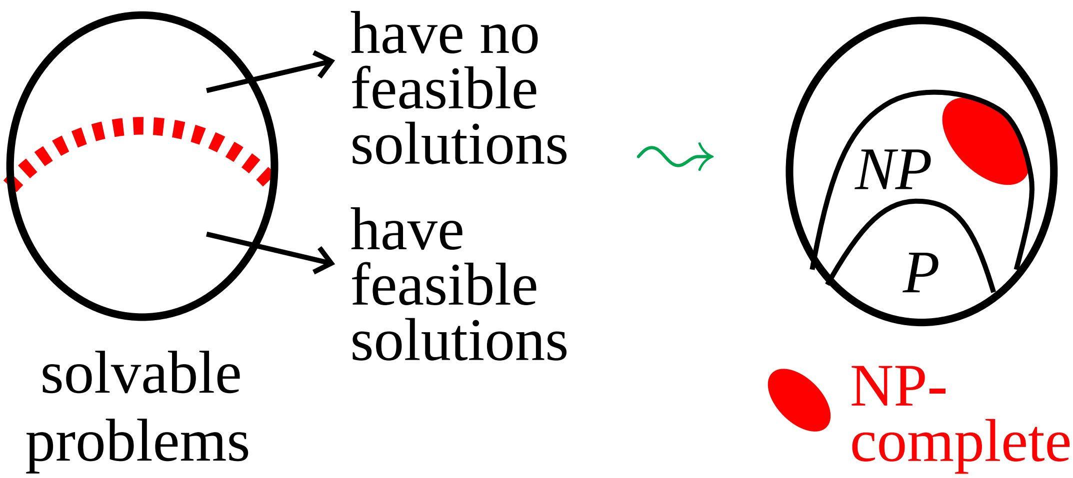
NP-completeness (review)



$$L \in \mathcal{NPC} \Leftrightarrow \begin{array}{l} L \in \mathcal{NP} \text{and} \\ L \in \mathcal{NP}\text{-hard} \end{array}$$

Today: Proving $\mathcal{NP}$-completeness

- **$L \in \mathcal{NP}$:** show that there is a "short"† **certificate** of membership in L ("id card").
- **$L \in \mathcal{NP}$-hard:** show that there is an "efficient"† **reduction** from a known $\mathcal{NP}$-hard problem L_{np} to L.

† polynomial (length, time …)

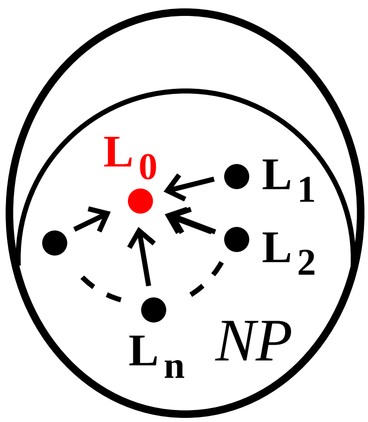
Skills to learn

- Transforming problems into each other.

Insight to gain

- Seeing unity in the midst of diversity: A variety of graph-theoretical, numerical, set & other problems are just variants of one another.

But before we can use reductions we need **the first $\mathcal{NP}$-hard problem**.



SATISFIABILITY (SAT)

Example

$I = C \cup U$
$C = \{(x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_2), (x_1 \vee x_2)\}$
$U = \{x_1, x_2\}$

$T = x_1 \mapsto \text{TRUE}, x_2 \mapsto \text{FALSE}$ is a satisfying truth assignment. Hence the given instance I is **satisfiable**, i.e. $I \in \text{SAT}$.

Further (basic) reductions

BOUNDED HALTING

↓

SATISFIABILITY (SAT)

↓

3SAT

↙ ↘

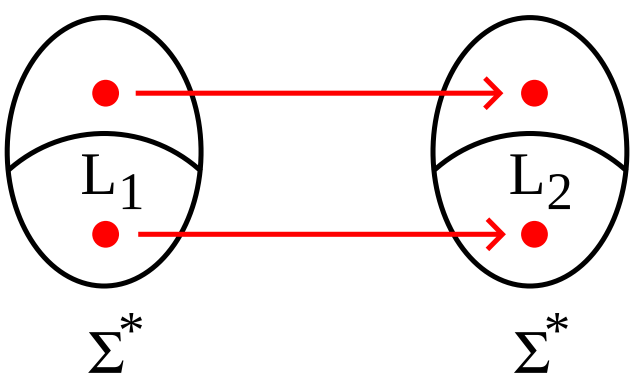
3-DIMENSIONAL MATCHING (3DM) → PARTITION

VERTEX COVER (VC) → HAMILTONICITY, CLIQUE

Polynomial-time reductions (review)

$L_1 \propto L_2$ means that

- $R : \sum^* \to \sum^*$ such that
  $x \in L_1 \Rightarrow f_R(x) \in L_2$ and
  $x \notin L_1 \Rightarrow f_R(x) \notin L_2$



- $R \in P_f$, i.e. $R(x)$ is polynomial computable

SATISFIABILITY $\propto$ 3-SATISFIABILITY

SAT		**3SAT**
Clauses with any number of literals	$\longmapsto$	Clauses with exactly 3 literals

- C_j is the j'th SAT-clause, and $C_j^{'}$ is the corresponding 3SAT-clauses.
- y_j are new, fresh variables, only used in $C_j^{'}$.

$\boldsymbol{C_j}$		$\boldsymbol{C_j^{'}}$
$(x_1 \vee x_2 \vee x_3)$	$\longmapsto$	$(x_1 \vee x_2 \vee x_3)$
$(x_1 \vee x_2)$	$\longmapsto$	$(x_1 \vee x_2 \vee y_j), (x_1 \vee x_2 \vee \neg y_j)$
(x_1)	$\longmapsto$	$(x_1 \vee y_j^1 \vee y_j^2), (x_1 \vee \neg y_j^1 \vee y_j^2),$ $(x_1 \vee y_j^1 \vee \neg y_j^2), (x_1 \vee \neg y_j^1 \vee \neg y_j^2)$
$(x_1 \vee \cdots \vee x_8)$	$\longmapsto$	$(x_1 \vee x_2 \vee y_j^1), (\neg y_j^1 \vee x_3 \vee y_j^2),$ $(\neg y_j^2 \vee x_4 \vee y_j^3), (\neg y_j^3 \vee x_5 \vee y_j^4),$ $(\neg y_j^4 \vee x_6 \vee y_j^5), (\neg y_j^5 \vee x_7 \vee x_8)$

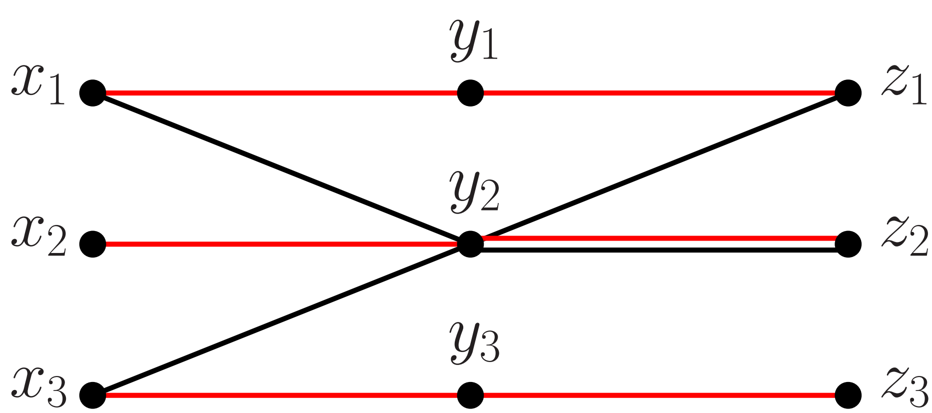
Question: Why is this a proper reduction?

3-DIMENSIONAL MATCHING (3DM)

Instance: A set M of triples (a, b, c) such that $a \in A$, $b \in B$, $c \in C$. All 3 sets have the same size q ($|A| = |B| = |C| = q$).

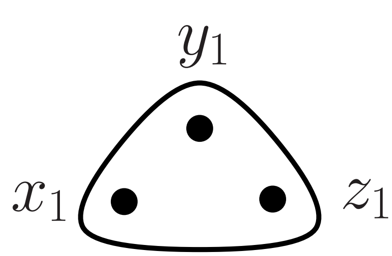
Question: Is there a **matching in M**, i.e. a subset $M' \subseteq M$ such that every element of A, B and C is part of exactly 1 triple in M'?

Example



$$M = \{(x_1, y_1, z_1), (x_1, y_2, z_2), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_3, y_2, z_1)\}$$

We will use sets with 3 elements to visualize triples:



Reductions are like translations from one language to another. The same properties must be expressed.

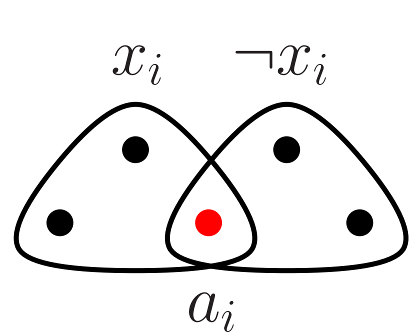
3SAT $\propto$ 3DM

3SAT			**3DM**	
variables	$x_1, \cdots, x_n$	$\longmapsto$	**variables**	$x_3^j, a_3^j, b_j^2, c_k^1$
literals	$x_1, \neg x_1$	$\longmapsto$	**variables**	$x_1^j, \neg x_1^j$
clauses		$\longmapsto$	**triples**	(x_1^j, b_j^1, b_j^2)
$C_j = (x_1 \vee \neg x_2 \vee \neg x_3)$				$(\neg x_3^j, b_j^1, b_j^2)$
"There exists a sat. truth assignment"		$\longmapsto$	"There is a matching"	

"There is a truth assignment T"

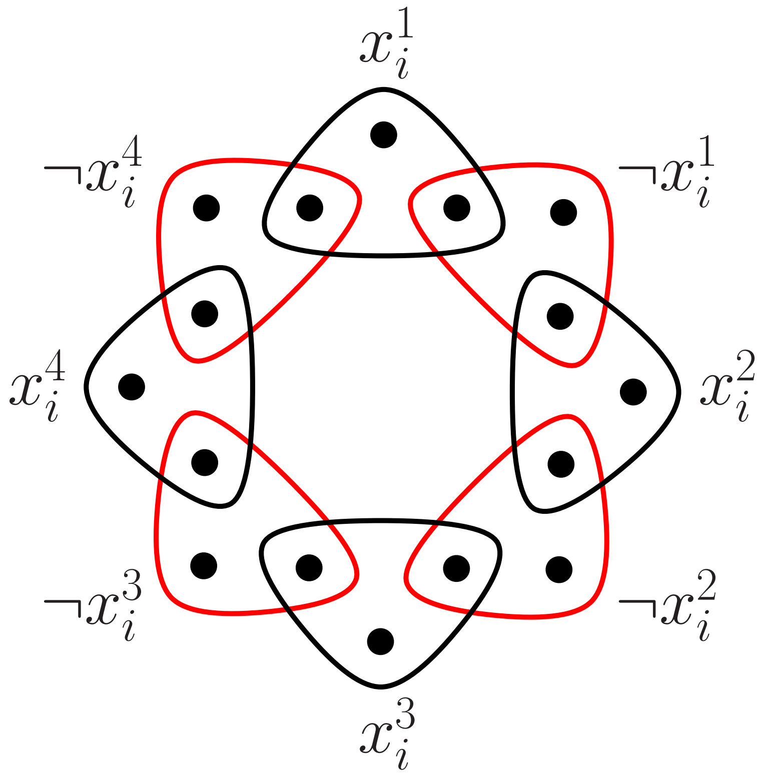
- $\exists T : \{x_1, \cdots, x_n\} \to \{\text{TRUE}, \text{FALSE}\}$
- $T(x_i) = \text{TRUE} \Leftrightarrow T(\neg x_i) = \text{FALSE}$

The second property is easily translated to the 3DM-world:



$T(X_i) = \text{TRUE} \quad \longmapsto \quad x_i$is not "married"

A literal x_i can be used in many clauses. In 3DM we must have as many copies of x_i as there are clauses:



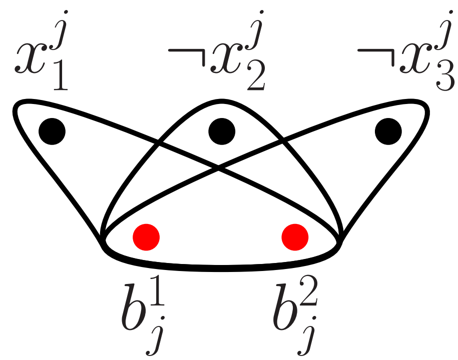
- Either all the black triples must be chosen ("married") or all the red ones!

- If $T(x_i) = \text{TRUE}$ then we choose all the red triples, and the black copies of x_i are free to be used later in the reduction. And vice versa.

- We make one such **truth setting component** for each variable x_i in 3SAT.

“T is satisfying”

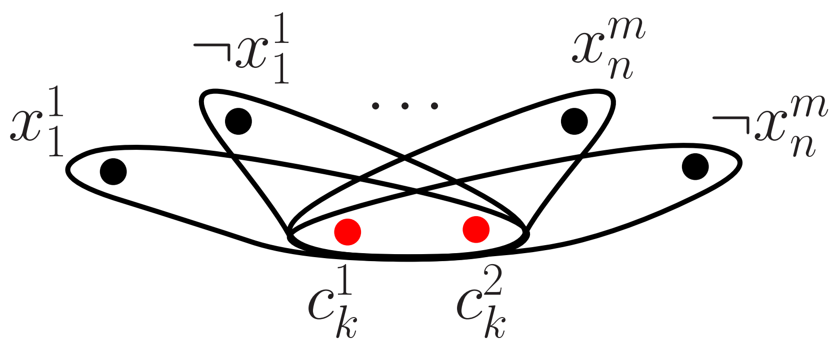
We translate each clause (example: $C_j = (x_1 \vee \neg x_2 \vee \neg x3))$ into 3 triples:



- b_j^1 and b_j^2 can be married if and only if at least one of the literals in C_j is not married in the truth setting component.
- If we have a satisifiable 3SAT-instance , then all b_j^1 and b_j^2-variables $(1 \leq j \leq m)$ can be married.
- If we have a negative 3SAT-instance , then some b_j^1 and b_j^2-variables will not be married.

Cleaning up ("Garbage collection")

There are many x_i^j who are neither married in the truth settting components nor in the "clause-satisfying" part. We introduce a number of fresh c-variables who can marry "everybody":



- There are $m \times n$ unmarried x-variables after the truth setting part.
- If all m clauses are satisfiable then there will remain $(m \times n) - m = m(n-1)$ unmarried x-variables.
- So we let $1 \leq k \leq m(n-1)$.

Partition

Instance: A finite set A and sizes $s(a) \in \mathbb{Z}^+$ for each $a \in A$.

Question: Can we **partition** the set into two sets that have equal size, i.e. is there a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$$

3DM $\propto$ Partition

We first reduce 3DM to Subset Sum where we are given A, as in Partition, but also a number B, and where we are asked if it is possible to choose a subset of A with sizes that add up to B.

3DM		Subset Sum
sets and triples (subsets)	$\longmapsto$	numbers
"There is a matching M'"	$\longmapsto$	"There is a subset with total size B"

Difficulty: We need to translate from subsets with 3 elements (triples) to numbers.

Solution: Use the **characteristic function** of a set!

Example

Given set $U = \{x_1, x_2, \ldots, x_n\}$ and subset $S = \{x_1, x_3, x_4\}$. The characteristic function of S is a binary number with n digits and bit 1, 3 and 4 set to 1: $\underbrace{101100\cdots0}_{n}$.

There is a matching $M' \longleftrightarrow$ There is a subset M' $\sum_{M'}$ sizes $= B$

It is natural to set $B = \overbrace{111\cdots11}^{n}$, since each element in the universe is used in exactly one of the triples in the matching.

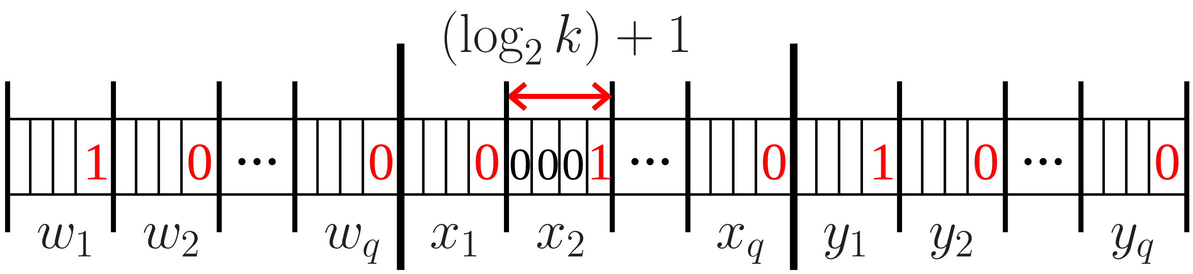
Technicality: Carry bits!

$01_b + 10_b = 11_b$, but also $01_b + 01_b + 01_b = 11_b$.

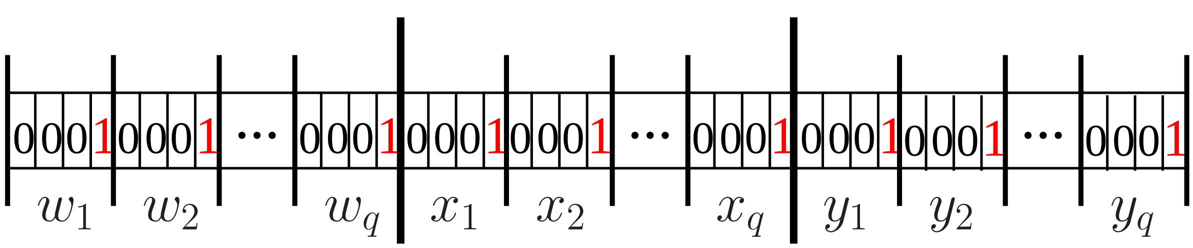
3DM-instance:

$M \subseteq W \times X \times Y$
$W = \{w_1, w_2, \cdots, w_q\}$
$Y = \{y_1, y_2, \cdots, y_q\}$
$Z = \{z_1, z_2, \cdots, z_q\}$
$M = \{m_1, m_2, \cdots, m_k\}$

- For each triple $m_i \in M$ we construct a binary number:

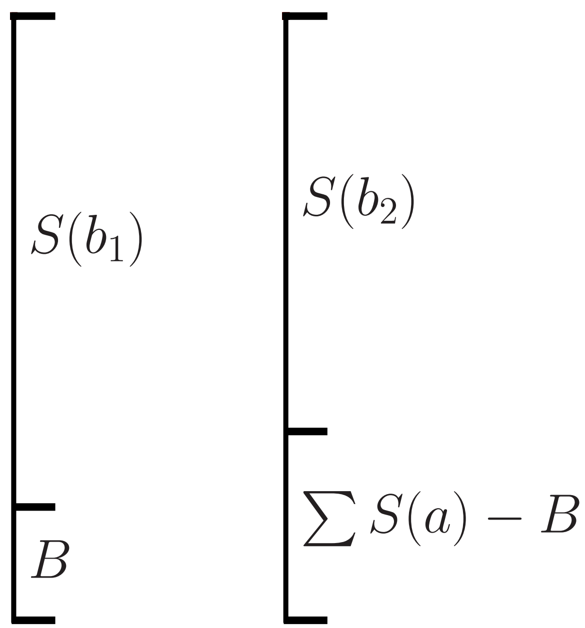


- This PARTITION/SUBSET SUM number corresponds to the triple (w_1, x_2, y_1).
- By adding $\log_2 k$ zeros between every "characteristic digit", we eliminate potential summation problems due to overflow / carry bits.
- We make B as follows:



Subset Sum $\propto$ Partition

- We introduce two new elements b_1 and b_2.
- We choose $s(b_1)$ and $s(b_2)$ so big that every partition into to equal halves must have $s(b_1)$ in one half and $s(b_2)$ in the other.



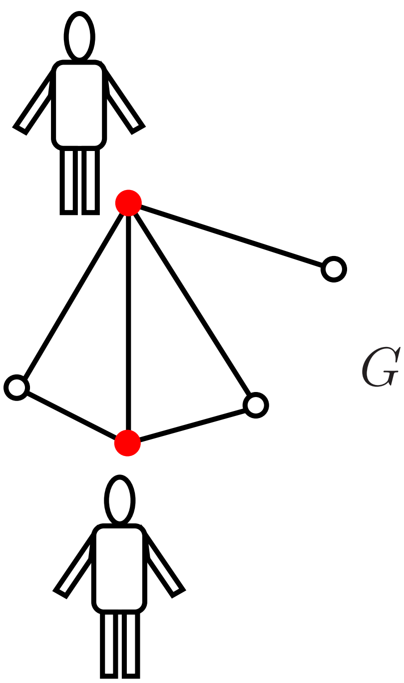
- We let $s(b_1) + B = s(b_2) + (\sum s(a) - B)$.
- We can pick a subset of A which adds up to B if and only if we can split $A \cup \{b_1, b_2\}$ into two equal halves.

VERTEX COVER (VC)

Instance: A graph G with a set of vertices V and a set of edges E, and an integer $K \leq |V|$.

Question: Is there a **vertex cover** of G of size $\leq K$?

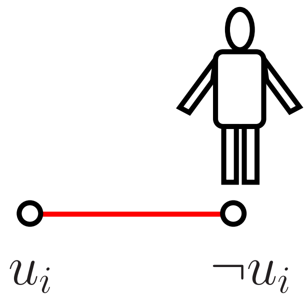
"Can we place guards on at most K of the intersections (vertices) such that all the streets (edges) are surveyed?"



3SAT $\propto$ VC

3SAT		**VERTEX COVER**
literals	$\longmapsto$	vertices
clauses	$\longmapsto$	subgraphs
"There exists a sat. truth assignment"	$\longmapsto$	"There is a VC of size K"

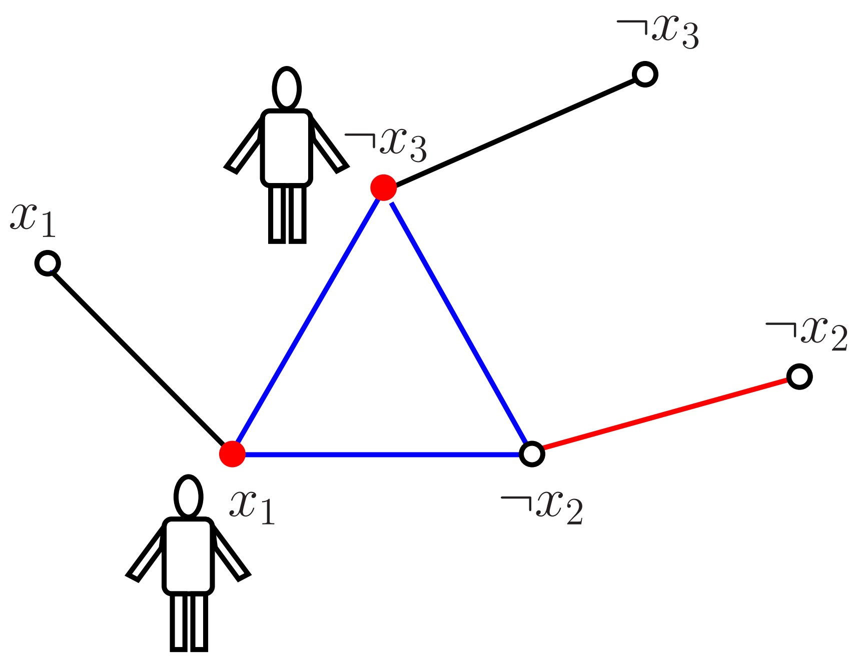
literals $\longmapsto$ vertices



- A guard must be placed in either u_i or $\neg u_i$ for the street between u_i and $\neg u_i$ to be surveyed.
- If we only allow $|V|$ guards to be used for all $|V|$ streets of this kind, then we cannot place guards at both ends.
- Placing a guard on u_i corresponds to the 3SAT-literal u_i being TRUE.
- Placing a guard on $\neg u_i$ corresponds to the 3SAT-literal $\neg u_i$ being TRUE (and the u_i-variable being assigned to FALSE).

clause $\longmapsto$ subgraph

For clause $C_j = (x_1 \vee \neg x_2 \vee \neg x_3)$ we make the following subgraph:

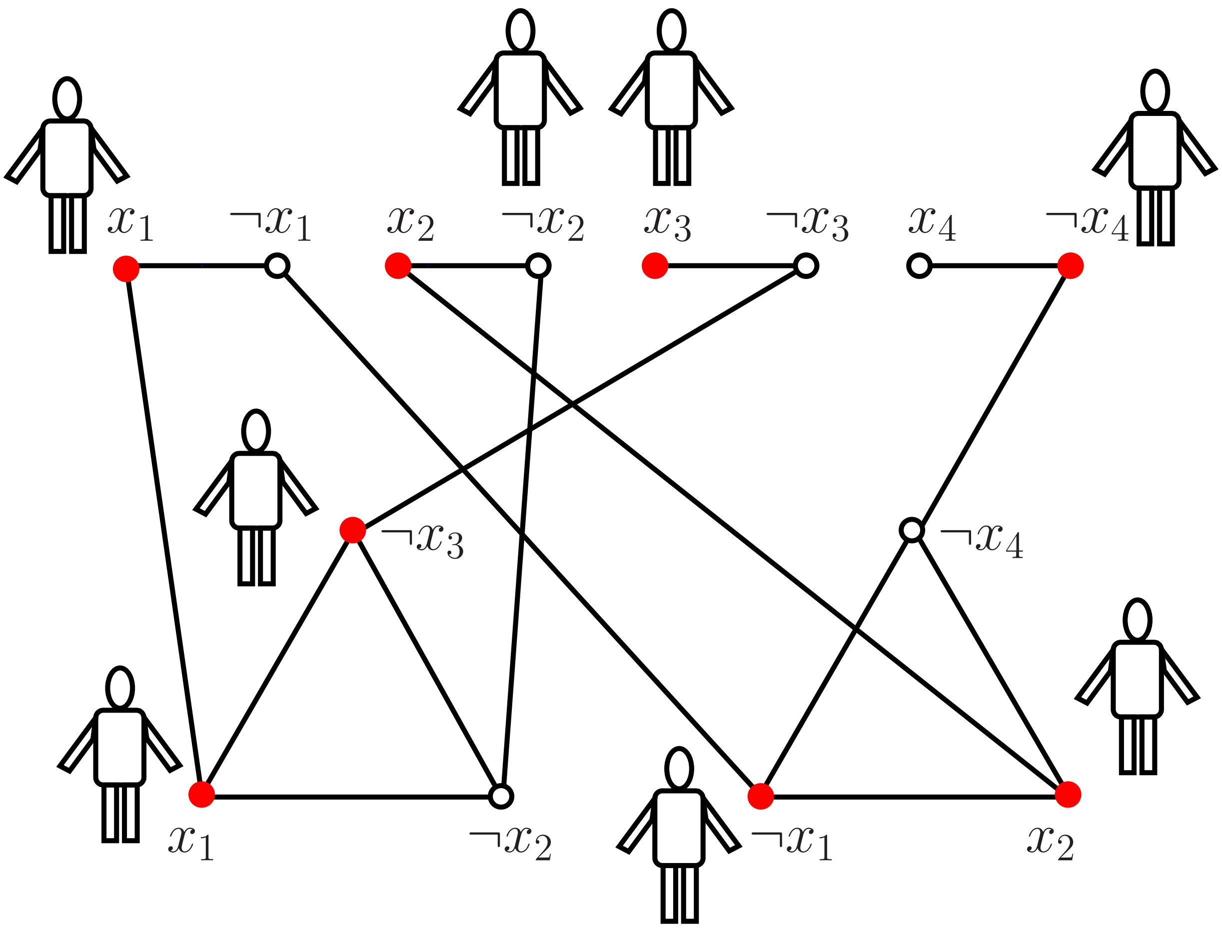


- We need guards on two of three nodes in the triangle to cover all three (blue) edges.
- If we are allowed to place only two guards per triangle, then we cannot cover all three outgoing edges.
- All 6 edges can be covered if and only if at least one edge (red) is covered from the outside vertex.
- By connecting the subgraph to the "truth-setting" components, this translates to one of the literals being TRUE (guarded)!

Example

3SAT-instance:

$U = \{x_1, x_2, x_3, x_4\} \qquad (n = 4)$
$C = \{\{x_1, \neg x_2, \neg x_3\}, \{\neg x_1, x_2, \neg x_4\}\} \qquad (m = 2)$



- Total number of guards $K = n + 2m = 8$.
- Should check that the reduction can be computed in time polynomial in the length of the 3SAT-instance …

VERTEX COVER, CLIQUE AND INDEPENDENT SET

For $G = (V, E)$ and subset $V_1 \subset V$, the following statements are equivalent:

(a) V_1 is a vertex cover of G

(b) $V - V_1$ is an independent set in G

(c) $V - V_1$ is a clique in G^c.

Corollary:

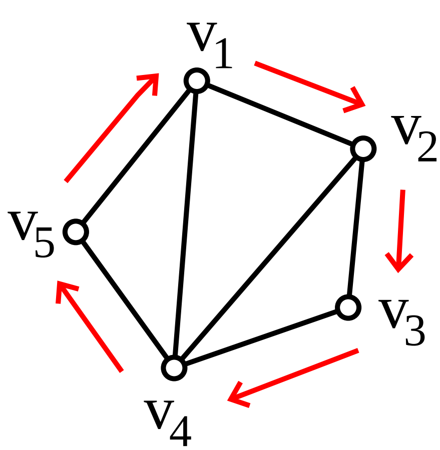
CLIQUE and INDEPENDENT SET are NP-complete.

HAMILTONICITY

Instance: Graph $G = (V, E)$.

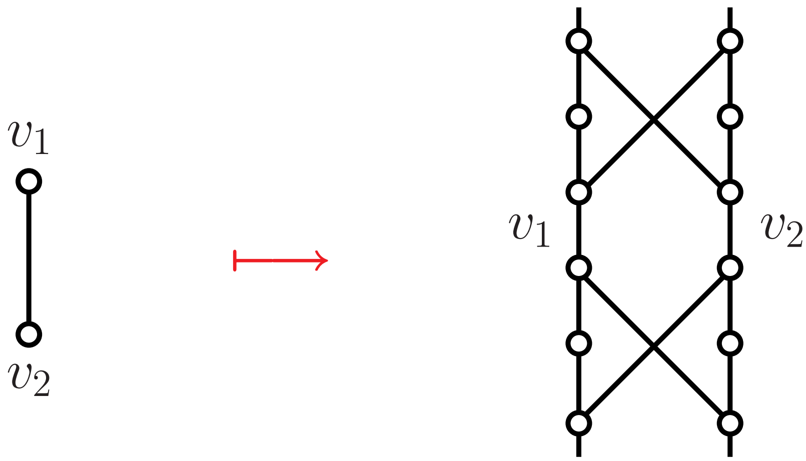
Question: Is there a **Hamiltonian cycle/path** in G?

Is there a "tour" along the edges such that all vertices are visited exactly once? (a Hamiltonian *cycle* requires that we can go back from the last node to the first node)

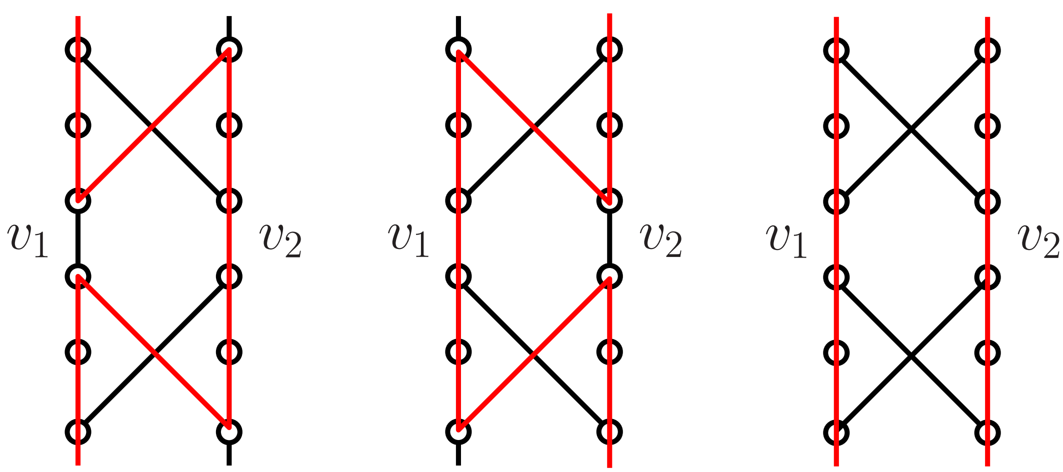


VC		HAMILTONICITY
edges	$\longmapsto$	edge gadgets
vertices	$\longmapsto$	how gadgets are connected
K guards	$\longmapsto$	K selector nodes

edges $\longmapsto$ edge gadgets



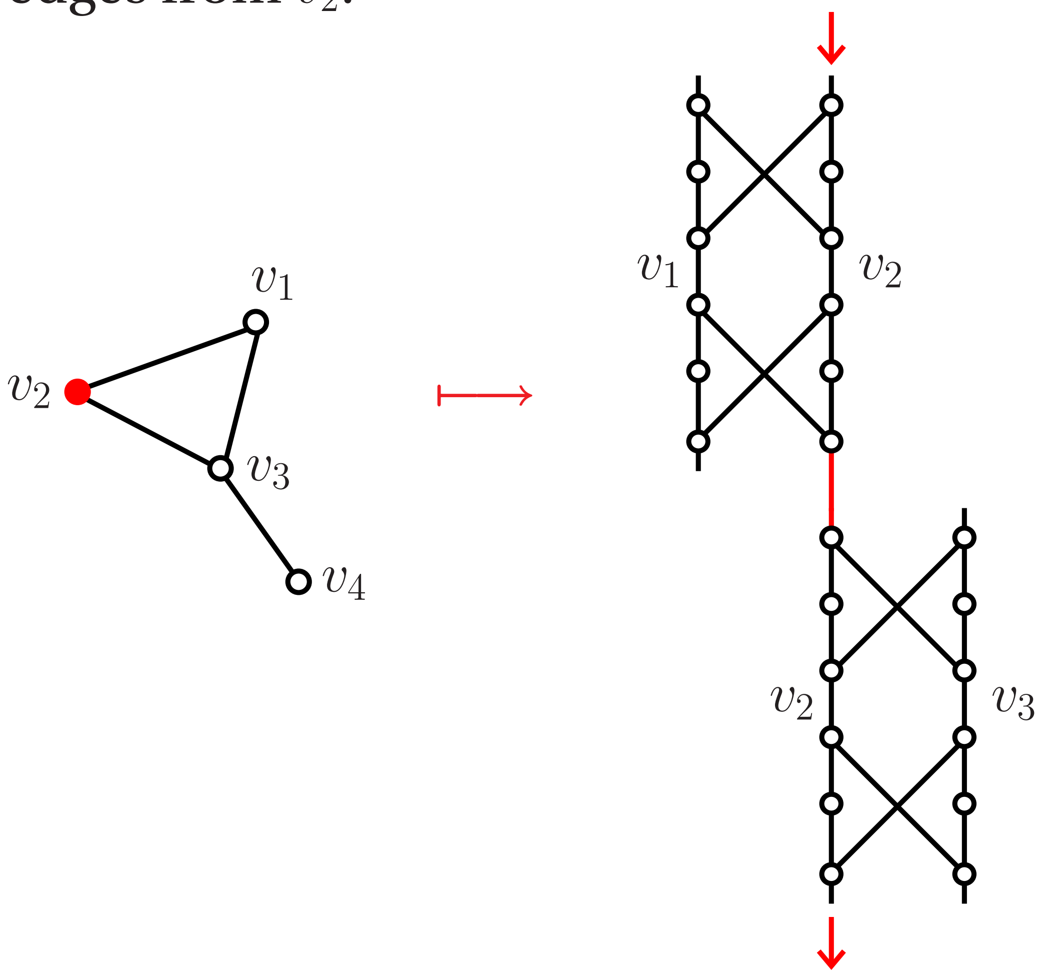
A Hamiltonian path can visit the vertices in the edge gadget in one of three ways:



We want this to correspond to guards being placed on v_1 or v_2 or both v_1 and v_2, respectively.

vertices ⟼ how gadgets are connected

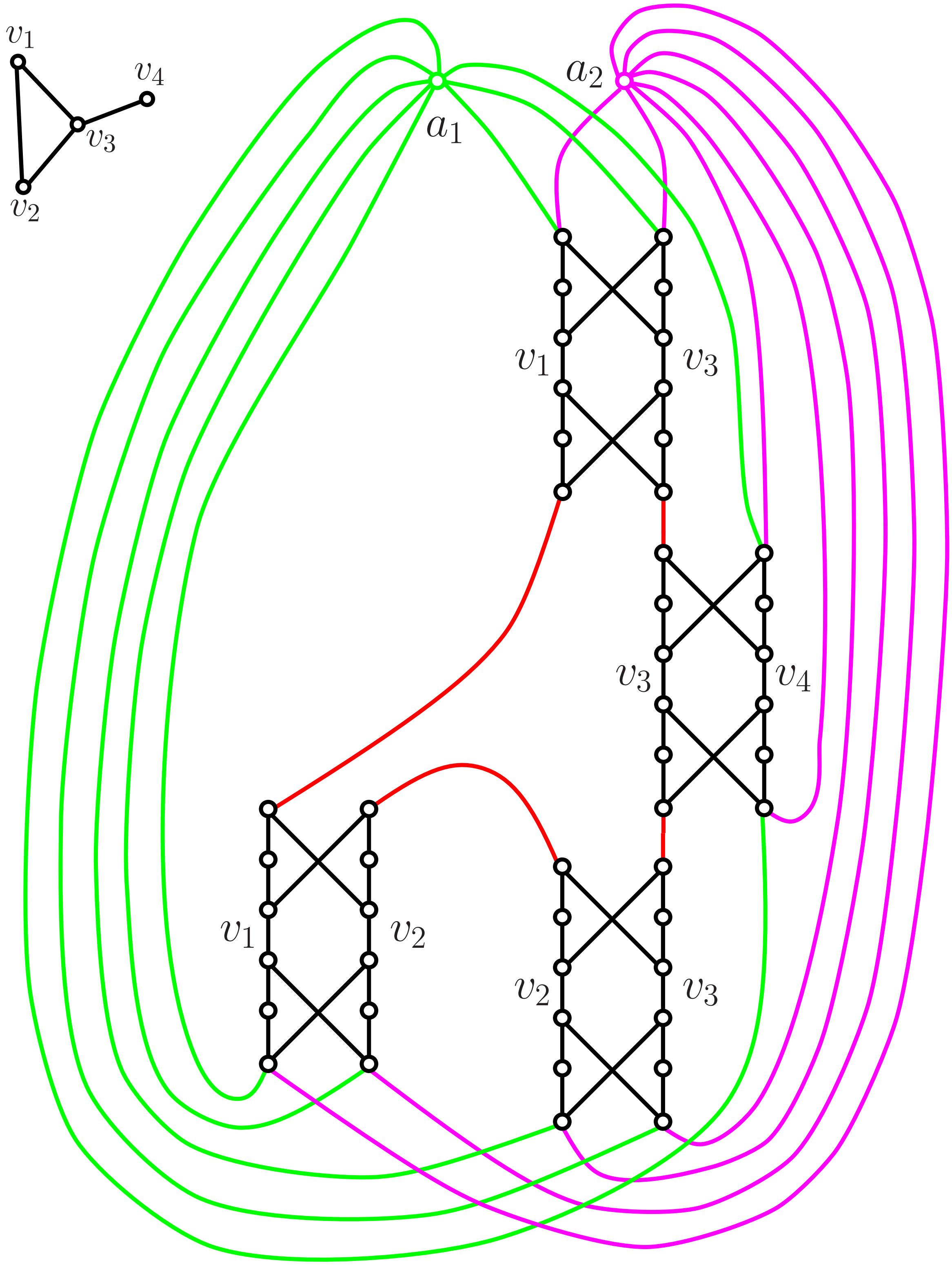
For each vertex v_2, we connect together **in serial** all edge gadgets corresponding to edges from v_2:



- Any Hamiltonian path entering at the v_2-side (red arrow) can visit (if necessary) all vertices in the serially-connected gadgets and will eventually exit at bottom on the v_2-side.
- This corresponds to the VC-property that a guard on v_2 covers all outgoing edges from v_2.

K guards $\longmapsto$ K selector nodes

We finish the construction by introducing K selector nodes a_i which are connected with all "loose" edges:



There is a VC which uses K guards $\Leftrightarrow$ There is a Hamiltonian cycle