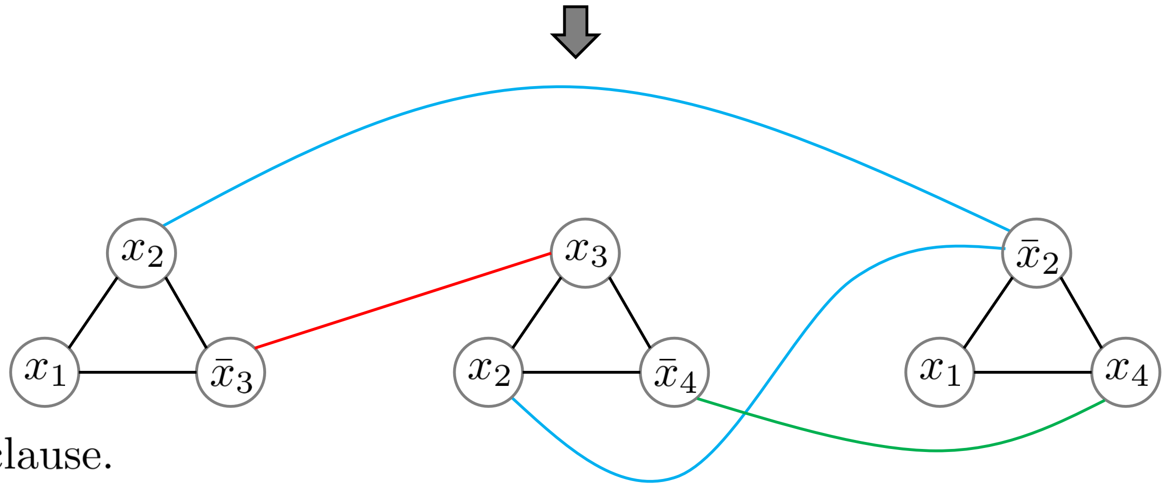
$$(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

Claim. SAT $\leq_P$ Independent Set.

Proof?

Satisfiability

Input formula: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4)$



Complete subgraph

Create a clique for every clause.

Add an edge between every pair of literals that are negations of each other.

Input formula: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$

Construct the graph corresponding to this input formula.

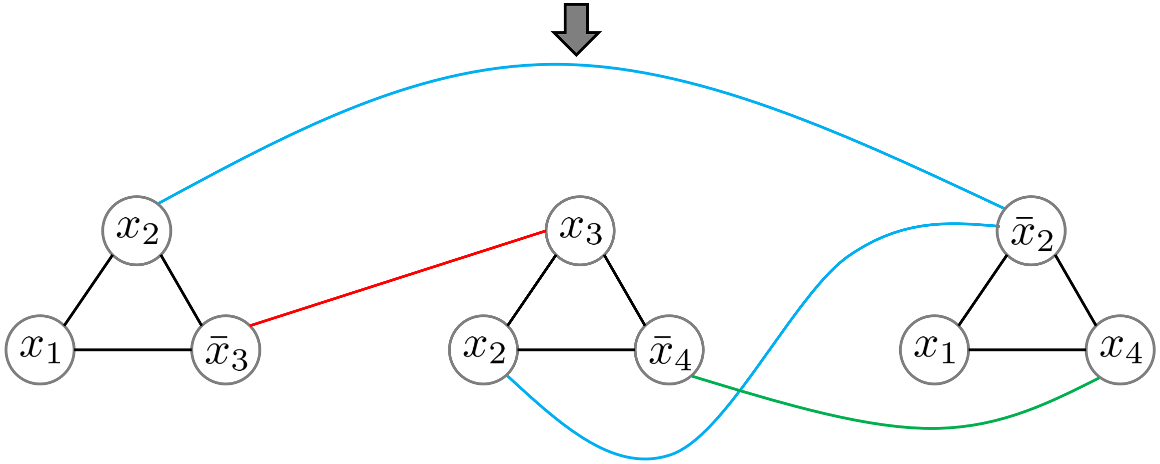
$(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

Satisfiability

Input formula: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4)$

Create a clique for every clause.

Add an edge between every pair of literals that are negations of each other.



What are the properties of an independent set on such a graph?

It cannot

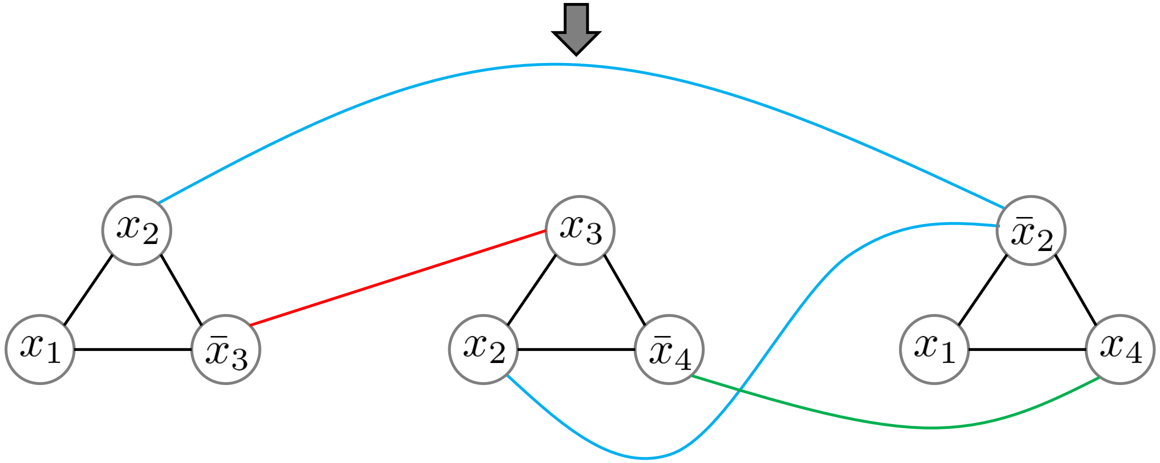
- have a literal and its negation
- 2 literals from the same clause
- Anything else?
- What is the relation between the independent set and the satisfiability of the formula? <u>Homework</u> Proof is on next slides

SATISFIABILITY **Claim.** SAT $\leq_P$ Independent Set.

Input formula: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4)$

Create a clique for every clause.

Add an edge between every pair of literals that are negations of each other.



What are the properties of an independent set on such a graph?

It cannot
- have a literal and its negation
- 2 literals from the same clause
- An independent set of size 3 above will have one literal from each clause and will be a solution

Satisfiability

Claim. The input formula is satisfiable iff the graph constructed has an independent set of size t where t is the number of clauses.

Proof. Given an independent set of size t, we can set the literals they correspond to, to `True`.

This satisfies all clauses since any independent set of size t must contain a node from each of the clauses.

We don't make contradictory assignments because of the edges connecting nodes corresponding to opposite literals.

Similarly, given a satisfying assignment we can get an independent set by picking one literal from each clause that is set to `True`.

■