Lecture 4 & 5 - Karp Reduction and NP-Completeness

Motivation : Establishing a link between all P and NP problems. If any one such problem is in P, then all problems are in P.

Definition 2.7 *(Reductions,* **NP***-hardness and* **NP***-completeness)* A language $L \subseteq \{0,1\}^*$ is *polynomial-time Karp reducible to a language* $L' \subseteq \{0,1\}^*$ (sometimes shortened to just "polynomial-time reducible"), denoted by $L \leq_p L'$, if there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in L$ if and only if $f(x) \in L'$.

The function *f* is not bijective, because it is neither necessarily injective, nor surjective.

Not injective: Square function on the domain of integers.
Not surjective : A function that maps everything to the same element.

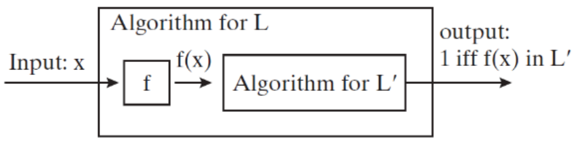
If L ≤p L' and L' ∈ P then L ∈ **P**

Theorem 2.8

1. (Transitivity) *If $L \leq_p L'$ and $L' \leq_p L''$, then $L \leq_p L''$.*

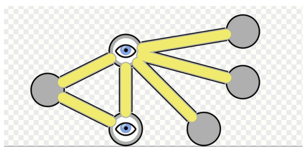
We say that L' is **NP**-*hard* if $L \leq_p L'$ for every $L \in$ **NP**. We say that L' is **NP**-*complete* if L' is **NP**-hard and $L' \in$ **NP**.

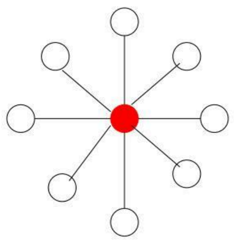
2. *If language L is* **NP***-hard and $L \in$* **P***, then* **P** $=$ **NP**.
3. *If language L is* **NP***-complete, then $L \in$* **P** *if and only if* **P** $=$ **NP**.

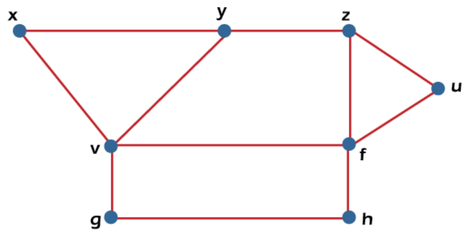


Example 1: Vertex Cover to Independent Set

Let's look at a few examples:







Reduction Idea: A set of vertices is a vertex cover *if and only if* its complement is an independent set.

Explanation of reduction idea: If a vertex is **not** in a vertex cover, it means none of the edges adjacent to it are covered by that vertex, so all the other vertices covering those edges must belong to the vertex cover. Therefore, the vertices not in the vertex cover must form an independent set (since none of them are adjacent, otherwise their edge would require one of them to be in the vertex cover).

Reduction:

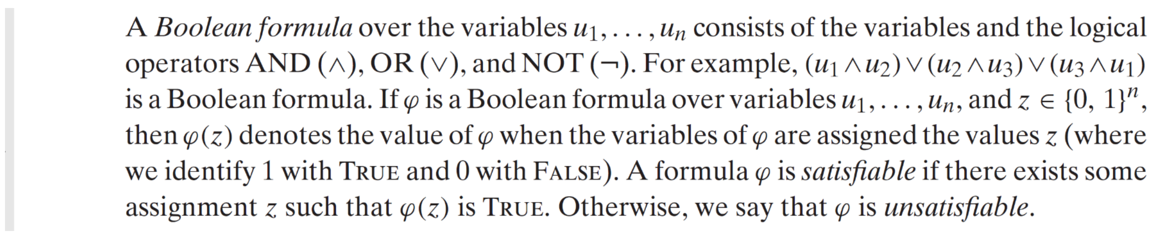
Input: (G, k), where graph G has a vertex cover of size k

f(G,k) = (G,n-k)

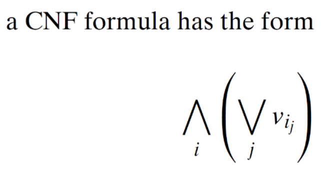
Output: 1 *iff* (G, n-k) ∈ INDSET, i.e. graph G has an independent set of size n-k

Practice with Non-deterministic TM: Suppose L_1, L_2 ∈ **NP**. Then is $L_1 \cup L_2$ in **NP**? What about $L_1 \cap L_2$?

Example 2: 3SAT to VERTEX COVER:



> A *Boolean formula* over the variables $u_1, \ldots, u_n$ consists of the variables and the logical operators AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). For example, $(u_1 \wedge u_2) \vee (u_2 \wedge u_3) \vee (u_3 \wedge u_1)$ is a Boolean formula. If φ is a Boolean formula over variables $u_1, \ldots, u_n$, and $z \in \{0, 1\}^n$, then $\varphi(z)$ denotes the value of φ when the variables of φ are assigned the values z (where we identify 1 with TRUE and 0 with FALSE). A formula φ is *satisfiable* if there exists some assignment z such that $\varphi(z)$ is TRUE. Otherwise, we say that φ is *unsatisfiable*.



a CNF formula has the form

$$\bigwedge_i \left(\bigvee_j v_{i_j} \right)$$

A kCNF formula has at most k literals in any clause.

To show Vertex Cover is NP-Complete

3SAT $\leq_p$ VC:

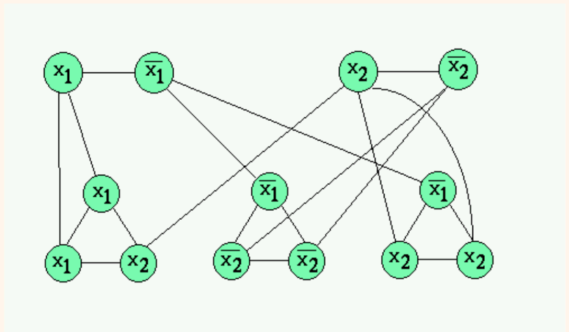
Reduction idea: A 3CNF formula is satisfiable *iff* the graph (constructed as below) has a vertex cover of size *n* + 2*m*

Variable gadgets

Clause gadgets

k=n+2m

Below is a sample construction of (x_1 V x_1 V x_2)(!x_1 V !x_2 V !x_2)(!x_1 V x_2 V x_2) and it's cover in the constructed graph which will yield values of x_1=FALSE and x_2=TRUE:



Proof of reduction idea: The literals selected in the variable gadgets form the truth assignment to the formula. Because:

1. Exactly *n* literals will be selected
2. If a literal is selected, its negation will not be selected
3. Each clause will have *at least one* of the selected literals

Example 3: 3SAT to SAT

Since 3NCF is already an instance of the more generalized family of boolean formulae, this reduction is trivial.

Example 4 : SAT to 3SAT

Example 5 : 3SAT to CLIQUE

CLIQUE: a set of vertices where every vertex is connected to every other vertex.

Idea: φ is a satisfiable 3CNF formula with ***k*** clauses only if the graph formed according to the construction given below has a clique of size of ***k*.**

Construct a graph G of k clusters with a maximum of 3 nodes in each cluster.

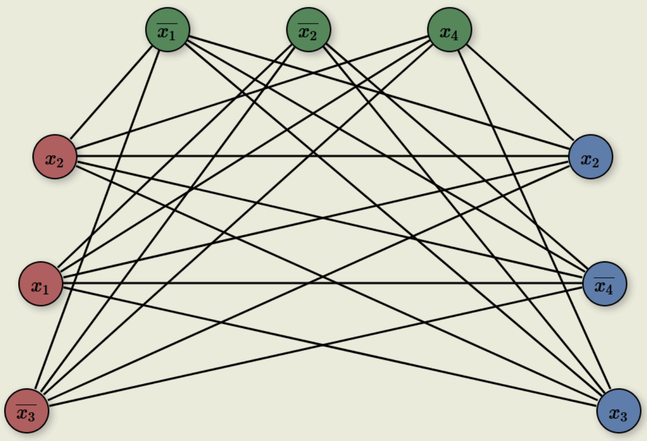
Each cluster corresponds to a clause in Φ.

Each node in a cluster is labeled with a literal from the clause.

An edge is put between all pairs of nodes in different clusters except for pairs of the form $(x, \overline{x})$

No edge is put between any pair of nodes in the same cluster.

$\Phi = (x_2 + x_1 + \overline{x_3}) \, . (\overline{x_1} + \overline{x_2} + x_4) \quad \cdot (x_2 + \overline{x_4} + x_3)$



1. **If the graph G has a k-clique,** the clique has exacty one node from each cluster.
(This is because no two nodes from the same cluster are connected to each other, hence they can never be a part of the same clique.)
All nodes in a clique are connected, hence all corresponding literals can be assigned $True$ simultaneously.
Each literal belong to exactly one of the k-clauses. Hence Φ **is satisfiable**

2. If Φ is satisfiable, let A be a satisfying assignment. Select from each clause a literal that is $True$ in A to construct a set S. $||S|| = k$.Since no two literals in S are from the same clause and all of them are simultaneously $True$, all the corresponding nodes in the graph are connected to each other, forming a k-clique.Hence **the graph has a k-clique**