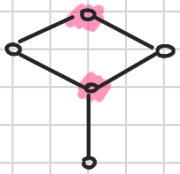
NOTE BOOK

Rokshana Ahmed

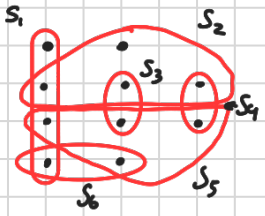
REDUCTION TO A MORE GENERAL CASE

Vertex Cover is a special case of a more general "covering" problems.



SET COVER: Given a ground set $U = \{1, 2, \dots, n\}$, and a collection of its subsets $S_1, S_2, \dots, S_m \subseteq U$ and an integer $k \geq 1$,

does there exists a subcollection of k subsets (that is, of k S_i's), whose union equals U?



Do there exist $k=3$ subsets that cover U? Yes (S_1, S_2, S_6)

Intuitively, Vertex Cover looks easier than set cover

L: Vertex Cover $\leq_p$ Set Cover

P: (We assume to have an oracle for Set Cover)

Let $G(V,E)$ is an instance of set cover.
In VC, we aim to cover each edge $e \in E$.
So, let us set $U = E$ in the Set Cover Instance.
In VC, when we pick a vertex $v \in V$, we cover all the ends that are incident on v.

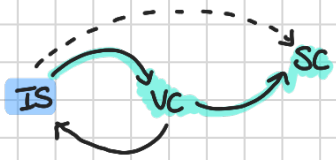
For each $v \in V$, we create a subset S_v as follows:

$$S_v = \{e \mid e \in E \text{ and } v \in V \text{ is incident on } e\}$$

CLAIM: U can be covered with k subsets IFF $G(V,E)$ can be = = k nodes.

P: Suppose that $S_{v_1}, S_{v_2}, \dots, S_{v_k}$ is a set cover, that is, $\bigcup_{i=1}^{k} S_{v_i} = U = E$.

Then, by our construction, $\{v_1, v_2, \dots, v_k\}$ is a vertex cover for $G(V,E)$.
Conversely, if $(v_1, \dots, v_k)$ is a Vertex Cover, then $S_{v_1}, S_{v_2}, \dots, S_{v_k}$ is a set cover. ■



$f(n) = n^c$
$g(n) = n^d$
$h(n) = g(f(n)) = (f(n))^d$
$= (n^c)^d = n^{cd}$

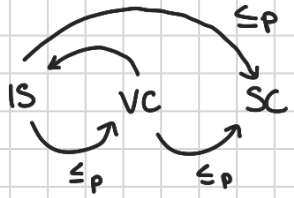
TRANSITIVITY OF REDUCTIONS

L: If $Z \leq_p Y$ and $Y \leq_p X$, then $Z \leq_p X$.

P: We aim to solve an instance of Z, using an oracle for X.

First, we could run the algorithm/reduction for solving instances of Z, using an oracle for Y.
Second, we can implement an algorithm for solving instances of Y, using an oracle for X.

Thus, I have produced an algorithm that solves instances of Z, using an oracle of X. ■



SATISFIABILITY (SAT)

Suppose we are given a set X of n boolean variables, $x_1, x_2 \dots x_n$. (a boolean variable can only have 2 values: True & False)

A term over X is either a variable x_i or the negation $\overline{x_i}$ of x_i ($x_i \equiv$ NOT x_i).

A clause is a disjunction (a "or") of distinct terms.

$t_1 \vee t_2 \vee t_3 = t_1$ OR t_2 OR t_3

(INCLUSIVE) OR
$\wedge \rightarrow$ AND

(t_i can be either a variable x_j or its negation $\overline{x_j}$)

A **formula** is a conjunction (an "AND") of clauses.

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

SAT: Can I find a truth assignment to the variables that makes the formula true?

3-SAT: Given a SAT instance where each clause has exactly 3 literals, is it satisfiable?

L: $3SAT \leq_p SAT$

P: Trivial (SAT is a generalization of 3-SAT)

L: $SAT \leq_p 3SAT$

P: Omitted

SAT and 3SAT are two algebraic problems - so it is not surprising that they can be reduced to one another.

L: $3SAT \leq_p IS$

REDUCTION THROUGH GADGETS

P: As usual, we assume to have an oracle for IS. We will be using the IS oracle to solve the input instance of the 3-SAT problem.

The 3-SAT instance is composed of a set X of variables, an of a set of clauses $\{C_1, \dots, C_k\}$ with $|C_i| = 3 \quad \forall i = 1, \dots, k$.

$$X = \{x_1, x_2, x_3, x_4\}$$
$$C_1 = \{x_1, \bar{x}_2, x_3\}$$
$$C_2 = \{\bar{x}_4, x_2, x_1\}$$

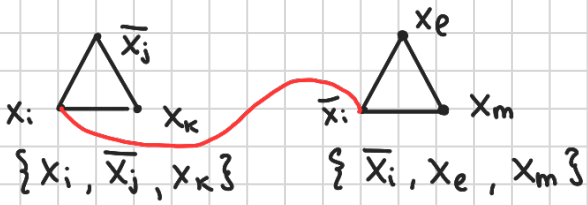
The key to our proof is to look at 3-SAT in a different way:

- we said that 3SAT can be solved by assigning a T/F value to each variable in order to have at least one true literal per clause;
- But, what if we select **one** term per clause, and choose the value of that term so to make it* true (for instance, we could choose the literal $\bar{x}_3$, which can be made true by setting $x_3 =$ FALSE), with

* and the clause

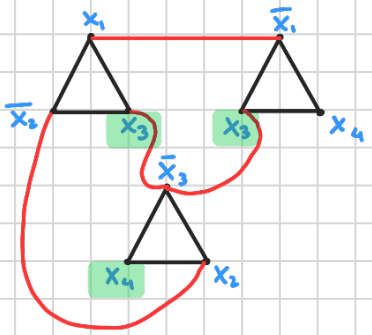
the goal of never selecting both x_i and $\bar{x}_i$, for $i = 1 \dots n$.

We will construct for a reduction on the second view.



- We will construct a graph $G(V, E)$ (starting from the formula) with $3k$ nodes (where k is the number of clauses in the formula):
  - for each $i = 1 \dots k$, we create 3 nodes v_i, v_{i2}, v_{i3} we "label" node v_{ij} with the jth term of the ith clause.
    The nodes v_i, v_{i2}, v_{i3} will form a triangle.
    The graph will also contain other edges:
    - if v_{ij} and $v_{i'j'}$ are two nodes with opposites labels (e.g. "x_k" and "$\bar{x}_k$") we add an edge between v_{ij} and $v_{i'j'}$.

This concludes the description of the reduction:



$C_1 = \{x_1, \bar{x}_2, x_3\}$

$C_2 = \{\bar{x}_1, x_3, x_4\}$

$C_3 = \{\bar{x}_3, x_4, x_2\}$

We will prove that the formula is satisfiable IFF the graph has an ind. set of k nodes.

- Formula SAT $\Rightarrow$ $\exists$ ind. set of size k

  Suppose to have a satisfying assignment τ.
  τ will guarantee that each clause has at least one true term.
  Since τ assigns a single value to each variable, it cannot make both x_i and $\bar{x}_i$ true, for any $i = 1 \dots n$. Thus, no red edge will see both its endpoints's labels set to true.
  Thus, an ind. set of size k can be obtained by picking one node (with "true" label) per triangle.

- ∃ independence set of size $K \Rightarrow$ formula SAT.

  As we said, if S is an ind. set of size $\geq K$, then (I) $|S| = K$ (the graph is composed of K disjoint triangles, and we can pick at most 1 node per triangle), and (II) S contains exactly 1 node per triangle.

  We create a satisfying assignment as follows:

  - for each variable x_i s.t. the ind. set contains no nodes labeled "x_i" or "$\bar{x}_i$", we set x_i arbitrarily to TRUE or FALSE;
  - for each other variable x_i, either the ind. set does not contain labels "x_i" (I set x_i = FALSE), or it contains labels "$\bar{x}_i$" (I set x_i = TRUE).

  This satisfies each clause. ■