Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover
2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

CS 401/MCS 401 Lecture 18
Computer Algorithms I
Jan Verschelde, 28 July 2025

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

reformulating problems

The Ford-Fulkerson algorithm computes maximum flow.

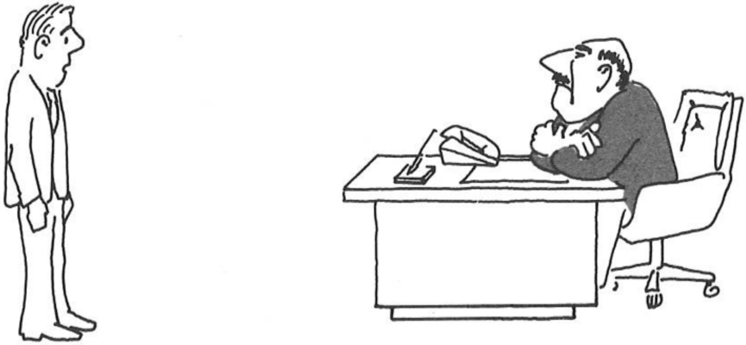
By reduction to a flow problem,
we could solve the following problems:

- bipartite matching,
- circulation with demands,
- edge-disjoint paths,
- survey design, and
- airline scheduling.

Because the Ford-Fulkerson algorithm is an efficient algorithm,
all those problems can be solved efficiently as well.

Our plan for the remainder of the course is to explore
computationally hard problems.

imagine a meeting with your boss ...



"I can't find an efficient algorithm, I guess I'm just too dumb."

From *Computers and intractability. A Guide to the Theory of NP-Completeness* by Michael R. Garey and David S. Johnson, Bell Laboratories, 1979.

what you want to say is

"I can't find an efficient algorithm, because no such algorithm is possible!"

From *Computers and intractability. A Guide to the Theory of NP-Completeness* by Michael R. Garey and David S. Johnson, Bell Laboratories, 1979.

you better have some backup

"I can't find an efficient algorithm, but neither can all these famous people."

From *Computers and intractability. A Guide to the Theory of NP-Completeness* by Michael R. Garey and David S. Johnson, Bell Laboratories, 1979.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

reformulating a problem in polynomial time

If we encounter a difficult problem X, we would like to be able to formally express that *problem X is at least as hard as problem Y.*

If we have a black box capable of solving X, then we can also solve Y.

Definition (1)

Problem Y is *polynomial-time reducible* to X, denoted by $Y \leq_P X$, if arbitrary instances of problem Y can be solved

- using a polynomial number of standard computational steps, and
- a polynomial number of calls to a black box that solves X.

an application of reducibility

Proposition (2)

Assume $Y \leq_P X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.

Proof. If $Y \leq_P X$, then we can solve Y using

1. a polynomial number of standard computational steps, and
2. a polynomial number of calls to a black box that solves X.

If X can be solved in polynomial time, then the black box that solves X runs in polynomial time.
A polynomial number of calls to a black box that runs in polynomial time is bounded by polynomial time because $f(g(x))$ is a polynomial for any two polynomials f and g.
Adding the polynomial that bounds the cost for calling the black box solver to the polynomial number of standard computational steps is again a polynomial. Thus Y can be solved in polynomial time. Q.E.D.

intractability

Definition (3)

A problem is *computationally tractable*
if we have an efficient algorithm to solve it.

A problem that is not computationally tractable is *intractable*.

Proposition (4)

Assume $Y \leq_P X$. If X cannot be solved in polynomial time, then Y cannot be solved in polynomial time.

The statement in the proposition is true because it is contrapositive to the statement in the previous preposition.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

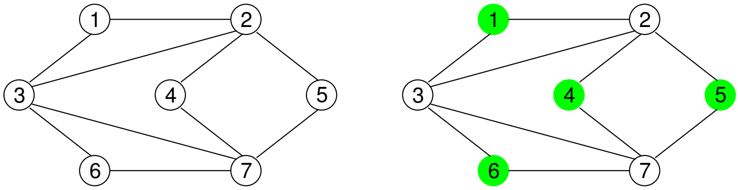
2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

the independent set problem

Definition (5)

Given a graph $G = (V, E)$, a subset $S \subseteq V$ is *an independent set* if for all $u, v \in S$: $(u, v) \notin E$.

Given a graph G and some number k,
does G contain an independent set of size at least k?



apply dynamic programming

Dynamic programming applies to an exhaustive search.

Given a graph $G = (V, E)$ and some $v \in V$,
the size of an independent set that contains v is one
plus the size of the independent set of the graph $G' = (V', E')$

1. $V' = V \setminus \{v\}$,
2. $E' = \{ (u, w) \in E \mid u \neq v, w \neq v \}$.

Exercise 1:

Apply dynamic programming to the independent set problem.

1. Develop a recurrence based on the definition of G' above.
2. Given k, the size of an independent set,
describe the algorithm to compute an independent set of size k,
or to return nothing if there is no independent set of size k.
3. What is the running time of this algorithm?

optimization and decision questions

The decision question for the Independent Set Problem:

> Given a graph G and some number k,
> does G contain an independent set of size at least k?

The optimization question for the Independent Set Problem:

> Given a graph G, find the largest independent set in G.

From the polynomial-time solvability, both problems are equivalent:

- A method to solve the optimization question also solves the decision question for any value of k.
- A method to solve the decision question can be combined with binary search with $O(\log(n))$, $n = \#V$, different values of k.

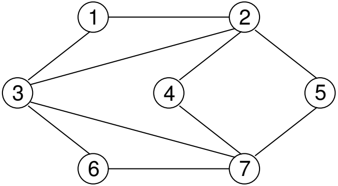
Decision questions are more convenient to work with.

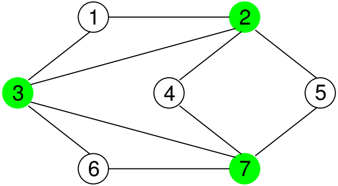
the vertex cover problem

Definition (6)

Given a graph $G = (V, E)$, a subset $S \subseteq V$ is *a vertex cover* if for all $(u, v) \in E$: $u \in S$ or $v \in S$.

Given a graph G and some number k,
does G contain a vertex cover of size at most k?





apply dynamic programming

Dynamic programming applies to an exhaustive search.

Given a graph $G = (V, E)$ and some $v \in V$,
the size of a vertex cover that does not contain v is one
plus the size of the vertex cover of the graph $G' = (V', E')$

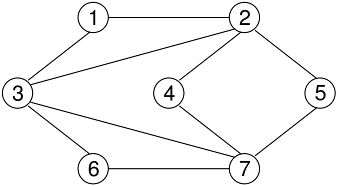
1. $V' = V \setminus \{v\}$,
2. $E' = \{ (u, w) \in E \mid u \neq v, w \neq v \}$.

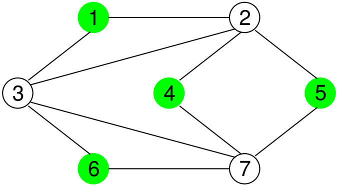
Exercise 2:
Apply dynamic programming to the vertex cover problem.

1. Develop a recurrence based on the definition of G' above.
2. Given k, the size of a vertex cover
describe the algorithm to compute a vertex cover of size k,
or to return nothing if there is no vertex cover of size k.
3. What is the running time of this algorithm?

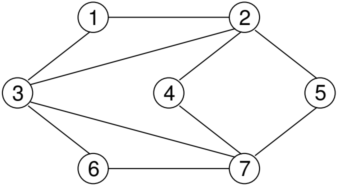
independent set is equivalent to vertex cover

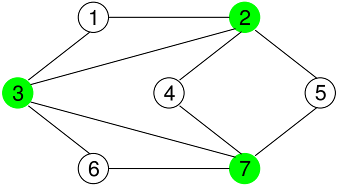
An example of the independent set problem:





An example of the vertex cover problem:





independent set is equivalent to vertex cover

A subset $S \subseteq V$ is a vertex cover if for all $(u, v) \in E$: $u \in S$ or $v \in S$.

A subset $S \subseteq V$ is an independent set if for all $u, v \in S$: $(u, v) \notin E$.

Theorem (7)

Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V \setminus S$ is a vertex cover.

Proof. If and only if means $\Rightarrow$ and $\Leftarrow$.

- $\Rightarrow$: Let S be an independent set. Consider any edge (u, v).

  Because an independent set contains vertices which are not connected by an edge, u and v cannot be both in S.

  Therefore, u or v must be in $V \setminus S$. So the edge (u, v) has at least one vertex in $V \setminus S$. Thus $V \setminus S$ is a vertex cover.

vertex cover is equivalent to independent set

A subset $S \subseteq V$ is a vertex cover if for all $(u, v) \in E$: $u \in S$ or $v \in S$.

A subset $S \subseteq V$ is an independent set if for all $u, v \in S$: $(u, v) \not\in E$.

Theorem (8)

Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V \setminus S$ is a vertex cover.

Proof. If and only if means $\Rightarrow$ and $\Leftarrow$.

- $\Leftarrow$: Let $V \setminus S$ be a vertex cover. Consider any $u, v \in S$, and note that u and v are not part of the vertex cover.

  If $(u, v) \in E$, then $V \setminus S$ would not be a vertex cover as (u, v) is an edge with u and v not in the vertex cover.

  So $(u, v) \not\in E$, for any pair $u, v \in S$.
  Thus S is an independent set. Q.E.D.

independent set reduces to vertex cover

Corollary (9)

Independent Set $\leq_P$ *Vertex Cover*

This means that arbitrary instances of the independent set problem can be solved using a polynomial number of standard computational steps and a polynomial number of calls to the black box solver that solves the vertex cover problem.

Proof. If we have a black box solver for the vertex cover problem,

then whether the graph has an independent set of size at least k

can be decided by asking the black box whether the graph has

a vertex cover of size at most $n - k$,

where n equals the number of vertices. Q.E.D.

vertex cover reduces to independent set

Corollary (10)

Vertex Cover $\leq_P$ *Independent Set*

This means that arbitrary instances of the vertex cover problem can be solved using a polynomial number of standard computational steps and a polynomial number of calls to the black box solver that solves the independent set problem.

Proof. If we have a black box solver for the independent set problem,

then whether the graph has a vertex cover of size at most k

can be decided by asking the black box whether the graph has

an independent set of size at least $n - k$,

where n equals the number of vertices. Q.E.D.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

the set cover problem

Independent set and vertex cover can be viewed as

- Independent set can be viewed as a *packing problem*.
  The goal is to *pack in* as many vertices as possible,
  avoiding conflicts represented by edges.
- Vertex cover can be viewed as a *covering problem*.
  The goal is to *cover* all edges in the graph,
  with as few as possible vertices.

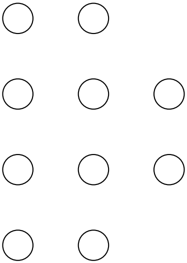
Vertex cover is a special instance of the Set Cover problem:

Given are $U = \{ 1, 2, \ldots, n \}$,
a collection $S_i \subseteq U$, $i = 1, 2, \ldots, m$,
and some number k.

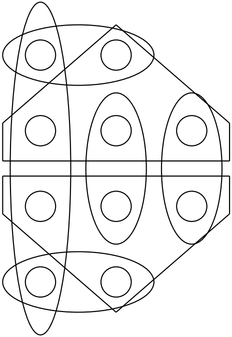
Is there an index set $I = \{ i_1, i_2, \ldots, i_k \}$: $\displaystyle\bigcup_{j \in I} S_j = U$?

an application of the set cover problem

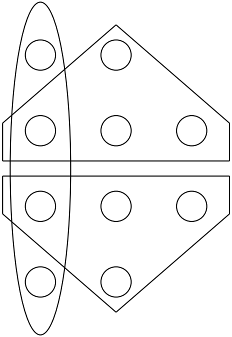
The set U are n capabilities we want to cover with
a subset of size k of m available software systems S_i,
each software system has capabilities $S_i \subseteq U$.



capabilities



all software



a set cover

reducing vertex cover to set cover

Theorem (11)

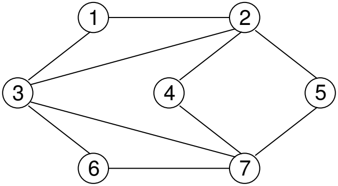
Vertex Cover $\leq_P$ Set Cover

Proof. An arbitrary instance of vertex cover is given by a graph $G = (V, E)$ and a number k. We have a black box to solve set cover.

Given $G = (V, E)$, define the input for a set cover as follows:

- $U = E$, and
- for every vertex i: S_i contains all edges incident to i.

from vertex cover to set cover



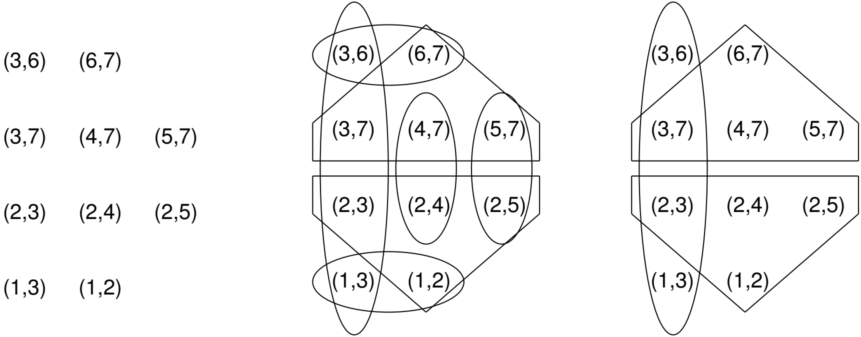
$U = \{(1,2),(1,3),(2,3),(2,4),(2,5),(3,6),(3,7),(4,7),(5,7),(6,7)\}$.

$S_1 = \{(1,2),(1,3)\}$, $S_2 = \{(1,2),(2,3),(2,4),(2,5)\}$,
$S_3 = \{(1,3),(2,3),(3,6),(3,7)\}$, $S_4 = \{(2,4),(4,7)\}$,
$S_5 = \{(2,5),(5,7)\}$, $S_6 = \{(3,6),(6,7)\}$,
$S_7 = \{(3,7),(4,7),(5,7),(6,7)\}$.

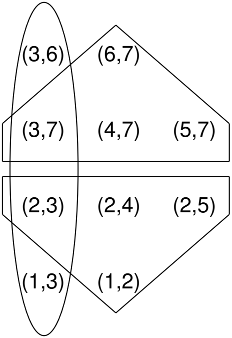
the corresponding set cover problem

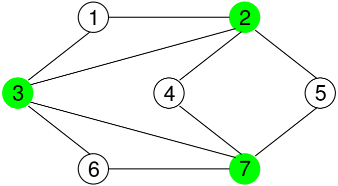
$$U = \{(1,2),(1,3),(2,3),(2,4),(2,5),(3,6),(3,7),(4,7),(5,7),(6,7)\}.$$

$S_1 = \{(1,2),(1,3)\}$, $S_2 = \{(1,2),(2,3),(2,4),(2,5)\}$,
$S_3 = \{(1,3),(2,3),(3,6),(3,7)\}$, $S_4 = \{(2,4),(4,7)\}$,
$S_5 = \{(2,5),(5,7)\}$, $S_6 = \{(3,6),(6,7)\}$,
$S_7 = \{(3,7),(4,7),(5,7),(6,7)\}$.



set cover solves vertex cover





Lemma (12)

U can be covered by at most k of the sets S_1, S_2, ..., S_m if and only if G has a vertex cover of size at most k.

Proof. If and only if means $\Rightarrow$ and $\Leftarrow$.

- $\Rightarrow$: If the sets S_{i_1}, S_{i_2}, ..., S_{i_ℓ}, for $\ell \leq k$ cover U, then every edge in G is incident to one of the vertices i_1, i_2, ..., i_ℓ. So the set $\{ i_1, i_2, \ldots, i_\ell \}$ is a vertex cover in G of size $\ell \leq k$.
- $\Leftarrow$: If the set $\{ i_1, i_2, \ldots, i_\ell \}$ is a vertex cover, $\ell \leq k$, then the sets S_{i_1}, S_{i_2}, ..., S_{i_ℓ} cover U. Q.E.D.

Thus the theorem Vertex Cover $\leq_P$ Set Cover follows from

1. using (V, E) to formulate the instance of the Set Cover problem,
2. pass the input to the Set Cover problem to the black box, and
3. answer yes if and only if the black box solver answers yes.

Q.E.D.

the set packing problem

The independent set problem generalizes to the set packing problem:

> Given a set U of n elements,
> a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and
> some number k,
> does there exists a collection of at least k of these sets
> so that no two of them intersect?

As an application, consider U as a set of non-sharable resources, and a set of m processes S_i, the i-th process requires $S_i \subseteq U$ resources to run, and some number k.
Can you run at least k of the processes?

Exercise 3:
Show that Independent Set $\leq_P$ Set Packing.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

CS 401/MCS 401 Lecture 18
Computer Algorithms I
Jan Verschelde, 28 July 2025

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

Boolean expressions

A Boolean variable takes values in $\{ 0, 1 \}$, *false*, or *true*.

Operations on Boolean variables x and y are

- $\overline{x} :=$ not x, the negation, $1 + x \bmod 2$
- $x \vee y := x$ or y, the disjunction, $x + y \bmod 2$
- $x \wedge y := x$ and y, the conjunction, $x \times y$

Those 3 operations are realized by 3 basic logic gates.

Definition (13)

Let $X = \{ x_1, x_2, \ldots, x_n \}$ be a set of Boolean variables.
A *clause* C is a disjunction of distinct terms

$$C = t_1 \vee t_2 \vee \cdots \vee t_\ell, \quad t_i = x \text{ or } t_i = \overline{x}, \text{ for } x \in X.$$

A clause corresponds to a circuit composed of logic gates: or, not.

assignments satisfy clauses

For which inputs do we obtain one as output?

Definition (14)

Let $X = \{ x_1, x_2, \ldots, x_n \}$ be a set of Boolean variables.
A *truth assignment for X* is a function ν:

$$\begin{array}{rcccc} \nu & : & X & \rightarrow & \{ 0, 1 \} \\ & & x & \mapsto & \nu(x). \end{array}$$

Is there an assignment of the variables for which a clause is true?

Definition (15)

Given a clause C and an assignment ν for the variables in C,
we say that *ν satisfies C* if the application of the rules of Boolean logic causes C to evaluate to 1.

satisfiability

A clause applies or, not operations to Boolean variables.

For a collection of clauses, we can ask whether there is an assignment of the variables that makes all of them true.

Definition (16)

Given a collection of clauses $C_1, C_2, \ldots, C_k$, and an assignment ν for the variables in the clauses, we say that *ν is a satisfying assignment with respect to $C_1, C_2, \ldots, C_k$* if the application of the rule of Boolean logic causes all C_i to evaluate to 1, or equivalently, if

$$C_1 \wedge C_2 \wedge \cdots \wedge C_k \text{ evaluates to 1.}$$

Then we say that $C_1 \wedge C_2 \wedge \cdots \wedge C_k$ is *satisfiable*.

an example

Consider for example

$$(x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_3}), (x_2 \vee \overline{x_3}).$$

The assignment ν: $\nu(x_1) = 1$, $\nu(x_2) = 1$, $\nu(x_3) = 1$ results in

$$(1 \vee 0) \wedge (0 \vee 0) \wedge (1 \vee 0) = 0$$

and is therefore not a satisfying assignment.

The assignment ν: $\nu(x_1) = 0$, $\nu(x_2) = 0$, $\nu(x_3) = 0$ results in

$$(0 \vee 1) \wedge (1 \vee 1) \wedge (0 \vee 1) = 1$$

and is therefore a satisfying assignment.

The set of clauses $(x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_3}), (x_2 \vee \overline{x_3})$ is satisfiable.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

the satisfiability and 3-satisfiability problems

The satisfiability problem, or SAT for short is

> Given $X = \{ x_1, x_2, \ldots, x_n \}$ a set of Boolean variables, and
> C_1, C_2, ..., C_k, a set of clauses over X,
> does there exist a satistying truth assignment?

A special case which is equally difficult is the 3-SAT problem.

The 3-satisfiability problem, or 3-SAT for short is

> Given $X = \{ x_1, x_2, \ldots, x_n \}$ a set of Boolean variables, and
> C_1, C_2, ..., C_k, a set of clauses over X,
> where each clause C_i has *exactly three terms*,
> does there exist a satistying truth assignment?

The combinatorial problem is that we have to make n independent decisions about the value (zero or one) of the n Boolean variables in order to satisfy a set of constraints simultaneously.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

reducing 3-SAT to independent set

Given a graph $G = (V, E)$, a subset $S \subseteq V$ is an independent set if for all $u, v \in S$: $(u, v) \notin E$. Given a graph G and some number k, does G contain an independent set of size at least k?

Theorem (17)

3-SAT $\leq_P$ Independent Set

Proof. We have a black box for independent set and an instance of 3-SAT consisting of k clauses C_1, C_2, ..., C_k over $X = \{ x_1, x_2, \ldots, x_n \}$.

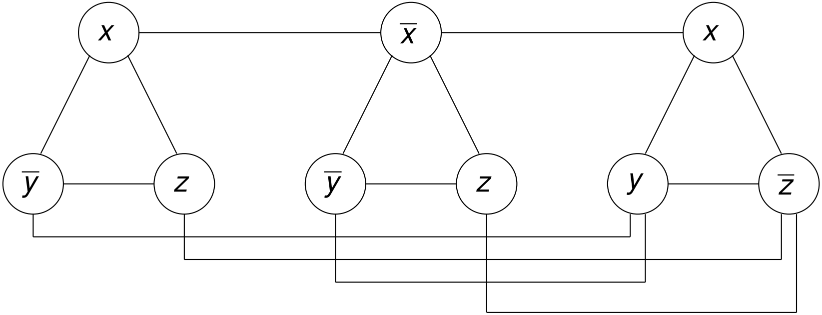
We will formulate the input to 3-SAT as an independent set problem.

satisfying clauses and independent sets

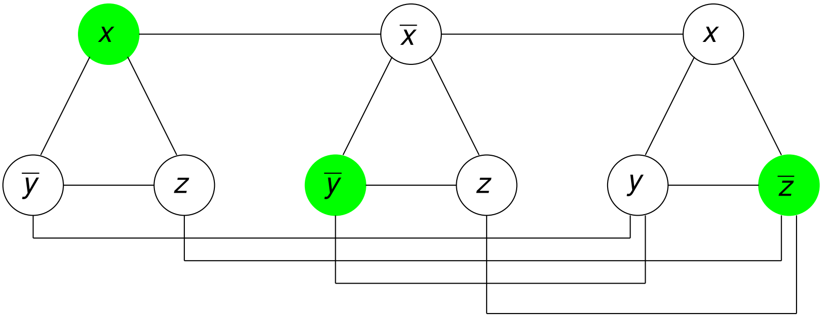
Consider for example

$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z})$$

and the corresponding graph:



independent set solves satisfiability



Take $x = 1$, $\overline{y} = 1$, $\overline{z} = 1$, and

$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z})$$

is satisfied. Observe: the green nodes define an independent set.

defining the corresponding graph G

Consider the Boolean variables as vertices

$$(v_{1,1} \vee v_{1,2} \vee v_{1,3}) \wedge (v_{2,1} \vee v_{2,2} \vee v_{2,3}) \wedge \cdots \wedge (v_{k,1} \vee v_{k,2} \vee v_{k,3}).$$

Every clause $v_{i,1} \vee v_{i,2} \vee v_{i,3}$ corresponds to a triangle

- with vertices $v_{i,1}$, $v_{i,2}$, $v_{i,3}$
- and edges $(v_{i,1}, v_{i,2})$, $(v_{i,1}, v_{i,3})$, $(v_{i,2}, v_{i,3})$.

For a clause to be true, it suffices that one term is true.

For every pair of clauses (i, j), we add the edge (u, v)

- if $\overline{u} = v$ or $u = \overline{v}$,
- for $u \in \{ v_{i,1}, v_{i,2}, v_{i,3} \}$ and $v \in \{ v_{j,1}, v_{j,2}, v_{j,3} \}$.

The graph G corresponding to k clauses has $3k$ vertices and edges defined by the above rules.

another numerical example

Exercise 4:
Consider the expression

$$(x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee y \vee z).$$

1. Define the corresponding graph.
2. Does there exist a satisying truth assignment?
   Justify your answer.

satisfiability implies an independent set

Lemma (18)

The 3-SAT problem with k clauses is satisfiable if and only if the corresponding graph G has an independent set of size at least k.

Proof. If and only if means $\Rightarrow$ and $\Leftarrow$.

- $\Rightarrow$: If 3-SAT is satisfiable, then every triangle contains at least one node with value equal to one.
  Let S collect the nodes of every triangle with value equal to one.
  We claim that S is an independent set.

  If for any $u, v \in S$, there would be an edge (u, v) in the graph.
  But if there is an edge (u, v), then the values at the nodes conflict, which contradicts the satisfiability.
  Thus for any $u, v \in S$, there is no edge (u, v)
  and thus S is an independent set of size k.

independent set implies satisfiability

Proof continued.

- $\Leftarrow$ If we have an independent set S of size k, then we construct an assignment ν as follows:
  - If x_i appears in S, then we set $\nu(x_i) = 1$.
  - If $\overline{x_i}$ appears in S, then we set $\nu(x_i) = 0$.

  By the construction of the graph, we can have at most one vertex from every triangle.
  Because $\#S = k$, we have assigned one in every clause, so the clause will evaluate to one and we obtained satisfiability. Q.E.D.

Thus the theorem 3-SAT $\leq_P$ Independent Set follows from

1. the construction of the corresponding graph G,
2. calling the black box for independent set on G, and
3. answering yes if and only if the black box solver answers yes.

Q.E.D.

Reductions and Satisfiability

1. Polynomial-Time Reductions
   - reformulating problems
   - reformulating a problem in polynomial time
   - independent set and vertex cover
   - reducing vertex cover to set cover

2. The Satisfiability Problem
   - satisfying truth assignments
   - SAT and 3-SAT
   - reducing 3-SAT to independent set
   - transitivity of reductions

transitivity of reductions

Theorem (19)

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

Proof. The result follows from a composition of black boxes:

- To solve an instance of Z, we call the black box for Y.
- To solve an instance of Y, we call the black box for X.

The polynomial number of computational steps in the definition of $Z \leq_P X$ is absorbed by the calls to the black box for Y. Q.E.D.

We have proved

$$\text{3-SAT} \leq_P \text{Independent Set} \leq_P \text{Vertex Cover} \leq_P \text{Set Cover},$$

we have therefore also

$$\text{3-SAT} \leq_P \text{Set Cover}.$$

3-SAT and the set packing problem

Exercise 5:

Formulate the relationship between 3-SAT and the Set Packing problem.
Illustrate your formulation with an example.

Justify your answer.