

Poly-time Reducibility - 2

3CNF-formulae:

Suppose we have n boolean variables x_1, \dots, x_n .

A literal is either a variable or its complete negation, x_i or $\neg x_i$.

If x_i is assigned to T, $\neg x_i$ evaluates to F under this assignment.

If x_i is assigned F, $\neg x_i$ evaluates to T under this assignment.

A clause is a disjunction of literals: For example $x_1 \vee \neg x_2 \vee \neg x_4$

Disjunction \equiv OR

Under an assignment to the variables, clause evaluates to whatever the OR of the ~~var~~ literals evaluates to.

For example,

$$x_1 = F \quad x_2 = F \quad x_3 = ? \quad x_4 = \overline{T}$$

causes the clause to evaluate to true.

Whereas

$x_1 = F \quad x_2 = T \quad x_3 = ? \quad x_4 = \overline{T}$ causes
the clause to evaluate to false.

So an assignment

So a clause evaluates to true under
an assignment if and only if at least
one literal in the clause evaluates
to true under the assignment.

A CNF formula ϕ is a collection of
clauses, treated as a Conjunction (AND).

ϕ evaluates to true under an assignment
if each clause evaluates to true
under the assignment.

So, ϕ evaluates to true under an assignment to the variables if and only if ~~one~~ at least one literal in each clause of ϕ evaluates to true under the assignment.

$$\phi = (x_1 \vee x_2) (\neg x_1 \vee x_2) (x_1 \vee \neg x_2)$$

ϕ evaluates to true under

$$x_1 = T, \quad x_2 = T.$$

ϕ evaluates to False under

$$x_1 = F \quad x_2 = \overline{T}.$$

The CNF-SAT problem is to determine, given a CNF formula ϕ , whether ϕ has an assignment that satisfies ϕ , that is, causes it to evaluate to true.

In the above example, ϕ had a satisfying assignment.

Here is a ϕ that does not have a satisfying assignment:

$$\phi: (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \\ \wedge (\neg x_1 \vee \neg x_2)$$

Here is another:

$$\phi: (x_1 \vee x_2) \quad (x_1 \vee \neg x_2) \quad (\neg x_1 \vee x_2) \\ (\neg x_1 \vee \neg x_2 \vee x_3) \quad (\neg x_1 \vee \neg x_2 \vee \neg x_3),$$

CNF-SAT arises in a number of constraint satisfaction applications. We are interested in 3CNF-SAT, the special case in which each formula in ϕ has exactly 3 literals.

Such a formula ϕ is called a 3CNF-formula.

We do not know if 3CNF-SAT has a polynomial time algorithm.

What we will show is that

$$\text{3CNF-SAT} \leq_p \text{Independent-Set.}$$

This means

- (a) A polynomial time algorithm for independent set implies a poly-time algorithm for 3CNF-SAT.
- (b) If there is no poly-time algo for independent set 3CNF-SAT, there is no poly-time algo for independent set.

$\text{3CNF-SAT} \leq_p \text{Independent-Set}$

We describe an algorithm that takes as input any instance

$$\phi = C_1 \wedge C_2 \dots \wedge C_m$$

of 3CNF-SAT ~~and~~ in variables x_1, \dots, x_n

and constructs an instance of independent set such that the answer to the independent set instance is yes if and only if there is an assignment that satisfies ϕ . The algorithm will have polynomial running time.

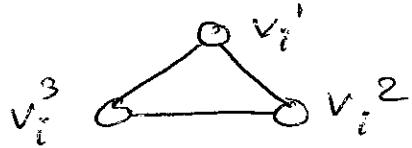
The algorithm will simply feed the independent set instance to the independent set black box and return whatever the black box returns - yes/no.

The graph is constructed as follows.

Corresponding to C_i , we add vertices v_i^1, v_i^2, v_i^3 . v_i^1 corresponds to first literal in C_i , v_i^2 to second literal in C_i , and so on.

So there are $3m$ vertices.

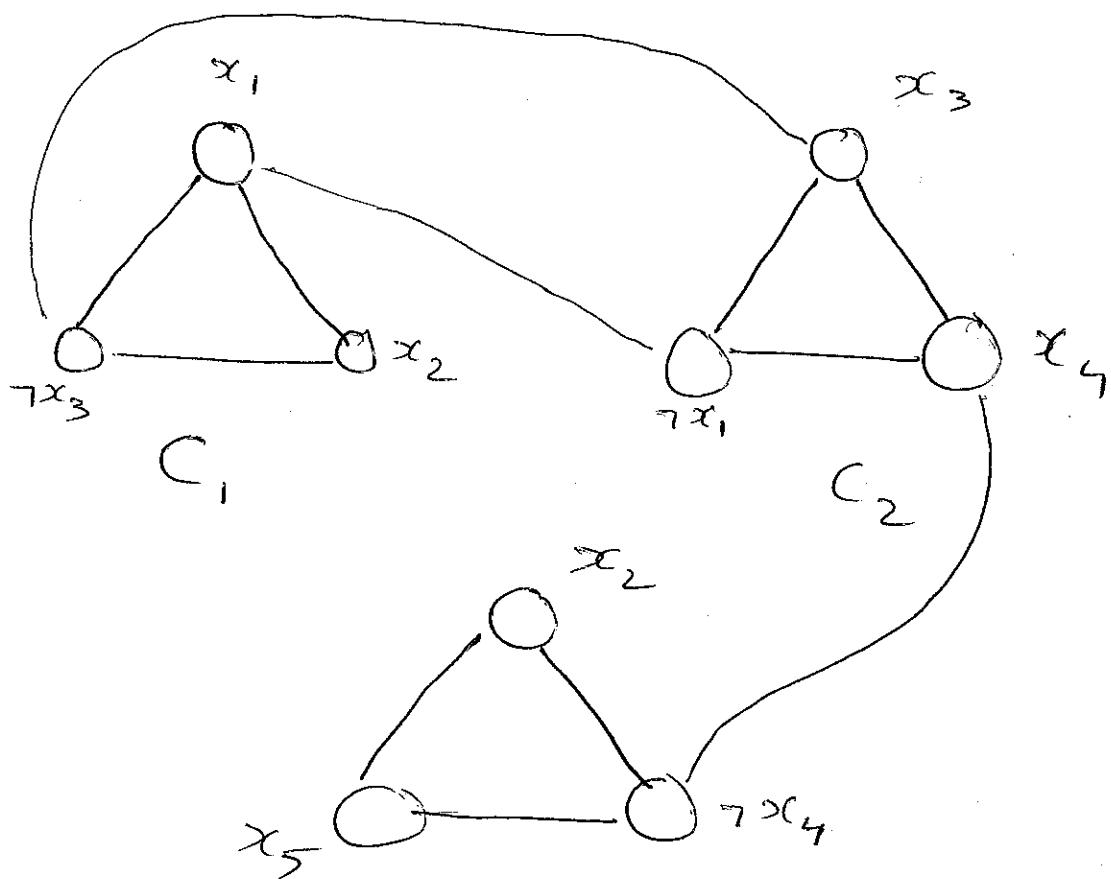
Now for the edges: For each i , we add $(v_i^1, v_i^2), (v_i^2, v_i^3), (v_i^3, v_i^1)$.



Let v_i^s and v_j^t be two vertices with $i \neq j$. We add an edge between v_i^s and v_j^t if corresponding literals are negations of each other - x_k and $\neg x_k$.

That completes the description of the graph. Finally, the independent set instance asks if the graph has an independent set of size at least m ?

$$\phi: (x_1 \vee x_2 \vee \neg x_3) \quad (x_3 \vee x_4 \vee \neg x_1) \\ (x_2 \vee \neg x_4 \vee x_5)$$



All we need to do is to argue that ϕ has a satisfying assignment if and only if the corresponding graph has an independent set of size m or greater. Notice that an independent set can include at most one vertex from each "triangle", so it has size at most m .

Suppose ϕ has a satisfying assignment. This assignment causes one literal in each clause to evaluate to true. The corresponding vertices in the graph form an independent set of size m . (~~If assignment~~ The set of literals we picked cannot include x_i and $\neg x_i$, because both literals must evaluate to true under assignment.)

Conversely, suppose the graph has an independent set of size n .

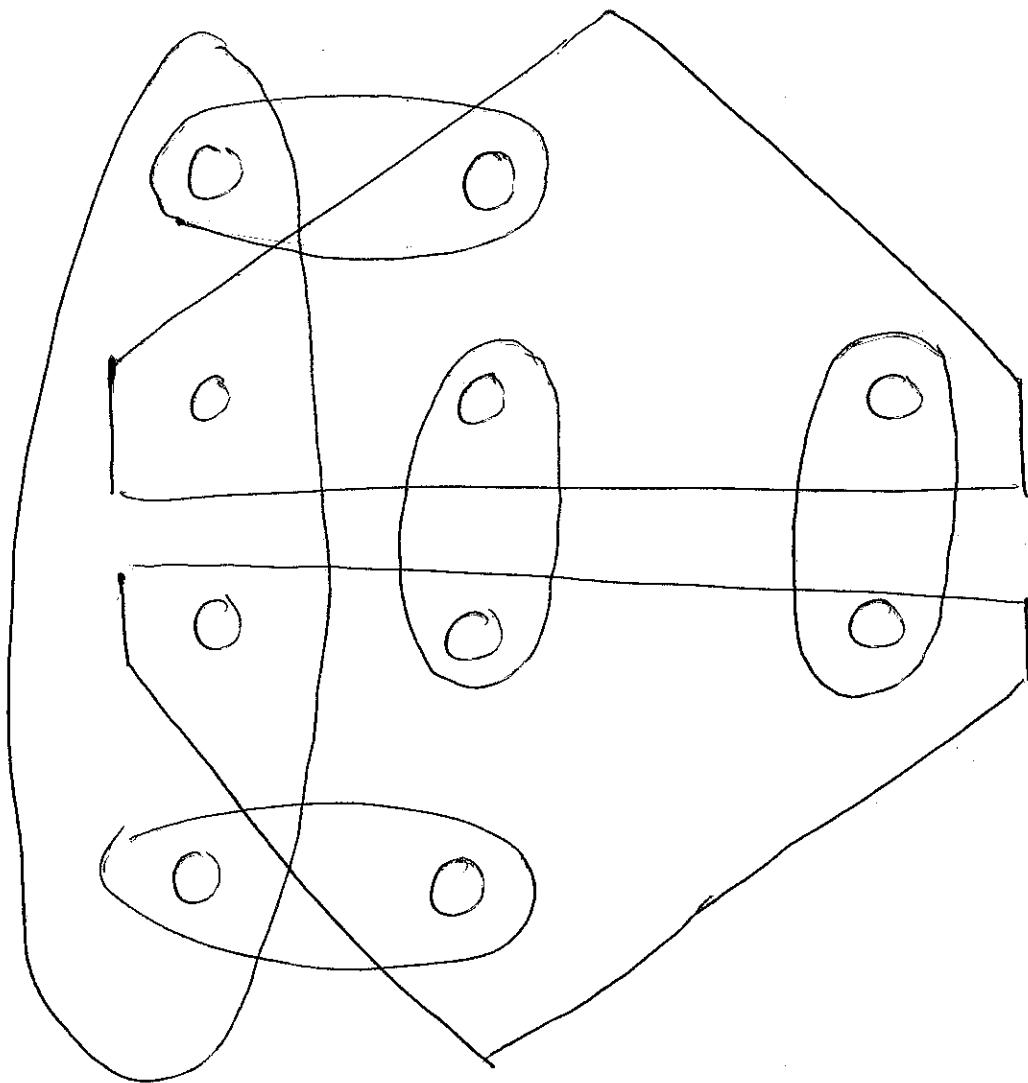
This must include exactly one vertex from each triangle. So we get one literal from each clause.

Since we started from an independent set, this set of literals does not both have x_i and $\neg x_i$ for any i .

So there is any assignment that causes each literal in the set to evaluate to true, and thus each clause to evaluate to true.

Set Cover:

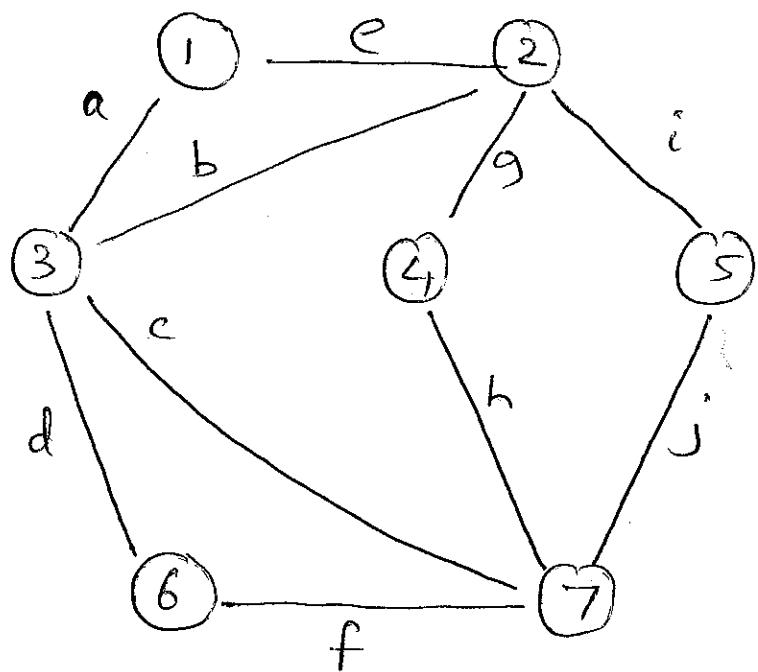
Given a set U of n elements, a collection S_1, \dots, S_m of subsets of U , and a number K , does there exist a collection of at most K of these sets whose union is equal to U ?



In the above example, answer is yes with $K=3$, but no with $K=2$.

Vertex Cover \leq_p Set-Cover.

This is simply because vertex cover is a special case of set cover.
We'll show why via an example.



In the corresponding set cover instance,

$$U = \{a, b, c, d, e, f, g, h, i, j\},$$

the set of edges. There is a set corresponding to each vertex - the set of edges incident to it.

$$S_1 = \{a, e\}$$

$$S_2 = \{e, b, g, i\}$$

$$S_3 = \{a, b, c, d\}$$

$$S_4 = \{g, h\}$$

$$S_5 = \{i, j\}$$

$$S_6 = \{d, f\}$$

$$S_7 = \{c, f, h, j\}$$