Learn DAA : From B K Sharma

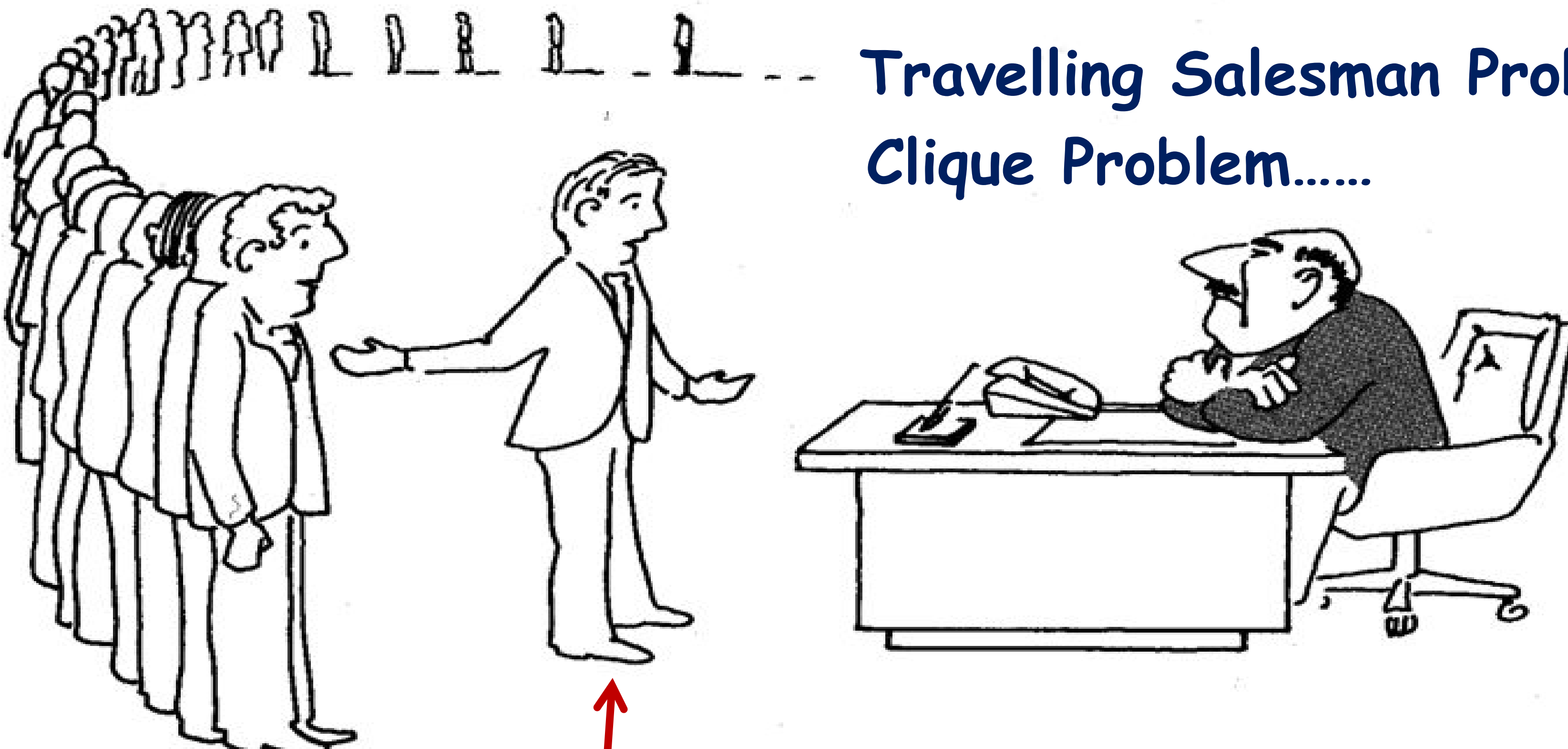
TCS-503: Design and Analysis of Algorithms

Unit V: Selected Topics: NP Completeness

Unit V

- Selected Topics:
  - NP Completeness
  - Approximation Algorithms
  - Randomized Algorithms
  - String Matching

NP-completeness



Travelling Salesman Problem,
Clique Problem……

I can't find an efficient algorithm for,

But neither can these famous people

NP-completeness

Three key concepts for showing a problem to be NP-C are

Decision Problem vs Optimization Problem

Reductions $A \le_p B$

A first NP- Complete Problem

NP-completeness

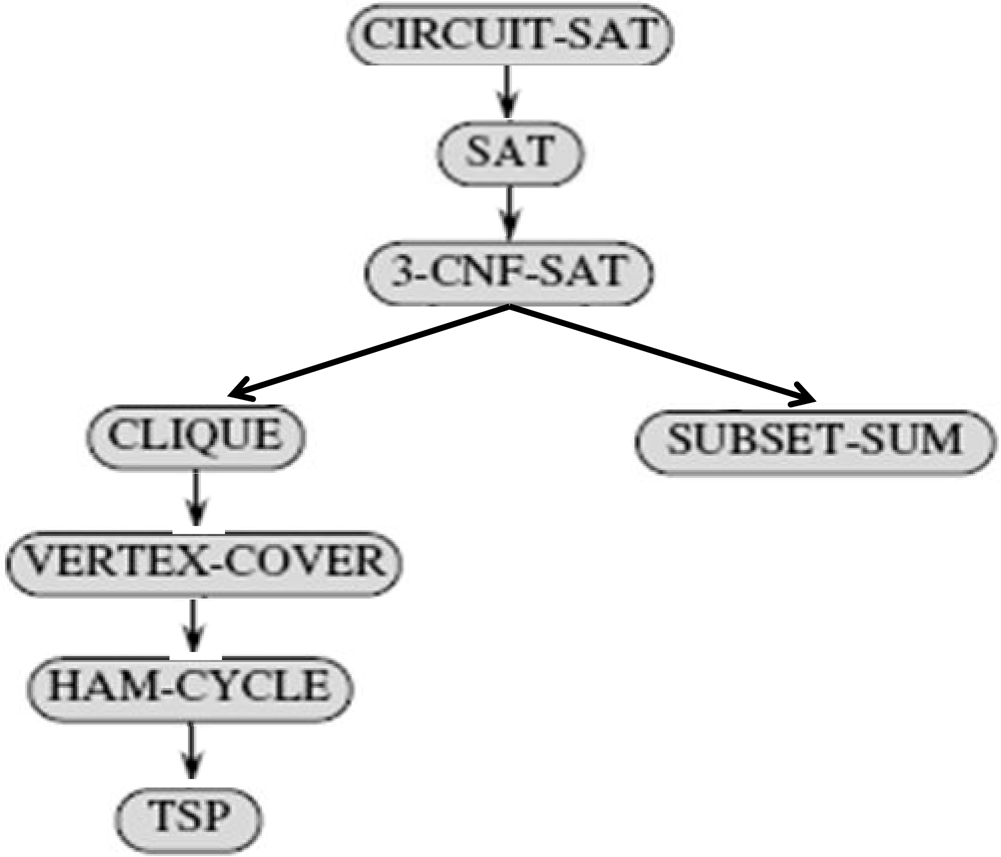
A First NP-Complete Problem

The technique for showing that a problem is in the class NP-Complete requires that we have one NP-Complete problem to begin with.

Cook First time showed that Circuit Satisfiability problem is in class NPC.

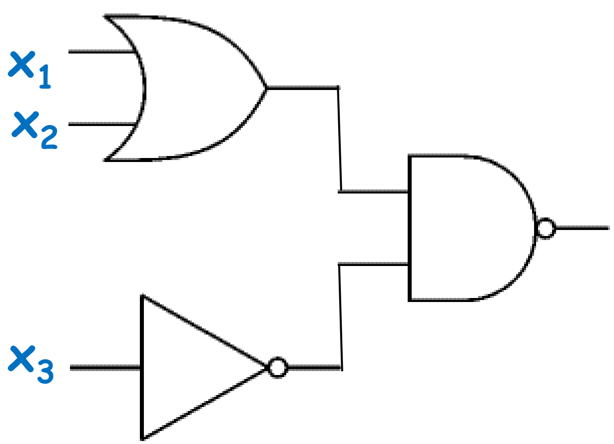
All proofs ultimately follow by reduction from NP-Completeness of Circuit-Satisfiability.

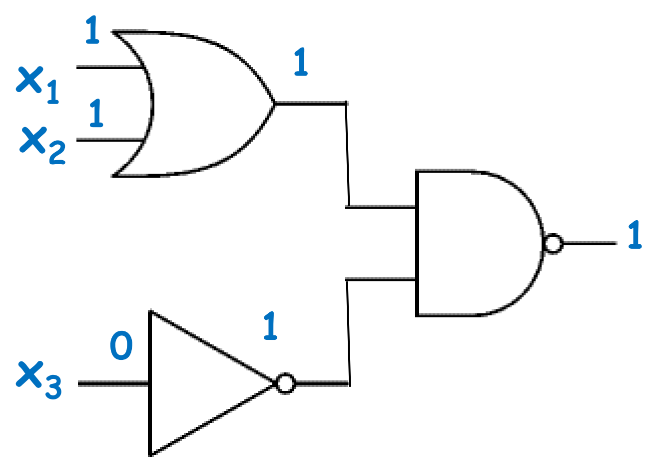
The Structure of NP-completeness



What is Circuit-Sat Problem?

We are given a Boolean combinational circuit composed of AND, OR, and NOT gates and we wish to know whether there is any set of Boolean inputs to this circuit that causes its output to be 1.





The assignment $x_1=1$, $x_2=1$ and $x_3=0$ to the inputs of this circuit causes the output of the circuit to be 1.

What is Formula Satisfiability(SAT) Problem?

A Boolean Formula Φ given, find an assignment of values (0,1) to the variables of Φ that causes it to evaluate to 1.

For example:

$$\Phi = (x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_2)$$

Let Certificate be :

$x_1 = 1$, $x_2 = 0$,

(a. k. a. assignment)

$$\Phi = 1 \land 1 \land 1 = 1.$$

Formula Satisfiability is first to be proven NP-Complete.

Prove that SAT is NPC

To prove that SAT is NPC:

1. First we prove that SAT ∈ NP and then
2. SAT is NP-hard by CIRCUIT-SAT $\leq_p$ SAT

To show that SAT belongs to NP, we show that a certificate consisting of a satisfying assignment for an input formula φ *can be verified in polynomial time.*

Prove that SAT is NPC

(1) SAT $\in$ NP

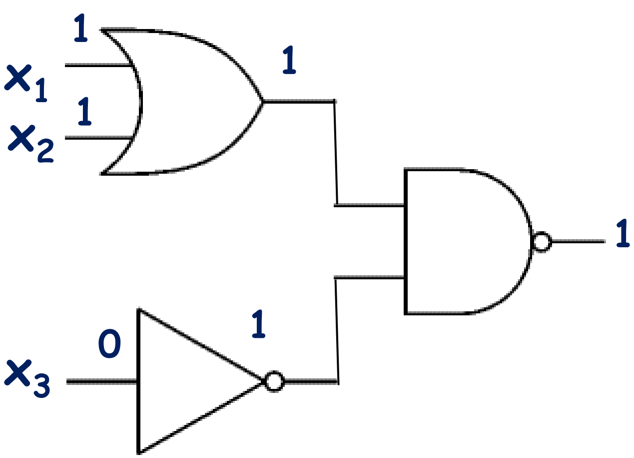
Let the formula be

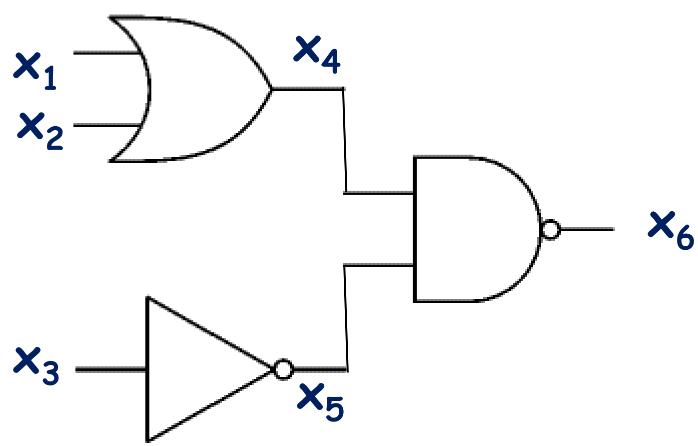
$\Phi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$ and **Certificate:** $x_1 = 1,\ x_2 = 0$

Then $\Phi = 1 \wedge 1 \wedge 1 = 1$

This can be easily verified that Φ *is satisfiable in* polynomial time O(n), where n is the number of clauses (steps) in the formula.

(2) SAT is NPH ,i.e. CIRCUIT-SAT $\leq_p$ SAT





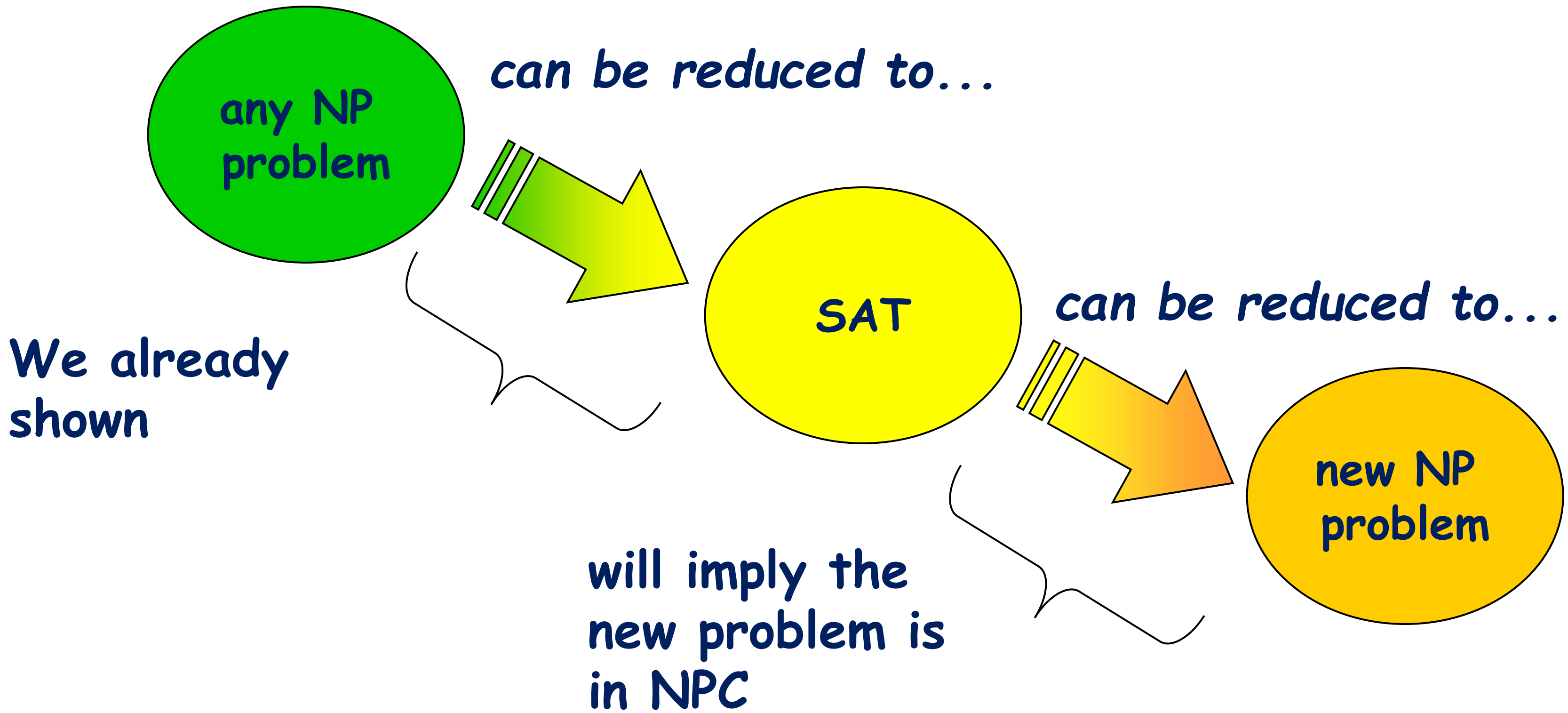
$$\Phi = 1 \wedge (1 \leftrightarrow 1 \vee 1) \wedge (1 \leftrightarrow 1) \wedge (1 \leftrightarrow (1 \wedge 1)$$
$$= 1 \wedge 1 \wedge 1 \wedge 1$$
$$= 1$$

$$\Phi = x_6 \wedge (x_4 \leftrightarrow x_1 \vee x_2) \wedge (x_5 \leftrightarrow \neg x_3) \wedge x_6 \leftrightarrow (x_4 \wedge x_5)$$

Given the Circuit, the formula Φ can be produced in Polynomial Time O(n+ m), where n is number of wires (6) and m is number of gates(3).

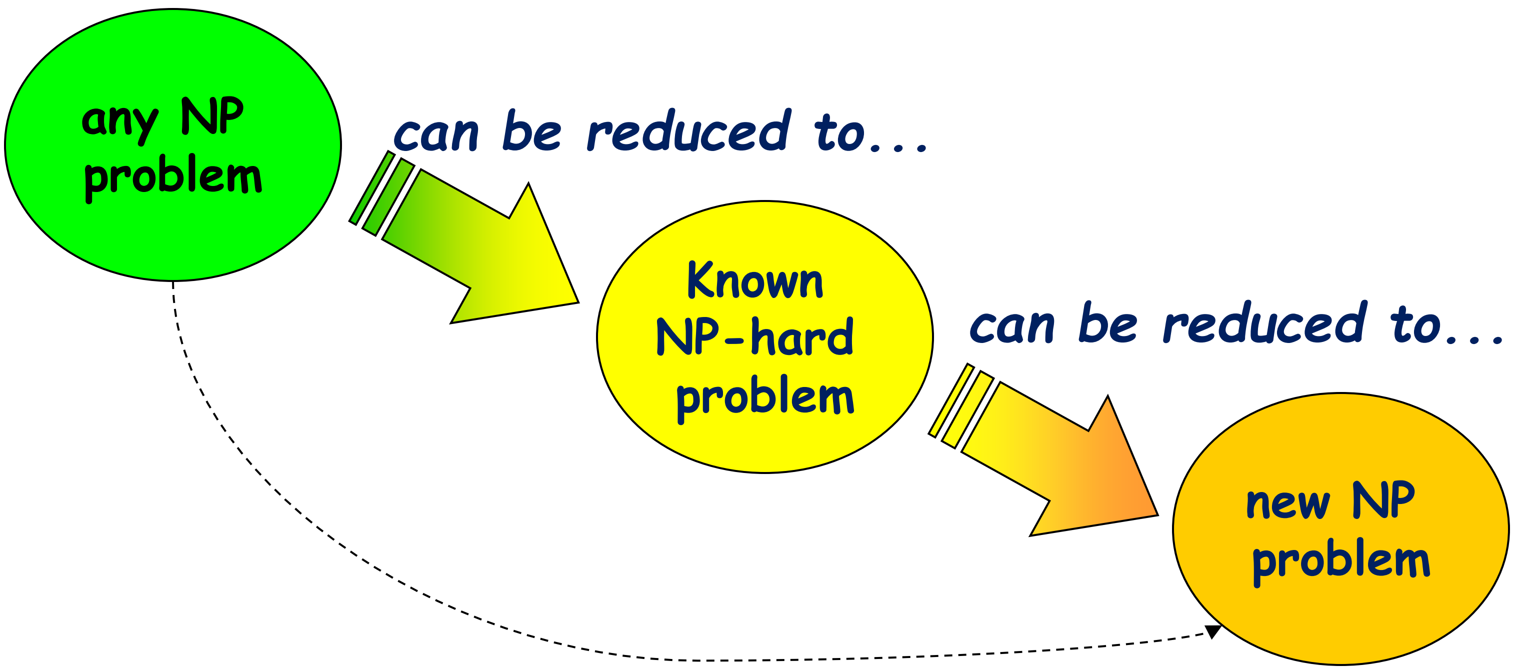
Looking Forward

From now on, in order to show some NP problem is NP-C , we merely need to reduce SAT to it.



... and Beyond!

Moreover, every NP-problem we discover, provides us with a new way for showing problems to be NP-C.



Prove that 3-CNF-SAT is NPC

(1) 3-CNF-SAT $\in$ NP

Let $\Phi=(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$ and

Certificate: $x_1=1, x_2=1, x_3=0, x_4=1$

Then $\Phi=(1 \vee 0 \vee 0) \wedge (0 \vee 1 \vee 1) \wedge (0 \vee 1 \vee 0)$

$=1 \wedge 1 \wedge 1=1$

Thus, the formula Φ is satisfiable.

This can be easily verified that Φ is satisfiable in polynomial time O(n), where n is the number of clauses (steps) in the formula.

Thus, 3-CNF-SAT $\in$ NP.

Prove that 3-CNF-SAT is NPC

(2) 3-CNF-SAT is NPH, i.e., SAT $\leq_p$ 3-CNF-SAT.

The reduction algorithm consists of three steps:

2(a). The first step converts given formula into CNF.

2(b) The second step transforms the formula so that each clause has exactly three distinct literals., that is in 3-CNF.

2(c) We must also show that the reduction can be computed in polynomial time.

Prove that 3-CNF-SAT is NPC

(2) 3-CNF-SAT is NPH, i.e., SAT $\leq_p$ 3-CNF-SAT.

2(a). The first step converts given formula into CNF.

Let $\Phi = x_1 \leftrightarrow (x_2 \wedge x_3)$

$= (x_1 \rightarrow (x_2 \wedge x_3)) \wedge ((x_2 \wedge x_3) \rightarrow x_1)$

Since, $[P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$

$= (\neg x_1 \vee (x_2 \wedge x_3)) \wedge (\neg(x_2 \wedge x_3) \vee x_1)$

$[$Since, $P \rightarrow Q \equiv \neg P \vee Q]$

$\Phi_1 = (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_1)$

[Distributive law $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$ and De Morgan's Laws $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$]

which is in CNF.

Prove that 3-CNF-SAT is NPC

(2) 3-CNF-SAT is NPH, i.e., SAT $\leq_p$ 3-CNF-SAT.

2(b) The second step transforms the formula so that each clause has exactly three distinct literals., that is in 3-CNF.

$\Phi_1 = (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_1)$

$= \quad C_1 \quad \wedge \quad C_2 \quad \wedge \quad C_3$

Since, C_1 and C_2 have 2 distinct literals,

$\Phi_2 = (\neg x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee x_2 \vee \neg y_1) \wedge$

Since, if $y_1=0$, $\neg y_1=1$ or if $y_1=1$, $\neg y_1=0$

$(\neg x_1 \vee x_3 \vee y_2) \wedge (\neg x_1 \vee x_3 \vee \neg y_2)$

Since, if $y_2=0$, $\neg y_2=1$ or if $y_2=1$, $\neg y_2=0$

$\wedge (\neg x_2 \vee \neg x_3 \vee x_1)$

Prove that 3-CNF-SAT is NPC

(2) 3-CNF-SAT is NPH, i.e., SAT $\leq_p$ 3-CNF-SAT.

2(c) We must also show that the reduction can be computed in polynomial time.

- ✓ The Construction of Φ_1 from Φ has introduced 1 more clause.
  [Φ Contains 2 clauses, Φ_1 contains 3 clauses.]
- ✓ The construction of Φ_2 from Φ_1 has introduced 2 more clauses.
  [Φ_1 Contains 3 clauses, Φ_2 contains 5 clauses.]

Thus, the size of the resulting formula Φ_2 is polynomial in the length of the original formula Φ.

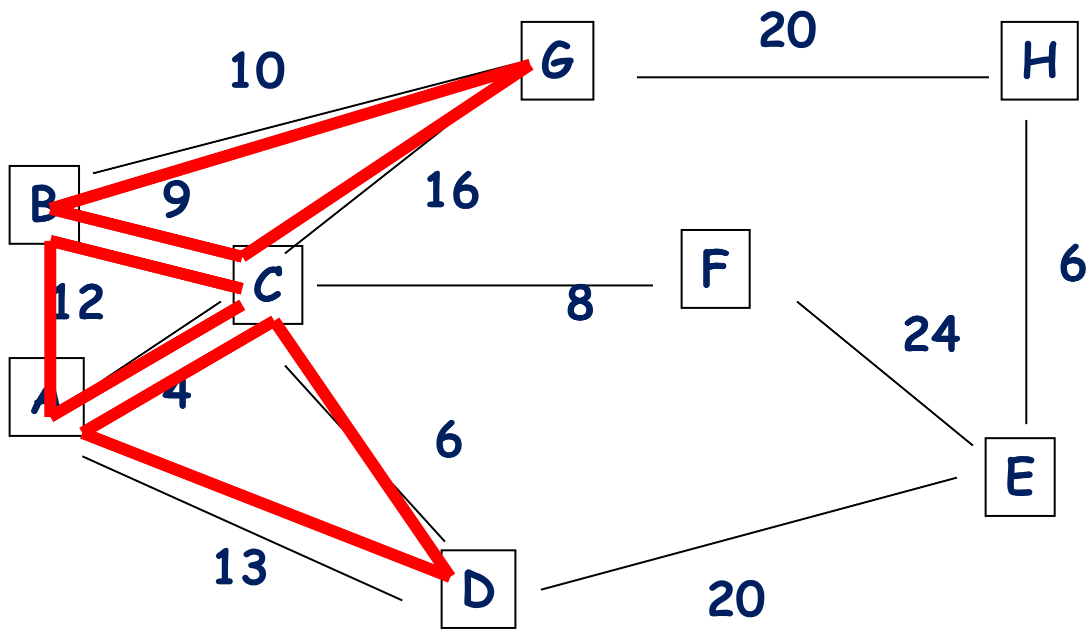
Φ has two clauses(n=2). Φ_2 has 5 clauses (n^2 +1).

Learn DAA : From B K Sharma

What is Complete Sub-graph?

Complete Sub-graph: a subset of vertices of a graph fully connected to each other.

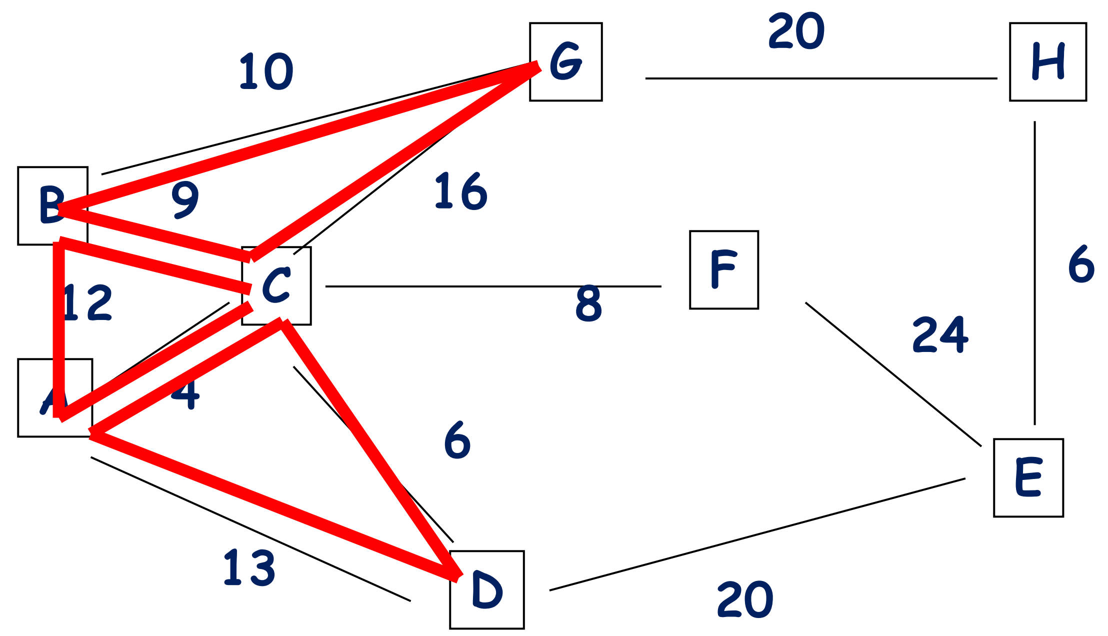
One where there is an edge between every two vertices.



Learn DAA : From B K Sharma

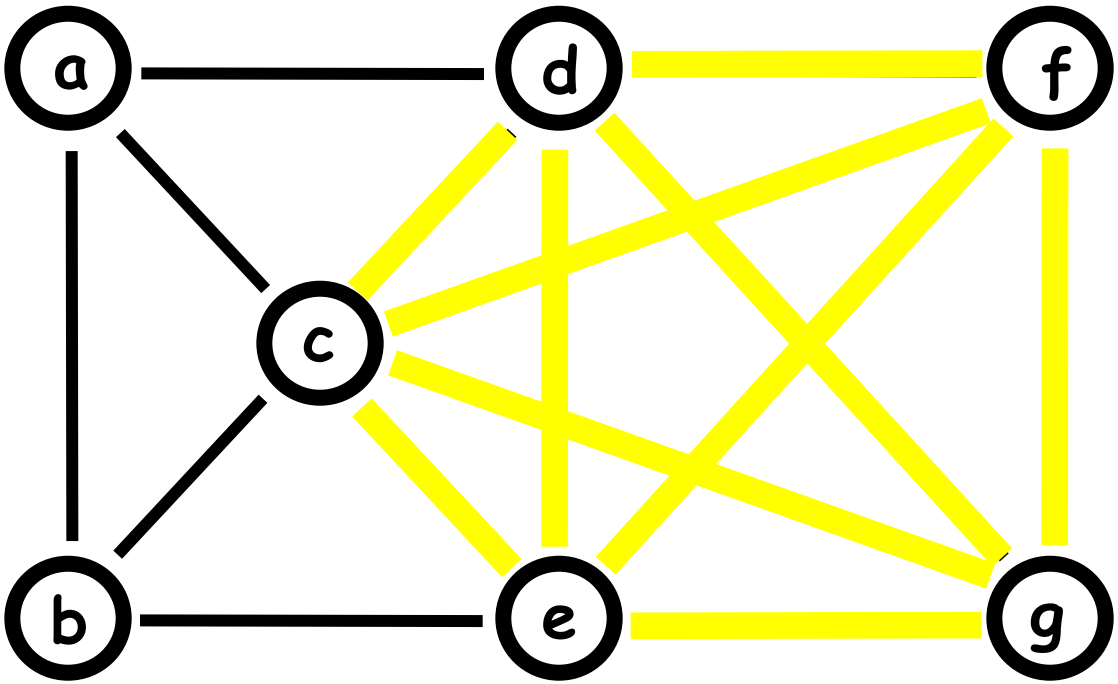
What is Clique?

Clique is a complete sub-graph of a graph.



Learn DAA : From B K Sharma

What is Clique?

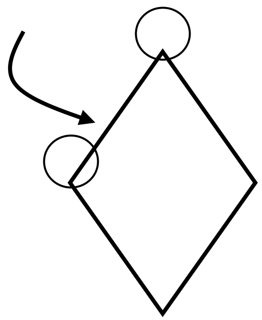


k-clique = complete subgraph of k nodes

What is Clique Problem?

Find the clique with the largest number of vertices in a graph.

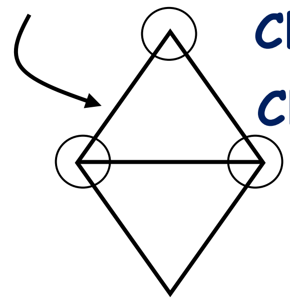
This graph has Clique of size 2



Clique(G, 2) = YES

Clique(G, 3) = NO

This graph has Clique of size 3



Clique(G, 3) = YES

Clique(G, 4) = NO

Optimization problem

Find a clique of maximum size.

Restated as Decision problem

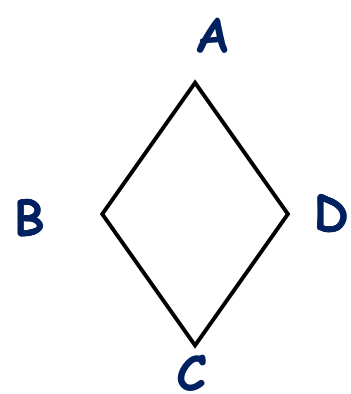
Does Graph have a clique of size k?

Prove that Clique Problem is NPC?

1) Clique $\in$ NP

1(a) To show that CLIQUE $\in$ NP, for a given graph $G = (V, E)$, we use the set $V' \subseteq V$ of vertices in the clique as a certificate for G.

$V=\{A, B, C, D\}$



Certificate: $V'=\{A,B\}$

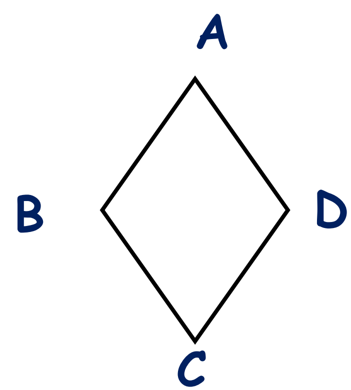
1(b) *Checking whether V' is a clique can be accomplished in polynomial time by checking whether, for each pair $u, v \in V'$, the edge (u, v) belongs to E.*

Prove that Clique Problem is NPC?

1) Clique $\in$ NP

1(b) *Checking whether* V' is a clique can be accomplished in polynomial time by checking whether, for each pair $u, v \in V'$, *the edge* (u, v) *belongs to* E.

E={ AB,BC, CD, AD}



Certificate: V'={A,B}

Check, is there an edge(A, B) and B,A? Answer is Yes.

Similarly we can check V'={A,C}, {AD}, {BC}, {C,D} etc.

Prove that Clique Problem is NPC?

1) Clique $\in$ NP

The length of the certificate: $O(n)$ ($n=|V|$)

Time complexity: $O(n^2)$

Prove that Clique Problem is NPC?

1) Clique $\in$ NP

Given a subset V' of V

Verify $|V'|=k$

$O(k)$ time.

Verify every pair of vertices in $|V'|$ have an edge in E.

$O(k^2 |E|)$ time.

Prove that Clique Problem is NPC?

2) Clique is NPH

3-CNF-SAT $\leq_P$ CLIQUE

Let $\Phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ be a 3-CNF formula with k clauses, each of which has 3 distinct literals.

For example,

$$\Phi = (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$

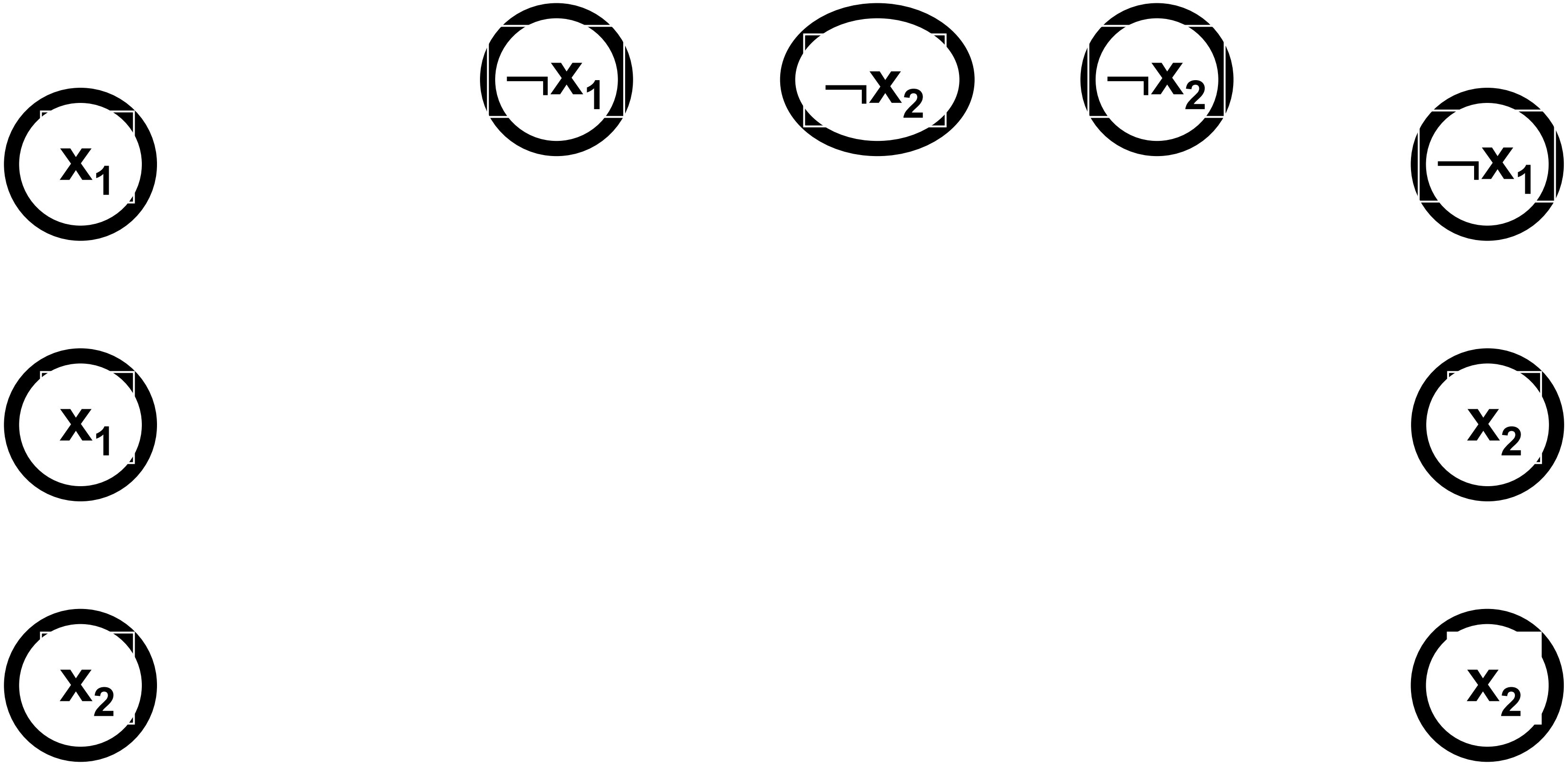
$$= \quad C_1 \quad \wedge \quad C_2 \quad \wedge \quad C_3$$

For each clause put a triple of vertices in the graph, one for each literal.

Prove that Clique Problem is NPC?

2) Clique is NPH

3-CNF-SAT $\leq_P$ CLIQUE

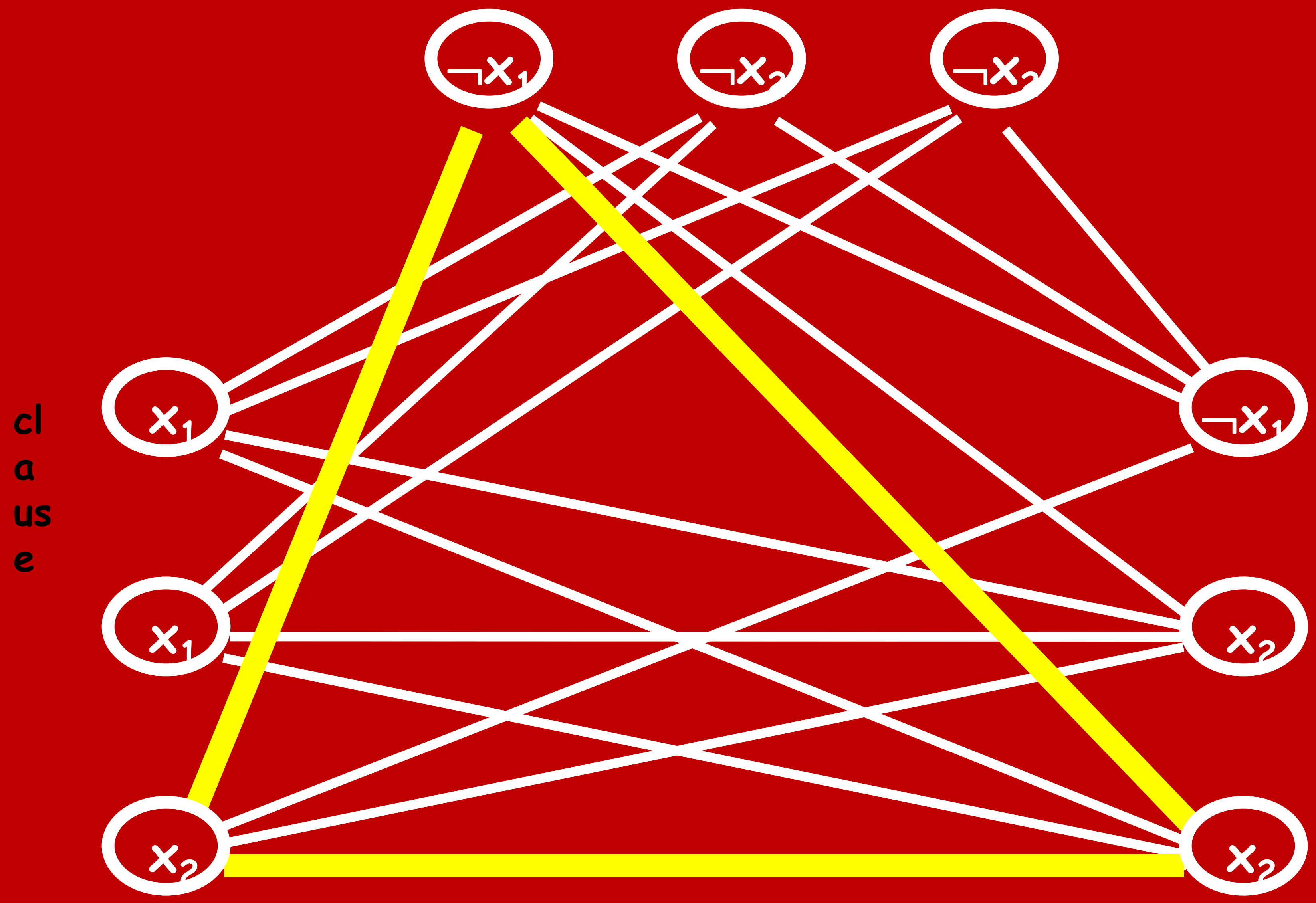


Prove that Clique Problem is NPC?

2) Clique is NPH

3-CNF-SAT $\leq_P$ CLIQUE

Put an edge between two vertices if they are in different triples and their literals are *consistent*, meaning not each other's negation.



#nodes = 3(# clauses) k = #clauses

Prove that Clique Problem is NPC?

2) Clique is NPH

$$\Phi = (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$

$$=(0 \vee 0 \vee 1) \wedge (1 \vee 0 \vee 0) \wedge (1 \vee 1 \vee 1)$$

$$=1 \wedge 1 \wedge 1 = 1$$

Now Pose Question:

Question : Is φ satisfiable? **Answer**: Yes: $\neg x_1 = 1$ (true) and $x_2 = 1$(true)

assign the value **true** to every variable occurring in the clique)

Prove that Clique Problem is NPC?

2) Clique is NPH

3-CNF-SAT $\leq_P$ CLIQUE

The 3-cnf formula is satisfiable iff the graph has a k-clique.

The 3-CNF formula ϕ has m=3-clauses and is satisfiable then the Clique must have the size k=3.

Prove that Clique Problem is NPC?

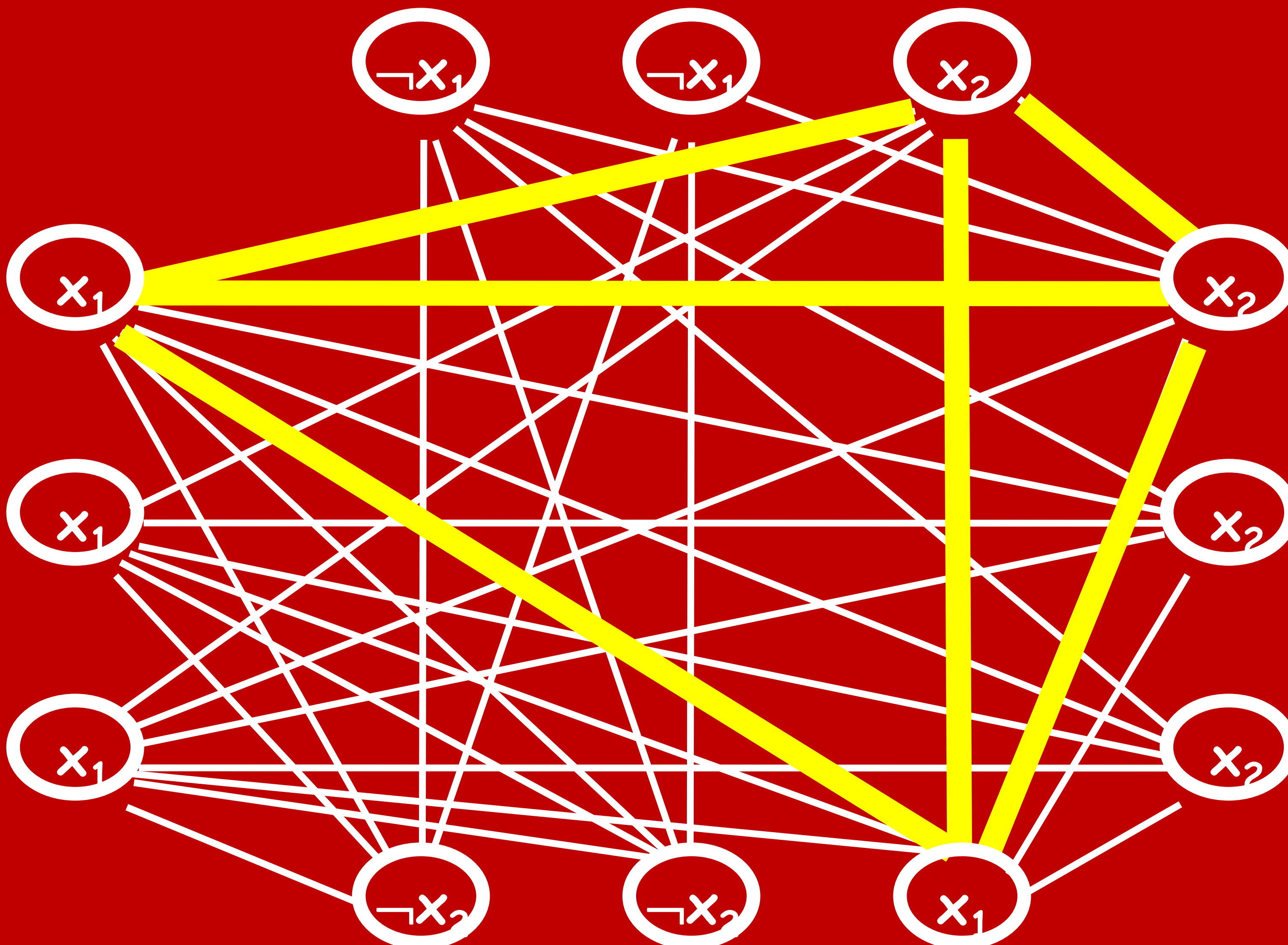
2) Clique is NPH

What's left?

Demonstrate the reduction is polynomial

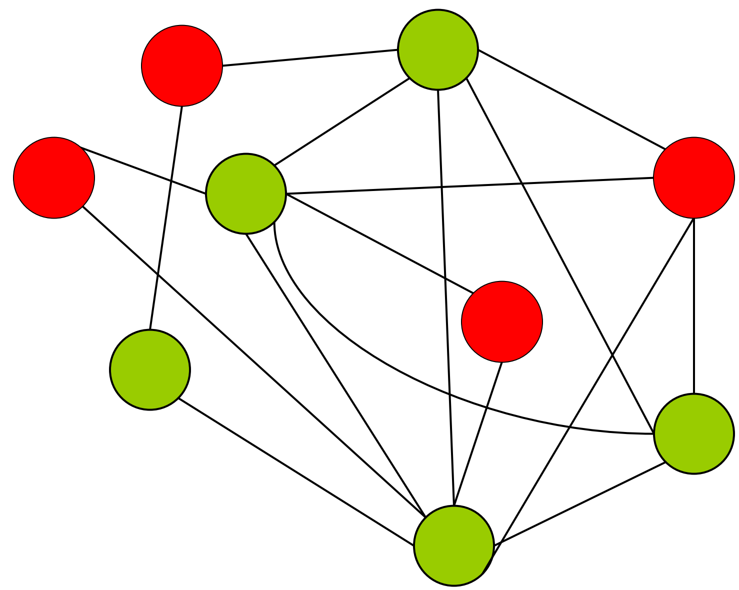
$|E| \leq (\#\text{ of literals in } \varphi)^2 = O(n^2)$

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



Independent Set

Given a graph G(V,E), an independent set V' is a collection of nodes in G such that no two nodes in V' have an edge connecting them.

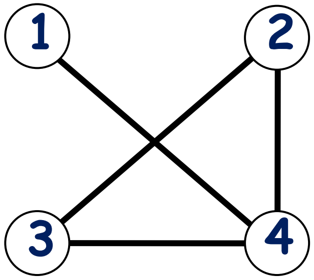


Independent Set

An independent set is a set of nodes with no edges between them.

independent sets

∅, {1}, {2}, {3}, {4}, {1, 2}, {1, 3}



Complement (G^c) of a graph (G)

The complement G^C of a graph G contains exactly those edges not in G.

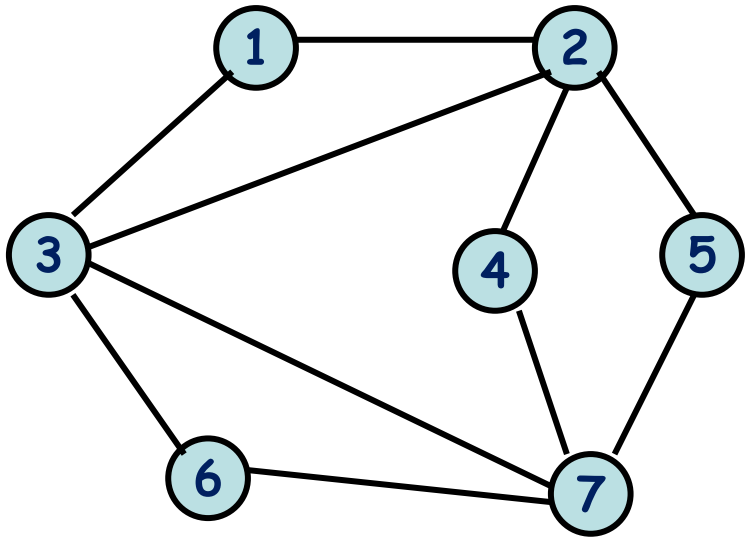
$$G = (V, E) \Rightarrow G^C = (V, E^C)$$

$$E^C = \{(u, v):, u, v \in V, \text{ and } (u, v) \notin E\}$$

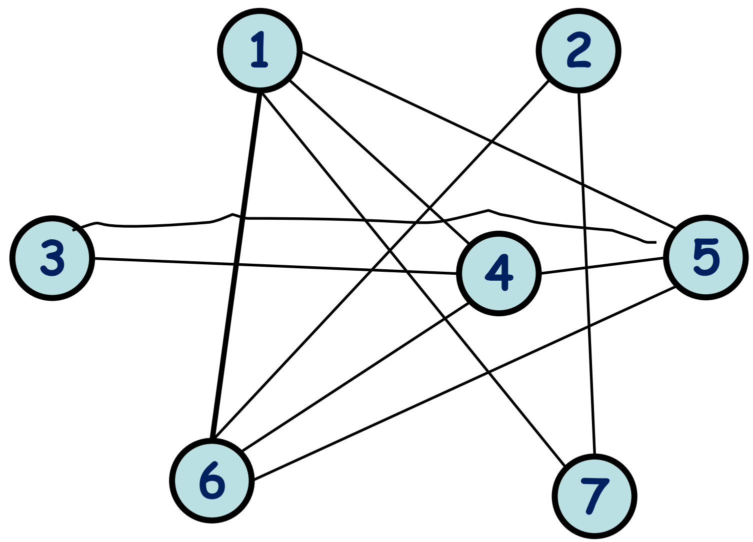
Complement (G^c) of a graph (G)

The complement G^C of a graph G contains exactly those edges not in G.

G



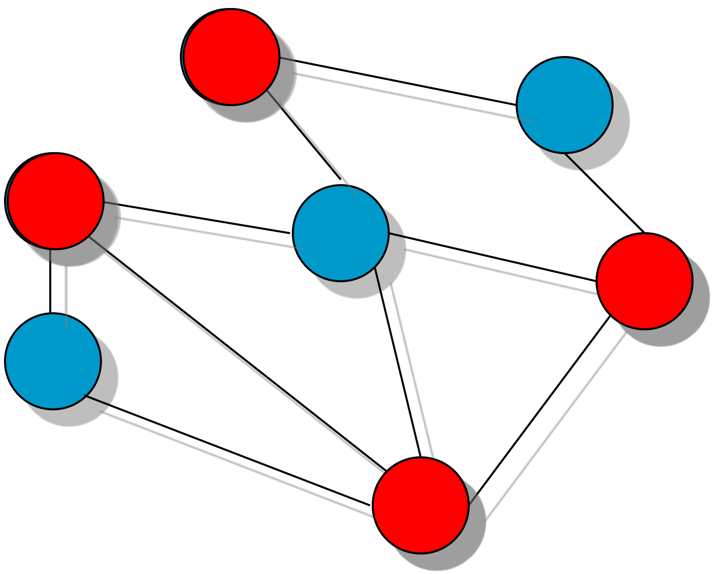
G^c

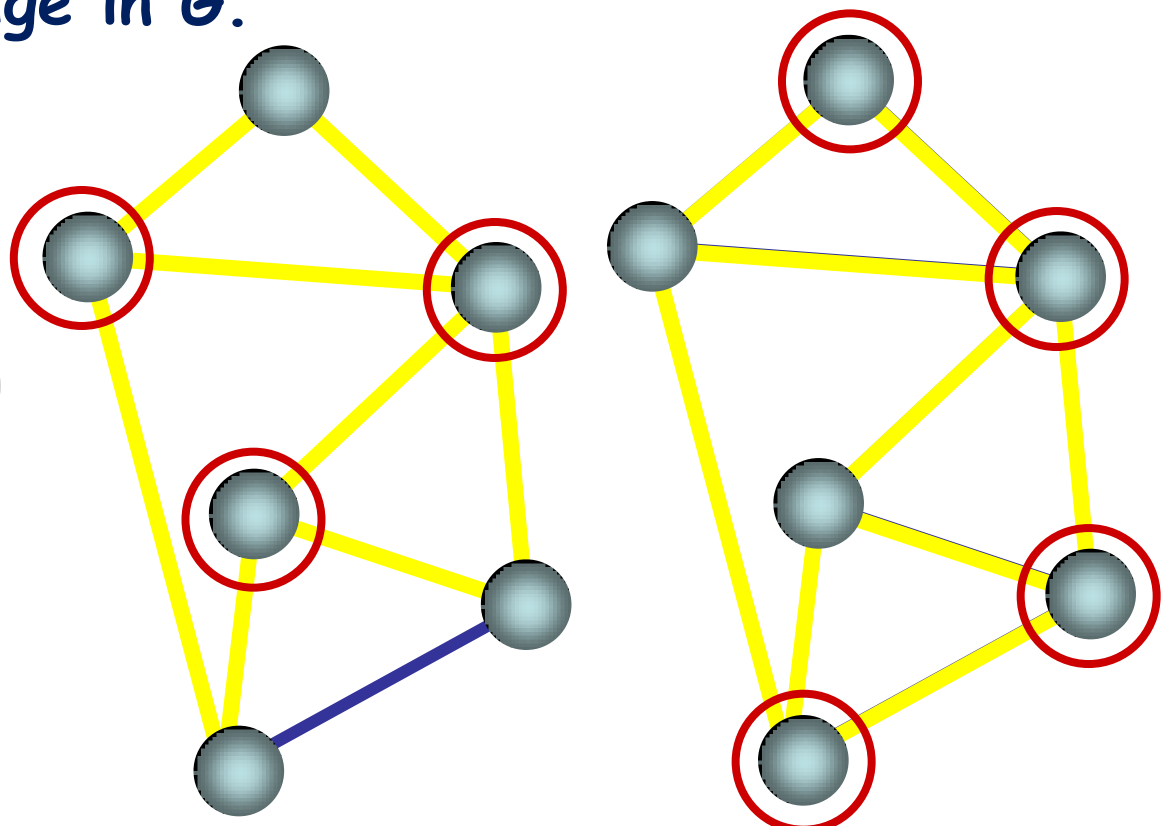


Learn DAA : From B K Sharma

Vertex Cover

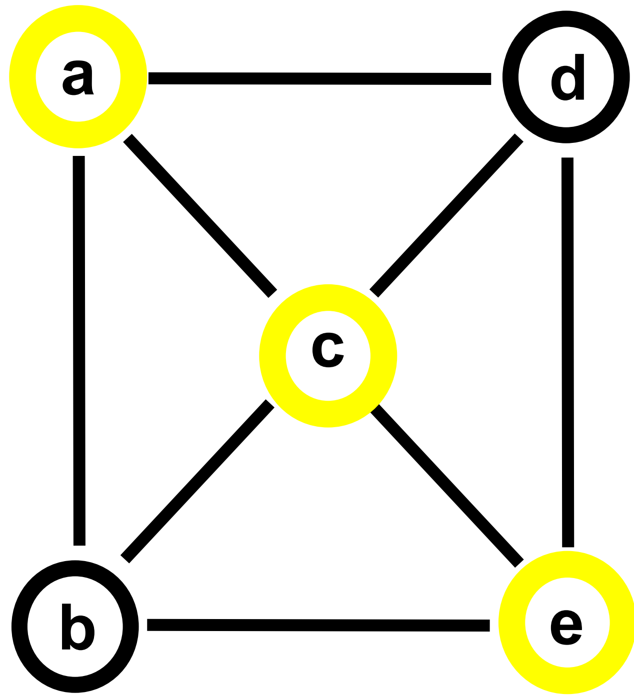
A *vertex cover* for a graph G is a subset of vertices incident to every edge in G.





Size of a vertex cover: the number of vertices in it.

Vertex Cover



vertex cover = set of vertices that cover all edges

Vertex Cover Problem

Optimization Problem

What is the minimum size vertex cover in *G*?

Restated as a decision problem:

Does a vertex cover of size *k* exist in *G*?

Prove that Vertex Cover Problem is NPC

CLIQUE $\leq_P$ VERTEX-COVER

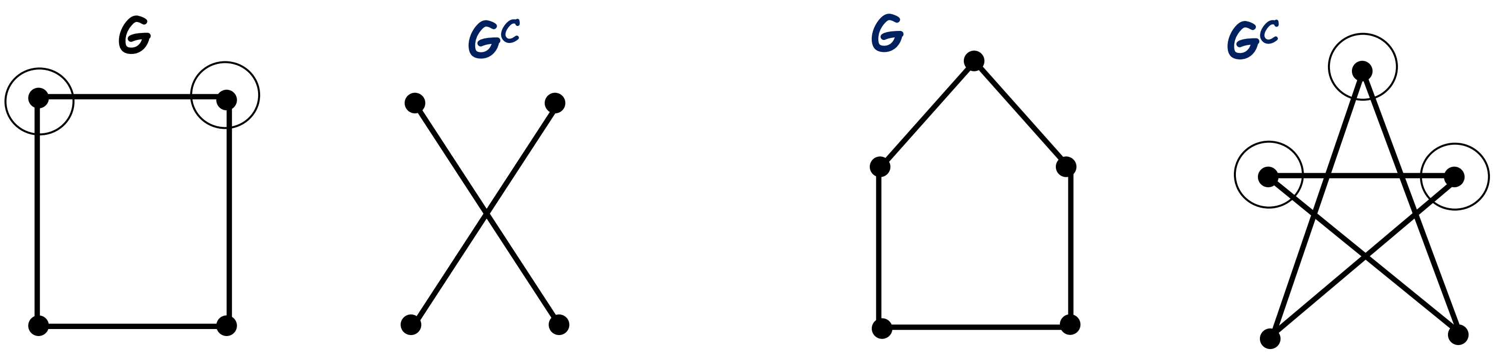
Reduce *k*-clique to vertex cover

Prove that Vertex Cover Problem is NPC

Compute G_C in polynomial time

G has a clique of size k iff G^C has a vertex cover of size $|V|$ - k

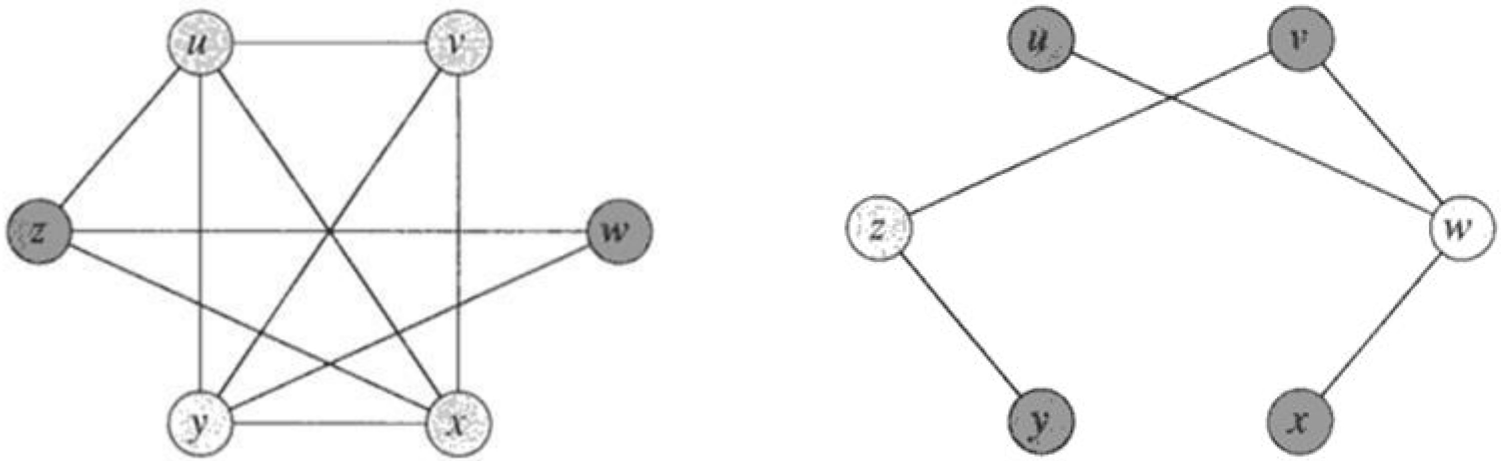
$\langle G, k\rangle$ (clique) $\rightarrow$ $\langle G^C, |V|-k\rangle$ (vertex cover)



Clique = 2   V-C = 2   Clique = 2   V-C = 3

G has a **clique** of size k $\Leftrightarrow$ G^C has a **vertex cover** of size n - k

Prove that Vertex Cover Problem is NPC



(a)

(b)

A graph G(V,E) with CLIQUE V'={u,v,x,y}

The graph G^C produced by the reduction algorithm that has vertex cover V-V'={w,z}