P, NP Theory I

Prof Devesh C Jinwala, PhD
Professor, Department of Computer Science and Engineering
http//www.svnit.ac.in/dcj/

Indian Institute of Technology Jammu

Contents

- What is an Algorithm ? Solving problems algorithmically
- Classifying the problems
- A Motivating Example
- Some Hard Problems
- Reductions
- Non-determinism
- Class P, NP
- NP Complete Problems
- Concluding Remarks

Contents

- What is an Algorithm ? Solving problems algorithmically
- Classifying the problems
- A Motivating Example
- Some Hard Problems
- **Reductions**
- Non-determinism
- Class P, NP
- NP Complete Problems
- Concluding Remarks

Reducing one problem to the other

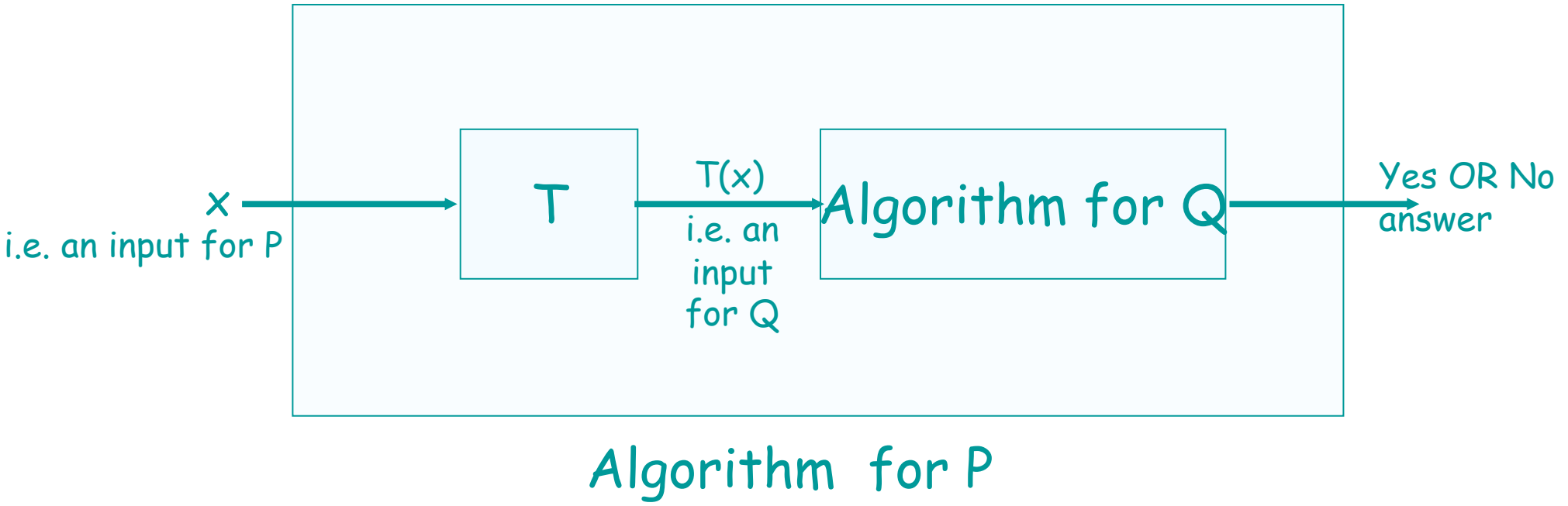
Reduction, broadly

- Consider two problems P and Q such that
  - Suppose we wish to solve P but, we have an algorithm to solve Q.
- Then, if we can
  - devise a function T that takes as input x - that P is supposed to take, produces T(x) – that Q accepts as input and
  - Q produces the output that actually is the output of P

  then, we say that P reduces to Q
- We have been able to solve P using the algorithm for Q.

Reduction, graphically

- P reduces to Q means Problem P is solved using Q and a function T and so
  - Q is at least as hard as P



  - Note that, here T is nothing but a converter……

Reductions…

- Formal definition……..
  - Let P_1 and P_2 be two problems. Then problem P_1 reduces to P_2 (i.e. P_1 α P_2) if and only if there is a way to solve P_1 by a deterministic polynomial time algorithm P_2 i.e. using a deterministic algorithm that also solves P_2 in polynomial time.

- Notations
  - P_1 α P_2 OR
  - $P_1 \leq_P P_2$

Reductions……

- Informally,
  - Given that a problem P_1 is reducible to P_2 (denoted by $P_1 \alpha P_2$ OR $P_1 \leq_P P_2$), it implies that if P_2 is solvable then P_1 also is solvable.
    - e.g. SELECTION α SORTING
    - i.e. if SORTING is solvable then SELECTION also is solvable.
  - Is the reverse true for this example ?
    - i.e. Is SORTING α SELECTION ?

Reductions…….

- Very very importantly, note that given that a problem P_1 is reducible to P_2 (denoted by $P_1 \alpha P_2$),
  - it also implies that P_2 is at least as difficult as P_1.
- This means that

SELECTION α SORTING
implies that
SORTING is as difficult as SELECTION

Intuitively, how can we use reduction to prove a new problem to be hard ?

Reduction…..

- How would you define the converter functions for reducing the SELECTION to the SORTING, in the previous example ?
- Now,
  - Input for T(x) – we simply input N different numbers viz. (x_1, x_2, x_3,……. x_n) to T_1 hiding the i^{th} largest value that is to be selected from these N different numbers.
  - i.e. let T be defined as

    $T(x_1, x_2, x_3, \ldots\ldots. x_n) = (y_1, y_2, y_3, \ldots\ldots y_n)$, such that $y_i = x_i$
  - Thus, Q would sort these N different numbers
  - Let T_2 select only the i^{th} value from these and return as the answer.

Reduction: Another Example

- Two problems
  - Sum_{Pair} - Compute the sum of two numbers
  - Sum_{List} - Compute the sum of n numbers in a vector
- Is it possible to use Sum_{List} to find the sum of n numbers in a vector i.e. the answer to Sum_{Pair} ?
- Hence, we would say

$$Sum_{Pair} \leq_P Sum_{List}$$

- Note that, in this case we can also say that

$$Sum_{List} \leq_P Sum_{Pair} \quad \text{i.e} \quad Sum_{Pair} \equiv_P Sum_{List}$$

Polynomial time Reduction

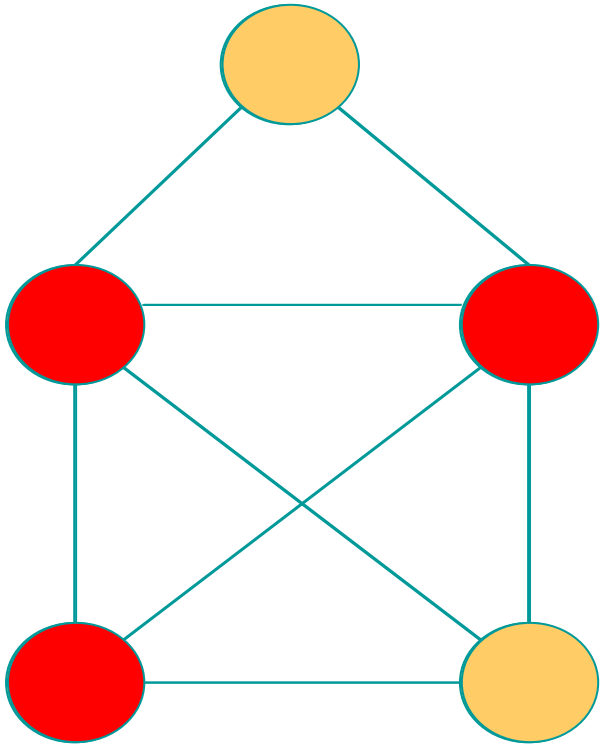
- Polynomial time reduction
  - It is essential to ensure that the functions T_1, T_2 and the algorithm for Q are all polynomial time to ensure

    Polynomial Time Reducibility.
- Also, reducibility is useful only if it is polynomial time reducibility.
- definition
  - We say that a problem P_1 polynomially reduces to another problem P_2 only if the time for CONVERTER 1 plus the time for CONVERTER 2 is DETERMINISTIC POLYNOMIAL TIME.

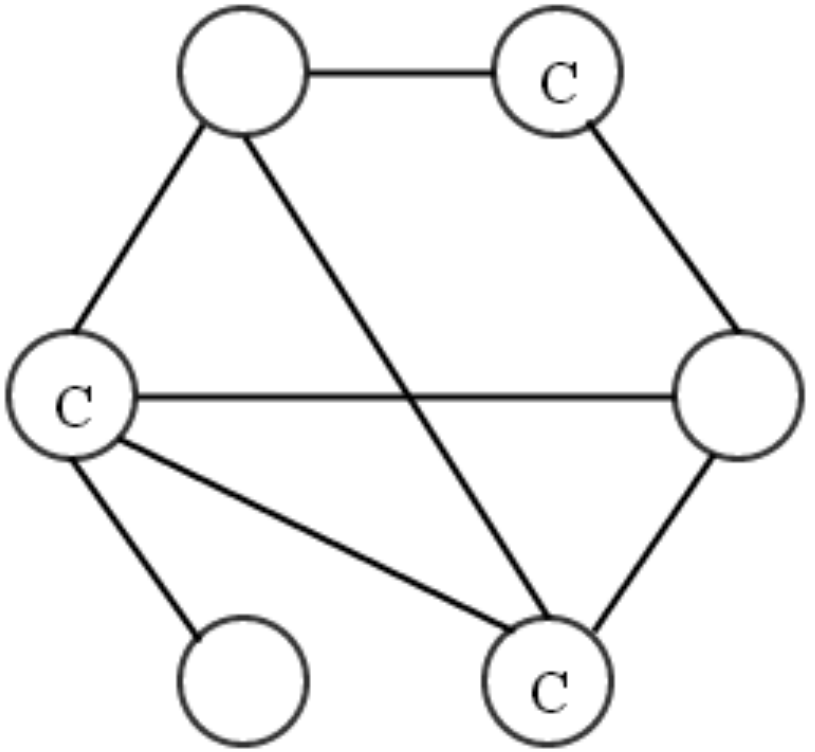
Reduction By Simple Equivalence

Vertex Cover and IS

- Minimal Vertex cover of a graph is
  - a minimum subset of the vertices of G which contains at least one of the two endpoints of each edge in G:



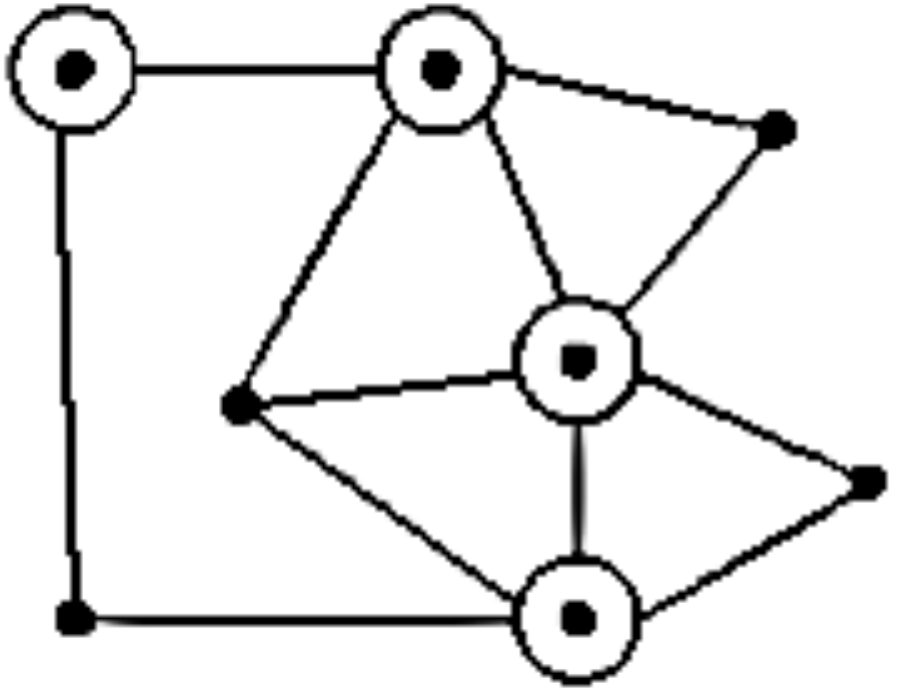
Vertex Cover and Independent Set



Vertex cover

Independent set

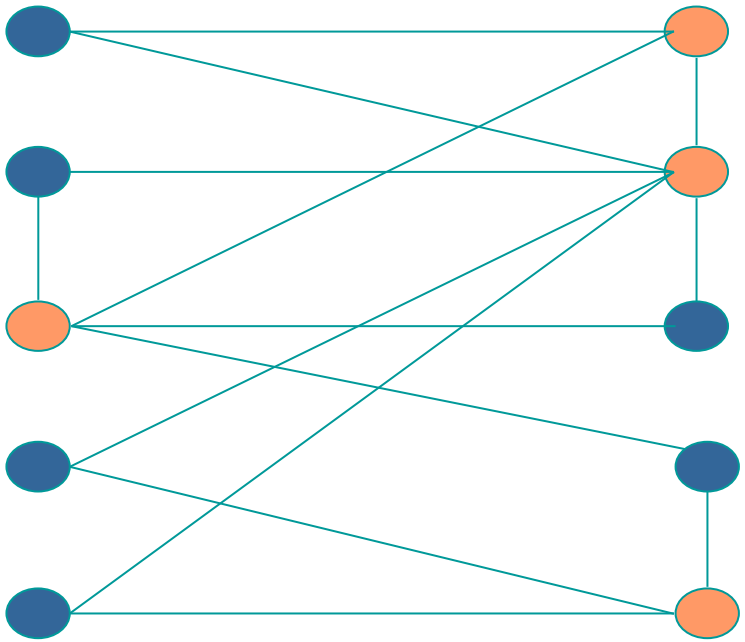
Vertex Cover and Independent Set



What is the vertex cover and the independent set of this graph ?

Maximum Independent Set

- What is an independent set of a graph ?
- Problem
  - Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at the most only one of its endpoints is in S?
    - E.g. Is there an independent set of size ≥ 6? Yes. ……size ≥ 7? No.

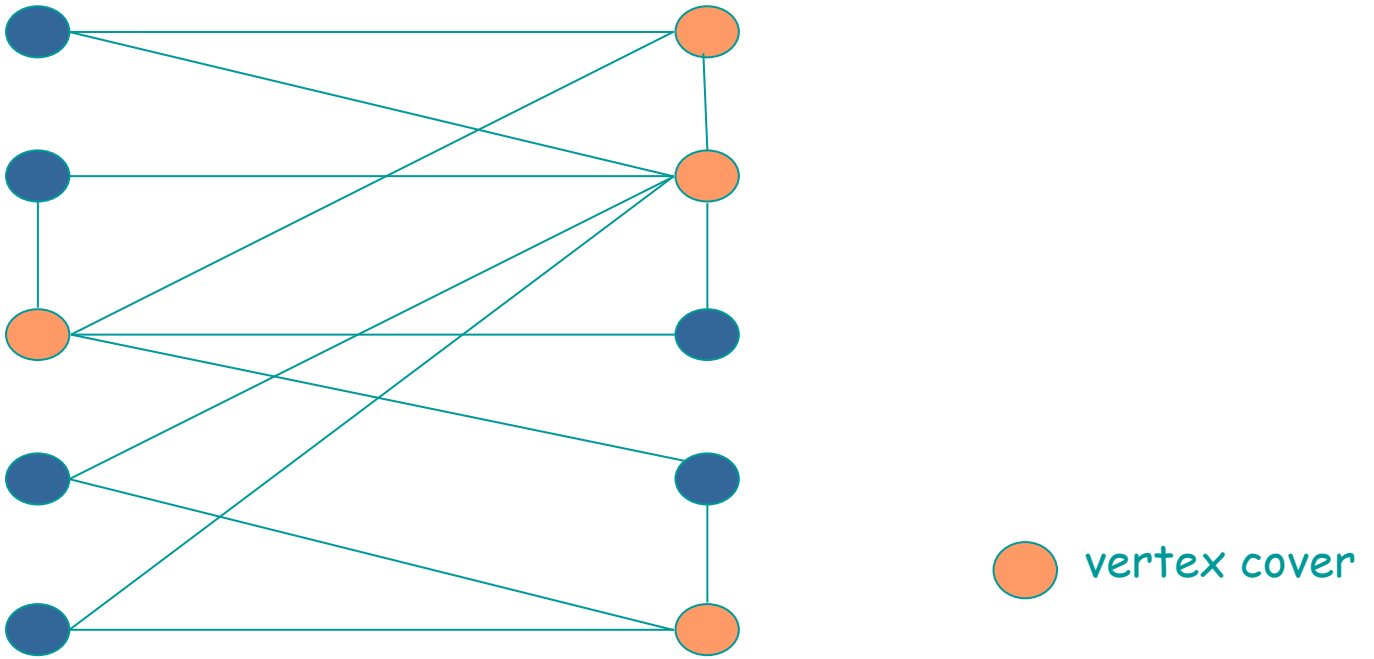


Informally - No two vertices in S are connected by an edge

independent set

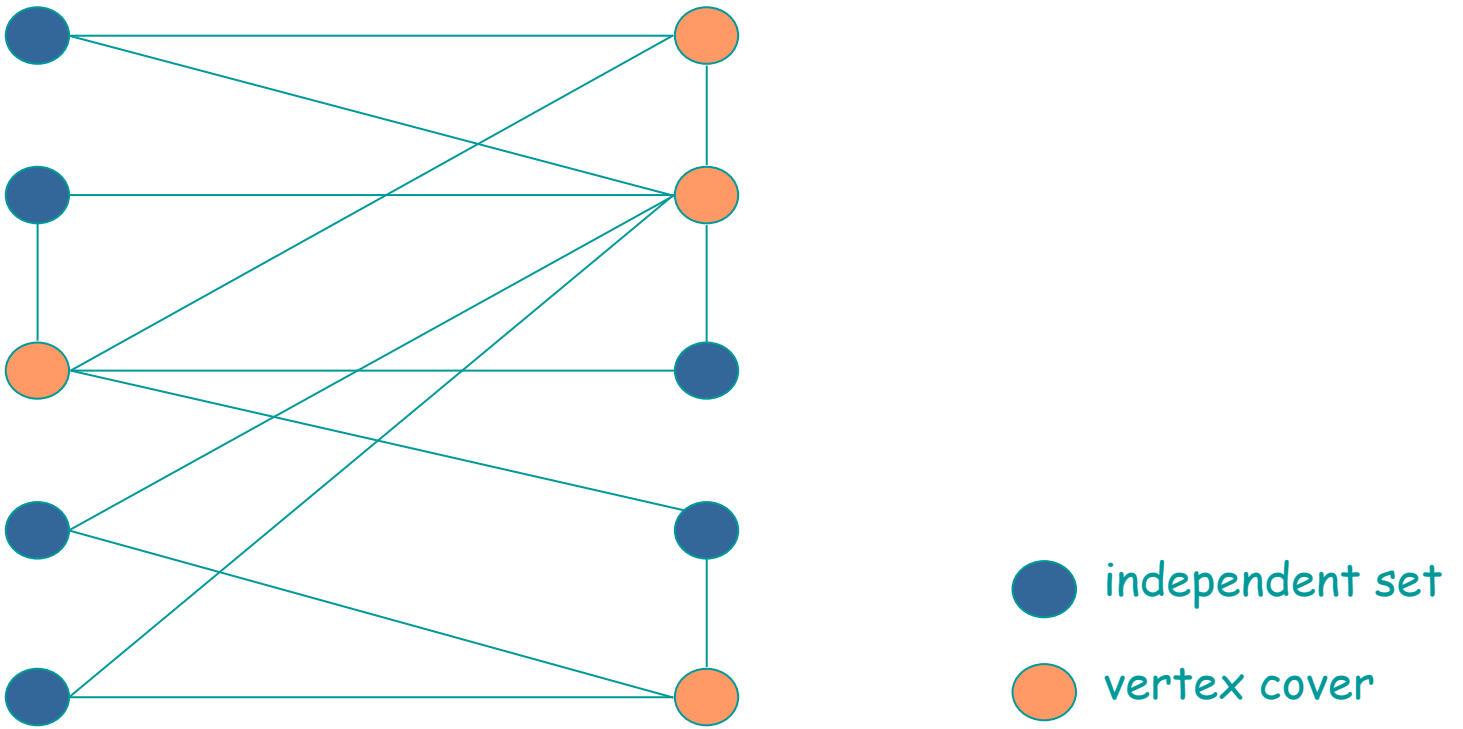
Minimum Vertex Cover

- VERTEX COVER
  - informally a set of vertices that include all the edges…..
  - Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?
- Ex. Is there a vertex cover of size ≤ 4? Yes. size ≤ 3? No.



Vertex Cover and Independent Set

- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
- Pf. We show S is an independent set iff V – S is a vertex cover.



Vertex Cover and Independent Set

- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
- Proof. We show S is an independent set iff V – S is a vertex cover.
- $\Rightarrow$ i.e. to prove that if S is an independent set, then V-S is a vertex cover

independent…
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

Vertex Cover and Independent Set

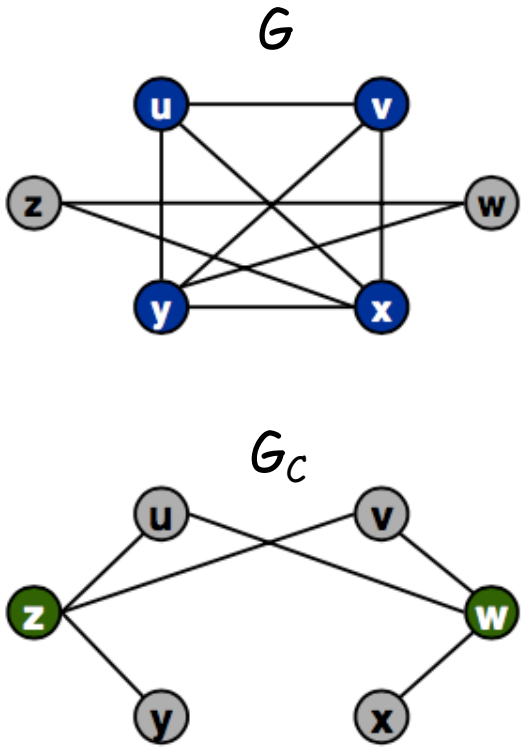
- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
- Proof. We show S is an independent set iff V – S is a vertex cover.
- $\Leftarrow$ i.e. to prove that if V-S is a vertex cover than S is an independent set

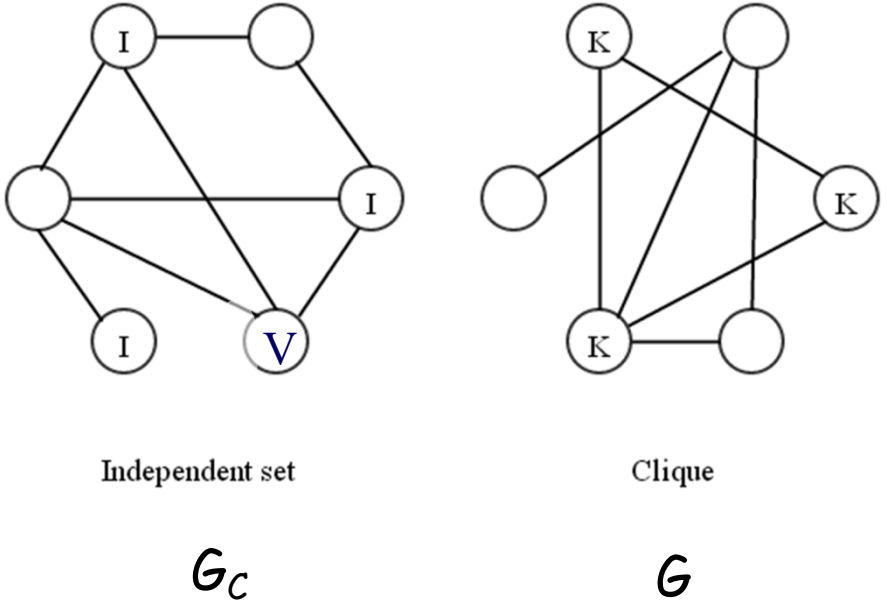
Vertex cover ...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at least
one of
its endpoints is in S

K-clique $\leq_p$ Vertex cover

- Claim If G has a clique of size *k*, the compliment graph G_C has a vertex cover of size |V| - *k*

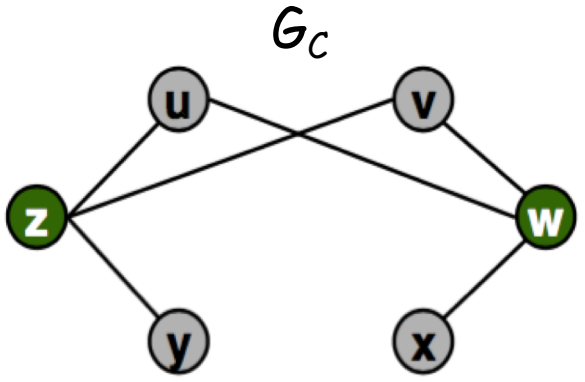
- Statement
  - Let S be the *k*-clique in G then we have to show that V - S is a vertex cover in G_C
- Proof:
  - An edge (v,w) in G_C

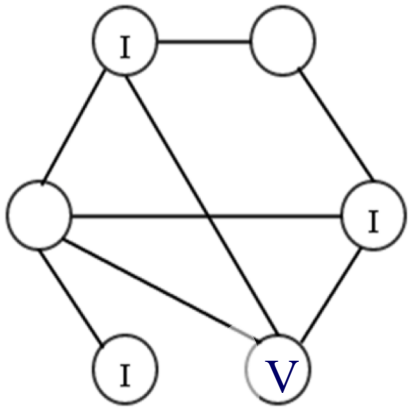




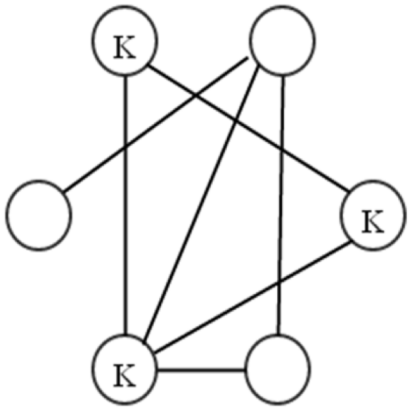
K-clique $\leq_p$ Vertex cover

- Claim If G_C has a vertex cover $V' \subseteq V$, with $|V'| = |V| - k$, then G has a clique of size *k*
- Statement
  - If S'=V-S is forming a vertex cover of size k'=|V|-k in Gc, then S is forming a clique of size k in G.
- Proof:
  - An edge (v,w) in G_C

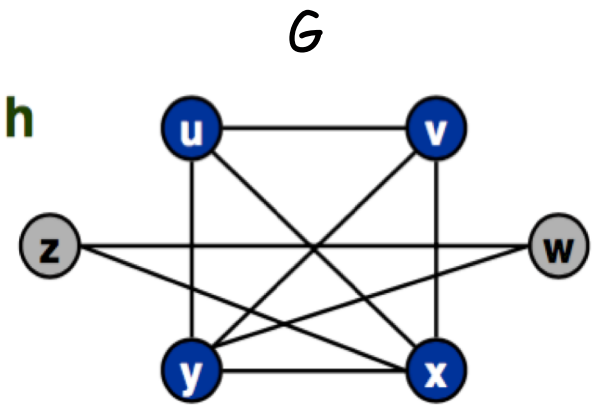




Independent set



Clique



Other Such Reductions

- Directed $\equiv_P$ Undirected Ham. Cycle
- Hamiltonian Cycle $\equiv_P$ TSP
- 3SAT $\equiv_P$ Integer Linear Programming
- 3SAT $\equiv_P$ 3-Colorablity - Graph coloring
- Subset Sum $\equiv_P$ 0-1 knapsack
- Hamiltonian path

Reduction
By
Encoding with Gadgets

Set Cover, formally

- Problem - An instance of Set Cover is given by
  - a ground set $U = x_1, x_2, x_3, \ldots\ldots x_n$
  - subsets $S \subseteq U$ of that ground set, and
  - an integer k.
- The question is,
  - Is it possible to select a collection C of at most k of these subsets such that taken together, they "cover" all of U?
  - That is, is there a n set $C \subseteq \{1, 2, \ldots, m\}$ such that $|C|=k$ and $\bigcup_{i \in C} S_i = U$?

Set Cover

- SET COVER
  - Given a set U of elements, a collection $S_1, S_2, \ldots \ldots S_m$ of subsets of U, and an integer k, does there exist a collection of less than or equal to k of these sets, whose union is equal to U?

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2

S_a = {3, 7} S_b = {2, 4}

S_c = {3, 4, 5, 6} S_d = {5}

S_e = {1} S_f = {1, 2, 6, 7}

Set Cover….

- Sample application.
  - m available pieces of software.
  - Set U of n capabilities that we would like our system to have.
  - The i^{th} piece of software provides the set $S_i \subseteq U$ of capabilities.
  - Goal achieve all n capabilities using fewest pieces of software.
- Ex

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$
$k = 2$

$S_a = \{3, 7\}$ $S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$ $S_d = \{5\}$

$S_e = \{1\}$ $S_f = \{1, 2, 6, 7\}$

Vertex Cover Reduces to Set Cover

- Claim. VERTEX-COVER $\leq_P$ SET-COVER.
- Proof
  - Given a VERTEX-COVER instance G = (V, E), k, we construct a set cover instance whose size equals the size of the vertex cover instance.
- Construction.
  - Create SET-COVER instance…….such that…….

- k = k, U = E, S_v = {e $\in$ E e incident to v }
- Set-cover of size $\leq$ k iff vertex cover of size $\leq$ k.

Vertex Cover Reduces to Set Cover

SET COVER

U = { 1, 2, 3, 4, 5, 6, 7 }, k = 2

S_a = {3, 7}, S_b = {2, 4}, S_c = {3, 4, 5, 6}, S_d = {5}, S_e = {1},

S_f= {1, 2, 6, 7}

- We need to design a gadget to obtain the equivalence of the set cover to the vertex cover. Why ?
  - Since their underlying data structures are different.
  - How to design one ?

Vertex Cover Reduces to Set Cover

SET COVER

U = { 1, 2, 3, 4, 5, 6, 7 }, k = 2

S_a = {3, 7}, S_b = {2, 4}, S_c = {3, 4, 5, 6}, S_d = {5}, S_e = {1},

S_f= {1, 2, 6, 7}

CLIQUE $\leq_p$ SAT

- Problem Show that the CLIQUE problem is an NPC.
- Proof /* work out the proof on board */

3SAT $\leq_p$ Independent Set

- Problem Show that the Independent Set problem is an NPC.
- Proof /* work out the proof on board */